Applying QVT in Order to Implement Secure Data Warehouses in SQL Server Analysis Services

Carlos Blanco, Ignacio García-Rodríguez de Guzmán, David G. Rosado and Eduardo Fernández-Medina

Dep. of Information Technologies and Systems, Escuela Superior de Informática
ALARCOS Research Group – Institute of Information Technologies and Systems
University of Castilla-La Mancha, Paseo de la Universidad, 4. 13071, Ciudad Real, Spain
{Carlos.Blanco, Ignacio.GRodriguez, David.GRosado, Eduardo.Fdezmedina}@uclm.es

Juan Trujillo

Dep. of Information Languages and Systems, Facultad de Informática
LUCENTIA Research Group, University of Alicante, San Vicente s/n. 03690, Alicante, Spain
jtrujillo@dlsi.ua.es

Data Warehouses manage historical information for the decision making process, and this information could be discovered by unauthorized users if security constraints are not established. It is therefore highly important for OLAP tools to consider the security rules defined at early stages of the development lifecycle. By following the MDA approach we have created an architecture with which to develop secure Data Warehouses, and in this paper we complete this architecture, thus obtaining secure multidimensional code in SQL Server Analysis Services from our secure multidimensional conceptual model (SECDW) by using QVT transformations. We focus on automatically obtaining code for the security constraints defined at upper abstraction levels.

Keywords: Data Warehouses, Security, MDA, OLAP, QVT
ACM Classification: D2.10, H0

1. INTRODUCTION

Multidimensional modelling is the foundation of Data Warehouses (DWs), multidimensional (MD) Databases and On-Line Analytical Processing Applications (OLAP). Data Warehouse systems are used by decision makers to analyze the status and the development of an organization (Kimball, 2002), based on large amounts of data integrated from heterogeneous sources into a multidimensional (MD) model.

Information security is, furthermore, a serious requirement that must be carefully taken into account, not as an isolated aspect, but as an element present in all stages of the development lifecycle: from requirement analysis to implementation and maintenance (Devanbu and Stubblebine, 2000; Mouratidis and Giorgini, 2006). Information assurance, security and privacy have thus moved from being considered by information systems designers as narrow topics of interest to becoming critical issues of fundamental importance in our society (Denker *et al*, 2005). Some authors indicate that the survival of organizations depends on the correct management of information security and confidentiality (Dhillon, 2001). Data Warehouses use enterprise

Manuscript received: 23 April 2008
Communicating Editor: Eduardo Fernandez-Medina Paton

information for the decision making process and a user may be able to discover extremely important information by using queries in OLAP tools. It is therefore necessary for the security measures defined in early stages of the development process to be applied in OLAP.

In addition, OMG Model Driven Architecture, MDA (OMG, 2003) is a standard for model-driven approaches for software development that is based on the separation between the specification of the system functionality and its implementation using specific platforms. MDA defines a Platform-Independent Model (PIM) which does not include information about specific platforms and technologies. This model (PIM) can be translated into: (1) one or more Platform-Specific Models (PSM) with information about the specific technology used; or (2) other PIMs with a different level of abstraction. Each PSM can then be translated into a code that can be executed in the specific platform. Several proposals for defining these translations between models exist (Czarnecki and Helsen, 2003), and OMG proposes the use of Query/Views/Transformations, QVT (OMG, 2005) to define transformations between models created by using Meta-Object Facility (MOF, 2006).

In Fernández-Medina *et al* (2007a) a proposal for modelling secure Data Warehouses using a MDA approach is presented, which will be described in the following section. This proposal does not deal with the final implementation in OLAP tools, but in Blanco *et al* (2008) it is discussed how security measures defined by using this approach could eventually be implemented in OLAP tools. In this paper, the authors focus on automatically obtaining secure multidimensional code for SQL Server Analysis Services by following the MDA approach.

The remainder of the paper is organized as follows: Section 2 will present related work; Section 3 will describe our MDA approach for developing secure DWs; Section 4 will present our QVT transformations with which to obtain secure multidimensional code; Section 5 will show an example of the application of the defined transformations and Section 6 will present our conclusions and future work. Code for the proposed rules will be presented in greater detail in the Appendices.

2. RELATED WORK

OLAP systems are mechanisms with which to discover business information and use a multidimensional analysis of data to make strategic decisions. This information is organized according to the business parameters, and users may be able to discover unauthorized data by applying a set of OLAP operations to the multidimensional view. Therefore, it is of vital importance for organizations to protect their data from unauthorized accesses. Several works attempting to include security issues in OLAP tools by implementing the previously defined security rules at a conceptual level have been proposed, but these works focus solely upon the Discretional Access Control (DAC) policy and use a simplified role concept implemented as a subject. For instance, Katic *et al* (1998) proposed a DW security model based on metamodels which provides us with views for each user group and uses DAC with classification and access rules for security objects and subjects. However, this model does not allow us to define complex confidentiality constraints. Kirkgoze *et al* (1997) defined a role-based security concept for OLAP by using a "constraints list" for each role, and this concept is implemented through the use of a discretional system in which roles are defined as subjects.

Priebe and Pernul (2001) later proposed a security design methodology, analyzed security requirements, classifying them into basic and advanced, and dealt with their implementation in commercial tools. In Priebe and Pernul (2001) the same authors used ADAPTed UML to define a DAC system with roles defined as subjects at a conceptual level. They then went on to implement this in SQL Server Analysis Services 2000 by using Multidimensional Expressions (MDX). They

created a Multidimensional Security Constraint Language (MDSCL) based on MDX and put forward HIDE statements with which to represent negative authorization constraints on certain multidimensional elements: cube, measure, slice and level.

On the other hand, Model Driven Development when applied to the development of secure information systems permits us to build robust secure information systems in which security is not improvised and incorporated once the system has been completely built. In this context, security models are embedded in and scattered throughout the high level system models, meaning that these integrated models can be transformed into implementation models, according to the MDA strategy.

One of the first and most relevant proposals that integrates security into information systems through the use of UML is UMLsec (Jürjens, 2002; Jürjens, 2004), which can be employed to specify and evaluate UML security specifications using formal semantics. The term Model Driven Security, MDS (Basin *et al*, 2006) was conceived as a new approach towards building secure information systems, in which designers specify high-level system models along with their security properties and use tools to automatically generate system architectures from the models, including security infrastructures. This proposal extends MDA in three aspects: i) the system models are enriched with primitives and rules in order to integrate security into the development process, ii) the model transformation techniques are extended to ensure that these security details are also transformed, and iii) the system is obtained, including the security properties and the corresponding security mechanisms. In order to fulfill this goal, the authors consider dialects which provide a bridge by defining the connection points with which to integrate elements of the security modelling language with elements of the system design modeling language. Within the context of MDS, the same authors propose SecureUML, an extension of UML for modelling a generalized role based access control (Lodderstedt *et al*, 2002).

Our proposal uses an access control and audit model specifically designed for DWs to define security constraints in early stages of the development lifecycle. By using an MDA approach we consider security issues in all stages of the development process and automatically transform models at the upper abstraction level into logical models over a relational or multidimensional approach and finally obtain secure code for DBMS or OLAP tools from these models.

3. MODEL DRIVEN ARCHITECTURE FOR DEVELOPING SECURE DWS

In this section our Model Driven Architecture for developing secure Data Warehouses is presented. We focus on the description of the source (Secure Multidimensional PIM) and target (SSAS Metamodel) models used by the QVT transformations proposed in this work. Figure 1 illustrates our MDA architecture for developing secure Data Warehouses (Fernández-Medina *et al*, 2007a). In this architecture security constraints specified at upper abstraction levels are translated into conceptual, logical and code levels.

At the business level we define both functional and non functional requirements for DWs by using a UML profile for i* called Secure Multidimensional CIM (SMD CIM) (Soler *et al*, 2008a). By using a QVT transformation it is possible to obtain a Secure Multidimensional PIM(SMD PIM) at the conceptual level, represented with our Secure Data Warehouse (SECDW) metamodel (Fernández-Medina *et al*, 2007b), which is a UML profile for DWs enriched with an Access Control and Audit model (Fernández-Medina *et al*, 2006). This conceptual model will be the source model for the proposed QVT transformations and will later be explained in more detail.

