Journal of Universal Computer Science, vol. 15, no. 8 (2009), 1607-1641
submitted: 25/7/08, accepted: 24/4/09, appeared: 28/4/09 © J.UCS

Designing Secure Data Warehouses by Using MDA and QVT

Emilio Soler
(Department of Computer Science, University of Matanzas, Autopista de Varadero km 3. Matanzas, Cuba
emilio.soler@umcc.cu)

Juan Trujillo
(LUCENTIA Research Group, Department of Software and Computing Systems, University of Alicante C/ San Vicente S/N 03690 Alicante, Spain
jtrujillo@dlsi.ua.es)

Carlos Blanco
(ALARCOS Research Group, Information Systems and Technologies Department, UCLM-Soluziona Research and Development Institute, University of Castilla-La Mancha, Paseo de la Universidad 4, 13071 Ciudad Real, Spain
carlos.blanco@uclm.es)

Eduardo Fernández-Medina
(ALARCOS Research Group, Information Systems and Technologies Department, UCLM-Soluziona Research and Development Institute, University of Castilla-La Mancha, Paseo de la Universidad 4, 13071 Ciudad Real, Spain
eduardo.fdzmedina@uclm.es)

Abstract: The Data Warehouse (DW) design is based on multidimensional (MD) modeling which structures information into facts and dimensions. Due to the confidentiality of the data that it stores, it is crucial to specify security and audit measures from the early stages of design and to enforce them throughout the lifecycle. Moreover, the standard framework for software development, Model Driven Architecture (MDA), allows us to define transformations between models by proposing Query/View/Transformations (QVT). This proposal permits the definition of formal, elegant and unequivocal transformations between Platform Independent Models (PIM) and Platform Specific Models (PSM). This paper introduces a new framework for the design of secure DWs based on MDA and QVT, which covers all the design phases (conceptual, logical and physical) and specifies security measures in all of them. We first define two metamodels with which to represent security and audit measures at the conceptual and logical levels. We then go on to define a transformation between these models through which to obtain the traceability of the security rules from the early stages of development to the final implementation. Finally, in order to show the benefits of our proposal, it is applied to a case study.

Key Words: Data Warehousing, Security, MDA, QVT
Category: D.2.10, H.0

1 Introduction

The widespread development and application of digital technology, and its exponential take-up which began in the mid-1980s, have changed our way of life [Grabosky (2007)]. Software is the biggest problem in computer security today [McGraw (1999)]. Organizations have begun to adopt more and more computerized information systems, which rely upon databases and DWs that require increasingly more quality and security. DWs frequently store historical and aggregated information, extracted from multiple heterogeneous, autonomous and distributed information sources; therefore, the very survival of the organization depends on the appropriate manipulation, security and confidentiality of this information [Dhillon and Backhouse (2000)].

In recent years DWs have received attention from both industry and the academic world, but within DW projects, the security aspects are normally implemented in the final phases of design. However, information security is a serious requirement which must be carefully considered, not as an isolated aspect, but as an element which appears as an issue in all stages of the development lifecycle, from requirements analysis to implementation and maintenance [Devanbu and Stubblebine (2000)]. In present-day real world Data Warehouse projects, the security rules are specified once the Data warehouse has been implemented. Therefore, the security rules are specified on top of relational structures (major implementation platforms for Data Warehouses) such as tables or columns. As a consequence, we obtain a huge number of security rules that do not correctly satisfy the security requirements for final users. This is due to the fact that we do not have semantic multidimensional information from these relational structures. For this reason (and as we will show throughout this paper), we believe that if these security rules are specified on top of the underlying multidimensional model of the Data Warehouse, it will be possible to define the exact required security rules for every group of users as these rules will be defined based on the semantic relationship of the corresponding multidimensional elements, which are the elements upon which the final users' queries are based.

A new standard which addresses the complete life cycle of developing applications by using models in software development has recently appeared: Model Driven Architecture (MDA) [Miller and Mukerji (2003)]. In MDA technology, the standard for defining transformations is Model Object Facility (MOF) 2.0 Query /Views/ Transformations (QVT) [OMG (2005)]. MDA promotes the specification of a Platform Independent Model (PIM) which does not contain specific information about the platform or about the technology to be used to develop it. This PIM can be transformed into one or several Platform Specific Models (PSMs) by including platform and development technology specific information. Later, each PSM is implemented in a code which will be executed on a platform in order to obtain the final software product. Besides these models, a Computation Independent Model (CIM) is provided by MDA as a means of modeling user requirements.

The QVT specification has a hybrid declarative/ imperative nature, with the declarative part being split into a two-level architecture [OMG (2005)]: Relations and Core metamodels. In the relation metamodel, the model transformation between model candidates is specified as a set of relations. These relations must hold for a successful model transformation. In the imperative style, a Black-box implementation of operations can also be used to allow the reuse of existing algorithms or domain-specific libraries in certain model transformations.

Bearing these considerations in mind, we have previously proposed a preliminary version of QVT relations between PIM and PSM [Soler et al., (2007)] in the design of DWs which was validated through a case study presented in [Soler et al., (2007a)]. The PIM allows us to represent security and audit measures at the conceptual level for the DWs design, and is defined by using the approaches of [Fernández-Medina et al., (2006), Villarroel et al., (2006)]. Moreover, in [Soler et al., (2007b)] we presented a framework for the development of secure DWs based on MDA which permits the consideration of security issues in the MDA approach. Hence, our preliminary versions of QVT relations and the MDA framework constitute a natural continuation of the works of [Fernández-Medina et al., (2006), Villarroel et al., (2006), Fernández-Medina et al., (2006a)] which only incorporate security requirements in the DW design at the conceptual level.

In this paper, we significantly improve and complete our prior approach (previously-described) as follows: (i) we propose a new framework for the design of secure DWs (by means of MDA and QVT) by adopting a new division of the security space in software and application based on the approach proposed by McGraw in [McGraw (2002)], (ii) we provide a more in-depth revision of the related work as a new section, structured into two subsections which are focused on security in DWs and MDA and Security respectively, (iii) we replace the preliminary versions of the PSM employed in [Soler et al., (2007)] and [Soler et al., (2007b)] with a formalized extension presented in [Soler et al., (2008)] in order to consider the automatic generation of Code in the both DBMS (Database Management Systems) and OLAP tools, (iv) we include a new subsection with which to define relations that assure the transformation between associations and foreign keys from PIM and PSM, and finally, (v) we provide a new section explaining the current complexity of implementing QVT and how commercial tools are used to transform all the security and audit measures represented in the PSM into the corresponding code in both DBMS and OLAP tools. Therefore, this paper constitutes a natural extension through which to correctly complete our previous approach in the necessary manner. To the best of our knowledge, this is the first approach in the area of Data warehouses and OLAP which automatically generates the security rules defined in conceptual models in the final implementation.

The remainder of this paper is structured as follows. Section 2 presents related work. Section 3 presents a framework for the development of secure DWs in which we define, amongst other things, a Secure Multidimensional PIM (SMD PIM), a Secure Multidimensional PSM Model (SMD PSM) and a set of QVT relations. Section 4 uses a case study to show the application of our QVT relations. The applicability and limitations of the Architecture proposed is discussed in Section 5. Finally, Section 6 presents our conclusions and outlines future work.

2 Related Work

This section on related work begins with the main approaches dealing with security in DWs. We then study work related to the integration of security within the new MDA approach.

2.1 Security in DWs

The DW's architecture is composed of several layers, and security is involves in all layers and operations of the warehouse [Thuraisingham et al., (2007)]. The first layer contains the Data Sources (DS) which, through ETL process will feed the data warehouse. The second layer is the DW model, which represents the data structure that will support the information. The third layer contains the tools which offer automatic methods for analyzing, querying and mining the DS data, and which are usually OLAP tools or Data Base Management Systems (DBMS).

Firstly, since DS are heterogeneous and can use different access control security policies (such as discretionary, mandatory or role-based access control), the security problem in this layer is their integration into the DW design. DW users will be different from those of DS, so although an integrated security policy cannot be directly used in the DW as a final security policy, it might be an interesting starting point. Furthermore, since ETL processes extract and transform information from DSs and finally load it into the DW, it is important that ETL processes take security information into account. With regard to this layer, the main proposals are related to the same problem studied for Federated Information Systems (FIS) [Thuraisingham (1994)]; [Jajodia and Wijesekera (2001)]. One of the most interesting works is that of [Saltor et al., (2002)], in which this parallelism is used to adapt a design architecture for FIS to DWs, and also to improve it with security capabilities that support the integration of mandatory access control policies. However, we found very few modeling proposals relating to ETL processes, and those that we did find did not consider security [Trujillo and Luján-Mora (2003), Simitsis and Vassiliadis (2008)].

The DW model (second layer) is highly important, and is usually designed through the definition of several models at different abstraction levels (business, multidimensional, logical and physical). Several works are focused on the secure modeling of DWs at a certain abstraction level. At the business level there are proposals based on ontologies, business process, UML, etc. Paim and Castro [Paim and Castro (2003)] include security requirements, but they do not offer any formal metamodel. Furthermore, the Tropos methodology [Giorgini et al., (2006)] considers the issues of security and trust as a part of its development process. This methodology is based on social hierarchies and adapts components of the $i*$ framework to define the obligations of actors (dependees) towards other actors (dependers). This proposal is improved with new security concepts: constraints, secure entities (secure goals, tasks, resources, ownership) and secure dependences between actors (such as trust of execution, trust of permission, delegation of permission and delegation of execution).

