HAL
open science

Enforcing privacy as access control in a pervasive context

Aurélien Faravelon, Stéphanie Chollet, Christine Verdier, Agnès Front

To cite this version:

Aurélien Faravelon, Stéphanie Chollet, Christine Verdier, Agnès Front. Enforcing privacy as access control in a pervasive context. CCNC 2012 - Consumer Communications and Networking Conference, Jan 2012, Las Vegas, NV, United States. pp.380-384, 10.1109/CCNC.2012.6181011 . hal-00745705

HAL Id: hal-00745705

https://hal.science/hal-00745705v1

Submitted on 26 Oct 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Enforcing Privacy as Access Control in a Pervasive Context

Aurélien Faravelon*†, Stéphanie Chollet*, Christine Verdier*, Agnès Front*
*Laboratoire d'Informatique de Grenoble
BP 53, 38041 Grenoble cedex 9
†Groupe de recherche Philosophie, Langage & Cognition
Bât. ARSH 2, 38040 Grenoble cedex 9
Email: {aurelien.faravelon,stephanie.chollet,christine.verdier,agnes.front}@imag.fr

***Abstract*—Pervasive applications promote a seamless integration of computer artifacts with our daily and business lives. However, they threaten privacy in two ways. Firstly, adaptation to a user's context necessitates a large collection of data. Secondly, context should be addressed when granting users access to information. This paper handles privacy management as an access control problem and argues that privacy should be specified from a global point of view. Investigating privacy specification at a high level of abstraction and its implementation leads to the proposition of a generative approach relying on model-driven engineering. This approach distinguishes a design level for privacy from its execution level. The design level provides a specification language for privacy which emphasizes its contextual features. It is implemented at the execution level as a service composition generated through model transformations. This composition gathers heterogenous entities, such as pieces of software code or devices. The approach is validated on the example of a medical workflow.**

***Keywords*-Service-Oriented Computing (SOC), Security, Access Control, Model-Driven Engineering.**

I. Introduction

Supported by the development of portable devices, like smartphones and sensors, pervasive applications are booming. They promise to enable users to provide users with added value services anywhere using the available resources. However, pervasive applications raises security issues, especially in terms of privacy when they involve sensitive data. In medicine for instance, pervasive applications may enable medical staff to access the relevant data from anywhere and at anytime thanks to the use of tablets and smart devices. However, two points must be considered. Firstly, medical staff's privacy must be safeguarded by minimizing the set of information necessary to compute the context they are in. Then, medical staff must be allowed to access a patient's data if their situation allows them to do so. Medical staff on holiday, for instance, does not satisfy this condition.

Consequently, constraining what a user can do and the data an application can access is necessary. This constraining activity is called access control [1] which thus appears as a way to protect privacy complementary with other security mechanisms. In the pervasive realm, access control management must be dynamic because devices, such as smartphones, may appear and disappear, and entities' features, such as users' location, often change. Therefore, context, *i.e.* the pieces of information which are of interest because they influence a set of rights' assignment of revocations, turns out to be primordial in pervasive access control.

Implementing such an access control promises to be of a high technical complexity and we claim that access control should be addressed right from the design of an application in order to be effectively enforced. Indeed, the design level offers a global point of view on the application which is crucial as access control rights must be managed according to the overall application's environment to be fully contextual. Moreover, the design phase permits to gather the actors involved in an application conception as security and functionalities are usually addressed by different people. Once this phase is performed, a mechanism to implement access control can then be provided.

This paper takes up the challenge of providing a generative approach to access control. The contribution relies on model-driven engineering to clearly separate the design level from the execution one and generate the later from the first through model transformations. At the design level, we provide an access control model and its logical grammar to specify privacy and separate access control preoccupations from functional ones. At the execution level, access control is enforced in a service composition to address the heterogeneity of pervasive applications' entities. Indeed, Service-Oriented Computing (SOC) [2] promotes the production of applications through the composition of already existing functionalities called Services that may be implemented as software code for Web Services or as devices for UPnP[1] and DPWS[2] Services. At runtime, the key element is the context registry that maintains an overall view on the application's state and feeds privacy requirements checking.