The specification of a platform-specific model (PSM) is designed according to the specific properties of the Database Management Systems (DBMS) such as Relational Online Analytical Processing (ROLAP), Multidimensional Online Analytical Processing (MOLAP) or Hybrid Online

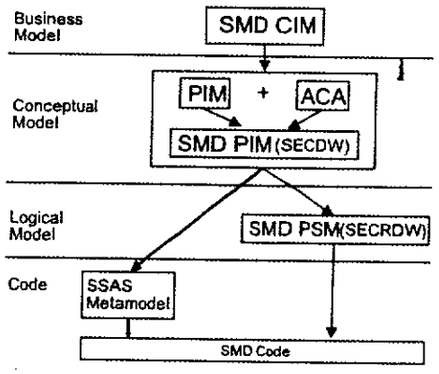

Figure 1: MDA approach for developing secure DWs

Analytical Processing (HOLAP). We have defined a Secure Multidimensional PSM (SMD PSM) at the logical level, which is a ROLAP approach called Secure Relational Data Warehouse (SECRDW) (Soler *et al*, 2008b). This metamodel extends the Relational Warehouse Metamodel (CWM, 2003) with security and audit capabilities and allows us to model STables, SColumn, PrimaryKey, ForeingKey, etc. The SecurityProperty and SecurityConstraint metaclasses are associated with the Table and Column metaclasses to allow us to define security at table and attribute levels. SecurityConstraints, moreover, permit us to express the constraints (SecurityRule, AuthoizationRule and AuditRule) that are defined in the SECDW metamodel with UML notes.

At code level we obtain metamodels with secure multidimensional code in the target platform that can be easily translated into final code. In this work we define several metamodels with which to represent constraints over a multidimensional approach (MOLAP) in SQL Server Analysis Services. These metamodels will be the target models for the proposed QVT transformation and will be presented in more detail in the following sections.

In order to complete the MDA architecture, it is necessary to define the transformation between models. The transformation between conceptual and logical levels using SMD PIM (SECDW) and SMD PSM (SECRDW) has already been defined (Soler *et al*, 2007). The transformation from our relational metamodel at the logical level (SECRDW) to secure code in a DBMS using Oracle Label Security has also been defined.

Due to the fact that OLAP tools are more frequently used in Data Warehouses than DBMS, our research effort is focused on developing multidimensional secure code in OLAP tools according to the above-defined security requirements at conceptual and logical levels. In this paper, we follow the methodology defined in Blanco et al (2008) to translate security constraints defined at the conceptual level (SECDW) into secure multidimensional code in SQL Server Analysis Services that can be automatically translated into final code.

3.1 Secure Multidimensional PIM (SECDW)

Our Secure Multidimensional PIM, called Secure Data Warehouse (SECDW) (Fernández-Medina *et al*, 2007b), allows us to represent the main security requirements for DWs at the conceptual level and is composed of a UML profile for DWs (Luján-Mora *et al*, 2006) enriched with an Access Control and Audit (ACA) model (Fernández-Medina *et al*, 2006).

Traditional access control models are based on relational concepts (tables, columns, rows, etc) and they are not appropriate for the multidimensional modeling used in Data Warehouses. The ACA model (Fernández-Medina *et al*, 2006) is an access control and audit model defined for DWs that allows us to specify security constraints in DW's multidimensional models. This model considers a combination of mandatory and role based access control which is based on the classification of subjects and objects in the system. ACA defines three means of classification: security levels which indicate the user's clearance level; security roles which are used by a company to organize users into a hierarchical role structure according to the responsibilities of each type of work (each user can play more than one role); and security compartments which are used by an organization to classify users into a set of horizontal compartments or groups. In the ACA model an authorization subject is an identity composed of: userID (to identify the user), roleID (one or more user roles), compartmentID (one or more user compartments), securityLevel (a security level or a level interval) and subjectExpression (an OCL expression concerning user profiles).

According to MD models, we can identify the main authorization objects as follows: facts, dimension, classification, hierarchy levels, measures, dimension attributes and instances. The authorization object component is composed of two parts: an identity (which may be one of the following subattributes: class name or attribute name) and an objectExpression in OCL.

Our ACA model also allows us to define several kinds of security rules: Sensitive information assignment rules (SIAR) which specify multilevel security policies and allow us to define sensitivity information for each element in the MD model; Authorization rules (AUR) which specify the subject to which the rule applies, the object to which the authorization refers, the action to which the rule refers and the sign describing whether the rule permits or denies access; and Auditing rules (AR) which help us to ensure that authorized users do not misuse their privileges.

The SMD PIM metamodel (SECDW) is shown in Figure 2 and includes the main characteristics of Data Warehouses such as many-to-many relations, degenerated dimensions, multiple classifications and the alternative path of hierarchies. We have improved this metamodel with several classes which allow us to represent our security classification in roles, levels and compartments in order to use them to carry out our transformations (i.e. for each security level we need to know what the upper and inferior levels are).

Security aspects can be defined according to our Access Control and Audit model. We can define security levels (SecurityLevels), user categories (SecurityCompartment), user roles (SecurityRoles) and security constraints (SConstraints) for each element of the metamodel: SecureFact, SecureDegenerateFact, SecureDimension, SecureBase, SecureDegenerateDimension, SecureFactAttribute, SecureDescriptor, SecureOID and SecureDimensionAttribute. Moreover, there is a UserProfile metaclass containing information about each user's right of access to the multidimensional model. According to our ACA model we can define security rules (SIAR), authorization rules (AUR) and audit rules (AU) by using OCL expressions and UML notes associated with the corresponding class.

3.2 Secure Multidimensional Code (SSAS Metamodel)

The implementation of security measures into OLAP tools has been carried out with SQL Server Analysis Services (SSAS), which was selected because it works with multidimensional models and allows us to define security measures over multidimensional elements (cube, dimension, cell). However, SSAS uses a role-based access control policy (RBAC) that is supposed to translate the measures defined according to our ACA model (with security roles, levels and compartments) into role approach.

We have first analysed source code in SSAS and have defined the necessary metamodels which represent multidimensional secure code with DW's structure and security measures in a step

