Decision Support Systems 52 (2012) 899–925

Contents lists available at SciVerse ScienceDirect

Decision Support Systems

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

A practical application of our MDD approach for modeling secure XML data warehouses

Belén Vela [a,*], Carlos Blanco [b,1], Eduardo Fernández-Medina [c,2], Esperanza Marcos [a,3]

[a] Rey Juan Carlos University, C/ Tulipán s/n, 28933 Móstoles, Madrid, Spain
[b] University of Cantabria, Av. De los Castros s/n 39071 Santander, Spain
[c] University of Castilla-La Mancha, Paseo de la Universidad, 4-13071 Ciudad Real, Spain

ARTICLE INFO

Available online 19 November 2011

Keywords:
Data warehouse
XML
Security
Model driven development
Transformation rules
Case study

ABSTRACT

Data warehouses are systems that provide useful information to support the decision making process, thus improving organizations' business processes. These systems integrate heterogeneous sources which are not only limited to their internal business data but also include data from the Web, the latter of which have become increasingly more important in the decision making process in recent years. This has motivated the extensive use of XML in the implementation of data warehouses, in a manner which facilitates data and metadata interchange among the heterogeneous data sources from the Web and the data warehouse. However, the business information that data warehouses manage is crucial and highly sensitive, and must be carefully protected. Security is thus a key issue in the design of data warehouses, regardless of the implementation technology used. It is important to note that the data available on the Web requires particular security considerations which 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.

In order to deal with this situation, in this paper we propose a methodological approach for the model driven development of secure XML data warehouses. We also specify a set of transformation rules that are able to automatically generate not only the corresponding XML structure of the data warehouse from secure conceptual data warehouse models, but also the security rules specified within the data warehouse XML structure, thus allowing both aspects to be implemented simultaneously. We additionally introduce our secure XML DW development approach, in which the secure conceptual DW data model, the PIM, is transformed into a secure XML DW, as a PSM, by applying a set of transformation rules. Our proposal is validated through the practical application of our model driven development approach for Modeling Secure XML Data Warehouses to a case study, which is based on a central Airport DW. We first describe the transformation rules defined, then use a step by step illustration to show how they will be applied to the secure conceptual model of the case study to obtain the Secure XML Data Warehouse, thus demonstrating the benefits of our proposal, and finally we analyze how to achieve the secure implementation into commercial database management systems, providing details of the secure implementation in Oracle XML DB 11 g.

1. Introduction

Data Warehouse (DW) systems provide a Multidimensional (MD) view of huge amounts of historical data from heterogeneous operational sources, thus supplying useful and sensitive information which allows decision makers to improve organizations' business processes. 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 can be analyzed (product, customer, time, etc.) by means of hierarchically organized dimension attributes.

Traditional DW systems allow business people to acquire useful knowledge from their organization's data by means of a variety of technologies, such as OnLine Analytical Processing (OLAP) or data mining. However, in order to provide richer insights into the dynamics of today's business, it is currently desirable that the data in the organization be combined with data from outside in order

* Corresponding author. Tel.: +34 91 488 70 03.
E-mail addresses: belen.vela@urjc.es (B. Vela), Carlos.Blanco@unican.es (C. Blanco), Eduardo.FdezMedina@uclm.es (E. Fernández-Medina), esperanza.marcos@urjc.es (E. Marcos).
[1] Tel.: +34 942 206762.
[2] Tel.: +34 926 295300.
[3] Tel.: +34 91 664 74 91.

0167-9236/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2011.11.008

to complement the company's internal data with value-adding information (e.g., retail prices of products sold by competitors). Since the amount of data available on the Web has been growing rapidly over the last decade, Web data prove to be more and more useful for this purpose. The principal problem with data from the Web is that they are rather heterogeneous and complex. DW systems designers confront this problem by employing XML technologies in order to make use of this data [49]. On the one hand, Web warehousing uses XML as a means to ameliorate the extraction and integration of heterogeneous Web data in the DW. On the other hand, document warehousing requires XML to deal with unstructured data in DW systems [48]. In both cases XML is used to implement the MD model underlying the DW by defining the corresponding design artifacts (facts, dimensions, measures, hierarchies and so on) in order to facilitate the interchange of data and metadata among heterogeneous data sources and the DW system [56]. The design of XML DWs is therefore a cornerstone when it is necessary to use Web data in the decision making process, a situation which is becoming more and more frequent. In fact, as is affirmed by Ravat et al. in a recent paper [53], the definition of design approaches for XML DW, which offer methodological frameworks based on the Model Driven Architecture (MDA), is necessary and is one of the most interesting challenges for the future in the area of DW development.

Furthermore, every design issue should be considered in the development process of an XML DW. More specifically, one of the most important design issues is security, which has, to date, been surprisingly overlooked during the development of XML DWs. Considering that the information managed by DWs is frequently highly sensitive, and sometimes refers to personal data (protected by the law in most countries), DWs should be protected from unauthorized information accesses (whatever the implementation platform is). In fact, a key requirement underlying these recently developed data management systems is a demand for adequate security, along with fine-grained flexible authorization models and access control mechanisms (since DWs deal mainly with read operations). Therefore, rather than considering security once the system has been completely built, we believe that security and privacy measures should be integrated into all layers of the DW design, from the early stages of its development as another relevant requirement, signifying that much more robust, secure and platform independent products will be produced [43,63].

Our intention is to develop secure XML DWs by considering confidentiality issues during the entire development process, from an early development stage to the final implementation. Our proposal has therefore been aligned with an MDA architecture in which security models are embedded and scattered throughout high level system models, which are then transformed until their final implementation according to the MDA strategy. MDA can be used for this purpose, since it shares some similarities with traditional MD modeling methods [54]: i) a requirements modeling phase is initially applied in order to obtain an abstract business model (i.e. a Computational Independent Model, CIM), which represents the information requirements (i.e. functional and security requirements) for the DW; ii) a conceptual design phase is carried out, whose output is a technology independent and expressive conceptual MD model for the DW (i.e. a Platform Independent Model, PIM); iii) a logical design phase aims to obtain a technology-dependent model (i.e. a Platform Specific Model, PSM) from the previously defined conceptual MD model, and iv) this logical model is then the basis for the implementation of the DW. In fact, this paper represents a piece of a complex architecture which we have developed for secure DW design and implementation over the last few years. In [66], we enriched a CIM model which extends the i* framework with security requirements [74]. In [20] we defined an access control and audit model for DW, which is independent of the modeling paradigm, and in [21] we then particularized that model through a UML extension for the conceptual modeling of a secure DW (PIM). Following with the architecture down, in [60] we defined a secure star schema for DW (PSM) as an extension of the Common Warehouse Model (CWM), and we also defined an OLAP model with which to represent multidimensional models of secure DW at the logical level and to obtain the implementation of a secure DW through SQL Server Analysis Services [8]. Finally, we have defined an engineering process for developing secure DWs though our architecture [65], and in [9] we presented an approach for the reverse engineering of secure DWs, starting from the implementation. The contribution of this paper is, therefore, the new PSM metamodel for Secure XML DW which is integrated into our architecture, and the definition of the transformations to obtain the Secure XML Data Warehouse (PSM) starting from a conceptual model (PIM) of a secure DW.

Section 2 presents the background and related works, and this is followed by an overview of Model Driven Development (MDD) and MDA in Section 3. In Section 4 we introduce the secure XML DW development approach. The PIM is the secure conceptual DW data model, which will be semi-automatically transformed into a secure XML DW, as a PSM, by applying a set of transformation rules. In Section 5 we describe the case study to which our proposal was applied. Section 6 describes the transformation rules defined, along with their application to the case study selected. Finally, in Section 8, we put forward our main conclusions and present our future work. We also include an Appendix A with the complete generated XML Schema code.

2. Background and related work

In this section the background and related work is organized according to the following themes: (1) DW modeling; (2) XML DW modeling; (3) security integration into the design process; and (4) security and access control models for DWs.

2.1. Data Warehouse modeling

DW modeling differs from traditional Database modeling and needs specific approaches which manage multidimensional concepts (facts, dimensions, measures, hierarchies, etc.). Various interesting methodologies for DWs exist, which can be classified according to how they define the DW.

Data-driven proposals are based on Inmon's DW definition [28], and focus on the development of the DW repository from data sources by using a top–down approach. Some interesting data-driven methodologies are: that of Cabbibo and Torlone [14], which proposes a UML logical model for OLAP systems but is limited to data sources defined by Entity-Relationship (ER) schemas; that of Golfarelli et al. [24,25], which deals with data sources expressed with star schemas and defines a methodology that includes what-if analysis models with UML, conceptual design by using its own notation (Dimensional Fact Models) and logical modeling; and finally, the Unified Process, which has been adapted for DWs [63] with the proposal of a six-step methodology, although models are not specified for each development stage.

User-driven proposals are based on Kimball's DW definition [32] and consider the users' requirements to develop the DW repository by using a bottom-up approach. These proposals do not define formal methods and do not include the typical development stages (modeling at the business, conceptual, logical and physical levels). Some of these user-driven methodologies are: Kimball's proposal [32], which builds the DW as a combination of the data marts and uses an ER notation for the multidimensional modeling; the Data Warehouse Method [17] which proposes five iterative stages and uses UML to analyze user's requirements but does not deal with the conceptual and logical modeling stages in any depth; and Carneiro and Brayner's

methodology [15] which, despite not proposing formal models, manages metadata.

Finally, hybrid proposals attempt to combine the advantages of data and user driven methodologies. Some of these proposals deal with conceptual, logical and physical modeling levels, such as Data Warehouse Quality [10] which improves data quality by enriching metadata; MIDEA [16] which defines several processes to develop the DW and uses Golfarelli's notation; and the Data Warehouse Engineering Process [38] which is based on the unified process and UML, and proposes models for the different DW development stages. The model driven development philosophy has also been applied in several works. Prat and Comyn-Wattiau's methodology [51] builds a conceptual model by using a UML metamodel based on the CWM OLAP package, and obtains logical relational models and physical schemas by applying transformations defined in the Object Constraint Language (OCL). The Model Driven Data Warehouse [50] has been proposed by the Object Management Group (OMG) but is based on CWM and does not deal with multidimensional modeling at the conceptual level.

Various interesting works which do not propose a complete methodology but which deal with modeling at certain abstraction level also exist. Some of these are focused on the requirements definition for DWs such as GRAnD [23] which extend TROPOS with multidimensional elements.

However, the most interesting proposals consider the conceptual level in order to model multidimensional concepts (facts, dimensions, classification hierarchies and so on) in a platform independent manner [26]. There are extensions of the classical ER model [58,67], definitions of their own graphical notation [25], and proposals which use the object-oriented paradigm such as YAM2 [2], based on UML, or [7], based on an object oriented metacube. The multidimensional data can be stored by using different strategies which can principally be classified in OLAP over a relational (ROLAP), multidimensional (MOLAP) and hybrid (HOLAP) approaches. Many modeling proposals do not consider security [1], and only CWM provides a formal metamodel with relational and multidimensional packages.

2.2. XML Data Warehouse modeling

Data Warehouses which are intended to integrate web data through XML documents should consider the specific needs and issues associated with this technology. There are several interesting contributions that combine XML and DWs in different ways, proposing different models and architectures [53].

Some solutions consider XML documents as data sources and attempt to integrate them in a traditional DW repository. XML documents can basically be highly structured data (data-centric XML) which are similar to relational data in which the order is not important (i.e., sales transactions), or less structured data (document-centric XML) which are text-rich documents in which the order is important (i.e., the order of paragraphs in an article). Those proposals that integrate XML documents, provide logical models based on star [47,73] and snowflake [35] schemas, and allow partial analysis capabilities to be applied for data-centric XML.

Other solutions use an XML-native system to store the information without the need for a data conversion. There are various proposals for the conceptual modeling of data-centric and document-centric XML, such as XFact [44]. Nevertheless, the majority are focused on the logical level without providing OLAP capabilities [3,12]. Some OLAP analysis support is provided by proposals focused on data-centric XML which define algebraic manipulation operators [6,11,26,70,72].

A further important issue with regard to XML DWs is that of performance. This is dealt with in several works, which model and analyze this kind of systems by considering the fact that cubes and dimensions are stored in XML documents [12,46], and extend the XQuery language with OLAP capabilities [6,26,72].

2.3. Secure Integration into the design process

Several relevant contributions can be found which concern a complete secure development but which focus on information systems in general without dealing with the specific characteristics of DW systems.

UMLSec [29] uses UML to define and evaluate security specifications by employing formal semantics (labels, stereotypes, etc.). It is principally focused on 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 [4] 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. Within the context of Model Driven Security, SecureUML [37] is proposed as an extension of UML for modeling security aspects in a technology independent manner by using a generalized role based access control. MDS has been applied to several works, including UMLSec [5] 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 [30].

TROPOS methodology [13] supports all the analysis and design activities of the development process using the i* framework [74] as a basis. TROPOS methodology has been extended [22] to permit the modeling and analysis of security requirements by using security concepts (constraints, secure goals, secure tasks, secure resources, ownership, trust of execution and permission, and delegation of execution and permissions). The unified process has been extended with security activities [62], thus allowing the definition of security requirements, architecture, design, implementation, testing, monitoring and auditing. Mokum [68] is an active object oriented knowledge base system for modeling which permits the specification of security and integrity constraints and automatic code generation.

