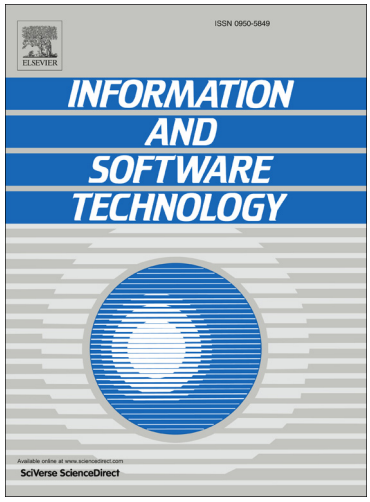


An architecture for automatically developing Secure OLAP applications from models

Carlos Blanco, Ignacio García-Rodríguez de Guzmán, Eduardo Fernández-Medina, Juan Trujillo

PII: S0950-5849(14)00219-5
DOI: http://dx.doi.org/10.1016/j.infsof.2014.10.008
Reference: INFSOF 5537

To appear in: *Information and Software Technology*

Received Date: 3 February 2014
Revised Date: 5 September 2014
Accepted Date: 26 October 2014

Please cite this article as: C. Blanco, I.G. de Guzmán, E. Fernández-Medina, J. Trujillo, An architecture for automatically developing Secure OLAP applications from models, *Information and Software Technology* (2014), doi: http://dx.doi.org/10.1016/j.infsof.2014.10.008

An architecture for automatically developing Secure OLAP applications from models

Carlos Blanco[a,*], Ignacio García-Rodríguez de Guzmán[b], Eduardo Fernández-Medina[c], Juan Trujillo[d]

[a] *GSyA Research Group. Dep. of Mathematics, Statistics and Computer Science. Faculty of Sciences. University of Cantabria. Av. De los Castros s/n. 39071. Santander. Spain.*
[b] *Alarcos Research Group. Institute of Information Technologies and Systems. Dep. of Information Technologies and Systems. Escuela Superior de Informática. University of Castilla-La Mancha. Paseo de la Universidad, 4. 13071. Ciudad Real. Spain.*
[c] *GSyA Research Group. Institute of Information Technologies and Systems. Dep. of Information Technologies and Systems. Escuela Superior de Informática. University of Castilla-La Mancha. Paseo de la Universidad, 4. 13071. Ciudad Real. Spain.*
[d] *Lucentia Research Group. Department of Information Languages and Systems. Facultad de Informática. University of Alicante. San Vicente s/n. 03690. Alicante. Spain.*

Abstract

Decision makers query enterprise information stored in Data Warehouses (DW) by using tools (such as On-Line Analytical Processing (OLAP) tools) which use specific views or cubes from the corporate DW or Data Marts, based on the multidimensional modelling. Since the information managed is critical, security constraints have to be correctly established in order to avoid unauthorized accesses. In previous work we have defined a Model-Driven based approach for developing a secure DWs repository by following a relational approach. Nevertheless, is also important to define security constraints in the metadata layer that connects the DWs repository with the OLAP tools, that is, over the same multidimensional structures that final users manage. This paper incorporates a proposal to develop secure OLAP applications into our previous approach: improves a UML profile for conceptual modelling; defines a logical metamodel for OLAP applications; and

*Corresponding author
Email addresses: Carlos.Blanco@unican.es (Carlos Blanco), Ignacio.GRodriguez@uclm.es (Ignacio García-Rodríguez de Guzmán), Eduardo.Fdezmedina@uclm.es (Eduardo Fernández-Medina), jtrujillo@dlsi.ua.es (Juan Trujillo)

defines and implements transformations from conceptual to logical models, and from logical models to the secure implementation into a specific OLAP tool (SQL Server Analysis Services).

Keywords: Security, Confidentiality, Model driven development, Data warehouse, OLAP, SQL Server Analysis Services

1. Introduction

A DW is a repository which manages a great amount of enterprise historical information integrated from different data sources [1]. This information is usually organized following a multidimensional approach by using facts (for instance, a product sale) and related dimensions with classifications by subjects (for instance, departments, cities or product categories). The information managed by DWs is very sensitive because involve critical business information which is used to support strategic decision making processes and furthermore, it usually involves personal data which are protected by legal regulations in most countries. Therefore, establishing the necessary security rules in design time and enforcing them when DWs are being queried by users is crucial for protecting the information stored in DWs [2, 3, 4, 5].

DWs development is composed of several stages. Firstly, data from heterogeneous data sources is integrated into the DW in an acquisition stage. The ETL (Extraction / Transformation / Loading) processes are responsible of extracting, transforming and loading these heterogeneous data. DWs repository stores data and is developed by following all the databases development stages: business, conceptual, logical and physical modeling. Finally, DWs are queried by tools (such as the OLAP tools SSAS, Oracle or Pentaho) which manages cubes or views of the corporative DW defined by using multidimensional models (facts, dimensions and classification hierarchies). The most critical point of the DW life cycle in order to incorporate security is the DW design, since security can be analyzed together with other requirements and integrated within the DW models. This is positive, because security is defined from an implementation independent approach and designers can make better design decisions. Regarding with the security characteristics that could be addressed in DWs' development (confidentiality, integrity, availability, etc.), confidentiality is the most important one since DWs mainly deal with read operations.

On the other hand, the Model-Driven Development (MDD) offers a great

change in the way we develop software, and particularly in the way we design data models. It is based on the definition of models at different abstraction levels (from the system modelling towards specific technologies) and transformations between models, by reducing as a consequence development times and costs. OMG proposes Model Driven Architecture [6] for the MDD development. MDA establishes three abstraction levels for models, that can be aligned with the traditional DW design process: business models (CIM) with system requirements; conceptual models (PIM) which do not include information about specific platforms and technologies; and logical models (PSM) with information about the specific technology used. Furthermore, it proposes Query / Views / Transformations (QVT) [7] as an intuitive language to implement the transformations between models and MOFScript for the transformations from models to text.

Our research efforts are thus applied to the development of secure DWs by following the model driven philosophy, defining models and transformations. In our proposal, security requirements are not improvised and incorporated once the system has been completely built. We early identify and include them in all the stages of the development process, fitting them into a most robust solution that provides us a better information assurance and a saving of time in maintenance.

In our previous works, we have developed a model driven architecture for secure DWs focused on a relational approach which leaded towards the final implementation into a DBMS [8]. However, DWs are finally queried by using tools which manage certain cubes or views of the corporative DW (multidimensional models with certain cubes, dimensions and hierarchies). That presents an intermediate metadata layer between decision makers and the DWs repository, in which is necessary the establishment of security aspects over. Therefore, we will be able to automatically check security constraints over every particular OLAP query posed on the DW.

This paper therefore compliments our MDA architecture for secure DWs with support for OLAP systems. In order to achieve this goal a new logical (PSM) metamodel for secure OLAP applications has been defined and connected to the architecture by developing the necessary transformations from conceptual (PIM) models and towards the eventually secure implementation. Furthermore, the previous UML profile for conceptual modeling (PIM) has been improved to support the representation and the automatic transformation of more complex security rules which were defined by using OCL expressions.

The rest of this paper is organized as follows: Section 2 will present related work; Section 3 will provide an overview of the model driven architecture for secure OLAP applications presented in this paper; the next three sections (Section 4, 5 and 6) will present the three stages of our proposal: conceptual modelling, transformation to logical models and transformation to OLAP implementation; Section 7 will describe the application of our proposal to a DW for a sales department; and Section 8 will finally present our conclusions and future work.

2. Related work

Security in software engineering is considered as a critical issue to be taken into account in the development of information systems. Several relevant works concerning with a complete secure development of information systems can be found, such as UMLsec [9, 10], MDS (Model Driven Security) [11] or PSSS (Process to Support Software Security) [12]. Nevertheless, although these are relevant contributions on secure information systems development they do not are specifically focused on DWs and their specific security problems.

Although research efforts in the development of secure DW and OLAP applications have been traditionally carried out, the majority of them have been focused on the final stage of development (logical modeling or implementation) [13, 14, 15, 2, 16]. Concerning with a complete secure DWs development we solely found the methodology of Priebe and Pernul [17] in which the authors analyze security requirements and their implementation into commercial tools by hiding multidimensional elements such as cubes, measures, slices and levels. They extend their proposal with a DWs representation at conceptual level with ADAPTed UML, but do not establish the connection between models in order to allow automatic transformations.

In our previous works, we have developed a model driven architecture for secure DWs focused on a relational approach which leaded towards the final implementation into a DBMS [8] (Figure 1). This approach was composed of several secure metamodels at different abstraction levels and automatic transformations from models to the final secure implementation in a specific DBMS, Oracle Label Security.

At the business level, a computational independent metamodel (CIM) supports an early definition of the security requirements. This metamodel [18] defines both functional and non functional requirements for DWs by

using a UML profile based on the i* framework [19], which is an agent oriented approach towards requirement engineering centering on the intentional characteristics of the agents.

For DWs secure modeling at conceptual level a Platform Independent Metamodel (PIM) called SECDW (Secure Data Warehouse) [20] has been defined as a UML profile specifically created for DWs complemented with an Access Control and Audit (ACA) model focused on DW confidentiality [21]. SECDW thus allows, on one hand, the specification of structural aspects of DWs such as facts, dimensions, base classes, measures or hierarchies, and on the other hand, the definition of security constraints by using the ACA model. The transformation from secure CIM models has been also dealt in [22] where the MDA methodology for secure DWs is described by using the standard Software Process Engineering Metamodel Specification (SPEM).

Following a relational approach at logical level (ROLAP) a Specific Platform Metamodel (PSM) called SECRDW (Secure Relational Data Warehouse) [23] has been defined as an extension of the relational package from Common Warehouse Metamodel [24]. SECRDW models secure relational elements such as tables, columns or keys, and express security rules defined at conceptual level. Moreover, the automatic transformation from conceptual models to relational logical models has been implemented by using QVT and also the eventual implementation into a DBMS, Oracle Label Security, has been dealt [25].