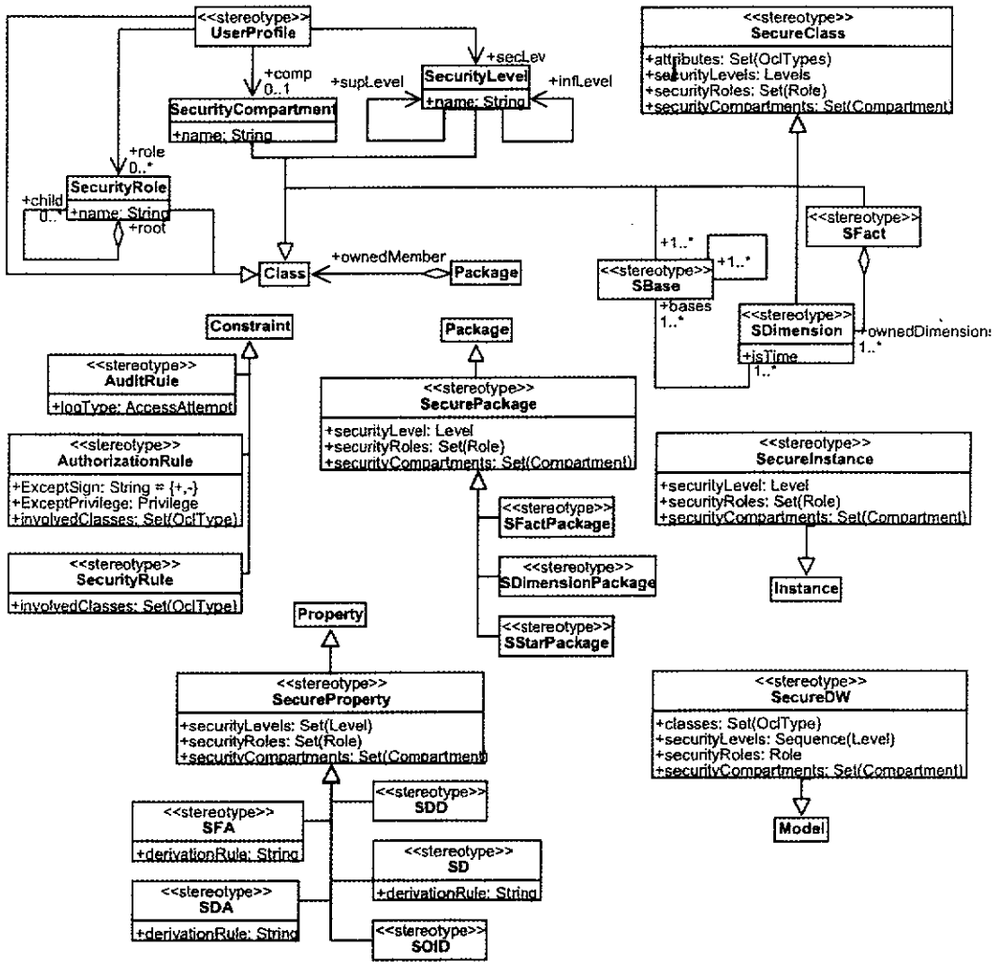

Figure 2: Secure Multidimensional PIM (SECDW)

previous to final code which can be automatically obtained from these metamodels. SSAS defines a DW by using several XML source files. To obtain our SSAS metamodels we focus on roles configuration, DW structure definition and how security constraints are specified.

Figure 3 shows the role metamodel used to define roles in SSAS. We will use this metamodel to define roles in SSAS for each security role, level and compartment defined at the conceptual level. Figure 4 shows the cube metamodel used to represent a cube in SSAS. We can define structural aspects for a cube as dimensions, attributes, hierarchies or measures, and security constraints by using permissions over cubes, dimensions or cells. Finally, Figure 5 shows the dimension metamodel used to define dimensions and bases in SSAS by using Attributes and Hierarchies. Security constraints can be established at dimension and cell levels and we can use complex MDX queries as allowed or denied sets to define advanced security constraints.

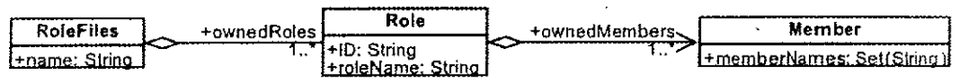

Figure3: SSAS Metamodel: Role configuration

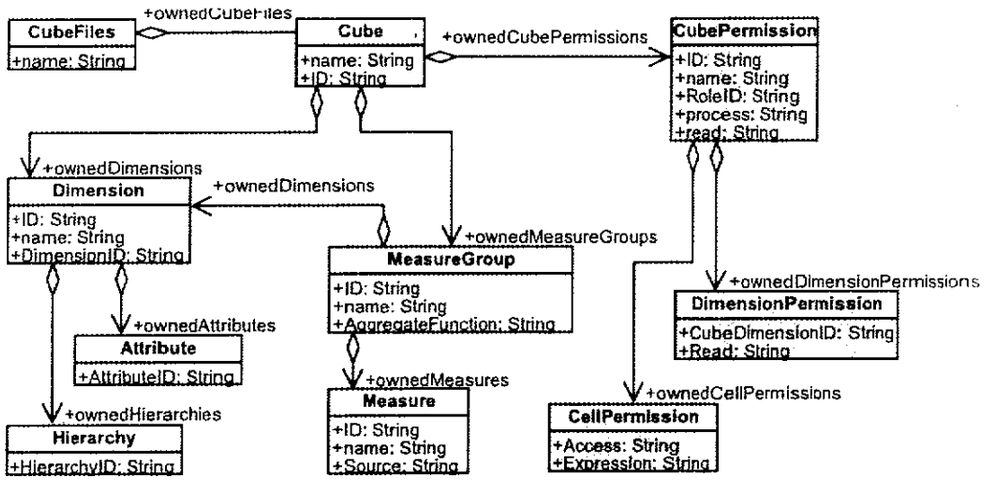

Figure 4: SSAS Metamodel: Cube

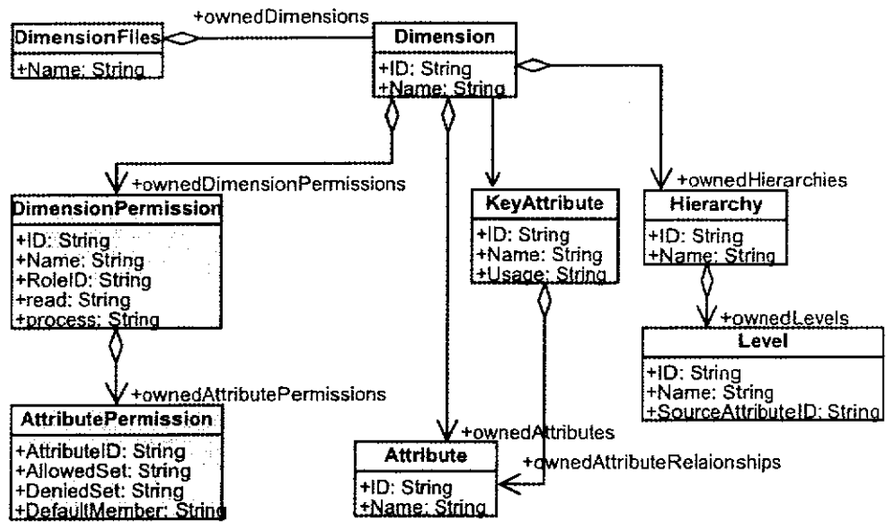

Figure 5: SSAS Metamodel: Dimension

4. TRANSFORMATIONS

In order to obtain secure multidimensional code we follow the methodology previously defined in Blanco *et al* (2008). According to our ACA model we are dealing with levels, compartments and roles but in SSAS we use a role-based policy and in a first step we have to translate this security information by creating new roles for each possible classification.

SSAS uses an open policy with specific denials and it is necessary to define what multidimensional elements are hidden for certain roles. It is therefore necessary to analyze the security rules defined at the conceptual level, to detect the roles involved and to hide certain DW elements from them. If we wish to hide these elements then we must consider the concepts of security role, level and compartment. When access is denied to a certain security role this access must also be denied to its descendants, and when it is denied to a certain security level then access must be denied to each role that represents a lower security level.

This paper does not include the structural transformation and is focused on obtaining the secure multidimensional code corresponding to the security rules defined in our SECDW metamodel at the conceptual level (see Figure 6). The analysis and the automatic transformation of advanced security rules (SIAR and AUR) defined in UML notes with OCL expressions will be dealt with in future works. To define Audit rules in SSAS, administrators can directly establish audit activity on data including information about when data has been read and modified. These defined rules are presented in greater detail in the Appendices and are composed of three main transformations (see Figure 6): SECDW2Role, SECDW2Cube and SECDW2Dimension. SECDW2Role is in charge of generating the set of models representing the "role files". After executing SECDW2Role the models produced contain enough information to easily produce the XML Role files. SECDW2Cube obtains Cube files from the SECDW model. The Cube files are also a very important part in the DW description. Thus, the SECDW2Cube transformation tackles the generation of the models representing these files. This transformation deals, on one hand, with the generation of structure of the DW, and on the other hand with the inclusion of the aforementioned security issues. Finally, SECDW2Dimensions is in charge of producing the set of models representing the dimensions of the DW. Owing to the extent of the transformation, only the security issues have been taken into account in this paper.