2.4. Security and access control models for DWs

DWs support the decision making process by managing sensitive information. The main security problem for DWs is therefore information confidentiality, as this information is usually accessed by read operations. Since final users work with an MD model when querying a DW (facts, dimensions, classification hierarchies, etc.), security constraints should be defined in terms of MD modeling and considered in all layers and operations of the DW [64]. There are several interesting initiatives for the inclusion of security in DWs, but these were not conceived for integration into MD modeling as part of the DW design process, and, inconsistent security measures may consequently be defined.

With regard to a complete secure DW development, we found only the methodology of Priebe and Pernul [52], in which the authors analyze security requirements from early development stages and finally achieve a secure implementation in commercial OLAP tools by extending the MultiDimensinoal eXpressions (MDX) language with security capabilities which allows multidimensional elements such as cubes, measures, slices and levels to be hidden. These authors represent the DW at the conceptual level by using ADAPTed UML [52] which permits security constraints to be defined through the use of a MAC security policy and roles, but do not establish the connection between models in order to permit automatic transformations.

However, some interesting works dealing with the secure modeling for DWs at certain abstraction levels have been developed. At the

business level there are proposals based on ontologies, business processes, UML, etc. but only Paim and Castro [45] include security requirements, although they do not offer any formal metamodel. At the conceptual level, the modeling of security issues is considered only by the AdaptedUML of Priebe and Pernul [52]. Several interesting logical modeling proposals dealing with security can therefore be found: Rosenthal and Sciore [55] use inference mechanisms to derive security policies from Data Sources and to establish the DW security policy at the logical level, but their models are focused on physical aspects; Saltor et al. [57] provide an architecture focused on integrating MAC security policies from Data Sources; and Katic et al. [31] model security based on metadata by defining data views for each user group, but this approach is focused on DAC policies and do not permit the specification of complex confidentiality constraints. Other proposals define authorization models and security for DWs [33,52,71], but deal solely with OLAP operations (such as roll-up or drill-down).

3. MDD and MDA overview

Model Driven Engineering (MDE) [59] is a software engineering discipline which is focused on models, considering them as the most important element for software development and for the maintenance and evolution of software, through model transformations [41]. This discipline not only offers independence between models but also clearly separates the business complexity from the implementation details by defining several software models at different abstraction levels.

In 2001 the OMG proposed the MDA, a framework for software development which was aligned with MDE. Its main characteristics are the definition of models as first class elements for the design and implementation of systems and the definition of (semi-)automatic mappings between these models.

The four primary goals of MDA are portability, productivity, interoperability and reusability by means of architectural separation of concerns and throughout the complete development lifecycle, covering analysis and design, programming, testing, component assembly, along with coding and maintenance [34,40]. MDA is an approach for software development whose use is enabled by other existing OMG specifications such as the UML, the Meta Object Facility (MOF), CWM and the Query/View/Transformation (QVT).

MDA defines three viewpoints of a system. These viewpoints are modeled with specific models: i) the Computation Independent Model (CIM), which is used by the business analyst, and is focused on the context and requirements of the system without considering its structure or processing, ii) the Platform Independent Model (PIM), which is used by software architects and designers, and is focused on the operational capabilities of a system without considering a specific platform or the technology to be used, and iii) the Platform Specific Model (PSM), which is used by software developers and programmers, and includes details related to the system for a specific platform [27].

This architecture not only proposes a set of models that represent the system at different abstraction levels, but also a software development life cycle [42] with which to: i) capture requirements in a CIM, ii) create a PIM (it is sometimes possible for part of the PIM to be obtained from the CIM), iii) transform the PIM into one or more PSMs, adding platform-specific rules and code that the transformation did not provide; iv) transform the PSM into code, and v) deploy the system in a specific environment. If we consider the importance of the model transformation for this approach, then the definition of metamodels is crucial. Metamodels permit the formal definition of correspondences between concepts of different metamodels (e.g. PIM and PSM models), and therefore provide mechanisms for the definition of model transformation rules. Fig. 1 shows the different models, along with the development sequence.

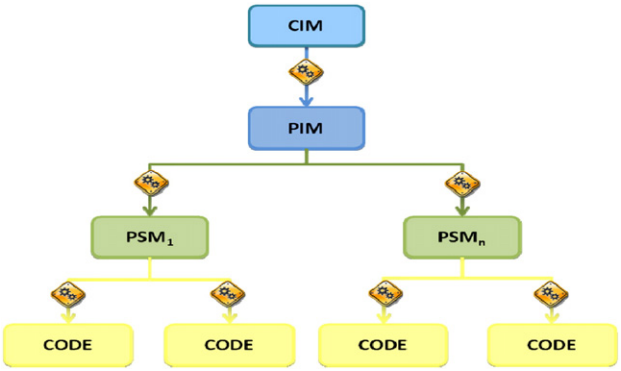

Fig. 1. Model Driven Architecture.

The most widely adopted MDE approach is that of Model-Driven Development (MDD), which focuses on applying MDE principles to the development of software [59,61], i.e. it is the part of MDE that is focused on producing software.

MDD has led to a huge explosion in research, and has been adapted to many areas of software development since the early 2000s. The most intuitive MDA applications are those used to develop databases and data warehouses, since central MDA models (PIM and PSM) fit perfectly with conceptual and logical data models. MDA ideas have been exploited: to redefine the traditional database based application development [36], to develop XML databases [69] and even to evaluate database quality [18]. A complete approach for the development of data warehouses [39] has also recently appeared which proposes: a goal-oriented model as a CIM, a UML profile [38] for the definition of the multidimensional model of data warehouses as a PIM, and an extension of the CWM for the definition of the relational model of the data warehouse as a PSM, in which the transformations between models are performed through QVT rules.

As we have already mentioned in the Related Works section, MDA has also been applied to the integration of security into software systems development through an engineering-based approach. There are many proposals related to this, all of which are brought together under the term Model Driven Security [4].

4. Secure XML DW modeling

In this paper, we use the MDA to define security in the multidimensional modeling of XML data warehouses. We specifically define security specifications in the Conceptual MD Data Model (PIM), independently of the target logical MD model. This Secure Conceptual MD

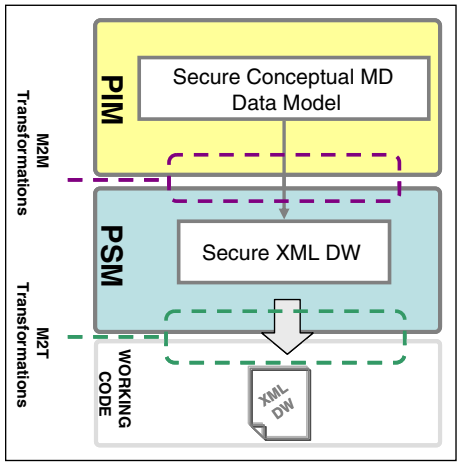

Fig. 2. Development approach for Secure XML DW.

Data Model will be used as a starting point and will be semi-automatically transformed into a Secure XML DW as a logical model (PSM) by applying Model to Model (M2M) Transformations. Finally, a Model to Text (M2T) transformation will generate the code for the Secure XML DW.

For the model driven development of a secure XML DW it is therefore necessary to perform the following tasks (see Fig. 2):

- At the *PIM level*, the Secure Conceptual MD Data 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 UML class diagram for DWs which also permits the specification of security constraints on the model.
- At the *PSM level*, the data logical design is performed, and the selected target platform in which the DW will be implemented is considered. In our case, XML technology will be used for the development of the Secure XML DW. We start from the Secure Conceptual MD PIM obtained at the previous level and apply the M2M Transformations summarized in Section 6 to obtain an XML Schema, conforming to the XML Schema Metamodel.
- Finally, by starting from the Secure XML DW we generate the XML code for the implementation of the Secure XML DW in any secure commercial database management system.

4.1. Secure MD PIM

As previously mentioned, our proposed development approach starts from the conceptual model of the secure MD PIM. In order to define this secure MD PIM, a secure UML profile denominated as SECDW has been developed (for more details, see [21]). SECDW (Fig. 3) allows us to represent specific aspects of DW modeling such as fact, dimension and base classes ("*SFact*", "*SDimension*" and "*SBase*" metaclasses), and their associated measures and attributes ("*SProperty*" metaclass). Multiple classifications or alternative paths of hierarchies can be also defined by relating "*SDimension*" classes with a set of "*SBase*" classes which represent the different aggregation levels (for example, a "Place" dimension could be aggregated by using the levels "City" and "Country").

SECDW is also complemented with an Access Control and Audit (ACA) model [20] which permits the specification of security constraints by using discretional (DAC), mandatory (MAC) and role-based (RBAC) access control security policies. The ACA model allows the subjects and objects of the DW to be classified into security roles, levels and compartments:

- Security roles ("*SRole*" metaclass) organize users into a hierarchical role structure according to the responsibilities of each type of work. For example, a role hierarchy could be composed of an "Employee" role with three sub-roles: "Administrator", "Security" and "Doctor".
- Security levels ("*SLevel*" metaclass) indicate the user's clearance level. For example, a list of security levels might be "Top Secret", "Secret", "Confidential" and "Undefined".
- Security compartments ("*SCompartment*" metaclass) classify users into a set of horizontal compartments or groups, such as different companies, airports, hospitals, etc.

Subjects are classified by using a user profile ("*UserProfile*" metaclass) which defines the information to be stored for each user. This information is composed of the user's security privileges (a "*SecurityInformation*" composed of security roles, compartments and level) and other interesting data such as the user's name, age, etc. Furthermore, the security privileges needed to access different multidimensional elements such as facts, dimensions, bases and attributes, can be specified by attaching security information (security roles, compartments and level) to the element.

The definition of several kinds of security rules related to the multidimensional elements of DWs is also permitted ("*SConstraint*" metaclass):

- Sensitive Information Assignment Rules (SIAR) ("*SecurityRule*" metaclass) specify multilevel security policies and allow sensitive information to be defined for each element in the multidimensional

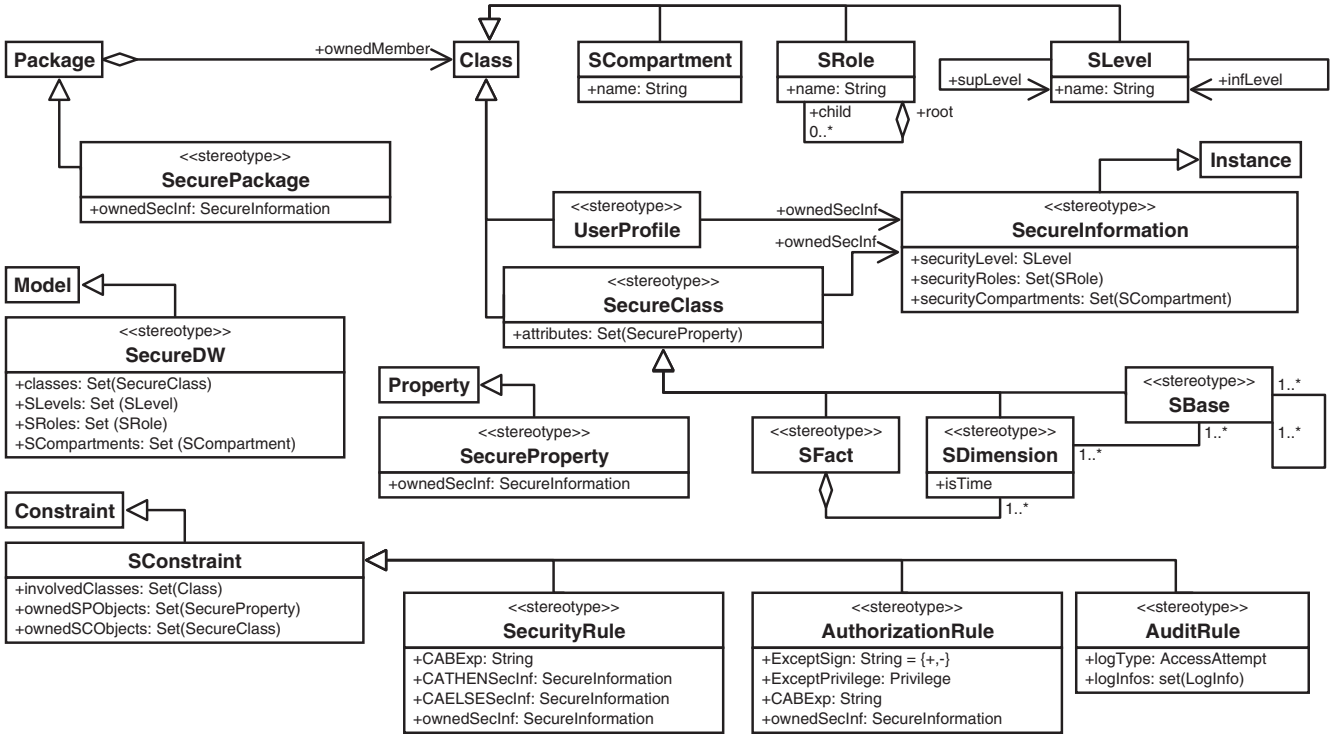

Fig. 3. Conceptual Secure MD Metamodel.

model. This kind of rules can include a condition ("*CABExp*" attribute) to be evaluated at execution time and, depending upon whether or not this is satisfied, to assign different security privileges ("*CATHENSecInf*" or "*CAELSESecInf*" attributes) to the instance of the element involved in the rule. For example, in a hospital DW which manages admissions, diagnosis and patients, a SIAR rule can establish more restrictive security privileges in order to access patients' information as regards a diagnosis related to the oncology area.