At the conceptual level there are interesting works for the modeling of DWs which consider their special characteristics by using extensions of the ER model, UML or their own notation, but they do not include security capabilities [Golfarelli et al., (1998)], [Sapia et al., (1998)], [Tryfona et al., (1999)], [Binh et al., (2000)], [Abelló et al., (2006)], [Luján et al., (2006)], [Prat et al., (2006)]. The conceptual modeling of security issues is solely considered by the AdaptedUML of Priebe and Pernul [Priebe and Pernul (2001)].

Logical modeling depends on the technology used (ROLAP, MOLAP, HOLAP, etc.). In [Katic et al., (1998)] the authors describe a prototype model for DW security based on metadata, which enables the definition of views of data for each group of users. However, this does not allow us to specify complex restrictions of confidentiality. Rosenthal and Sciore [Rosenthal et al., (2000)] extend SQL grants and create a mechanism of inferences through which to establish the security of DWs, which gives permission to access the tables and views of the system.

DBMS and OLAP tools have also considered security constraints in order to avoid unauthorized accesses. In fact, the most popular tools integrate functionality, which allows developers to specify security constraints. This is important, but it is not a good solution from the engineering point of view, because security, as with any other requirement, should be identified and modeled at the moment of analysis and design, and should then be implemented in accordance with this analysis and design. On the other hand, the inference problem remains as a challenge in DW security and is an important research branch [Thuraisingham et al., (2007)]. This problem is similar to the that previously studied, problem for statistical databases which store summarized data such as sum or averages [Shoshani (1997)].

Finally, a global proposal exists which attempts to integrate security into the complete DW development process [Priebe and Pernul (2001)], and which is based on the classical database design methodology (requirement, analysis, conceptual, logical, and physical design). This proposal covers requirements and concrete implementations in SQL Server Analysis Services (SSAS), creating a Multidimensional Security Constraint Language (MDSCL) by extending multidimensional expressions (MDX) with hide statements for cubes, measures, slices and levels. The same authors extend the ADAPTed UML (which uses ADAPT symbols as UML stereotypes) model for the previous conceptual phase, specifying a methodology and a MD security constraint language for the conceptual modeling of OLAP security. This proposal is extremely interesting, but although roles are used, the authors do not use a role-base access control policy. They include a simplified role concept (without hierarchies) that represents a subject in a discretional access control policy. Furthermore, they only hide multidimensional elements without supporting the complex security constraints involving conditional evaluation. Finally, this methodology does not establish the connection between levels in order to permit automatic transformations.

2.2 MDA and Security

The use of Model Driven Development is one of the most intuitive strategies through which to develop more secure information systems. However, the philosophy of model driven engineering when applied to the development of secure information systems is different to that of traditional security models which describe the protection needs of the systems. Security models are therefore 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.

Several works dealing with the integration of security with UML and other modeling languages exist. One of the most relevant proposals is UMLsec ([Jürjens (2002), Jürjens (2004)]), which integrates security into the information systems through UML and can be employed to specify and evaluate UML security specifications using formal semantics. Furthermore, [Burt et al., (2003)] propose a PIM for several existing PSMs, but only over the description of the transformations, and the work in general is devoted to access control patterns for an area related to Middleware and Grid Technologies. In [Sivanandam and Karpagam (2004)] a schema appears through which to design and develop security services by using an MDA based approach which takes the smart card as its application. This work helps in the development of MDA Components for Security services in new and legacy systems. OpenPMF

[Lang and Schreiner (2004)] is a technology framework based on MDS, which allows the centralized, integrated management of security policies for complex distributed systems.

A novel methodological approach for the model driven development of secure XML databases (DB) can be found in [Vela et al., (2006)]. This proposal is within the context of Web Information Systems based on MDA. A tooling framework with which to generate Web service security configurations by using MDA and Service-Orientated Architecture (SOA) appears in [Nakamura et al., (2005)]. In [Hafner et al., (2006)] the authors propose a novel approach based on MDA and MOF-QVT standards. Their approach is very interesting, but refers only to the Global workflows that specify the message flow between a pattern distributed environment with no central control.

In SecureUML [Lodderstedt et al., (2002)], an approach through which to include security in UML, the term Model Driven Security (MDS) appears for the first time. MDS ([Basin and Doser (2006)]) applies the MDA approach to include security properties in high-level system models and uses tools to automatically generate system architectures from the models, including security infrastructures to automatically generate secure system architectures. The approach is very rich but focuses exclusively on access control in the context of a platform which is oriented towards logic applications and target objects (.NET and J2EE). MDS extends MDA in three respects: i) the system models are enriched with primitives and rules for integrating 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 fulfil this goal, the authors consider dialects which provide a bridge by defining the connection points with which to integrate elements of the security modeling language with elements of the system design modeling language.

Other applications of MDS also exist, such as that of [Lang and Schreiner (2004), Hafner et al., (2006)], which clearly establishes the PIM, the PSM and the necessary transformation between them, or the MDS through UMLsec of [Best et al., (2007), Jürjens et al., (2008)], which defines three abstraction levels (requirements, models and code), and provides both direct and reverse engineering, verification, configuration, etc., thanks to a rich set of tools [Jürjens and Shabalin (2007), Jürjens (2009)], etc. All these proposals are of great interest, and offer great contributions towards the development of more secure information systems, but none of them refers to the development of secure DWs, which have different security requirements, and a different development process.

In the following section we present the core of our proposal (A Framework for the Development of Secure DWs). Our goal is attained through the integration of the proposal of [Mazón et al., (2008)] based on MDA and QVT with the MD modeling of secure DWs [Villarroel et al., (2006)]. Within this context, we use as our CIM model the work presented in [Soler et al., (2008a)], by applying the $i*$ framework to elicit both functional and security requirements for the DW design at the business level. The PIM model corresponds to the extension presented in [Fernández-Medina et al., (2006)] and [Villarroel et al., (2006)], which is called the Secure Data Warehouse (SECDW) metamodel. The PSM corresponds to the extension of the Relational metamodel [Soler et al., (2008)] from the Common Warehouse Metamodel (CWM) [OMG (2003)] at the logical level. The extended metamodel is called Secure Relational Data Warehouse (SECRDW). The final section deals with the code generation for both DBMS and OLAP tools and with the reverse engineering process of the framework proposed.

3 A Framework for the Development of Secure DWs

This section proposes a framework for the development of secure DWs, which integrates MDA and QVT with the MD modeling of secure DWs (see Fig. 1). We focus on defining transformation T_2 between a Secure Multidimensional PIM (SMD PIM) and a Secure Multidimensional PSM (SMD PSM). Therefore, Subsections 3.1 and 3.2 introduce an SMD PIM and an SMD PSM. In Subsection 3.3 we define QVT relations between SMD PIM and SMD PSM, i.e, transformation T_2 in Fig. 1. Subsection 3.4 explains certain issues related to transformation T_3. Subsections 3.5 and 3.6 are devoted to the discussion of secure DWs implementation for DBMS (Oracle 11g) and OLAP tools. Finally, Subsection 3.7 introduces the latest technical advances from OMG through which to carry out reverse engineering.

MDA is based on building system-independent models and transforming them into efficient implementations. Hence, we need to choose metamodels whose instances will serve as PIM and PSM. We moreover define a transformation between them. Fig. 1 uses a diagram to illustrate our Secure multidimensional MDA framework for the development of secure DWs. The upper section presents the CIM which defines both functional and non functional requirements for the DWs. It represents a perspective of the DW within its business environment and thus plays an important role in reducing the gap between those who are experts in the domain and their requirements and those who are experts in the design and development of the DW, which needs to satisfy these requirements.