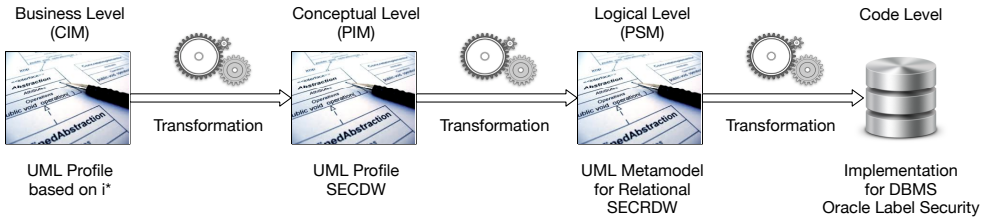


Figure 1: Model driven architecture for developing secure relational DWs

3. An Architecture for Automatically Developing Secure OLAP Applications

This section presents an overview of the contribution of this paper, a model driven architecture for developing secure OLAP applications (Figure 2).

Since most DW are queried by OLAP tools following a multidimensional approach, this paper fulfills our previous architecture by including: an improved conceptual model, called SECDW; a new logical model for secure OLAP applications, called SECMDDW (Secure Multidimensional Data Warehouse); the final implementation into an OLAP tool (SSAS); and the transformations which are necessary to automatically obtain the secure OLAP implementation from models.

As a guideline, the process to automatically obtain a secure OLAP implementation from its conceptual model is composed of three steps that will be further explained in next sections:

- **Step 1: Conceptual modelling.** This is a manual stage in which designers have to model the DW according to our UML profile. Our UML profile can be used by any UML diagramming tool that admits profiles, this facilitates the creation of the conceptual model permitting that designers include both structural and security aspects of the DWs in an easy way.

- **Step 2: Transformation to logical model for OLAP.** Once the conceptual model has been created, in this stage the transformations created in our approach are applied to automatically obtain the corresponding logical model for the OLAP technology. The resulting diagram is bigger than the conceptual model and includes all the details introduced by designers in the previous stage, but now converted to this specific technology (OLAP). This stage is completely automated and designers do not need to modify the logical model generated, although they could do it. The main utility of this logical model is to serve as an intermediate model with all the information expressed with OLAP terminology that will be useful for generating the final implementation for different OLAP tools.

- **Step 3: Transformation to OLAP implementation.** This is the final stage and also is completely automated. It runs the transformations developed in this paper to generate the implementation for a certain OLAP tool. In this paper we obtain the implementation for SQL Server Analysis Services.

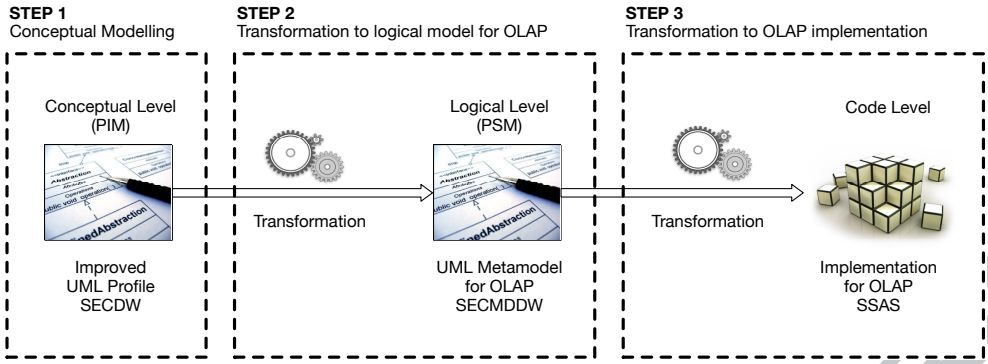


Figure 2: Model driven architecture for developing secure OLAP applications

4. Step 1: Conceptual Modelling for Secure DWs

SECDW [20] is a UML profile which has been previously defined in order to allow a secure conceptual modeling of the DW. It is based on a UML profile specifically created for conceptual modeling of DWs complemented with an Access Control and Audit (ACA) model [21] which includes security capabilities in the conceptual design by considering several security policies (Discretional Access Control, DAC; Mandatory Access Control, MAC; and Role-Based Access Control, RBAC). SECDW has been improved in this paper in order to include support for the definition and transformation of complex security rules.

Figure 3 shows SECDW. It permits the conceptual modeling of DWs structural aspects by using packages (SecurePackage metaclass); classes (SecureClass) for facts (SFact), dimensions (SDimension) and bases (SBase); properties (SecureProperty). The security configuration of the system which we want to model is defined by using three points of view: a hierarchical structure of Security Roles (SRole); a list of Security Levels (SLevel) with the clearance levels of the users; and a set of horizontal Security Compartments or groups (SCompartment). Once this configuration has been established, sets of certain security configurations composed of roles, levels and compartments can be defined as instances of secure information (SecureInformation). Then, the security configuration for the elements of our conceptual model (packages, classes, attributes, etc.) is established by associating them with specific secure information instances. Furthermore, since the user profile (UserProfile) defines all the properties that the systems manage from users, it has also a security information associated.

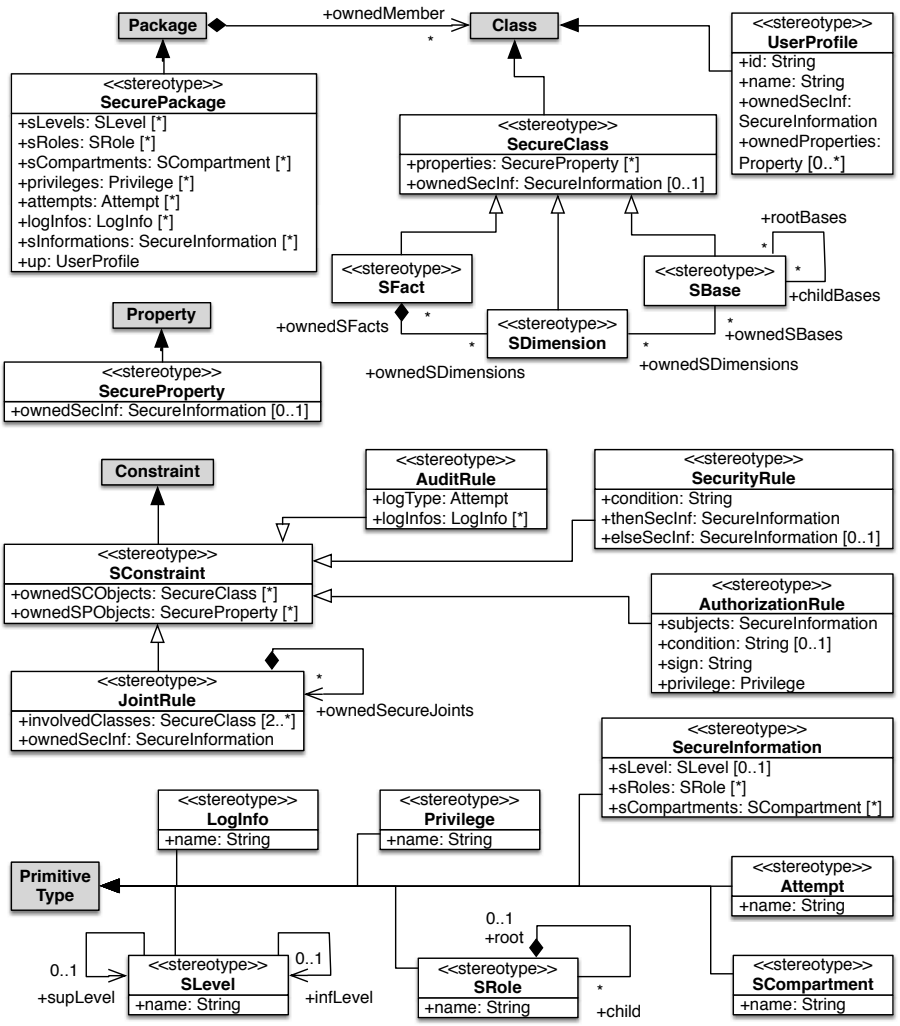


Figure 3: Profile for conceptual modelling of Secure DWs (SECDW)

ACA model permits the definition of three kinds of security rules (SConstraint) over the different elements of the DW by using Object Constraint Language (OCL) notes: the definition of sensitive information for multidimensional elements by sensitive information assignment rules (SIAR); the authorization or denegation of certain elements to specific subjects by authorization rules (AUR); and the inclusion of audit rules (AR) to ensure that authorized users do not misuse their privileges.

These rules could be complex, involving information about subjects, ob-

jects, conditions, security information, privileges, log types, etc. They were represented in SECDW by using OCL notes associated with a certain multidimensional element, but these OCL expressions are difficult to analyze and transform in an automatic way. Thus, in this paper SECDW has been improved to provide a better representation and management of complex security rules including the information necessary to support their transformation. This improvement does not affect our complete MDA architecture and our previously defined transformations.

Security rules have been included as subclasses of SConstraint. Then, SIAR security rules use a SecurityRule metaclass representing conditions with boolean expressions and the secure information that will be assigned whether the condition is satisfied or not. AUR rules use an AuthorizationRule metaclass with information about the security information associated, the sign of the authorization (positive or negative), the privilege (read, insert, update, delete, all) and a boolean expression for conditions. Finally, AR rules are specified by an AuditRule metaclass defining the access attempt and the logged information (subject, object, action, time and response).

5. Step 2: Transformation to Logical Model for OLAP

This section firstly describes the logical metamodel that has been proposed for the logical modeling of secure OLAP applications. Next, the transformations developed for automatically transform conceptual into logical models are described.