Model-driven security has already been applied to Component-Based Software Engineering [3] and transfer security in service composition [4]. However, its use for pervasive access control in service composition is still pendant. At design level, the use of an access control model expressive enough is yet to come. At the execution level,

[1] http://www.upnp.org/
[2] http://schemas.xmlsoap.org/ws/2006/02/devprof/

pervasive access control cannot be expressed with XACML[3]. This specification is only adapted to access control specification for Web Services, not to UPnP or DPWS Services for instance and not suitable for dynamic access control as it does not offer temporal mechanisms.

Services selection according to their access control features [5], [6] is not satisfying either. It focuses on access control at the level of each service but do not take into account the application's environment.

Filters to enforce access control to a resource, described for instance in [7], solve this problem and are often implemented as middlewares in the realm of SOC. Hashem [8] integrates heterogenous components in an e-government workflow with a middleware in charge of enforcing access control. Adage [9], another middleware, was integrated to CORBA and can deal with separation of duty but cannot address contextual constraints. It is thus not adapted to pervasive applications. Bhatti [10] enforces temporal constraints in the sense of GTRBAC but leaves aside the rest of contextual constraints. Sohr [11] provides a specification language for separation of duty and contextual constraints. However, the authors rely on the Object Constraint Language (OCL) that do not possess temporal operators. Stating the chain of permissions in a workflow to express, for instance, that a physician must update a medical file after interacting with a patient, is thus impossible.

Our contribution aims at filling these gaps and is presented in the rest of this paper. In Section 2, we introduce a running example. Section 3 outlines the general idea of our model-driven approach. Section 4 details the access control meta-model used at design time. Section 5 explains the architecture and the implementation of our approach to secure services. We eventually conclude in Section 6.

II. Running Example

Our approach, though generic, is applied to the example of medical emergency management, displayed on Figure 1. This process summarizes several issues faced in pervasive access control enforcement.

Let's consider two actors, Bob and Alice, who respectively embody the roles of physician and nurse and work in the same hospital. As an emergency arises, Bob, who is an available physician must be notified. Bob is considered available because he is on duty, in the hospital and not already engaged in an emergency. As Bob may be anywhere in the hospital, this notification must be transmitted to his cell phone. When Bob acknowledges the reception of the notification, he must be granted the access to all the information needed to take care of the patient and primarily to their digital file. Depending on his access point to this file, for instance his cell phone or his laptop, the information must be adapted to be easy to display. Bob must also be authorized to modify the patient's file in order to monitor the cares he provides and save the prescriptions he makes.

If the emergency requires a nurse to take care of the patient after the physician is done, Alice, who is available because she is on duty, in the hospital and has not exceeded the number of patients she can take care of, is notified and granted access to the information necessary to perform her task. Among this information lies a subset of the patient's digital file encompassing prescriptions.

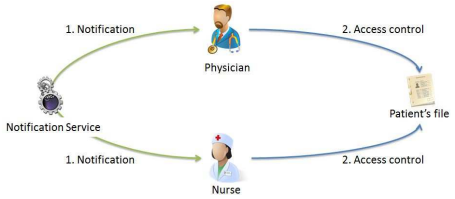

Figure 1. Global Approach.

This example has three prominent features:

- **Access control requirements** guarantee the patient's **privacy** and the efficiency of **risk management**.
- **Notifications are pushed** to the actors according to their access point. Accordingly, the system is **proactive** and **efficient** and **smoothly integrated** to the actors' environment.
- Each **actor's context** is taken into account into access control management in order to enforce the **least privilege principle**. This context awareness allows a **dynamic management of rights** which are thus granted and denied only when necessary.

