Security architecture metamodel for Model Driven security

Makhlouf Q1, Adel Alti, Mohamed Gasmi, Philippe Roose

HOSTED BY
ELSEVIER
Available online at www.sciencedirect.com
ScienceDirect
journal homepage: www.elsevier.com/locate/jides

Security architecture metamodel for Model Driven security

Q1 *Makhlouf Derdour*[a], *Adel Alti*[b,*], *Mohamed Gasmi*[a], *Philippe Roose*[c]

[a] *LRSD Laboratory, University of Tebessa, 25000, Algeria*
[b] *LRSD Laboratory, University of Setif-1, 19000, Algeria*
[c] *UPPA – Anglet, 64000, France*

ARTICLE INFO

Article history:
Received 26 October 2015
Received in revised form
1 December 2015
Accepted 1 December 2015

Keywords:
SMSA
Security connector
UML profile
ADL
OCL
Vulnerability point's detection architectures

ABSTRACT

A key aspect of the design of any software system is its architecture. One issue for perpetually designing good and robust architectures is the new security concepts. Many new applications are running on powerful platforms that have ample rich architecture models to support multiples security techniques and to explicit several security constraints. The design of an architecture meta-model that considers security connectors is required in order to ensure a realistic secure assembly and to address the problems of vulnerability of exchanging data flow. Our research proposes a generic meta-modelling approach called SMSA (Security Meta-model for Software Architecture) for describing a software system as a collection of components that interact through security connectors. SMSA metamodel is modeled as a UML SMSA profile. We exploit UML powerful capacities (meta-models and models) to define security concepts of SMSA (e.g. security connectors, composite and domain). A major benefit of UML profile is to the faithful representation of connectors to support the definition of security connector types explicitly and to support them with the ability to associate semantic properties. We also provide a set of model transformations to fit security requirements of a system. These transformations are detailed and validated with phosphate support system (SAGE) for the company FERPHOS: a case study described in SMSA. The model is tested and validated with the semantic constraints defined by the profile using Eclipse 3.1 plug-in in this case study.

1. Introduction

Nowdays, modern computer systems and applications are heterogeneous, connected to Internet and deployed on large scale machines independently administered, serving people anytime and anywhere. Development environments that support their implementation are unstable (e.g. develop applications whose heart is independent of volume, users and devices using adaptive technologies to respond to each case) and applications must deal with the volatility of resources

Peer review under responsibility of Qassim University.
* *Corresponding author.*
E-mail addresses: m.derdour@yahoo.fr (M. Derdour), altiadel2002@yahoo.fr, alti.adel@univ-setif.dz (A. Alti), mohamed_gasmi@yahoo.fr (M. Gasmi), Philippe.Roose@iutbayonne.univ-pau.fr (P. Roose).

http://dx.doi.org/10.1016/j.jides.2015.12.001

Please cite this article in press as: M. Derdour, et al., Security architecture metamodel for Model Driven security, Journal of Innovation in Digital Ecosystems (2015), http://dx.doi.org/10.1016/j.jides.2015.12.001

and services [1]. They must be flexible and be able to adapt dynamically (*e.g. data persistence, data exchanges between heterogeneous applications, move data to remote sites, management of continuous data consistency, interoperability platforms, application portability, managing concurrency, openness and security*). So, it may be very useful to provide an appropriate approach for designing applications taking into account both security needs and problems, such as Architecture Description Language (ADL).

In IT, security is always a major concern and has been well studied. Earlier works [1] have focused on discovering new security techniques, whereas more recent studies [2] have suggested that the security efficiency for the whole system is actually more important to make our lives safety and easier. To achieve security efficiency is to put as many system components in the security mode for communicating and processing. One issue for perpetually providing new secure services (e.g. e-commerce and sensitive communications, etc.) on many distributed devices is the new security aspects. Running large distributed devices and communicating among them will need several security mechanisms. Researchers have proposed various security techniques [2] usually incorporated too late into an application using ad hoc solutions. This raises several problems:

- Integration of security mechanisms into a complete system is difficult and regards as a poor approach because many security properties emerge from the arrangement of all almost essential components of an application;
- Failure to address security concerns (e.g. security system management, user sessions and roles management, etc.) from employing individual developers without concrete guidelines for building more stringent and robust applications;
- The security management applications for e-commerce and sensitive communications depend on distributed components and platforms;—Difficulty to meeting service security because few tools supports for security analysis and secure system design.

These disadvantages can be tided, if we manage significant concerns such as security communication management and information exchanges between components at architecture level that must be consistent and correct. Thus an efficient mechanism is provided making the security requirements easy to manage and associate with the intelligent design tool for discovering vulnerability points that require security techniques and implementation mechanisms between the components during system design. In this way, we can effectively SMSA: Security Approach for Model-Driven Security 3 satisfy security-related requirements and achieve robust configuration for system success.

In this paper, we propose a Security Meta-model of Software Architecture (SMSA) for maintaining architectures coherency by preserving consistency of distributed components throughout a (re) assembly or a (re) configuration. We provide a new set of common and generic architectural security elements (domain: which gives a direct support of distribution components in several geographically remote sites, security services: which allows a complete a need of security, etc.) and various security connectors which allows to add semantic details to architectural security elements and their interactions. These connectors incorporate the required security services as well as qualitative extensions of those services to provide a measure of QoS reflecting the evolution security needs of data stream exchanged between components. Our approach provides the intelligent detection of possible vulnerability points between components, and conducts mapping security connectors among them. In this way, our approach can support easy (re) assembly of components and connectors and robust configuration for IT applications. Our contribution in this paper includes:

- We define a Security Meta-model of Software Architecture (SMSA), a software architecture meta-model that takes into consideration the concept of security separately from functional components by means of security connectors.
- We propose model transformations for integrating security connectors.
- We build a UML 2.0 profile for SMSA to define a complete specification for integrating new concepts of security into UML.
- We provide full support for supporting the UML 2.0 profile for SMSA and semantics checking of architectural security properties.

The reminder of this paper is organized as follows: Section 2 presents the security concepts of SMSA software architecture meta-model. Section 3 details how we propose model transformations and vulnerability point detection to include secure connector, which could adopt various security techniques in one architecture model, to select best security strategy that guarantees robustness of architecture model. Section 4 details UML SMSA Profile. Section 5 provides a case study described in SMSA. Section 6 discusses related work. Section 7 concludes this paper and gives resources for further reading.

2. SMSA metamodel

The intention is to include security issue at the architectural design in a sole approach called Security Software Architecture Meta-model (SMSA) benefits from a precise and common vocabulary definition for design actors (*architects*, *designers*, *developers*, *integrators and testers*). SMSA approach specifies the abstract architecture of components without implementation details. They explicitly de- fine interactions between system components and provide modeling support to help designers to structure and compose the different elements. Obliging components to communicate via secure connectors has number of significant benefits including: increasing reusability (the same component can be used in different environments, each of them providing specific security techniques (*i.e. watermarking technique, DCT-XOR technique, etc.*) direct support for distribution, mobility and connectivity of components. This approach includes a composition description including dependencies between components and communication rules and separates a connector's interface from its security behavior.

Architectural Description Language (ADL) means three C: *Components*, *Connectors* and *Configurations* [3]. Components

Please cite this article in press as: M. Derdour, et al., Security architecture metamodel for Model Driven security, Journal of Innovation in Digital Ecosystems (2015), http://dx.doi.org/10.1016/j.jides.2015.12.001

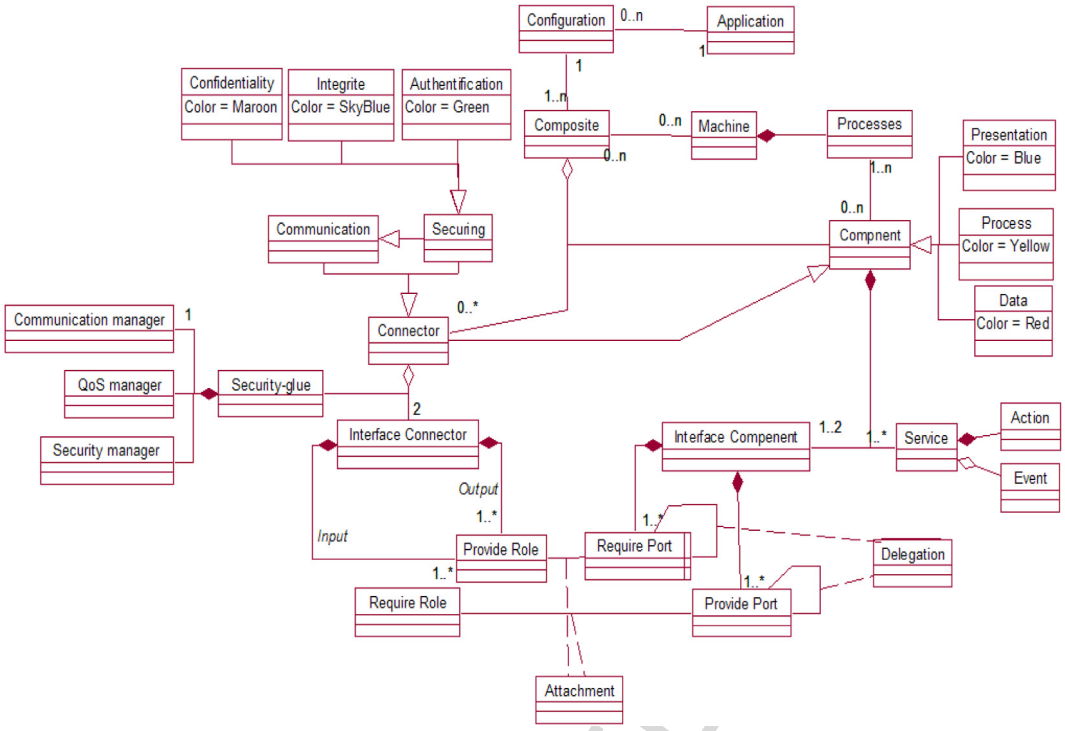


Fig. 1 – Metamodel of SMSA approach.