5.1. Logical Metamodel for Secure OLAP Applications

This section presents a multidimensional logical metamodel (PSM), called SECMDDW (Secure Multidimensional Data Warehouse), which allows the specification of both structural and security aspects closer to OLAP applications. Previously to the construction of our metamodel, the security capabilities provided by OLAP tools and how secure DWs could be implemented into them were studied in [26]. The SECMDDW metamodel has been developed based on the CWM metamodel [24] which is the OMG proposal for representing and interchanging metadata for data warehousing and business intelligence. SECMDDW extends the OLAP package of CWM which is focused on data analysis and defines a metamodel of essential OLAP concepts common to most OLAP applications.

Figure 4 shows the logical metamodel for secure OLAP applications developed. Three main parts can be identified: security configuration (upper part of Figure 4), cubes (middle part) and dimensions (bottom part). The security configuration of the system is represented by considering a role based access control (RBAC) policy which is used by the most OLAP tools. Since, conceptual models are more abstract than logical models, there is a semantic loss when we move towards lower levels. At the conceptual level, SECDW considers the definition of the security configuration by using roles, levels and compartments, but at the logical level our metamodel is solely focused on RBAC and security levels and compartments should be adapted as security roles. That is, the security configuration is defined as a set of security roles (Role metaclass) and the users who are members of each role (Member metaclass).

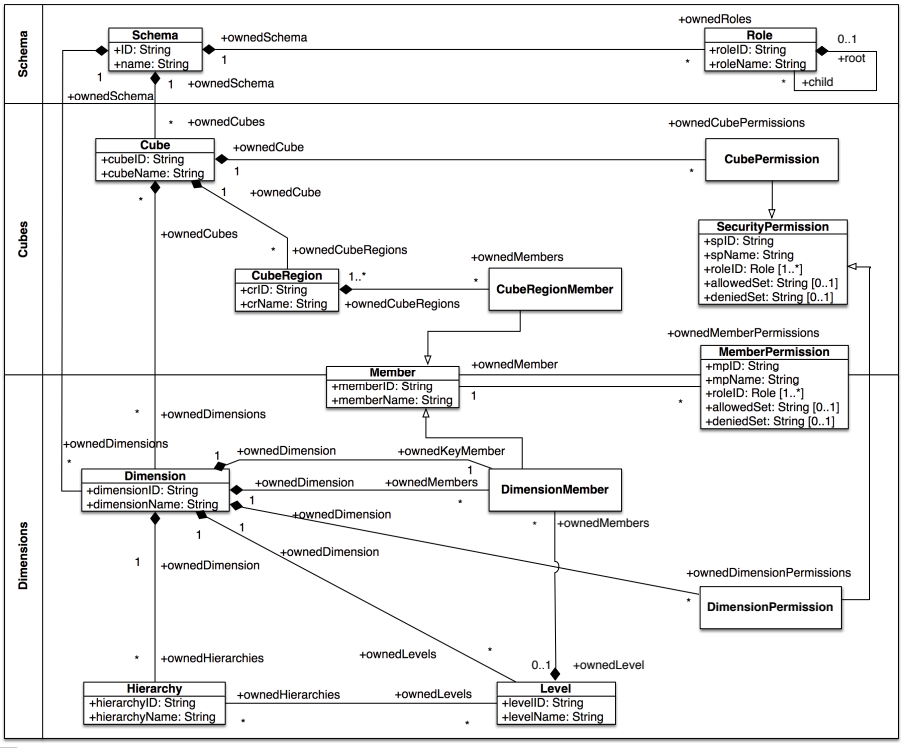


Figure 4: Logical Metamodel for Secure OLAP applications (SECMDDW)

The remainder parts of our metamodel define both structural and security aspects for cubes and dimensions. Fact classes from our conceptual model

are defined at the logical level as cubes (Cube metaclass) and their related attributes as measures (MeasureGroup and Measure metaclasses). Furthermore, conceptual dimension classes are defined as dimensions (Dimension metaclass) with a set of attributes (Attribute metaclass) and an identifier (KeyAttribute metaclass). Base classes from conceptual model are specified as attributes related with the corresponding dimension, creating therefore the necessary hierarchies (Hierarchy and Level metaclasses).

Considering the abstraction gap between the conceptual and logical levels, security rules (SIAR and AUR) from conceptual models (including complex security rules defined with OCL expressions) have been represented at the logical level by using sets of security permissions associated with cubes, cells, dimensions and attributes. Nevertheless, since OLAP platforms provide specific auditing tools which are directly managed by administrators, audit rules (AR) have not been transformed into the logical model.

The SIAR and AUR security rules specified in conceptual models related with fact classes, are defined in logical models as permissions associated with the corresponding cubes (CubePermission metaclass). Each cube permission is related with a certain security role (RoleID attribute) and uses positive and negative expressions (AllowedSet and DeniedSet attributes) in order to establish the information which has to be shown or hidden for that role. This metamodel also allows the establishment of fine grained permissions over cube measures by using cell permissions associated with the corresponding cube permission (CellPermission metaclass).

On the other hand, security constraints established in conceptual models involving dimension and base classes are defined at the logical level as permissions associated with dimensions (DimensionPermission metaclass) by including information about the security role (RoleID attribute) and conditions with the information which can be accessed or not for that role (AllowedSet and DeniedSet attributes). The definition of fine grained security constraints is also permitted with the use of attribute permissions associated with the corresponding dimension (AttributePermission metaclass).

5.2. Transformation from Conceptual Models

This section describes the transformation rules developed for generating logical models for secure OLAP applications (according to SECMDDW) from conceptual models (according to SECDW). Currently, it is possible to find many languages to implement model to model transformations, but in this work we have served from QVT [7], that is the standard proposed by the

OMG. These transformations uses the relational layer of QVT that supports the specification of relationships that must hold between MOF models by means of a relations language. Each QVT transformation is composed of a top relation that starts the execution of several rules (relations).

The mapping between conceptual and logical models has been organized in five main transformations:

- SECDW2Role which generates the security configuration for the OLAP system by using a role based access control (RBAC) policy.
- SECDW2Cube and SECDW2Dimension which analyze both structural aspects and security information associated with cubes and dimensions, and create in the logical model measures, attributes, hierarchies, etc. and security permissions attached to multidimensional elements (cubes, dimensions, cells and attributes).
- SecurityRules2CubePermissions and SecurityRules2DimensionPermissions transformations, which process more complex security rules defined in the conceptual model by using the SecurityRule and AuthorizationRule metaclasses, and creates in the logical model the necessary security permissions which are associated to cubes and dimensions (or to cube cells and dimension attributes if fine grained security permissions are needed).

Next, each transformation is described, but only the graphical notation for some rules are shown.

5.2.1. Processing Security Configuration

Firstly, SECDW2Role analyzes and transforms the security configuration defined in the conceptual model into the logical level which is closer to OLAP applications, which use to consider a RBAC security policy. Since our conceptual model is richer than our logical model, allowing the definition of security roles (SR), levels (SL) and compartments (SC), this information has to be adapted to a RBAC policy. The transformation SECDW2Role explores the security configuration and establishes an RBAC policy by inferring the necessary security roles.

The top relation SecurePackage2RoleSchema explores the source model (the conceptual model) looking for SecurePackage classes which contain the structure of the DW linked with the security information (that is composed

of security levels, roles and compartments). For each SecurePackage found, a Schema element is created in the target model (the logical model). Then, the relations SCompartment2Role, SLevel2Role and SRole2Role generate the roles considered in the RBAC policy. That is, each security compartment, role and level defined in our source model is added to the Schema as a role.

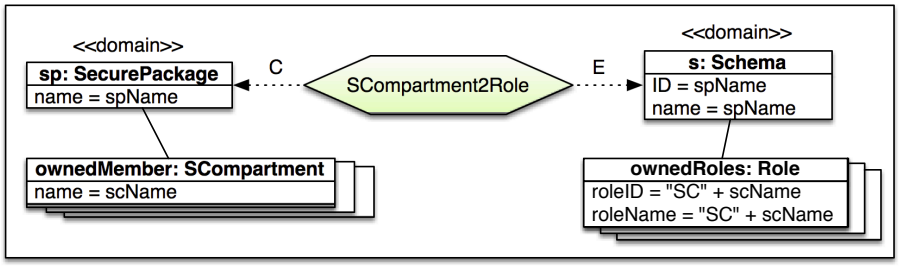


Figure 5: Relation SCompartment2Role

Figure 5 shows the graphical notation for the relation SCompartment2Role which checks secure packages (sp: SecurePackage) and their associated security compartments (ownedMember: SCompartment). Then, if it does not exist, a schema for the secure package is created (s: Schema) and new roles are attached to the schema (ownedRoles: Role) for each secure compartment. Role attributes, roleID and roleName, are set to SC concatenated with the name of the security compartment.

5.2.2. Processing Facts

The secure fact classes defined in the conceptual model are represented in the logical model as cubes. SECDW2Cube is the transformation in charge of processing this information and as Figure 6 shows, it is composed of a top relation SPackage2CubeSchema and several relations which have been grouped in several categories: structural relations and security relations (for cube and cell permissions). Structural relations generate in the logical model structural elements such as cubes and measures, whereas security relations process security information associated with secure fact classes and their attributes in order to generate security permissions associated with cubes and their measures in the logical model. More complex security rules defined over secure fact classes, which evaluate expressions to assign different security permissions, will be processed by the transformation SecurityRules2CubePermissions.

The top relation SPackage2CubeSchema is in charge of creating the schema (if it has not been previously created) and after that, it creates the structural

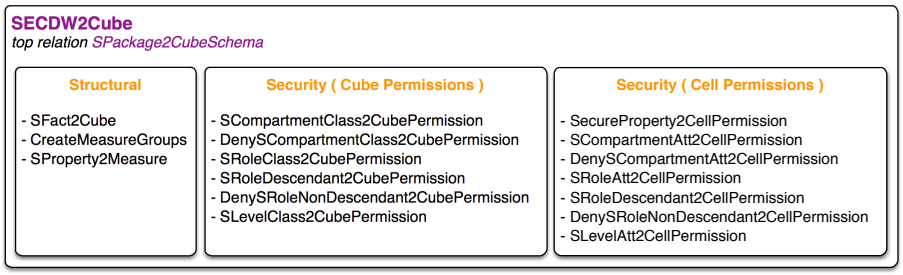


Figure 6: SECDW2Cube transformation

elements by using the relations SFact2Cube which defines a Cube for each SFact class, and CreateMeasureGroups which creates a MeasureGroup associated with each cube and uses the relation SProperty2Measure to analyze the attributes of the SFact and for each one, to include a Measure in the MeasureGroup previously created.