Although the aforementioned transformations are included in Appendix A, some rules from these transformations are shown in this section. The following relations are shown in a graphical manner for a better understanding:

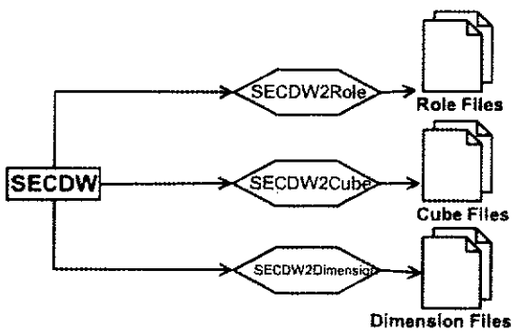

Figure 6: General view of the proposed transformations

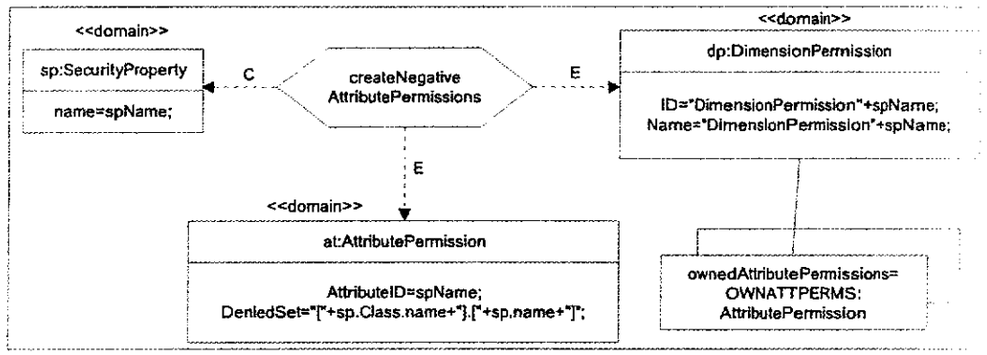

Figure 7: Graphical representation of the SRole2Role relation (SECDW2Role transformation)

- SRole2Role relation (see Figure 7): In this relation, each SRole class from the source model is transformed into a Role class in the target model (that is to say, the Security Configuration Model).
- createNegativeAttributePermissions relation (see Figure 8): This relation creates the corresponding negative permissions for all the corresponding levels and roles belonging to each SecurityProperty. This relation is invoked from the SecureProperty relation process (see Appendix A.3)
- SRole2CubePermission relation (see Figure 9): For each SRole in each SFact class from the source model this transformation creates a CubePermission class in the Cube Model. This relation assumes that the Cube class has been created in the previous execution of another relation. In this relation the Cube class is updated with the new CubePermission.

5. EXAMPLE

In this section, the defined transformations are applied by using an example of a hospital that wishes to automate its patient admission process. Figure 10 shows a part of the conceptual model used in this example which has been defined as an instance of our SECDW metamodel. In this example, users and objects are classified into security levels (SL) and security roles (SR). The classification

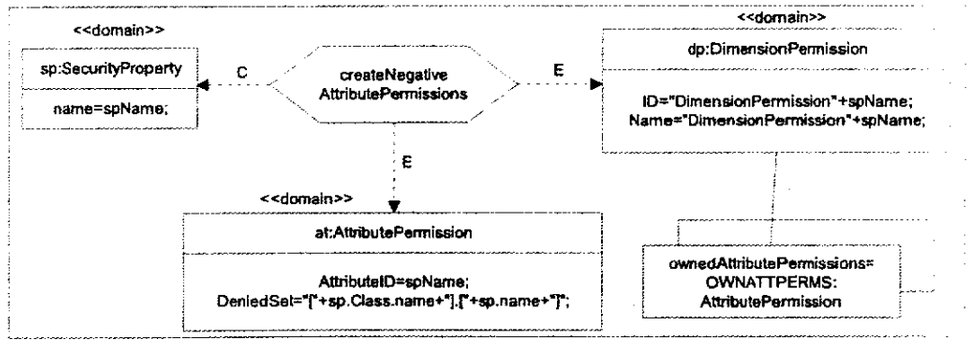

Figure 8: Graphical representation of the createNegativeAttributePermissions relation (SECDW2Dimension transformation)

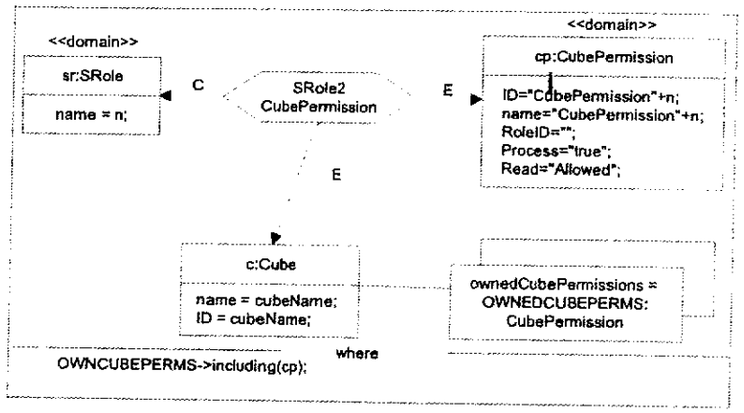

Figure 9: Graphical representation of the SRole2CubePermission relation (SECDW2Cube transformation)

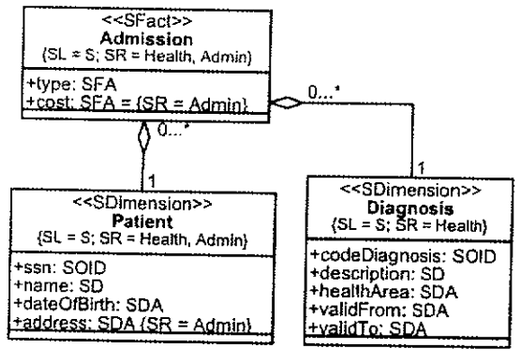

Figure 10: Example SMD PIM

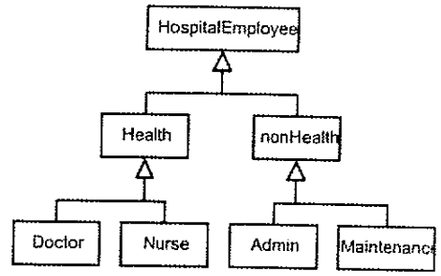

Figure 11: Roles hierarchy

within the levels of security is Top Secret (TS), Secret (S), Confidential (C) and Undefined (U), and the hierarchy of roles can be seen in Figure11.

Several sensitivity information assignment rules (SIARs) have also been defined at class and attribute levels (see Table 1).

SIAR 1
OBJECTS MDCL Admission Patient SECINF SL Secret SR Health Admin
Tagged values SL and SR of the fact class *Admission* and the dimension class *Patient*
SIAR 2
OBJECTS MDCL Diagnosis SECINF SL Secret SR Health
Tagged values SL and SR of the dimension class *Diagnosis*
SIAR 3
OBJECTS MDCL Admission.cost Patient.address SECINF SR Admin
Tagged value SR of the attribute cost in the fact class *Admission* and the attribute *address* in the dimension class *Patient*

Table 1: SIAR defined on the example

The XML files which resulted from applying the proposed transformation to the example are shown in the following sections. We usually obtain the same code fragments (for each dimension, attribute, etc) but, due to space constraints, we have only included the first repetitive code fragment and we have indicated that this fragment must be repeated for certain elements (each security role, level, dimension, attribute, etc).