represent the computational units and data storage in a software system. The interaction between these components is encapsulated by the connectors. A configuration corresponds to components and connectors instantiations. It binds them together in order to form the complete system. Some ADLs such as Rapide [4] or Darwin [5], for example, do not clarify the concept of connector. Other ones agree to hierarchical description of the components. Components can be seen as white boxes and might contain subcomponents. In some ADLs such as Rapide, components are considered primarily as black boxes. In UniCon [6], Wright [7], Acme [8] and MMSA [9] we can define composite connectors, whereas it's impossible in others ADL. In most ADLs we find the following:

- The management of the non-functional concerns of components is ensured after the definition of the global architecture and the configuration of the components.
- The management of assembly does not take into account the vulnerability problems and the management risks caused by the distributed applications nature, which makes difficult the employment of security after configuring and deploying the application;
- Few models are able to define new connectors with various services that ensure non-functional concerns of components (*security*, *communication*, *conversion*, *etc.*);
- There is no automatic and direct correspondence between architectures (models) and its corresponding applications (instances).

Fig. 1 presents a model of the SMSA (Security Software Architecture Metamodel) approach. SMSA supports number of architectural elements including *components*, *connectors*, *configurations* and *domains*.

An important aspect of SMSA architecture is to offer a container of various components of composite machine and process in several configurations called *domains*.

The key role of *configurations* in SMSA is to abstract the details of different components and connectors. A configuration has a name and defined by interfaces (ports and services).

Components represent the computational elements (*Process*), user interfaces (*Presentation*) and data stores of a system (*Data*). Each component may have an interface with multiple security services. The interface consists of a set of points of interactions between the component and the external world that allow the invocation of services and is attached with security properties. A component can be primitive or composite.

Connectors represent interactions among components; they provide the link for architectural designs. SMSA connector is mainly represented by an interface and a security glue specification. In principle, the interface shows the necessary information about the connector, including the service type that a connector provides (communication, authentication, integrity and confidentiality). Connectors can be composite or primitive.

Interfaces in SMSA are first-class entities. They provide connection points among architecture elements. Likewise, they define how the communication between these elements can take place. A component/configuration interfaces connection point is called port and a connector interfaces connection point is Q3

2.1. SMSA component

The concept of component is used to represent any element providing functionality within an application. In other words,

any feature that is part of the logic application is explicitly supported by a component [10]. A component can represent a complex application that consists of other less complex applications. It can also be a feature as a simple arithmetic operation.

In our meta-model we distinguish three types of SMSA components: *Presentation*, *Process* and *data* which help us to detect vulnerability points that require consideration of security. SMSA component is described by two interfaces (provided and required) and a set of services that are presented in the form of actions and events.

The component interface (Fig. 1) is the definition of a set of interaction points between the component and the external environment; it forms what are known as protocols. SMSA extends the concept of interface as defined in most other ADL by a set of security services of different types (*authentication*, *integrity*, and *confidentiality*). Indeed, a service can be provided or required by the interface. Provided service must be implemented by the component that exposes the interface and required service is required by this component.

The interface of SMSA component describes ports, through which they communicate. For example, a component of *DATA* type may have two ports, one for *consultation* and one for the *update*.

2.2. SMSA connector

The main element of our proposal is the connector, which is the key communication structure between components. The semantic connection is not only the exchange of information or the invocation of component services, but also the proposal of solutions to address security issues between the business components in order to avoid changes to system functionality. The connector is a first class entity because he does not play the traditional simple role related to communication, but it also includes security insurance of data flows exchanged. There are two types of connector:

- *Communication connector*: The communication connector provides a connection for exchanging information between business components. We find this type of connector between two components residing in the same address space.
- *Security connector*: The role of the security connector is to ensure a secured exchange of data between components. This function applies according to security aspect used and controlled by the QoS manager. It can parameterize security services according to sefety needs of components and the environment in order to ensure proper delivery of data flow (see Fig. 1). This type of connector connects tow components that are encapsulated in two different abstraction spaces (*processes, machines, composite*).

For example if a manufacturer component provides important information in a buffer and another component consumes contents of this buffer. The encryption and decryption of data is provided by a connector because the two components are not in the same process.

The SMSA connector also includes two parts: the first is the visible part: the interface describing the roles of participants in an interaction. These roles de- fine communication modes (*synchronous, asynchronous and continuous*) and connection types (e.g. *GPRS*, *WAP*, *MMS*, *etc.*) connection between components. The second part is the glue that implements security mechanisms for communication/exchange of information and services for securing and managing QoS of components.

A SMSA connector is defined by two interfaces *Input/Output* and *glue* which are represented by three managers: communications, security and QoS. It manages the data transfer between components and allows security operations. An interface *required/provided* of a connector consists of a set of roles. Each role serves as a point through which the connector is connected to a component. Thus two components can be linked by a connector, furthermore two connectors can be linked together to ensure complex securities. For the connector, the glue was enriched by a *security manager* that works with a service quality manager to ensure the security task. The *security manager* is a set of security services that cooperate to achieve security.

2.3. SMSA configuration

An architectural configuration (or just architecture or system) is a graph that shows how a set of components are connected to each other via connectors. The graph is obtained by combining ports of components with roles of connectors that are suitable to build the application.

The goal is to abstract details of configurations of various components and connectors (*restricting components access through interfaces*). Configuration has a name and can have an interface represented by components interfaces *provided/required* oriented *to/from* external environment and a set of services encapsulated in components. A *composite* is a unit of description of a configuration and entity structuring an application into cooperating components. The composite is considered as a hierarchy of component types where the root represents the application. A composite is a set of *machines*. Each machine executes a process that composed of sub components. These elements (*Composite*, *Machine*, *Process*, *and Component*) may be very useful to guide the security process. In a configuration, SMSA determines two types of connections:

- *Attachment:* a communication link between a port of a component and a role of a connector. A component needs a minimum of a connector to communicate with another component; however it can use more than one connector according the complexity of the security task.
- *Delegation*: a communication link between a port of a primitive component and a component of a composite port of the same type. An atomic component communicates only through its extern composite. So, SMSA approach supports implicit delegation in their configuration.

The particularity of our ADL is that it is dedicated to the structural description of the architecture at different hierarchies of abstract spaces via the concept of domain.

2.4. Domain

An important element in our approach is the concept of domain. A domain is direct support of distribution. It defines

Please cite this article in press as: M. Derdour, et al., Security architecture metamodel for Model Driven security, Journal of Innovation in Digital Ecosystems (2015), http://dx.doi.org/10.1016/j.jides.2015.12.001

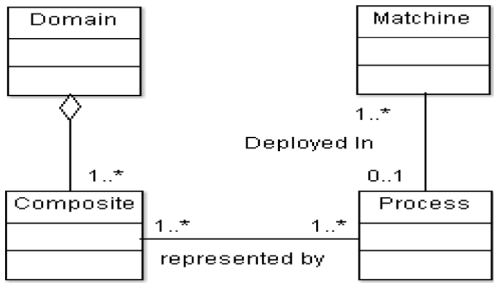


Fig. 2 – Deployment metamodel.

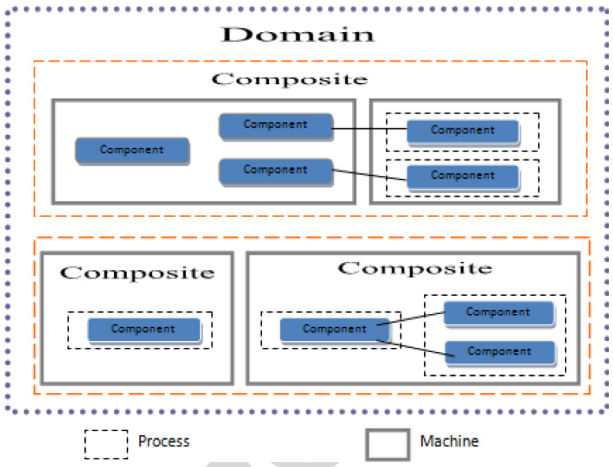


Fig. 3 – Example of domain.