The transformation T_1 in Fig. 1 is used to map SMD CIM into SMD PIM. The use of formal languages to capture the key business activities is too complex for analysts and requirements engineers. The modeling of the CIM with no formal languages entails ambiguity in the CIM's definition. That makes the formal transformation from CIM to PIM (transformation T_1) difficult. We are currently

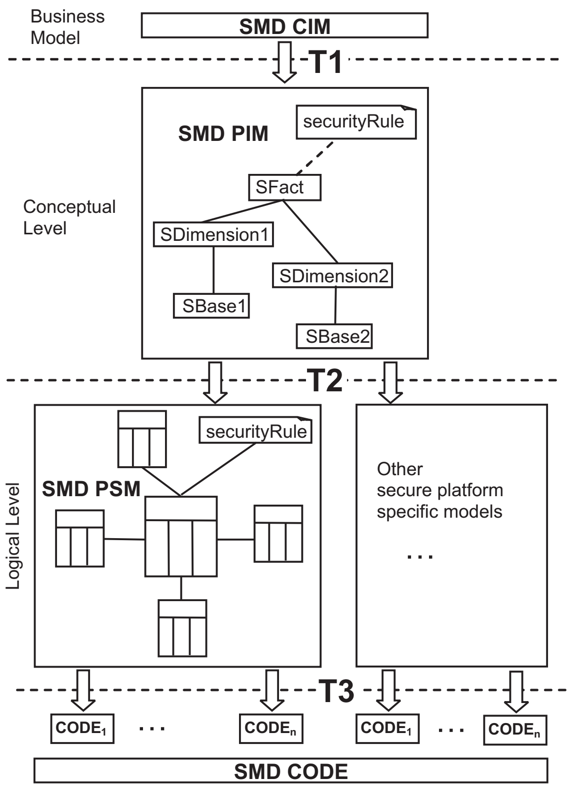

Figure 1: A Framework for the Development Secure DWs.

working on the transformation T_1, but it has not yed been developed due to its complexity. According to MDA, several PSMs can be transformed from a PIM. Therefore, our secure relational platform (SECRDW) is represented on the left hand side of the area corresponding to the logical level, while any other secure PSMs are represented on the right hand side of the same area. By secure PSM we understand a PSM whose metamodel supports the security modeling at the logical level. Consider as an example, an extension for modeling security issues for the Multidimensional metamodel from CWM. The right hand side of Fig. 1 can therefore represent a secure Multidimensional Online Analytical Processing (MOLAP) PSM. Hence, transformation T_2 allows us to derive an SMD PSM or other secure PSMs.

Starting from each SMD PSM, the corresponding code is derived for the target platform, which corresponds to ours at the physical level. This code is called a Secure Multidimensional Code, (SMD Code) and it is represented in the lower section of Fig. 1 by using transformation T_3. Note that the security restriction defined by using OCL (represented as a UML note) is transformed from the conceptual level to the logical level by means of T_2, and is later transformed into code by means of T_3.

3.1 Definition of the SMD PIM

The SECDW metamodel allows us to represent the main security requirements for the conceptual modeling of the DWs. Fig. 2 represents the SECDW metamodel, and its instances are modeled through the *secureDW* class. Both *secureDW* and certain attributes are omitted to make the metamodel more comprehensible.

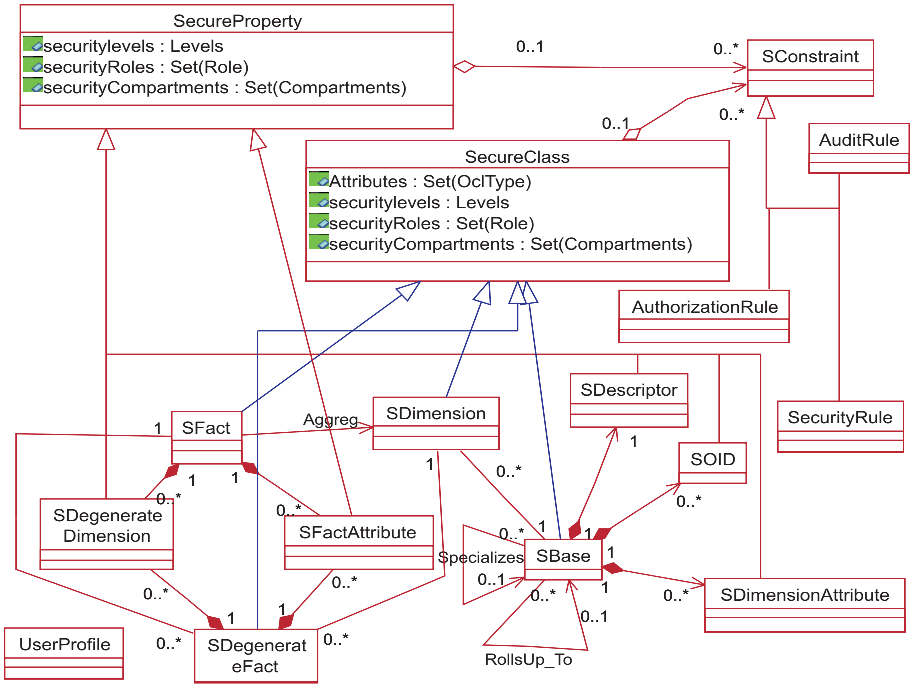

Figure 2: Metamodel used in the design of SMD PIM.

As security requirements are modeled in this PIM, it is therefore denominated as SMD PIM (Secure Multidimensional PIM). In the metamodel *SFacts* are specified as composed classes by means of aggregation relationships of n *SDimensions* classes. *SDimensions* classes are composed of classification hierarchy levels; every classification hierarchy level is specified by an *SBase* class. An association *RollUp_To* between *SBases* classes specifies the relationship between two levels of a classification hierarchy. This classification hierarchy does not contain any cycle. The multiplicities defined by an association *RollUp_To* addresses the concepts of *strictness*, *non-strictness* and completeness of a classification hierarchy. The *Specializes* associations between *SBases* classes guarantee the categorization of *SDimensions* classes represented by generalization-specialization relationships.

FactAttributes and *SDegenerateDimension* represent attributes for the *SFact* class. *SOIDs*, *SDescriptors*, and/or *SDimensionAttributes* represent *SBase* attributes. *SDegenerateDimension* attributes are defined in the *SFact*. A *SDegenerateFact* represents a UML association class attached to a *many-to-many* aggregation relationship between an *SFact* class and an *SDimension* class, which may contain *SFactAttributes* and *SDegenerateDimensions*. *SDegenerateDimension* is an *SDimension* which is stored as an attribute of the *SFact* class. *SDegenerateDimensions* are useful when attempting to associate the *SFacts* in the DW with the original data sources. The *UserProfile* metaclass contains information about each user's right of access to the MD model.

The metamodel also allows us to represent the main security information on data and their constraints in the multidimensional modeling at the conceptual level. The security information is based on a combination of the Multilevel Security Model which allows us to classify both information and users into security classes, enforcing the mandatory access control (MAC), together with the role-based access control (RBAC). For each element of the metamodel (*SFact*, *SDegenerateFact*, etc) its security information is defined in three ways. (i) a sequence of security levels (*SecurityLevels*) that indicates the clearance level of the user. (ii) A set of user categories (*SecurityCompartment*) used by an organization to classify users into a set of horizontal groups, such as geographical location, area of work, etc. (iii) A set of user roles (*SecurityRoles*) used by the company to organize users into a hierarchical role structure, according to the responsibilities of each type of work. Therefore, the metamodel includes six data types to define the tagged values contained within the *SecureClass* and *SecurityProperty* classes in order to establish security information in the systems' classes (*SFact*, *SBase*, etc). The security information can thus be defined in each instanced object of the metamodel, thereby specifying with high accuracy which users can access each particular object.

Additionally, the *SFact*, *SDegenerateFact*, *SDimension*, *SBase*, *SDegenerateDimension*, *SFactAttribute*, *SDescriptor*, *SOID* and *SDimensionAttribute* classes have security constraints (*SConstraint*) to indicate the security level and the rights conceded to a user who wishes to access certain information. For example, an authorization rule (*AuthorizationRule*) could represent an exception made to permit specific users to access certain information. The access type may depend on the value of certain attributes contained in several classes. This fact can be captured in the model by means of a security rule (*SecurityRule*). If a user attempts to access information to which his/her access is denied, then this fact can be modeled with an audit rule (*AuditRule*). These restrictions are defined by using an Object Constraint Language (OCL) extension [Fernández-Medina and Piattini (2004)] and are represented at model level with a UML note associated with the corresponding class. More details of this profile can be found in [Fernández-Medina et al., (2006)] and [Villarroel et al., (2006)].

3.2 Definition of the SMD PSM

In the design of databases and DWs, conceptual modeling provides the PIM, and the logical modeling of the PSM. In MD modeling, the logical level is designed according to the specific properties of the Database Management Systems (DBMS) such as Relational Online Analytical Processing (ROLAP), MOLAP or Hybrid Online Analytical Processing (HOLAP). Nevertheless, Kimball [Kimball and Ross (2002)] assures us that the most common representation is through the relational platform (ROLAP systems).

It it necessary to consider types of secure PSMs which allow us to represent, at the logical level, all the security and audit rules captured by using the SECDW metamodel during the conceptual modeling stage of the DWs. To the best of our knowledge, the SECRDW metamodel is the only existing extension for the relational metamodel with which to represent at the logical level all the security and audit measures captured during the conceptual modeling phase of the DWs design. Unfortunately, as we stated previously, other metamodels for PSMs such as MOLAP or HOLAP have to be extended in order to support security issues. Also, the definition of the transformation between previous PSMs can be defined by starting from our QVT transformations.

Our PSM is described by an extension of the Relational Metamodel from the CWM. The main purpose of the CWM [OMG (2003)] is to enable the easy interchange of warehouse and business intelligence metadata between warehouse tools, warehouse platforms and warehouse metadata repositories in distributed heterogeneous environments. In order to distinguish the security aspects that the SECRDW metamodel comprises, it will, from here on, be called the Secure Multidimensional PSM (SMD PSM). See Fig. 3.