- Authorization Rules (AUR) ("*AuthorizationRule*" metaclass) 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 ("*ExceptPrivilege*" attribute which can be read, write, etc.) and the sign describing whether the rule permits or denies access ("*ExceptSign*" attribute). A condition ("*CABExp*" attribute) can also be included in order to select a set of subjects or objects. For example, if age information is included in the user profile it would be possible to establish a condition with which to select users of at least 18 years of age.
- Audit Rules (AR) ("*AuditRule*" metaclass) ensure that authorized users do not misuse their privileges by storing information ("*logInfos*" attribute) about any attempts made to obtain certain information ("*logType*" attribute). For example, an AR rule could check frustrated attempts to access a specific dimension and store information related to the attempt, such us time, subject, etc.

4.2. Secure XML PSM

We propose the use of an XML Schema to represent the PSM level of a Secure XML DW. In this paper, we use a graphical representation of the XML Schema to present the metamodel of our ***Secure XML DW PSM***, which includes both the MD and the security aspects.

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

5. Case study

This section describes one of the case studies carried out to validate our proposal. The practical application of the approach proposed for the Model Driven Development of a Secure XML Data Warehouse shown in Section 4 will be demonstrated by means of the case study selected.

This case study is based on a central Airport DW that manages information such as Trips, Flights and Incidents. The Secure XML DW (PSM) is obtained from the secure conceptual MD data model (PIM).

Fig. 5 shows the secure conceptual multidimensional model (PIM) used in this case study, which is focused on trips. A starpackage 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 "Trip" secure fact class 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 us fingerprints, 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 concerning the number of baggage items, identification codes, weight, whether the baggage has been inspected and whether it is suspicious.

The other secure dimension classes ("Place", "Date" and "Flight") solely 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 example, 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

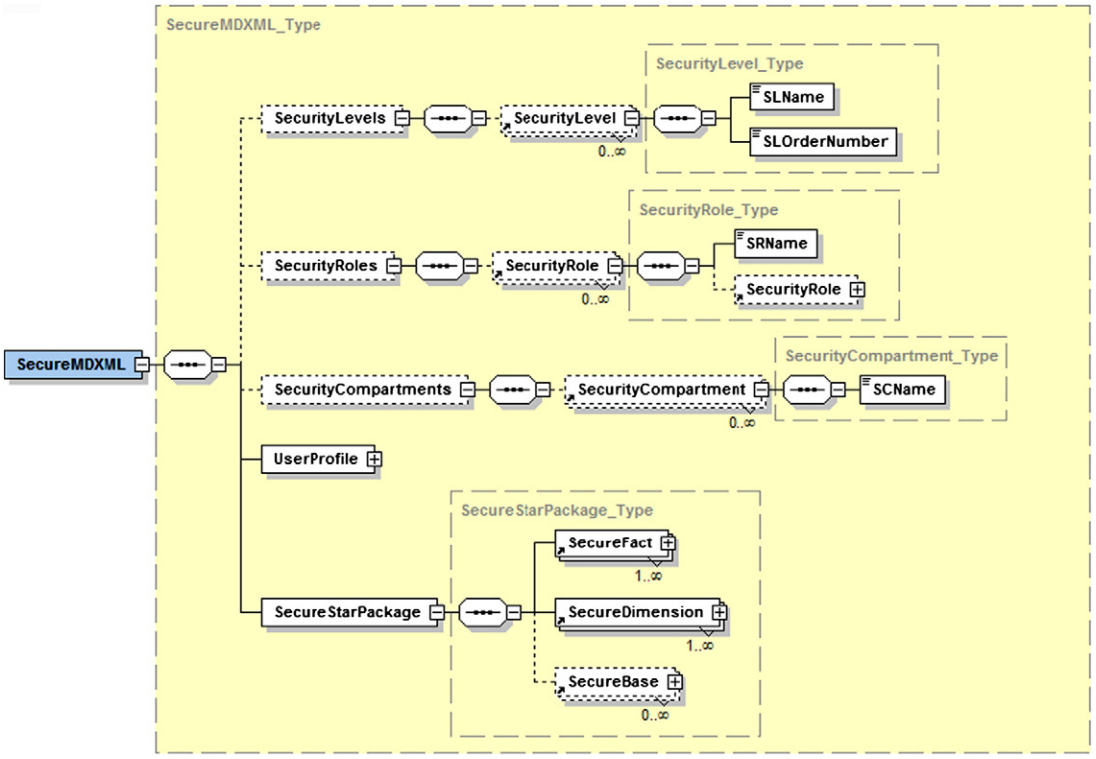

Fig. 4. Secure XML DW PSM-Metamodel.

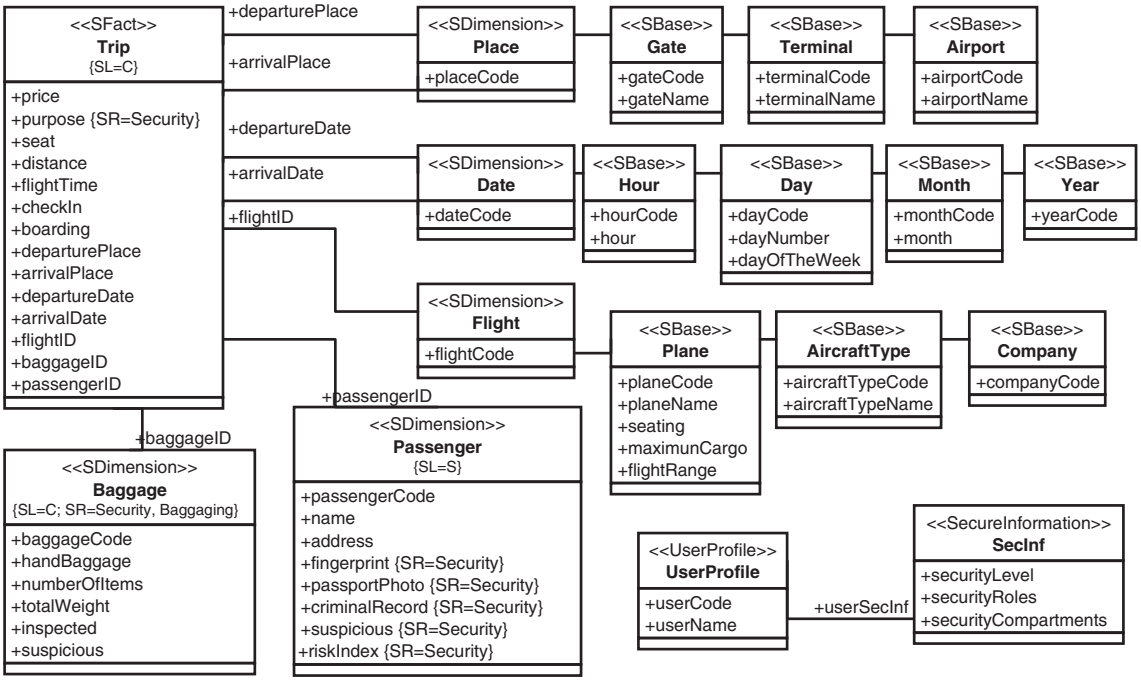

Fig. 5. PIM model for Airport case study.

"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. 6 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 the different airlines (companies A, B and C). The "UserProfile" class (Fig. 5) 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).

The security privileges needed to access the information of facts, dimensions, bases or attributes, can be specified in the conceptual model. These security constraints are defined by including the security roles (SR), compartments (SC) and level (SL) required to access its information in the element concerned (Fig. 5). In this case study, the secure fact class "Trip" can be accessed by users with a confidential (or higher) security level, and this is indicated with the value SL = C in the "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 "purpose" (from the "Trip" fact class) and "fingerprint", "passportPhoto", "criminalRecord", "suspicious" and "riskIndex" (from the "Passenger" dimension class) attributes.

More complex security (SIAR), authorization (AUR) and audit (AR) rules have been also defined by using the "SecurityRule", "AuthorizationRule" and "AuditRule" metaclasses (Fig. 7). 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

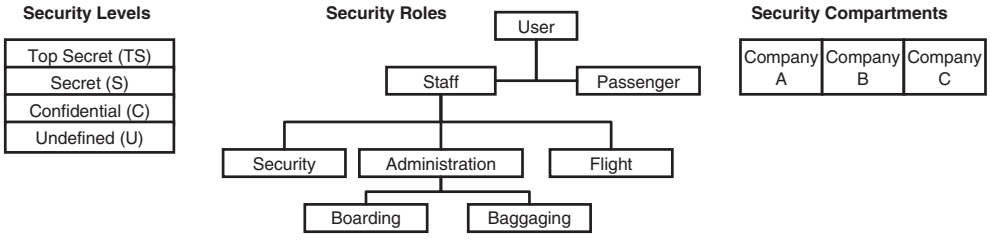

Fig. 6. Security Configuration for Airport case study.

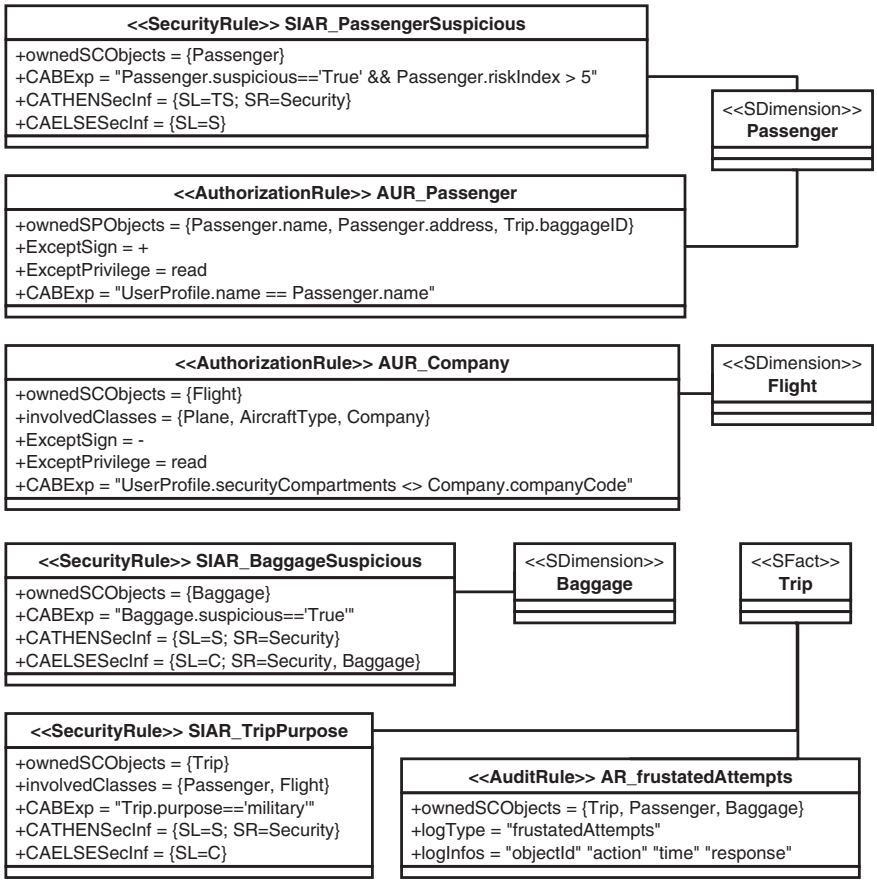

Fig. 7. Security Rules for Airport case study.

"Security" security role, while the "SIAR_PassengerSuspected" rule also checks the risk index of the passenger. If this is more than "5" then a security level of "Top Secret" and a security role of "Security" are 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 (name, address and baggage identification number).

Finally, an audit rule (AR) called "AR_frustatedAttempts" 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.

6. Mappings from PIM to PSM

In the same way that methodologies for relational or object-relational databases propose certain rules for the transformation of a conceptual schema into a logical schema, in this subsection we propose the mappings from the secure MD PIM to the XML Schema for the Secure XML DW. The basis for this is the work of [69], in which the different mappings used to obtain the schema of a secure XML Database were defined but did not take into consideration security aspects for MD modeling issues. The mapping rules for the transformation of a secure multidimensional model (PIM level) into an XML Schema model (PSM level) will now be described. For each of the components of the source model, i.e. the Secure MD PIM, the transformation rule to obtain the corresponding element/s will be specified in the target model, which, in our case, will be the Secure XML PSM, i.e., the XML Schema of the Secure XML Data Warehouse.

In this section we also provide a step by step description of the application of the transformation rules to the Secure MD PIM (Fig. 5) of the case study described in Section 5 to obtain the secure XML Data Warehouse. A description of the dependencies between the different transformation rules that need to be applied in order to obtain the secure XML Data Warehouse is shown at the end of this section. The complete generated XML Schema of the Secure XML DW is shown in the Appendix A.

6.1. Transformation of the Secure MD PIM

The complete MD PIM will be transformed into an XML Schema which will include the root Element *SecureMDXML* of *SecureMDXML_Type*, which includes the Security Levels, Security Roles Hierarchy, Security Compartments, Star Packages and User Profile. Fig. 8 illustrates the result of applying the transformation defined.