Once structural elements have been created, SFact2Cube launches the relations that analyze and process the security information (ownedSecInf) associated with SFact classes (group security cube permissions in Figure 5) and their attributes (group security cell permissions in Figure 5).

Figure 7 shows, using the QVT graphical syntax, a detailed view of the SFact2Cube relation. The way in which the security information associated with SFact classes and their attributes are processed by other relations can be shown in the where clause of the relation (see bottom side of the Figure 7).

Firstly, security compartments, roles and levels defined over SFact classes are analyzed creating security permissions that affect the Cube as a whole (CubePermission). For instance, the relation SCompartmentClass2CubePermission is focused on the security compartments and creates positive cube permissions authorizing the access to the cube for the involved roles (that are the representation of these compartments as roles in the logical model). Then, the relation DenySCompartmentClass2CubePermission creates negative cube permissions denying the access for the remainder security compartments.

Since security information could be also attached to SFact attributes, the relation SecureProperty2CellPermission processes fine grained security information defined over SFact attributes in a similar way as SFact2Cube does, but using auxiliary relations specifically created for this purpose (group se-

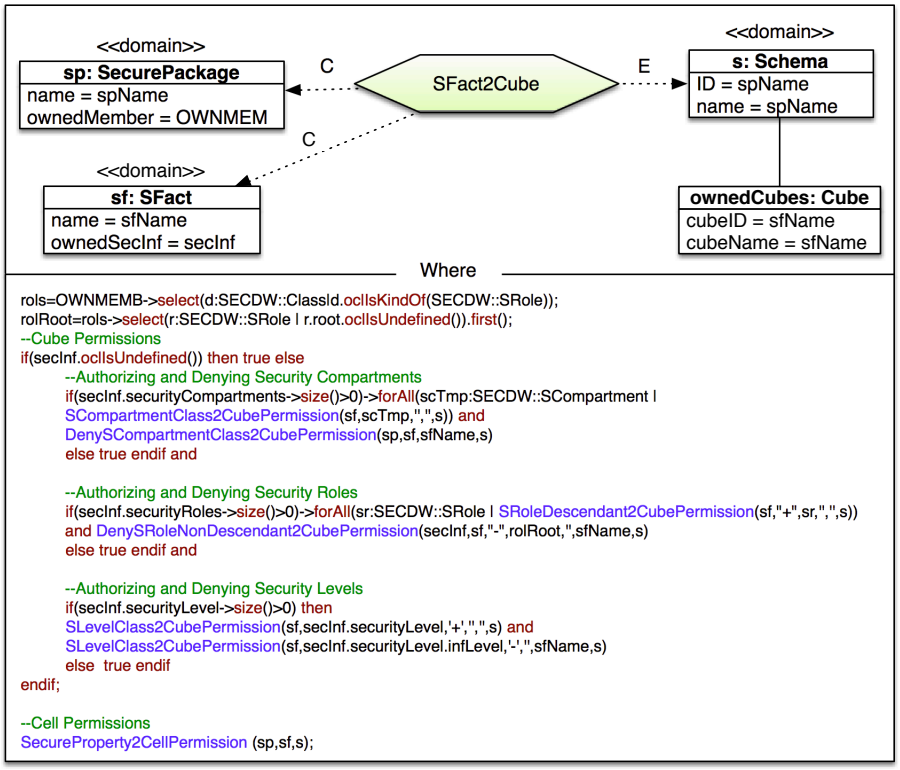


Figure 7: Relation SFact2Cube

curity cell permissions in Figure 6). That is, the definition of the security level, compartments and roles which can access to the attribute. For instance, the specification of certain security compartments is analyzed by the SCompartmentAtt2CellPermission relation (shown in Figure 8) which creates a positive cell permission granting the access to the information for authorized compartments (which are represented as roles in the logical model). Then, the relation DenySCompartmentAtt2CellPermission includes several negative cell permissions denying accesses for the rest of compartments.

5.2.3. Processing Dimensions and Bases

The secure dimension and base classes defined in the conceptual model are processed by the transformation SECDW2Dimension which generates in the logical model dimensions, bases, attributes, hierarchies and security permissions defined over dimensions and attributes. As Figure 9 shows, this transformation is composed of the top relation SPackage2DimensionSchema

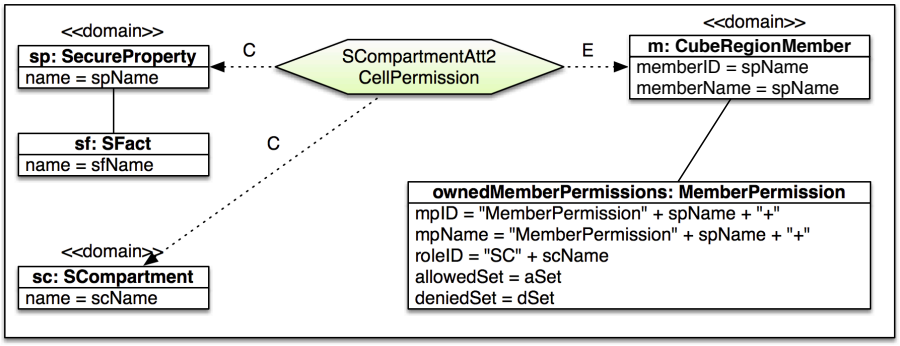


Figure 8: Relation SCompartmentAtt2CellPermission

and several sets of relations focused on structural aspects such as dimensions, attributes, bases and hierarchies; and the processing of security constraints defined over dimensions and attributes.

Firstly, the relation SDimension2Dimension processes secure dimension classes (SDimension) defined in the conceptual model and creates in the logical model dimensions (Dimension) attached to the corresponding cube (ownedDimensions). Then, dimension properties (SProperty) are transformed into attributes (KeyAttribute and Attribute). On the other hand, the remainder structural relations analyze the secure base classes (SBase) that represent the different aggregation levels in which dimensions can be classified. Thus, base classes and their attributes are represented in the logical model as attributes associated with dimensions, classification hierarchies and aggregation levels.

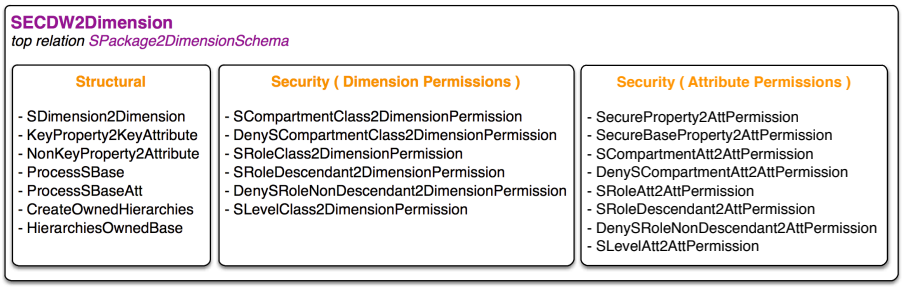


Figure 9: SECDW2Dimension transformation

Figure 10 shows the relation CreateOwnedHierarchies as an example of these relations focused on base classes. This relation checks the base classes

(ownedSBases) related with each dimension (sd:SDimension) and in the logical model, attaches to the corresponding dimension (dim:Dimension) the classification hierarchies needed (ownedHierarchies) and the different aggregation levels that compose each hierarchy (ownedLevels).

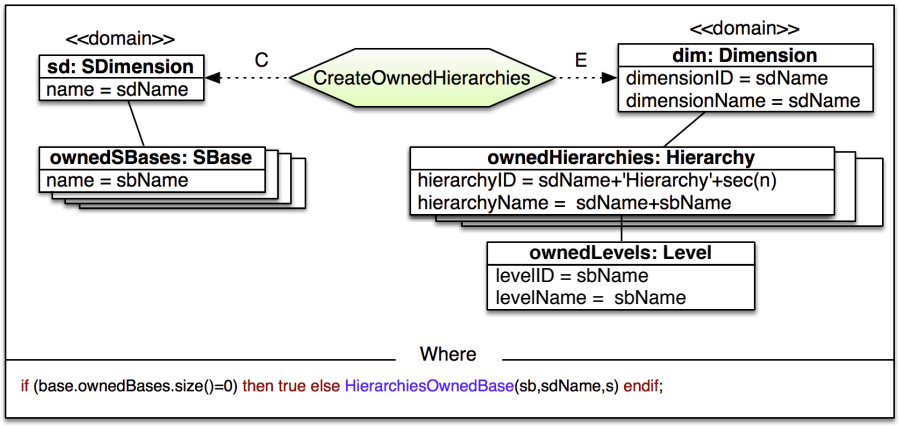


Figure 10: Relation CreateOwnedHierarchies

Next, security information associated with dimension classes, base classes or their attributes (ownedSecInf) is processed by specific relations (the groups security dimension permissions and security attribute permissions showed in Figure 9). Focusing on dimensions, the relation SDimension2Dimension analyzes the security compartments, roles and levels associated with dimension classes and uses auxiliary security relations (group security dimension permissions) to transform it into permissions defining which roles (at the logical level) can access each dimension (DimensionPermission). Then, the relation SecureProperty2AttPermission processes fine grained security information defined over dimension attributes by using the group of relations security attribute permissions which defines in the logical model security permissions associated with attributes (AttributePermission). Finally, security information defined over base classes and their attributes are also analyzed by using the relation SecurityBaseProperty2AttPermission and the group of relations security attribute permissions. In this case attribute permissions are also created, since the base classes (and their attributes) defined in the conceptual model are transformed into dimension attributes in the logical model.