As a consequence, this example demands a high degree of automation in order to efficiently handle risk and safeguard privacy, this is what we undertake in the rest of this paper.

III. Global Approach: Privacy by Design

Privacy by design is doomed to failure if no mechanism is provided to actually enforce it. We take up this challenge by providing a model driven approach to privacy, understood as an access control problem, elicitation and enforcement. This approach is divided into two parts: the expression of privacy by design and the privacy at runtime, both illustrated by Figure 2.

The design level hides the complexity of the privacy code for services and permits to have a global view of privacy while separating each preoccupation. Indeed, it provides meta-models and models for the service composition annotated by security concepts for privacy derived from a privacy meta-model which rests on a logical grammar.

The execution code, generated from the models, is composed of two parts: the service composition one and the the privacy one. This code is executed on an execution platform.

The service composition code can be seen as an orchestration to invoke the concrete services implemented as, for

[3]http://www.oasis-open.org/committees/xacml/

instance, Web Service, UPnP or DPWS services. The privacy code is added to the composition code without altering the later: the separation of concerns is thus respected. The privacy code contains the management of the access control (defined in the access control model) and the checking of the logical constraints. We introduce a context registry in order to feed the privacy management. The context registry provides a global view of the environment. It catches all the events of the environment. Thus, the privacy is globally managed for all the applications. It avoids conflicts between the different actors and applications. The context registry is the keystone element of our architecture.

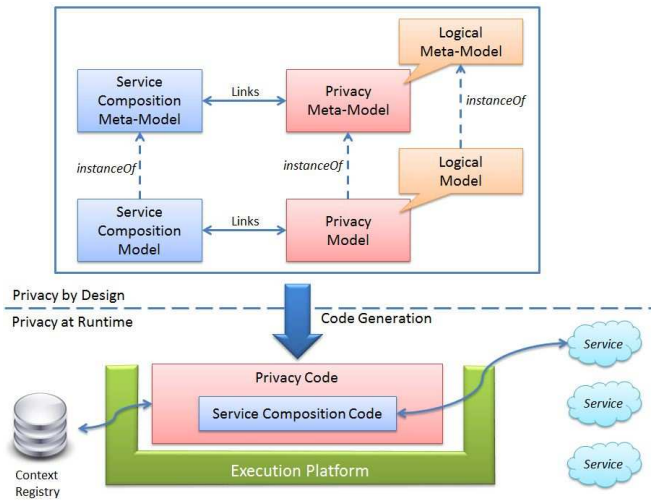

Figure 2. Global Approach.

After introducing the salient features of our meta-model in Section IV, we detail our execution level in Section V.

IV. Access Control for Privacy Meta-Model

As previously said, privacy can be seen as an access control problem. We provide in this Section a meta-model based on a logical language to guide access control for privacy expression and computation.

A. Access Control for Privacy Meta-model

The meta-model relies on five basic concepts, `Subject`, `Object`, `Task`, `Right` and `Context`. Its structure is presented on Figure 3.

`Subjects` designate people or software agents, such as Bob, a physician. `Objects` refer to the resources of the systems, such as sensors or peripherals. `Tasks` are all the access modes to an `Object` and `Rights` designated the modalities of these access modes, such as obligation, permission or interdiction.

`Roles` express positions in an organisation, such as being a physician, they facilitate access control management as `Permissions` are not attributed to `Users`, who may always change but to `Roles`, *i.e.* organizational categories. A `User` may be part of several `Roles`.