5.1 Security configuration

Firstly, security configuration is obtained for the SECDW2Role (see Appendix A.1) transformation creating a new role (a new XML file with ".role" extension) for each security role, level and compartment defined in the SECDW model. This is the "SRHospitalEmployee.role" file generated for security role "HospitalEmployee":

```
<Role>
        <ID>SRHospitalEmployee</ID>
        <Name>SRHospitalEmployee</Name>
        <Members>
                <Member><Name>Employee1</Name></Member>
                ...
        </Members>
</Role>
```

5.2 Cubes

Next, cubes in the SECDW model are processed by the SECDW2Cube (see Appendix A.2) transformation by creating an "Admission.cube" file. Structural issues such as cubes, related dimensions and their attributes, and measures, are defined.

```
<Cube>
        <ID>Admission</ID>
        <Name>Admission</Name>
        <Dimensions>
                <Dimension>
                        <ID>Diagnosis</ID>
                        <Name>Diagnosis</Name>
                        <DimensionID>Diagnosis</DimensionID>
                        <Attributes>
                                <Attribute>
                                        <AttributeID>codeDiagnosis</AttributeID>
                                </Attribute> ...for each attribute (codeDiagnosis, description, healthArea, validFrom and
                                validTo)
                        </Attributes>
                </Dimension> ...for each dimension (Diagnosis and Patient)
        </Dimensions>

        <MeasureGroups>
                <MeasureGroup>
                        <ID>Admission</ID>
                        <Name>Admission</Name>
                        <Measures>
                                <Measure>
                                        <ID>Cost</ID>
                                        <Name>Cost</Name>
                                        <Source></Source>
                                </Measure> ...for each FAC attribute in SFact Admission (cost and type)
                        </Measures>
                </MeasureGroup>
        </MeasureGroups>
</Cube>
```

We shall now show the results obtained by applying transformations which deal with the security measures defined in the cubes. SIAR 1 defines security constraints over the "Admission" fact class, and a code fragment with a cube permission is generated for each authorized role. A cube permission for an authorized role (SLS) which represents a security level of "Secret" it is shown below.

```
<CubePermissions>
        <CubePermission>
                <ID>CubePermissionSLS</ID>
                <Name>CubePermissionSLS</Name>
                <RoleID>SLS</RoleID>
                <Process>true</Process>
                <Read>Allowed</Read>
                <DimensionPermissions>
                        <DimensionPermission>
                                <CubeDimensionID>Measures</CubeDimensionID>
                                <Read>Allowed</Read>
                        </DimensionPermission>
                </DimensionPermissions>
                <CellPermissions>
                        <CellPermission>
                                <Access>Read</Access>
                                <Expression>[Measures].[type]</Expression>
                        </CellPermission>
                </CellPermissions>
        </CubePermission>
</CubePermissions>
```

Cube permissions are also defined for the remaining authorized roles: for security levels >= "Secret" (SLS and SLTS), security roles = "Health" and its descendants (SRHealth, SRDoctor and SRNurse) and security roles = "Admin" and its descendants (SRAdmin).

SIAR 3 defines that the attribute "cost" can only be accessed by security role "Admin".

We include a cell permission for the security role "Admin" and its descendants:

```
<CellPermissions>
        <CellPermission>
                <Access>Read</Access>
                <Expression>[Measures].[type] & [Measures].[cost]</Expression>
        </CellPermission>
</CellPermissions>
```

5.3 Dimensions

Finally, classes which define dimensions and bases in SECDW model are processed by the SECDW2Dimension (see Appendix A.3) transformation to obtain their structural and security issues. An XML file with "dim" extension is created for each dimension. For the "Patient" dimension class we obtain:

```
<Dimension>
        <ID>Patient</ID>
        <Name>Patient</Name>
        <Attributes>
                <Attribute>
                        <ID>Patient</ID>
                        <Name>Patient</Name>
                        <Usage>Key</Usage>
                        <KeyColumns>
                                <KeyColumn>
                                <DataType>Integer</DataType>
                                <Source><ColumnID>ssn</ColumnID></Source>
                                </KeyColumn>
                        </KeyColumns>
                        <AttributeRelationships>
                                <AttributeRelationship>
                                        <AttributeID>name</AttributeID>
                                        <Name>name</Name>
                                </AttributeRelationship> ...for each related attribute (name, dateOfBirth and address)
                        </AttributeRelationships>
                </Attribute> ...for each attribute  (ssn, name, dateOfBirth and address)
        </Attributes>
</Dimension>
```


We shall now show the results obtained from applying transformations which deal with the security measures defined in the cubes. SIAR 1 defines security constraints over "Admission" fact class, and a code fragment with a cube permission is generated for each authorized role. A cube permission for an authorized role (SLS) which represents a security level of "Secret" it is shown below.

SIAR 1 establishes that the "Patient" dimension class can only be accessed by users with a security role of "Health" or "Admin" and a security level of "Secret". A dimension permission is defined for each authorized role, that is, for security levels >= "Secret", and security roles = "Health" or "Admin" and their descendants. This is an example of a dimension permission for the SLS (Security Level Secret) role:

```
<DimensionPermissions>
        <DimensionPermission>
                <ID>DimensionPermissionSLS</ID>
                <Name>DimensionPermissionSLS</Name>
                <RoleID>SLS</RoleID>
                <Process>true</Process>
                <Read>Allowed</Read>
                <AttributePermissions>
                        <AttributePermission>
                                <AttributeID>name</AttributeID>
                        </AttributePermission> ...for each attribute of Patient dimension (name, dateOfBirth and address)
                </AttributePermissions>
        </DimensionPermission>
</DimensionPermissions>
```

A specific denied set is defined for each unauthorized role, that is, roles with SL < S and SR which are distinct from Health, Admin and their descendants.

```
<AttributePermission>
        <AttributeID>name</AttributeID>
        <DeniedSet>[Patient].[name]</DeniedSet>
</AttributePermission> ...for each attribute of Patient dimension (name, dateOfBirth and address)
```

An SIAR 3 security rule has been set up at the attribute level which denies accesses to "address" attribute if the security role is distinct from "Admin". For each role with an SR which is distinct from "Admin" and its descendants:

```
<AttributePermission>
        <AttributeID>address</AttributeID>
        <DeniedSet>[Patient].[address]</DeniedSet>
</AttributePermission>
```

6. CONCLUSIONS

We have accomplished an important step towards completing our MDA architecture for developing secure Data Warehouses with the automatic generation of secure multidimensional code in a specific OLAP tool, SQL Server Analysis Services, by using QVT transformations.

In a previous work we obtained secure code for Oracle Label Security from a relational PSM metamodel, but we used a relational approach in a DBMS. OLAP tools are more frequently used in

Data Warehouses than DBMS, so we have therefore focused our research effort on OLAP tools and multidimensional approaches.

Therefore, in this work we use the multidimensional capabilities of SSAS to translate the security measures defined in our secure multidimensional model at conceptual level into secure multidimensional code for SSAS. We obtain code with the role configuration of our system, the structural definition of the DW and the main part of the security measures defined at upper levels. The final code adds other elements which can be easily obtained from our multidimensional secure code with an automatic transformation.

We furthermore realize that OLAP tools do not support the complete security requirements definition over the multidimensional model at upper abstraction levels and deal with partial security establishment. SSAS only uses RBAC as an access control policy and our security model is richer than the security capabilities that SSAS offers. We have adapted our security information to a role-based approach and we have hidden multidimensional elements for certain roles to represent security rules. Moreover, we cannot represent security constraints on OLAP operations at upper abstraction levels which allow users to restrict access to unauthorized information by using navigations or inferences.

Our MDA architecture for developing secure DWs allows us to define security requirements but should be extended with the possibility of establishing navigation and inference constraints which may be translated into OLAP code. Tools should similarly be extended to give us control over navigations and inferences which may discover unauthorized business information.

As a future work, we will define the remainder of the QVT transformations in order to obtain secure multidimensional code in SSAS from conceptual models. We will extend this work by presenting the structural transformations that were not included due to space constraints. We will analyze the advanced security rules defined with OCL expressions and we will create the corresponding QVT transformations to complete our secure code. Finally, we will also extend this complete approach to obtain secure multidimensional code in other OLAP tools such as Pentaho.

ACKNOWLEDGEMENTS

This research is part of the ESFINGE (TIN2006-15175-C05-05) Project financed by the Spanish Ministry of Education and Science, and of the MISTICO (PBC-06-0082) and QUASIMODO (PAC08-0157-0668) Projects financed by the FEDER and the Regional Science and Technology Ministry of Castilla-La Mancha (Spain).

REFERENCES

BASIN, D., DOSER, J. and LODDERSTEDT, T. (2006): Model driven security: from UML models to access control infrastructures. *ACM Transactions on Software Engineering and Methodology*, 15: 39-91.

BLANCO, C., FERNÁNDEZ-MEDINA, E., TRUJILLO, J. and PIATTINI, M. (2008): Implementing multidimensional security into OLAP tools. *Third International Workshop "Dependability Aspects on Data WArehousing and Mining applications" (DAWAM 2008)* Barcelona, Spain, IEEE Computer Society.