6.2. Transformation of Security Levels

The Security Levels defined for a specific DW will be transformed into an XML Element called *SecurityLevels,* including a *SecurityLevel_Type* complexType sequence in which all the Security Levels defined as Subelements will be denominated as the Security Level at a fixed value. In our case, we will have the following Security Levels: *Top Secret*, *Secret*, *Classified* and *Unclassified*. Each of them will contain the attribute: *order number* (beginning with 1, as the highest security level), i.e., the respective values for this attribute are: TopSecret:1,

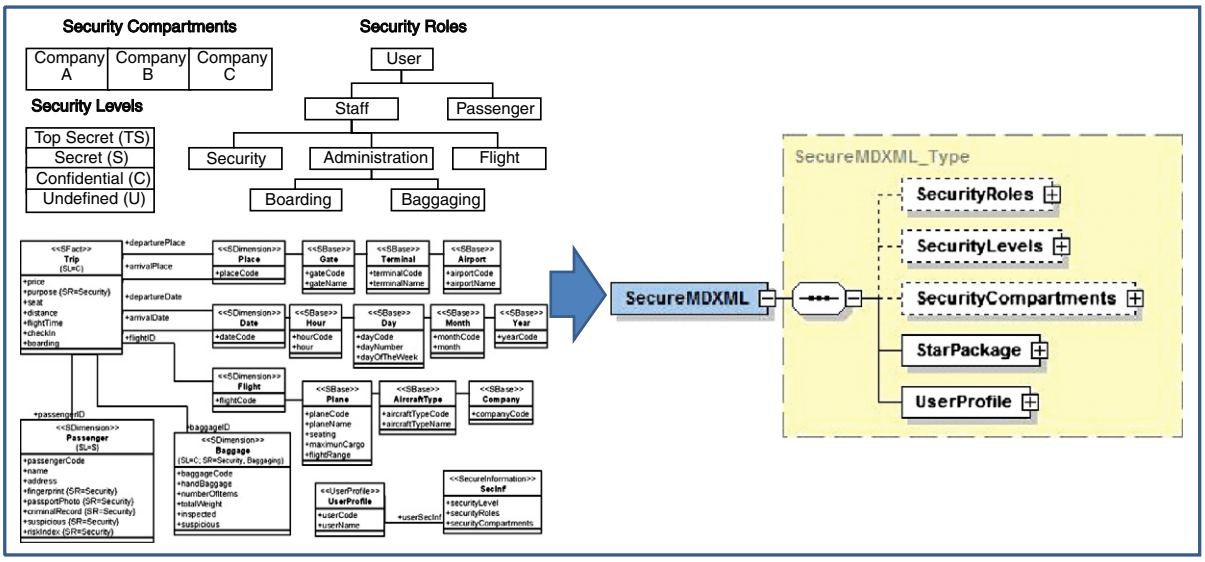

Fig. 8. Mapping the Secure MD PIM.

Secret:2, Confidential:3, Unclassified:4. Fig. 9 illustrates the result of applying the transformation defined.

6.3. Transformation of Security Compartments

The set of Security Compartments defined for a DW in the conceptual model will be transformed into an XML Element called *SecurityCompartments,* including a *SecurityCompartments_Type* complexType sequence with all the defined Security Compartments as Subelements, denominated as the Security Compartment as a fixed value. In our case, we will have the following Security Compartments: *Company A*, *Company B* and *Company C*. Fig. 10 shows the result of applying the defined mapping.

6.4. Transformation of Security Roles

The Security Roles Hierarchy defined for a Data Warehouse will be transformed into an XML Element called *SecurityRoles,* including a *SecurityRole_Type* complexType sequence with all the defined Security Roles as Subelements denominated as the Security Roles as a fixed value (*name* attribute). Moreover, since the Security Roles are represented by means of a hierarchy, each Subelement will contain an Element reference to its parent (*fatherRole* attribute) with the name of its father in the hierarchy as a fixed value, i.e., the security role in which it is included. Fig. 11 illustrates the result of applying this transformation.

6.5. Transformation of the Secure Information

The Secure Information class, which contains three security attributes (SecurityLevel, SecurityRoles and SecurityCompartments) associated with a specific element of the DW (e.g. Secure Fact, Dimension or Base), will be transformed into an XML Element called *SecureInformation* including a complexType denominated as (SecureInformationType) with three Subelements:

- SecurityLevel_Type SecurityLevels with the corresponding attributes as XML Subelements.
- SecurityRoles_Type SecurityRoles with the corresponding attributes as XML Subelements.
- SecurityCompartments_Type SecurityCompartments with the corresponding attributes as XML Subelements.

6.6. Transformation of Secure Packages

Each Secure Star Package of the MD PIM will be transformed into an XML Element called *StarPackage* within the *SecureMDXML* element, including a complexType sequence named *StarPackage_Type* with the Fact, Dimension and Base XML Subelements, each of which will contain the corresponding Security Information and Security Constraints as XML Subelements. We assume that there is only one Secure Star Package if no Star Packages appear in the Secure MD PIM. In our case study, since no StarPackage appears in the Secure MD PIM, we have assumed that there is only one. Fig. 12 illustrates the result of the application of this transformation.

6.7. Transformation of the User Profile

The User Profile class will be transformed into a *UserProfile* XML Element of a *UserProfile_Type* complexType sequence, whose Subelements are: its code, the name, the specific class attributes and the SecureInformation Subelement of SecureInformation_Type. The User Profile class defined in the MD PIM of our case study (Fig. 5) includes the following Subelements: UserCode, the UserName and the SecInf

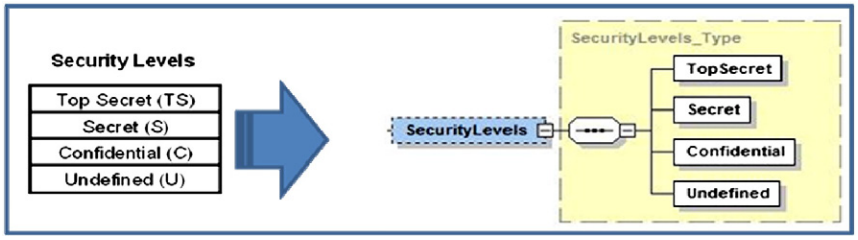

Fig. 9. Mapping the Security Levels.

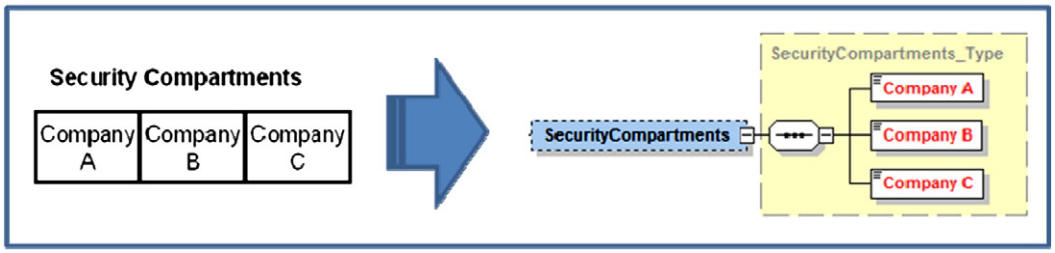

Fig. 10. Mapping the Security Compartments.

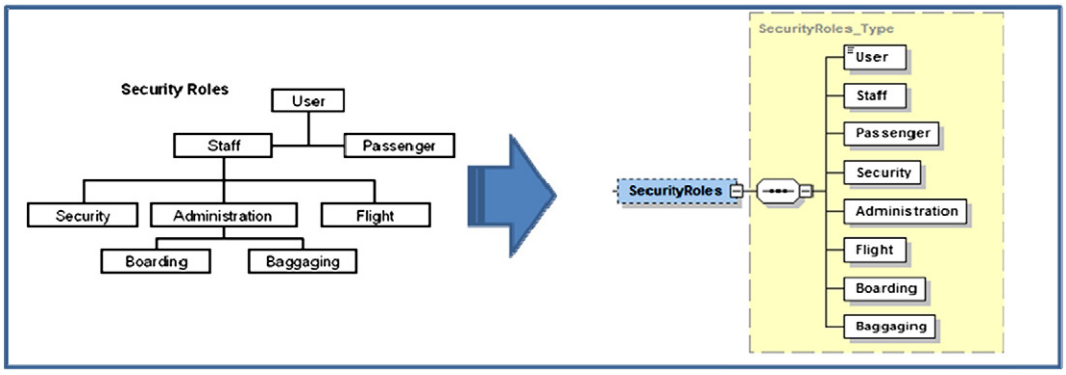

Fig. 11. Mapping the Security Roles.

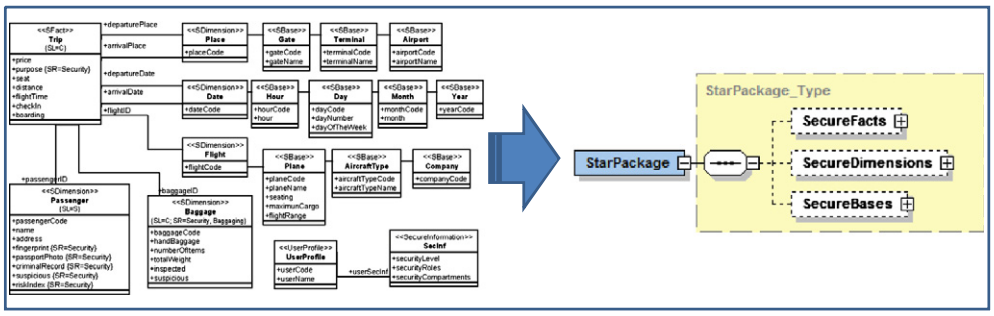

Fig. 12. Mapping the Secure Star Package.

of SecureInformation_Type, including the specific Security Level, Security Roles and Security Compartments for the User Profile as Subelements. Fig. 13 shows the result of this transformation.

6.8. Transformation of Secure Facts

Each Secure Fact will be transformed into an XML Subelement called the Secure Fact included in the *SecureFacts* Subelement of *SecureFacts_Type* defined in the Secure Star Package. The Subelement corresponding to the Secure Fact includes a complexType sequence with an ID attribute, signifying that the element can be referenced by other elements by means of an IDREF/S Element, with the (secure) Fact attributes as XML Subelements and an IDREFS Element to reference the Dimensions. This XML Subelement will also include all the Security Constraints associated with this Secure Fact, that is, security information which specifies the security privileges needed to access the fact class (*SecureInformation*), and the security rules that affect the fact (security rules, authorization rules and audit rules).

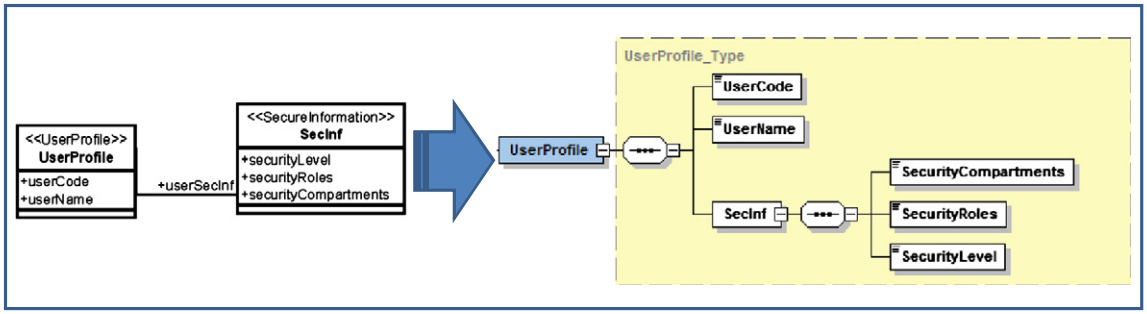

Fig. 13. Mapping the User Profile Class.

In our validation case study, the Secure Fact Class *Trip* will be transformed into an XML Subelement of the *SecureFacts_Type* complexType. The *Trip* Element will include a complexType sequence with an ID attribute. The element can therefore be referenced by other elements by means of an IDREF/S Element. The attributes of the *Trip* fact are represented as XML Subelements (*price*, *seat*, etc.), while the attributes with associated security information (such as the *Purpose* attribute) are defined by using *SecureAttribute* elements (explained in more detail later). The dimensions in which the *Trip* fact can be classified are referenced as IDREFS in the *Ref_Dimensions* Element.

The security privileges needed to access the *Trip* Secure Fact Class (a security level of Confidential) are indicated as a *SecurityInformation* element composed of a security level of Confidential. Furthermore, the security rules associated with the Secure Fact Class are represented with specific elements, in this case, the *SIAR_TripPurpose* and *AR_FrustedAttempts*, detailed later in Subsection 6.13. Fig. 14 shows the application of the corresponding mapping for the *Trip* Secure Fact class.

6.9. Transformation of Secure Dimensions

Each Secure Dimension will be transformed into an XML Subelement called the Secure Dimension included in the *SecureDimensions* Subelement of *SecureDimensions_Type* defined in the Secure Star Package. This Subelement includes a complexType sequence with the name of the Dimension attribute as an XML Subelement, an IDREFS Element to reference the Facts and an IDREF Element to represent the association with the Base XML Element. The XML Subelement associated with the Secure Dimension will also include all the Security Constraints. As all the Secure Dimension classes will be transformed in the same way, we shall focus on the *Baggage* Secure Dimension Class, which includes a complexType sequence with an ID attribute and the Dimension attributes (in this Dimension there is no Secure Dimension Attribute) as XML Subelements (for example, the *baggageCode*, *handBaggage*, *numberOfItems*, etc.), and the Secure Information of the Secure Dimension Class and an IDREFS Element *Ref_Facts* to reference the Facts. Fig. 15 shows the result of applying this transformation to the *Baggage* Dimension Class.

