Information and Software Technology 55 (2013) 1651–1677

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Development of Secure XML Data Warehouses with QVT

B. Vela [a,*], J.N. Mazón [b,1], C. Blanco [c], E. Fernández-Medina [d], J. Trujillo [b], E. Marcos [a]

[a] Languages and Computing Systems, II Department, Rey Juan Carlos University, C/Tulipán s/n, 28933 Móstoles, Madrid, Spain
[b] Languages and Computing Systems, Department University of Alicante, C/San Vicente s/n, 03690 Alicante, Spain
[c] Department of Mathematics, Statistics and Computer Science Facultad de Ciencias, University of Cantabria, Av. de los Castros s/n, 39071 Santander, Spain
[d] Information Systems and Technologies, Department University of Castilla-La Mancha, Paseo de la Universidad, 4 – 13071 Ciudad Real, Spain

ARTICLE INFO

Article history:
Received 27 June 2012
Received in revised form 15 January 2013
Accepted 8 March 2013
Available online 20 March 2013

Keywords:
XML
Data warehouse
Security
MDA
QVT

ABSTRACT

Context: Data warehouses are systems which integrate heterogeneous sources to support the decision making process. Data from the Web is becoming increasingly more important as sources for these systems, which has motivated the extensive use of XML to facilitate data and metadata interchange among heterogeneous data sources from the Web and the data warehouse. However, the business information that data warehouses manage is highly sensitive and must, therefore, be carefully protected. Security is thus a key issue in the design of data warehouses, regardless of the implementation technology. It is important to note that the idiosyncrasy of the unstructured and semi-structured data requires particular security rules that have been specifically tailored to these systems in order to permit their particularities to be captured correctly. Unfortunately, although security issues have been considered in the development of traditional data warehouses, current research lacks approaches with which to consider security when the target platform is based on XML technology.
Objective: We shall focus on defining transformations to obtain a secure XML Schema from the conceptual multidimensional model of a data warehouse.
Method: We have first defined the rationale behind the transformation rules and how they have been developed in natural language, and we have then established them clearly and formally by using the QVT language. Finally, in order to validate our proposal we have carried out a case study.
Results: We have proposed an approach for the model driven development of Secure XML Data Warehouses, defining a set of QVT transformation rules.
Conclusion: The main benefit of our proposal is that it is possible to model security requirements together with the conceptual model of the data warehouse during the early stages of a project, and automatically obtain the corresponding implementation for XML.

1. Introduction

Data Warehouse (DW) systems [1,2] provide a Multidimensional (MD) [3] view of huge amounts of historical data from heterogeneous operational sources. These systems supply useful information which allows decision makers to improve business processes in organizations. The MD paradigm structures information into facts and dimensions. A fact contains the interesting measures (fact attributes) of a business process (sales, deliveries, etc.), whereas a dimension represents the context in which a fact is analyzed (product, customer, time, etc.) by means of hierarchically organized dimension attributes.

Traditional DW systems allow business people to acquire useful knowledge from their organizations' data by means of a variety of technologies, such as OLAP or data mining. However, if richer insights into the dynamics of today's business are to be provided, it is desirable to combine data from inside the organization with data from the outside, thus complementing company-internal data with value-adding information (e.g., retail prices of products sold by competitors). The fact that the amount of data available on the Web has been growing rapidly in the last decade signifies that Web data are more and more useful for this purpose. Furthermore, as is argued in [4], considering external data sources entails non-traditional data types such as Geographic Information Systems or data streams related to Business Process Monitoring, which will play a crucial role in the next generation of DWs. Dealing with these new data types therefore leads to the need for other design

* Corresponding author. Tel.: +34 91 488 70 03; fax: +34 91 488 85 58.
E-mail addresses: belen.vela@urjc.es (B. Vela), jnmazon@dlsi.ua.es (J.N. Mazón), Carlos.Blanco@unican.es (C. Blanco), Eduardo.FdezMedina@uclm.es (E. Fernández-Medina), jtrujillo@dlsi.ua.es (J. Trujillo), esperanza.marcos@urjc.es (E. Marcos).

[1] Jose-Norberto Mazón had developed this work during a research internship in the University of Castilla-La Mancha, funded by the "Consejería de Ciencia y Tecnología of the Junta de Comunidades de Castilla-La Mancha" and the European Social Fund under the contract 10/38-C.

0950-5849/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2013.03.003

and implementation approaches rather than the traditional ones based on relational technologies, such as XML.

1.1. Motivation and starting point

The main problem with external data is that it is rather heterogeneous and complex.

Interestingly, the designers of DW systems have overcome the aforementioned drawbacks by making use of XML technologies [5–7] in different ways [8]:

- XML has ameliorated the extraction and integration of heterogeneous Web data in the DW (see [4]).
- XML has provided a large data exchange framework within corporate information systems. Its flexibility and openness has led to the appearance of an increasing quantity of XML data on the Web, as outputs of e-commerce applications or simply as Web pages. This represents an important data source for decision support systems [8]. XML technology consequently helps to deal with unstructured data in DW systems.
- XML is used profusely in order to increase the level of interoperability between different kinds of data analysis tools and the DW [4].
- XML allows complex data to be dealt with [7].
- In some of these cases, the different MD elements (facts, dimensions, measures, hierarchies and so on) that underlie the DW should be defined by using XML [4].

In this way, using XML technologies in the context of DWs gives us a standardized basis to make secure the sharing of information between different applications and people. Unfortunately, in this scenario security may be jeopardized owing to the fact that information is shared between different applications and people.

Bearing in mind that the information managed by DWs is frequently highly sensitive and sometimes refers to personal data (protected under law in most countries), all the layers and operations of the DWs should be protected [9]. Each layer of a typical DW architecture (data sources integration, DW repository and access tools) has specific security concerns. For example, in the integration layer, data from heterogeneous data sources are extracted, transformed and loaded into the DW repository. In this layer, the main security problem is therefore to define processes that assure the integrity in the integration of heterogeneous data sources which usually use different security policies and configurations.

Nevertheless, the proposal presented in this paper is focused on the development of the main layer of a DW: the repository, by including security constraints into the models and the final implementation. Since end-users will analyze the information stored in the DW solely achieving read operations on the repository, the main security issue when it is being developed is to assure confidentiality [9–11]. Thus, our proposal focuses on confidentiality constraints (flexible access control mechanisms in particular) needed to assure that end-users do not access to unauthorized information. Our proposal includes different models for specifying the structure of the DW within security constraints at requirement, design and logical levels, and assisted by transformations, these models are automatically derived from the requirement model to the final implementation.

Security has been investigated in the context of XML but specially focused on assuring the contents of XML files. These advances are more related with the final implementation of the DW and can be applied in the final development stages or once the DW has been built [4]. Nevertheless, we advocate considering confidentiality issues in the whole XML DW development process, from an early development stage to the final implementation, in the sense of [12–17]. In this way, security and privacy measures should be integrated during the entire design cycle of the DW, from the early stages of its development as another relevant requirement. These security requirements are thus considered in the following development stages, for taking into account design decisions, signifying that much more robust and secure XML DW will be implemented.

In our previous works we have successfully proposed a hybrid Model Driven Architecture (MDA) [18] framework [19,20] for the design of DWs, where the DW repository can be implemented by using different kinds of database technologies: relational approaches store MD data by using relational database technology (i.e. tables, columns, foreign keys and so on); approaches based on a multidimensional database technology store data in proprietary structures such as MD arrays; and XML technology can be used to store the DW repository as XML semi-structured documents (as proposed in this paper). With regard to including security in the design and modeling of DWs, we have defined security specifications on the conceptual MD model (i.e. Platform Independent Model, PIM) [21], independently from the target logical MD model [22–24]. In relation to the XML technology, this secure conceptual MD data model it is used as a starting point and is then semi-automatically[2] transformed into a secure XML DW, as a logical model (i.e. Platform Specific Model, PSM), by applying Model to Model (M2M) Transformations.

In our previous work [25] we have focused on developing a specific case study in order to define an initial set of informal guidelines to obtain a secure XML DW from the corresponding MD PIM. However, M2M transformations were not formally defined which hinders the framework from applying the guidelines in an automatic and structured manner.

1.2. Contributions

In this paper we complement and improve our previous MDA framework [19,20] by proposing a new approach in which security models are embedded in and scattered throughout the high level MD models of the DW, which are then transformed towards the final implementation in XML according to the MDA strategy.

Our former model-driven architecture framework [19,20] is the starting point of our approach, since it is a generic framework for the development of data warehouses in a structured and formal manner. However, our former framework is generic and it was not instantiated to technologies in which the data warehouse can be deployed such as XML. Also, our former framework only considers data requirements, and it did not include other quality-of-service requirements such as security.

Therefore, the main improvement of our work is to provide our former model-driven framework with a set of transformations in order to generate automatically an XML Schema from a conceptual MD model, whilst no additional effort is needed to maintain the level of security required.

In this paper we specifically improve our previous work by focusing on:

1. Further refining and describing the secure MD PIM and secure XML PSM metamodels.
2. Describing the rationale behind the transformations rules and how they have been developed.
3. Formalizing transformation rules by using the Query/View/Transformation (QVT) language [26] in order to apply them automatically.

[2] Although, the transformation described in this paper is executed automatically, once we have the target secure PSM of the DW (based on XML) we need to manual refine it to be useful for a specific tool. Therefore, our approach is said to be semi-automatically.

4. Validation of the correctness of the defined QVT relations by means of their application to different examples and to a case study. Therefore the process of formalizing the transformation rules into QVT relations is a refinement process in which new constraints are included into QVT relations to achieve correctness.

The remainder of the paper is structured as follows: after presenting in Section 2 the background on our work, including an overview of QVT and the description of the framework for the model driven development of secure XML DW, in Section 3 we define the QVT transformations. For this purpose, first, in Sections 3.1 and 3.2, the metamodels at PIM and PSM levels have been refined and then, in Section 3.3, the PIM will be automatically transformed into a secure XML DW, as a PSM, by applying a set of transformation rules defined and formalized by means of a set of QVT transformations. In addition, in Section 4 we shall present a case study to show the application of a selected subset of the QVT relations defined. In Section 5 some related works are presented. Finally, in Section 6, we shall put forward our main conclusions and present our future work.

Appendix A contains the *list* of *acronyms* used in the paper and Appendix B shows the complete XML Schema generated in the Secure XML DW.

2. Background

For the sake of completeness, a brief overview of QVT is provided in this section. Afterwards, our model-driven framework for considering security issues in the DW development is outlined.

2.1. Query/View/Transformation language

One of the most crucial issues in model-driven development is the formal definition of transformations between models [18,27]. These formal transformations must allow models to be automatically derived while ensuring semantic correctness [28,29]. They must also be easily readable, understandable, adaptable, and maintainable [30]. The OMG has therefore proposed the MOF 2.0 Query/View/Transformation (QVT) language [26], a standard approach with which to define formal relations between MOF-compliant models.

QVT consists of two parts:

(1) The declarative part: provides mechanisms to define relations that must hold between the model elements of a set of candidate models (source and target models). A set of these relations (or transformation rules) defines a transformation between models.

The declarative part of QVT can be split into two layers, according to the level of abstraction:

(a) The relational layer: provides graphical and textual notation for a declarative specification of relations.
(b) The core layer: provides a simpler but verbose means of defining relations.

(2) The imperative part: defines operational mappings that extend the declarative part with imperative implementations when it is difficult to provide a purely declarative specification of a relation.

In this paper, we focus on the relational layer of QVT. This layer supports the specification of relationships that must hold between MOF models by means of a relational language. The abstract syntax of QVT is shown in Fig. 1. The metamodel of QVT states that a transformation is composed of a set of rules. Each rule may contain several domains. Each domain is a distinguished set of elements of a candidate typed model (source or target model). This set of elements (denoted by a <<domain>> label) must be matched in that model by means of patterns. A domain pattern can be considered as a template for elements, their properties and their associations that must be located, modified or created in a candidate model in order to satisfy the relation. A relation between domains can be marked as check-only (labeled as C) or as enforced (labeled as E). When a relation is executed in the direction of a check-only domain, it is only checked if a valid match exists in the model that satisfies the relationship (without modifying any model if the domains do not match); whereas for a domain that is enforced, when the domains do not match, model elements are created, deleted, or modified in the target model in order to satisfy the relationship. Moreover, a when clause specifies the condition under which the relation needs to hold (i.e., it forms a precondition), while a where clause specifies the condition that must be satisfied by all model elements participating in the relation (i.e., it forms a post-condition). These clauses may contain arbitrary OCL (Object Constraint Language) [31] expressions in addition to the relation invocation expressions.

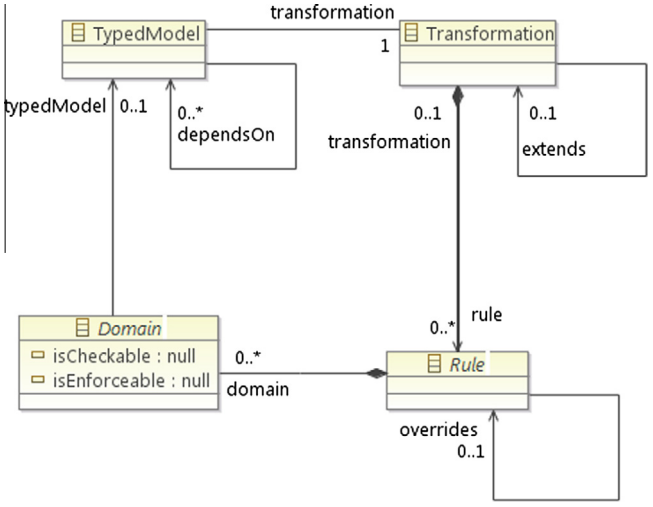


Fig. 1. Excerpt of the QVT relation metamodel.

The definition of relations by using the QVT language has the following advantages:

- QVT is a standard language.
- Relations are formally specified, and can be automatically executed with transformation engines (e.g., SmartQVT [32], mediniQVT [33], or ATL [34]).
- Formalizing the mappings before implementing them leads to the detection of errors and inconsistencies in the early stages of software development and can help to increase the quality of both the models built and the subsequent code generated from them. The formalization of mappings also simplifies the development of tools that support any model driven approach.
- Sets of QVT relations can easily be integrated as transformations within an MDA approach.

2.2. Framework for the development of secure XML DWs

We have proposed a hybrid MDA approach for the design of DWs (see Fig. 2) by (i) firstly specifying users' requirements [35] in a Computation Independent Model (CIM), (ii) obtaining a conceptual model as an implementation-independent and expressive

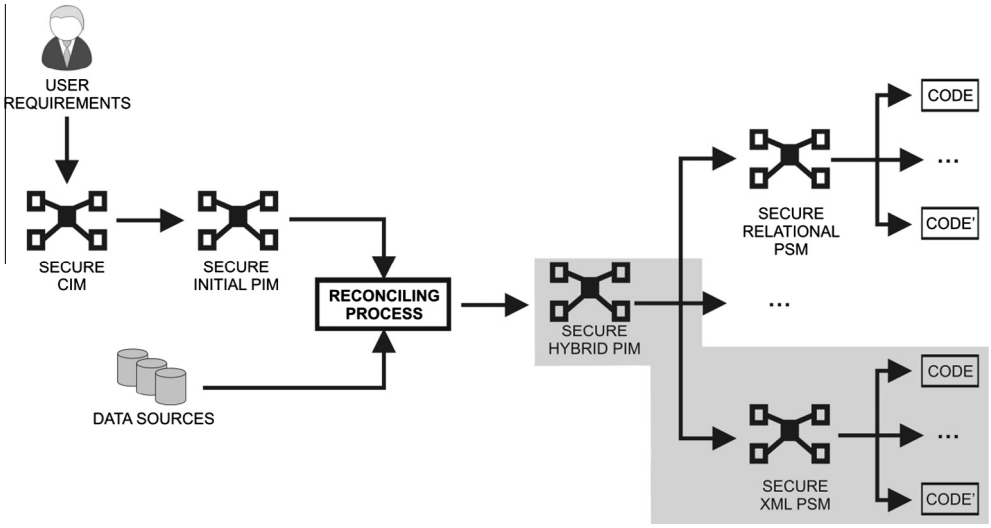


Fig. 2. Hybrid development approach for Secure XML DW.

conceptual MD model for the DW (i.e. a PIM) [36], (iii) reconciling data sources with PIM models [36], (iv) obtaining a logical and a technology-dependent model (i.e. PSM) from the previously defined conceptual MD model [36], and, finally, (v) obtaining the corresponding code in order to implement the DW on a concrete platform.