The SECRDW metamodel defines a container *SSchema* which is inherited from Schema. See Fig. 3. *SSchema* is a collection of *STables* and *securityProperties* and is aimed at security at the schema (catalog) level. A *ColumnSet*

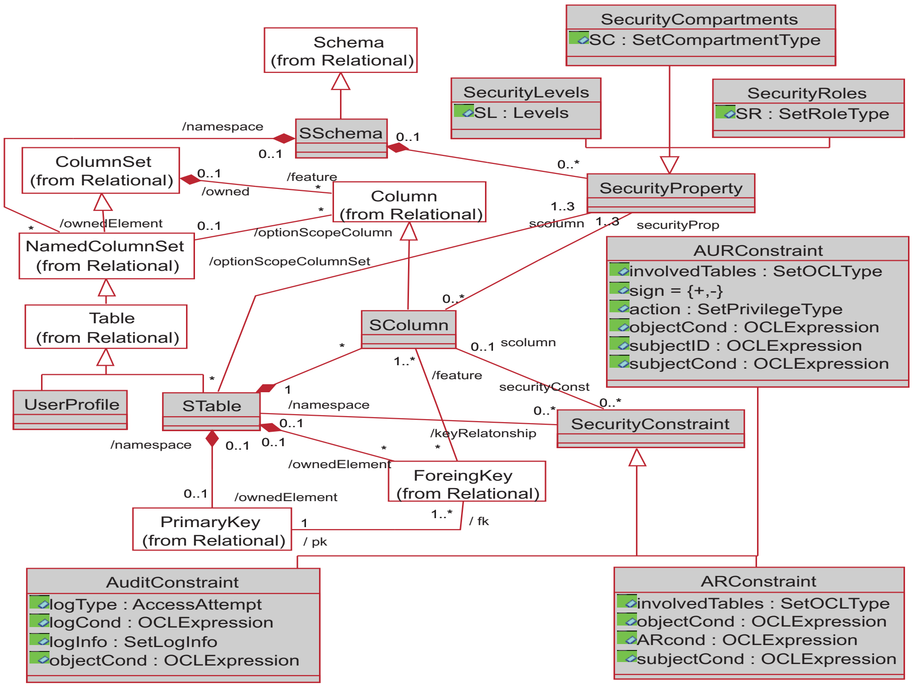

Figure 3: Metamodel used in the design of SMD PSM.

represents any form of relational data. An *STable* and *UserProfile* are inherited from *Table*, which contains *Columns*. *SColumn* specializes in the *Column* metaclass, and is owned by the *STable* metaclass. The *UserProfile* table contains columns through which to specify the access properties (*securityProperty*) that the user has. *UserProfile*, unlike *STable*, is unique and has no association with the other tables in the system. A *ForeignKey* associates columns from one table with columns from another table. The *PrimaryKey* class inherits from the *UniqueConstraint*. The *PrimaryKey* and *ForeignKey* metaclasses are owned by the *STable* metaclass. Certain metaclasses are used to represent security and audit measures in the metamodel. The *SecurityProperty* metaclass inherits from the Class (from the Core) metaclass and specializes in *SecurityLevels*, *securityCompartments* and *securityRoles* classes. The associations between *SecurityProperty* with *STable* and *SColumn* allow us to establish information security by means of *securityLevel*, *securityCompartment* and *securityRole*.

Furthermore, *SecurityConstraint* is used to represent security constraints. *SecurityConstraint* (which inherits from *Constraints* metaclass) allows us to define *AuditConstraint*, *ARConstraint* and *AURConstraint*. *AuditConstraint* is useful both as a deterrent against misbehaviour and as a means to analyze user behaviour by employing the system to discover possible attempted or actual violations. *AuditConstraint* is essential to record the accesses to tables and columns which are performed by users. *ARConstraint* allows us to define rules with which to specify multilevel security policies in tables and columns. *AURConstraint* enables us to specify access to the tables and columns, thus permitting us to specify much more elaborate security models. The associations between *SecurityConstraint* with *STable* and *SColumn* allow us to establish security rules by using *AuditConstraint*, *ARConstraint* and/or *AURConstraint*. Moreover, the *SecurityConstraint* allows us to express the (*AuditRule*, *AuthorizationRule* and *SecurityRule*) constraints which are modeled through the UML notes in the SECDW metamodel (SMD PIM).

3.3 The SMD PIM - SMD PSM QVT relations

Fig. 4 uses textual notation to show the main transformation between SMD PIM and SMD PSM. The *top* keyword which precedes the relations specifies that these relations will never be invoked by any other relations throughout the transformation. Each of these relations has its own *when* and *where* clauses which correspond (respectively) to the pre and post-conditions which have to be satisfied. We shall now explain the general idea of the transformation shown in Fig. 4.

The execution of the transformation shown in Fig. 4 first calls the relation (8), then for each *secureDW* in the SECDW model, the relation first checks whether an *SSchema* with the same name exists in the SECRDW model, and if it does not, a new *SSchema* is created in that model with the given name. Relation (9) transforms *UserProfile* with its attributes into a *RUserProfile* table with its corresponding columns. Relation (10) is defined in Subsection 3.3.1, and its main relations invoked in the *where* clause are explained in subsections 3.3.2 and 3.3.3. We shall not define the *SDegenerateFact2STable* relation (see (11) in Fig. 4) because it is very similar to the *SFact2STable*. Next, relation (12) is defined in Subsection 3.3.4 by following the snowflake schema paradigm for representing DWs at the logical level, as we can see in Subsection 3.3.5 in which we explain the relations defined in its *where* clause. Finally, the relations marked with the numbers (13), (14) and (15) assure the relationships between the *STables* corresponding to the *SRootBase*-*SDimension*, *SFact*-*SDimension*, and *SDegenerateFact*-*SDimension* associations in the SECDW model.

The definition of the *SDimension2STable* relation is very interesting, since it determines the complexity of the QVT transformation. We should note that in order to build a star schema at the logical level we need to transform the whole dimension hierarchy into the same *STable*, i.e, all the security information for *SBases* that conform with the *SDimension* are mapped into the same *STable*.

```
Transformation SMD To SREL(SMD: SECDW,)  SREL: SECRDW)
{
(1) key Table{name,  SSchema};
(2) key Column {name, owner};
(3) key UserProfile{name, SSchema};
(4) key PrimaryKey{name, owner};
(5) key ForeignKey{name, owner};
(6) key SecurityProperty{name, owner};
(7) key SecurityConstraint(name, owner);
(8) top relation SecureDW2SSchema{}
(9) top relation UserProfile2RUserProfile{}
(10) top relation SFact2STable {}
(11) top relation SDegenerateFact2STable{}
(12) top relation SDimension2STable{}
   //Association SFact with SDimension
(13) top relation AssocSF_D2FKey{}
   //Association SDegenerateFact with SDimension
(14) top relation Assoc SDF_SD2F keyFKey
   // Association SDegenerateFact with SFact
(15) top relation AssocSDF_SF2FKey{}
}
```

Figure 4: Textual notation for the main Transformation.

Therefore, the complexity of the QVT transformation is increased. Nevertheless, the transformation can be defined by applying several functions based on tree structures in the *where* clause which returns correct security values for *STables* and *SColumns*.

3.3.1 The SFact to STable Transformation

Fig. 5 illustrates the *SFact2STable* relation in its graphical notation. One table corresponds to *SFact* and has the same name. This table has a column with a name (specified in the *where* clause), which is also the primary key of the table. The security information represented with tagged values in the *SFact* is transformed into objects associated with the table. This security information is modeled at the logical level under the heading of the *SFact* table. The *SFAct2STable* relation is satisfied only when the *SecureDW2SSchema* pre-condition is satisfied, thus ensuring that the table will be contained in a DW SSchema. The *SFact* attributes, together with their security information and constraints, are transformed according to the *SFactAttribute2SColumn* (if the attribute's type is *SFactAttribute*) or *SDegenerateDimension2SColumn* (if the attribute is of an *SDegenerateDimension* type) relations. Every *SFactAttribute* involved in the *SFactAttribute2Column* relation inherits security information and restrictions from the *SecureProperty* class, as the latter contains the *SConstraint* metaclass. We shall now define the *SFactAttribute2SColumn* relation presented in the *where*

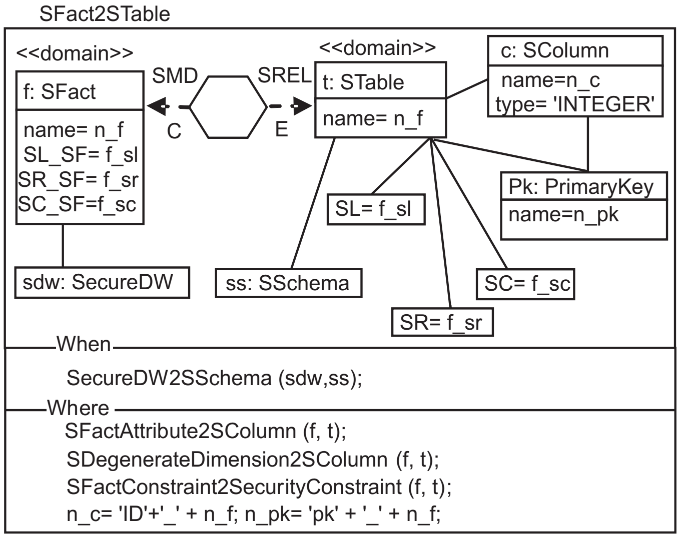

Figure 5: Transforming *SFact* into *STable*.

clause of the *SFact2SColumn* relation.

