Introducing Security Access Control Policies into Legacy Business Processes

Fáber Danilo Giraldo Velásquez, Mireille Blay-Fornarino, Sébastien Mosser

Introducing Security Access Control Policies into Legacy Business Processes

Fáber D. Giraldo
System and Computer Engineering
University of Quindío
Armenia, Colombia
fdgiraldo@uniquindio.edu.co

Mireille Blay-Fornarino
I3S CNRS UMR 6070
University of Nice-Sophia Antipolis
Nice, France
blay@polytech.unice.fr

Sébastien Mosser
INRIA Lille-Nord Europe
LIFL CNRS UMR 8022,
University of Lille I
sebastien.mosser@inria.fr

***Abstract*—Applying separation of concerns approaches into business process context generally results in several initiatives oriented to automatic generation of aspect code, generation of specific code according to the kind of concern (code for mapping roles and permissions derived from RBAC model for example), or proposition of new mechanisms as dedicated aspectual languages. Most of these initiatives only consider functional behaviours of business process, omitting special behaviours derived from quality attributes such as security, which can be modelled as concerns that must be supported in the business process. In this paper we propose the integration of cross-cuttings standardized control access policies (based on RBAC model and Oasis XACML) into legacy business processes, using a separation of concerns approach.**

***Keywords*-Security Standards, Separation of Concerns, Business Processes, Service–oriented Architecture.**

Note for the proceeding reader: this paper makes use of colors. Although not mandatory for its understanding, an online (colored) version of this paper will ease the reading.

I. Introduction

Developing complex software is known to be a tedious task. From a technical point of view, it implies to use multiple paradigms (and the associated frameworks), and consists of thousands of artifacts (*e.g.,* documentation, models, requirements, unit tests). To fill the gap between the technical and enterprise points of views, the *Service-Oriented Architecture* (SOA) paradigm [1] promotes the design of complex software as the collaboration of decoupled services. According to this paradigm, these collaborations are reified as *business processes*, designed by business experts according to their concerns. These processes are supported by existing norm such as the *Business Process Execution Language* (BPEL, [2]). However, the SOA paradigm does not support cross–cuttings concerns (*e.g.,* security, persistence) *per se*; business experts design *tangled* processes, tangling both functional and cross–cuttings concerns together at the process level. This situation increases systems' entropy[1], leading to the deployment of inefficient processes (*e.g.,* anti-patterns [3]).

[1]According to Princeton's WordNet, *entropy* is "a quantity representing the amount of energy in a system that is no longer available for doing work". It is interpreted as the degree of disorder or randomness in the modeled system.

Moreover, the quality requirements imposed in industry rely on several norms or standardized policies (*e.g.,* Oasis XACML, Oasis SAML, WS-SECURITYPOLICY, WS-TRUST, etc). These policies may be interpreted differently (according to the associated business context and stakeholders), leading to misunderstanding issues and avoiding their reuse in another context. *Separation of Concerns* (SoC) advocates the design of large systems through the composition of elementary artifacts. This paradigm provides powerful mechanisms to support the design of complex business processes, as it naturally handles cross–cutting concerns. SoC is a good candidate to support the implementation of standardized composition mechanism for integrating separated concerns into the final system.

The ultimate goal of this work is to automate the integration of cross–cuttings standardized policies (focusing on *security*, using the XACML profile for the *Role-Based Access Control* model - RBAC) into legacy business processes, using a separation of concerns approach. The immediate benefits are the reduction of management and the reuse of previously built RBAC services. To illustrate the approach, we use the *Car Crash Crisis Management System* (CCCMS) case study, defined as a framework to compare several approaches dealing with separation of concerns [4]. We describe in this contribution how a SoC approach can be used to integrate XACML artifacts into existing business processes.

II. Running example: The CCCMS

Kienzle et al. [4] propose a common case study - a *Car Crash Crisis Management System,* CCCMS - to compare existing Aspect Oriented Modeling approaches with each other. According to the definition given by this case study, the CCCMS *"includes all the functionalities of general crisis management systems, and some additional features specific to car crashes such as facilitating the rescuing of victims at the crisis scene and the use of tow trucks to remove damaged vehicles."* The requirements document defines ten use cases, described using textual scenario. Each scenario defines the flow of actions to handle a crisis (*e.g.,* retrieve witness identity, contact firemen located near to the crash location).

We use this case study as a running example to illustrate the problem tackled in this paper and the contribution we made.

A. Introducing Security into Legacy Processes

Traditionally, secure specifications that are part of business processes are implemented as modifications of the business processes themselves. Consequently a non-secure business process is modified to deal with specific secure operations (*e.g.,* authentication). In the CCCMS, we have to deal with secure operations such authentication, access control and encryption.

When several secure operations are involved, security services are invoked several times, producing a repetitive use of security controls. For example, if user authentication is required before invoking any critical operation, the business process is modified to deal with this concern. But when several security services are involved, user can be asked to authenticate several times in the same business process. The problem is even more complex, when several concerns apply to the same process business. Indeed, each new aspect in turn may require security checks. The problem of repetitive checks and late detections of denial of access is then difficult to identify.

One important approach for managing security is represented by the development of policy-based security services in order to provide security operations relevant to business processes. Security is a complex task that cannot be achieved by a single service, multiple coordinated services are needed. Using a policy-based approach increases flexibility, reduces software development costs, and simplifies security management. Changes in security requirements simply involve changing policies without requiring changes to the software, business processes and/or access control and authentication mechanisms.