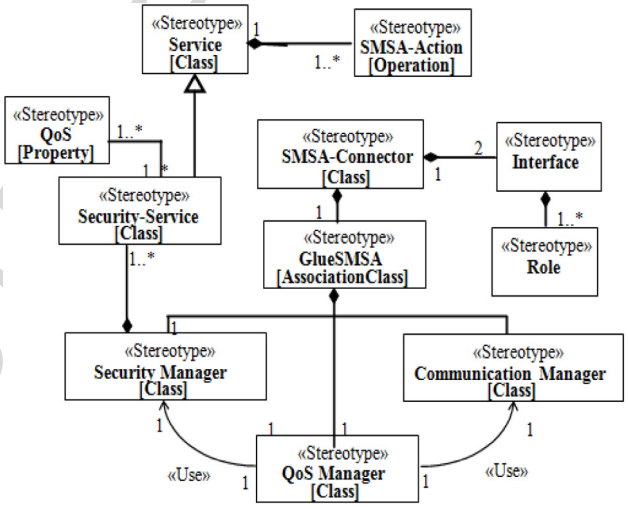


Fig. 4 – Description of SMSA connector.

the distribution of components in several geographically remote sites. A domain can contain one or more composite and one or more machines, each with the process in a system. All components are running in one or more processes (Fig. 2).

To illustrate the concept of the field and the dispersion of components in composites, machine and processes, we present an example (Fig. 3) which describes an area with three composites and three machines. The composite consists of five components distributed over three processes in two different machines. The other composite, shown in the lower part, performs all components on three processes, dividing them into two separate machines. Communications between its different components, whether in-process, inter-process, or intermachine [11]. The concept of *domain* is used in our approach, which gives us an assembly structure wider than the composite; it also provides multiple spaces of abstraction. These areas include them when the components depending on its location and related to other components. In other words, the concept of area providing perimeters of cooperation between component and allows the detection of points that need to take account of security. The concept of *domain* at brought much for SMSA, especially in the choice of security connectors that allow the consideration of environment's constraints when designing the application architecture.

2.5. *SMSA connector taxonomy*

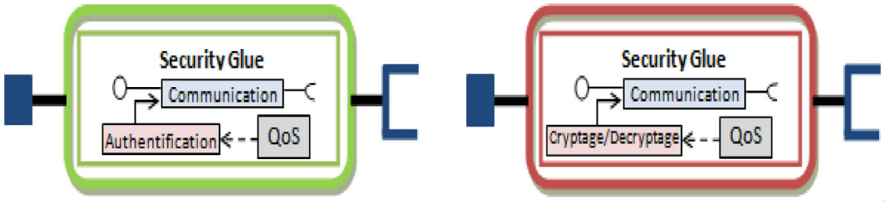
Several ADLs have been proposed. However, except for Ren [12] and xADL language [13], most ADLs not support security description of architectural elements. In addition, most of them are not formally defined. Compared the description to other ADLs [7,14], connectors can be composite or primitive as well as ensuring security services. Connectors are a description of the communication and the security among components.

Connectors in SMSA are first-class entities. The key role of connectors is to provide secure interactions between components. Connectors can be composite or primitive. Fig. 4 presents the Meta-model of our SMSA connector. The Connector is mainly specified by two interfaces and a glue specification. There are two types of interfaces: input and output. A glue specification defined three managers: *communications*, *security* and *quality of service*.

They manage the data transfer and the security among components. Connector interface required/provided consists of a set of roles. Each role provides the link between the connector and the component. Consequently two components can be linked by a connector, so that two connectors can be related together to create complex security task. We have also extended the glue by a security manager which cooperates with a QoS manager to ensure the security task. This security manager is a set of security services that cooperates to achieve security.

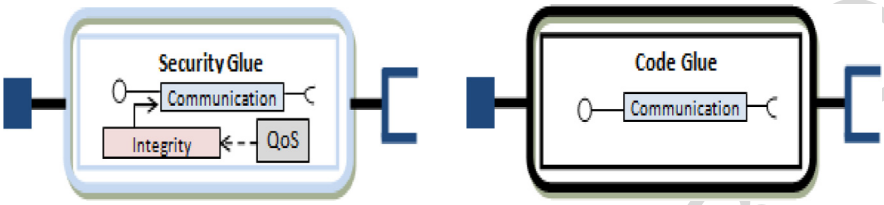
Three types of security aspects can be realized in software architectures: authentication, confidentiality and integrity. SMSA approach offers two services. The first one is to detect possible interaction points that require security. The second service allows a semantic integration of secure connectors between insecure components. There are two types of connectors: communication connectors and security connectors. Communication connectors are used to link two components that are encapsulated within a same process. Security connectors are used to express security interactions among components. We distinguish three types: Authentication, Confidentiality and Integrity.

Authentication connector: Typically is installed between two components have same composite and running on two



(a) Structure of authentication connector.

(b) Structure of confidentiality connector.



(c) Structure of integrity connector.

(d) Structure of simple connector.

Fig. 5 – Taxonomy of connectors.

different machines. Authentication connector is composed of three managers: the communication manager, the QoS manager and the security service manager that proofs the data subject's identity and ensures compliance with data protection.

Confidentiality connector: assure that the information is shared only among authorized components that share the same privileges and access rights. Usually is installed between two components located in two different processes or between a DATA component and a PRESENTATION/PROCESS component. This connector involves the implementation of encryption/decryption service of exchanging data flow.

Integrity connector: Integrity Connector provides fingerprint services such as MD5 to protect against malicious persons or software. Usually is installed between two components have different composites.

Simple connector: Usually is installed between two components have the same process at the same machine. It provides a link between two components and consist a simple communication (see Fig. 5).

3. Proposed transformations

Security is a principal consideration when designing, implementing and managing communications and information exchanges. To meet security of software components, we must include security considerations at a high level of design. Security can be considered from different views, e.g. security on the level of the user presentation interface-GUI, security on the level of the network and security on the level of the process level. At the GUI layer, security problems occur when the presentation component is not enabled for visual sensitive data filters over unknown persons.

Several services attacks that occurs at the network layer, when a malicious software agents joins the network; it aims and targets the information identity in such a way that it will update the flow of data traffic. At the process layer, it is possible to connect process component with un-trusted data component. Managing these problems at an architectural level provides developers with security mechanisms to guide in security development process.

3.1. *Vulnerability point detection*

To ease vulnerability point detection and to automatic integrate more accurately a secure connector, we have included components into processes, processes into machines, machines into composites and composites into domains and assignment of graphical notation with different colors to each of them. In this way, SMSA makes the detection of vulnerability points that requires security easier and automatic. Its visually identified by different colors assigned to each container types. Table 1 serves as guide that contains some security directives for solving the detected vulnerability points. The detection is done automatically by the checking of the constraints of container and colors. For instance, a JAVA component needs to communicate with SQL DataBase and NoSQL un-trusted data items, which are a composite running into another machine of JAVA component. Security connectors can be used to provide non-functional concerns of these components (e.g. confidentiality and authenticated access right).

3.2. *Architectural transformations*

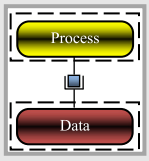
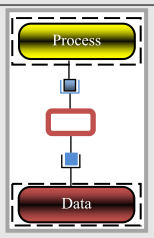
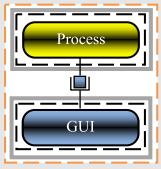
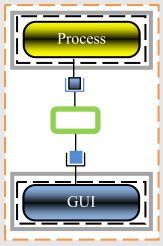
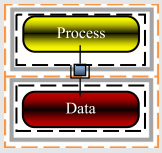
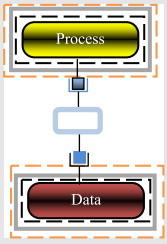
Once the vulnerability points are detected the transformation can be implemented. In order to better support security design and to better reduce design efforts, we propose a set of architectural transformations to integrate secure connectors types with different security strategies and QoS. We start with a global architecture without secure connectors and then we include our proposed secure connectors.

Three transformations according components' container are proposed to guide designer to securely system at an architectural level. These transformations are: *Confidential connector at Provided/Required Ports*, *Integrity connector at Provided/Required Ports*, and *Authentication connector at Provided/Required Ports*.

Please cite this article in press as: M. Derdour, et al., Security architecture metamodel for Model Driven security, Journal of Innovation in Digital Ecosystems (2015), http://dx.doi.org/10.1016/j.jides.2015.12.001

- *Confidential connector at Provided/Required Ports*: This transformation place the confidentiality connector between two components in two different processes or before a data component at architectural level. The preconditions of the transformation are the existence of components of different processes. Specifically, the confidentiality connector is required to secure communication only between two components to their own confidential properties over an open networked environment. In order to change the parameters of security services to provide adequate quality to component needs, the QoS manager controls the security manager in its work at runtime. It's possible to combine several connectors which implements hierarchical information based encryption.
- *Authentication connector at Provided/Required Ports*: Several malicious nodes are present to perform its malicious activities (e.g. update data identity) at the network level. If we use same composite that can host both components between two different machines, then the interaction point requires the authentication transmission of data. The preconditions of the transformation are two components which have different machines for the same composite.
- *Integrity connector at Provided/Required Ports:* Various services invocations among components of different composites require integrity properties. It's possible to place integrity connector between components that implements data flow interceptors for fingerprinting and signing. Such a transformation helps maintain consistency with corresponding composites and contributes to a compliant system implementation.

Table 1 – Model transformations.