In [25], we extended the hybrid MDA architecture for secure MD modeling of DWs (previously defined in [19]) in order to be able to deal with XML (see shadowed part of Fig. 2). More specifically, a requirements analysis stage [21] and the available data sources [21] are used to define security specifications on the conceptual MD model (secure hybrid PIM), independently of the target logical MD model. A set of guidelines should be applied to this secure conceptual data model in order to obtain a secure XML DW, as a logical model (secure XML PSM). In this paper, we have further refined the metamodels and defined a set of Model to Model (M2M) Transformations, based on the previously defined informal guidelines, in order to increase the degree of automation.

3. Using QVT for the automatic generation of secure XML DWs

Our QVT transformations have been developed by bearing in consideration our previous MDA framework for the development of a secure XML DW (see Fig. 2). Consequently, before being able to define our transformations, it is necessary to perform the following tasks:

- Refining metamodels at the **PIM level**: the secure multidimensional model is created without considering the selected technology, since this model is independent of the platform. This MD PIM (described in more detail in the following subsection) is represented through an extended class diagram designed for DWs which additionally allows the specification of security constraints over the model.
- Refining metamodels at the **PSM level**: the data logical design is performed, taking into account the selected target platform on which the DW will be implemented. In our case, XML technology will be used for the implementation of the DW in any secure commercial database management system (e.g. Oracle XML DB 11g). We will start from the secure MD PIM obtained at the previous level and will apply the M2M mappings summarized in Section 3.3 to obtain an XML Schema, conforming to the XML Schema Metamodel [37] (see Section 3.2).

3.1. Secure MD PIM

As previously mentioned, our proposed development approach starts from the conceptual model of the secure MD PIM.

This secure MD PIM has been defined by developing a secure UML profile called SECDW (for more details, see [38]). SECDW (Fig. 3) uses an Access Control and Audit (ACA) model [39] to consider both DW specific modeling aspects (such as facts, dimensions, base classes, measures, hierarchies, many-to-many relations, degenerated dimensions, multiple classifications or alternative paths of hierarchies) and security capabilities.

The ACA model classifies authorization subjects and objects into security roles ("SRole" metaclass) which organize users into a hierarchical role structure according to the responsibilities of each type of work, levels ("SLevel" metaclass) which indicate the user's clearance level, and compartments ("SCompartment" metaclass) which classify users into a set of horizontal compartments or groups.

The definition of several kinds of security rules related to the multidimensional elements of DWs is also permitted: sensitive information assignment rules (SIAR) ("SecurityRule" metaclass) which specify multilevel security policies and allow sensitive information to be defined for each element in the multidimensional model; authorization rules (AUR) ("AuthorizationRule" metaclass) which permit or deny access to certain objects by defining the subject that the rule applies to, the object that the authorization refers to, the action that the rule refers to and the sign describing whether the rule permits or denies access; and audit rules (AR) ("AuditRule" metaclass) which ensure that authorized users do not misuse their privileges.

3.2. Secure XML DW PSM

We propose the use of an XML Schema to represent the PSM level of a Secure XML DW. In this paper, we use one of the most frequently employed graphical representations of the XML Schema to present the metamodel of our Secure XML DW PSM, which includes both the MD and the security aspects. The discontinuous line of the classes indicates that the class is optional.

This XML Schema is presented in Fig. 4 with the root XML Element *SecureMDXML*. Its XML subelements include the *Security Levels*, the *Security Roles* Hierarchy, *Security Compartments*, along with the *User Profile* and the *Secure Star Package*.

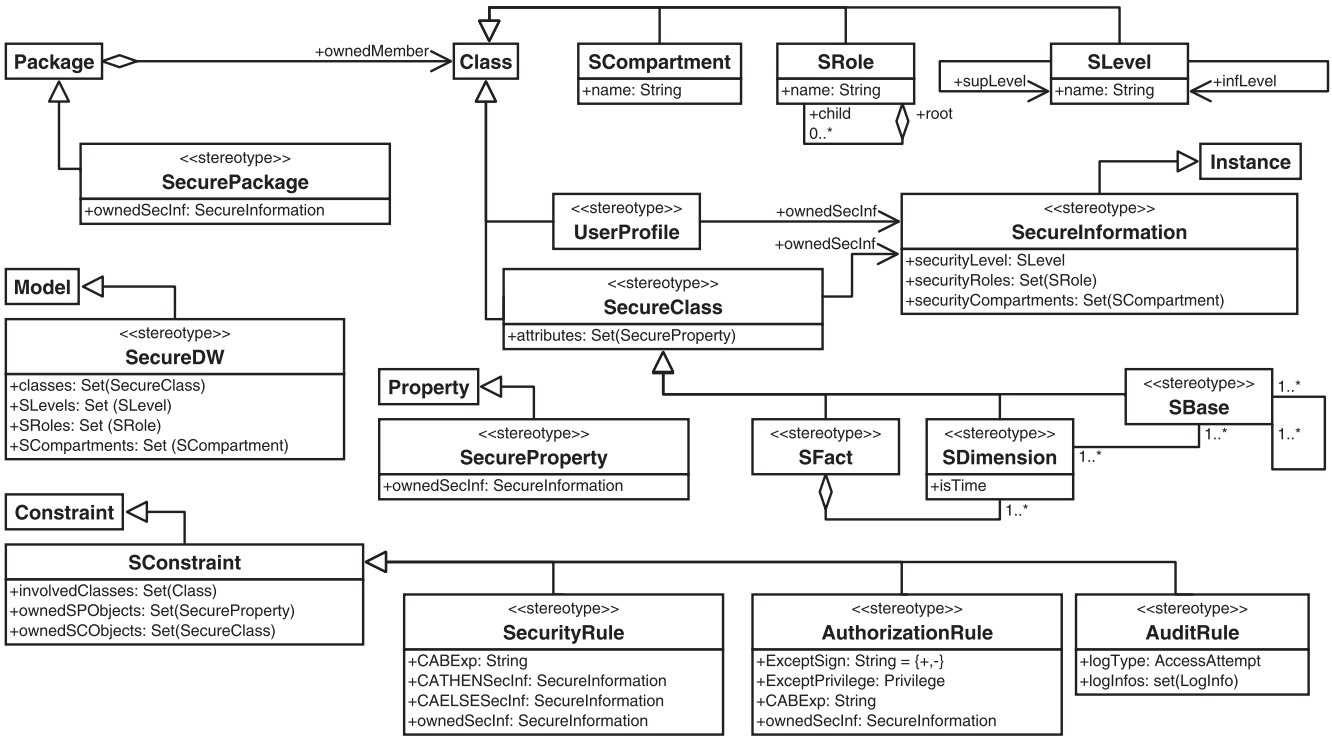


Fig. 3. Conceptual Secure MD Metamodel.

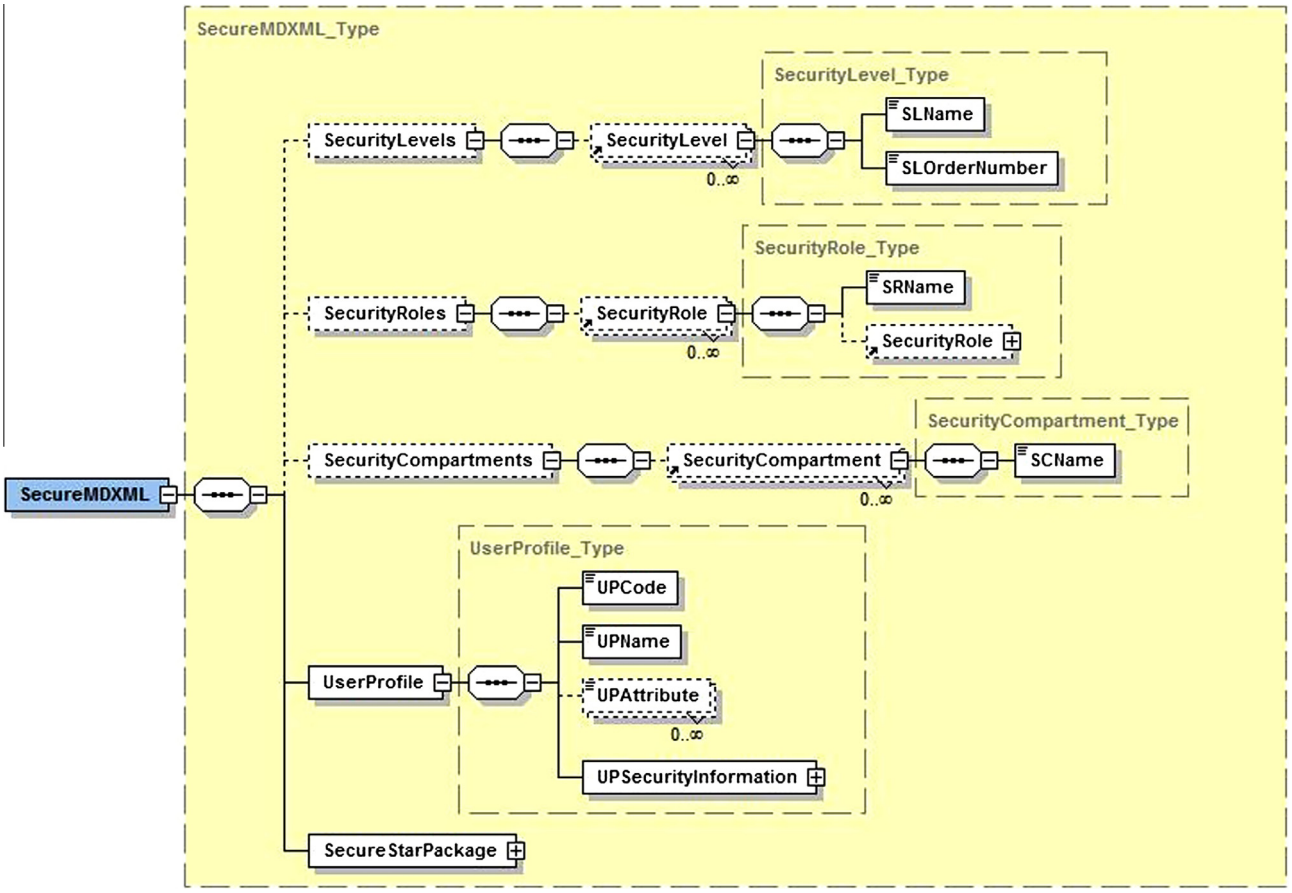


Fig. 4. Secure XML DW PSM-Metamodel.

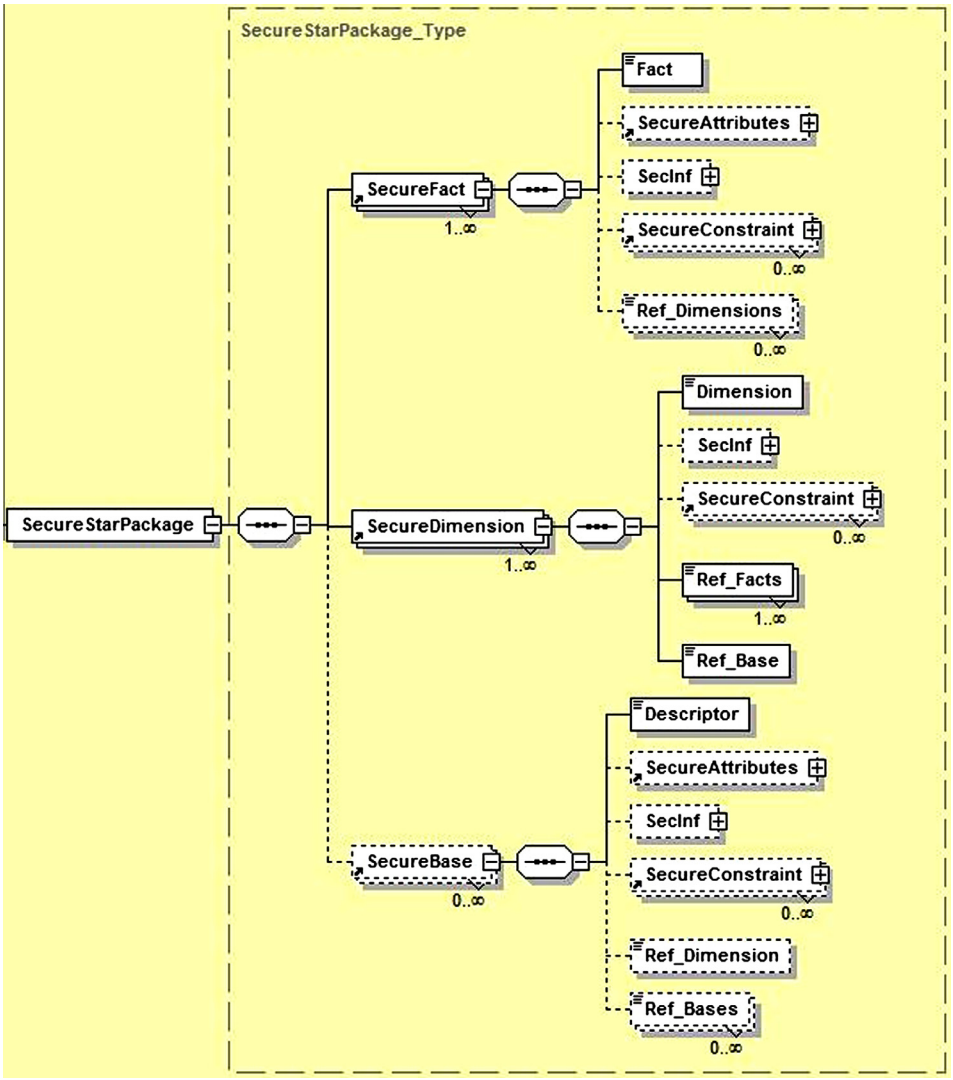


Fig. 5. Secure XML DW PSM-Secure Star Package.

The XML Element ***UPSecurityInformation*** will be represented with an XML sequence complexType, which includes the Security Level, Security Role/s and Security Compartment/s, all of whose attributes are represented as XML subelements.

Fig. 5 shows the XML Element ***SecureStarPackage***. It includes Secure Base, Secure Fact and Secure Dimension XML subelements. These three subelements can appear several times within a Secure Star Package but at least one (Secure) Fact and one (Secure) Dimension Elements must appear. The Secure Facts, Dimensions and Bases can, if necessary, include the corresponding Security Information and Security Constraints as XML subelements. The Secure Attributes correspond to the class attributes defined by the designer and each of them can include optionally Secure Information.

Fig. 6 shows the part of the XML Schema Metamodel corresponding to the Security Constraints. Each ***Secure Constraint*** XML Element includes its attributes (*involvedObjects, ownedSPObjects, ownedSCObjects*) and is of a specific type (choice complexType): Security Rule, Audit Rule or Authorization Rule.

3.3. PIM to PSM transformations

In the same way that methodologies for relational or object-relational DBs or XML DBs [40] propose particular rules for the transformation of a conceptual schema into a logical schema, in this section we propose the mappings from the secure MD PIM to the XML Schema for the Secure XML DW. Once the PIM and the PSM are defined within an MDA approach, the most challenging task is to develop a set of formal transformations that can be used to derive a PSM from a PIM in an automatic manner. The obtained PSM has to be manually refined for deployment in a specific tool. We have therefore first described the mappings informally to then formalize them by using the declarative approach of QVT, since it provides a graphical notation that allows us to specify model transformations, which are easily readable, understandable, adaptable and maintainable. Actually, our definition of correct QVT relations is a refinement process that can be summarized as follows:

- First, we have developed a set of guidelines to transform our Secure MD PIM into a Secure XML PSM.
- Second, several toy examples (incrementally done) have been carried out in order to validate the target of our informal transformation process, thus checking if the generated PSM corresponds to a PSM manually designed by a DW expert. Some initial QVT relations are defined at this point, and the required constraints for achieving the examples successfully are incrementally added to them (by means of when and where clauses).
- Finally, we have used our approach in a real-world case study to check the correctness of the transformation process according to the requirements of end-users.