B. RBAC and XACML

Access control is a strategy for protecting information in a system; it is performed by defining *roles* associated to *access permissions* over specific resources. Access authorization is explicitly described through *policies* or *permissions*. Generally access control includes access security sub-features such *authentication* (degree to which the identity of a subject or resource is checked), *compliance* (adherence to standards, conventions or regulations), and *integrity* (degree to which a system or component prevents unauthorized access to, or modification of, programs or data).

The RBAC model is a generalized approach for access control where roles represent functions defined in an organization and the permissions (*e.g.,* ability or right to perform some action on a resource, possibly under certain conditions previously specified) depend of roles rather than users. Authorizations associated to a role are strictly related to data objects and resources that are necessary to perform the operations associated with that role. Users are simply allowed to play the appropriate role. The main benefit of adopting RBAC is to simplify the definition of security policies for business users who do not have knowledge of software.

The OASIS *eXtensible Access Control Markup Language* (XACML) standard is a proposal to specify and enforce access control policies independent of any software platform. XACML (currently version 3.0) has become the *de facto* standard for specifying access control policies widely used in Web servers. Since 2004, OASIS has defined a XACML profile for RBAC in order to link RBAC practical solutions in web services environments[2]. Management of privileges in a distributed environment, based designs on RBAC solutions, gives greater control of privileges, decreasing the complexity involved in this process. The XACML-RBAC profile also enables the management of security policies in a distributed environment. XACML allows administrators to define access control requirements for protecting resources. The access decision language XACML is used for representing a query that asks if a particular action should be allowed on a resource. If the service finds a policy for the consulted resource, the attributes in the request are compared with the attributes listed in the policy rules. Finally, the control access service provides a response which determines if the request should be allowed for that resource using one of four values: *Permit*, *Deny*, *Indeterminate* (query was failed or a required value was missed, so the service cannot make a decision) or *Not Applicable* (the request cannot be answered by a service).

The XACML architecture establishes the presence of three key components in the access control to resources: the *Policy Enforcement Point (PEP)*, the *Policy Decision Point (PDP)* and *Policy Information Point (PIP)*. A XACML control access process involves the following steps:

1) An user sends to a PEP a request for accessing to a resource.
2) The PEP forms a new request with attributes of the subject, resource and action, and sends this new request to a PDP.
3) The PDP analyzes the new request and obtains a policy or set of policies that apply to the new request from a policy repository.
4) The PDP queries to PIP service in order to obtain the values of attributes related to the subject, resource, or context, that are in the politics, and get a response with the values of these attributes.
5) The PDP compares the attributes of the request sent in step 2 against attributes contained in the politics' rules and makes a response about whether or not should be allowed access. This response is sent to the PEP, which can allow or deny access to the user.

[2]Profile of XACML 3.0 for RBAC can be found in http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-cd-03-en.pdf

III. Reifying Security as a Concern

A RBAC artifact may be considered as a concern (*e.g.,* aspects weaving), and its application could be extended to a set of services, applications or business processes, if it exists a mechanism for adding compositions of services in a particular business process and considering security operations at the level of role-based control access policies.

SoC using aspect-based techniques has been widely accepted as a mechanism to encapsulate functional and non-functional features into reusable modules over a software solution; it can increase the level of abstraction by identifying common characteristics (which is a targeted goal of business process development).

A. *SoC and quality attributes*

The use of SoC and aspects are extended to the treatment of quality attributes (as security and its derived implications, *e.g.,* control access) so that business processes managed within a workflow consider additional features to functionality [5], [6]. Understanding access control as a non-functional concern that cross-cuts the functional part of a system raises suitable considerations for a solution based on aspect-oriented programming [7].

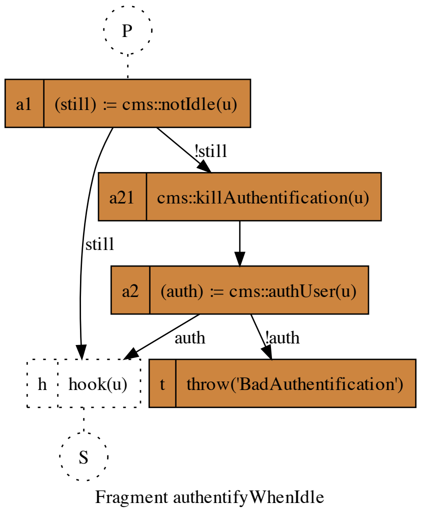
In most contemporary SOA practices focused on the separation of concerns, the properties related with quality attributes are specified and mapped in a set of services. This strategy involves that developers and SOA architects must configure properly the quality attributes in a range of services (usually every quality attribute covers multiple services simultaneously).

Consistent specification and validation of non-functional properties over a set of services of a complex business process, is a tedious, costly and error prone process. Therefore, it is necessary to specify attributes of quality for business processes rather than services, reducing costs and surcharges in the development and application maintenance. Unfortunately, UML, BPMN and BPEL do not support separation of concerns *per se* [8]. The use of concerns managed by models allows us to adjust service orchestrations in high level of abstraction, so that business processes can be enriched with additional features [9] without major modifications or changes in code-level source.