3.3.2 The SFactAttribute to SColumn Transformation

Fig. 6 shows the graphical notation for the *SFactAttribute2SColumn* relation. Each *SFactAttribute* inherits information and security restrictions from the *SecureProperty* class, as this contains the *SConstraint* metaclass. Hence the *SFactAttribute2SColumn* not only transforms attributes into columns, but also transforms all the associated security information that the *SFactAttribute* contains at the conceptual level. This information is modeled at the logical level next to each column in the table that represents the *SFact*. The relation that appears as a post-condition transforms the security restrictions of each *SFactAttribute* into an object associated with the corresponding column. This object is modeled as a note associated with the column. It should be noted that the *SMDType2SRELType* () function converts a data type of the initial metamodel (i.e., Secure DW, SECDW) into a certain data type of the final metamodel (i.e., Secure Relational DW, SECRDW).

3.3.3 The SFactConstraint to SecurityConstraint Transformation

Let us assume that in the case of the *SFactAttribute2SColumn* relation, all the relations of the *where* clause have been defined. If we return to the content of

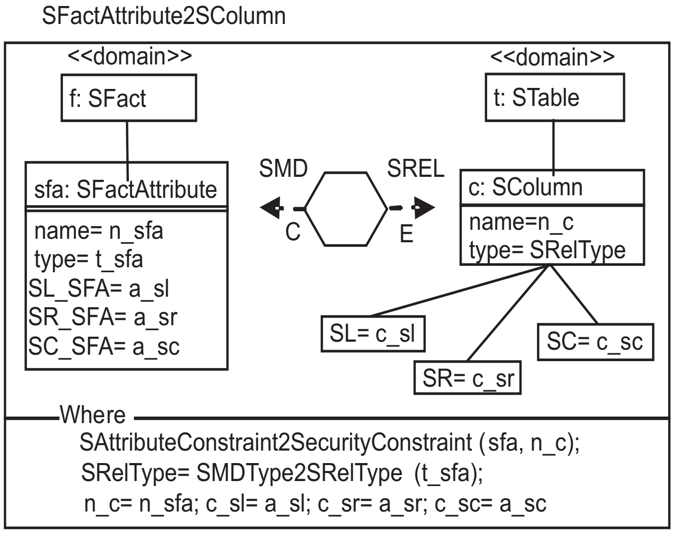

Figure 6: Transforming *SFactAtribute* into *SColumn*.

the post-conditions that appear in the *SFact2STable* relation, we should now establish the *SConstraint2SecurityConstraint* transformation.

Fig. 7 illustrates the graphical notation provided by QVT to define the *SFactConstraint2SecurityConstraint* relation. Certain attributes have been omitted in order to make it more comprehensible. When the relation is applied, the *AuditRule*, *AuthorizationRule* and *SecurityRule* constraints are transformed into *AuditConstraint*, *ARConstraint* and *AURConstraint* respectively, but are now associated with the table that represents the *SFact*. It may be noted that this relation's *where* clause employs OCL to ensure that the "s_icsr" attribute corresponds to the tables that represent the *involvedTables* attribute of the *ARConstraint* restriction.

3.3.4 The SDimension to STable Transformation

If we continue with the main transformation that appears in Fig. 4, it now follows that we apply the *SDegenerateFact2STable* relation that guarantees a *many-to-many* relationship between *SFact* and *SDimension*. As we explained previously, this is not defined. In Fig. 8 we show the definition of the *SDimension2STable* relation. The way in which we define the *SDimension2STable* relation determines the logical schema type in the relational platform (star, factconstellations or snowflake schemas). In our context, we employ the snowflake schema. In our approach, the dimensions do not have attributes [Luján et al., (2006)], and for

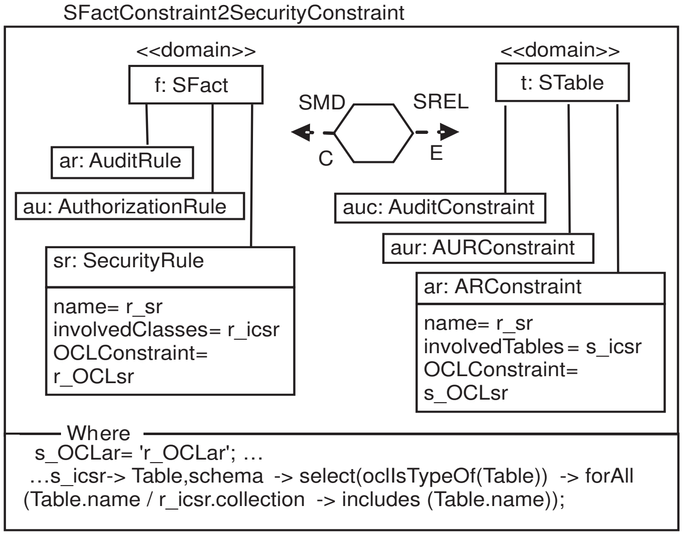

Figure 7: *SFactConstraint* to *SecurityConstraint* Tranformation.

this reason, when the *SDimension2STable* relation is executed, a table is created whose name is merged with the names from the dimension and *rootBase* respectively. The *rootBase* is the only *SBase* associated with the *SDimension*. All of the security information associated with the *rootBase* is transformed into the security properties of the table, and the execution of the relations that appear in the *where* clause of the *SDimension2STable* relation is used to guarantee that all the attributes of the *rootBase* will conform to the columns in the table.

3.3.5 The SBase to STable Transformation

Fig. 9 shows the definition of the *SBase2STable* relation. This relation creates a table with both a primary key, and a foreign key in the table which it receives as a parameter when it is invoked; the primary key and the foreign key will be associated in order to guarantee that the tables form a part of a *one-to-many* relation between the *SBases*. In the *where* clause this relation, along with the *SpecializedSBase2STable* relation, are re-called to assure that we cover the entire hierarchy of bases to which the dimension conforms.

The first four relations that appear in the *where* clause of the *SBase2STable* relation guarantee the transformation of all the *SBase* attributes in the columns of the table that represents the *SBase*, as well as the transformation of all the constraints associated with the *SBase* in constraints associated with the table that represents the *SBase*. The calls to the *SBase2STable* and *SpecializedBase2STable*

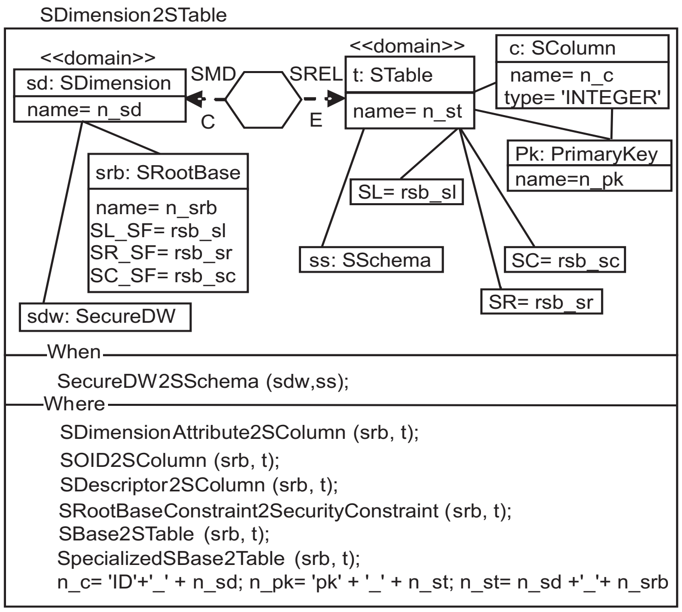

Figure 8: Transforming *SDimension* into *STable*.

relations permit us to recursively cover all the hierarchy of the bases to which the dimension conforms. The *SpecializedBase2STable* relation has a certain similarity to the *SBase2STable* relation, and will not therefore be defined. This comment closes our definition of both the *SBase2SColumn* and the *SDimension2STable*.