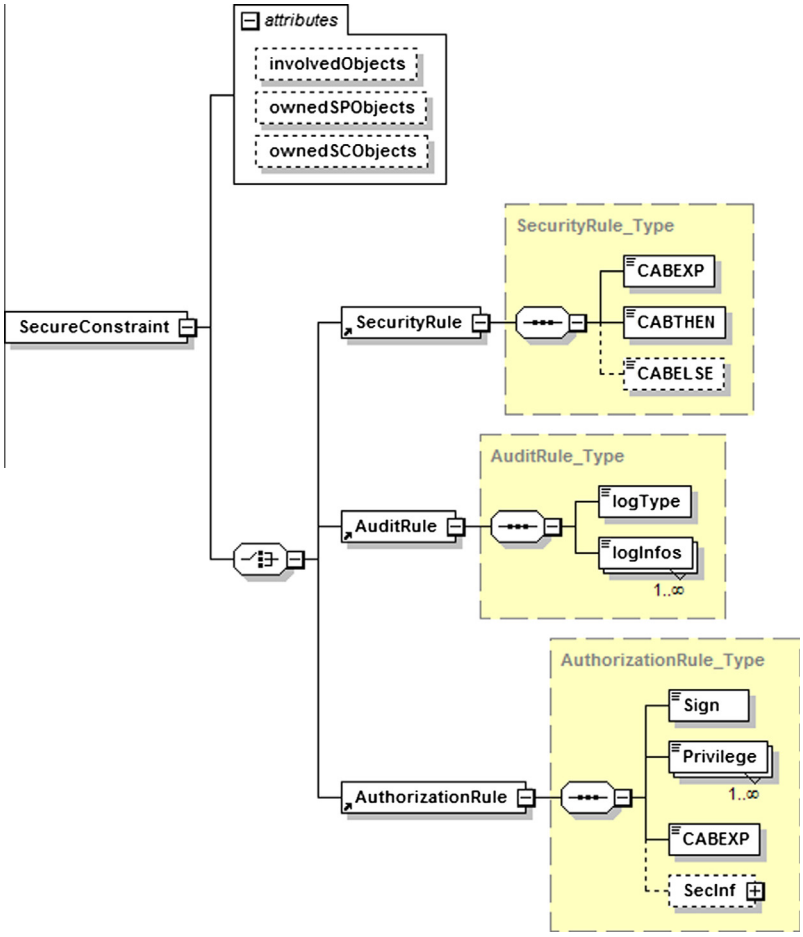


Fig. 6. Secure XML DW PSM-Secure Constraints.

According to the QVT relations language, we have therefore formally specified our mappings to obtain a transformation between a Secure MD PIM (source model) and a Secure XML PSM (target model).

Fig. 7 shows an overview of every designed QVT relation, and the dependencies among them. Each relation has been referenced by means of the modeling elements that it maps. Each relation depends on many others to complete the overall mapping. QVT also allows us to specify which relations are the entry points to start the transformation process by means of identifying them as a top relation type. Hence, during the transformation process, each top relation calls its depending non-top to complete the remaining mappings involved. Each non-top relation also calls the depending non-top relations in a recursive manner. It is worth noting that in this paper we focus on the QVT relations that are related to the security capabilities of the source metamodel (the non-secure part of the metamodel was previously considered in [20]).

For the sake of understandability, QVT relations are explained by taking into account their dependencies. This section has therefore been structured as follows:

Transformation of the SecureDW: the complete MD PIM will be transformed into an XML Schema which will include the root Element SecureMDXML.

Transformation of the Security Levels, Roles and Compartments: the Security Levels, Roles Hierarchy, and Compartments conceptually defined for a DW in the PIM will be transformed into their corresponding XML Elements.

Transformation of the User Profile: subjects are classified in the PIM by using a user profile which defines the information to be stored for each user. It will be transformed into several XML Elements.

Transformation of the Secure Information: the Secure Information defined at the conceptual level for each specific element of the DW (e.g. Secure Fact, Dimension or Base), will be transformed into its corresponding XML Elements.

Transformation of the Secure Star Package: each Secure Star Package of the MD PIM will be transformed into an XML Element.

Transformation of the Secure Facts: each Secure Fact will be transformed into an XML Element.

Transformation of the Secure Dimensions: each Secure Dimension will be transformed into an XML Element.

Transformation of the Secure Bases: each Secure Base of a hierarchy defined in the MD PIM will be transformed into an XML Element.

Transformation of the Secure Constraints: Security Constraints are at two different levels: PIM and PSM. Therefore, it is first necessary to specify the security constraints at a PIM level with the OCL language and to then transform them into XPath expressions, which are supported by most XML Database Management Systems (both native and XML-enabled).

3.3.1. Transformation of the SecureDW

When a Secure XML PSM is derived from the Secure MD PIM (SecureDW), the starting point is the creation of a "SecureMDXML" root Element of "SecureMDXML_Type" within the XML Schema. This will include the Security Levels, Security Roles Hierarchy, Security Compartments, User Profile and Secure Star Package. This has been done by developing a QVT transformation rule: SecureDW2SecureMDXML (see Fig. 8).

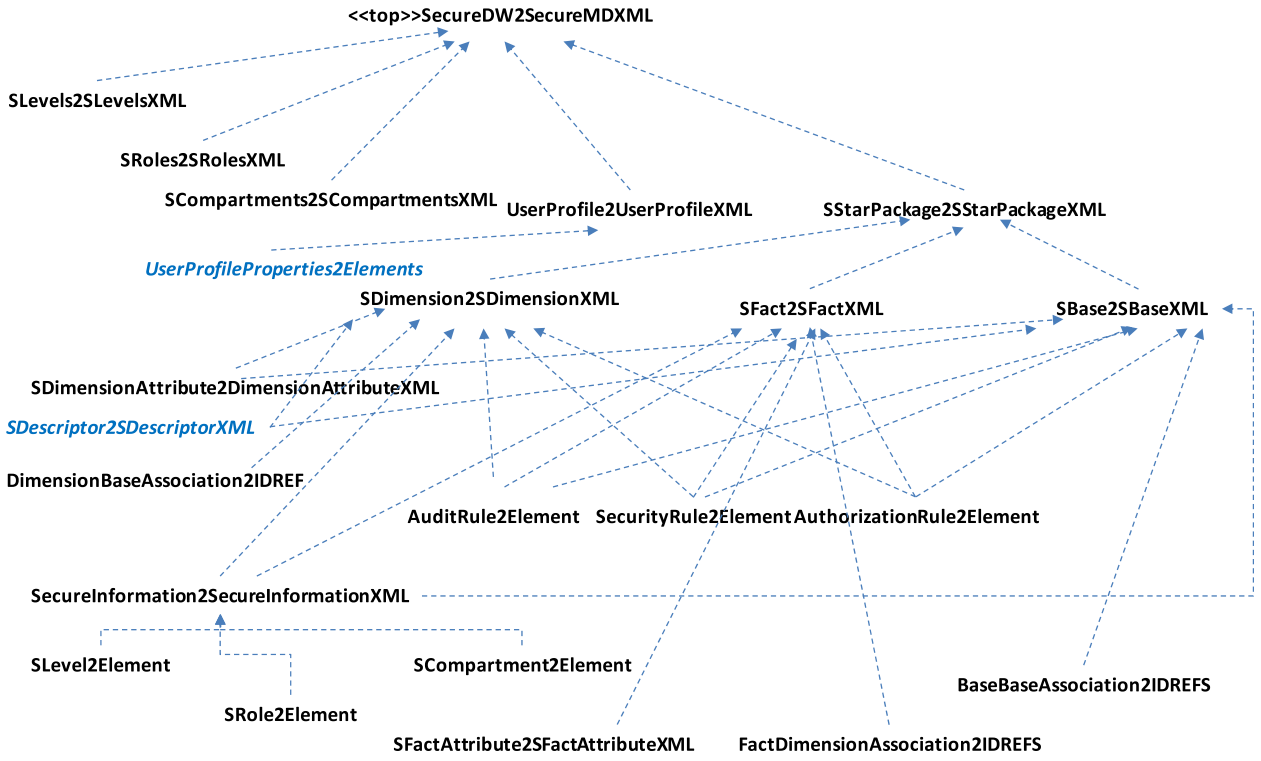


Fig. 7. Dependency Graph of the QVT relations implementing the mappings.

This QVT transformation rule matches a "SecureDW" element in the data source in order to enforce the following set of elements in the target model: a "Schema" with an "Element" named "SecureMDXML" of the complexType "SecureMDXML_Type". This complexType contains a sequence element "Sequence" with five "Element" classes, namely ***esl*** (corresponding to "SecurityLevels" and with the complex type "SecurityLevels_Type"), ***esr*** (corresponding to "SecurityRoles" and with the complex type "SecurityRoles_Type"), ***esc*** (corresponding to "SecurityCompartments" and with the complex type "SecurityCompartments_Type"), ***eup*** (corresponding to "UserProfile" and with the complex type "UserProfile_Type"), and ***esp*** (corresponding to "SStarPackage" and with the complex type "SStarPackage_Type"). A "Sequence" is created for each of these elements in which the SLevels, SRoles, SCompartments, and the counterparts of the UserProfile and the SStarPackage will be created by executing the QVT rules called in the *where* clause.

3.3.2. Transformation of the Security Levels, Roles and Compartments

The Security Levels Hierarchy defined for a specific MD PIM is transformed, by using the previously defined QVT rule (see Fig. 8), into an XML Element called "SecurityLevels" including a sequence complexType ("SecurityLevels_Type"). Each of the Security Levels defined is then transformed into subelements of that complexType with a sequence. They will all contain the following attributes: *SLname* (name of the Security Level) and *ordernumber* (1 being the highest security level). The SLevels2SLevelsXML QVT relation (shown in Fig. 9) matches a SLevel in the source model (together with a SecureDW class) to create an "Element" in the target model. Its name will be the same as the original SLevel and the order number will be calculated by the function defined in the *where* clause.

The Security Roles Hierarchy defined for a DW in the conceptual model is transformed into an XML Element called "SecurityRoles" including a sequence complexType ("SecurityRole_Type") with all the defined Security Roles as subelements denominated as the Security Roles. Each subelement will contain the name and a reference Element to its parent (*fatherRole* attribute), i.e., the security role in which it is included. After applying the SecureDW2SecureMDXML QVT relation, a new QVT relation named SRole2SRoleXML (shown in Fig. 10) is executed to match a SRole class in the MD PIM and this is transformed into an "Element" whose name is the name of the SRole. The father role is obtained from the function defined in the *where* clause.

The set of Security Compartments defined for an MD PIM in the conceptual model will be transformed into an XML Element called "SecurityCompartments", including a sequence complexType ("SecurityCompartments_Type") with all the defined Security Compartments as subelements, denominated as the Security Compartment. A QVT relation is defined to create the SecurityCompartments in the MD PIM, thus creating an "Element" with the same name in the target model: the SCompartments2SCompartmentsXML (see Fig. 11).

3.3.3. Transformation of the User Profile

The User Profile class is transformed into an XML Element of a complexType "UserProfile_Type", including a sequence element, by using the SecureDW2SecureMDXML QVT (see Fig. 8). Different subelements must now be included: its code, the name, the specific class attributes and the "SecInf" XML subelement of the complexType "SecureInformation_Type", which contains the three security attributes as XML subelements named SecurityLevel, SecurityRoles and SecurityCompartments. A QVT relation has been defined to carry out this mapping: UserProfile2UserProfileXML. A QVT relation UserProfileProperties2Elements is called in the *where* clause in order to deal with the properties that may have the UserProfile class. This QVT relation is not described since non-security capabilities are not within the scope of this paper.

3.3.4. Transformation of the Secure Information

We have developed a set of QVT relations in order to deal with security information, regardless of the kind of SecureClass, as fol-

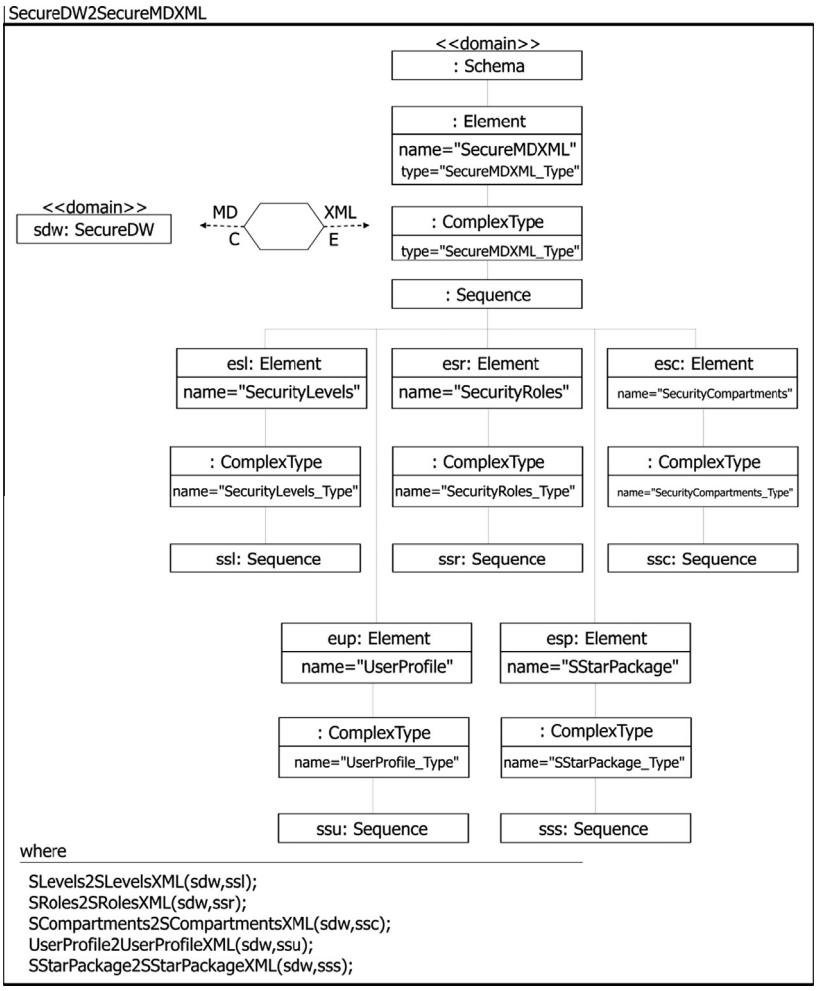


Fig. 8. Transforming SecureDW into an XML Schema.

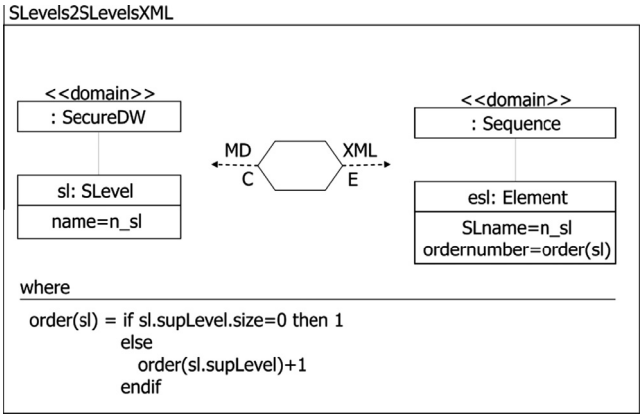


Fig. 9. Transforming Security Levels into an XML Element.

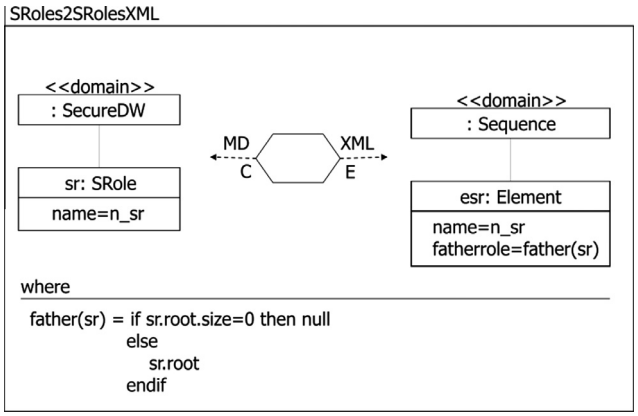


Fig. 10. Transforming Security Roles into an XML Element.

lows: if a SecureClass has an association with a SecureInformation class in the Secure MD PIM, then the SecureInformation2SecureInformationXML QVT relation is executed to create the required security information in the Secure MD XML PSM.