B. *Taming* CCCMS *Complexity Using* ADORE

The **ADORE** approach [10] provides a compositional technique for modeling complex orchestrations, where models describe small orchestrations of services that are composed to produce a model that describes an orchestration of a wide range of services [6]. Small models, called **orchestration fragments**, describe different aspects of a complex business.

In [11] authors expose how ADORE method was applied in order to separate and compose process aspects in a SOA design of the CCCMS. They define service orchestrations from basic flow of use cases, and fragments of orchestration from alternative flows of use cases and non-functional requirements specification. All orchestrations are made by calls to services (even the fragments themselves), due to ADORE focus on the modeling of orchestrations rather than modeling the internal behavior of services or activities.


P
a1
(still) := cms::notIdle(u)
!still
a21
cms::killAuthentification(u)
still
a2
(auth) := cms::authUser(u)
auth
!auth
h
hook(u)
t
throw('BadAuthentification')
S
Fragment authentifyWhenIdle

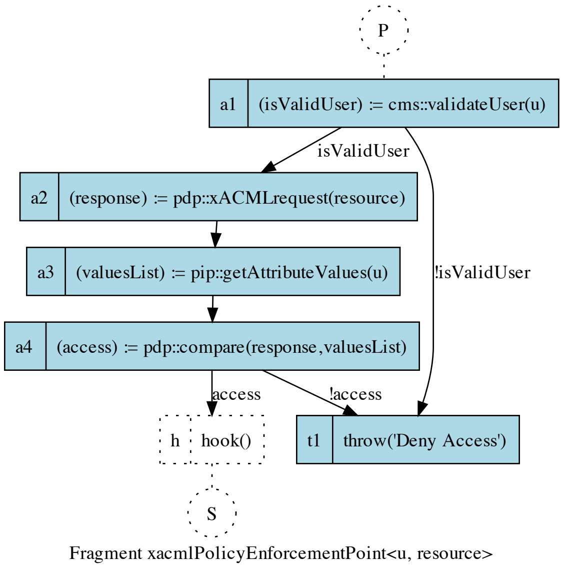
We use here the graphical representation defined by ADORE *to represent business processes. Boxes represent activities* (e.g., *message reception, service invocation), and arrows represent causality relations (*i.e., *the associated partial order). A wait relation (*$\mathtt{a} \to \mathtt{b}$*) means that* b *will wait for the end of* a *to start its own execution. A guard relation (*$\mathtt{a} \xrightarrow{v} \mathtt{b}$*) strengthens the wait semantics, and conditions the start of* b *to the value of* v. *In this example, relations are combined using a conjunctive semantics* ($\wedge$).

Figure 1. *AuthentifyWhenIdle* concern defined with ADORE.

Figure 1 presents an example of an ADORE fragment called *authentifyWhenIdle*. It implements one of the security requirements specified in the CCCMS. Thus, It can be integrated in any orchestration that requires this behavior. The fragment manages the services invoked when an idle employee of CCCMS must be reauthenticated (no active interaction in the last 30 minutes). This fragment contains four activities: ***a1*** activity queries about inactive time of user; ***a21*** activity is the invocation of a service to finish the user session if inactivity is detected; ***a2*** activity is the new invocation of an authentication service, and ***t*** that implements error throwing if user does not re-authenticate properly. ***h*** or *hook* (in ADORE terminology) represents where the fragment will be connected into an existing orchestration. ***P*** represents hook predecessor and ***S*** represents hook successors in an ADORE process structure.

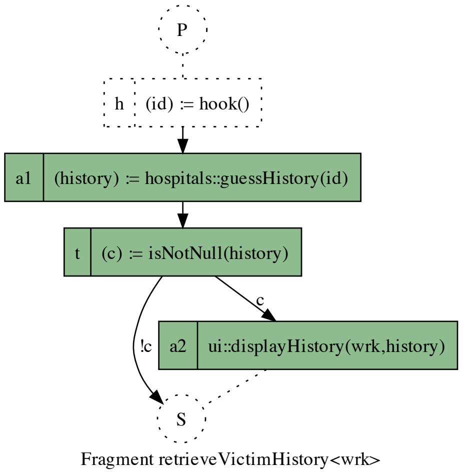
IV. Application to the complete CCCMS

We define a generic RBAC-XACML fragment that implements XACML logic in order to *i)* determine which roles a given user has (*e.g.,FirstAidWorker* coordinator) and



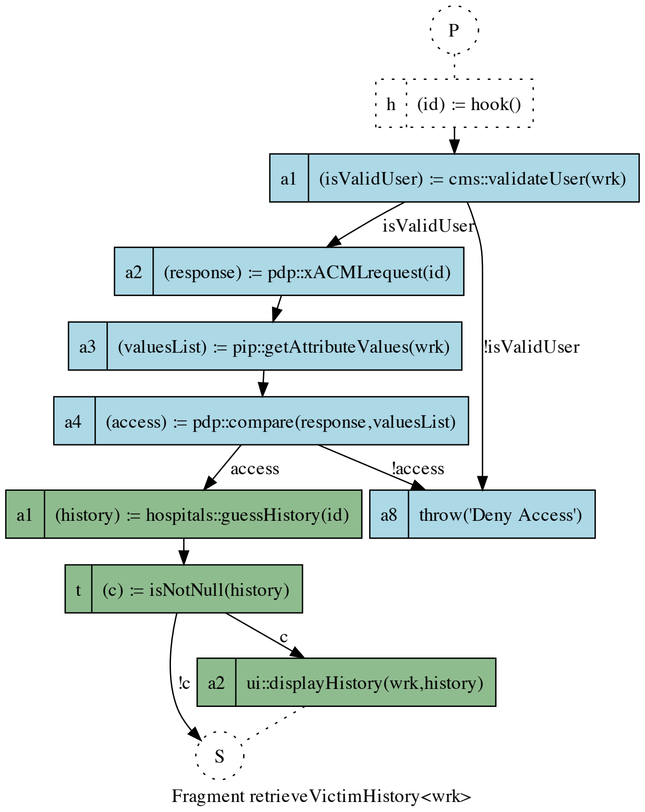
P
a1 (isValidUser) := cms::validateUser(u)
isValidUser
a2 (response) := pdp::xACMLrequest(resource)
a3 (valuesList) := pip::getAttributeValues(u)
!isValidUser
a4 (access) := pdp::compare(response,valuesList)
access
!access
h hook()
t1 throw('Deny Access')
S
Fragment xacmlPolicyEnforcementPoint<u, resource>