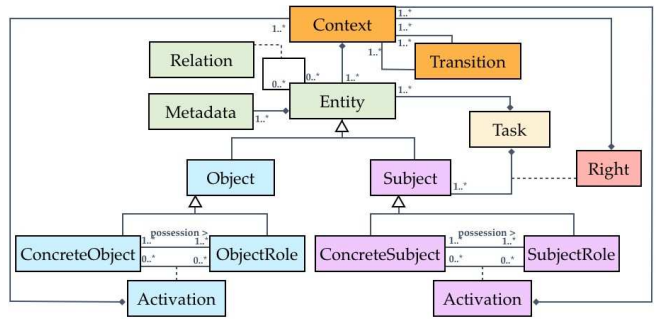

Figure 3. Access control for privacy meta-model's structure

`Context` refers to a situation of interest, *i.e.* influencing a set of rights for a `SubjectRole` over a `ResourceRole`. A `Context` is defined by the values of the attributes and the relationships of the `Entities` of the composition, *i.e.* the `Subjects` and the `Objects`. `Context` affects right management because it conditions their use and `Roles` activation. `Context`'s dynamicity is accommodated by `Transitions` between different `Contexts`.

B. Meta-model's Logical Foundations

The meta-model relies on a logical semantics based on Computational Tree Logic (CTL) [12]. Thus, the meta-model's instances, *i.e.* the access control policies, constitute Kripke structures [13], *i.e.* oriented graphs where each node is a state of the access control policy expressed as a tuple comprising a `Subject Role`, a `Permission`, a `Context` and a set of `Objects Roles`.

As a result, checking access control policies turns out to be a model checking problem. This problem is solved by checking that the current state of the service composition and the chain of the previous states is conform to the obtained Kripke Structure [12].

In the rest of this paper, we posit that access control requirements have already been captured and we focus on the execution of model checking.

V. Architecture and Implementation

The general principle is, for the `Service Provider` to hide the concrete implementations of the Service behind a set of components dedicated to access control enforcement. These components take part in access control policy checking: they verify that the current state of the composition satisfies the access control policy. We present these components and their communication in this Section.

A. Architecture

Thanks to the `Service Provider`, Access Control enforcement is transparent in the frame of a service composition because the service provider exports itself as a Web

Service. It can thus be discovered and bound as any other Service. Figure 4 presents the global architecture.

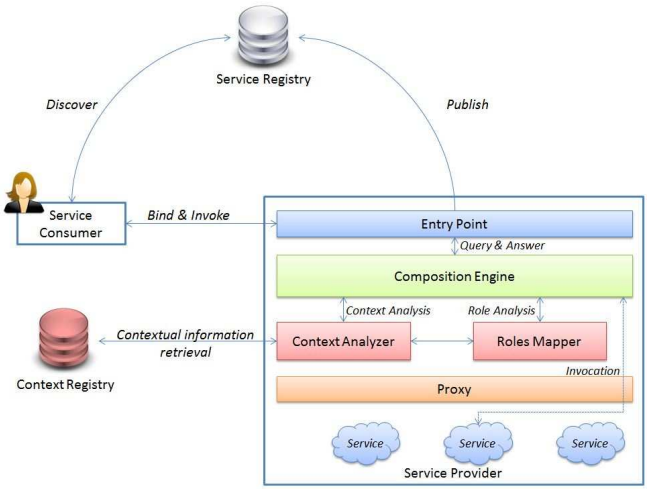

Figure 4. Global architecture.

The `Service Provider` comprises the following elements:

- The `Entry Point` constitutes the interface that the proxy publishes to the service registry. The `Entry Point` also publishes the service's metadata notably related to its security abilities.
- The `Composition Engine` is in charge of orchestrating the access control policy checks and the invocation to the concrete services. It checks that the current state of the composition, *i.e.* the tuple comprising the `Service Consummer`'s roles, the `Objects` that the `Service Consummer` wants to access and the the `Service Consumer's Context` satisfies the policy.
- The `Context Analyzer` is in charge of retrieving the contextual information concerning the `Service Consumer` and the composition's environment of execution. Thus, the `Context Analyzer` deals with static features, such as a person's identity and assigned roles and dynamic features, such as time, location and the arising of events such as an emergency. These features are the basis of context computation.
- The `Roles Mapper` computes the activated roles. To do so, it checks if the `Service Consumer` satisfies all the constraints which define a `Role` and if the necessary `Context` is activated.
- The `Proxy` constitutes a connector to the concrete Services which takes into account their implementation in order to adequately transfer them the query. The addressed implementation features notably encompass the services type (Web Service, UPnP or DPWS), the data format and the translation between the organizational access control policy and the Service's own policy.