6.10. Transformation of Secure Bases

Each Secure Base will be transformed into an XML Subelement called the Secure Base included in the *SecureBases* Subelement of *SecureBases_Type* defined in the Secure Star Package. This Subelement includes a complexType sequence with an ID attribute, all the (secure) Base attributes (all of which are XML Subelements), an IDREF Element with which to reference the Dimension and an IDREFS Element with which to reference the associated Bases. This XML Subelement, which is associated with the Secure Base, will also include the Security Constraints. As all the Secure Base Classes will be transformed in the same way, we shall focus on the Secure Base Class *Plane*. The *Plane* Element will include a complexType sequence with an ID attribute. The element can therefore be referenced by other elements by means of an IDREF/S Element, all (secure) Base attributes, such as *planeCode*, *planeName*, *seating*, etc. (all of which are XML Subelements), and an IDREFS Element *Ref_Bases* to reference the associated Secure Bases. Fig. 16 illustrates the result of this mapping.

6.11. Transformation of secure class attributes

The secure attributes of the classes are attributes with associated security information, which indicates the security privileges (security level, roles and compartment) needed to access the information. Each secure attribute will be transformed into an XML Subelement called SecureAttribute in the complexType that contains the other properties of the corresponding class. This SecureAttribute Element includes a complexType sequence with the secure class attribute and the corresponding Secure Information properties as XML Subelements. For example, in the case of the *Trip* Secure Fact Class (any other Secure Class Attribute will be transformed in the same way), the *purpose* attribute requires a security role of *Security* and is thus represented as a SecureAttribute Element which includes a complexType sequence with the secure class attribute *purpose* and the corresponding Secure Information properties (a security role of Security) as XML Subelements (see Fig. 17).

6.12. Transformation of associations

Various kinds of associations have been considered for the transformation of the associations that appear in the Secure MD PIM:

- Fact – Dimension
  The Secure Fact XML Element includes an IDREFS Element to reference the 0..n Dimensions and the Secure Dimension XML Element includes an IDREFS Element to reference the 1..n Facts.
- Dimension – Base
  The Secure Dimension XML Element includes an IDREF Element that references the corresponding Base (optional) and the Secure Base Element includes an IDREFS to the Dimension (obligatory).

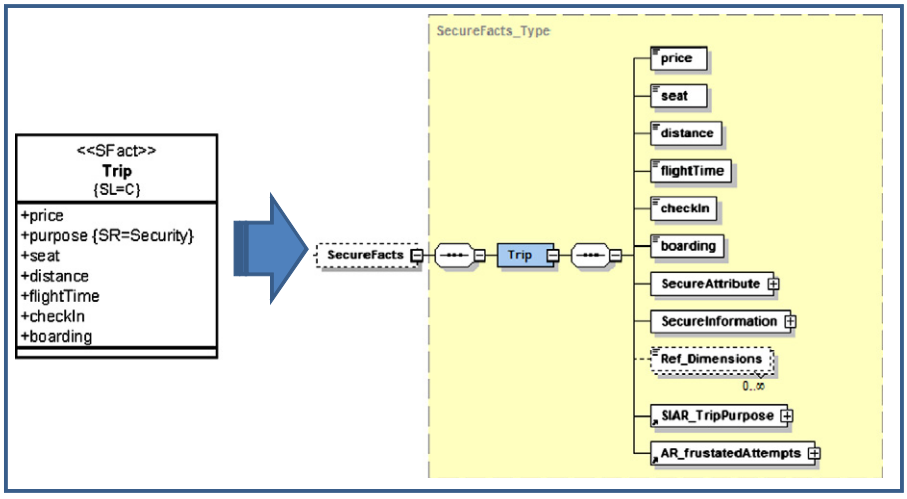

Fig. 14. Mapping the Secure Fact Classes.

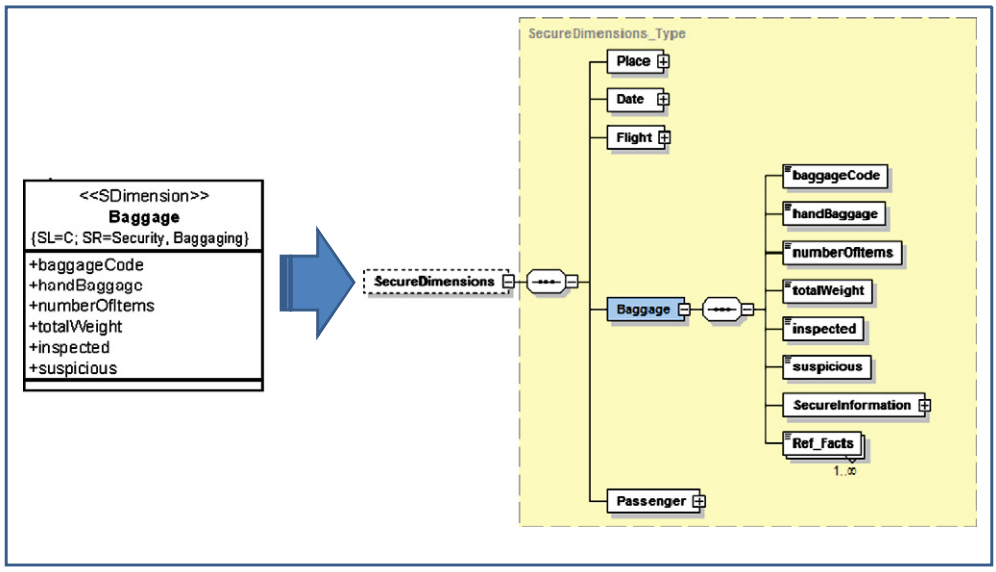

Fig. 15. Mapping the Secure Dimension Classes.

- Base – Base
  The Secure Base XML Element includes an IDREFS Element that references the associated Bases.

6.13. Transformation of Secure Constraints

There are several approaches for the representation of the Security Constraints in XML [19]. However, as our approach is MDA-based, they must be dealt with at two different levels: PIM and PSM. It is first, therefore, necessary to specify the security constraints at a PIM level with the OCL language and to then transform them, as shown as follows, into XPath expressions, which are supported by most XML Database Management Systems (both native and XML-enabled).

Three kinds of security constraints can be defined at the PIM level: security, authorization and audit rules. These contain three optional attributes which represent the elements affected by the rule: ownedSCObjects which indicates a set of secure facts, dimensions or base classes that are directly affected by the rule, ownedSPObjects which indicates a set of attributes, and involvedClasses which allows a list of fact, dimension or base classes to be specified, which are affected by the rule if they are queried together. For example, querying Flights related to Passengers and Baggage could be more sensitive that querying this information separately, and a security rule could therefore be defined to increase the security privileges required to access this information (by using an involvedClasses = Flights,Passenger,Baggage attribute).

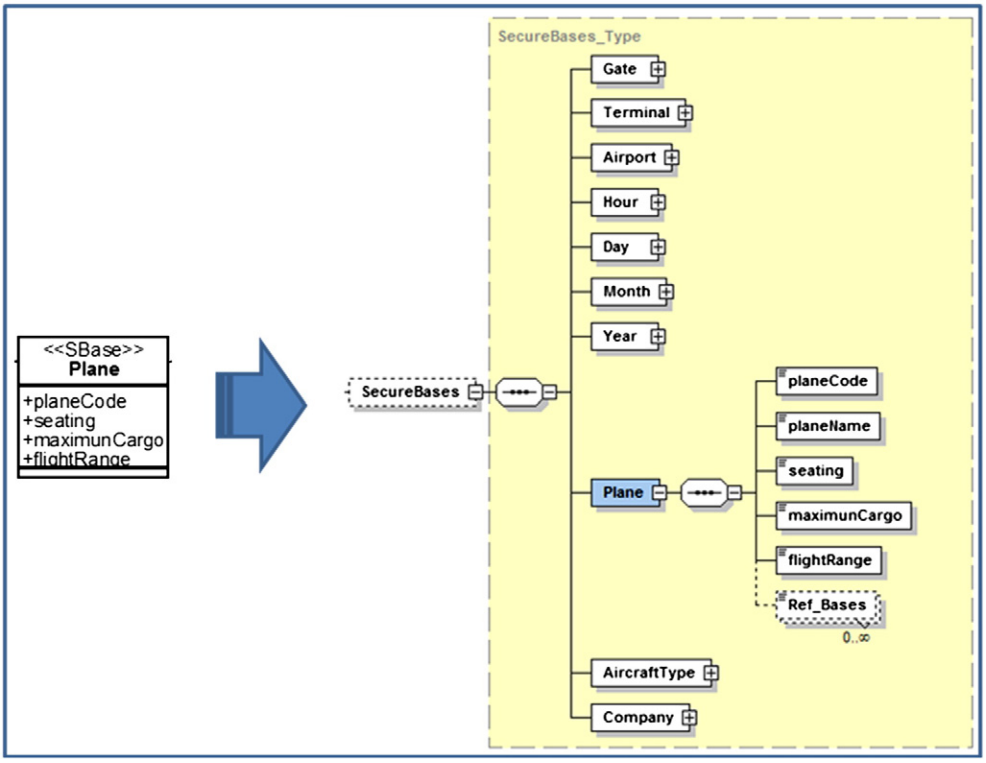

Fig. 16. Mapping the Secure Base Classes.

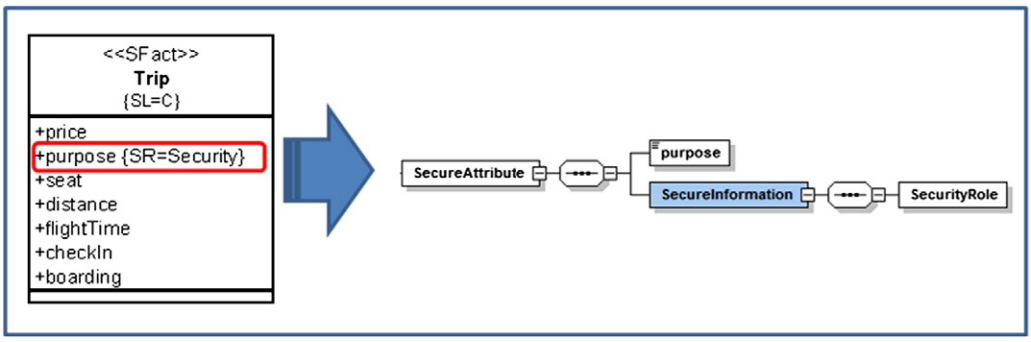

Fig. 17. Mapping the Secure Class Attributes.

All the elements affected by the rule have to be considered in order to set the security privileges needed to access each fact, dimension, base and attribute. Each Secure Constraint will be transformed into a complexType XML Subelement of the corresponding Base, Dimension or Fact Element, denominated as the corresponding constraint, with the three optional attributes as optional string type SubElements (involvedObjects, ownedSPObjects, ownedSCObjects). 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 PIM level. This complexType Subelement can contain the three aforementioned Subelements and includes the following Subelements: logType and logInfos, both of which are string types. In the case selected, this will be *AR_frustatedAttempts* which is associated with the *Trip* Secure Fact Class. This complexType Subelement includes the following Subelements: *ownedSCObjects* with the string value fixed as "*Trip, Passenger, Baggage*", *logType* with the fixed value "*frustatedAttempts*", and *logInfos*, also with the string value fixed as "*objected action time response*". Fig. 18 shows the result of this transformation for the *AR_frustatedAttempts* audit rule.
- Transforming the Authorization Rule 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 complexType Subelement can contain the three aforementioned Subelements and includes the following Subelements: the *AuthorizationRuleSign* with the fixed value {+,−}, *Privilege* with a fixed value {read, write},and the *CabExp*, which contains the Authorization Rule Condition, which is a string type, that will contain the XPath expression associated with the OCL expression.
  In the case chosen, this will be *AUR_Passenger* associated with the *Passenger* Secure Dimension Class. This complexType Subelement includes the following Subelements: *ownedSPObjects* with the string value fixed at ="*Passenger.name, Passenger.address, Trip.baggageID*", the *AuthorizationRuleSign* with the fixed value "+", *Privilege* with a fixed value *"read"* (in this case) and the *CabExp*, containing the Authorization Rule Condition, which is a string type, that will contain the XPath expression associated with the OCL expression, in this case ="*UserProfile.name* <> *Passenger.name*". Fig. 19 shows the result of applying this transformation.
- Transforming the Security Rule 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 PIM level. This complexType Subelement can contain the three aforementioned Subelements and includes the following Subelements: *CABExp* that will contain a string with the expression in XPath; the *CATHEN* that will contain the Security Information if the expression (in XPath) is TRUE and the CAELSE Subelement that will contain the Security Information if the expression is FALSE.

In the case chosen, *SIAR_Trip_Purpose* will be associated with the *Trip* Secure Fact Class. This complexType Subelement includes 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"). Fig. 20 shows the result the application of the defined mapping.

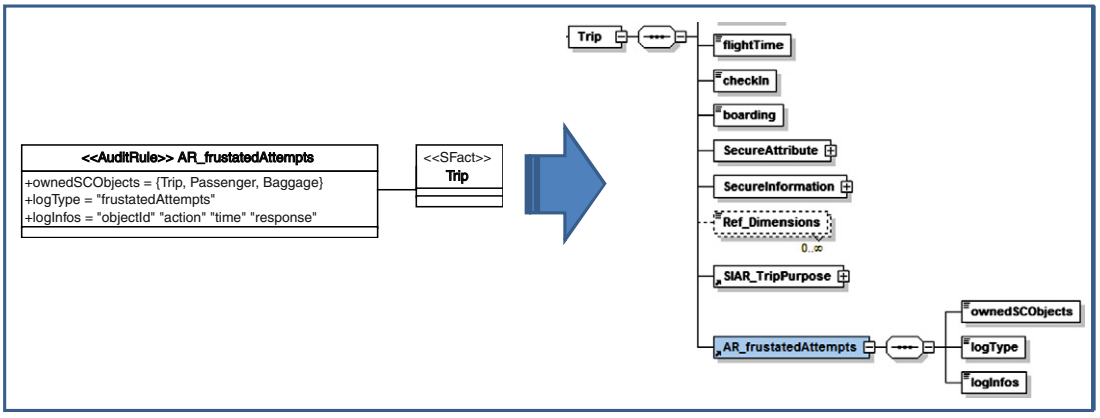

Fig. 18. Mapping the Audit Rule Constraint.