3.3.6 Associations to ForeignKeys Transformations

Returning to the transformation shown in Fig. 4, this now corresponds to relations (13), (14) and (15), which map associations from SMD PIM (*SFact-SDimension*, *SDegenerateFact-SDimension* and *SDegenerateFact-SDimension*) into the corresponding *ForeignKey* from SMD PSM. Hence, each relation (i.e, (13), (14) and (15)) is responsible for adding a foreign key in a previously created *STable*. This necessitates retrieving the existing *STable*. Fig. 10 depicts how the *SFact-SDimension* aggregation (relation (12) from Fig. 4) is transformed into a foreign key (FKey) belonging to the *STable* mapped from *SFact*. The FKey is associated with the *PrimaryKey* of the *STable* mapped from *SDimension*. Therefore, the transformation assures a *many-to-one* relationship between the *STables* mapped from *SFact* and *SDimension* respectively. Relations (14) and (15) from the main transformation shown in Fig. 4 are similar to the relation defined in Fig. 10. These relations establish a *many-to-one* relationship between both the

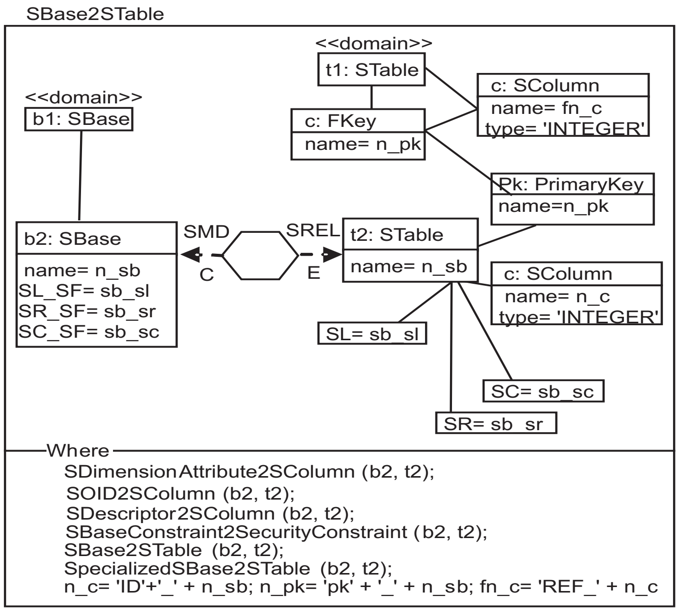

Figure 9: Transforming *SBase* into *STable*.

STables mapped from *SDegenerateFact* and *SDimension* and *SDegenerateFact* and *SFact* respectively. Hence, they are not defined.

3.4 Implementing the SMD PIM - SMD PSM transformation

In this subsection, we briefly explain how the transformations T_2 and T_3 shown in Fig. 1 can be implemented. Due to space constraints, we shall only refer to the complexity of implementing QVT relations in implementing platforms and the platform we have used. As has already been described, we have defined QVT relations in order to transform the SMD PIM into the SMD PSM.

However, one of the current pitfalls in using QVT is how to implement its relationships. As we have previously stated, the declarative and imperative styles conform to the QVT specification. In our proposal we have chosen the QVT declarative approach, since it is more understandable than the imperative one. However, there is no available QVT transformation engine that allows us to directly implement these declarative QVT relations. One solution is that of deriving an imperative version of these declarative relations and implementing these imperative QVT relations in an imperative QVT engine such as that provided by the Eclipse platform [Borland (2007)].

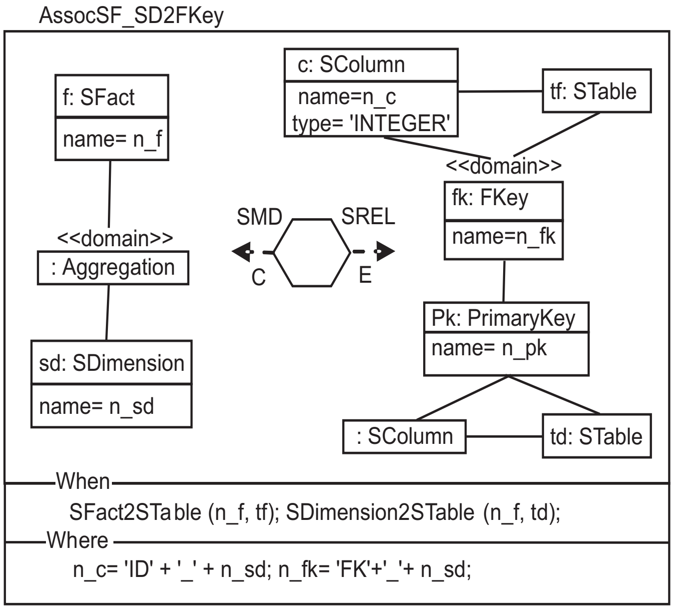

Figure 10: Transforming *SFact-SDim* aggregation into FKey.

Eclipse is a platform which supports MDA and allows us to use the QVT language in order to implement transformations between models. We have also used the SmartQVT open source [SmartQVT (2007)] within the Eclipse platform in order to correctly implement the imperative part of our QVT relations. Therefore, we have translated our declarative relations described in Section 3 into their corresponding imperative versions in an absolutely straightforward way according to the QVT specification [OMG (2005)].

Fig. 11 shows a snapshot of the Eclipse platform for modeling the case study described in the following section. The Plug-in we have developed allows us to model secure DWs by using our proposed PIM. The corresponding QVT and Model to Text relations will thus allow us to obtain the final implementation of the DW together with their corresponding specified secure rules.

3.5 SMD PSM - MSD Code for DBMS

This subsection briefly shows the possibilities that Oracle 11g DBMS offers in implementing the main aspects of the security issues represented in SMD PSM [Jeloka (2007)]. Oracle 11g supports security and audit facilities by means of its components, namely Oracle Label Security (OLS11g), Virtual Private Databases (VPD) and Oracle Fine-Grained Auditing (FGA).

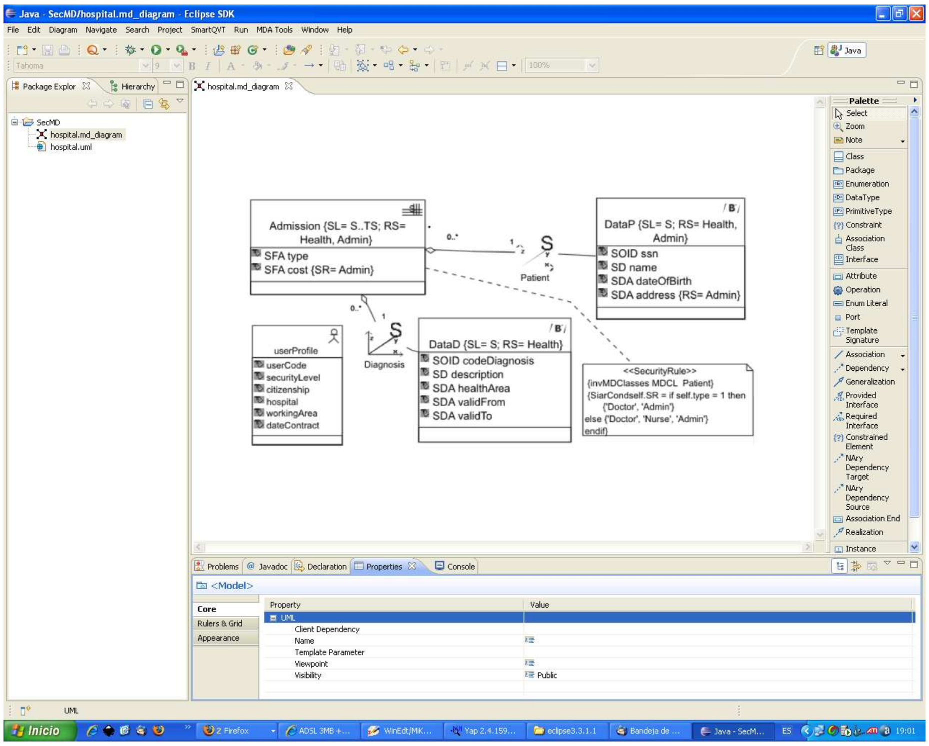

Figure 11: An Eclipse snapshot using our approach.

OLS11g defines labels that contain confidentially for rows, and authorization information for users. All security information specified in OLS11g has to be performed within the context of a security policy. This mechanism is called a labeling function and defines the information of the security label according to the value of the columns of the rows that are inserted or updated. Once the security policies have been defined, they are applied to tables and attributes in order to assure information confidentiality. VPD allows us to restrict access to specific rows in a table. What is more, VPD can be used to define a policy function which will allow any individual user to see a completely different set of data, which consists only of the data that a particular user is authorized to see. FGA allows us to define and implement policies through which to audit select, update, insert and delete statements in tables and views according to an audit condition.

Since OLS 11g manages the main security concepts specified by our logical models, the transformation from the logical level into SMD code for this tool can be easily obtained. That is, OLS 11g permits a classification of subjects and objects in security levels, roles and compartments, along with the specification of security constraints. However, other tools need more complex procedures to transform our logical concepts into final code. The current proposal from OMG used to define the transformations model-to-code is the "MOF Model to Text Transformation Language RF".

The steps needed to implement a secure DW in OLS 11g are the following: (1) the definition of a security policy; (2) the establishment of the systems's security configuration by creating the security levels, compartments and roles needed; (3) the definition of labeling functions for each security constraint by establishing the security compartments, levels and roles which can access the information; and (4) the application of the defined security policies in the DW tables.

3.6 SMD PSM - MSD Code for OLAP tools

Apart from the generation of the corresponding Code to automatically implement the Security rules in a DBMS (as is shown in the previous section), our MDA architecture also supports the automatic generation of the Code in order to generate the secure required information for OLAP tools. In this section, we will focus on MOLAP (Multidimensional OLAP) tools, which are based on proprietary multidimensional models based on cubes and hierarchies of dimensions and base classes. In order to provide this information, we need: (1) to create a new PSM metamodel focused on a MOLAP approach; (2) to define the automatic transformation from conceptual models (PIM) to multidimensional logical models (PSM); and (3) to implement the logical model in a specific OLAP tool and to define transformation rules from PSM models to code.