The "SecureInformation" class, which contains three security attributes (SecurityLevel, SecurityRole and SecurityCompartment) is associated with a specific element of the Secure MD PIM (e.g. Secure Fact, Dimension or Base). It will be transformed into an XML Element called "Secure Information", including a complexType "SecureInformationType" according to the SecureInformation2SecureInformationXML with a sequence element "Sequence".

This sequence may include three subelements depending on the class matched in the source model together with the SecureInformation class:

- "SecurityLevel" of "SecurityLevel_Type" with the corresponding attributes as XML subelements by means of SLevel2Element QVT relation.

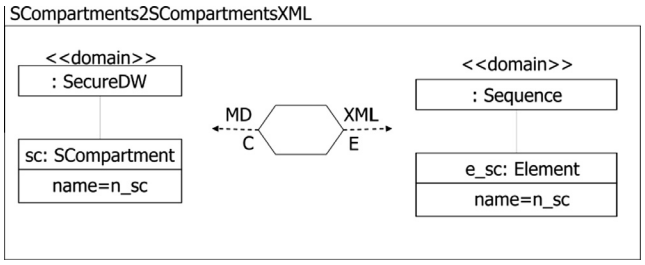


Fig. 11. Transforming Security Compartments into an XML Element.

- "SecurityRole" of "SecurityRoles_Type" with the corresponding attributes as XML subelements by using SRole2ElementQVT relation.
- "SecurityCompartment" of "SecurityCompartments_Type" with the corresponding attributes as XML subelements by using SCompartment2Element QVT relation.

3.3.5. Transformation of the Secure Star Package

Each Secure Star Package is transformed into an XML Element called "SStarPackage" within the "SecureMDXML" XML Element, including a complexType "SStarPackage_Type" with a sequence according to the aforementioned QVT relation SecureDW2SecureMDXML. This sequence will contain the Fact, Dimension and Base XML subelements, each of which will contain the corresponding Security Information and Security Constraints as XML subelements. In the *where* clause of the SStarPackage2SStarPackageXML relation, several QVT relations are executed to match the different MD elements in the source model and create their counterparts in the target model. These QVT relations are explained in the following subsections.

3.3.6. Transformation of the Secure Facts

The SFact2SFactXML QVT relation (see Fig. 12) considers a SFact class that is included in a SecureStarPackage, together with its Secure Information (SecureInformation class). Once these elements have been matched in the source model, a set of elements is created in the target model. Each Secure Fact will be transformed into an XML subelement with the same name as its PIM counterpart (***n_sf***) and included in the XML Element "SecureFacts" of XML complexType "SecureFacts_Type" of the Secure Star Package. The subelement, corresponding to each of the facts includes a complexType with a sequence (***sa***) in order to store the (secure) Fact attributes as XML subelements and an ID attribute denominated the fact + "_ID" (so that the element can be referenced by other elements by means of an IDREF/S Element). This mapping has been completed with the implementation of several QVT relations, which are called in the *where* clause in order to obtain the fact attributes, the security constraints, the security information associated with this Secure Fact, and the IDREFS from this Fact to the Dimensions. It is worth noting that one of the parameters of the SFactAttribute2SFactAttributeXML QVT rule is the name of the Fact which is used as a prefix to the names of each of its fact attributes.

The SFactAttribute2 SFactAttributeXML QVT relation matches each secure fact attribute (with its Secure Information), and a prefix is added to the name of the XML Element that represents the secure fact attribute in the PSM (***n_e***). The value of ***n_e*** is determined in the *where* clause.

Once matched, a set of classes are created in the PSM. It is worth noting that the secure information is dealt with by calling the SecureInformation2SecureInformationXML QVT relation shown in the *where* clause of the QVT relation.

The FactDimensionAssociation2IDREF has been defined in order to create an XML Element in the Fact element to refer the Dimensions.

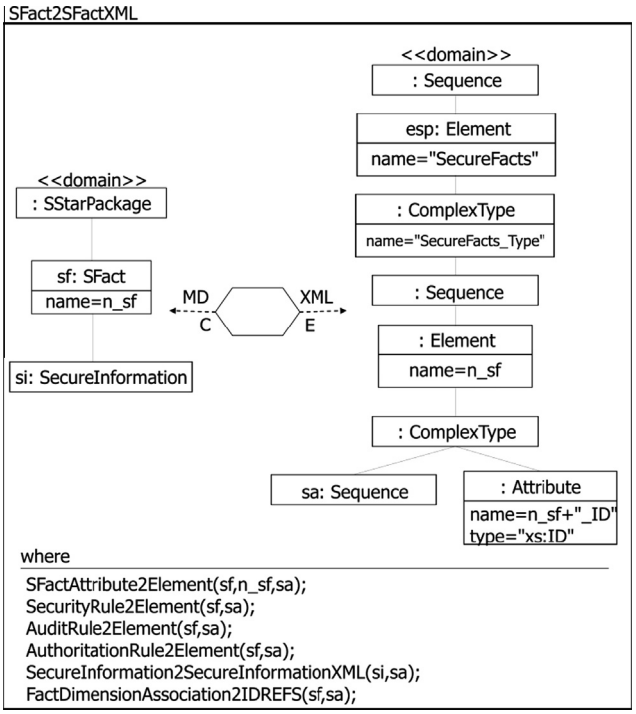


Fig. 12. Transforming Secure Fact into an XML Complex Type.

The QVT relations associated to the Security Constraints will be detailed in the following Section 3.3.9.

3.3.7. Transformation of the Secure Dimensions

The SDimension2SDimensionXML QVT relation (see Fig. 13) considers a SDimension class that is included in the SecureStarPackage, together with its Secure Information (SecureInformation class). Once these elements have been matched in the source model, a set of elements are created in the target model. Each Secure Dimension will be transformed into an XML subelement which has the same name as its PIM counterpart (***n_sd***) and is included in the XML Element "SecureDimensions" of the XML complexType "SecureDimensions_Type" of the Secure Star Package. The subelement corresponding to the specific dimension includes a complexType with a sequence (***sa***) in order to store the (secure) Dimension attributes as XML subelements and an ID attribute denominated as the dimension + "_ID" (so that the element can be referenced by other elements by means of an IDREF/S Element). Several QVT relations have been implemented to carry out this mapping, which are called in the *where* clause in order to obtain the dimension attributes, the security constraints, the security information associated with this Secure Dimension and the IDREF from this Dimension to the Base. The SDescritor2SDesriptorXML is also called in the *where* clause in order to create the Secure Descriptor classes in the corresponding Secure Base class. However, since this QVT rule is quite similar to the SDimensionAttribute2SDimensionAttributeXML, it is not shown in this paper.

Finally, it is worth noting that one of the parameters of the SDimensionAttribute2DimensionAttributeXML rule is the name of the Dimension which is used as a prefix to the names of each of its dimension attributes.

The SDimensionAttribute2SDimensionAttributeXML QVT relation matches each secure dimension attribute (with its Secure Information) that belongs to the Dimension to enforce an XML Element that represents the secure Dimension attribute in the PSM (***e_sd***) whose name is calculated in the *where* clause by adding a

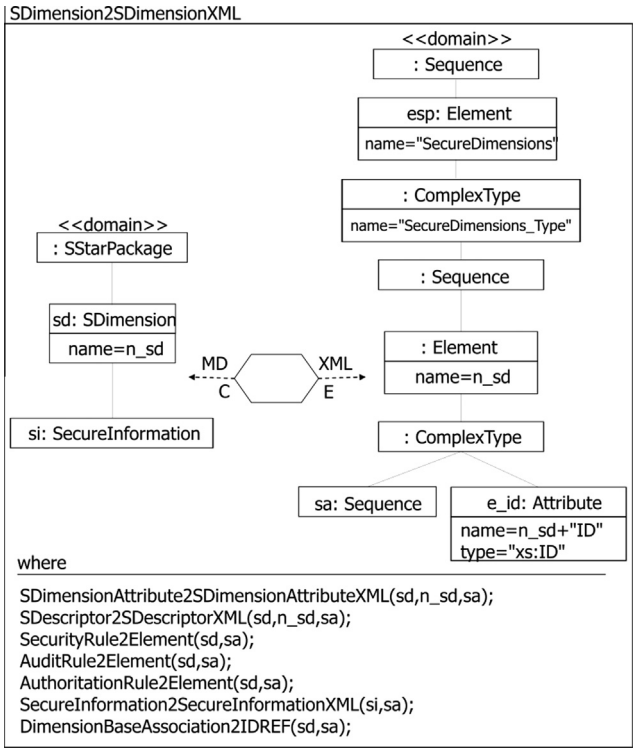


Fig. 13. Transforming Secure Dimension into XML Secure Dimension.

prefix. Once matched, a set of classes are created in the PSM. It is worth noting that the secure information is dealt with by calling the SecureInformation2SecureInformationXML QVT relation shown in the *where* clause.

The DimensionBaseAssociation2IDREF has been defined in order to create an XML Element in the Dimension element to refer the first Base in the dimension hierarchy.

The QVT relations associated to the Security Constraints will be detailed in the following Section 3.3.9.

3.3.8. Transformation of the Secure Bases

The SBase2SBaseXML QVT relation considers a SBase class that is included in the SecureStarPackage, together with its secure information (SecureInformation class). Once these elements have been matched in the source model, a set of elements are created in the target model. Each Secure Base will be transformed into an XML subelement which has the same name as its PIM counterpart (***n_sb***) and is included in the XML Element "SecureBases" of XML complexType "SecureBases_Type" of the Secure Star Package. The subelement corresponding to the specific base includes a complexType with a sequence (***sa***) in order to store the (secure) Base attributes as XML subelements and an ID attribute denominated as the base + "_ID" (so that the element can be referenced by other elements by means of an IDREF/S Element).

This mapping has been carried out by implementing three QVT relations, which are called in the *where* clause in order to obtain the dimension attributes, the descriptor attributes (although this rule has not yet been not described in the paper because it is similar to the SDimensionAttribute2SDimensionAttributeXML rule), the security constraints, the security information associated with this Secure Base and the IDREFS element to reference other Bases.

It is worth noting that one of the parameters of the SDimensionAttribute2DimensionAttributeXML rule is the name of the Base (***n_sb***) which is used as a prefix to the names of each of its dimension attributes in the SDimensionAttribute2SDimensionAttributeXML relation.

This element is related to a complexType that contains an IDREFS Element and a sequence (***sa***) in order to store the (secure) Dimension attributes as XML subelements and an IDREFS Element to reference other Bases of the hierarchy.

The SDimensionAttribute2SDimensionAttributeXML QVT relation matches each secure dimension attribute (with its Secure Information) that belongs to a Base to enforce an XML "Element" that represents the secure Dimension attribute in the PSM (***e_sb***) whose name is determined in the *where* clause by adding a prefix to the name of the source dimension attribute. Once matched, a set of classes are created in the PSM. It is worth noting that the secure information is dealt with by calling the SecureInformation2SecureInformationXML QVT relation shown in the *where* clause.

The BaseBaseAssociation2IDREFS has been defined in order to create an XML tag in the Base element to refer the other Bases in the dimension hierarchy. After applying this rule, the Secure Base XML Element will include an IDREFS Element that references the associated Bases.

3.3.9. Transformation of the Secure Constraints

The Secure Constraints contain three optional attributes: involvedClases, ownedSPObjects and ownedSCObjects. Each Secure Constraint will be transformed into an XML subelement of the corresponding constraint, with the three optional attributes as optional string type subelements (involvedObjects, ownedSPObjects, ownedSCObjects), with references to the elements, secure properties or secure elements.

Various Secure Constraints have been considered in accordance with the Secure MD PIM:

- Transforming the Audit Rule Constraints: an Audit Rule is a Secure Constraint that will be transformed as an XML subelement of the corresponding Secure Base, Dimension or Fact Element, denominated as the Audit Rule at the PIM level. This element is of the complexType "AuditRule_Type" and can contain the three aforementioned XML subelements and the following two subelements: logType and logInfos, both of which are string types, with the same value as their PIM counterpart.
- Transforming the Authorization Rules Constraints: an Authorization Rule is a Secure Constraint that will be transformed as an XML subelement of the corresponding Secure Base, Dimension or Fact Element, denominated as the Authorization Rule at PIM level. This element is of a complexType "AuthorizationRule_Type" and can contain the three aforementioned subelements and the following subelements: the ExceptSign, of SimpleType with an enumeration constraint and the fixed value {+,-}, ExceptPrivilege also of SimpleType with an enumeration constraint with fixed value {read, write}, and the CabExp. It contains the Authorization Rule Condition, which is a string type, that will contain the XPath expression associated with the OCL expression.
- Transforming the Security Rules Constraints: a Security Rule is a Secure Constraint that will be transformed as an XML subelement of the corresponding Secure Base, Dimension or Fact Element, denominated as the Security Rule at the PIM level. This element is of a complexType "SecurityRule_Type" and can contain the three aforementioned subelements and the following subelements: CABExp, which will contain a string with the expression in XPATH."; the CABTHEN, which will contain the Security Information if the expression (in XPath) is TRUE; and the CABELSE subelement, which will contain the Security Information if the expression is FALSE.

4. Case study

In this section, we will briefly give some details about the examples and the real-world case study we used for designing our QVT transformation. Then, a case study is used to show how our previously defined QVT relations are properly applied to obtain the Secure XML DW (PSM) from the secure conceptual MD data model (PIM).

Two examples were used for refining the QVT relations. The first one is related to the management of a pharmacies consortium business as stated in [22], and the second one is based on a DW for the security department of an Airport (it will be described in this section). Both case studies are toy examples with data sources containing a range from 10 to 20 tables, and a range from 100 to 200 instances. In these toy examples the average number of measures by table is around 0.5.

Importantly, our approach is being validated in the context of a project in which we are involved in the development of a DW for a Chilean university [41]: Universidad de La Frontera (UNIVFRONTERA1-09I). This project focused on developing a DW for a higher education institution. Therefore, in order to know the requirements of the project, the Business Strategic Plan 2006–2010 of the University of La Frontera[3] were considered.

Personnel from the "Dirección de Análisis y Desarrollo Institucional de la Universidad de La Frontera"[4], which is the office in charge of the business strategic plan of the University of La Frontera. These meetings and interviews were very valuable for discussing the aforementioned documentation in order to determine the conceptual models and the security constraints. At the end we have one model per strategic axis. Then, we focused on designing a data mart for personnel, having data sources with 183 tables and an average of instances per table of 21350.89. The average number of measures per table was 1.4590164.

This project has provided us with a lot of knowledge about the required constraints for moving from the informal rules to the QVT transformation, thus allowing us to check its correctness.

For the sake of understandability, in this paper, we present a simplified case study based on an airport security department to show the benefits of our proposal.

Our case study is inspired by both, (i) security departments stated in airports, such as the Airport Security Unit at the Hong Kong International Airport[5] or the Airport Police Service in airports of Ireland[6]; and (ii) security departments of airports that belong to an upper department, such as the Department for Transport in UK[7] or the Federal Aviation Administration in the USA[8]. The main goal of these security departments is to prevent harm to aircraft, passengers and crew, as well as support national security and counter-terrorism policy. To this aim they need managing information about trips, flights, departure and arrival places and dates, passengers and baggage. Furthermore, sensitive information collected in the passport control, such as the fingerprints, must be also considered. Due to the fact that large amount of people pass through airports every day, a secure DW must be designed to store and manage this huge amount of data. Moreover, as the security department of an Airport manages external data from heterogeneous sources, we have chosen XML as the DW repository technology.