B. Service Provider's Functioning

Now that the `Service Provider`'s structure has been defined, we turn to its behavior. As an input, the `Service Provider` takes the `Service Consumer`'s query. As an output, it rejects the query if the tuple comprising the `Service Consumer`, the resources it wants to access and the activated `Context` does not satisfy the access control policy, or delivers the query's result it does.

Figure 5 presents the flow of messages between the Service Provider's components to determine if a query can be processed. This invocation transits through the Proxy and the Composition Engine composes the concrete Services answers before delivering the query's answer to the Service Consumer.

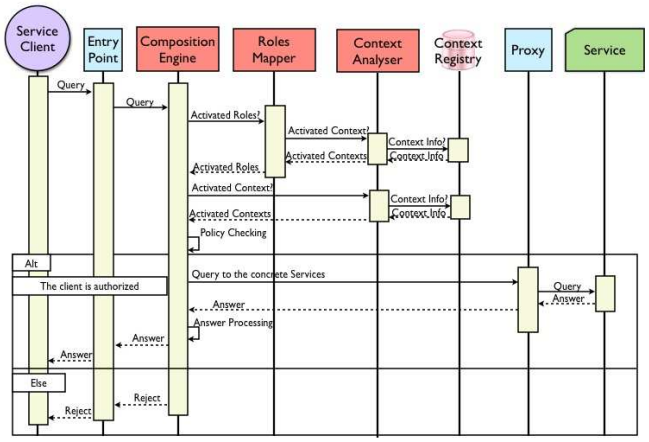

Figure 5. Functioning of the Service Provider.

The `Composition Engine` orchestrates the access control policy checking by querying the `Roles Mappers` and the `Context Analyzer`. The first determines the `Roles` the `Service Consumers` can activate according to the `Contexts` that are activated for it. The later determines the `Context` activated at the level of the composition. The `Composition Engine` checks if the access control policy is satisfied and rejects the query if not or invokes the concrete Services otherwise.

C. Implementation

We now apply our approach to our running example. To do so, we first model the access control requirements and then provide details on their enforcement at execution.

1) Design Phase: Modeling Access Control Requirements: We focus here on the modeling of the access rights to an emergency notification and a medical file for a physician. We identify a `Role`, Physician, two `Contexts`, Available and Engaged in an Emergency Management, and a `Transition` between these two contexts, the acceptation of the notification. The instantiation of our meta-model is presented on Figure 6.

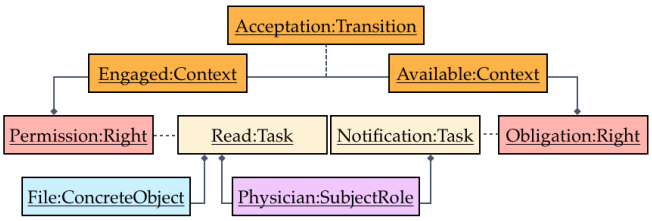

Figure 6. Instanciation of our meta-model for a physician's access rights.

2) Execution Phase: Enforcing Access Control: As we said, the problem of verifying our policy is a model checking problem, consisting in verifying that the composition's current state satisfies the access control policy. We implement it with the algorithm proposed by Bozelli [14]. It runs in polynomial time according to the size of the considered Kripke structure and in doubly exponential time in the size of the logical formula describing the composition.

The generated Service Provider consists in a Web Service. The generation process relies on a Jet Template which takes the instantiation of our meta-model as an input and generates J2EE Code.