Container	Before transformation	After transformation
Process		
Machine		
Composite		

4. SMSA UML profile

The primordial interest of defining a UML 2.0 profile for SMSA is to represent SMSA concepts using the UML 2.0 notations and therefore to formally model SMSA software architecture and for the long run to integrate software architecture in the framework MDA (*Model Driven Architecture*), which unifies all modeling approaches. The use of stereotypes, constraints and tagged values permit to capture the semantics of SMSAs architectural concepts. Thus, the advantages provided by the UML 2.0 profile permit to define a complete specification to structure SMSA software architecture and to achieve the mapping of SMSAs architectural concepts into UML 2.0.

We define the security aspects of the SMSA meta-model using UML 2.0 profile. The UML 2.0 profile provides a rigorous verification of architectural elements security. Each service is provided by component (i.e. configuration) and its global security is provided by UML 2.0 SMSA connectors. We decide to use the UML standard metamodel in order to profit from its advantages:

- To profit from *the precise semantics of UML notations* and its *powerful model abstraction* for describing more stringent and robust security mechanisms of the entire system;
- To profit from *variety of UML tools*, Eclipse, NetBeans, this aims to *describe the concepts at the top level* and the initial glossary (for easy communication);
- To profit from variety of UML tools, for designing security properties meet our needs, then composing and deploying robust services for IT applications;
- To profit from *OCL to check the consistency of the SMSA model* with the semantic constraints defined by the profile and OCL language is useful for architecture revision in case of inconsistency.

To specify this profile, we adopted the common rules described in [3]. They define the following technical aspects:

- Identification of the UML subset for the introduction of new construction;
- Description of stereotypes and tagged values introduced;
- Description semantics of these new buildings;
- Description of usage constraints: constraints are expressed in OCL; OCL expresses constraints on the known UML elements (e.g. component, classes, attributes, and associations). This section is devoted to the technical definition of UML-profile for SMSA metamodel. Such a profile includes a set of stereotypes and a set of OCL constraints applied on UML2.0 meta-classes. The UML profile for the SMSA description language is based on four packages (SMSA components package, SMSA interfaces package, SMSA security Package, SMSA Composition Package) detailed as follows:

4.1. *SMSA components package*

This package provides support to represent the functional part of component regardless of their environment. The

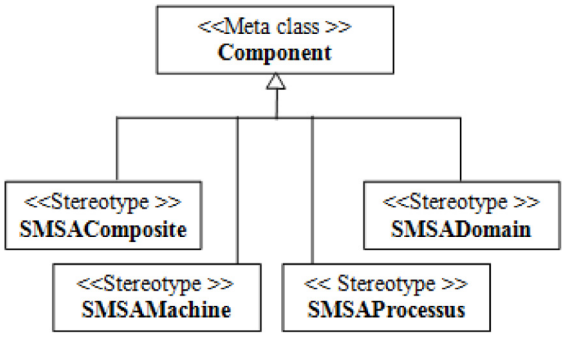


Fig. 6 – SMSA components package.

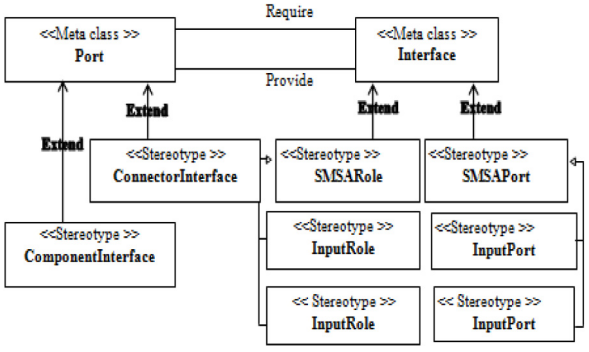


Fig. 7 – SMSA interfaces package.

most important concept of this package is the stereotype SMSA-Component. In SMSA, component is described as a UML class stereotyped "SMSA-Component" these bodies are similar to instances of UML component. SMSA component may provide services through ports connected to the interface input/output. This component has multiple types stereotyped as "Presentation", "Process" and "Data" (see Fig. 6).

We have added tagged-values to capture "*SMSA-Component*" semantics and to distinguish between component types. We have also defined the value of each tagged value related to each component type. The SMSA component must have at least one interface component. This constraint can be described in OCL as follows:

```
Context Component
Invariant: self.isStereotyped("SMSAComponent")
implies
  (self.ownedPort->size()>=2) and
  (self.ownedOperation->IsEmpty()) and
  (self.ownedPort->
   forAll(p|p.isStereotyped("ComponentInterface")) and
    (self.taggedvalue->select (tag |tag.name = "Type" and
        (tag.value= "Presentation" or tag.value ="Process"
          or tag.value ="Data")))
```

4.2. *SMSA interfaces package*

In our metamodel, the package interface defines two types of interfaces: *component interface* and *connector interface* which are extensions of the port class of UML and are stereotyped

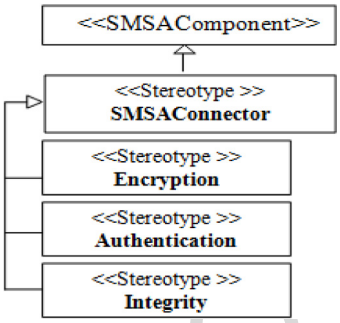


Fig. 8 – SMSA connectors package.

"Component-Interface" and "Connector-Interface", indicating that the constraints on the relation-port interface in UML is not the same in the SMSA metamodel (see Fig. 7).

In SMSA, component interface has a set of *Input/output* ports. A UML Port which has multiple interfaces (provided or required) and supports bidirectional communication, matches SMSA interfaces. SMSA components interface must have at least a port stereotyped "*Input-Port*" or "*Output-Port*". This constraint can be described in OCL as follows:

```
Context Port
Invariant: self.isStereotyped("ComponentInterface")
implies
  (self.owner.isStereotyped("SMSAComponent")) and
  (self.ownedOperation->IsEmpty()) and
  (self.required->size() >= 1 or self.provided->size() >= 1)
  and (self.required.isStereotyped("Input") and
  (self.provided.isStereotyped("Output")
```

We distinguish in the metamodel SMSA two types of interactions points: *input port* and *output port*. Each service required (provided) by a component must be expressed by input port (output port) of its corresponding required (provided) services. The class *Port* of UML represents SMSA ports in the UML metamodel 2.0 and each one is associated with a stereotype.

4.3. *SMSA security package*

4.3.1. *SMSA connectors package*

Components and connectors in SMSA have the same level of abstraction and are explicitly defined. Thus, we include in the UML profile two stereotypes: a stereotype to represent the concept of component "*SMSAComponent*" corresponding to the component class of meta-metamodel and UML stereotype representing the concept of connector "*SMSA-Connector*" corresponding to the meta-class Class UML metamodel (see Fig. 8).

A security connector is a mediator between two heterogeneous components or component and a connector that does not have same SMSA interface. A UML class, which has at least two interfaces (provided and required), and class "Security-Glue" matches SMSA connector. We have added a tagged-value *Security-Type* that allows the distinction between different security connector types.

Please cite this article in press as: M. Derdour, et al., Security architecture metamodel for Model Driven security, Journal of Innovation in Digital Ecosystems (2015), http://dx.doi.org/10.1016/j.jides.2015.12.001

The designer can specify security properties which can be an expression which refers to additional constraints and restrictions. These constraints are expressed in OCL as follows:

```
Context Class
Invariant : self.isStereotyped("SMSAConnector")
implies
  (self.ownedPort->size ()>=1) and
  (self.ownedPort->select(p|p.isStereotyped
      ("ConnectorRole"))-> size()>=2) and
  (self.nestedClassifier->select(m| m.oclIsTypeOf(Class)) ->
   forAll(g.isStereotyped("SecurityGlue")))-> size()=1) and
  (self.clientDependency.target->forAll->
   (t|t.oclIsKindOf(Interface))) ->isEmpty())  and
     (self.taggedvalue->select (tag.name = "TypeSecurity" and
      if (tag.value = "Encryption" ) then
      implies
       (self.connection-> exists(isStereotyped("SMSAComponent")
       and self.tagged="Data") and
       (self.connection-> exists(isStereotyped("SMSAComponent")
       and self.tagged="Presentation") or
       (self.connection-> exists(isStereotyped("SMSAComponent")
       and self.tagged="Process")))
```

4.3.2. *SMSA connector interface*

A connector interface contains a set of roles. They provide connection points among components. Roles interfaces have security services that guarantee data securities of the component with which they are associated. There are two types of roles: required role (or *InputRole*) and provided role (or *OutputRole*). SMSA Role only supports one-way communication. SMSA role can be used only in one oriented direction (provided/required).

```
Context Interface
Invariant : self.isStereotyped("ConnectorInterface")
implies
  (self.owner.isStereotyped("MMSAConnector")) and
  (self.required->size() = 1 or self.provided->size() = 1) and
  (self.required->forAll(p|p.isStereotyped("InputRole")) and
  (self.provided->forAll(p|p.isStereotyped("OutputRole" ))
```