Figure 2. ADORE XACML Fragment.



P
h (id) := hook()
a1 (history) := hospitals::guessHistory(id)
t (c) := isNotNull(history)
c
!c
a2 ui::displayHistory(wrk,history)
S
Fragment retrieveVictimHistory<wrk>

Figure 3. *retrieveVictimHistory* fragment



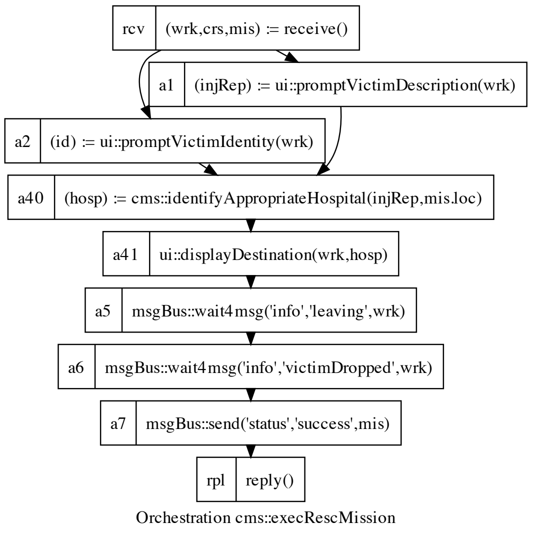
P
h (id) := hook()
a1 (isValidUser) := cms::validateUser(wrk)
isValidUser
a2 (response) := pdp::xACMLrequest(id)
a3 (valuesList) := pip::getAttributeValues(wrk)
!isValidUser
a4 (access) := pdp::compare(response,valuesList)
access
!access
a1 (history) := hospitals::guessHistory(id)
a8 throw('Deny Access')
t (c) := isNotNull(history)
c
!c
a2 ui::displayHistory(wrk,history)
S
Fragment retrieveVictimHistory<wrk>

Figure 4. *retrieveVictimHistory*, after composition with XACML fragment

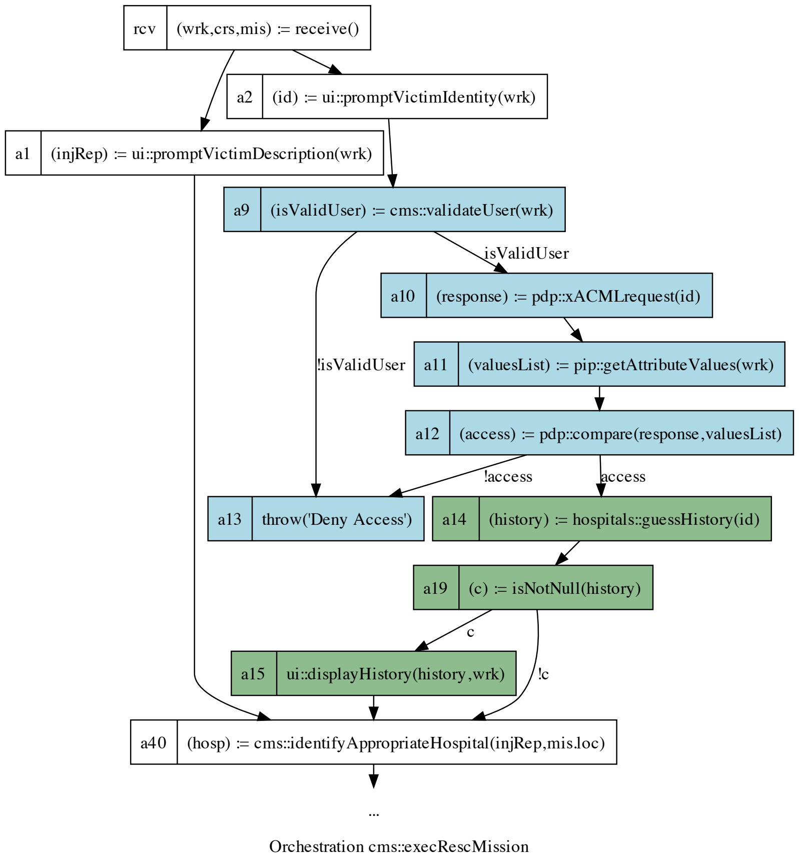
ii) determine if that role has access to a specific resource (*e.g.,* medical history information). This fragment contains three basic XACML-RBAC operations: *i)* it validates user with regard to roles defined in XACML policy file, *ii)* it establishes which resources can be accessed by role, and *iii)* establishes permission of role according to the resource. This fragment could be woven in several orchestrations of CCCMS, where the user accesses any critical resource.