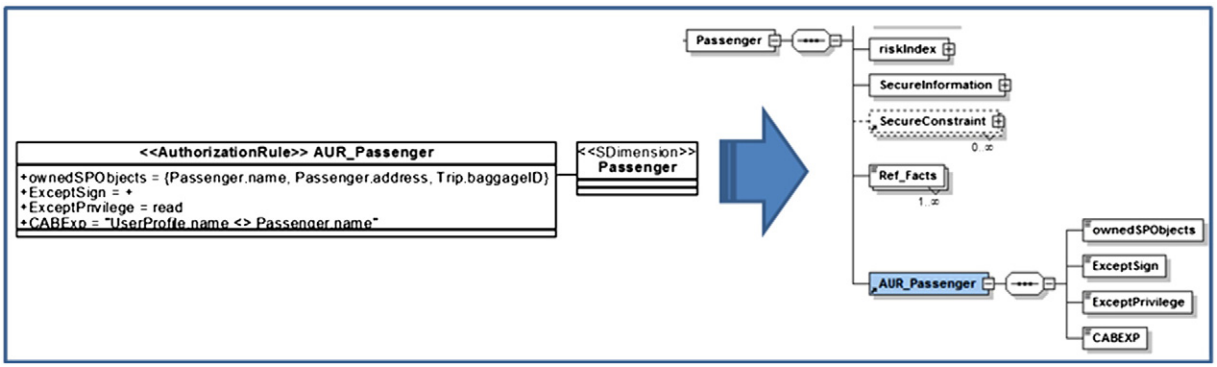

Fig. 19. Mapping the Authorization Rule Constraint.

6.14. Transformation rule dependencies

Having provided a detailed description of the transformation rules defined, along with a step by step illustration of their application to the case study selected in order to obtain the secure XML Data Warehouse, in this section we shall show the dependencies among the PIM to PSM mappings designed, along with the sequence in which they must be applied. Each mapping has been named by means of the modeling elements that it maps. The illustration shows how each mapping rule depends on others to complete the overall mapping (Fig. 21).

The starting point when a Secure XML MD PSM is derived from the Secure MD PIM is to use the ***SecureMDPIM2SecureMDXML*** transformation to create a "SecureMDXML" root Element transformation, which includes the Security Levels, Security Roles Hierarchy, Security Compartments, User Profile and Secure Star Package. Each of these XML Subelements is created through the use of the following transformation rules: ***SLevel2SLevelsXML***, ***SRoles2SRolesXML***, ***SCompartments2SCompartmentsXML***, ***UserProfile2UserProfileXML*** and ***SStarPackage2SStarPackageXML***. The first three transformation rules (*SLevel2SLevelsXML*, *SRoles2SRolesXML*, *SCompartments2SCompartmentsXML*) are used to obtain the Security Levels Hierarchy, the Security Roles Hierarchy and the Security Compartments of the Secure XML DW.

When a User Profile class is transformed using the ***UserProfile2UserProfileXML*** transformation, it is also necessary to use the ***UserProfileProperties2Elements*** transformation rule to deal with the individual properties that the UserProfile class may contain.

For the transformation of a Secure Star Package, the ***SStarPackage2SStarPackageXML*** transformation is used to obtain an XML Element called "SStarPackage" which will contain the Secure Dimensions, Facts and Bases of the Secure Star Package as XML Subelements. Each of these is transformed through the use of the following transformation rules: ***SDimension2SDimensionXML***, ***SFact2SFactXML*** and ***SBase2SBaseXML***.

When a Secure Dimension is transformed (using the *SDimension2SDimensionXML* mapping) it is also necessary to use the following transformation rules to complete the transformation of the Dimension: ***SAttribute2Element*** *and* ***DimensionBaseAssociation2IDREF*** transformations, along with the corresponding ***SecureInformation2SecureInformationXML*** transformation (detailed as follows).

In order to complete the transformation of a Secure Fact (using the *SFact2SFactXML* transformation) it is also necessary to use the following transformation rules: ***SAttribute2Element***, ***FactDimensionAssociation2IDREFS*** and the ***SecureInformation2SecureInformationXML*** transformations.

Similarly, when we transform a Secure Base (using the ***SBase2SBaseXML*** transformation) it is also necessary to use the ***SAttribute2Element, BaseBaseAssociation2IDREFS*** and ***SecureInformation2SecureInformationXML*** transformation rules.

In the three previous cases, it is also necessary to transform the existing security constraints of the Dimensions, Facts and Bases. An Audit Rule is transformed by using the ***AuditRule2AuditRuleXML*** transformation, a Security Rule is transformed by using the ***SecurityRule2SecurityRuleXML*** transformation, and an Authorization Rule is transformed by using the ***AuthorizationRule2AuthorizationRuleXML*** transformation.

When the Security Information of a Dimension, Fact or Base is transformed (using the ***SecureInformation2SecureInformationXML*** transformation), it is also necessary to use the transformation rules that allow us to obtain the XML Element corresponding to the Security Level (***SLevel2Element*** transformation), Security Roles (***SRoles2Element*** transformation) and Security Compartments (***SCompartmenst2Element*** transformation).

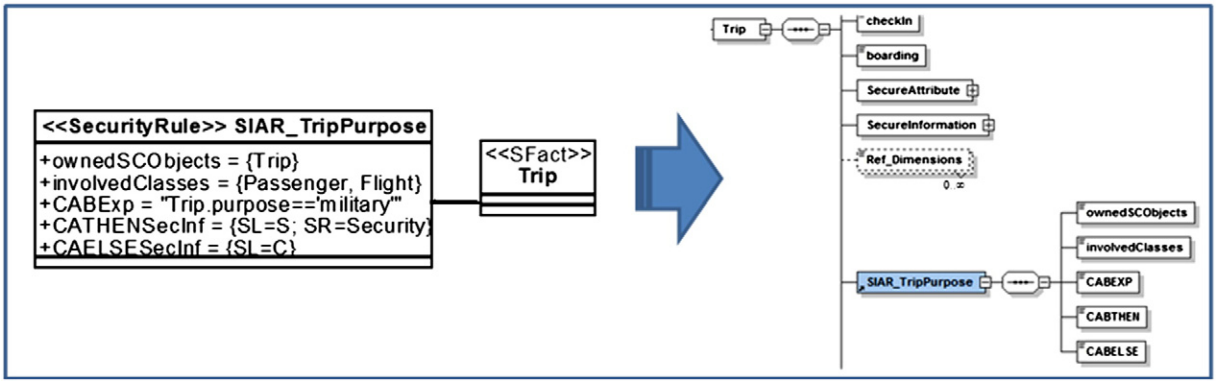

Fig. 20. Mapping the Security Rule Constraint.

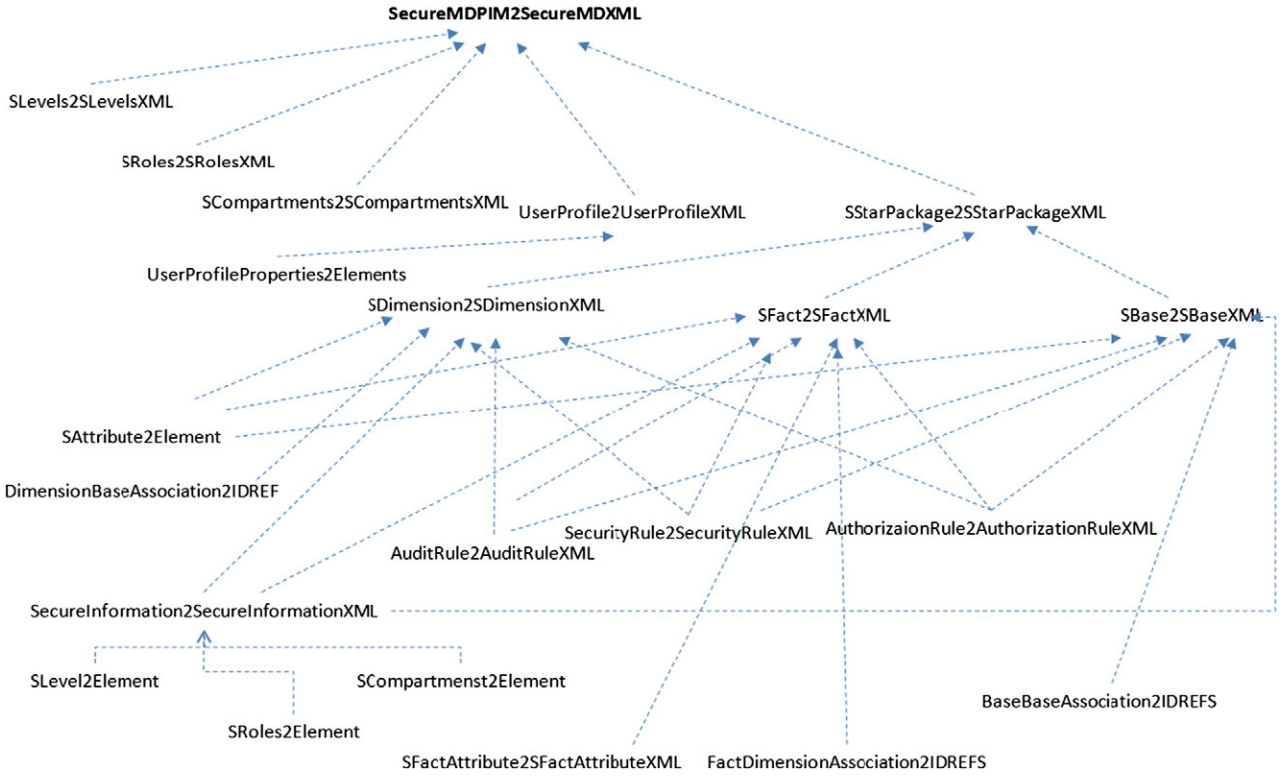

Fig. 21. Dependency Graph of the defined mappings.

7. Secure XML DW'S implementation

In the previous section we have transformed our conceptual models for secure DWs (PIM) into Secure XML DW models (PSM) which contain all the elements necessary for the final implementation of the Secure XML DW in any secure commercial database management system. Thus, this section describes how to implement our PSM models into commercial tools and provides further detail of the eventually implementation of the case study presented in this paper into Oracle XML DB 11 g.

First, the XML schema generated must be registered, creating a structure for our XML schema which allows us to insert XML files. Table 1 shows the syntax in Oracle XML DB 11 g to register the schema "airportDW.xsd" storing the information concerning to its definition into a directory called "AIRPORTDW". Next, XML data can be inserted by using an "insert into" command. Table 1 shows an example in which a file "Trip.xml" is inserted in our airport DW by using the table defined as default (XML_DEFAULT).

Once the structural aspects of our Airport DW have been implemented, security constraints must be established. The majority of commercial XML database management systems provide security mechanisms focused on a role based security policy (RBAC) and action control lists (ACL). An ACL is a list of access control entries (ACE) that determine which subjects have access to a given resource, thus each ACE is composed of: (i) an entity or principal, which can be database roles, groups, users, etc.; (ii) a privilege, which is a particular right that can be granted or denied. Since DW' users mainly deal with read operations, our proposal focuses on managing read privileges to avoid confidentiality problems.; and (iii) a value that indicates if the privilege is granted or revoked. Finally, the ACLs defined must be applied to specific resources.

Now, we describe the strategy that we have followed for implementing the security constraints over the DW. Although we describe the secure implementation by using the case study previously presented and Oracle XML DB 11 g, these steps can be applied to achieve a secure implementation in other XML database management systems.

7.1. Security configuration

Our Secure XML DB model (PSM) allows us to specify security compartments (SC), roles (SR) and levels (SL), whereas the majority of XML database management systems use a role based security policy. Thus, the information concerning with the security configuration must be adapted from our PSM model to an RBAC policy. To achieve this goal, each security compartment, role and level is implemented as a role with an identification name composed of a prefix ("SC", "SR" or "SL" depending of its source element) concatenated with the original name of the element in the PSM model. Then, each user must be added as a member of the roles

Table 1
Schema registration.

```
DBMS_XMLSCHEMA.REGISTERSCHEMA(
  schemaurl => 'airportDW.xsd',
  schemadoc => BFILENAME ('AIRPORTDW','airportDW.xsd') );
insert into XML_DEFAULT values (XMLTYPE(BFILENAME
  ('AIRPORTDW','Trip.xml'),nls_charset_id('AL32UTF8')));
```

Table 2
Security configuration.

PSM model	Implementation
Security Compartment "CompanyA"	Role "SCCompanyA"
Security Role "User"	Role "SRUser"
Security Level "TopSecret"	Role "SLTopSecret"

that represent their privileges (security level, roles and compartments). For instance, in our case study the following elements: "CompanyA" security compartment, "User" security role and "TopSecret" security level; are implemented as the roles "SCCompanyA", "SRUser" and "SLTopSecret" (see Table 2).