An example of relation focused on security is shown in Figure 11. The relation SLevelClass2DimensionPermission analyzes the security level (SLevel)

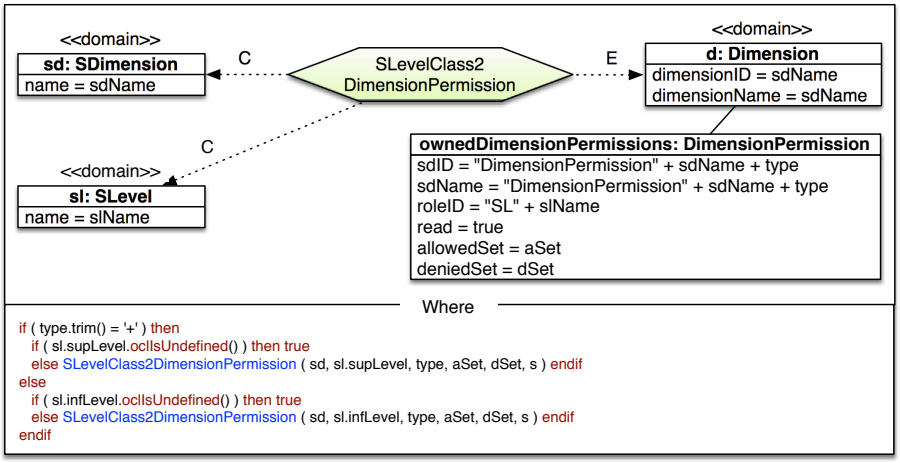


Figure 11: Relation SLevelClass2DimensionPermission

associated with secure dimension classes (SDimension) and transforms it into permissions (DimensionPermission) attached to the corresponding dimension (ownedDimensionPermissions) at the logical level. For the security level specified, it creates in the logical model a set of positive permissions to allow the access for the roles that represent the security level specified and their upper levels, and a set of negative permissions to deny accesses from inferior security levels. Positive and negative permissions are defined by setting the properties allowedSet and deniedSet with the multidimensional elements which should be shown or hidden.

5.2.4. Processing Security Rules

Security constraints defined as security information (security compartments, roles and levels) attached to cubes, dimensions, bases and their attributes (ownedSecInf), have been processed by the previous transformations. Nevertheless, our conceptual model allows the definition of more complex security rules by using the metaclasses SecurityRule (for SIAR rules), AuthorizationRule (for AUR rules) and AuditRule (for AR rules). Since these rules can involve several multidimensional elements and the evaluation of conditions, requires specific transformations to process them. The remainder transformations, SecurityRule2CubePermission and SecurityRule2-DimensionPermission, deal with the transformation of SIAR and AUR security rules from conceptual to logical models, but in this proposal the transformation of audit rules, has not been considered since commercial tools include

specific tools for auditing which administrators directly manage.

The transformation SecurityRule2CubePermission processes security rules attached to secure fact classes and their attributes. It is mainly composed of a top relation ProcessCubeSecurityRules and two relations: ProcessCubeSIAR which analyzes security information assignment rules (SIAR) and ProcessCubeAUR which are focused on authorization rules (AUR). On the other hand, the transformation SecurityRule2DimensionPermission processes security rules associated with dimension and base classes and their attributes and its composition is similar to the previous transformation. It is composed of a top relation ProcessDimensionSecurityRules and two relations for SIAR and AUR security rules, ProcessDimensionSIAR and ProcessDimensionAUR.

These relations analyze security rules defined in the conceptual model attached to fact, dimension or base classes or their attributes, and represent this information in the logical model by using security permissions attached to cubes, dimensions or attributes. In order to achieve this goal, both transformations serve from the groups of relations previously commented which are focused on the creation of the different kind of security permission in logical models. That is, the groups of relations focused on cubes: security cube permissions and security cell permissions (Figure 5); and those focused on dimensions: security dimension permissions and security attribute permissions (Figure 8).

Security Information Assignment Rules (SIAR) assign to several objects a specific set of security privileges depending on a condition. This kind of rule is defined in the conceptual model by using the SecurityRule metaclass. The classes and properties affected by the rule are expressed in the properties involvedClasses, ownedSCObjects and ownedSPObjects. The condition to be evaluated is expressed in the CABExp property as a boolean expression. Finally, two sets of security information (security level, compartments and roles) establish the security privileges needed to access the objects affected by the rule if the condition if satisfied (CATHENSecInf) or not (CAELSESecInf).

The relations ProcessCubeSIAR and ProcessDimensionSIAR are focused on SIAR security rules, analyzing their information and generating in the logical model the security permissions (cube, cell, dimension and attribute permissions) needed to avoid unauthorized accesses to the affected objects. As can be seen in Figure 12, the relation ProcessCubeSIAR checks SecurityRule (SIAR) elements and their attributes (the affected secure fact classes, the condition to evaluate and the security information sets to be applied).

Then, it analyzes the security information expressed in the SIAR (that is composed of security compartments, roles and levels) and uses auxiliary relations previously defined (such as SLevelClass2CubePermission, SRoleDescendant2CubePermission, etc.) in order to generate positive and negative cube and cell permissions for the authorized and unauthorized security roles at the logical level (which represent security compartments, roles and levels from the conceptual model). The condition established in the CABExp property is used in the definition of the negative permissions (deniedSet attribute) to hide the instances that satisfy the condition. The relation ProcessDimensionSIAR works in a similar way but focusing on secure dimension and base classes by checking these kind of elements in the objects affected by the SIAR rules and creating dimension and attribute permissions with the corresponding auxiliary relations.

Authorization Rules (AUR) are defined in the conceptual model by using the metaclass AuthorizationRule. This kind of security rule grants or denies (ExceptSign property with values + or -) certain privileges (ExceptPrivilege property) to certain elements (involvedClasses, ownedSCObjects and ownedSPObjects properties) to the users that satisfy the condition expressed in the AUR (CABExp property) and the security information (ownedSecInf property).

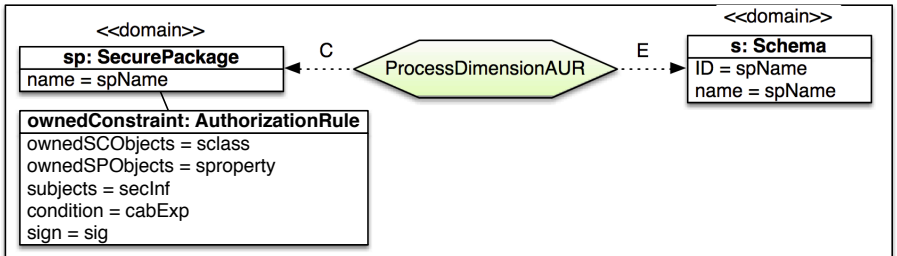
Two relations check the AURs established in the conceptual model and create the necessary security permissions in the logical model, which will be positive or negative permissions depending on the AURs sign. The relation ProcessCubeAUR establishes cube and cell permissions, whereas ProcessDimensionAUR defines dimension and attribute permissions.

Figure 13 shows the relation ProcessDimensionAUR. This relation checks AURs from the conceptual model and for each secure dimension or base class involved, establish dimension and attribute permissions in the logical model by considering the condition and the security information (security compartments, roles and levels) defined in the AUR. Solely positive or negative permissions are created depending on AURs sign. For instance, if AURs sign is negative, permissions to deny accesses are created by considering the security information. That is, negative permissions for roles (at the logical level), which represent the level (and inferior levels), role (and their descendants), and compartment established in the AURs ownedSecInf property. In order to achieve this goal, auxiliary relations previously defined are used with specific parameters.

Figure 12: Relation ProcessCubeSIAR

6. Step 3: Transformation to OLAP implementation.

This section shows the transformations which have been developed in order to automatically obtain the secure OLAP implementation from logical models. In this case, we have defined model-to-text transformations by using



Where

```
SDIM=class->select(d:SECDW::Class | d.oclIsKindOf(SECDW::SDimension or SECDW::SBase));
sp=sproperty->select(spro:SECDW::SecureProperty | spro.secureClass.oclIsKindOf(SECDW::SDimension or
SEWDW::SBase));

--DimensionPermissions
if(secInf.oclIsUndefined()) then true else

--Authorizing and Denying Security Levels
if(secInf.securityLevel->size()=0) then true else
  if(sig='+') then SDIM->forAll(sd:SECDW::SDimension |
    SLevelClass2DimensionPermission(sd,secInf.securityLevel,sig,cabExp,'',s))
  else if(sig='-') then SDIM->forAll(sf:SECDW::SDimension |
    SLevelClass2DimensionPermission(sd,secInf.securityLevel.infLevel,sig,'',cabExp,s))
  else true endif endif
endif and
--Authorizing and Denying Security Roles
if(secInf.securityRoles->size()=0) then true else
  if(sig='+') then SDIM->forAll(sd:SECDW::SDimension | secInf.securityRoles->forAll(sr:SECDW::SRole |
    SRoleDescendant2DimensionPermission(sd,sig,sr,cabExp,'',s)))
  else if(sig='-') then SDIM->forAll(sd:SECDW::SDimension | secInf.securityRoles->forAll(sr:SECDW::SRole |
    DenySRoleNonDescendant2DimensionPermission(sd,sig,sr,'',cabExp,s)))
  else true endif endif
endif and
--Authorizing and Denying Security Compartments
if(secInf.securityCompartments->size()=0)then true else
  if(sig='+') then SDIM->forAll(sd:SECDW::SDimension |
    secInf.securityCompartments->forAll(scTmp:SECDW::SCompartment |
    SCompartmentClass2DimensionPermission(sd,scTmp,cabExp,'',s)))
  else if(sig='-') then SDIM->forAll(sd:SECDW::SDimension |
    secInf.securityCompartments->forAll(scTmp:SECDW::SCompartment |
    DenySCompartmentClass2DimensionPermission(sd,scTmp,'',cabExp,s)))
  else true endif endif
endif and

--AttributePermissions
--Authorizing and Denying Security Levels
if(secInf.securityLevel->size()=0) then true else
  if(sig='+') then sp->forAll(spr:SECDW::SecureProperty |
    SLevelAtt2AttPermission(spr,secInf.securityLevel,sig,cabExp,'',s))
  else if(sig='-') then sp->forAll(spr:SECDW::SecureProperty |
    SLevelAtt2AttPermission(spr,secInf.securityLevel.infLevel,sig,'',cabExp,s))
  else true endif endif
endif and
--Authorizing and Denying Security Roles
if(secInf.securityRoles->size()=0) then true else
  if(sig='+') then secInf.securityRoles->forAll(sr:SECDW::SRole | sp->forAll(spr:SECDW::SecureProperty |
    SRoleDescendant2AttPermission(spr,sig,sr,cabExp,'',s)))
  else if(sig='-') then secInf.securityRoles->forAll(sr:SECDW::SRole | sp->forAll(spr:SECDW::SecureProperty |
    SRoleNonDescendant2AttPermission(spr,sig,sr,'',cabExp,s)))
  else true endif endif
endif and
--Authorizing and Denying Security Compartments
if(secInf.securityCompartments->size()=0) then true else
  if(sig='+') then secInf.securityCompartments->forAll(scTmp:SECDW::SCompartment |
    sp->forAll(spr:SECDW::SecureProperty | SCompartmentAtt2AttPermission(spr,scTmp,cabExp,'',s)))
  else if(sig='-') then secInf.securityCompartments->forAll(scTmp:SECDW::SCompartment |
    sp->forAll(spr:SECDW::SecureProperty | DenySCompartmentAtt2AttPermission(spr,scTmp,'',cabExp,s)))
  else true endif endif endif
endif;
```