The XACML security fragment (Fig. 2) for accessing control is in itself a Policy Enforcement Point (PEP), according to the XACML request/response process defined in [12]. This *PEP* captures the access request done by the user, invokes to *PDP* for getting the policies related to a request (cf. action **a2** in Fig. 2), invokes the *PIP* for getting the attributes of the policy found by the *PDP* (cf. action **a3** in Fig. 2), and then compares them with the attributes of the access request for proceeding to allow or deny the access (cf. action **a4** in Fig. 2). The XACML access control process is encapsulated in an ADORE fragment, and subsequently this fragment can be integrated with larger services orchestrations or fragments in order to adapt the business process. Thus it is possible to adapt the workflow through concern composition.

For example, the third step of main success scenario of *Execute Rescue Mission* explicitly specifies: *System requests victims medical history information from all connected HospitalResourceSystems. FirstAidWorker administers first aid procedures to victim.* So, we can compose the XACML fragment to *retrieveVictimHistory* fragment (Fig. 3) to support this specification. Figure 4 shows the result of the composition. Figure 5(a) presents the orchestration *Execute Rescue Mission* in its initial state, and Figure 5(b) the results of the composition. In the same way we can weave the XACML fragment in other CCCMS fragments. It can be weaved in activities that represent calls to hospital services in order to validate the access to medical history of victim by *FirstAidWorker* or *System* roles.


rcv
(wrk,crs,mis) := receive()
a1
(injRep) := ui::promptVictimDescription(wrk)
a2
(id) := ui::promptVictimIdentity(wrk)
a40
(hosp) := cms::identifyAppropriateHospital(injRep,mis.loc)
a41
ui::displayDestination(wrk,hosp)
a5
msgBus::wait4msg('info','leaving',wrk)
a6
msgBus::wait4msg('info','victimDropped',wrk)
a7
msgBus::send('status','success',mis)
rpl
reply()
Orchestration cms::execRescMission

(a) Initial version


rcv
(wrk,crs,mis) := receive()
a2
(id) := ui::promptVictimIdentity(wrk)
a1
(injRep) := ui::promptVictimDescription(wrk)
a9
(isValidUser) := cms::validateUser(wrk)
isValidUser
a10
(response) := pdp::xACMLrequest(id)
!isValidUser
a11
(valuesList) := pip::getAttributeValues(wrk)
a12
(access) := pdp::compare(response,valuesList)
!access
access
a13
throw('Deny Access')
a14
(history) := hospitals::guessHistory(id)
a19
(c) := isNotNull(history)
c
a15
ui::displayHistory(history,wrk)
!c
a40
(hosp) := cms::identifyAppropriateHospital(injRep,mis.loc)
...
Orchestration cms::execRescMission

(b) After composition with additional concerns

Figure 5. The *ExecuteRescueMission* orchestration

V. Deriving other security fragments

Through XACML fragment previously exposed, it is possible to deduce a complex context of critical information exchange in CCCMS where having access control rules is not enough for preventing malicious appropriation of the information that will be exchanged between the CCCMS and all hospitals resources to which it connects to consult the medical history of a victim. This type of information is considered highly protected due to legal and ethical implications. Works like [13] and [14] proposed strengthening and flexibility of access control mechanisms to incorporate encryption operations in the exchange of information, so that only users with valid access privileges are responsible to decrypt and access medical information.

We propose a small generic fragments to manage decryption behaviors using RSA - X.509 based standard, according to CCCMS security requirement (*All communications in the system shall use secure channels compliant with AES 128 encryption standard*), depicted in Figure 6. This fragment can be composed with XACML and *retrieveVictimHistory* fragments: we interlace it with XACML fragment, desencrytping the message (Victim History) that arrives from an hospital. Once users with valid access control privileges is established. Figure 7(a) shows the composition of *decrypt* and *XACML* fragment. Finally, Figure 7(b) presents the adapted *Execute Rescue Mission* orchestration with the addition of three news fragments for managing secure operations.