SQL Server Analysis Services (SSAS) has been selected as the target OLAP tool to be used for our multidimensional implementation because it allows security measures to be defined in multidimensional structures. SSAS uses a role-based security policy in which each role contains one or more specific user accounts or user groups. These roles are used to establish security constraints for metadata and multidimensional elements, permitting: administrative authorizations for processing a database, cube, dimension or mining structure; read authorizations on metadata which show or hide definitions of databases, cubes, dimensions or mining structures; and access control through which to grant or revoke user accesses to data sources, cubes, cell data, dimensions, dimension data, and mining structures. We can also define more complex access control constraints involving conditions at a fine-grain level by establishing Multidimensional Expression (MDX) with permitted and denied sets.

The "Visual Totals" property is another interesting characteristic of SSAS, which filters the aggregate values in order to calculate them by using only the visible elements with regard to the defined security measures for the role. SSAS also allows us to control inferences on derived cells by defining contingence permissions for read accesses. Cell data can therefore only be shown if we have read permissions over all the cells from which they are derived. Finally, SSAS offers data auditing capabilities which include audit objects, auditing DDL commands and support for multiple logging targets.

Therefore, once the multidimensional logical model (PSM) has been obtained, the first step towards implementing a secure DW in SSAS is to define a hierarchy of roles which establish the security configuration of the system by means of a role-based policy. The security constraints defined over the multidimensional elements must then be implemented as permissions related to the corresponding cubes, cells, dimensions or attributes. Sets of positive and negative permissions for authorized and unauthorized roles are therefore created for each security constraint. More complex security rules that involve condition evaluations are also implemented by permissions, but in this case, the condition to be evaluated is included as a denied or permitted set a by using multidimensional expression (MDX).

3.7 Modernization processes

Reverse engineering analyzes legacy systems, identifying the system's elements and their interrelationships and carrying out representations of the system at a higher level of abstraction. MDA provides the necessary formalization with which to reengineer a process for it to converge on the so-called Architecture-Driven Modernization (ADM), another OMG initiative [OMG (2006)]. ADM advocates reengineering processes in which each artifact involved in these processes is depicted and managed as a model [Khusidman and Ulrich (2007)].

Our MDA architecture is being fulfilled with an ADM process focused on the multidimensional path which allows us to automatically obtain higher abstraction models (PIM) from OLAP code [Mazón et al., (2007)]. In a first stage, the according multidimensional logical model (PSM) is obtained from the source code of the OLAP tool by applying a static analysis [Canfora and Penta (2007)], which is a reengineering method based on the generation of lexical and syntactical analyzers for the specific tool. Code files are thus analyzed and a set of code-to-model transformations create the corresponding elements in the target logical model. Once the logical multidimensional model (PSM) has been obtained, several sets of QVT rules carry out a model-to-model transformation towards the corresponding conceptual model (PIM).

The modernizing of DWs provides us with various benefits, such as theability to generate diagrams in a high abstraction level (PIM) in order to facilitate the identification of security lacks and to include new security constraints which solve these identified problems. Transformation rules can be then applied, thus obtaining an improved logical model and the final implementation. By following the MDA philosophy the system can also migrate to different technologies (MOLAP, ROLAP, HOLAP, etc.) and different final tools (SSAS, OLS, etc.).

4 Applying QVT relations - a case study

This section explains how the defined relations are applied. If we suppose that a hospital wishes to automate its patient admission process, then the type of information involved requires confidentiality.

Fig. 12 shows an instance of the SECDW metamodel, (i.e., SMD PIM) to illustrate the part of the data warehouse that is required to solve the aforementioned problem. The instance of the SECDW uses the Level, Levels and Role data types defined in [Villarroel et al., (2006)]. The data types allow us to define the *SecurityLevels* and *SecurityRoles* tagged values depicted in Fig. 12 with SL and SR respectively. Furthermore, the *SecurityRule* is presented by following the syntax for sensitivity information assignment rules (SIARs) defined in [Fernández-Medina et al., (2006)].

The levels of security (SL) employed in the case study are *confidential*, *secret* and *topSecret*. The user roles (SR) might be Health (including Doctor and Nurse) and *notHealth* (including the Administrative and Maintenance roles). The root of this hierarchy is *HospitalEmployee*. The user categories have not been considered in this example. The *SFact* Admission (stereotype) contains all the individual patient admissions, and can be acceded by users who have secret or *topSecret* security levels and who play a role in Administrative or Health. The *SFact* Admission contains two *SDimensions* (Admission and Patient) (stereotype). Access to these *SBase* (stereotype) hierarchies is established by employing the same procedure that was used with the *SFact*. The *UserProfile* (stereotype) metaclass contains information about all the users who will have access to this secure MD model.

If any instance of the Admission fact class or the Patient dimension class is queried, and if the admission type is a primary diagnosis, then the security role will be Doctor and Admin, otherwise it will be Doctor, Nurse, and Admin. A primary diagnosis(1) is considered to be the most important reason for treatment, while secondary diagnosis(2) completes the view of the patient's condition. This restriction is specified by means of a *SecurityRule*.

The application of the main transformation (see Fig. 4) in the case of this example begins with the application of the *SecureDW2SSchema* relation which transforms the Hospital package into an Admission schema, whilst simultaneously defining the tagged values. The proposal of [Soler et al., (2008)] guarantees that the original tagged values from SMD PIM can be represented in equivalent instances of metaclasses from the extended Relational Metamodel from CWM.

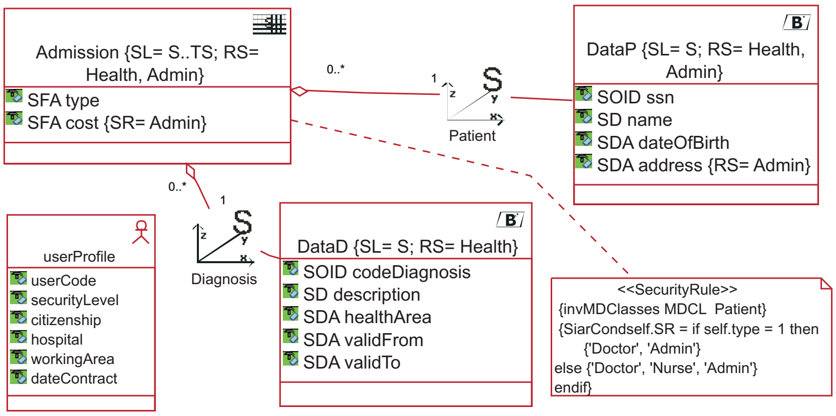

Figure 12: Example of Modeling Secure Multidimensional.

The *UserProfile2RUserProfile* relation guarantees that the Schema contains a table whose columns correspond with the attributes of the *UserProfile* metaclass. Fig. 13 illustrates the result of applying the *SFact2STable* relation. The *SFact* Admission is transformed into the *Admission STable*, which contains as a primary key the column ID of type Number and the SL and SR objects which specify the users who will have access to this type of information.

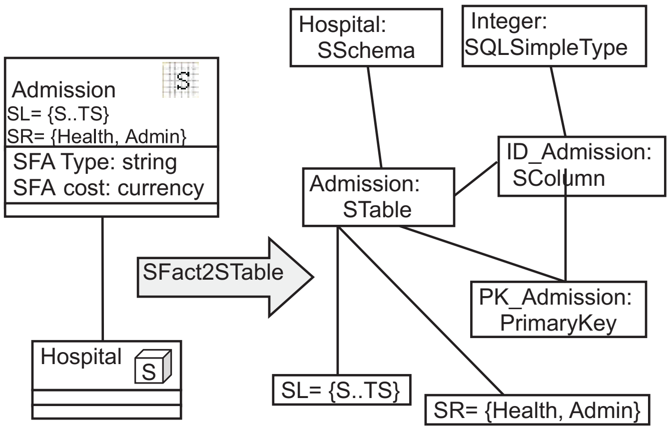

Figure 13: Transforming *SFact* into *STable*.

The *SFactAttribute2SColumn* relation appears in the *where* clause of the

SFact2STable relation. This relation transforms the attributes of the *SFact* Admission into columns of the Admission table. In Fig. 14 we assume that the *SFactAttribute2SColumn* relation is performed, hence the Admission table appears with these columns.

If we continue with the relations that appear in the *where* clause of the *SFact2STable*, we should now apply the *SFactConstraint2SecurityConstraint* relation. Fig. 14 presents a *SecurityRule* which is transformed into an *ARConstraint* associated with the Admission *STable*. Note the representation of the access levels which were previously transformed by means of the *SFact2STable* in the table heading. Let us begin with Fig. 14. Here we clarify that the constraints could be changed depending upon the transformation type between class and table (*OCLExpression*).

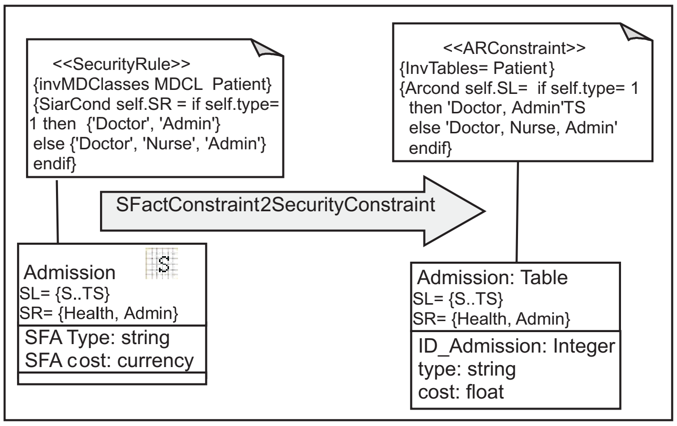

Figure 14: Transforming *SFactConstraint* into *SecurityConstraint*.