4.1. MD data model

Fig. 14 shows the secure conceptual multidimensional model (PIM) used in this case study, which is focused on trips. A Secure Star Package with a central fact for trips ("Trip" secure fact class) has therefore been defined, which is related to dimensions for passengers, baggage, flights, departure and arrival places and dates ("Passenger", "Baggage", "Flight", "Place" and "Date" secure dimension classes). The secure fact class "Trip" includes attributes with trip information regarding price, purpose (which can be "tourist", "business" or "military"), seat, distance, flight time, and whether or not the check-in and boarding procedures have been carried out.

The secure dimension class "Passenger" includes attributes containing personal information about passengers (code, name and address) and extended security information, such as fingerprint, passport photo, criminal record, whether the passenger is considered to be suspicious, and his/her estimated risk index (a number from 1 to 10). The secure dimension class "Baggage" has several attributes containing information about the number of baggage items, identification codes, weight, whether the baggage has been inspected and whether it is suspicious.

The other secure dimensions classes ("Place", "Date" and "Flight") only include identification attributes. These dimensions are also related to secure base classes, forming navigation hierarchies which allow the information to be aggregated in different levels. For instance, the "Place" dimension is related to "Gate", "Terminal" and "Airport" base classes and represents a hierarchy which allows the information to be aggregated by gate, terminal and airport. The "Date" dimension can be similarly aggregated by hours, days, months and years, and the "Flight" dimension by planes, aircraft types and companies.

Our access control and audit model permits a three point of view security classification by using security levels ("Security Levels", SL) with the users' clearance levels; a hierarchical structure of security roles ("Security Roles", SR); and a set of horizontal security compartments or groups ("Security Compartments", SC). Fig. 15 shows the security configuration used in this case study: the levels of security (SL) used are top secret (TS), secret (S), confidential (C) and undefined (U); the hierarchy of security roles (SR) has a main system user "User" specialized into "Passenger" and airport "Staff" which is composed of "Security", "Flight" and "Administration" (specialized into "Boarding" and "Baggaging") roles; and the security compartments are different airlines (companies A, B and C). The "UserProfile" class (Fig. 14) contains information about all the users who will have access to the system, with their user characteristics (user code and name) and an associated security profile (an instance of security information composed of a security level, roles and compartments).

A set of security rules has additionally been defined over some classes and attributes by using stereotypes (Fig. 14). These are Sensitive Information Assignment Rules (SIAR) which establish the security privileges needed to access certain information. The secure fact class "Trip" has a SIAR associated with it, indicating that it can be accessed by users with a confidential (or higher) security level (stereotype "SL = C" in "Trip" fact class). The "Passenger" dimension could similarly be accessed by users with a secret (or higher) security level (SL = S); and the "Baggage" dimension by a confidential (or higher) security level (SL = C) and "Security" or "Baggaging" security roles (SR = Security, Baggaging). Several attributes also have fine grain security constraints which permit users with a security role of "Security" to access the attributes "purpose" (in the "Trip" fact class) and "fingerprint", "passportPhoto", "criminalRecord", "suspicious" and "riskIndex" (in the "Passenger" dimension class).

[3] http://analisis.ufro.cl/index.php?option=com_content&view=article&id=49&Itemid=54.
[4] http://analisis.ufro.cl.
[5] http://www.hongkongairport.com/eng/passenger/departure/all/airport-security.html.
[6] http://www.dublinairport.com/gns/at-the-airport/airport-security.aspx.
[7] http://www.dft.gov.uk/.
[8] http://www.faa.gov/.

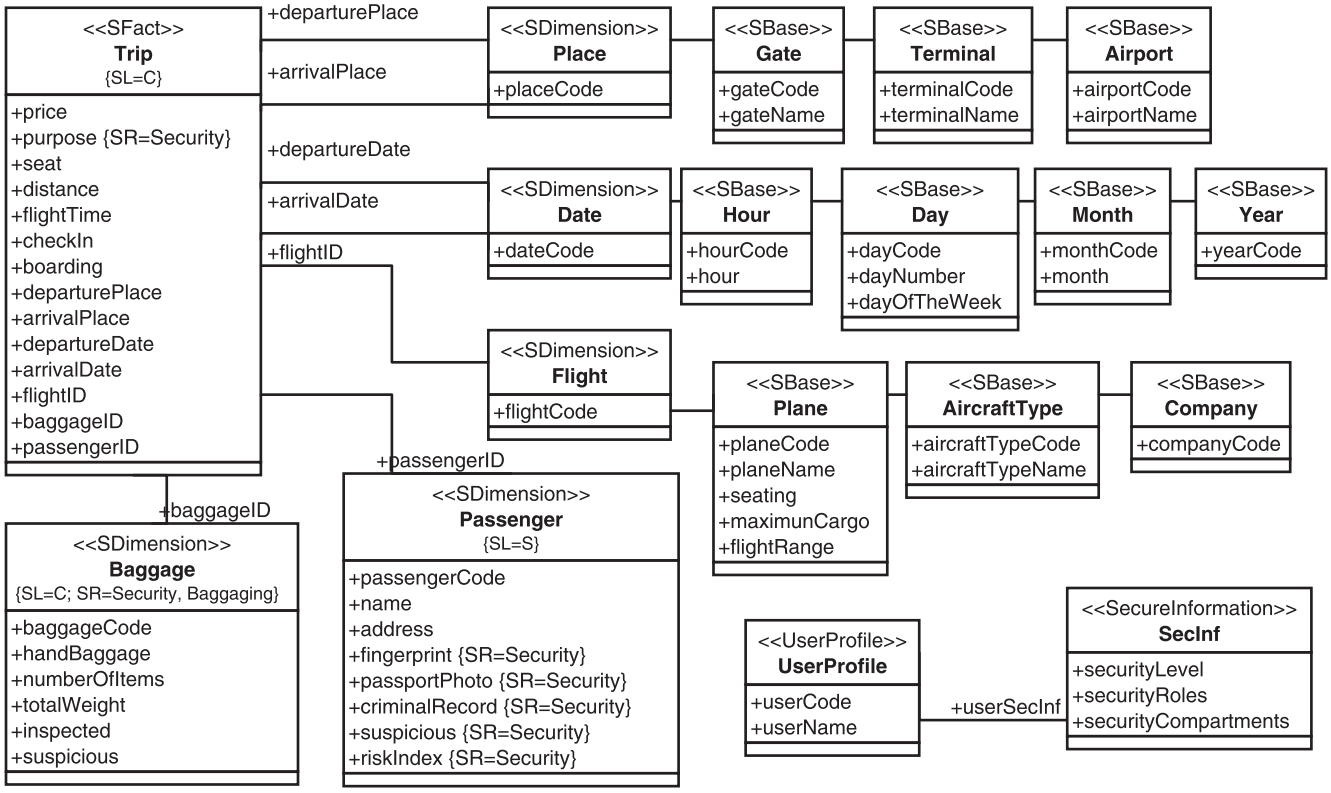


Fig. 14. PIM model for Airport case study.

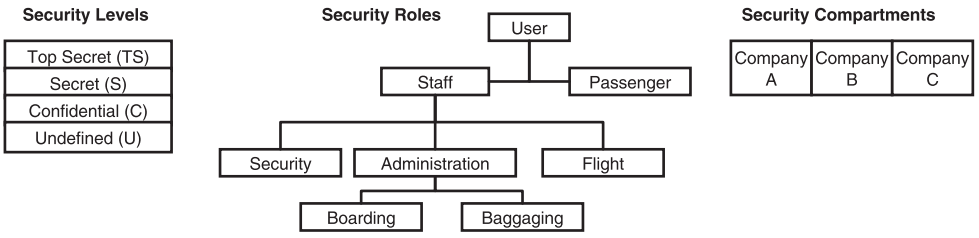


Fig. 15. Security Configuration for Airport case study.

More complex security (SIAR), authorization (AUR) and audit (AR) rules have also been defined by using the "SecurityRule", "AuthorizationRule" and "AuditRule" metaclasses (Fig. 16). The "SIAR_TripPurpose" rule is associated with the "Trip" fact class and involves "Passenger" and "Flight" dimension classes. This rule increases the security requirements for the fact class and the classes involved if the purpose of the trip is "military" ("purpose" attribute). In this case, a security level of "Secret" and a security role of "Security" will be required (expressed as Security Information in the "CATHENSecInf" attribute of the security rule).

The other SIAR rules ("SIAR_PassengerSuspicious" and "SIAR_BaggageSuspicious") are associated with the "Passenger" and "Baggage" dimension classes, and if the established conditions are satisfied then the security requirements needed to access them also increase (i.e., the security level and role required). The "SIAR_BaggageSuspicious" rule checks whether the baggage is suspicious and, if so, increases the security requirements to a "Secret" security level and a "Security" security role, whereas the "SIAR_PassengerSuspected" rule also checks the risk index of the passenger, and if it is more than "5" then a security level of "Top Secret" and a security role of "Security" is required to access the information.

Two authorization rules (AUR) have additionally been defined: a negative authorization rule "AUR_Company" which checks the user's company (security compartment) and denies access to information related to other companies (information about flights and their related base classes "Plane", "AircraftType" and "Company"); and a positive authorization rule "AUR_Passenger" which checks the user name ("name" attribute of "UserProfile") and provides access to his/her basic information (his/her name, address and baggage identification number).

Finally, an audit rule (AR) called "AR_frustratedAttempts" logs all the frustrated access attempts over several multidimensional elements ("Trip" fact class, and "Passenger" and "Baggage" dimension classes). For each frustrated attempt, it stores the information expressed by the "*logInfos*" attribute, which is the identification of the object, the action achieved, the time and the response.

4.2. Sample application of QVT relations

We have used the Secure MD PIM shown in the previous subsection as a starting point to apply the QVT relations defined.

We have selected only some sample elements of the MD PIM used in this case study to show how some of the QVT relations

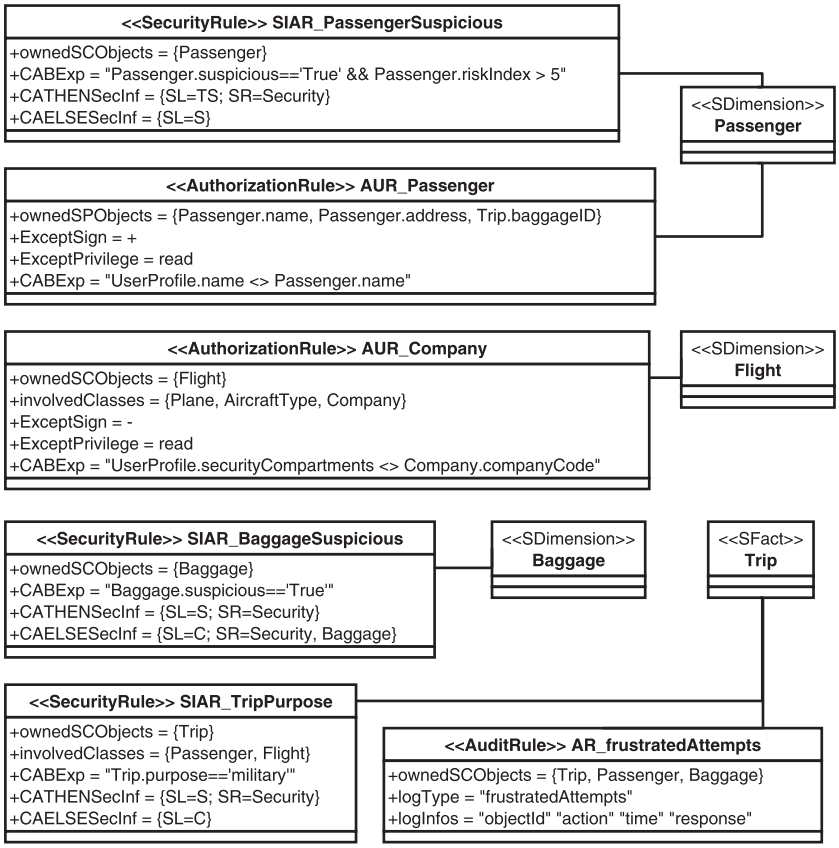


Fig. 16. Security Rules for Airport case study.

defined are applied to obtain the corresponding part in XML of the Secure XML DW. The complete XML Schema generated in the Secure XML DW is shown in the Appendix B, where we have indicated the used QVT relations by means of labels.

The application of the main transformation in our case study begins with the application of the ***SecureDW2SecureMDXML*** relation (Fig. 8), which transforms the complete MD PIM "SecureDW" into an XML Schema, including the root Element "SecureMDXML" of "SecureMDXML_Type" that includes the Security Roles Hierarchy, Security Levels, Security Compartments, Secure Star Package and User Profile. Fig. 17 illustrates the result of applying the *SecureDW2SecureMDXML*.

If we continue with the QVT relations that appear in the ***where*** clause of the *SecureDW2SecureMDXML*, we should now apply the SLevels2LevelsXML, the SRoles2RolesXML, the SCompartments2SCompartmentsXML, the UserProfile2UserProfileXML and the SStarPackage2SStarPackageXML relations. Next, we shall therefore focus solely on the most representativ of these relations, since the others will be applied in a similar way (see Appendix B).

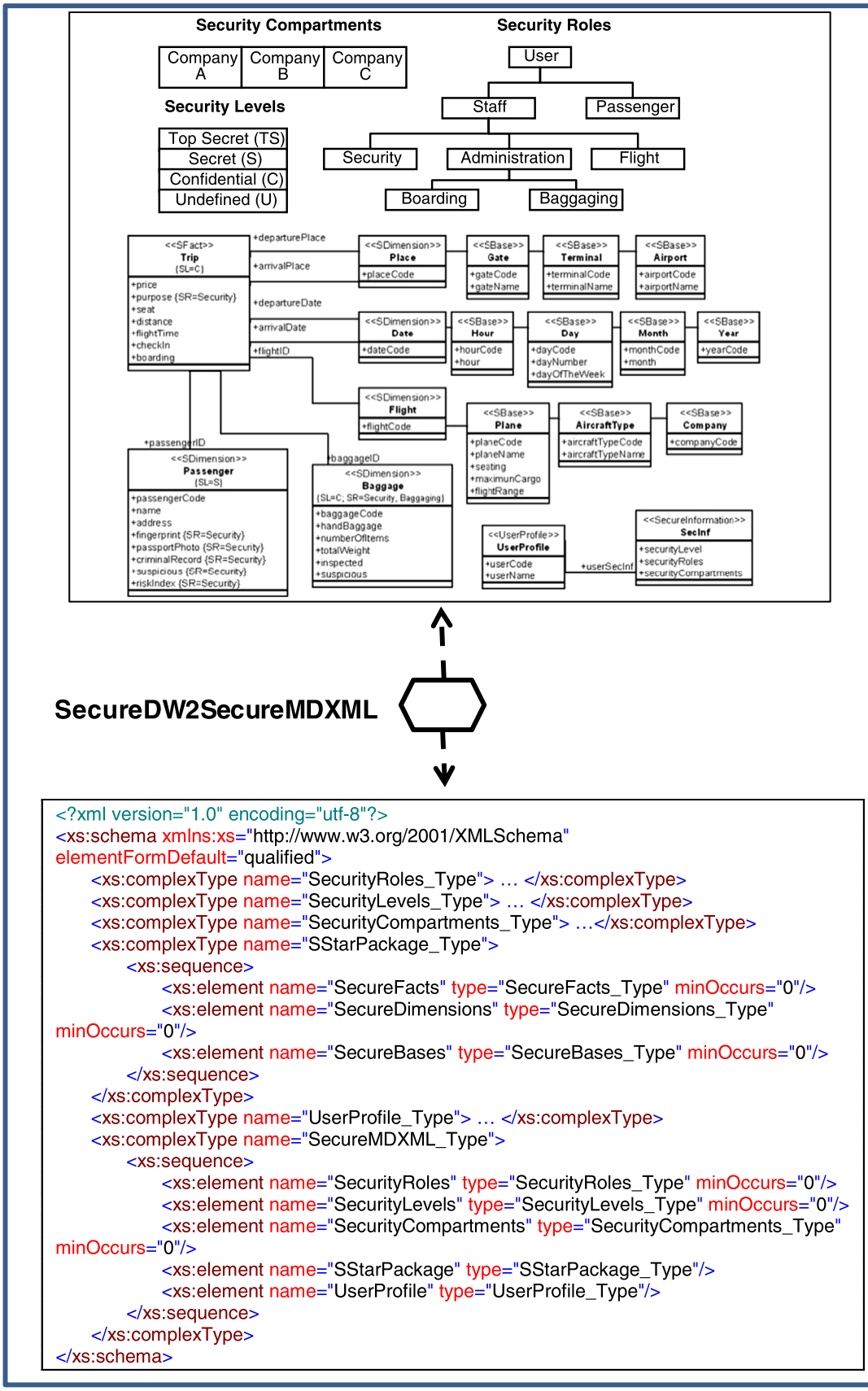
The ***SStarPackage2SStarPackageXML*** relation will be applied to transform the Secure Star Packages of the MD PIM into the XML DW. In our case study, since no Secure StarPackage appears in the Secure MD PIM, we have assumed that there is only one. This will be transformed into the complexType SStarPackage_Type, which contains a sequence with the Secure Facts, Secure Dimensions and Secure Bases XML subelements. In order to generate these subelements, the SFact2FactXML, SDimension2DimensionXML and SBase2BaseXML relations should be applied in the ***where*** clause of the QVT relation.