4.3.3. *SMSA security glue*

A glue specification define a connector's behaviour: is a way in which to receive data on certain roles, secures them according to three security techniques (*authentication*, *confidentiality and integrity*) and produces on those roles. The Glue indicates how the behavior of the roles corresponds to ensure a complete interaction. Likewise, they define three managers: communication manager, security manager and QoS manager work together to ensure the interaction between components. The SMSA glue concept is relative to the UML Class in which it provides communication between components, but it remains defining its semantics with the following OCL constraint:

```
Context Connector
Invariant: self.isStereotyped("SecurityGlue")
implies
  (self.owner.isStereotyped("SMSAConnector")) and
  (self. nestedClassifier ->select(m| m.oclIsKindOf(Class)) ->
   select(isStereotyped("ComunicatMng "))-> size()=1)) and
   (self. nestedClassifier ->select(m| m.oclIsKindOf(Class))->
    select(isStereotyped("SecurityMng "))-> size()=1)) and
   (self. nestedClassifier ->select(m| m.oclIsKindOf(Class)) ->
    select(isStereotyped("QoSMng "))-> size()=1))
```

4.3.4. *SMSA attachment*

Attachments define the link between two roles or between a provided port (or a required role) and a required role (or a provided role). A UML assembly connector corresponds to the SMSA concept Attachment. This constraint is expressed in OCL as follows:

```
Context Connector
Invariant : self.isStereotyped("SMSAAttachment" )
implies
  (self.kind=#assembly) and (self.memberEnd.type ->
    forAll(m|m.oclIsKindOf(Interface))) and
   (self.end->(exists(cp1,cp2|cp1.name <> cp2.name and
    ((self.end->select(isStereotyped("InputRole"))->size >=1) and
     self.end->select(isStereotyped("OutputRole"))->size >=1))
```

4.3.5. *SMSA delegation*

Delegations define the link between ports of the same type (*required/provided*) of a component and its container (*composite*). Delegation allows of related interface components made of composite with the interface of this last. This constraint is expressed in OCL as follows:

```
Context Connector
inv: self.isStereotyped("SMSADelegation")
implies
  (self.kind=#assembly) and
  (self.memberEnd.type ->
    forAll(m|m.oclIsKindOf(Interface))) and
    (self.end->(exists(cp1,cp2|cp1.name <> cp2.name and
     ((self.end->select(isStereotyped("InputPort"))->size >=1) and
     self.end->select(isStereotyped("InputPort"))->size >=1)) or
      ((self.end->select(isStereotyped("OutputPort"))->size >=1)
     and  self.end->select(isStereotyped("OutputPort"))->size >=1))
```

4.4. *SMSA composition package*

In SMSA, there are four types of composition that encapsulate different architectural elements together: *process to compose components*, *machine to encapsulate processes*, *composite to encapsulate machines and domains to encapsulate composites*. We consider these compositions as special types of UML Class (Fig. 9).

Please cite this article in press as: M. Derdour, et al., Security architecture metamodel for Model Driven security, Journal of Innovation in Digital Ecosystems (2015), http://dx.doi.org/10.1016/j.jides.2015.12.001

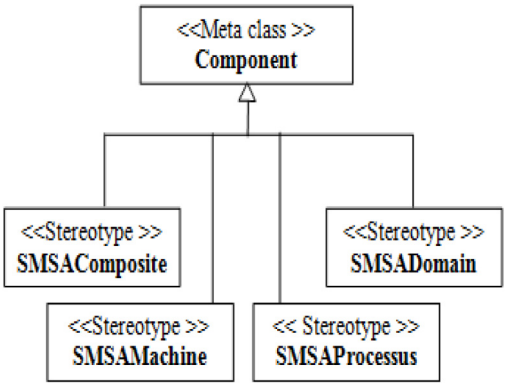


Fig. 9 – Composition package.

4.4.1. *SMSA composite*

An important aspect of the SMSA architecture is the composite as graph of components and connectors. As a UML component can contain sub-components and sub-classes, the composite SMSA are mapped to a graph with SMSA machine following this constraint:

```
Context Component
Invariant : self.isStereotyped("SMSAComposite")
implies
  (self.ownedPort->size()=1)and
  (self.ownedOperation->isEmpty()) and
    (self.ownedPort->
      forAll (p|p.isStereotyped("ComponentInterface"))) and
  (self.member->select(m|m.oclIsKindOf(Component))->
    forAll ->(c|c.isStereotyped("SMSAMachine")))->size()>=1)
```

4.4.2. *SMSA machine*

In our approach, applications are modeled as distributed system consisting of a set of computing machines deployed in different locations in a target environment. On each SMSA machine, there are several SMSA processes available for computing and several connectors available for communication and safety. This semantic feature is described in OCL as follows:

```
Context Component
Invariant: self.isStereotyped("SMSAMachine")
implies
  (self.ownedPort->size()=1) and
  (self.ownedOperation->isEmpty()) and
  (self.ownedPort->
    forAll (p|p.isStereotyped("ComponentInterface"))) and
  (self.member->select(m|m.oclIsKindOf(Component))->
    forAll ->(c|c.isStereotyped("SMSAProcessus")))->size()>=1)
```

4.4.3. *SMSA process*

SMSA processes are abstractions that include primitive components interconnected together by connectors (*simple or secure*). Since a UML component can contain subcomponents and subclasses, SMSA processes are mapped into UML components with the following constraints:

```
Context Component
Invariant: self.isStereotyped("SMSAProcessus")
implies
  (self.ownedPort->size()=1) and
  (self.ownedOperation->isEmpty()) and
  (self.ownedPort->
  forAll(isStereotyped("ComponentInterface"))) and
  (self.member->select(m|m.oclIsKindOf(Component) ->
   forAll (isStereotyped("SMSAComponent")))->size()>=1) and
  (self.member->select(m|m.oclIsTypeOf(Class))->forAll->
    (c|c.isStereotyped("SMSAConnector")))->size()>=1)
```

5. Validation industrial case study

5.1. *The SAGE system*

In our study, our metamodel was used in the process of developing a phosphate support system (SAGE) for the company FERPHOS, i.e. PIS (*FERPHOS Information System*). The SAGE system covers three activities: human resource management and pay provider, formations, invoice and finances provider. The SAGE system is ease management of 300 workers in the society. Goal of the company FERPHOS security is to allow a user with known identity and correct access rights to manipulate the SAGE system. The SAGE system is modeled as a components diagram in UML 2.0 consisting of four applications, each application as a components sets. These applications deployed on different devices in a target environment, connected by wireless or fixed line communication networks.

Basically, a main component (*e.g. Consolidate GUI*) receives its results from a Treatment Consolidate component with itself receives its data from all database components (*e.g. Consolidate DataBase, personal and human resource DataBase, Immobilier DataBase, Invoice and Finance DataBase*). Simple connectors are used between those components to exchange data and informations.

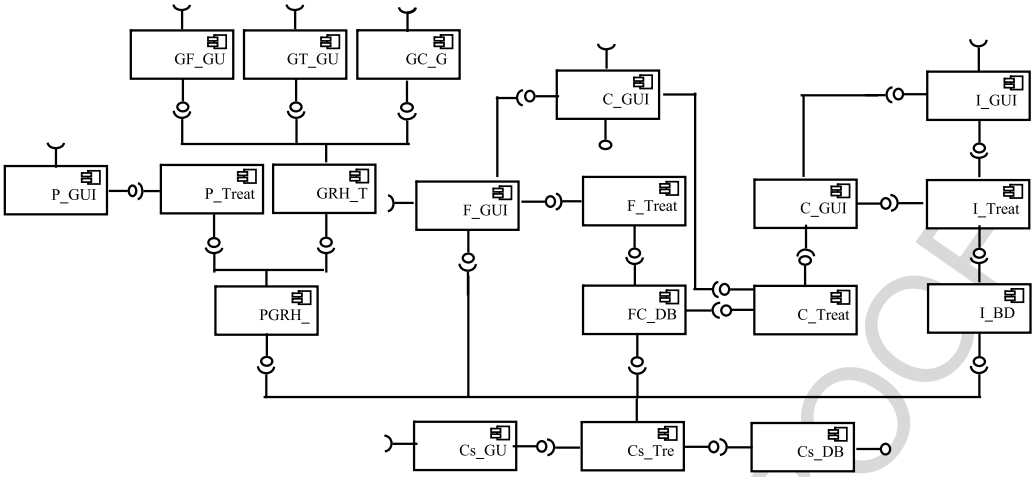
We have proposed a UML diagram corresponding to the SAGE system illustrated in Fig. 10. Table 2 shows an overview of the SAGE system in SMSA.

The security goals of the company FERPHOS are summarized as follows:

- **Prevention**: all of 19 components of the SAGE system should be compliant with the occurrence of unwanted security problems.
- **Correction**: an integrated security management process and strategies could protect each service of the SAGE system related to each component.
- **Analysis**: requirements of security goals should be analysed with the consideration of FERPHOS-specific characteristics.
- **Detection**: identification of vulnerability events during the design phase or after they are occurred.

5.2. *Architecture modelling and transformations*

It's essential to consider security aspects of the company FERPHOS at a high level of design, the SMSA metamodel is practically significant as well related to security aspects of SAGE system. We start with a global architecture without



I: Invoice, C: Accountancy, F: Finance, P: pay, RH: human resource, G: Management, GT: G. Time, GF: G. Formation, GC: G. Comportment, Cs: Consolidate.

Fig. 10 – SAGE system in UML 2.0.

Table 2 – Containers of components.