For the purpose of the experiment, we run our Service Provider on a Glassfish server. We implement our Service Registry as an UDDI registry and the Context Registry as a Oracle 11i database. As contextual sources, we rely on Google Agenda for the participant's calendar and their smartphone's location facilities.

```
...<task>
        <parameter name="technology"
                value="<\%=serviceImplem\%>"/>
        <parameter name="name"
                value="<\%=taskName\%>"/>
        <parameter name="returnType"
                value="<\%=taskreturn\%>"/>...
```

Figure 7. Extract of the JET for the Service Provider generation.

VI. Conclusion

This paper addresses the enforcement of privacy in a pervasive environment. Privacy is here understood as a problem of access control. We propose a model-driven approach to privacy enforcement based on the clear separation of a design and an execution levels. At the design level, we promote separation of concerns by providing a meta-model for access control. The execution level is generated from the design level by model transformations that bypass manual programming, which is error prone.

Our approach is validated on examples of medical workflows. Medical applications summarize several issues that are highly transferable to a wide variety of domains of application. As such, the rest of our work will be devoted to the application of our proposition to other domains.

References

[1] R. S. Sandhu and P. Samarati, "Access control: Principles and practice," *IEEE Communications Magazine*, vol. 32, pp. 40–48, 1994.

[2] M. P. Papazoglou, "Service-Oriented Computing: Concepts, Characteristics and Directions," in *WISE'03: Proceedings of the Fourth International Conference on Web Information Systems Engineering*, Los Alamitos, CA, USA, December 2003, pp. 3–12.

[3] D. Basin, J. Doser, and T. Lodderstedt, "Model driven security: From uml models to access control infrastructures," *ACM Trans. Softw. Eng. Methodol.*, vol. 15, pp. 39–91, 2006.

[4] S. Chollet and P. Lalanda, "An extensible Abstract Service Orchestration Framework," in *Proceedings of IEEE International Conference on Web Services*. Los Alamitos, CA, USA: IEEE Computer Society, July 2009, pp. 831–838.

[5] B. Carminati, E. Ferrari, and P. Hung, "Security conscious web service composition," in *ICWS*, 2006, pp. 489–496.

[6] M. Srivatsa, A. Iyengar, T. A. Mikalsen, I. Rouvellou, and J. Yin, "An access control system for web service compositions," in *ICWS*, 2007, pp. 1–8.

[7] A. Kumar, N. Karnik, and G. Chafle, "Context sensitivity in role-based access control," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. 3, pp. 53–66, 2002.

[8] F. Hashem and H. Al-Obaidy, "A secure workflow management system (swms) for e-government web based application," in *CSREA EEE*, 2010, pp. 239–246.

[9] M. E. Zurko, R. Simon, and T. Sanfilippo, "A user-centered, modular authorization service built on an rbac foundation," in *IEEE Symposium on Security and Privacy*, 1999, pp. 57–71.

[10] R. Bhatti, A. Ghafoor, E. Bertino, and J. Joshi, "X-gtrbac: an xml-based policy specification framework and architecture for enterprise-wide access control," *ACM Trans. Inf. Syst. Secur.*, vol. 8, no. 2, pp. 187–227, 2005.

[11] K. Sohr, T. Mustafa, X. Bao, and G.-J. Ahn, "Enforcing role-based access control policies in web services with uml and ocl," in *ACSAC*, 2008, pp. 257–266.

[12] E. A. Emerson, "Temporal and modal logic," in *Handbook of theoretical computer science*, J. van Leeuwen, Ed. MIT Press, 1990, ch. Temporal and modal logic, pp. 995–1072.

[13] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1977, pp. 46–57.

[14] L. Bozzelli, "The complexity of ctl* + linear past," in *Proceedings of the Theory and practice of software, 11th international conference on Foundations of software science and computational structures*, ser. FOSSACS'08/ETAPS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 186–200.