We shall now show how the ***SFact2FactXML*** relation is applied for the facts of the Secure Star Package (see Fig. 12). The ***SDimension2DimensionXML*** and ***SBase2BaseXML*** relations will be applied in the same way to transform the Dimensions and Bases of the Secure Star Package. According to the *SFact2FactXML* relation, the Secure Fact Class *Trip*, like the other the Secure Fact classes, will be transformed into an XML subelement of the sequence of "SecureFacts_Type" complexType included in the XML Element Secure StarPackage. Each Secure Fact Element will include a complexType with a sequence which includes all the attributes corresponding to the Secure Fact Class, along with an ID attribute. The element can therefore be referenced by other elements by means of an IDREF/S Element.

If we continue with the QVT relations that appear in the ***where*** clause of the SFact2FactXML, we should now apply the SFactAttribute2Element, SecurityRule2Element, AuditRule2Element, AuthorizationRule2Element, SecureInformation2SecureInformationXML and FactDimensionAssociation2IDREFS relations (the other QVT relations not related to security aspects are not with the scope of this paper).

The ***SFactAttribute2SFactAttributeXML*** relation will be applied if Secure Class Attributes are included in the Secure Fact class. They will be transformed by including an XML subelement called "SecureAttribute" in the sequence of the complexType that contains the other properties of the Secure Fact Class. We shall show how the Secure Attribute ***purpose*** will be transformed by applying the SFactAttribute2SFactAttributeXML (the other Secure Class Attributes will be transformed in the same way). The "SecureAttribute" Element includes a complexType sequence with the secure class attribute ***Trip_purpose*** (note that the name of the attribute takes the name of the fact, Trip, as its prefix according to the ***where*** clause of the SFactAttribute2SFactAttributeXML relation) and the corresponding Secure Information properties as XML subelements, which appear in the ***where*** clause of the QVT relation.

The next QVT relation that will be applied is the ***SecurityRule2Element*** relation to generate the security rules associated with the Secure Fact Class Trip, in this case, the SIAR_TripPurpose.




```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
    <xs:complexType name="SecurityRoles_Type"> … </xs:complexType>
    <xs:complexType name="SecurityLevels_Type"> … </xs:complexType>
    <xs:complexType name="SecurityCompartments_Type"> …</xs:complexType>
    <xs:complexType name="SStarPackage_Type">
        <xs:sequence>
            <xs:element name="SecureFacts" type="SecureFacts_Type" minOccurs="0"/>
            <xs:element name="SecureDimensions" type="SecureDimensions_Type"
minOccurs="0"/>
            <xs:element name="SecureBases" type="SecureBases_Type" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="UserProfile_Type"> … </xs:complexType>
    <xs:complexType name="SecureMDXML_Type">
        <xs:sequence>
            <xs:element name="SecurityRoles" type="SecurityRoles_Type" minOccurs="0"/>
            <xs:element name="SecurityLevels" type="SecurityLevels_Type" minOccurs="0"/>
            <xs:element name="SecurityCompartments" type="SecurityCompartments_Type"
minOccurs="0"/>
            <xs:element name="SStarPackage" type="SStarPackage_Type"/>
            <xs:element name="UserProfile" type="UserProfile_Type"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Fig. 17. SecureDW2SecureMDXML Transformation.

A Security Rule will be transformed as an XML subelement of the corresponding Fact Element *Trip*, denominated as the Security Rule at PIM level, *SIAR_Trip_Purpose*. It includes a complexType *Security-Rule_Type* with the following subelements: *ownedSPObjects* with the string value fixed at *"Trip"*; *involvedClasses* with the string value fixed at *"Passenger, Flight"*; *CABExp* which will contain the expression in XPATH."Trip.purpose=='military'; the *CATHEN* that will contain the Security Information if the expression (in XPath) is TRUE (SL = "Secret" and SR = "Security") and the CAELSE subelement that will contain the Security Information if the expression is FALSE (SL = "Confidential").

The next QVT relation that is applied is the ***AuditRule2Element*** relation to obtain the audit rules associated with the Secure Fact Class *Trip*, in this case, the *AR_frustratedAttempts*. An Audit Rule will be transformed as an XML subelement of the corresponding Secure Base, Dimension or Fact Element, denominated as the Audit Rule at PIM level. This subelement includes the following subelements: *ownedSCObjects* with the string value fixed at = "*Trip, Passenger, Baggage*", *logType* with the fixed value *"frustratedAttempts"*, and *logInfos*, also with the string value fixed at = "*objectId*" "*action*" "*time*" "*response*". In the Appendix B the result obtained after applying the *AuditRule2Element* relation is shown.

In our case study, no Authorization Rule is specified for the Secure Fact class Trip, so the AuthorizationRule2Element relation will not be applied.

The last relation to be applied in our case study is the ***FactDimensionAssociation2IDREFS*** relation, where an IDREFS Element Ref_Dimensions is included as a subelement of the Secure Fact Element *Trip* to reference the dimensions.

The application of the other QVT relations defined has been carried out in the same way, in order to validate them (see Appendix B).

5. Related work

In this section, the related work is organized according to the following topics: (1) XML DWs modeling, (2) security integration into the design process and (3) security and access control models for DWs.

5.1. XML DWs modeling

Works focused on the specific modeling of XML DWs can be found in [4]. These works provide different strategies with which to develop an XML Data Warehouse depending on the nature of the XML data sources. If source documents are structured, the solutions are based on traditional DWs which store the XML documents, whereas XML native DWs are better at working with semi-structured documents.

Some proposals use logical models to model systems which consider XML files as data sources that will be transformed and integrated into a non-XML repository. These proposals are based on structured XML documents, which are similar to relational data and can be modeled at the logical level with star [42] and snowflakes schemas [43].

Other work models and analyzes this kind of systems by taking into account a native XML DW in which cubes and dimensions are stored in XML documents. They define models at the conceptual level such as XFact [44] or at the logical level such as [7] or [45,46], which define algebraic manipulation operators that provide some OLAP analysis support. Nevertheless, the XML technology that is applied to DWs has a lower performance when compared with traditional data warehousing systems. This issue is considered by [47,48] or [45,46], which attempt to improve the performance by adding OLAP functionalities to the XQuery language.

Although these proposals are interesting contributions for developing XML DWs they are not prepared for directly including security issues in the development process. These proposals provide us conceptual and logical models specifically created for modeling the structural concepts of DWs considering the XML technology but they do not allow us to include security constraints in these models. Our proposal is also focused on XML DWs permitting the definition of structural elements of the DW at different levels (requirements, conceptual and logical models), and furthermore allow us to include into the models the security constraints needed to assure the DW.

5.2. Secure integration into the design process

Since security has been identified as an important aspect to consider in the development of information systems, many proposals attempt to identify and incorporate these constraints in early development stages [9,12,14].

UMLsec [13] uses UML to define and evaluate security specifications by employing formal semantics (labels, stereotypes, etc.). It is an approach for security in general, including access control policies and the specification of confidentiality and integrity requirements. This proposal uses the majority of UML diagrams: use-case diagrams to capture security requirements; activity diagrams to detail security specifications from use-cases; sequence diagrams to specify security protocols; and deployment diagrams to ensure that security requirements are present in the physical layer communications.

Model Driven Security (MDS) [13,49–51] uses the MDA approach to include security properties in high-level system models. Its authors have also enriched models and model transformation techniques with security capabilities in order to automatically generate secure system architectures. MDS has been applied to several proposals, including UMLsec in which three abstraction levels (requirements, modeling and code) are defined, and tools are provided to assist in the development process, re-engineering, verification and configuration. Within the context of MDS, SecureUML [52] is proposed as an extension of UML for modeling security aspects in a technology independent manner by using a generalized role based access control.

The development methodology TROPOS [53], which is based on i*, has also been improved to permit the modeling of security requirements [54] by using concepts such as secure goals, secure tasks, constraints, etc. The Unified Process has furthermore been extended with specific security activities [55] in order to define security requirements, design, implementation, testing, monitoring and auditing. On the other hand, Mokum [56] is an active object oriented knowledge base system for modeling which permits the specification of security and integrity constraints, and automatic code generation. We have been working on the integration of security in the development process applied to: the development of applications based on Web services (PWSSec process) [57]; processes for requirements engineering (SREP) [58] and product lines (SREPPLine) [59]; a methodology for secure databases that covers requirements gathering, analysis, relational logical design and specific logical design for Oracle Label Security [60]; model driven development of secure systems from secure business processes modeled with extensions of BPMN [61] or UML [62]. These are relevant contributions to secure information systems development but are not specifically focused on DWs. Our Secure MD model has been specifically developed for taking into account all the multidimensional concepts used in a DW design and has been complemented with an access control and audit model that allows us to specify security constraints over these multidimensional elements. Nevertheless, in further works we have considered to extend these proposals (such as UMLsec) with the concepts needed

to model secure DWs and also defining transformations, providing thus other alternatives for PIM model, such as in [22] in which an MDA-based engineering process has been developed to consider an extension of UML as a secure PIM [24], and its transformations to an extension of the CWM (Common Warehouse Metamodel) [23].

5.3. Security and access control models for DWs

DWs manage sensitive information whose security must be ensured by including the necessary security constraints in all layers and operations of the DW [9]. Since end-users work with a MD model when querying a DW (facts, dimensions, classification hierarchies, etc.), security constraints should be defined in terms of MD modeling. There are several interesting initiatives for the inclusion of security in DWs, but they are not conceived for their integration into MD modeling as part of the DW design process, and inconsistent security measures might consequently be defined.

Only Priebe and Pernul [63] propose a complete methodology which allows: security requirements to be specified in an early development stage; their modeling at the conceptual level by using ADAPTed UML [63]; and the implementation of a secure solution in an OLAP tool by extending the MultiDimensional eXpressions (MDX) language. Nevertheless, this proposal focuses on a Mandatory Access Control (MAC) security policy and does not define the connection between models. Other works present DW security models but focus on a particular abstraction level. At the business level, Paim and Castro [64], despite not offering a formal metamodel, include security requirements for DWs. At the conceptual level, security aspects are only included in the modeling by ADAPTed UML [63]. At the logical level, Katic et al. [65] present a security model based on metadata to define user groups and views; Saltor et al. [66] provide an architecture to integrate MAC security policies from data sources; and Rosenthal and Sciore [67] integrate security from the data sources and propagate it to DW design. Other proposals define authorization models and security for DWs [68,69] but only deal with OLAP operations (such us roll-up or drill-down).

To summarize, various interesting methodologies exist for DWs, in addition to several contributions that combine XML and DWs in different ways. Moreover, some proposals can be found in which security issues have been considered in the development of traditional data warehouses. However, to the best of our knowledge, current research lacks approaches with which to consider security in the development of DWs when the target platform is based on XML technology. We therefore propose a methodological approach for the model driven development of Secure XML Data Warehouses, taking advantage of the benefits of applying MDA to this kind of development.

6. Conclusions and future work

In this paper, we have proposed an approach for the model driven development of Secure XML Data Warehouses. Our approach begins by defining the secure conceptual MD model (PIM) represented by means of the secure UML profile called SECDW, independently of the target logical MD model. This PIM is transformed into a secure XML DW, as a logical model (PSM), by applying Model to Model (M2M) Transformations. The transformation described in this paper is executed automatically. However, it is said to be semi-automatic, since once we automatically obtain the target secure PSM of the DW (based on XML) we need to manual refine it to be useful for a specific tool.

In this work, we have focused on the specification of the transformation rules that are necessary to be able to automatically generate not only the corresponding XML structure of the DW from the secure conceptual models of the DW, but also the security rules specified within the DW XML structure, thus allowing us to implement both aspects simultaneously. For the sake of understandability, we have first defined the rationale behind these transformation rules and how they have been developed in natural language, and we have then established them clearly and formally by using the QVT language. In order to validate our proposal we have carried out several examples and case studies.

The great benefit of our proposal is that it is possible to model security requirements together with the DW model in the conceptual model during the early stages of a DW project, and automatically obtain the corresponding implementation for different target platforms, according to different logical data models. We therefore believe that we set the basis for future Business Intelligence platforms, in which traditional DWs should be complemented with both semi-structured and unstructured information from the Web.

We are now working on several different lines of work, in an attempt to overcome some limitations we found. One of these limitations is related to the security constraints expressed in textual form, for example expressed in OCL. The automation of the transformations of the OCL constraints defined at the PIM level should be addressed, to convert them into XPATH language. Another limitation to address is to validate the completeness of our approach by conducting a set of experiments on a more complete set of case studies, by using the case tool that we are developing for supporting the development of Secure XML DW.

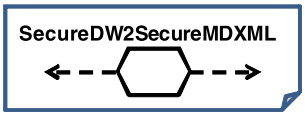
Acknowledgements

We would like to thank the referees for their helpful comments and suggestions that allow us to improve our research. This research has been carried out in the framework of the following projects: CoMobility (TIN2012-31104) financed by the Spanish Ministry of Economy and Competitiveness, MASAI (TIN-2011-22617) and MESOLAP (TIN2010-14860) financed by the Spanish Ministry of Education and Science, SISTEMAS (PII2I09-0150-3135) financed by the "Consejería de Ciencia y Tecnología of the Junta de Comunidades de Castilla-La Mancha", and BUSINESS (PET2008-0136), financed by the Spanish Ministry of Science and Innovation (MCI).

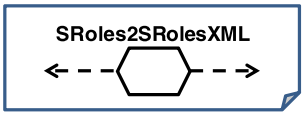
Appendix A. List of acronyms

ACA	Access Control and Audit
ATL	Atlas Transformation Language
BPMN	Business Process Modeling Notation
DW	Data Warehouse
M2M	Model to Model
MAC	Mandatory Access Control
MD	Multidimensional
MDA	Model Driven Architecture
MDS	Model Driven Security
MDX	MultiDimensional eXpressions
MOF	MetaObject Facility
OCL	Object Constraint Language
OLAP	On-Line Analytical Processing
OMG	Object Management Group
PIM	Platform Independent Model
PSM	Platform Specific Model
QVT	Query/View/Transformation
UML	Unified Modeling Language
XML	Extensible Markup Language

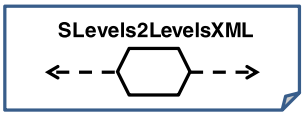
```
<?xml version="1.0" encoding="utf-8"?>
```




```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="SecureMDXML" type="SecureMDXML_Type"/>
```



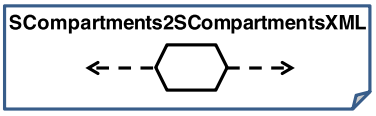

```
    <!-- Security Roles Hierarchy -->
    <xs:complexType name="SecurityRoles_Type">
        <xs:sequence>
            <xs:element name="User" type="xs:string"/>
            <xs:element name="Staff">
                <xs:complexType>
                    <xs:attribute name="fatherRole" fixed="User"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="Passenger">
                <xs:complexType>
                    <xs:attribute name="fatherRole" fixed="User"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="Security">
                <xs:complexType>
                    <xs:attribute name="fatherRole" fixed="Staff"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="Administration">
                <xs:complexType>
                    <xs:attribute name="fatherRole" fixed="Staff"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="Flight">
                <xs:complexType>
                    <xs:attribute name="fatherRole" fixed="Staff"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="Boarding">
                <xs:complexType>
                    <xs:attribute name="fatherRole" fixed="Administration"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="Baggaging">
                <xs:complexType>
                    <xs:attribute name="fatherRole" fixed="Administration"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```



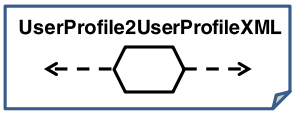

```
    <!-- Security Levels: TS, S, C and U -->
    <xs:complexType name="SecurityLevels_Type">
        <xs:sequence>
            <xs:element name="TopSecret" fixed="TopSecret">
                <xs:complexType>
                    <xs:attribute name="OrderNumber" type="xs:integer" fixed="1"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="Secret" fixed="Secret">
                <xs:complexType>
                    <xs:attribute name="OrderNumber" type="xs:integer" fixed="2"/>
                </xs:complexType>
```

Fig. B1. XML Schema Code for the Airport example.