7.2. Definition of ACLs

Once structural aspects and security roles have been specified, the security constraints defined must be implemented as ACLs which will be applied to the corresponding objects. We use an open policy in which the access to all the resources of the DW is allowed, and then, we explicitly apply ACLs to deny certain resources to certain roles. In order to achieve this goal, we create an ACL for each role, in which the read privilege is denied. Finally, these ACLs will be applied depending on the security constraints defined.

Table 3 shows the ACL for the role "SLUndefined" which correspond with the security level "Undefined" in the PSM model. This ACL defines an access control entry (ACE) to deny the read privilege to the role "SLUndefined" that is represented as "principal".

7.3. Applying ACLs

Table 4 shows how to apply the ACLs previously defined in order to implement the security constraints defined in the PSM model. Firstly, an ACL provided by Oracle XML DB ("ro_all_acl.xml") is applied to grant read privileges to all roles. Then, based on the security privileges needed to access each resource of the DW (facts, dimensions, bases and attributes) ACLs are applied to deny unauthorized accesses to each element. This information is defined in the PSM model associated with the resource by using a security information element composed of the level, roles and compartments which are needed to access the resource.

Since the security information attached to "Trip" specifies that it can be accessed by users with a security level of confidential (or upper security level), the ACL which denies read accesses to the role "SLUndefined" has to be applied to "Trip". The remainder security information sets associated with facts, dimensions, bases and attributes are processed in the same way. In this case study, it is necessary to apply ACLs to: "purpose" attribute of "Trip", denying roles different to "Security"; "Baggage" dimension, denying read privileges to security levels lower than "Confidential" and roles different to "Security" and "Baggaging"; "Passenger" dimension, denying security levels lower than "Secret"; and for each attribute of "Passenger" which requires a security role of "Security" (fingerprint, passportPhoto, etc.) denying read accesses to roles different to "Security".

Nevertheless, our proposal also allows us to define security rules which assign different security privileges depending on the evaluation of a condition specified. Fig. 7 shows the security rules defined for this case study. The Security Information Assignment Rules (SIAR) defined, establish a condition and if it is satisfied, the security privileges necessary to read the involved object are increased. This kind of rules has been implemented in Oracle XML DB 11 g by using procedures which evaluate the condition established in the SIAR and if it is satisfied, additional ACLs are applied in order to avoid unauthorized accessed (Table 5). For instance, the SIAR "SIAR_PassengerSuspicious" increases the security privileges needed to access suspicious passengers from a security level of "Secret" to a security level of "Top Secret" and a security role of "Security". This rule considers a passenger as suspicious if satisfies the following condition "passenger.suspicious==true and passenger.riskIndex>5". The procedure created to implement this SIAR evaluates the condition and if it is satisfied applies a set of ACLs which increase the privileges needed to access "Passenger" dimension to a security level of "Top Secret" and a security role of "Security", by denying read accesses to the roles that represent the security level "Secret" and the roles "Administration", "Flight" and "Passenger". The remainder SIAR rules are processed in a similar way.

In our case study there are also several Authorization Rules (AUR) defined. One of them ("AUR_Passenger") is a positive rule which allows users to access their own passenger information.

Table 3
ACL for "SLUndefined".

```
<acl description="aclSLUndefined"
    xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
    xmlns:dav="DAV:"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
      http://xmlns.oracle.com/xdb/acl.xsd">
  <ace>
    <principal>SLUndefined</principal>
    <deny>true</deny>
    <privilege><dav:read/></privilege>
  </ace>
</acl>
```

Table 4
Applying ACLs.

```
CALL DBMS_XDB.setAcl('/AIRPORTDW', 'ro_all_acl.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Trip', 'aclSLUndefined.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Trip/purpose',
  'aclSRAdministration.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Trip/purpose',
  'aclSRFlight.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Trip/purpose',
  'aclSRPassenger.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Baggage', 'aclSLUndefined.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Baggage', 'aclSRBoarding.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Baggage', 'aclSRFlight.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Baggage',
  'aclSRPassenger.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger',
  'aclSLConfidential.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger',
  'aclSLUndefined.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/fingerprint',
  'aclSRAdministration.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/fingerprint',
  'aclSRFlight.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/fingerprint',
  'aclSRPassenger.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/passportPhoto',
  'aclSRAdministration.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/passportPhoto',
  'aclSRFlight.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/passportPhoto',
  'aclSRPassenger.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/criminalRecord',
  'aclSRAdministration.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/criminalRecord',
  'aclSRFlight.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/criminalRecord',
  'aclSRPassenger.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/suspicious',
  'aclSRAdministration.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/suspicious',
  'aclSRFlight.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/suspicious',
  'aclSRPassenger.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/riskIndex',
  'aclSRAdministration.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/riskIndex',
  'aclSRFlight.xml');
CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger/riskIndex',
  'aclSRPassenger.xml');
```

Table 5
Implementation of sensitive information assignment rule.

```
CREATE FUNCTION SIAR_PassengerSuspicious (suspicious: Boolean,
riskIndex: Integer)
BEGIN
  IF suspicious=true and riskIndex>5
    CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger',
  'aclSLSecret.xml');
    CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger',
  'aclSRAdministration.xml');
    CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger',
  'aclSRFlight.xml');
    CALL DBMS_XDB.setAcl('/AIRPORTDW/Passenger',
  'aclSRPassenger.xml');
  ENDIF;
END;
CREATE FUNCTION SIAR_BaggageSuspicious (suspicious: Boolean)
BEGIN
  IF suspicious=true
    CALL DBMS_XDB.setAcl('/AIRPORTDW/Baggage',
  'aclSLConfidential.xml');
    CALL DBMS_XDB.setAcl('/AIRPORTDW/Baggage',
  'aclSRBaggage.xml');
  ENDIF;
END;
CREATE FUNCTION SIAR_TripPurpose (purpose: Varchar2(20))
BEGIN
  IF purpose='military'
    CALL DBMS_XDB.setAcl('/AIRPORTDW/Trip/purpose',
  'aclSLConfidential.xml');
    CALL DBMS_XDB.setAcl('/AIRPORTDW/Trip/purpose',
  'aclSRAdministrator.xml');
    CALL DBMS_XDB.setAcl('/AIRPORTDW/Trip/purpose',
  'aclSRFlight.xml');
    CALL DBMS_XDB.setAcl('/AIRPORTDW/Trip/purpose',
  'aclSRPassenger.xml');
  ENDIF;
END;
```

This rule is implemented (Table 6) by using a procedure which applies a positive ACL to grant read privileges to each user to his/her own passenger information, represented as an XPath expression. This procedure receives a parameter with an ACL which grants read privileges to the user. The other AUR ("AUR_Company") is a negative rule which denies accesses to "Flight" information relative to a company different to the user's company. This rule has been also implemented (Table 6) with a procedure which received as parameters the company of the user and the ACL to apply.

8. Conclusions and future works

In this work we have shown the applicability of our approach for the model driven development of Secure XML DWs. This approach contributes in the area of two of the most interesting challenges identified in the field of DW development: i) the definition of methodological approaches for the design of XML DW, which is integrated into our MDA approach and covers a new PSM model, and ii) the inclusion of security in DWs, a highly sensitive key requirement of this type of systems, thus focusing our proposal on XML DW security. Our approach commences with the definition of 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 used as a starting point and is then semi-automatically transformed into a secure XML DW, as a logical model (PSM), by applying Model to Model (M2M) Transformations. In this paper, we have specified those transformation rules with which 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 both aspects to be implemented simultaneously.

Our proposal has been validated with several case studies. In this paper, we have presented the practical application of our model driven development approach for the Modeling of Secure XML Data Warehouses to one of the case studies developed, which is based on a central Airport DW and includes information concerning trips, flights and incidents. We have provided a step by step illustration of how the proposed mappings have been applied to the secure conceptual model to obtain the secure XML data warehouse, thus showing the benefits of our development approach.

Some of the most relevant lessons learned during the application of our approach in the case studies developed, particularly in the case study detailed in this paper, have made it possible to significantly improve and refine the transformation rules defined to obtain the secure XML Schema from the secure conceptual multidimensional model. Starting from these refined transformation rules specified with natural language, the next step will be to formalize the mappings using the QVT standard language and to implement them in a tool for the automatic generation of the Secure XML Data Warehouse working code. The formalization of the transformation rules will assist us in the more systematic detection of errors. It will also help us to detect inconsistencies in the early stages of software development, which may help to increase the quality of the models built, along with the subsequent working code automatically generated from them.

We are currently working on several different lines in an attempt to extend the proposal presented in this paper. One of these, on which we have already started work, is the automation of the transformations of the constraints expressed in OCL at the PIM level, in order to convert them into XPath language. We are also working on the formalization and automation of the transformations between the metamodels and their corresponding models using the QVT proposal. A further goal is that of performing several additional case studies in order to detect new needs. These would also analyze the advantages of incorporating the security aspects provided by the different XML Database administrators, and not only those which are native. The next step will be to include our proposal in the case tool that we are developing for the semi-automatic development of Secure XML DW.

Table 6
Implementation of authorization rules.

```
CREATE FUNCTION AUR_Passenger (userName: Varchar2(20), userAcl:
Varchar(50))
BEGIN
  Expr='/AIRPORTDW/Passenger[@name=' +username+']';
  CALL DBMS_XDB.setAcl(Expr, userAcl);
END;
CREATE FUNCTION AUR_Company (userSC: Varchar2(20), userAcl:
  Varchar(50))
BEGIN
  Expr='/AIRPORTDW/Flight[./Plane/AircraftType/Company/
  companyCode<>' +userSC+']';
  CALL DBMS_XDB.setAcl(Expr, userAcl);
END;
```

Acknowledgments

This research has been carried out in the framework of the following projects: MODEL-CAOS (TIN2008-03582/TIN) financed by the Spanish Ministry of Education and Science, BUSINESS (PET2008-0136) financed by the Ministry of Science and Innovation, and SISTEMAS (PII2I09-0150-3135) and SERENIDAD (PEII11-0327-7035), all financed by the Local Government of Castilla-La Mancha, in Spain.

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

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

```
<?xml version="1.0" encoding="utf-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by BVS (BVS) -->
<!-- edited with XMLSpy v2010 (http://www.altova.com) by Belén Vela Sánchez (Rey Juan Carlos University) -->
<!--Created with Liquid XML Studio - FREE Community Edition 7.1.6.1440 (http://www.liquid-technologies.com)-->
<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>
            </xs:element>
            <xs:element name="Confidential" fixed="Confidential">
                <xs:complexType>
```

Fig. 22. XML Schema Code for the Airport example.

```
                    <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"/>
                <!-- involved Classes-->
                <!--ownedSCObjects-->
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <!– Audit Rule Transformation: AR_frustatedAttempts-->
    <xs:element name="AR_frustatedAttempts">
        <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="frustatedAttempts "/>
                <xs:element name="logInfos" type="xs:string" fixed="'objectId','action','time','response' "/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <!– 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="+"/>
```

Fig. 22 (*continued*).

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

Fig. 22 (*continued*).

```
                    </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_frustatedAttempts"/>
            </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 all the 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>
        </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"/>
```

Fig. 22 (*continued*).

```
                    <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>
                            </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>
```

Fig. 22 (*continued*).

```
                        <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>
            <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>
```

Fig. 22 (*continued*).

```
                <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"/>
                </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"/>
```

Fig. 22 (*continued*).

```
                    <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>
<!--Transformation of  complexType: StarPackage_Type, which includes all the Secure Facts, Dimensions and
Bases of a Starpackage-->
<xs:complexType name="StarPackage_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>
<!--Transformation of  complexType: SecureMDXML_Type, which includes the Security Roles Hierarchy, the
Levels, Compartments, StarPackage and UserProfile of the Secure XML DW-->
<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="StarPackage" type="StarPackage_Type"/>
        <xs:element name="UserProfile" type="UserProfile_Type"/>
    </xs:sequence>
 </xs:complexType
</xs:schema>
```

Fig. 22 (*continued*).