Application and services	*SMSA components types*		
	Process	**Data**	**Presentation**
HR management			
Service:			
• HR affectation • Needs analysis • Personal selection • HR planning • Formation planning • Time map design	GRH_Treat	PGRH_DB	GC_GUI GT_GUI GF_GUI
Pay application			
Service:			
– Payment – Salary augmentation	Pay_Treat		P_GUI
Finance application			
Services:			
• finance planning • Cash prediction • Needs analysis • Success/failure analysis	F_Treat C_Treat I_Treat I_GUI	FC_DB I_DB	F_GUI C_GUI
Consolidation			
Service:			
• Finance planning • Cash prediction • HR planning	Cs_Treat	Cs_DB	Cs_GUI

secure connectors and then we include our proposed secure connectors.

Later in Section 5.3 (see Section 5.3), we will illustrate how our strategy of mapping can be used; we apply it to the SAGE system. Fig. 10 illustrates the description of the system using SMSA. Fig. 11, shows the architecture in UML 2.0 after applying the profile.

The security goals of the company FERPHOS are summarized as follows:

– First, the SAGE architecture was modeled using SMSA using a set of components types (*e.g. data, process and presentation*). Each component type belongs to a colors class. For example, in Fig. 11, bleu color can be used for presentation components: *Personal GUI* and *Pay GUI* respectively. Fig. 10 shows a SAGE domain with 10 machines and 19 components (e.g. in *the category Data* we find 4 *databases*, *in the category Process* we find 6 *components* and *in the category Presentation* we *find* 9 *components*);

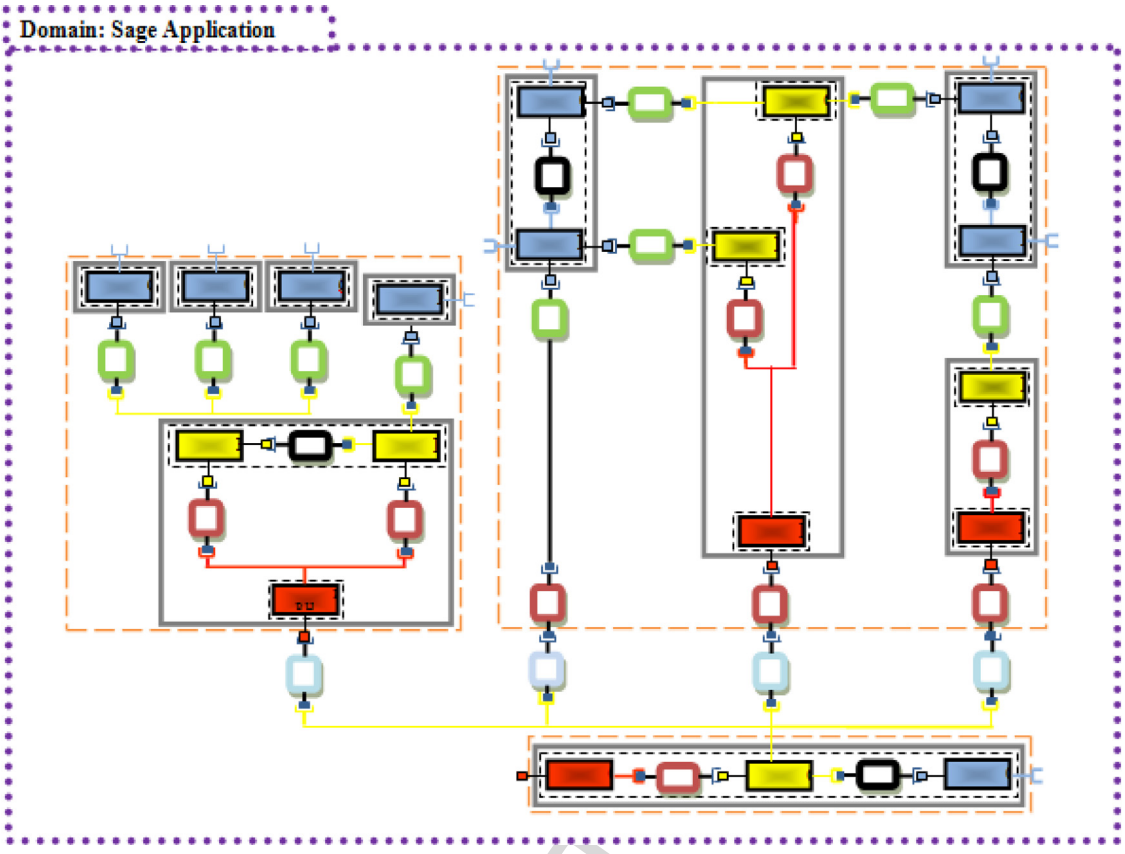


Fig. 11 – Description of the SAGE application in SMSA. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

- We define processes to include *k* components sets for different applications, processes set deployed into machines, machines into composites and composites into a SAGE domain. Different colors are assigned to each of them related to container types of the SAGE system. For example, in Fig. 10, on the top-left, 4 bleu components that run in 4 processes on 4 machines, on the bottom-left, one finds 2 yellow presentations in one processes and a single data in another process but on the same machine;
- Finally, the designer finishes the encapsulation when all components are settled, leaving the component architecture as the SAGE domain.

The uses of graphic notations make the detection of vulnerability points that requires security easier and automatic. For each container of the SAGE components, if it is described by different processes, the interaction point among components requires the confidentiality connector. But if we use same composite that can host both components, then the interaction point among components requires authentication connector. In the context of integrating SMSA connector types, it is necessary to respect the structural and semantic features of SMSA that mean for:

- **Authentication connector**: we integrate the authentication connector explicitly between two machines in the same composite.
- **Confidentiality connector**: we integrate the confidentiality connector explicitly between two components in two different processes or before a data component.
- **Integrity connector**: we integrate the integrity connector explicitly between two composites.

For example, in Fig. 10, authentication connector C_1 used for P_1 and P_2 respectively. In order to better represent SMSA connectors with respect of some criteria such as visual clarity, it is essential to well distinguish between connector types by colors assigned to each one of them. This is the motivation for our security sentient integration strategy. The set of security connectors provided in Table 3 make the 19 SAGE related components secure.

5.3. *SMSA-UML 2.0 visual plug-in*

We have implemented the SMSA metamodel in IBM Rational Software Modeler for Eclipse 3.1. In this section, we show how we build up a mapping environment, the consideration for integration SMSA concepts into UML 2.0 and present the evaluation results for SMSA concepts. With SMSA-UML mapping environment, the following features may be performed on the modeled system:

- Checking the structural coherency of a given system and to validate its semantics with SMSA approach.
- Providing an easy way to describe complex software architectures in one easy-to-use visual editor and diagramming facilities.
- Deriving architectural security constraints form security requirements.

Please cite this article in press as: M. Derdour, et al., Security architecture metamodel for Model Driven security, Journal of Innovation in Digital Ecosystems (2015), http://dx.doi.org/10.1016/j.jides.2015.12.001

Table 3 – Security connectors for SAGE.

Container type	SAGE elements	Connector type
Same process	PRH_DB RH_Treat P_Treat	Integrity connector
Different processes	P_GUI P_Treat	Confidentiality connector
Same machine		
Different machines	PRH_DB, FC_DB, I_DB Cs_Treat	Authentication connector
Same composite with different processes	I_GUI I_Treat	Integrity connector
Different composites	PRH_DB Cs_Treat	Authentication connector
Different machines, processes and composites	F_GUI	Composition of authentication, confidentiality and integrity connectors

- Implementing most architectural security concepts (data ports, user defined connectors, structures such as configurations of complex components and complex connectors).
- The detection of heterogeneity is done automatically by checking of the constraints of formats and data type.
- Providing a more suitable representation of security connectors which are defined at the meta-level (Class concept of UML 2.0) rather than using a simple attributes for this purpose.

Fig. 12 shows the final mapped of the SAGE system in UML 2.0 after applying the profile. The model is tested and validated with the semantic constraints defined by the profile.

5.4. *Comparison and lessons learned*

For evaluation of our approach, we designed more complicated systems with/without model transformations (see Section 3.2). We have performed these experiments on a Laptop running Windows 7 (x64) with 6 GB of RAM and i7-2630QM quadruple coreprocessor (2 GHz). The architect can use our graphical tool provides to compare the performance and security risk values for various architectures. We provide easy and quick access to the required security connectors.

After some performance tests, we concluded that SMSA UML profile constraints execution gets alarmingly slow when the system grows in components instances as can be seen in Table 4. Once this problem was detected, we decided to use our JAVA implementation to integrate security connectors. After determination of the components container, the system integrates required security connectors. This made the checking time considerably faster by applying each rule separately. Moreover, the execution time remains constant at any model size. This experiment enables architects to gain insights into performance and security tradeoffs in their architectures.

Our approach currently gives only an answer of vulnerability point's detection architectures retrieved from structural contexts. It does not identify the behavior source of vulnerability attacks as the result of a virus.

Table 4 – Performance results.