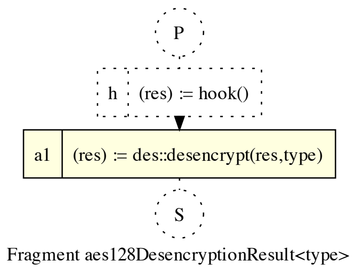
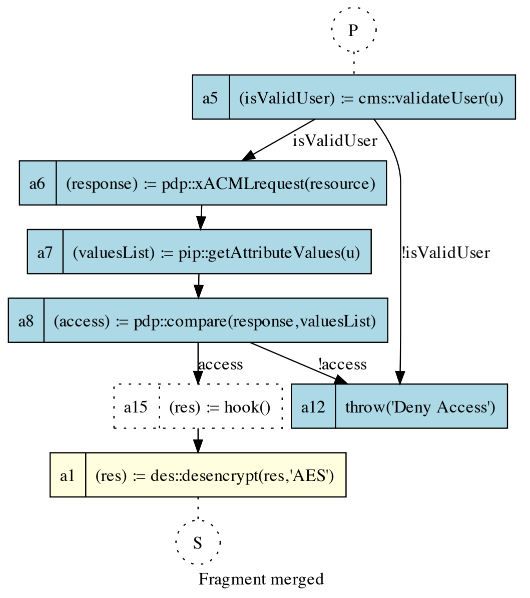
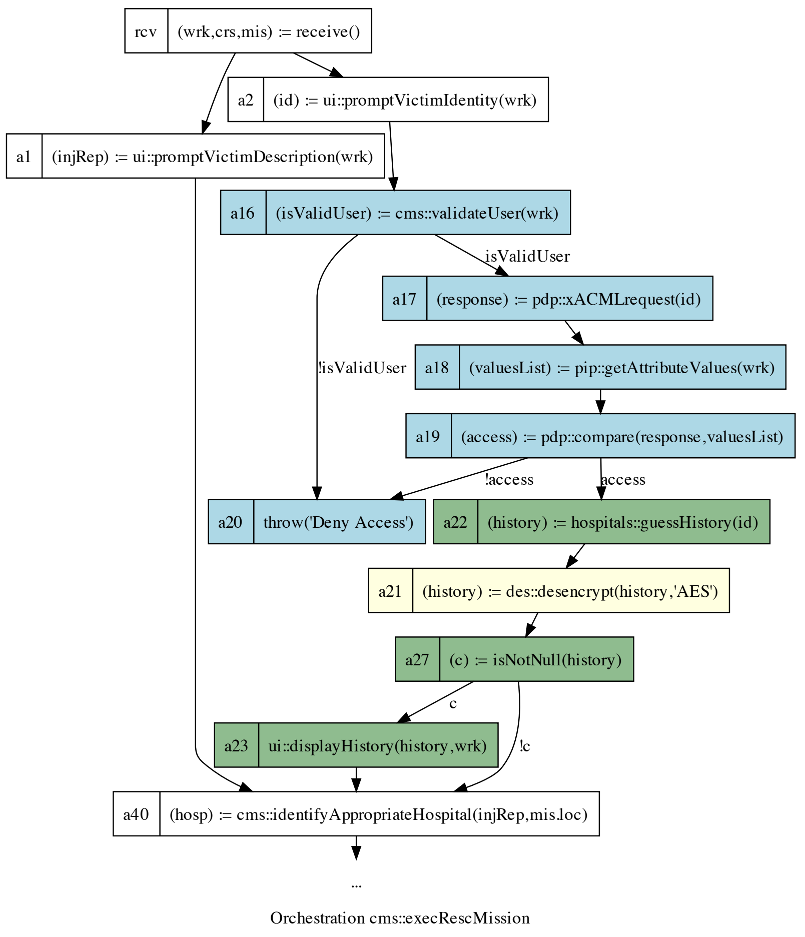

P
h
(res) := hook()
a1
(res) := des::desencrypt(res,type)
S
Fragment aes128DesencryptionResult<type>

Figure 6. *RSA X.509 desencryption* operation defined as a fragment



P
a5 (isValidUser) := cms::validateUser(u)
isValidUser
!isValidUser
a6 (response) := pdp::xACMLrequest(resource)
a7 (valuesList) := pip::getAttributeValues(u)
a8 (access) := pdp::compare(response,valuesList)
access
!access
a15 (res) := hook()
a12 throw('Deny Access')
a1 (res) := des::desencrypt(res,'AES')
S
Fragment merged

(a) Composing *decrypt* and XACML fragments



rcv (wrk,crs,mis) := receive()
a2 (id) := ui::promptVictimIdentity(wrk)
a1 (injRep) := ui::promptVictimDescription(wrk)
a16 (isValidUser) := cms::validateUser(wrk)
isValidUser
!isValidUser
a17 (response) := pdp::xACMLrequest(id)
a18 (valuesList) := pip::getAttributeValues(wrk)
a19 (access) := pdp::compare(response,valuesList)
!access
access
a20 throw('Deny Access')
a22 (history) := hospitals::guessHistory(id)
a21 (history) := des::desencrypt(history,'AES')
a27 (c) := isNotNull(history)
c
!c
a23 ui::displayHistory(history,wrk)
a40 (hosp) := cms::identifyAppropriateHospital(injRep,mis.loc)
...
Orchestration cms::execRescMission

(b) Resulting *Execute Rescue Mission* orchestration

Figure 7. Final orchestration: *ExecRescMission* ⊕ (*requestVictimHistory* ⊕ (*decrypt* ⊕ XACML))

VI. Related Work

A. SoC applied to business process

The design and implementation of business workflows can be viewed as high-level programming, where management functionality and quality attributes characterize the blocks within a workflow. Process modeling techniques can be used to provide explicit representations of the relationships between enterprise applications, business workflows sections and quality attributes. Process models provide the conceptual basis to define when and under what conditions business applications are invoked in the context of integration scenarios [15], such the integration between business process definition and RBAC security exposed in this work. Principles derived from *AOSD* and *MDD* provide a high degree of flexibility: AOSD can be applied to identify common concerns and visualize scenarios where those concerns can be applied throughout the business process that is being automated in a workflow; model process can be adapted to get new requirements, and further changes to models process can be applied immediately to adapt business processes.

Sánchez et al [16] proposed the concept of *Early Aspect* to focus on the identification and separation of common concerns in requirements specification and system architecture levels, by combining aspect oriented development principles and model-driven development (MDD). In [17] authors proposed the use of cross-cutting aspects for *Enterprise Architecture* modeling, so cross-cutting aspects refer to EA-related concepts, like*goals, standards, lifecycle* or *projects*, which may exert influence on other concepts of the EA model.