Figure 13: Relation ProcessDimensionAUR

the MOFScript specification language that is the standard proposed by the OMG.

Since logical models defined according SECMDDW are very close to OLAP platforms, managing multidimensional elements and security constraints defined over them, the automatic transformation towards a secure implementation for different OLAP tools can be easily achieved. In this paper SSAS has been selected as the target OLAP platform. Therefore, this section describes the MOFScript transformations developed to obtain the secure implementation for SSAS from logical models. Only the implementation for some transformations are shown.

SSAS uses several kinds of XML files (with role, cube and dim file extensions) to manage information about security roles, cubes and dimension. Thus, the MOFScript M2T transformation developed to generate the secure SSAS implementation from logical models has been grouped in three sets, which are security configuration, cubes and dimensions. Next, each set of transformations is briefly explained and some pieces of code are shown. Firstly, the security configuration established in the logical model by using a RBAC policy is processed (show Table 1). For each role, a XML file (with role file extension) with information about its members.

```
texttransformation roles (in psm:SECMDDW) {
psm.Schema::main() {
self.ownedRoles->forEach(r:psm.Role) {
 file rolefile (r.ID + .role);
 rolefile.print(<Role>);
 rolefile.print(<ID> + r.ID + </ID>);
 rolefile.print(<Name> + r.name + </Name>);
 rolefile.print(<Members >);
 r.ownedMembers->forEach(m:psm.Member) {
  rolefile.print(<Member><Name>+m.memberName+</Name></Member>); }
rolefile.print(</Members ></Role>); } }
```

Table 1: MOFScript transformation: security configuration

Then, structural and security aspects related with cubes are analyzed. Table 2 shows a piece of code of this transformation. It creates a cube file (cube file extension) for each detected cube in the logical model, including information about the cube and its measures. Next, cube files are fulfilled by other transformations which include the remainder information about hierarchies and security permissions defined over cubes and cells.

```
texttransformation cubes (in psm:SECMDDW) {
psm.Schema::main() {
self.ownedCubes->forEach(c:psm.Cube) {
 file cubefile (c.cubeID + .cube);
 cubefile.print(<Cube>);
 cubefile.print(<ID> + c.cubeID + </ID>);
 cubefile.print(<Name> + c.cubeName + </Name>);
 cubefile.print(<MeasureGroups>);
 c.ownedMeasureGroups->forEach(mg:psm.MeasureGroup) {
  cubefile.print(<MeasureGroup>);
  cubefile.print(<ID> + mg.mGroupID + </ID>);
  cubefile.print(<Name> + mg.mGroupName + </Name>);
  cubefile.print(<Measures>);
  mg.ownedMeasures->forEach(m:psm.Measure) {
   cubefile.print(<Measure>);
   cubefile.print(<ID> + m.measureID + </ID>);
   cubefile.print(<Name> + m.measureName + </Name>);
   cubefile.print(</Measure>); }
  cubefile.print(</Measures>);
  cubefile.print(</MeasureGroup>); }
 cubefile.print(</MeasureGroups>);
cubefile.print(</Cube>); }  }
```

Table 2: MOFScript transformation: cubes

Table 3 shows a transformation for dimensions which is focused on security issues represented in the logical model. Finally, dimensions are processed creating the dimension files needed (dim file extension) and the structural aspects related with dimensions. After that, the security permissions defined over this dimension or their attributes are analyzed and the security information needed is included in the corresponding dimension file: information about processing and reading privileges, MDX expressions defining the sets which are denied and allowed for each role, etc. (see Table 3).

7. Validation Example

This section describes the application of our proposal for the development of an OLAP application for a sales department. The DW used in this example analyzes sales according to different perspectives (products, dates, customers and stores) and also considers several security constraints. Firstly, the system is modeled in a conceptual level. Then, the secure logi-

```
texttransformation dimensionPermissions (in psm:SECMDDW) {
psm.Schema::main() {
self.ownedCubes->forEach(c:psm.Cube) {
 c.ownedDimensions->forEach(d:psm.Dimension) {
  file dimfile (d.dimensionID + .dim);
  dimfile.print (<DimensionPermissions>);
  d.ownedDimensionPermissions->forEach(dp:psm.DimensionPermission) {
   dimfile.print (<DimensionPermission>);
   dimfile.print (<ID>+dp.dpID+</ID>);
   dimfile.print (<Name>+dp.dpName+</Name>);
   dimfile.print (<RoleID>+dp.roleID+</RoleID>);
   dimfile.print (<Process>+dp.process+</Process>);
   dimfile.print (<Read>+dp.read+</Read>);
   dimfile.print (<AllowedSet>+dp.allowedSet+</AllowedSet>);
   dimfile.print (<DeniedSet>+dp.deniedSet+</DeniedSet>);
   dimfile.print (<AttributePermissions>);
   dp.ownedAttributePermissions->forEach(ap:psm.AttributePermission) {
    dimfile.print (<AttributePermission>);
    dimfile.print (<AttributeID>+ap.attributeID+</AttributeID>);
    dimfile.print (<AllowedSet>+ap.allowedSet+</AllowedSet>);
    dimfile.print (<DeniedSet>+ap.deniedSet+</DeniedSet>);
    dimfile.print (</AttributePermission>"); }
   dimfile.print (</AttributePermissions>);
   dimfile.print (</DimensionPermission>); }
  dimfile.print ("</DimensionPermissions>");  }  }  }
```

Table 3: MOFScript transformation: dimensions

cal model for OLAP applications is automatically obtained (by applying the QVT transformations defined). Finally, the secure implementation for SSAS is automatically obtained from the logical model (by applying the MOFScript rules developed).

7.1. Step 1: Conceptual Model

Figure 14 shows the conceptual model for the Sales DW, defined according to SECDW. This example is composed of a central fact Sale (secure fact class) with measures amount and quantity, which can be classified by using different dimensions Product, Store, Date and Customer (secure dimension classes). Furthermore, different aggregation levels have been defined (by using secure base classes) for Products which can be grouped by Category and for Customers which can be grouped by City.

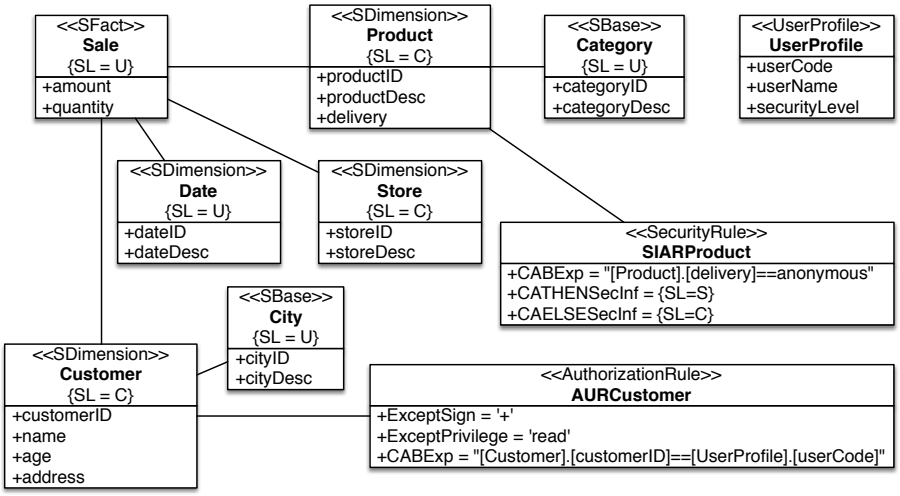


Figure 14: Example: Conceptual Model (PIM)

On the other hand, the security configuration is established. Although our proposal allows to specify it based on security roles, levels and compartments, this example only considers the following set of security levels: secret (S), confidential (C) and undefined (U). Therefore, the user profile stores the security level associated with the user as well as information about its identification and name. Security constraints have been established over multidimensional elements by using this security configuration, that is, by using security information sets composed of a specific security level required. Firstly, the security privileges needed to access fact, dimension and base classes are defined (as tagged values): a security level of C is required to access Product, Store and Customer dimensions; and a level of U for the fact Sale, dimension Date and bases Category and City.