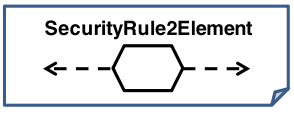
```
            </xs:element>
            <xs:element name="Confidential" fixed="Confidential">
                <xs:complexType>
                    <xs:attribute name="OrderNumber" type="xs:integer" fixed="3"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="Undefined" fixed="Unclassified">
                <xs:complexType>
                    <xs:attribute name="OrderNumber" type="xs:integer" fixed="4"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```



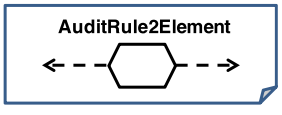

```
    <!-- Security Compartments: Company A, B and C -->

    <xs:complexType name="SecurityCompartments_Type">
        <xs:sequence>
            <xs:element name="Company A" type="xs:string" fixed="Company A"/>
            <xs:element name="Company B" type="xs:string" fixed="Company B"/>
            <xs:element name="Company C" type="xs:string" fixed="Company C"/>
        </xs:sequence>
    </xs:complexType>
```




```
    <!-- User Profile Transformation (Type)-->
    <xs:complexType name="UserProfile_Type">
        <xs:sequence>
            <xs:element name="UserCode" type="xs:integer"/>
            <xs:element name="UserName" type="xs:string"/>
            <xs:element name="SecInf">
                <xs:complexType name="SecureInformation_Type">
                    <xs:sequence>
                        <xs:element name="SecurityCompartments" type="xs:string"/>
                        <xs:element name="SecurityRoles" type="xs:string"/>
                        <xs:element name="SecurityLevel" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```




```
    <!-- Security Rule Transformation: SIAR_TripPurpose -->
    <xs:element name="SIAR_TripPurpose">
        <xs:complexType name="SecurityRule_Type">
            <xs:sequence>
                <xs:element name="ownedSCObjects" type="xs:string" fixed="Trip"/>
                <xs:element name="involvedClasses" type="xs:string" fixed=" Passenger, Flight"/>
                <xs:element name="CABEXP" type="xs:string" fixed="Trip.purpose='military' "/>
                <xs:element name="CABTHEN" type="xs:string" fixed="SL=S; SR=Airport Security"/>
                <xs:element name="CABELSE" type="xs:string" fixed="SL=C"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
```




```
    <!-- Audit Rule Transformation: AR_frustratedAttempts-->
    <xs:element name="AR_frustratedAttempts">
        <xs:complexType name="AuditRule_Type">
            <xs:sequence>
                <xs:element name="ownedSCObjects" type="xs:string" fixed="Trip, Passenger, Baggage"/>
                <xs:element name="logType" type="xs:string" fixed="frustratedAttempts "/>
                <xs:element name="logInfos" type="xs:string" fixed="'objectId','action','time','response' "/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
```

Fig. B1. (*continued*)

```
<!-- Authorization Rule Transformation: AUR_Passenger-->
<xs:element name="AUR_Passenger">
    <xs:complexType name="AuthorizationRule_Type">
        <xs:sequence>
            <xs:element name="ownedSPObjects" type="xs:string" fixed=" Passenger.name,
Passenger.address, Trip.baggageID "/>
            <xs:element name="ExceptSign" type="xs:string" fixed="+"/>
            <xs:element name="ExceptPrivilege" type="xs:string" fixed="read"/>
            <xs:element name="CABEXP" type="xs:string" fixed="UserProfile.name!=Passenger.name'"/>
            <!--ownedSCObjects-->
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

```
<!-- Definition of constraints types: SecurityRule_Type, AuditRule_Type and AuthorizationRule_Type-->
<xs:complexType name="SecurityRule_Type">
    <xs:sequence>
        <xs:element name="CABEXP" type="xs:string"/>
        <xs:element name="CABTHEN" type="xs:string"/>
        <xs:element name="CABELSE" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="AuditRule_Type">
    <xs:sequence>
        <xs:element name="logType" type="xs:string"/>
        <xs:element name="logInfos" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="AuthorizationRule_Type">
    <xs:sequence>
        <xs:element name="Sign" type="xs:string"/>
        <xs:element name="Privilege" type="xs:string" maxOccurs="unbounded"/>
        <xs:element name="CABEXP" type="xs:string"/>
        <xs:element name="SecureInformation" type="SecureInformation_Type" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

```
<!-- Definition of constraints: SecurityRule, AuditRule and AuthorizationRule-->
<xs:element name="SecurityRule" type="SecurityRule_Type"/>
<xs:element name="AuditRule" type="AuditRule_Type"/>
<xs:element name="AuthorizationRule" type="AuthorizationRule_Type"/>
<!-- Transformation of SecureInformation_Type -->
<xs:complexType name="SecureInformation_Type">
    <xs:sequence>
        <xs:element name="SecurityLevel" minOccurs="0"/>
        <xs:element name="SecurityRole" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="SecurityCompartment" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

```
<!--Transformation of SecureFacts_Type complexType, including all the existing Secure Facts-->
<xs:complexType name="SecureFacts_Type">
    <xs:sequence>
        <!--Transformation of Secure Facts -->
        <xs:element name="Trip">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="price" type="xs:string"/>
                    <xs:element name="seat" type="xs:string"/>
                    <xs:element name="distance" type="xs:string"/>
                    <xs:element name="flightTime" type="xs:string"/>
                    <xs:element name="checkIn" type="xs:string"/>
                    <xs:element name="boarding" type="xs:string"/>
```

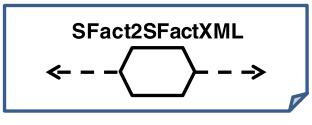
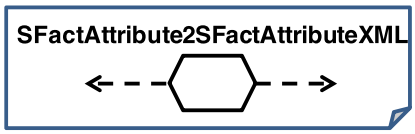
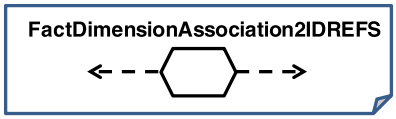


Fig. B1. (*continued*)




```
                    <xs:element name="SecureAttribute">
                        <xs:complexType>
                            <xs:sequence>

                                <xs:element name="Trip_purpose" type="xs:string"/>
                                <xs:element name="SecureInformation">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="SecurityRole" fixed="Security"/>
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="SecureInformation">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="SecurityLevel" fixed="C"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
```




```
                    <xs:element name="Ref_Dimensions" type="xs:IDREFS" minOccurs="0"
maxOccurs="unbounded"/>
```

```
                        <xs:element ref="SIAR_TripPurpose"/>
                        <xs:element ref="AR_frustratedAttempts"/>
                    </xs:sequence>
                    <xs:attribute name="SFact" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
            <!--End of transformation of Secure Facts TRIP-->
        </xs:sequence>
    </xs:complexType>
```

```
    <!--Transformation of SecureDimensions_Type complexType, including existing Secure Dimensions-->
    <xs:complexType name="SecureDimensions_Type">
        <xs:sequence>
            <!--Transformation of Secure Dimension Place-->
            <xs:element name="Place">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="placeCode" type="xs:string"/>
                        <xs:element name="Ref_Facts" type="xs:IDREFS" maxOccurs="unbounded"/>
                        <xs:element name="Ref_Base" type="xs:IDREF"/>
                    </xs:sequence>
                    <xs:attribute name="Place" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
            <!--Transformation of Secure Dimension Date-->
            <xs:element name="Date">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="dateCode" type="xs:string"/>
                        <xs:element name="Ref_Facts" type="xs:IDREFS" maxOccurs="unbounded"/>
                        <xs:element name="Ref_Base" type="xs:IDREF"/>
                    </xs:sequence>
                    <xs:attribute name="Date" type="xs:ID" use="required"/>
                </xs:complexType>
```

Fig. B1. (*continued*)

```
</xs:element>
<!--Transformation of Secure Dimension Flight-->
<xs:element name="Flight">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="FlightCode" type="xs:string"/>
            <xs:element name="Ref_Facts" type="xs:IDREFS" maxOccurs="unbounded"/>
            <xs:element name="Ref_Base" type="xs:IDREF"/>
        </xs:sequence>
        <xs:attribute name="Flight" type="xs:ID" use="required"/>
    </xs:complexType>
</xs:element>
<!--Transformation of Secure Dimension Baggage-->
<xs:element name="Baggage">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="baggageCode" type="xs:string"/>
            <xs:element name="handBaggage" type="xs:string"/>
            <xs:element name="numberOfItems" type="xs:string"/>
            <xs:element name="totalWeight" type="xs:string"/>
            <xs:element name="inspected" type="xs:string"/>
            <xs:element name="suspicious" type="xs:string"/>
            <xs:element name="SecureInformation">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="SecurityLevel" fixed="C"/>
                        <xs:element name="SecurityRole" fixed="Security"/>
                        <xs:element name="SecurityRole" fixed="Baggaging"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="Ref_Facts" type="xs:IDREFS" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Baggage" type="xs:ID" use="required"/>
    </xs:complexType>
</xs:element>
<!--Transformation of Secure Dimension Passenger-->
<xs:element name="Passenger">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="passengerCode" type="xs:string"/>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="fingerprint">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="fingerprint" type="xs:string"/>
                        <xs:element name="SecureInformation">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="SecurityRole" fixed="Security"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="passportPhoto">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="passportPhoto" type="xs:string"/>
                        <xs:element name="SecureInformation">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="SecurityRole" fixed="Security"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
```

Fig. B1. (*continued*)

```
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="criminalRecord">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="criminalRecord" type="xs:string"/>
                            <xs:element name="SecureInformation">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="SecurityRole" fixed="Security"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="suspected">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="suspected" type="xs:string"/>
                            <xs:element name="SecureInformation">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="SecurityRole" fixed="Security"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="riskIndex">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="riskIndex" type="xs:string"/>
                            <xs:element name="SecureInformation">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="SecurityRole" fixed="Security"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="SecureInformation">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="SecurityLevel" fixed="S"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element ref="SecureConstraint" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="Ref_Facts" type="xs:IDREFS" maxOccurs="unbounded"/>
                <xs:element ref="AUR_Passenger"/>
            </xs:sequence>
            <xs:attribute name="Passenger" type="xs:ID" use="required"/>
        </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

```
<!--Transformation of SecureBases_Type complexType, including all the existing Secure Bases-->
  <xs:complexType name="SecureBases_Type">
  <xs:sequence>
    <!--Transformation of Secure Base Gate-->
    <xs:element name="Gate">
      <xs:complexType>
```

Fig. B1. (*continued*)

```
                    <xs:sequence>
                        <xs:element name="gateCode" type="xs:string"/>
                        <xs:element name="gateName" type="xs:string"/>
                        <xs:element name="Ref_Dimension" type="xs:IDREF" minOccurs="0"/>
                        <xs:element name="Ref_Bases" type="xs:IDREFS" minOccurs="0"
maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="Gate" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
            <!--Transformation of Secure Base Terminal-->
            <xs:element name="Terminal">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="terminalCode" type="xs:string"/>
                        <xs:element name="terminalName" type="xs:string"/>
                        <xs:element name="Ref_Bases" type="xs:IDREFS" minOccurs="0"
maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="Terminal" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
            <!--Transformation of Secure Base Airport-->
            <xs:element name="Airport">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="airportCode" type="xs:string"/>
                        <xs:element name="airportName" type="xs:string"/>
                        <xs:element name="Ref_Bases" type="xs:IDREFS" minOccurs="0"
maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="Airport" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
            <!--Transformation of Secure Base Hour-->
            <xs:element name="Hour">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="hourCode" type="xs:string"/>
                        <xs:element name="hour" type="xs:string"/>
                        <xs:element name="Ref_Dimension" type="xs:IDREF" minOccurs="0"/>
                        <xs:element name="Ref_Bases" type="xs:IDREFS" minOccurs="0"
maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="Hour" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
        <!--Transformation of Secure Base Day-->
            <xs:element name="Day">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="dayCode" type="xs:string"/>
                        <xs:element name="dayNumber" type="xs:string"/>
                        <xs:element name="dayOfTheWeek" type="xs:string"/>
                        <xs:element name="Ref_Bases" type="xs:IDREFS" minOccurs="0"
maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="Day" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
        <!--Transformation of Secure Base Month-->
            <xs:element name="Month">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="monthCode" type="xs:string"/>
                        <xs:element name="month" type="xs:string"/>
                        <xs:element name="Ref_Bases" type="xs:IDREFS" minOccurs="0"
maxOccurs="unbounded"/>
```

Fig. B1. (*continued*)

```
                    </xs:sequence>
                    <xs:attribute name="Month" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
        <!--Transformation of Secure Base Year-->
            <xs:element name="Year">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="yearCode" type="xs:string"/>
                        <xs:element name="Ref_Bases" type="xs:IDREFS" minOccurs="0"
maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="Year" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
        <!--Transformation of Secure Base Plane-->
            <xs:element name="Plane">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="planeCode" type="xs:string"/>
                        <xs:element name="planeName" type="xs:string"/>
                        <xs:element name="seating" type="xs:string"/>
                        <xs:element name="maximunCargo" type="xs:string"/>
                        <xs:element name="flightRange" type="xs:string"/>
                        <xs:element name="Ref_Bases" type="xs:IDREFS" minOccurs="0"
maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="Plane" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
        <!--Transformation of Secure Base AircraftType-->
            <xs:element name="AircraftType">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="aircraftTypeCode" type="xs:string"/>
                        <xs:element name="aircraftTypeName" type="xs:string"/>
                        <xs:element name="Ref_Bases" type="xs:IDREFS" minOccurs="0"
maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="AircraftType" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
        <!--Transformation of Secure Base Company-->
            <xs:element name="Company">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="companyCode" type="xs:string"/>
                        <xs:element name="Ref_Bases" type="xs:IDREFS" minOccurs="0"
maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="Company" type="xs:ID" use="required"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <!--Transformation of SecureConstraints-->
    <xs:element name="SecureConstraint">
        <xs:complexType>
            <xs:choice>
                <xs:element ref="SecurityRule"/>
                <xs:element ref="AuditRule"/>
                <xs:element ref="AuthorizationRule"/>
            </xs:choice>
            <xs:attribute name="involvedObjects" type="xs:IDREFS" use="optional"/>
            <xs:attribute name="ownedSPObjects" type="xs:IDREFS" use="optional"/>
            <xs:attribute name="ownedSCObjects" type="xs:IDREFS" use="optional"/>
        </xs:complexType>
    </xs:element>
```

Fig. B1. (*continued*)