CWM, O.M.G. (2003): Common warehouse metamodel (CWM).

CZARNECKI, K. and HELSEN, S. (2003): Classification of model transformation approaches. *2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*. Anaheim.

DENKER, G., KAGAL, L. and FININ, T. (2005): Security in the semantic web using OWL. *Information Security Technical Report*, 10: 51-58.

DEVANBU, P. and STUBBLEBINE, S. (2000): Software engineering for security: a roadmap. ACM Press. *Future of Software Engineering*, 227-239.

DHILLON, G. (2001): *Information Security Management: Global challenges in the new millennium*, Idea Group Publishing.

FERNÁNDEZ-MEDINA, E., TRUJILLO, J. and PIATTINI, M. (2007a): Model driven multidimensional modelling of secure data warehouses. *European Journal of Information Systems*, 16: 374-389.

FERNÁNDEZ-MEDINA, E., TRUJILLO, J., VILLARROEL, R. and PIATTINI, M. (2006): Access control and audit model for the multidimensional modelling of data warehouses. *Decision Support Systems*, 42: 1270-1289.

FERNÁNDEZ-MEDINA, E., TRUJILLO, J., VILLARROEL, R. and PIATTINI, M. (2007b): Developing secure data warehouses with a UML extension. *Information Systems*, 32: 826-856.

JÜRJENS, J. (2002): UMLsec: Extending UML for secure systems development. In JÉZÉQUEL, J., HUSSMANN, H. and COOK, S. (Eds.) *UML 2002 – The Unified Modeling Language, Model engineering, concepts and tools*. Dresden, Germany, Springer. LNCS 2460.

JÜRJENS, J. (2004): *Secure Systems Development with UML*, Springer-Verlag.

KATIC, N., QUIRCHMAYR, G., SCHIEFER, J., STOLBA, M. and MIN TJOA, A. (1998): A prototype model for data warehouse security based on metadata. *9th International Workshop on Database and Expert Systems Applications (DEXA'98)*. Vienna, Austria., IEEE Computer Society.

KIMBALL, R. (2002): *The Data Warehouse Toolkit 2° Edition*, John Wiley & Sons.

KIRKGÖZE, R., KATIC, N., STOLDA, M. and MIN TJOA, A. (1997): A security concept for OLAP. *8th International Workshop on Database and Expert System Applications (DEXA'97)*. Toulouse, France, IEEE Computer Society.

LODDERSTEDT, T., BASIN, D. and DOSER, J. (2002): SecureUML: A UML-based modelling language for model-driven security. *UML 2002. The Unified Modeling Language. Model Engineering, Languages Concepts, and Tools. 5th International Conference*. Dresden, Germany, Springer.

LUJÁN-MORA, S., TRUJILLO, J. and SONG, I.Y. (2006): A UML profile for multidimensional modeling in data warehouses. *Data & Knowledge Engineering*, 59: 725-769.

MOF, O.M.G. (2006) Meta-Object Facility (MOF) Core Specification v2.0.

MOURATIDIS, H. and GIORGINI, P. (2006): An Introduction. *Integrating Security and Software Engineering: Advances and Future Visions*. Idea Group Publishing.

OMG (2003): Model driven architecture guide Version 1.0.1.

OMG (2005): MOF QVT final adopted specification.

PRIEBE, T. and PERNUL, G. (2001): A pragmatic approach to conceptual modelling of OLAP security. *20th International Conference on Conceptual Modeling (ER 2001)*. Yokohama, Japan, Springer-Verlag.

SOLER, E., STEFANOV, V., MAZÓN, J-N., TRUJILLO, J., FERNÁNDEZ-MEDINA, E. and PIATTINI, M. (2008a): Towards comprehensive requirement analysis for data warehouses: Considering security requirements. *Proceedings of The Third International Conference on Availability, Reliability and Security (ARES)* Barcelona, Spain, IEEE Computer Society.

SOLER, E., TRUJILLO, J., FERNÁNDEZ-MEDINA, E. and PIATTINI, M. (2007): A set of QVT relations to transform PIM to PSM in the design of secure data warehouses. *IEEE International Symposium on Frontiers on Availability, Reliability and Security (FARES 2007)*. Vienna, Austria.

SOLER, E., TRUJILLO, J., FERNÁNDEZ-MEDINA, E. and PIATTINI, M. (2008b): Building a secure star schema in data warehouses by an extension of the relational package from CWM. *Computer Standard and Interfaces*, 30: 341-350.

APPENDICES

APPENDIX – QVT TRANSFORMATIONS: SECDW2ROLE, SECDW2CUBE AND SECDW2DIMENSION

In this Appendix the QVT transformations defined to obtain secure multidimensional code from conceptual models defined according to our SECDW metamodel are presented focusing on security rules. The proposed transformations are divided into several relations, each of them is in charge of transforming elements from the source model into elements of the target model. It is possible to distinguish between two kinds of relations: relations and top relations. Top relations are mandatory and must always be held by both the source and target models. On the opposite, simple relations and only executed when other relations invokes them. Thus, top relation could be considered as a "main" function or method.

APPENDIX A.1: SECDW2ROLE – GENERATION OF ROLE SECURITY ISSUES FROM DW SECURE SPECIFICATION

In this section, the transformation SECDW2Role is presented. This transformation is in charge of transforming all the aspects related to the roles included in the SECDW model (see Figure 2). Following the relations that made up the SECDW2Role transformation, an instance of the Role metamodel (see Figure 3) is created.

```
transformation SECDW2Role(SECDW pim, Role psm) {

top relation Package2RoleFiles {
  checkonly domain psm p:Package{
    name = n;
    ownedMember = OWNMEMB:Set(PackageableElement);  }
  enforce domain pim rf:RoleFiles{
    name = n;
    ownedRoles = OWNROLES:Set(Role);  }
  where{
    OWNMEMB->forAll(sr:SRole | OWNROLS->including
      (SRole2Role(sr)));
    OWNMEMB->forAll(sc:SCompartment |
      OWNROLS->including(SCompartment2Role(sc)));
    OWNMEMB->forAll(sl:SLevel | OWNROLS->including
      (SLevel2Role(sl)));  }
}
```

```
relation SCompartment2Role {
  checkonly domain psm sc:SCompartment{
    name = n;  }
  enforce domain pim r:Role{
    fileName = "SC"+n+".role";
    ID = "SC"+n;
    roleName = "SC"+n;
    ownedMembers = OWNEDMEMBS:Set(Member);}}

relation SRole2Role {
  checkonly domain psm sr:SRole{
    name = n;  }
  enforce domain pim r:Role{
    fileName = "SR"+n+".role";
    ID = "SR"+n;
    roleName = "SR"+n;
    ownedMembers = OWNMEMBS:Set(Member);}}

relation SLevel2Role {
  checkonly domain psm sl:SRole{
    name = n;  }
  enforce domain pim r:Role{
    fileName = "SL"+n+".role";
    ID = "SL"+n;
    roleName = "SL"+n;
    ownedMembers = OWNEDMEMBS:Set(Member);}}
```