References

[1] A. Abelló, J. Samos, F. Saltor, A Framework for the Classification and Description of Multidimensional Data Models, 12th International Conference on Database and Expert Systems Applications (DEXA'01). LNCS 2113, 2001, pp. 668–677.
[2] A. Abelló, J. Samos, F. Saltor, YAM2: a multidimensional conceptual model extending UML, Information Systems 31 (6) (2006) 668–677.
[3] X. Baril, Z. Bellahsene, Designing and Managing an XML Warehouse, XML Data Management: Native XML and XML- Enabled Databse Systems, Addison Wesley, 2004, pp. 455–474.
[4] D. Basin, J. Doser, T. Lodderstedt, Model driven security: from UML models to access control infrastructures, ACM Transactions on Software Engineering and Methodology 15 (1) (2006) 39–91.
[5] B. Best, J. Jürjens, B. Nuseibeh, Model-Based Security Engineering of Distributed Information Systems Using UMLsec, International Conference on Software Engineering, IEEE Computer Society, Minneapolis, MN, USA, 2007.
[6] K.S. Beyer, et al., Extending XQuery for Analytics, ACM SIGMOD Infernational Conference on Management of Data, ACM, Baltimore, Maryland, 2005, pp. 503–514.
[7] N.T. Binh, A.M. Tjoa, R. Wagner, An object oriented multidimensional data model for OLAP, Web-Age Information Management, 2000, pp. 69–82.
[8] C. Blanco, et al., Applying QVT in order to implement Secure Data Warehouses in SQL Server Analysis Services, Journal of Research and Practice in Information Technology 41 (2) (2009) 135–154.
[9] C. Blanco, et al., Towards a modernization process for Secure Data Warehouses, in: T.B. Pedersen, M.K. Mohania, A.M. Tjoa (Eds.), 11th International Conference on Data Warehousing and Knowledge Discovery (DaWaK), Springer-Verlag Berlin Heidelberg, Linz, Austria, 2009, pp. 24–35.
[10] A. Bonifati, et al., Designing data marts for data warehouses, ACM Transactions on Software Engineering and Methodology 10 (4) (2001) 452–483.
[11] R. Bordawekar, C.A. Lang, Analytical Processing of XML documents: opportunities and callenges, SIGMOD Record 34 (2) (2005) 27–32.
[12] O. Boussaid, et al., X-Warehousing: An XML-Based Approach for Warehousing Complex Data, East European Conf. on Advances in Databases and Information Systems (ADBIS), Springer, 2006, pp. 39–54.
[13] P. Bresciani, et al., Tropos: Agent-Oriented Software Development Methodology, Journal of Autonomous Agents and Multi-Agent System 8 (2004) 203–236.
[14] L. Cabibbo, R. Torlone, A logical approach to multidimensional databases, International Conference on Extending Database Technology: Advances in Database Technology, 1998, pp. 183–197.
[15] L. Carneiro, A. Brayner, X-meta: A methodology for data warehouse design with metadata management, International Workshop on Design and Management of Data Warehouses (DMDW), 2002, pp. 13–22, CEUR-WS.org.
[16] J.M. Cavero, M. Piattini, E. Marcos, Midea: A multidimensional data warehouse methodology, International Conference on Enterprise Information Systems (ICEIS), 2001, pp. 138–144.
[17] N.T. Debevoise, The Data Warehouse Method, Prentice-Hall, New Jersey, USA, 1999.
[18] I. Dubielewicz, et al., Evaluation of MDA-PSM database model quality in the context of selected non-functional requirements, International Conference on Dependability of Computer Systems, IEEE Computer Society, Szklarska Poreba, Poland, 2007.
[19] A. Ekelhart, et al., XML Security – A Comparative Literature Review, Journal of Systems and Software 10 (10) (2008) 1715–1724.
[20] E. Fernández-Medina, et al., Access control and audit model for the multidimensional modeling of data warehouses, Decision Support Systems 42 (2006) 1270–1289.
[21] E. Fernández-Medina, et al., Developing Secure Data Warehouses with a UML extension, Information Systems 32 (6) (2007) 826–856.
[22] P. Giorgini, H. Mouratidis, N. Zannone, Modelling Security and Trust with Secure Tropos, Integrating Security and Software Engineering: Advances and Future Visions, Idea Group Publishing, 2006.
[23] P. Giorgini, S. Rizzi, M. Garzetti, Grand: a goal-oriented approach to requirements analysis in data warehouses, Decission Support Systems 45 (1) (2008) 4–21.
[24] M. Golfarelli, S. Rizzi, UML-Based Modeling for What-If Analysis, DaWaK, 2008, pp. 1–12.
[25] M. Golfarelli, D. Maio, S. Rizzi, The Dimensional Fact Model: A Conceptual Model for Data Warehouses, International Journal of Cooperative Information Systems (IJCIS) 7 (2–3) (1998) 215–247.
[26] M. Hachicha, H. Mahboubi, J. Darmont, Expressing OLAP operators with the TAX XML algebra, EDTB workshop on Database Technologies for Handling XML Information on the Web (DataX-EDBT), ACM, Nantes, France, 2008, pp. 61–66.
[27] Harmon, P., *The OMG's Model Driven Architecture and BPM.* Newsletter of Business Process Trends, 2004.
[28] Inmon, W., *2.0 - architecture for the next generation of data warehousing.* DMReview. DM Direct Newsletter, 2006.
[29] J. Jürjens, Secure Systems Development with UML, Springer-Verlag, 2004.
[30] J. Jürjens, P. Shabalin, Tools for Secure Systems Development with UML, Invited submission to the FASE 2004/05 special issue of the International Journal on Software Tools for Technology Transfer, 9(5–6), 2007, pp. 527–544.
[31] N. Katic, et al., A Prototype Model for Data Warehouse Security Based on Metadata, 9th International Workshop on Database and Expert Systems Applications (DEXA), IEEE Computer Society, Vienna, Austria, 1998.
[32] R. Kimball, The Data Warehouse Toolkit, John Wiley & Sons, 2002.
[33] R. Kirkgöze, et al., A Security Concept for OLAP, 8th International Workshop on Database and Expert System Applications (DEXA'97), IEEE Computer Society, Toulouse, France, 1997.
[34] A. Kleppe, J. Warmer, W. Bast, MDA Explained; The Model Driven Architecture: Practice and Promise, Addison-Wesley, 2003.
[35] Y. Li, A. An, Representing UML Snowflake Diagram from Integrating XML Data Using XML Schema, Int. Workshop on Data Engineering Issues in E-Commerce (DEEC), IEEE Computer Society, 2005, pp. 103–111.
[36] B. Li, S. Liu, Z. Yu, Applying MDA in traditional Database-based Application Development, International Conference on Computer Supported Cooperative Work in Design, LNCS 3865, Springer, Coventry, UK, 2006.
[37] T. Lodderstedt, D. Basin, J. Doser, SecureUML: A UML-based modeling language for model-driven security, UML 2002. The Unified Modeling Language. Model Engineering, Languages Concepts, and Tools. 5th International Conference, Springer, Dresden, Germany, 2002.
[38] S. Luján-Mora, J. Trujillo, I.-Y. Song, A UML profile for multidimensional modeling in data warehouses, Data & Knowledge Engineering 59 (3) (2006) 725–769.
[39] J.-N. Mazón, J. Trujillo, An MDA approach for the development of data warehouses, Decision Support Systems 45 (1) (2008) 41–58.
[40] S. Mellor, et al., MDA Distilled: Principles of Model-Driven Architecture, Addison Wesley, 2004.
[41] T. Mens, P. Van Gorp, A taxonomy of model transformation, Electronic Notes in Theoretical Computer Science 152 (2006) 125–142.
[42] T. Meservy, K. Fenstermacher, Transforming Software Development: An MDA Road Map, Computer 38 (9) (2005) 52–58.
[43] H. Mouratidis, Software Engineering for Secure Systems: Industrial and Research Perspectives, IGI Global, 2011.
[44] V. Nassis, et al., An XML Document Warehouse Model, Int. Conf. on Database Systems for Advanced Applications (DASFAA), Springer, 2006, pp. 513–529.
[45] F.R.S. Paim, J. Castro, DWARF: An approach for requirements definition and management of data warehouse systems, IEEE International Conference on Requirements Engineering, 2003, pp. 75–84.
[46] B.-K. Park, H. Han, I.-Y. Song, XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses, Data Warehousing and Knowledge Discovery, 2005, pp. 32–42, LNCS 3589.
[47] D. Pedersen, J. Pedersen, T.B. Pedersen, Integrating XML Data in the TARGITOLAP System, Int. Conference on Data Engineering (ICDE), IEEE Computer Society, 2004, pp. 778–781.
[48] J.M. Pérez, et al., A relevance-extended multi-dimensional model for a data warehouse contextualized with documents, DOLAP, 2005, pp. 19–28.
[49] J.M. Pérez, et al., Integrating Data Warehouses with Web Data: A Survey, IEEE Transaction Knowledge Data Engineering 20 (7) (2008) 940–955.
[50] J. Poole, Model-driven data warehousing, www.cwmforum.org/POOLEIntegrate2003.pdfIntegrate 2003 conference, 2003, Burlingame, CA.
[51] N. Prat, J. Akoka, I. Comyn-Wattiau, A UML-based data warehouse design method, Decision Support Systems 42 (3) (2006) 1449–1473.
[52] T. Priebe, G. Pernul, A Pragmatic Approach to Conceptual Modeling of OLAP Security, 20th International Conference on Conceptual Modeling (ER 2001), Springer-Verlag, Yokohama, Japan, 2001.
[53] F. Ravat, et al., Finding an application-appropriate model for XML data warehouses, Information Systems 35 (6) (2010) 662–687.
[54] S. Rizzi, et al., Research in data warehouse modeling and design: dead or alive? DOLAP, 2006, pp. 3–10.
[55] A. Rosenthal, E. Sciore, View Security as the Basic for Data Warehouse Security, 2nd International Workshop on Design and Management of Data Warehouse (DMDW'00), CEUR Workshop Proceedings (CEUR-WS.org), Sweden, 2000.
[56] L. Rusu, W. Rahayu, D. Taniar, A Methodology for Building XML Data Warehouses, IJDWM 1 (2) (2005) 23–48.
[57] F. Saltor, et al., 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.
[58] C. Sapia, et al., Extending the E/R Model for the Multidimensional Paradigm, 1st International Workshop on Data Warehouse and Data Mining (DWDM'98), Springer-Verlag, Singapore, 1998.
[59] B. Selic, The pragmatics of Model-Driven Software Development, IEEE Software 20 (5) (2003) 19–25.
[60] E. Soler, et al., Building a secure star schema in data warehouses by an extension of the relational package from CWM, Computer Standard and Interfaces 30 (6) (2008) 341–350.
[61] T. Stahl, M. Volter, K. Czarnecki, Model-Driven Software Development: Technology, Engineering, Management, John Wiley and Sons, 2006.
[62] C. Steel, R. Nagappan, R. Lai, The Alchemy of Security Design Methodology, Patterns, and Reality Checks, Core Security Patterns: Best Practices and Strategies for J2EE, Web Services, and Identity Management, Prentice Hall, 2005, p. 1088.
[63] N. Szirbik, C. Pelletier, T. Chaussalet, Six methodological steps to build medical data warehouses for research, International Journal of Medical Informatics 75 (9) (2006) 683–691.
[64] 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 (4) (2007) 367–382.
[65] J. Trujillo, et al., An Engineering Process for Developing Secure Data Warehouses, Information and Software Technology 51 (6) (2009) 1033–1051.
[66] J. Trujillo, et al., A UML 2.0 Profile to define Security Requirements for DataWarehouses, Computer Standard and Interfaces 31 (5) (2009) 969–983.
[67] N. Tryfona, F. Busborg, J. Christiansen, starER: A Conceptual Model for Data Warehouse Design, ACM 2nd International Workshop on Data Warehousing and OLAP (DOLAP'99), ACM, Missouri, USA, 1999.

[68] 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 (2) (2008) 293–329.
[69] B. Vela, et al., Model Driven Development of Secure XML Databases, ACM Sigmod Record 35 (3) (2006) 22–27.
[70] H. Wang, et al., OLAP for XML Data, Int. Conf. on Computer and Information Technology (CIT), IEEE Computer Society, 2005, pp. 233–237.
[71] E. Weippl, et al., An Authorization Model for Data Warehouses and OLAP, Workshop on Security in Distributed Data Warehousing, IEEE Computer Society, New Orleans, Louisiana, USA, 2001.
[72] N. Wiwatwattana, et al., X^3: A Cube Operator for XML OLAP, International Conference on Data Engineering (ICDE), IEEE Computer Society, Istanbul, Turkey, 2007, pp. 916–925.
[73] X. Yin, T.B. Pedersen, Evaluating XML-extended OLAP queries based on a physical algebra, ACM Int. Workshop on Data Warehousing and OLAP (DOLAP), ACM Press, 2004, pp. 73–82.
[74] E. Yu, L. Liu, A. Mylopoulos, Social Ontology for Integrating Security and Software Engineering, Integrating Security and Software Engineering: Advances and Future Visions, Idea Group Publishing, 2006.

Belén Vela Sánchez is Assistant Professor at the Rey Juan Carlos University (Madrid, Spain). She received her MSc in Computer Science from the Carlos III University and her PhD in Computer Science from the Rey Juan Carlos University. She is a member of the Department of Computing Languages and Systems II, and her research topics focus mainly on Database Design, Conceptual Modeling, Information Systems Engineering, especially on Model-Driven methodologies, Web Information Systems, etc. She has co-authored one book and has published several articles in international journals and conferences on these topics. She has participated and managed several research projects.

Carlos Blanco has an MSc in Computer Science from the University of Castilla-La Mancha. He is currently a PhD student and a member of the GSyA Research Group at the School of Computer Science at the University of Castilla-La Mancha (Spain). His research activity is in the field of Security in Information Systems focused on Data Warehouses, OLAP tools, MDA and Ontologies. He is author of several papers on these topics.

Eduardo Fernández-Medina holds a PhD in Computer Science from the University of Castilla-La Mancha. He leads the GSyA Research Group of the Department of Computer Science at the University of Castilla-La Mancha. His research activity is in the field of security in databases, data warehouses, web services and information systems, and also in security metrics. Fernández-Medina is co-editor of several books and book chapters on these subjects and has presented several dozens of papers at national and international conferences (DEXA, CAISE, UML, ER, etc.). He is the author of several manuscripts in national and international journals (DSS, ACM Sigmod Record, IS, IST, C&S, ISS, etc.) and belongs to various professional and research associations (AEC, ISO, IFIP WG11.3, etc.).

Esperanza Marcos is Full Professor at the Department of Computing Languages and Systems II, Rey Juan Carlos University of Madrid. She received the Ph.D. degree in computer sciences from the Thecnical University of Madrid, Madrid, Spain, in 1997. She chairs the Kybele research group which focused on Information System Engineering. Her current research interests include service engineering and model-driven engineering. She has co-authored several books and has published several book chapters and articles in international journals and conferences. She has participated and managed several research projects.