B. Security in traditional business process approaches

Business processes are built by combining services through a process specification language. This language determines the tasks and the order in which these tasks should be executed. The *Business Process Execution Language* (BPEL) has become the *de facto* language to implement business processes based on web services. However, despite of the significant progress that has led to the development of a language for business process, major changes relating to the handling of quality attributes need to be managed before workflow execution.BPEL does not provide any support for specifying authorization policies or constrains over the execution of activities of a business process.

Paci et al in [18] expose an approach for encompassing the BPEL processes resiliency problem in RBAC-WS-BPEL, an authorization BPEL model that supports the specification of authorizations for the execution of BPEL process activities by roles and users and authorization constraints. Authors in [19] also propose an access control model called BPEL4RBAC in order to secure BPEL codes. Our proposal raise the level of abstraction and works on business process models instead of raw source code.

C. Access Control and Business Processes

There have been previous attempts for deriving RBAC control access policies from high abstraction levels that combine Model-driven approaches and Aspects Oriented development. In [20] authors propose the formalization of a translation process of access-control policies into aspect code. They propose the *role slice* concept which denotes a set of class methods that a given role can access, and represents the separate concern that captures permissions for roles. Role-slice access-control policy (RBAC requirements) are translated to an aspect-oriented programming (AOP) enforcement code. Braga in [7] proposes a MDA approach for supporting the specification and validation of code generation for control access policies, targeting an aspect-based infrastructure. His proposal contains *i)* a code generator that is a transformation from SecureUML models, and *ii)* a language Aspects for Access Control (AAC) which is proposed by the author in order to map the transformation from SecureUML models. In [21] authors propose role-based access control policies for Web Services using Layered Model-driven architectures and Agile modeling security principles for enhancing security requirements. Authors use Agile Methods that support concerns testing throughout *acceptance tests* and model driven approaches for specifying models that support the integration of Agile Modeling with Security Activities. However, the above proposals do not considerate the reuse of orchestrations that implement standards such as RBAC and XACML and their introduction into business processes. Additionally, these previous attempts do not consider the encapsulation of special behaviours derived from quality attributes as security, using aspectual cluster of services invocations. Instead, the proposals are oriented to aspect code generation from RBAC policies expressed in class models that represent control access requirements.

Gronmo proposes in [22] the use of algebraic graph transformation for specifying BPEL aspects at the modeling level, but his work does not encompass concerns derived from quality attributes such security. Authors in [23] propose an aspect-oriented extension to BPMN called *AO4BPMN* under the motivation of incorporating aspect-oriented concepts in business process modeling languages; this proposal is intended to resolve the lack for expressing concerns in BPMN and business process modeling languages, but AO4BPMN does not include information about how to weave the aspects with a base model. In our proposal we show the weaving between fragments in order to add new behaviours (concerns) derived from control access security considerations into a workflow previously conceived.

VII. CONCLUSIONS

This work describes an approach for integrating concerns derived from security requirements (control access and encryption operations) into legacy process. Our work proposes to include security properties into a business workflow at business modeling level, controlling its impact on the business process itself, and managing several concerns that intertwine with each other. We have shown how concerns derived from security properties can be managed in an efficient, maintainable, reusable and extensible way at model level, according to the context, variations of rules and requirements of business process, and using service secure standards. Concretely, we have built ADORE fragments for representing security services calls and woven them (in AOD terminology) with previous fragments and orchestrations that represent legacy business process. In particular we have defined a control access concern as a fragment that invokes existing XACML services in a typical role based access context.

We have applied this approach to secure a common case study (CCCMS) that involves multiple concerns that affect the business processes. We have shown how this approach supports the integration of concerns by several experts, without adding some extra mechanism for supporting weaving of concerns : an expert defines an enrichment of the business process according with a specific business requirement, and another secures the business process by incorporating secure operations as role- based access control and encryption. If the composition is managed at the middleware level, the first expert is not aware of the change in the process. In particular it is here that we can find security needs such as the *emergency access to confidential information*. Conversely, the security expert must ensure the protection in all information access during the adjustment process. This approach has been useful for taming the complexity of concerns integration in early stages of business process building, without radical extensions or modifications to languages such WS-BPEL and BPEL.

For future work, we would like to explore automatic compositions of sets of fragments according to ”concern directives”. We plan also to dig further for optimizing the result of weavings in multiple business processes.

Acknowledgments

This work is supported by Ministry of Higher Education and Research, Nord-Pas de Calais Regional Council and FEDER through the Contrat de Projets Etat Region Campus Intelligence Ambiante (CPER-CIA) 2007-2013. It is also partly funded by the ANR SALTY project (https://salty.unice.fr) under contract ANR-09-SEGI-012. We also thanks the Latin American and Caribbean Collaborative ICT Research Federation (LACCIR Virtual Institute, http://www.laccir.org/laccir/) for partial funding of this work through the LACCIR Short Stay Program 2010, grant code S1110LAC009. F.G. thanks Prs Alexandre Bergel and Sergio Ochoa (University of Chile), Raquel Anaya and Luis Fernando Londoño (EAFIT University) for their collaborations.