APPENDIX A.2: SECDW2CUBE – GENERATION OF CUBE SECURITY ISSUES FROM DW SECURE SPECIFICATION

Next, the transformation SECDW2Cube is presented. This transformation implements all the rules read the SECDW model (see Figure 2) and generate all the required security issues related with DW Cubes (that is to say, CubePermissions and CellPermissions from the elements of the Fact Class of the SECDW model). Figure 4 shows the target metamodel.

```
transformation SECDW2Cube (SECDW psm, Cube pim) {
top relation Package2CubeFiles {
  checkonly domain psm p:Package{
    name = n;
    ownedMember = OWNMEMB:Set(PackageableElement); }
  enforce domain pim d:DataWareHouse{
    name = n;
    ownedCubes = OWNCUBS:Set(Cube);  }
  where{
    OWNMEMB->forAll(sf:SFact |
    OWNCUBS->including(SFact2Cube(sf)));  }}

relation SFact2Cube {
  checkonly domain psm sf:SFact{
    name = n;
    securityLevels= SECLEVS:Set(Level);
    securityRoles = SECROL:Set(Role);
    securityCompartments = SECCOMPS:Set(Compartment);
    ownedAttribute = OWNATTR:Set(Property); }
  enforce domain pim c:Cube{
    name = n;
    ID = n;
    ownedCubePermissions =
OWNCUBEPERMS:Set(CubePermission); }
  where{
    securityCompartments->forAll(sc:SCompartment |
      SCompartmentClass2CubePermission(sc,c));
    securityRoles->forAll(sr:SRole |
      functionGetRoleChilds(sl)->forAll(srChild:SRole|
      SRoleClass2CubePermission(sr,c));
    securityLevels->forAll(sl:SLevel |
      functionGetUpperLevels(sl)->forAll(slUpper:SLevel|
      SLevelClass2CubePermission(sl,c));  }}

relation CreateMeasureGroups {...}
relation SProperty2Measure {...}
relation SDimension2Dimension {...}
relation ProcessSBase {...}
relation CreateOwnedHierarchies {...}
relation SProperty2Attribute {...}

relation SCompartmentClass2CubePermission {
  checkonly domain psm sc:SCompartment{
    name = n;  }
  enforce domain pim c:Cube{
    name = cubeName;
```



```
    ID = cubeName;
    ownedCubePermissions =
OWNCUBEPERMS:Set(CubePermission);  }
  enforce domain pim cp:CubePermission{
    ID = "CubePermission"+n;
    name = "CubePermission"+n;
    RoleID = n;
    Process = "true";
    Read = "Allowed";  }
  where{
    OWNCUBEPERMS->including(cp);  }}

relation SRoleClass2CubePermission {
  checkonly domain psm sr:SRole{
    name = n;  }
  enforce domain pim c:Cube{
    name = cubeName;
    ID = cubeName;
    ownedCubePermissions =
OWNCUBEPERMS:Set(CubePermission);  }
  enforce domain pim cp:CubePermission{
    ID = "CubePermission"+n;
    name = "CubePermission"+n;
    RoleID = n;
    Process = "true";
    Read = "Allowed";  }
  where{
    OWNCUBEPERMS->including(cp);  }}

relation SLevelClass2CubePermission {
  checkonly domain psm sl:SLevel{
    name = n;  }
  enforce domain pim c:Cube{
    name = cubeName;
    ID = cubeName;
    ownedCubePermissions =
OWNCUBEPERMS:Set(CubePermission);  }
  enforce domain pim cp:CubePermission{
    ID = "CubePermission"+n;
    name = "CubePermission"+n;
    RoleID = n;
    Process = "true";
    Read = "Allowed";  }
  where{
    OWNCUBEPERMS->including(cp);  }}

relation SecureProperty2CellPermission {
  LEVELS: Set(SLevel);
  ROLES: Set(SRole);
  COMPARTMENTS: Set(SCompartment);
  checkonly domain pim sp:SecureProperty{
    Name = spName,
    class = sf,
    securityLevels = SPSECLEVS:Set(SLevel),
    securityRoles = SPSECROLS:Set(SRole),
    securityCompartments =
SPSECCOMPS:Set(SCompartment) }
  checkonly domain pim sf:SFact{
    Name = sfName,
    securityLevels = SFSECLEVS:Set(SLevel),
    securityRoles = SFSECROLS:Set(SRole),
    securityCompartments =
SFSECCOMPS:Set(SCompartment) }
  where{
.....//CellPermissions from SLevels Attributes
    LEVELS = (if SPSECLEVS->size = 0
                then SFSECLEVS
                else SPSECLEVS)
                endif;
    LEVELS->forAll(sl:SLevel|getUpperLevels(sl)->
forAll(tmpSLevel:SLevel|SLevelAtt2CellPermission
(tmpSLevel,sp))),
.....//CellPermissions from SRoles Attributes
    ROLES = (if SPSECROLS->size = 0
                then SFSECROLS
                else SPSECROLS)
                endif;
    ROLES->forAll(sr:SRole|getRoleChilds(sr)->
forAll(tmpSRole:SRole|SRoleAtt2CellPermission
(tmpSRole,sp)))
.....//CellPermissions from SCompartment Attributes
    COMPARTMENTS = (if SPSECCOMPS->size = 0
                then SFSECCOMPS
                else SPSECCOMPS)
                endif;
 COMPARTMENTS->forAll(sl:SCompartment |
SCompartmentAtt2CellPermission
(SCompartment,sp)))  }}

relation SLevelAtt2CellPermission {
  checkonly domain pim sp:SLevel{
    name = slName }
  checkonly domain pim sl:SecureProperty{
    name = spName  }
  enforce domain psm cellp:CellPermission{
    Access = read,
    Expression = exp }
  enforce domain psm cp:CubePermission{
    ID = id,
    name = "CubePermission"+slName,
    ownedCellPermissions =
OWNEDCELLPERMS:Bag(CellPermission) }
  where{
    If exp.length > 0
    then exp = exp + "&[Measures]" + "[" + spName + "]"
    else exp = "[Measures]" + "[" + spName+ "]"
    endif
    OWNEDCELLPERMS->including(cellp)  }}

relation SRoleAtt2CellPermission {
  checkonly domain pim sl:SRole{
    name = srName  }
  checkonly domain pim sl:SecureProperty{
    name = spName  }
  enforce domain psm cellp:CellPermission{
    Access = read,
```

```
    Expression = exp  }
  enforce domain psm cp:CubePermission{
    ID = id,
    name = "CubePermission"+srName,
    ownedCellPermissions =
OWNEDCELLPERMS:Bag(CellPermission) }
  where{
    If exp.length > 0
    Then exp = exp + "&[Measures]" + "[" + spName + "]"
    Else exp = "[Measures]" + "[" + spName+ "]"
    endif
    OWNEDCELLPERMS->including(cellp)  }}

relation SCompartmentAtt2CellPermission {
  checkonly domain pim sl:SCompartment{
    name = scName }
  checkonly domain pim sl:SecureProperty{
    name = spName  }
  enforce domain psm cellp:CellPermission{
    Access = read,
    Expression = exp }
  enforce domain psm cp:CubePermission{
    ID = id,
    name = "CubePermission"+scName,
    ownedCellPermissions = OWNEDCELLPERMS:Bag
(CellPermission) }
  where{
    If exp.length > 0
    Then exp = exp + "&[Measures]" + "[" + spName + "]"
    Else exp = "[Measures]" + "[" + spName+ "]"
    endif
    OWNEDCELLPERMS->including(cellp)  }}
```