Case 1: Secure electronic transaction system (components size = 4)	
OCL rules without model transformation	Elapsed time: 989 ms
OCL rules with model transformation	Elapsed time: 427 ms
Case 2: Secure client/server system (components size = 7)	
OCL rules without model transformation	Elapsed time: 9975 ms
OCL rules with model transformation	Elapsed time: 1487 ms
Case 3: Parking access control system (components size = 11)	
OCL rules without model transformation	Elapsed time: 12 613 ms
OCL rules with model transformation	Elapsed time: 2613 ms

6. Related work

In modern applications, security is always a major concern and has been well studied. Earlier works [1] has focused on improving security on quality development process, whereas more recent works have suggested that the security strategies for the whole system is actually more important to generate more stringent and robust systems [15–18].

A common strategy [7,19–26] to achieve ease security integration is to use of well-known languages (ADLs, UML) and a clear separation between functional concerns from non-functional concerns of a system. We mainly distinguish two categories of approach: Component-Based Software Engineering (CBSE) and Service-Oriented Architecture (SOA). In the first case [7,22,23,20] focus on the static structure of the system: the software elements are components assembled by connectors in configurations. Whereas in the second case [25, 23,20,19] focus on the functional structure of the system: the software elements are functionalities (services) linked by relations of collaboration or combination.

Modern applications are more and more developed according to ADL-based development processes [21]. It proposes security analysis and verification of security properties (e.g. availability, confidentiality and integrity) early at architecture level while meeting the system requirements on the number of components for each service at the design

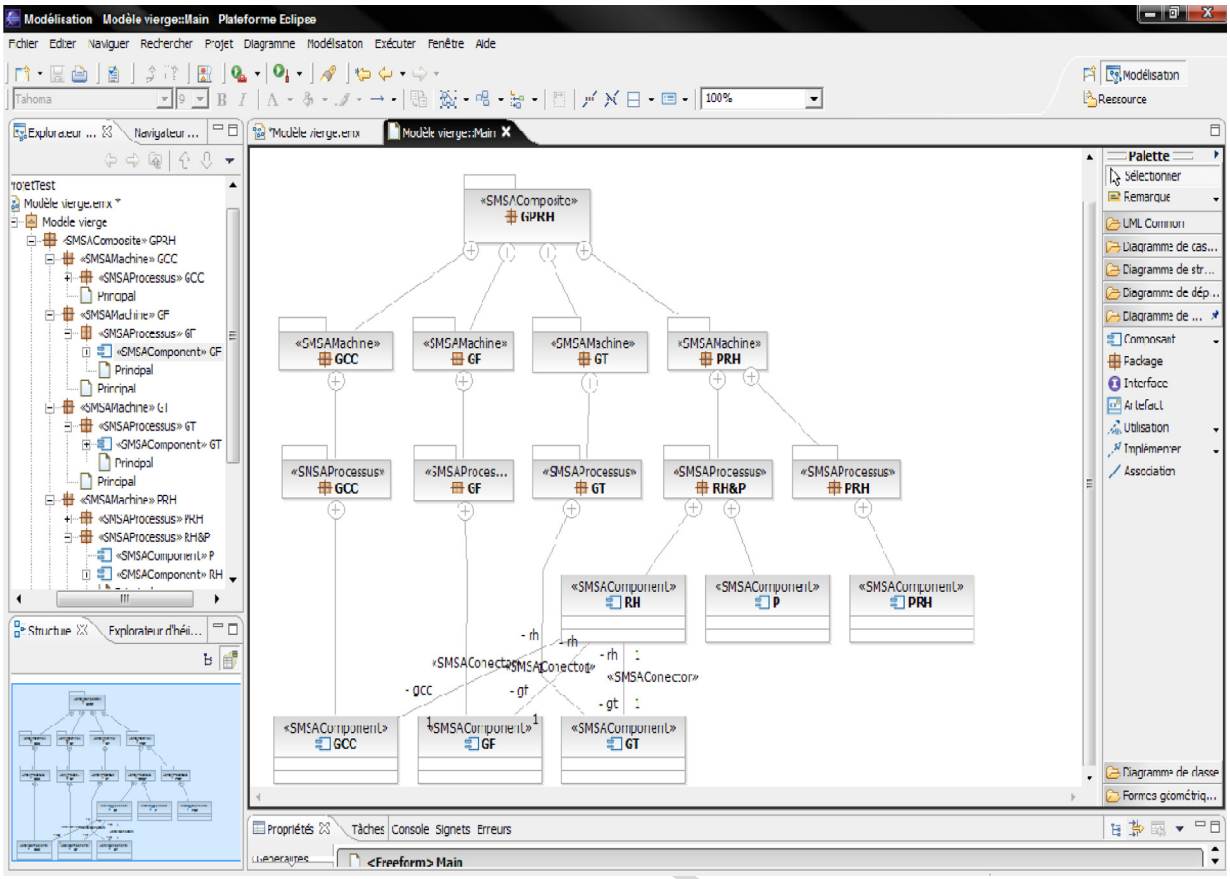


Fig. 12 – SAGE system in UML 2.0 after transformation.

time. ADLs can be classified in three different categories [27]: ADL without connector, ADL with a set of predefined connectors and ADL with explicit connector types. Several ADLs consider connectors as first-class entities such as: Wright [7], ACME C2 [8], XADL [13], AADL [28], etc. These ADLs points a major impact on how architecture is practiced and how components and connectors are reused. In our approach, we support the definition of security connector types explicitly and independently of any particular use of them and to support them with the ability to specialize them in which system is being used are changing in significant ways.

Most of existing ADLs such as SPT-UML [29], MARTE [24], Fractal [30], SCA [31], Kmelia [20] and AADL [32] do not support security description of data flow among components to their locations during the design phase. In addition, most of them are not formally defined. However, except for architecture meta-models proposed by Marcel and al. [33,34] which support security entities updates and security services at model level. Another approach, C3 (Component Connector Configuration) [27] proposes taxonomy of connectors with better visual clarity, namely logic-based and physic-based; to automatic generate physic architecture for each application instance and to fulfil various connections among components. It supports three levels modeling defined by the OMG [24]. But this work supports neither security connectors nor security transformations out.

In [35], authors discussed how to use UML to specify authorization constraints and to specify access control-based informations. It can be exploited for generating architectures instances playing security roles. Nevertheless, since OCL is an important factor of how analysis security properties of a system. Basin et al. [36] studied the OCL oriented mapping. Compared to our work, our approach offers a very high level modeling and considers several security concepts (e.g. security connectors, composite, domain, and service security).

[36] propose a components diagram in UML 2.0 for describing port types (required or provided) of the system and then exploit SysML to define data flow direction. Its approach allows a well description of different interfaces, but disagrees in the more integration of explicit security connectors available in SMSA. Other approaches [37,38] consider a data flow in embedded systems with security mechanisms as an important feature of how secure system could be. Previous works done by Menzel et al. [39,40] rely on high-level metamodel to define service-oriented security intensions such as trust relationships, identity provisioning, and confidentiality and how to transform them on UML using a set of stereotypes. This work not provides security connectors as mediator between heterogeneous services and does not provides a formal analysis to validate transformation process. [40] studies various security problems related to SOA environments and provides an optimal solution to integrate security at the design time for SOA applications using Model Driven Development.

Our SMSA Plug-in can be compared with similar architecture tools, such as AcmeStudio [41], COSABuilder [42] and

UML 2.0 Profile for C3 [27]. Indeed, these two applications allow graphical representation of architectures and automatic security constraints verification of models using OCL standard. The use of SMSA-Plugin offers number of advantages compared to these tools, including:

- The extension of existing UML profiles to include explicit security aspects such as security connector types. It enriches profiles with semantic constraints at a high level of abstraction accurate the generation of robust configuration;
- The automatic vulnerability point detection that solves all the problems of security through visual semantic rules defined on components containers (domain, machine, composite, and process);
- The component services which violate security properties filtered at a high level of abstraction, making our development process easier and robust;
- The visual design of system can be customized to make it more suitable for particular user security needs and a particular domain area;
- In summary, from previous comparison, we believe that is good for electronic commerce system and sensitive communications) that need to support security mapping, security policies and QoS management at architecture level.

7. Conclusion

This paper presents a generic metamodel for integrating and managing security flow-based IT applications on distributed environments. Our work proposes to integrate security connectors at a high level of design by using distribution concepts (*e.g. domain as a very important components assemblies and configurations*). Since security is often to be considered to be one of the most important concerns for IT applications, we have designed a transformation strategy from insecure system to secure system that will minimize the total design costs for secure communication. We detect vulnerability points of system at the architecture level, and define transformation rules to integrate security connector types for secure system architecture to make security services co-assemblies.

To profit from the advantages of SMSA including the explicit definition and support of security connectors, a direct transformation strategy from SMSA to UML 2.0 in needed. We define the UML 2.0 profile for SMSA that can be integrated in MDA. This profile contains a set of stereotypes which all tagged values and OCL constraints to grantee correct mapping of robust systems. Our contributions can be used as a support to guarantee security aspects for the numerical resources (DAM: Digital Asset Management) at architectural level. Such applications handle a wide variety of media, and communicate with users through various platforms (Smartphones, tablets, desktops, laptops, etc.. . .). SMSA can bring an effective solution to DAM development. Especially in parts of: acquisition, processing, distribution and content use. SMSA provides way to talk objectively about security problems of media contents. Their security connectors have many important properties: for instance it improves various security properties by managing QoS and reconfiguring connectors at the execution level.