Moreover, several security rules complement the model. A security rule (SIAR) attached to Product dimension, which increases the security level necessary to access sales information grouped by Product from C to S if the kind of delivery is anonymous (delivery property from Product dimension). An authorization rule (AUR) attached to Customer allows each user to access its own customers information (although users security level was lower than the required one for Customer, that is C).

7.2. Step 2: Transformation to Logical Model for OLAP

Once the conceptual model has been defined, in this section the transformation rules developed are applied in order to obtain a logical model for secure OLAP applications (according to SECMDDW metamodel). The resulting logical model shown in this section splits in several figures. Firstly, the security configuration defined is analysed by the transformation SECDW2Role. In this case, as can be seen in Figure 15, just a specific role (SLS, SLC and SLU) is created in the logical model for each security level from the conceptual model (S, C and U).

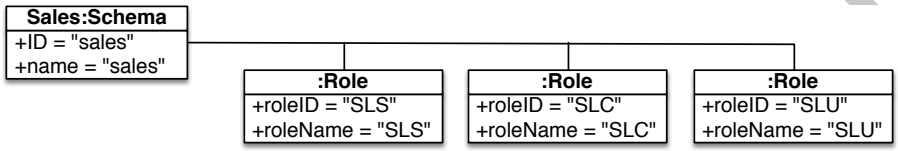


Figure 15: Example: Logical Model (PSM). Security Configuration

Secure fact classes defined in the conceptual model are processed by SECDW2Cube transformation. The secure fact class Sale generates at the logical model a cube (Cube) with an attached measure group (MeasureGroup) composed of two measures (Measure): amount and quantity (see Figure 16). The security level required for accesses to the Sale secure fact class (a U security level) is represented in the logical model as a set of positive security cube permission which grant accesses (process and read attributes) to the cube Sale (allowedSet attribute) for the role SLU (roleID attribute), and also for the roles SLC and SLC, since these roles represent users with upper security privileges (upper security level).

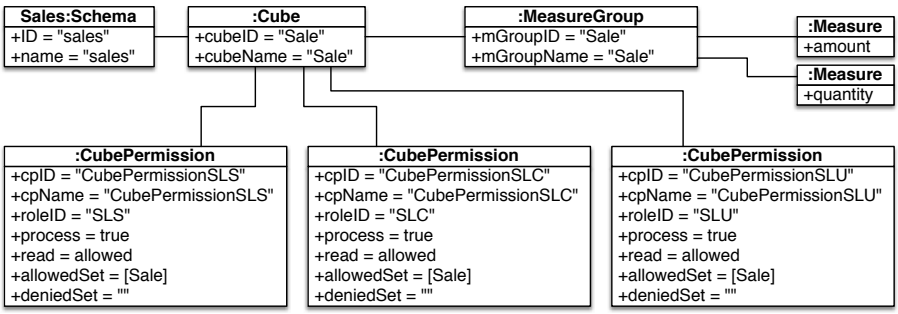


Figure 16: Example: Logical Model (PSM). Cubes

Then, both structural and security aspects related with dimension and base classes are analyzed by the transformation SECDW2Dimension. As Figure 17 shows, each secure dimension class from the conceptual model generates a dimension (Dimension) in the logical model with associated attributes (Attribute) and a key attribute for each dimension (KeyAttribute). The different classification hierarchies defined for each dimension with secure base classes in the conceptual level, are specified in the logical model as hierarchies (Hierarchy) and different aggregation levels (Level) associated with dimensions (for instance, Customer can be grouped by City). Bases properties are represented as dimension attributes in the logical level (for instance, cityID and cityDesc from the base City are now attributes of Customer dimension).

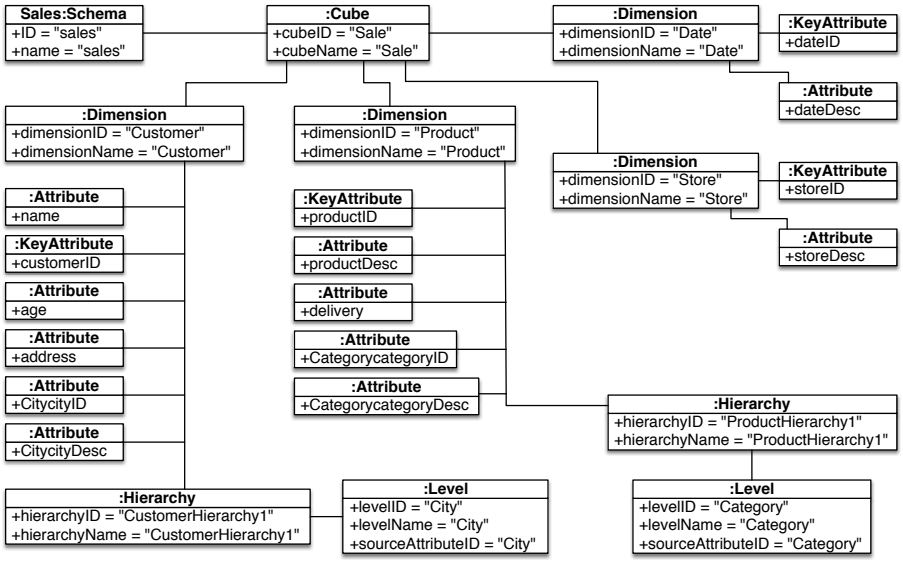


Figure 17: Example: Logical Model (PSM). Dimensions Structure

The security privileges needed to access dimensions and bases are modeled at the logical level as dimension permissions (as can be shown in Figure 18). Since a security level of C is required to access the dimension Store, three security dimension permissions have been defined: two of them to grant access for users with roles SLS and SLC (security levels S and C), and one of them to deny access for users with a SLU role (security level U). The remainder dimension permissions in Figure 18 are associated with the dimension Date and are obtained in a similar way.

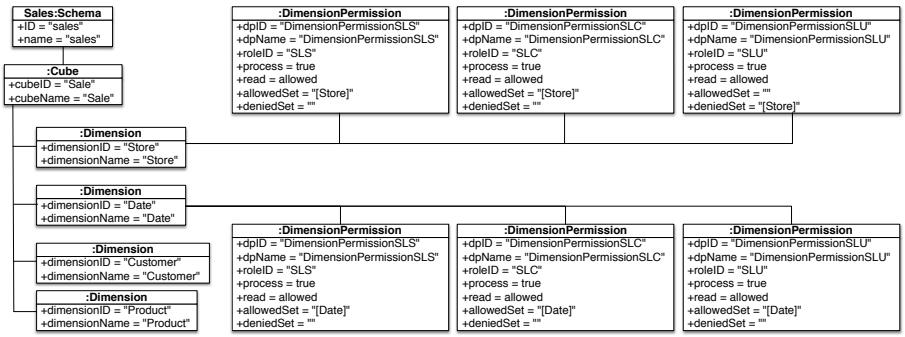


Figure 18: Example: Logical Model (PSM). Dimensions Security

The dimension and attribute permissions needed to represent at the logical level the security constraints that affect Product dimension and their associated base class Category are shown in Figure 19. Firstly, dimension permissions are created for each role (SLS, SLC and SLU). The security level required to access Product is C, but there is also a security rule (SIAR) that increase it to S when the kind of delivery is anonymous. The transformation SecurityRules2DimensionPermission processes this SIAR generating dimension permissions that represent this situation by using allowed and denied sets: SLS role can access all products, with anonymous delivery or not (then, the allowedSet is set to [Product]); SLC role can only access products with not anonymous delivery (deniedSet set to [Product].[delivery]==anonymous); and SLU role cannot access any product (deniedSet set to [Product]). Nevertheless, the last dimension permission is denying all accesses to products for the SLU role and in the conceptual model, the security level required for the base class Category is U. In order to provide accesses to Category information (categoryID and categoryName properties) for the role SLU, positive attribute permissions for the attributes CategorycategoryID and CategorycategoryName are created in the logical model attached to the dimension permission corresponding to the role SLU.

Dimension permissions associated with Customer dimension are shown in Figure 20. In the conceptual model, it was established a security level of C for Customer dimension and U for City base. Security dimension permissions are created in the logical model for each role: granting accesses for SLS and SLC, and denying accesses for SLU. Furthermore, since users with a U security level cab see City information, positive attribute permissions for their attributes (CitycityID and CitycityName) are attached to the dimen-

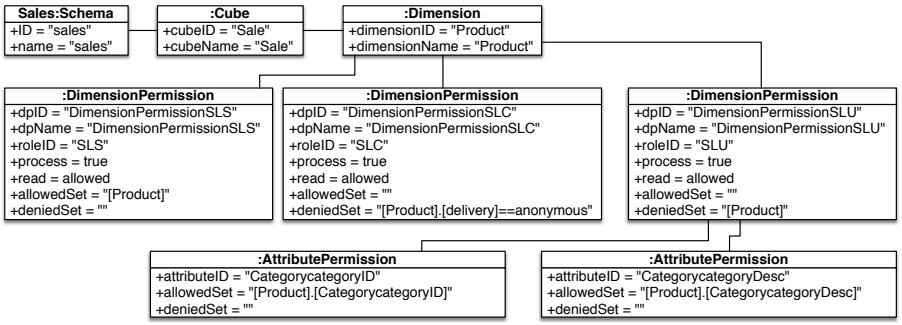


Figure 19: Example: Logical Model (PSM). Security Rules

sion permission for the role SLU. Nevertheless, an authorization rule (AUR) allows users to access their own customer information. The transformation SecurityRule2DimensionPermission changes in the logical model the information needed to represent this constraint. In this case, since SLS and SLC roles can access all Customer information, only the dimension permission for the SLU role has to be modified by setting the allowed set to the condition [Customer].[customerID] == [UserProfile].[userCode].