APPENDIX A.3: SECDW2DIMENSION – GENERATION OF DIMENSION SECURITY ISSUES FROM DW SECURE SPECIFICATION

Finally, this section presents the transformation SECDW2Dimension. After executing the previous transformations, this one generates the security information to generate the source code for the third security file (see Figure 6). Using the SECDW metamodel (see Figure 2) as source metamodel, the security information is generated in terms of a model according to the metamodel shown in the Figure 5.

```
transformation SECDW2Dimensions (SECDW pim,
Dimensions psm) {
top relation Package2DimensionFiles {
  checkonly domain pim p:Package{
    name = n;     ownedMembers  }
  enforce domain psm df:DimensionFiles{
    name = n;
    ownedDimensions = OWNDIMS:Set(Dimension);  }
  where{
    OWNMEMB->forAll(sd:SDimension |
SDimension2Dimension(df,sd));  }}

relation SDimension2Dimension {
  enforce domain psm df:DimensionFiles{
    name = n;  }
  checkonly domain pim sd:SDimension{
    name = sdName;
    securityLevels = SECLEVS:Set(SLevel);
    securityRoles = SECROLS:Set(SRole);
    securityCompartments = SECCOMPS:Set
(SCompartment);
    ownedAttribute = OWNATTR:Set(Property);  }
  enforce domain psm d:Dimension{
    ID = sdName;     Name = sdName;  }
  when{  n = sd.owner.name;  }
  where{
    SDimension.owner.owner->forAll(sc:SCompartment |
      createDimensionSlARForSCompartment(sc,d));
      SECCOMPS->forAll(sc:SCompartment |
      authorizeSCompartment(sc,tempSCompartment));
    SDimension.owner.owner->forAll(sr:SRole |
      createDimensionSlARForSRole(sr,d));
      SECROLS->forAll(sr:SRole |
      getLeafSecurityRoles(sr)->forAll(tempSRole:SRole |
      authorizeSRole(d,tempSRole));
    SDimension.owner.owner->forAll(sl:SLevel |
      createDimensionSlARForSLevel(sl,d));
      SECLEVS->forAll(sl:SLevel |
      getUpperSecurityLevels(sl)->
forAll(tempSLevel:SLevel |
      authorizeSLevel(d,tempSLevel));
    OWNATTR->select(oclIsKindOf(SecureProperty))->
  forAll(sp:SecureProperty |
processSecureProperty(sd,d,sp))))}

relation KeyProperty2KeyAttribute {...}
relation NonKeyProperty2Attribute {...}
relation SBase2Attributes {...}

relation createDimensionSlARForSCompartment{
  checkonly domain pim sc:SCompartment{  name =
compartmentName;  }
  enforce domain pim d:Dimension{
    name = dimName;
    ownedDimensionPermissions =
OWNDIMPERMS:Set(DimensionPermission);  }
  enforce domain psm ap:DimensionPermission {
    ID = "DimensionPermission" + compartmentName;
    Name = "DimensionPermission" +
compartmentName;
    RoleID = compartmentName;  }}
```



```
relation createDimensionSlARForSRole {
  checkonly domain pim sr:SRole{  name = roleName;  }
  enforce domain pim d:Dimension{
    name = dimName;
    ownedDimensionPermissions =
OWNDIMPERMS:Set(DimensionPermission);  }
  enforce domain psm ap:DimensionPermission{
    ID = "DimensionPermission" + roleName;
    Name = "DimensionPermission" + roleName;
    RoleID = roleName;  }}

relation createDimensionSlARForSLevel {
  checkonly domain pim Sl:SLevel {
    name = levelName;  }
  enforce domain pim d:Dimension{
    name = dimName;
    ownedDimensionPermissions =
OWNDIMPERMS:Set(DimensionPermission);  }
  enforce domain psm ap:DimensionPermission {
    ID = "DimensionPermission" + levelName;
    Name = "DimensionPermission" + levelName;
    RoleID = levelName;  }}

relation authorizeSCompartment {...}

relation authorizeSRole {
  enforce domain psm d:Dimension{
    name = n;
    ownedDimensionPermissions =
OWNDIMPERMS:Set(DimensionPermission);  }
  checkonly domain pim sr:SRole{
    name = sRolName;  }
  where{
    let authDimPer:DimensionPermission =
OWNDIMPERMS->select
(ID = ("DimensionPermission" + sRolName)) in
    authDimPer.Read = "Allowed";
    authDimPer.Process = "true";  }}

relation authorizeSLevel {
  enforce domain psm d:Dimension{
    name = n;
          ownedDimensionPermissions =
          OWNDIMPERMS:Set(DimensionPermission);  }
  checkonly domain pim sl:SLevel {  name = slevelName;  }
  where{
    let authDimPer:DimensionPermission = OWNDIMPERMS->
      select(ID = ("DimensionPermission"+slevelName)) in
      authDimPer.Read = "Allowed";
      authDimPer.Process = "true";  }}

relation processSecureProperty {
  checkonly domain pim sd:SDimension{
    name = n;  }
  enforce domain psm d:Dimension{
    name = n;
    ownedDimensionPermissions =
OWNDIMPER:Set(DimensionPermission);  }
  checkonly domain pim sp:SecureProperty{
    name = spName;
    securityLevels = SECLEVS:Set(SLevel);
    securityRoles = SECROLS:Set(SRole);
    securityCompartments = SECCOMPS:Set(SCompartment);  }
  where{
    SECLEVS->forAll(sl:SLevel |
      getUpperLevels(sl)->forAll(tmpSLevel:SLevel |

createPositiveAttributePermisions(d.ownedDimensionPermisions
->   select(ID=("DimensionPermission"+tmpSLevel.name)),sp);
      getLowerLevels(sl)-> forAll(tmpSLevel:SLevel |

createNegativeAttributePermisions(d.ownedDimensionPermisions
->  select(ID=("DimensionPermission"+tmpSLevel.name)),sp);
      SECROLS->forAll(sr:SRole |
        getLeafSRoles(sr)->forAll(tmpSRole:SRole |
          createPositiveAttributePermisions
          (d.ownedDimensionPermisions->
          select(ID=("DimensionPermission"+tmpSRole.name)),sp);
        getNonLeafRoles(sr)->forAll(tmpSRole:SRole |

createNegativeAttributePermisions(d.ownedDimensionPermisions
->  select(ID=("DimensionPermission"+tmpSRole.name)),sp);
      SECLEVS->forAll(sc:SCompartment |
      createPositiveAttributePermisions
      (d.ownedDimensionPermisions->
      select(ID=("DimensionPermission"+sl.name)),sp);  }}

relation createPositiveAttributePermisions {
  checkonly domain pim sp:SecureProperty{
    name = spName;  }
checkonly domain name :string{}
  enforce domain psm dp:DimensionPermission{
    ownedAttributePermissions =
OWNATTPERMS:Set(AttributePermission);  }
  enforce domain psm at:AttributePermission{
    AttributeID = spName;  }}

relation createNegativeAttributePermissions {
  checkonly domain pim sp:SecureProperty{
    name = spName;  }
checkonly domain name :string{}
  enforce domain psm dp:DimensionPermission{
    ownedAttributePermissions =
OWNATTPERMS:Set(AttributePermission);  }
  enforce domain psm at:AttributePermission{
    AttributeID = spName;
    DeniedSet = "["+sp.class.name+"].["+sp.name+"]"; }}
```

BIOGRAPHICAL NOTES


Carlos Blanco

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 Alarcos Research Group at the School of Computer Science at the University of Castilla-La Mancha (Spain), and his research activity is in the field of security in Data Warehouses, MDA, Information Systems and Ontologies.


Ignacio García Rodríguez de Guzmán

Ignacio García Rodríguez de Guzmán has an MSc in computer science and a PhD from the University of Castilla-La Mancha (UCLM). He is currently a member of the Alarcos Research Group in the School of Computer Science of the UCLM and his research activity is in the field of MDA, Architecture-Driven Modernization, Reverse Engineering and Reengineering, SOA and Web Services.


Eduardo Fernández-Medina

Eduardo Fernández-Medina holds a PhD in computer science from 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 coeditor of several books and chapter books 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 (Decision Support Systems, ACM Sigmod Record, Information Systems, Information Software Technology, Computers and Security, Information Systems Security, etc.) and belongs to various professional and research associations (AEC, ISO, IFIP WG11.3, etc.).


Juan Trujillo

Juan Trujillo received a PhD in computer science from the University of Alicante (Spain) in 2001. His research interests include database modelling, the conceptual design of data warehouses, multidimensional databases, OLAP, as well as OO analysis and design with UML. With papers published in international conferences and journals such as ER, UML, ADBIS, CAiSE,WAIM, Journal of DatabaseManagement (JDM), IEEE Computer, DSS or IS. Trujillo has served as a Program Committee member of several workshops and conferences such as ER, DOLAP, DSS, and SCI and has also spent some time as a reviewer for several journals such as JDM, DKE, KAIS, ISOFT and JODS. He has also been Program Chair of DOLAP'05 and BP-UML'05, and Program Co-chair of DAWAK'05, DAWAK'06 and FP-UML'06-'07.


David G. Rosado

David G. Rosado has an MSc in Computer Science from the University of Málaga (Spain) and currently he is a PhD Student at the University of Castilla-La Mancha. His research activities are focused on security architectures for Information Systems and Mobile Grid Computing. He has published several papers in national and international conferences on these subjects. He is a member of the AL ARCOS research group of the Information Systems and Technologies Department at the University of Castilla-La Mancha, in Ciudad Real, Spain.