Our future works will be the integration of security policies and QoS management and the integration of SMSA profile in the approach MDA (Model Driven Architecture) to ensure the automatism of the process of transformation

Uncited references

Q7

[43] and [44].

REFERENCES

[1] ISO/IEC 27000, Information technology and security techniques, 2014. http://www.iso.org/iso/.
[2] B. Issac, N. Israr, Case studies in secure computing, Achiev. Q8 Trends (2014).
[3] J. Magee, J. Kramer, Dynamic structure in software architectures, in; Proceedings of the Fourth ACM SIGSOFT Symposium on Foundations of Software Engineering, FSE'96, San Francisco, USA, 1996, pp. 3–14.
[4] D.C. Luckham, J.L. Kenney, L.M. Augustin, J. Vera, D. Bryan, W. Mann, Specification and analysis of system architecture using rapide, IEEE Trans. Softw. Eng. 21 (4) (1995) 336–355.
[5] N.R. Mehta, N. Medvidovic, S. Phadke, Towards a taxonomy of software connectors, in: ICSE'00, ACM Press, 2000, pp. 178–187.
[6] David A. Basin, J. Doser, T. Norderstedt, Model-driven security: From UML models to access control infrastructures, ACM Trans. Softw. Eng. Methodol. 15 (1) (2006) 39–91.
[7] R. Allen, D. Garlan, A formal basis for architectural connection, ACM Trans. Softw. Eng. Methodol. 6 (3) (1997) 213–249.
[8] D. Garlan, R.T. Monroe, D. Wile, Acme: Architectural description component-based systems, in: Foundations of Component-Based Systems, Cambridge University Press, 2000, pp. 47–68.
[9] M. Derdour, R. Roose, M. Dalmau, N. Ghoualmi Zine, A. Alti, MMSA: Metamodel multimedia software architecture, Adv. Multimed. 2010 (2010) 17. http://dx.doi.org/10.1155/2010/386035. Hindawi Ed., Article ID 386035.
[10] Society of Automotive Engineers, Architecture Analysis & Design Language, AADL, 2008.
[11] C. Szyperski, Component Software: Beyond Object-Oriented Programming, AddisonWesley Publishing Company, 1997.
[12] R. Roshandel, N. Medvidovic, Multi-view software component modeling for dependability, in: Architecting Dependable Systems II, in: Lecture Notes in Computer Science, (ISSN: 0302-9743), vol. 3069, (ISSN: 0302-9743), 2004, pp. 286–304.
[13] E. Dashofy, A.v.d. Hoek, R.N. Taylor, A comprehensive approach for the development of XML-based software architecture description languages, ACM Trans. Softw. Eng. Methodol. (TOSEM) 14 (2) (2005) 199–245.
[14] OMG, Unified Modeling Superstructure, 2006, 2000. http://www.omg.org/docs/ptc/06-04-02.pdf.
[15] G. Georg, I. Ray, R. France, Using aspects to design a secure system, in: The 8th International Conference on Engineering of Complex Computer Systems (ICECCS '02), pp. 117, Washington, DC, USA, 2002, IEEE Computer Society, 2002.

[16] J. Jurjens, Towards development of secure systems using UMLsec, in: Heinrich Hussmann (Ed.), Fundamental Approaches to Software Engineering (FASE/ETAPS 2001), in: LNCS, vol. 2029, Springer-Verlag, 2001, pp. 187–200.

[17] J. Jurjens, Jean-Marc Jézéquel, Heinrich Hussmann, Stephen Cook, UMLsec: Extending UML for secure systems development, in: UML 2002—The Unified Modeling Language, in: LNCS, vol. 2460, Springer-Verlag, 2002, pp. 412–425. Editors.

[18] Anneke Kleppe, Wim Bast, Jos B. Warmer, Andrew Watson, MDA Explained: The Model Driven Architecture–Practice and Promise, Addison-Wesley, 2003.

[19] B. El Asri, A. Kenzi, M. Nassar, A. Kriouile, Vers une architecture MVSOA pour la mise en œuvre des composants multi-vue, in: 3ème Conférence Francophone sur les Architectures Logicielles, CAL'2009, RNTI, 2009, pp 1–17.

[20] C. Attiogbé, P. André, M. Messabihi, Correction d'assemblages de composants impliquant des interfaces paramétrées, in: 3ème Conférence Francophone sur les Architectures Logicielles, CAL'2009, Hermès, 2009.

[21] P. Avgeriou, Uwe Zdun, Modeling architecture patterns using architecture primitives, in: The ACM SIGPLAN Conference on Systems, Programming, Languages and Applications OOPSLA'2005, Vol. 40, No. 10, 2005, pp. 133–146.

[22] K. Bergner, A. Rausch, M. Sihling, A formal model for component ware, in: Foundations of Component-Based Systems, Cambridge University Press, New York, 2000, pp. 189–210. Eds..

[23] N. Medvidovic, R.N.A Taylor, Classification and comparison framework for software architecture description languages, IEEE Trans. Softw. Eng. 26 (1) (2000) 70–93.

[24] OMG, Unified Modeling Language: Infrastructure, 2007. http://www.omg.org/docs/formal/07-02-06.pdf.

[25] Jie Ren, Richard N. Taylor, A secure software architecture description language, in: Workshop on Software Security Assurance Tools, Techniques, and Metrics, 2005.

[26] M. Shaw, R. DeLine, D.V. Klein, T.L. Ross, D.M. Young, G. Zelesnik, Abstractions for software architecture and tools to support them, IEEE Trans. Softw. Eng. 21 (4) (1995) 314–335.

[27] A. Amirat, M. Oussalah, First-class connectors to support systematic construction of Hierarchical software architecture, J. Object Technol. 8 (7) (2009) 107–130.

[28] R. Allen, S. Vestal, B. Lewis, D. Cornhill, Using an architecture description language for quantitative analysis of real-time systems, in: Proceedings of the Third International Workshop on Software and Performance, ACM Press, Rome, Italy, 2002, pp. 203–210.

[29] S. Graf, I. Ober, How useful is the UML real-time profile SPT without semantics, SIVOES 2004, associated with RTAS 2004, Toronto Canada, 2004.

[30] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, J.-B. Stefani, An open component model and its support in Java, in: I. Crnkovic, J.A. Stafford, H.W. Schmidt, K.C. Wallnau (Eds.), CBSE, in: Lecture Notes in Computer Science, vol. 3054, Springer, Berlin, 2004, pp. 7–22.

[31] G. Barber, What is SCA, 2007. http://www.oasis-opencsa.org/sca.

[32] L. Seinturier, P. Merle, D. Fournier, N. Dolet, V. Schiavoni, J.-B. Stefani, Reconfigurable SCA applications with the FraSCAti platform, in: IEEE International Conference on Services Computing, Bangalore, India, September 2009.

[33] C. Marcel, R. Michel, M. Christian, L. Calin, Costin, Adaptation dynamique de services, in: Déploiement et (Re) Configuration de Logiciels, DECOR'04, Grenoble, France, 2004. Q9

[34] C. Marcel, R. Michel, M. Christian, Autonomic adaptation based on service-context adequacy determination, J. Electron. Notes Theoret. Comput. Sci. (ENTCS) 189 (2007) 35–50. Elsevier;
E. Maximilien, M. Singh, Self-adjusting trust and selection for web services, in: IEEE Second International Conference on Autonomic Computing, ICAC'05, 2005, pp. 385–386. Q10

[35] D.A. Basin, M. Clavel, J. Doser, M. Egea, A metamodel-based approach for analyzing security-design models. in: MoDELS 2007, 2009.

[36] T. Lodderstedt, D. Basin, J. Doser, SecureUML: A UML-based modeling language for model-driven security, in: 5th International Conference, 2002.

[37] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Trans. Dependable Secure Comput. (2004) 11–33.

[38] S. Balsamo, M. Bernado, M. Simeoni, Performance evaluation at the architecture level, in: Formal Methods for Software Architectures, in: LNCS, vol. 2804, Springer, Berlin, Germany, 2003, pp. 207–258.

[39] R.B. Michal Hafner, Security Engineering for Service-Oriented Architectures, Springer-Verlag, Berlin Heidelberg, 2009.

[40] M.Q. Saleem, J. Jaafar, M.F. Hassan, Model driven security frameworks for addressing security problems of service oriented architecture, in: ITSim 2010, 2010.

[41] AcmeStudio, The ACME ADL Toolkit, 2004. http://www.cs.cmu.edu/~acme/AcmeStudio/index.html.

[42] A. Alti, A. Boukerram, A. Smeda, S. Maillard, M. Oussalah, COSABuilder and COSAInstantiator: An extensible tool for architectural description, Int. J. Softw. Eng. Knowl. Eng. (ISSN: 0218-1940) 20 (3) (2010) 423–455.

[43] M.M. Menzel, Security meta-model for service-oriented architectures. in: IEEE International Conference on Services Computing, SCC '09, 2009.

[44] M. Menzel, C. Meinel, SecureSOA modelling security requirements for service-oriented architectures. in: 2010 IEEE International Conference on Services Computing, SCC, 2010.

Please cite this article in press as: M. Derdour, et al., Security architecture metamodel for Model Driven security, Journal of Innovation in Digital Ecosystems (2015), http://dx.doi.org/10.1016/j.jides.2015.12.001