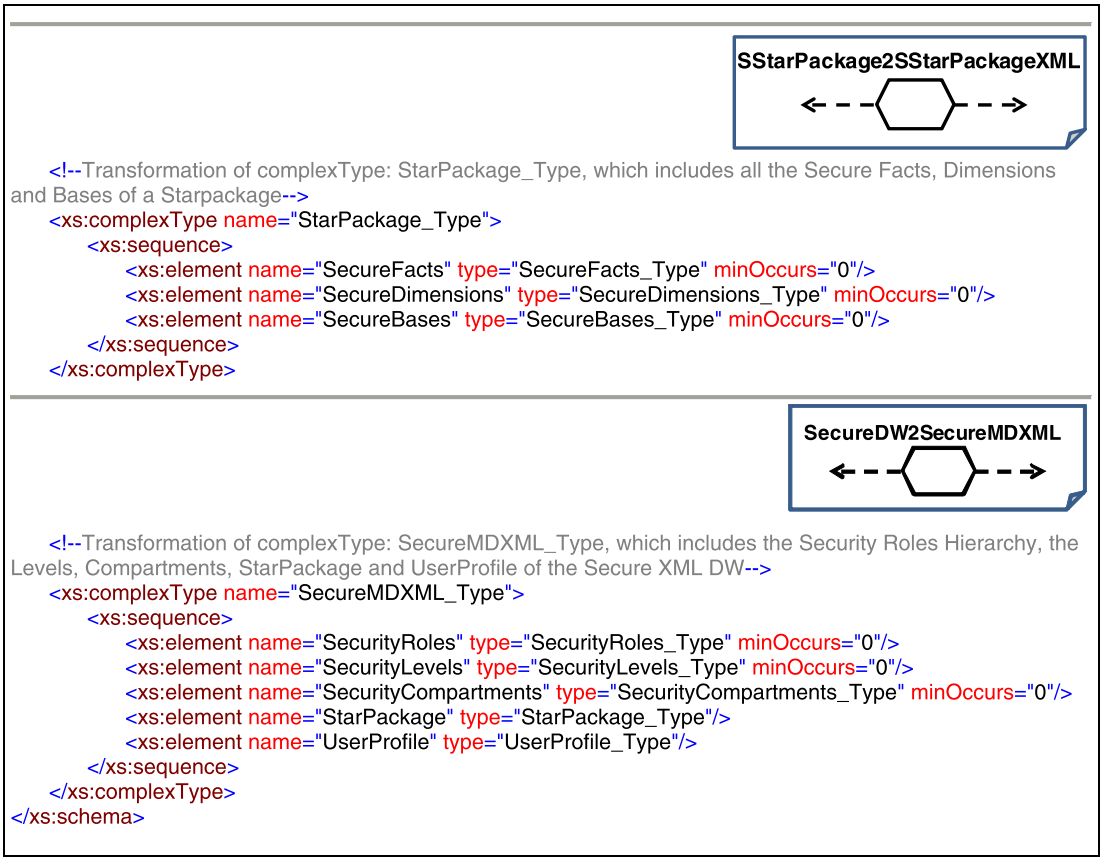


Fig. B1. (*continued*)

Appendix B. Generated XML Schema code of the Secure XML DW

The complete XML Schema code generated after applying the transformation rules is presented as follows.

See Fig. B1.

References

[1] W.H. Inmon, 2.0 – Architecture for the next generation of data warehousing, Morgan Kaufman, 2008.

[2] W.H. Inmon, Building the Data Warehouse, Wiley, 2005.

[3] R. Kimball, L. Reeves, M. Ross, W. Thornthwaite, The Data Warehousing Lifecycle Toolkit, John Wiley & Sons, New York, USA, 2008.

[4] F. Ravat, O. Teste, R. Tournier, G. Zurfluh, Finding an application-appropriate model for XML data warehouses, Information Systems 35 (2010) 662–687.

[5] J.M. Pérez, R.B. Llavori, M.J. Aramburu, T.B. Pedersen, Integrating data warehouses with web data: a survey, IEEE Transaction Knowledge Data Engineering 20 (2008) 940–955.

[6] H. Mahboubi, M. Hachicha, J. Darmont, XML warehousing and OLAP, in: Encyclopedia of Data Warehousing and Mining, second ed., IGI Publishing, 2008, pp. 2109–2116.

[7] O. Boussaid, R.B. Messaoud, R. Choquet, S. Anthoard, X-warehousing: an XML-based approach for warehousing complex data, in: 10th East-European Conference on Advances in Databases and Information Systems (ADBIS), Springer Verlag, ThessalonikiGreece, 2006, pp. 39–54.

[8] M. Golfarelli, S. Rizzi, B. Vrdoljak, Data Warehouse Design from XML Source, in: DOLAP 2001, 2001.

[9] B. Thuraisingham, M. Kantarcioglu, S. Iyer, Extended RBAC-based design and implementation for a secure data warehouse, International Journal of Business Intelligence and Data Mining (IJBIDM) 2 (2007) 367–382.

[10] A. Abelló, J. Samos, F. Saltor, A framework for the classification and description of multidimensional data models, in: 12th International Conference on Database and Expert Systems Applications (DEXA'01), Springer-Verlag, Munich, Germany, 2001, pp. 668–677.

[11] O. Romero, A. Abelló, A survey of multidimensional modeling methodologies, International Journal of Data Warehousing and Mining (IJDWM) 5 (2009) 1–23.

[12] H. Mouratidis, Software Engineering for Secure Systems: Industrial and Research Perspectives, IGI Global, 2011.

[13] J. Jürjens, Secure Systems Development with UML, Springer-Verlag, 2004.

[14] E. Fernández-Medina, J. Jürjens, J. Trujillo, S. Jajodia, Model-driven development for secure information systems, Information and Software Technology 51 (2009) 809–814.

[15] B. Husemann, J. Lechtenborger, G. Vossen, Conceptual data warehouse design, in: Proceedings of the 2nd. International Workshop on Design and Management of Data Warehouses (DMDW'2000), Technical University of Aachen (RWTH), Stockholm, Sweden, 2000, pp. 3–9.

[16] J. Jürjens, H. Schmidt, UMLsec4UML2-Adopting UMLsec to support UML2, in: Technical Reports in Computer Science. Technische Universitat Dortmund, 2011. <http://hdl.handle.net/2003/27602>.

[17] H. Mouratidis, P. Giorgini, Integrating Security and Software Engineering: Advances and Future Vision, IGI Global, 2006.

[18] OMG, Model Driven Architecture Guide Version 1.0.1, 2003.

[19] J.N. Mazón, J. Trujillo, A hybrid model driven development framework for the multidimensional modeling of data warehouses, SIGMOD Record 38 (2009) 12–17.

[20] J.-N. Mazón, J. Trujillo, An MDA approach for the development of data warehouses, Decision Support Systems 45 (2008) 41–58.

[21] J. Trujillo, E. Soler, E. Fernández-Medina, M. Piattini, A UML 2.0 profile to define security requirements for data warehouses, Computer Standard and Interfaces 31 (2009) 969–983.

[22] J. Trujillo, E. Soler, E. Fernández-Medina, M. Piattini, An engineering process for developing secure data warehouses, Information and Software Technology 51 (2009).

[23] E. Soler, J. Trujillo, E. Fernández-Medina, M. Piattini, Building a secure star schema in data warehouses by an extension of the relational package from CWM, Computer Standard and Interfaces 30 (2008) 341–350.

[24] E. Fernandez-Medina, J. Trujillo, R. Villarroel, M. Piattini, Developing secure data warehouses with a UML extension, Information Systems 32 (2007) 826–856.

[25] B. Vela, C. Blanco, E. Fernández-Medina, E. Marcos, A practical application of our MDD approach for modeling secure XML data warehouses, Decision Support Systems 52 (2012) 26.

[26] OMG, QVT. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, OMG, 2008.

[27] A. Kleppe, J. Warmer, W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wesley, 2003.

[28] A. Gerber, M. Lawley, K. Raymond, J. Steel, A. Wood, Transformation: the missing link of MDA, in: H.E. A. Corradini, H.-J. Kreowski, G. Rozenberg (Eds.), ICGT 2002, Springer-Verlag, 2002, pp. 90–105.

[29] K. Czarnecki, S. Helsen, Classification of model transformation approaches, in: 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, Anaheim, 2003.

[30] S. Sendall, W. Kozaczynski, Model transformation: the heart and soul of model-driven software development, IEEE Software 20 (2003) 42–45.
[31] OMG, OCL 2.0 Specification. Version 2.0, in: Object Management Group (OMG), 2005, pp. 185.
[32] SmartQVT, An Open Source Model Transformation Tool Implementing the MOF 2.0 QVT-Operational Language. <http://smartqvt.elibel.tm.fr/>.
[33] mediniQVT, mediniQVT. <http://projects.ikv.de/qvt>.
[34] A. Group, ATLAS Transformation Language. <http://www.eclipse.org/m2m/atl/>.
[35] J.N. Mazón, J. Trujillo, A model-driven goal-oriented requirement engineering approach for data warehouses, in: Advances in Conceptual Modeling – Foundations and Applications, ER 2007 Workshops CMLSA, FP-UML, ONISW, QoIS, RIGiM, SeCoGIS, Auckland, New Zealand, 2007, pp. 255–264.
[36] J.N. Mazón, J. Trujillo, J. Lechtenborger, Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms, Data & Knowledge Engineering 63 (2007) 725–751.
[37] W3C, XML Schema Working Group, XML Schema Parts 0-2: Primer, Structures, Datatypes, 2004. <http://www.w3c.org/TR>.
[38] E. Fernández-Medina, J. Trujillo, R. Villarroel, M. Piattini, Developing secure data warehouses with a UML extension, Information Systems 32 (2007) 826–856.
[39] E. Fernandez-Medina, J. Trujillo, R. Villarroel, M. Piattini, Access control and audit model for the multidimensional modeling of data warehouses, Decision Support Systems 42 (2006) 1270–1289.
[40] B. Vela, C.J. Acuña, E. Marcos, A model driven approach for XML database development, in: International Conference on Conceptual Modeling, Shanghai, China, 2004, pp. 780–794.
[41] P. Hernández, A. Castro, J.N. Mazón, J. Trujillo, C. Cares, Modeling requirements with i* in the development of a data warehouse for a university: the UNIVFRONTERA1-09I project, in: C.U. London (Ed.), iStar Showcase'11, London, 2011.
[42] D. Pedersen, J. Pedersen, T.B. Pedersen, Integrating XML data in the TARGITOLAP system, in: Int. Conference on Data Engineering (ICDE), IEEE Computer Society, 2004, pp. 778–781.
[43] Y. Li, A. An, Representing UML snowflake diagram from integrating XML data using XML schema, in: Int. Workshop on Data Engineering Issues in E-Commerce (DEEC), IEEE Computer Society, 2005, pp. 103–111.
[44] V. Nassis, T.S. Dillon, R. Rajagopalapillai, J.W. Rahayu, An XML document warehouse model, in: Int. Conf. on Database Systems for Advanced Applications (DASFAA), Springer, 2006, pp. 513–529.
[45] K.S. Beyer, D.D. Chamberling, L.S. Colby, F. Ozcan, H. Pirahesh, Y. Xu, Extending XQuery for analytics, in: ACM SIGMOD Infernational Conference on Management of Data, Baltimore, Maryland, 2005, pp. 503–514.
[46] N. Wiwatwattana, H.V. Jagadish, L.V.S. Lakshmanan, D. Srivastava, X^3: a cube operator for XML OLAP, in: International Conference on Data Engineering (ICDE), Istanbul, Turkey, 2007, pp. 916–925.
[47] O. Boussaid, R.B. Messaoud, R. Choquet, S. Anthoard, X-warehousing: an XML-based approach for warehousing complex data, in: East European Conf. on Advances in Databases and Information Systems (ADBIS), Springer, 2006, pp. 39–54.
[48] B.-K. Park, H. Han, I.-Y. Song, XML-OLAP: a multidimensional analysis framework for XML warehouses, Data Warehousing and Knowledge Discovery, LNCS 3589 (2005) 32–42.
[49] D. Basin, J. Doser, T. Lodderstedt, Model driven security: from UML models to access control infrastructures, ACM Transactions on Software Engineering and Methodology 15 (2006) 39–91.
[50] S.H. Houmb, S. Islam, E. Knauss, J. Jürjens, K. Schneider, Eliciting security requirements and tracing them to design: an integration of Common Criteria, Heuristics, and UMLsec Requirements Engineering 15 (2010) 30.
[51] J. Jürjens, P. Shabalin, Tools for secure systems development with UML, International Journal on Software Tools for Technology Transfer (STTT) Archive 9 (2007) 18.
[52] R. Matulevicius, M. Dumas, Towards model transformation between SecureUML and UMLsec for role-based access control, in: Proceeding of the 2011 Conference on Databases and Information Systems VI: Selected Papers from the Ninth International Baltic Conference, DB&IS 2010, IOS Press, Amsterdam, The Netherlands, The Netherlands, 2011, pp. 339–352.
[53] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, A. Perini, Tropos: agent-oriented software development methodology, Journal of Autonomous Agents and Multi-Agent System 8 (2004) 203–236.
[54] P. Giorgini, H. Mouratidis, N. Zannone, Modelling security and trust with secure tropos, in: Integrating Security and Software Engineering: Advance and Future Visions, Idea Group Publishing, 2006.
[55] C. Steel, R. Nagappan, R. Lai, The alchemy of security design methodology, patterns, and reality checks, in: Core Security Patterns: Best Practices and Strategies for J2EE, Web Services, and Identity Management, Prentice Hall, 2005, p. 1088.
[56] R.P. van de Riet, Twenty-five years of Mokum: for 25 years of data and knowledge engineering: correctness by design in relation to MDE and correct protocols in cyberspace, Data & Knowledge Engineering 67 (2008) 293–329.
[57] D.G. Rosado, C. Gutiérrez, E. Fernández-Medina, M. Piattini, Security patterns related to security requirements, in: Workshop on Security in Information Systems (WOSIS'06) in Conjunction with ICEIS'06, INSTICC, Paphos, Cyprus, 2006, pp. 163–173.
[58] D. Mellado, E. Fernandez-Medina, M. Piattini, A common criteria based security requirements engineering process for the development of secure information systems, Computer Standards & Interfaces 29 (2007) 244–253.
[59] D. Mellado, E. Fernández-Medina, M. Piattini, Towards security requirements management for software product lines: a security domain requirements engineering process, Computer Standard and Interfaces 30 (2008) 361–371.
[60] E. Fernandez-Medina, M. Piattini, Designing secure databases, Information and Software Technology 47 (2005) 463–477.
[61] A. Rodríguez, E. Fernández-Medina, M. Piattini, A BPMN extension for the modeling of security requirements in business processes, IEICE Transactions on Information and Systems E90-D (2007) 745–752.
[62] A. Rodriguez, E. Fernandez-Medina, M. Piattini, An MDA approach to develop secure business processes through a UML 2.0 extension, Computer Systems, Science and Engineering 22 (2007) 307–319.
[63] T. Priebe, G. Pernul, A pragmatic approach to conceptual modeling of OLAP security, in: 20th International Conference on Conceptual Modeling (ER 2001), Springer-Verlag, Yokohama, Japan, 2001.
[64] F.R.S. Paim, J. Castro, DWARF: an approach for requirements definition and management of data warehouse systems, in: IEEE International Conference on, Requirements Engineering, 2003, pp. 75–84.
[65] N. Katic, G. Quirchmayr, J. Schiefer, M. Stolba, A. Min Tjoa, A prototype model for data warehouse security based on metadata, in: 9th International Workshop on Database and Expert Systems Applications (DEXA'98). IEEE Computer Society, Vienna, Austria, 1998, pp. 300–308.
[66] F. Saltor, M. Oliva, A. Abelló, J. Samos, Building secure data warehouse schemas from federated information systems, in: H. Bestougeff, J.E. Dubois, B. Thuraisingham (Eds.), Heterogeneous Inf. Exchange and Organizational Hubs, Kluwer Academic Publisher, Dordrecht, The Netherlands, 2002, pp. 123–134.
[67] A. Rosenthal, E. Sciore, View security as the basic for data warehouse security, in: 2nd International Workshop on Design and Management of Data Warehouse (DMDW'00), Sweden, 2000, pp. 8.1–8.8.
[68] E. Weippl, O. Mangisengi, W. Essmayr, F. Lichtenberger, W. Winiwarter, An authorization model for data warehouses and OLAP, in: Workshop on Security in Distributed Data Warehousing, New Orleans, Louisiana, USA, 2001.
[69] L. Wang, S. Jajodia, D. Wijesekera, Securing OLAP data cubes against privacy breaches, in: IEEE Symposium on Security and Privacy, Berkeley, California, 2004, pp. 161–178.