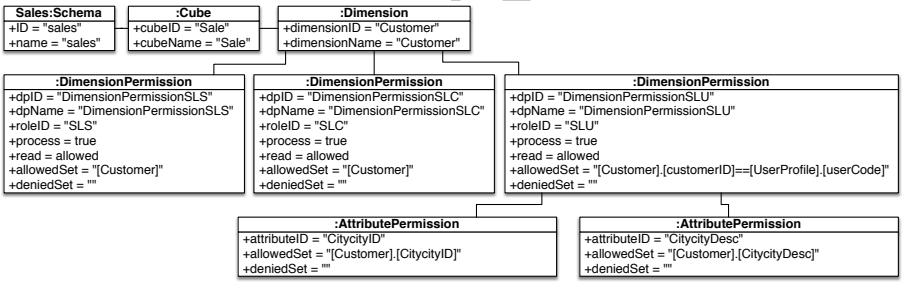


Figure 20: Example: Logical Model (PSM). Authorization Rules

7.3. Step 3: Transformation to OLAP implementation

The logical model obtained in the previous section is closer to OLAP applications than the conceptual model, allowing the generation of the eventually secure implementation for different OLAP tools. In this paper, MOFScript transformations have been developed for generating the implementation in one of the most used OLAP tools, SSAS. This transformation generates different kind of files: for each security role (files SLS.role, SLC.role and

SLU.role); for the Sales cube (Sales.cube file); and for each dimension (files Product.dim, Customer.dim, etc.).

Table 4 shows a piece of code from the Product.dim file in which can be shown how the dimension permission for the role SLU has been represented for SSAS by using its XML syntax and MDX expressions for the allowed and denied sets.

```
<Dimension>
 <ID>Product</ID><Name>Product</Name>
 <!--Structural aspects have been omitted-->
 <DimensionPermissions>
  <!--Dimension Permissions for SLS and SLC have been omitted-->
  <DimensionPermission>
   <ID>DimensionPermissionSLU</ID>
   <Name>DimensionPermissionSLU</Name>
   <RoleID>SLU</RoleID>
   <Process>true</Process><Read>Allowed</Read>
   <AllowedSet></AllowedSet>
   <DeniedSet>[Product]</DeniedSet>
   <AttributePermissions>
    <AttributePermission>
     <AttributeID>CategorycategoryID</AttributeID>
     <AllowedSet>[Product].[CategorycategoryID]</AllowedSet>
     <DeniedSet></DeniedSet>
    </AttributePermission>
    <AttributePermission>
     <AttributeID>CategorycategoryDesc</AttributeID>
     <AllowedSet>[Product].[CategorycategoryDesc]</AllowedSet>
     <DeniedSet></DeniedSet>
    </AttributePermission>
   </AttributePermissions>
  </DimensionPermission>
 </DimensionPermissions>
</Dimension>
```

Table 4: Example: SSAS implementation

8. Conclusions

DWs manage vital business information which is very sensitive and has to be correctly assured in order to avoid unauthorized access. Because an early detection of security requirements has influence in the further design decisions

providing better security specifications and final products, security measures should be considered in the whole development process from early stages to final tools. Furthermore, since DWs are queried by tools which manage specific cubes or views from the corporative DW, security constraints should be also defined in this metadata layer by using the same multidimensional elements that will be managed by the final users of OLAP tools.

Thanks to our proposal we are able to develop secure OLAP applications, providing a complete MDA architecture composed of several security models and automatic transformations towards the final secure implementation. That is: (i) a new logical multidimensional metamodel (SECMDDW) which is based on the OLAP package of CWM extended with security issues allowing the definition of the security configuration of the system and both structural and security aspects of cubes and dimensions; (ii) a set of QVT transformations from our conceptual models (PIM) to the new multidimensional logical models (PSM), which integrates the new PSM metamodel in the MDA architecture; (iii) the corresponding model-to-text transformations which allows the automatic code generation into a specific OLAP tool (SSAS) from the new multidimensional logical models (PSM).

In order to fulfill our MDA architecture, in this paper has been also included support to the transformation of complex security rules which are defined in conceptual models as OCL expressions. To achieve this goal, our UML profile for the conceptual modelling of secure DWs (PIM) has been improved with new metaclasses for managing information about complex security rules which are next used by QVT rules to obtain PSM models. The improvements carried out in the PIM do not affect the MDA architecture and the previously defined rules for the relational approach. Furthermore, an application example has been presented in order to validate our proposal, in which a conceptual model is defined and transformation rules are applied to generate the secure multidimensional logical model and the eventually implementation into SSAS.

Our further work will improve this architecture in several lines: (i) including new PSM models, giving support to other final platforms (such as Pentaho); (ii) defining inverse transformations for allowing modernization processes; and (iii) including dynamic security models which complement the existing models dealing with the inference security problem.

Acknowledgements.

This research is part of the following projects: SERENIDAD (PEII11-037-7035) financed by the Viceconsejera de Ciencia y Tecnologa de la Junta de Comunidades de Castilla-La Mancha (Spain) and FEDER, and SIGMA-CC (TIN2012-36904) and GEODAS (TIN2012-37493-C03-01) financed by the Ministerio de Economa y Competitividad (Spain).

Bibliography

References

[1] W. Inmon, 2.0 - architecture for the next generation of data warehousing, Morgan Kaufman Series in Data Management Systems, Morgan Kaufmann, 2008.

[2] 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.

[3] E. Fernández-Medina, J. Jurjens, J. Trujillo, S. Jajodia, Model-driven development for secure information systems, Information and Software Technology 51 (5) (2009) 809–814, 0950-5849 doi: DOI: 10.1016/j.infsof.2008.05.010.

[4] A. Abraham, J. Lloret Mauri, J. Buford, J. Suzuki, S. Thampi, K. Khajaria, M. Kumar, Evaluation of Approaches for Modeling of Security in Data Warehouses, Vol. 191, Springer Berlin Heidelberg, 2011, pp. 9–18.

[5] S. Fischer-Hübner, S. Katsikas, G. Quirchmayr, A. Salem, S. Triki, H. Ben-Abdallah, N. Harbi, O. Boussaid, Verification of Security Coherence in Data Warehouse Designs, Vol. 7449, Springer Berlin Heidelberg, 2012, pp. 207–213.

[6] OMG, Mda. model driven architecture guide version 1.0.1, http://www.omg.org/cgi-bin/doc?omg/03-06-01 (2003).

[7] OMG, Qvt. meta object facility 2.0 query/view/transformation specification, http://www.omg.org/spec/QVT/1.1 (2011).

[8] E. Fernández-Medina, J. Trujillo, M. Piattini, Model driven multidimensional modeling of secure data warehouses, European Journal of Information Systems 16 (4) (2007) 374–389.

[9] J. Jurjens, Secure Systems Development with UML, Springer-Verlag, 2004.

[10] J. Jurjens, H. Schmidt, Umlsec4uml2 - adopting umlsec to support uml2, Tech. rep., Technical Reports in Computer Science. Technische Universitat Dortmund, http://hdl.handle.net/2003/27602 (2011).

[11] 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.

[12] F. J. B. Nunes, A. D. Belchior, A. B. Albuquerque, Security engineering approach to support software security, Services, IEEE Congress on 0 (2010) 48–55. doi:http://doi.ieeecomputersociety.org/10.1109/SERVICES.2010.37.

[13] A. Cuzzocrea, V. Russo, Privacy Preserving OLAP and OLAP Security, IGI Global, Hershey, PA, USA, 2009, pp. 1575–1581. doi:10.4018/978-1-60566-010-3.ch241.

[14] A. Rosenthal, E. Sciore, View security as the basic for data warehouse security, in: 2nd International Workshop on Design and Management of Data Warehouse (DMDW'00), Vol. 28, Sweden, 2000, pp. 8.1–8.8.

[15] F. Saltor, M. Oliva, A. Abelló, J. Samos, Building secure data warehouse schemas from federated information systems (2002).

[16] E. Weippl, O. Mangisengi, W. Essmayr, F. Lichtenberger, W. Winiwarter, An authorization model for data warehouses and olap, in: Workshop on Security in Distributed Data Warehousing, New Orleans, Louisiana, USA, 2001.

[17] T. Priebe, G. Pernul, A pragmatic approach to conceptual modeling of olap security, in: 20th International Conference on Conceptual Modeling (ER 2001), Springer-Verlag, Yokohama, Japan, 2001.

[18] J. Trujillo, E. Soler, E. Fernández-Medina, M. Piattini, A uml 2.0 profile to define security requirements for datawarehouses, Computer Standards and Interfaces (CSI) 31 (5) (2009) 969–983.

[19] E. Yu, Towards modelling and reasoning support for early-phase requirements engineering, in: 3rd IEEE International Symposium on Requirements Engineering (RE'97), Washington, DC, 1997, pp. 226–235.

[20] E. Fernández-Medina, J. Trujillo, R. Villarroel, M. Piattini, Developing secure data warehouses with a uml extension, Information Systems 32 (6) (2007) 826–856.

[21] E. Fernández-Medina, J. Trujillo, R. Villarroel, M. Piattini, Access control and audit model for the multidimensional modeling of data warehouses, Decision Support Systems 42 (2006) 1270–1289.

[22] J. Trujillo, E. Soler, E. Fernández-Medina, M. Piattini, An engineering process for developing secure data warehouses, Information and Software Technology 51 (6) (2009) 1033–1051.

[23] E. Soler, J. Trujillo, E. Fernández-Medina, M. Piattini, Building a secure star schema in data warehouses by an extension of the relational package from cwm, Computer Standards and Interfaces (CSI) 30 (6) (2008) 341–350.

[24] OMG, Cwm. common warehouse metamodel. version v1.1, http://www.omg.org/spec/CWM/1.1 (2003).

[25] E. Soler, J. Trujillo, C. Blanco, E. Fernández-Medina, Designing secure data warehouses by using mda and qvt, Journal of Universal Computer Science (JUCS) 15 (8) (2009) 1608–1641.

[26] C. Blanco, E. Fernández-Medina, J. Trujillo, M. Piattini, How to implement multidimensional security into olap tools, Int. J. of Business Intelligence and Data Mining - IJBIDM 3 (3) (2008) 255.