References

[1] OASIS, "Reference Model for Service Oriented Architecture 1.0," OASIS, Tech. Rep. wd-soa-rm-cd1, Feb. 2006.

[2] M. Robinson, H. Shen, and J. Niu, "High Assurance BPEL Process Models," in *High Assurance Services Computing*, L.-J. Zhang, R. Paul, and J. Dong, Eds. Springer US, 2009, pp. 219–240.

[3] N. Trcka, W. van der Aalst, and N. Sidorova, "Data-Flow Anti-patterns: Discovering Data-Flow Errors in Workflows," in *Advanced Information Systems Engineering*, ser. Lecture Notes in Computer Science, P. van Eck, J. Gordijn, and R. Wieringa, Eds. Springer Berlin / Heidelberg, 2009, vol. 5565, pp. 425–439, 10.1007/978-3-642-02144-2_34.

[4] J. Kienzle, N. Guelfi, and S. Mustafiz, *Crisis Management Systems: A Case Study for Aspect-Oriented Modeling*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6210, pp. 1–22.

[5] F. Agung, "Concern-Oriented Model-Driven Development Framework," in *Software Engineering Conference, Australian*, B. Clive, Ed., vol. 0, pp. 527–535.

[6] S. Mosser, M. Blay-Fornarino, and M. Riveill, "Web Services Orchestrations Evolution: A Merge Process for Behavioral Evolution," in *Software Architecture*, ser. Lecture Notes in Computer Science, R. Morrison, D. Balasubramaniam, and K. Falkner, Eds., vol. 5292. Springer Berlin / Heidelberg, 2008, pp. 35–49.

[7] C. Braga, "A Transformation Contract to Generate Aspects from Access Control Policies," *Software and Systems Modeling*, pp. 1–15, 2010.

[8] H. Wada, J. Suzuki, and K. Oba, "Early Aspects for Non-Functional Properties in Service Oriented Business Processes," pp. 231–238, 2008.

[9] C. Courbis and A. Finkelstein, "Weaving Aspects into Web Service Orchestrations," in *IEEE International Conference on Web Services*, pp. 219 – 226.

[10] S. Mosser, "Behavioral Compositions in Service-Oriented Architecture," Ph.D. dissertation, 2010.

[11] S. Mosser, M. Blay-Fornarino, and R. France, "Workflow Design Using Fragment Composition - Crisis Management System Design through ADORE," *T. Aspect-Oriented Software Development VII*, vol. 7, pp. 200–233, 2010.

[12] X. Jin, "Applying Model Driven Architecture approach to Model Role Based Access Control System," 2006.

[13] A. E. Flores, K. T. Win, and W. Susilo, *Secure Exchange of Electronic Health Records* . IGI Global, 2011, pp. 1–22.

[14] E. Mytilinaiou, V. Koufi, F. Matamateniou, and G. Vassilacopoulos, *A Context-Aware Authorization Model for Process-Oriented Personal Health Record Systems* . IGI Global, 2011, pp. 46–65 pp.

[15] M. Weske, *Business Process Management Concepts, Languages, Architectures*, springer ed. Springer, 2007.

[16] P. Sanchez, A. Moreira, L. Fuentes, J. Araujo, and J. Magno, "Model-Driven Development for Early Aspects," *Information and Software Technology*, vol. 52, no. 3, p. 24, 2010.

[17] S. Buckl, F. Matthes, and C. M. Schweda, "Conceptual Models for Cross-Cutting Aspects in Enterprise Architecture Modeling," *Enterprise Distributed Object Computing Conference Workshops, IEEE International*, vol. 0, pp. 245–252, 2010.

[18] F. Paci, R. Ferrini, Y. Sun, and E. Bertino, "Authorization and User Failure Resiliency for WS-BPEL Business Processes," in *Service-Oriented Computing ICSOC 2008*, ser. Lecture Notes in Computer Science, A. Bouguettaya, I. Krueger, and T. Margaria, Eds. Springer Berlin / Heidelberg, 2008, vol. 5364, pp. 116–131, 10.1007/978-3-540-89652-4-12. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-89652-4-12

[19] X. Wang, Y. Zhang, H. Shi, and J. Yang, "BPEL4RBAC: An Authorisation Specification for WS-BPEL," in *Web Information Systems Engineering - WISE 2008*, ser. Lecture Notes in Computer Science, J. Bailey, D. Maier, K.-D. Schewe, B. Thalheim, and X. Wang, Eds., vol. 5175. Springer Berlin / Heidelberg, 2008, pp. 381–395, 10.1007/978-3-540-85481-4-29. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85481-4-29

[20] J. Pavlich-Mariscal, L. Michel, and S. Demurjian, *A Formal Enforcement Framework for Role-Based Access Control Using Aspect-Oriented Programming*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, vol. 3713, pp. 537–552.

[21] K. Rao, M. Rao, K. Devi, D. Kumar, and M. Kumar, "Web Services Security Architectures using Role-Based Access Control," *(IJCSIT) International Journal of Computer Science and Information Technologies*, vol. 1(5), pp. 402–407, 2010.

[22] R. Gronmo, "Can Graph Transformation Make Aspect Languages for BPEL Redundant?" *Enterprise Distributed Object Computing Conference, IEEE International*, vol. 0, pp. 153–162, 2010.

[23] A. Charfi, H. Mller, and M. Mezini, *Aspect-Oriented Business Process Modeling with AO4BPMN*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6138, pp. 48–61.