According to the *SDimension2STable* relation (see Fig. 8) the *SDimensions* Diagnosis and Patient, along with their security information, are transformed into the *Patient_DataP* and *Diagnosis_DataD* tables respectively. Finally, the Admission, *Patient_DataP* and *Diagnosis_DataD* tables are related according to the *AssocSF_D2FKey* relation.

It is now easy to obtain the code for a secure platform such as Oracle 11g [Jeloka (2007)]. Firstly, the security configuration is set up as Fig. 15 shows. A security policy called "MyPolicy" and a hidden column called "MyLabel", which stores information about labels, are created. Furthermore, since DWs are not modified by users, the user privileges are established to read access.

The security configuration is then established by using security levels and

```
CREATE_POLICY('MyPolicy, 'MyLabel', 'HIDE');
SET_USER_PRIVS('MyPolicy', 'User1', 'READ');
CREATE_LEVEL('MyPolicy, 1, 'LTS', 'topSecret');
CREATE_LEVEL('MyPolicy, 2, 'LS', 'Secret');
CREATE_LEVEL('MyPolicy, 3, 'LC', 'Confidential');
SET_LEVELS('MyPolicy, 'User1', 'LTS', 'LC', 'LS', 'LS');
CREATE_GROUP('MyPolicy, 1, 'GHE', 'HospitalEmployee', NULL);
CREATE_GROUP('MyPolicy, 2, 'GH', 'Health', 'GHE');
CREATE_GROUP('MyPolicy, 3, 'GNH', 'nonHealth', 'GHE');
CREATE_GROUP('MyPolicy, 4, 'GD', 'Doctor', 'GH');
CREATE_GROUP('MyPolicy, 5, 'GN', 'Nurse', 'GH');
CREATE_GROUP('MyPolicy, 6, 'GA', 'Administrative', 'GNH');
CREATE_GROUP('MyPolicy, 7, 'GM', 'Maintenance', 'GNH');
SET_GROUPS('MyPolicy, 'User1', 'GHE, GH, GNH, GD, GN, GA, GM', NULL, 'GHE, GH, GNH,
GD, GN, GA, GM', 'GHE, GH, GNH, GD, GN, GA, GM');
```

Figure 15: Security configuration.

roles. In order to establish the security levels, each security level ("topSecret", "Secret" and "Confidential") has to be created by "*CREATED_LEVEL*", assigning short and long names to the level. Once the levels have been created, "SET_LEVELS" defines the level used as maximum ("topSecret"), as minimal ("Confidential"), by default ("Confidential") and for row access ("Confidential").

The hierarchy of security roles is defined by "*CREATING_GROUP*" statements which, for each role, establish short and long names, and its root's name. Finally, "*SET_GROUP*" establishes sets of roles permitted for read operations (all), write operations (since we only manage read operations this field is null), by default (all) and for row access (all). Although security compartments have not been considered in this example, OLS 11g also support their specification by using the "*CREATE_COMPARTMENT*" and "*SET_COMPARTMENT*" statements.

Once the security policy and security configuration have been defined, it is necessary to create labeling functions which indicate values for the labels depending on the security rules established and then apply this security policy to the corresponding tables.

Fig. 16 defines two labeling functions for the security policy and applies them to the "Admission" table. The first establishes that the "Admission" table can only be accessed by users with a "Secret" (or higher) security level and a "Health" or "Administrative" security role (or their descendant roles such as "Doctor" or "Nurse"). The second implements the "*ARConstraint*" associated with "Admission", in which the value of the "type" attribute determines the security requirements needed to access the information. If the value of "type" is "1", then it can only be accessed by users with a "Secret" (and higher) security level and "Doctor" or "Administrative" security role (or their descendants), but if "type" has a different value, users with the "Nurse" security role are also allowed access. Finally, Fig. 17 creates labeling functions for "*DataD*" and "*DataP*" tables and applies the policy. Both tables can be accessed by users with a "Secret" (or higher) security level and a "*Health*" security role (or its descendants), and "*DataP*" can also be read by "*Administrative*" role.

```
CREATE FUNCTION FunctionAdmission1 (type: Varchar2(20))
   Return LBSCSYS.LABC_LABEL
As MyLabel varchar2(80);
Begin
  MyLabel:=‘Secret::Health,Administrative’;
  Return TO_LBAC_DATA_LABEL (‘MyPolicy’, ‘MyLabel’);
End;
APPLY_TABLE_POLICY (‘MyPolicy’, ‘Admission’, ‘Scheme’, ‘FunctionAdmission1’);
CREATE FUNCTION FunctionAdmission2 (type: Varchar2(20))
   Return LBSCSYS.LABC_LABEL
As MyLabel varchar2(80);
Begin
  If type = ‘1’ then     MyLabel:=‘Secret::Doctor,Administrative’;
  Else     MyLabel:=‘Secret::Doctor,Nurse,Administrative’;
  Endif;
  Return TO_LBAC_DATA_LABEL (‘MyPolicy’, ‘MyLabel’);
End;
APPLY_TABLE_POLICY (‘MyPolicy’, ‘Admission’, ‘Scheme’, ‘FunctionAdmission1’);
```

Figure 16: Labeling functions for Admission table.

```
CREATE FUNCTION FunctionDataD1 (type: Varchar2(20))
   Return LBSCSYS.LABC_LABEL
As MyLabel varchar2(80);
Begin
  MyLabel:=‘Secret::Health;
  Return TO_LBAC_DATA_LABEL (‘MyPolicy’, ‘MyLabel’);
End;
APPLY_TABLE_POLICY (‘MyPolicy’, ‘DataD’, ‘Scheme’, ‘FunctionDataD1’);
CREATE FUNCTION FunctionDataP1 (type: Varchar2(20))
   Return LBSCSYS.LABC_LABEL
As MyLabel varchar2(80);
Begin
  MyLabel:=‘Secret::Health,Administrative;
  Return TO_LBAC_DATA_LABEL (‘MyPolicy’, ‘MyLabel’);
End;
APPLY_TABLE_POLICY (‘MyPolicy’, ‘DataP’, ‘Scheme’, ‘FunctionDataP1’);
```

Figure 17: Labeling functions for *DataP* and *DataD* tables.

5 Applicability and limitations of the Architecture

One of the most interesting advantages of our MDA based proposal is the fact that once the DW has been completely developed, any new security requirement can be easily integrated into the models, providing an excellent schema evolution approach. Our proposal allows designers to modify any detail of an existent model, and generate the data warehouse for the target platform through the model transformations defined between models, and through model to code transformation. The transformations are automatic, but obviously the designer has to make some design decisions, since we deal with models at different abstraction levels, and the semantic power decreases as we go down in our architecture. In fact, high abstraction level models (i.e. requirements or multidimensional) tend to be much more expressive than low level models (i.e. logical or final tools), and some semantic loss is provoked when our top-down transformation is performed, which is dealt by our transformation rules. For instance, our multidimensional model supports several grouping methods for users (security levels, roles and compartments). However, SQL Server Analysis Services only support security roles, so our QVT rules transform high level user groups into platform dependent user groups.

We consider our approach to be applicable and effective in the design and implementation of secure data warehouses but this, however, depends on the target platform we choose to implement our data warehouse. This limitation is clear for any MDA implementation, since the high level models will eventually be implemented through a specific platform which does not support all the concepts defined in these models. However, the fact of modeling security requirements within high level models has an important consequence, which is the better understanding and modeling of the problem, in order to find the best solution, through the best final tool. The chosen final tool will probably not directly support the security solutions for these requirements, but will probably have functionalities to offer ad-hoc solutions to these requirements. We are attempting to reduce this limitation with the integration of reverse engineering capabilities in our architecture. This possibility will allow us to change the target platform and even the logical paradigm of our data warehouse, thanks to the formal definition of the metamodels, and thanks to the direct and reverse QVT transformations.

We believe that the types of security specifications we have included in the architecture are complete and sufficiently flexible to specify the necessary security requirements for data warehouses. In our research we consider only confidentiality, despite the fact that most traditional definitions of security also consider integrity and availability as key components. We do not integrate availability in our data warehouse modeling because this is a requirement that should be solved by the target platform, and because there are no relevant design decisions through which to ensure the availability of the data warehouses. Moreover, protecting the integrity of the information is important, but mainly for information which can be modified or destroyed, while the typical use of the information in data warehouses is that of query. Nonetheless, we believe that confidentiality constraints should be integrated into the data warehouse design, and that relevant design decisions can be made to improve the confidentiality of information stored into a data warehouse. We can therefore consider three technologies that have been widely used to protect information against improper disclosure or modifications. Authentication, access control and audit jointly provide the foundation for information confidentiality. Authentication establishes the identity of one party to another. Access control determines what one party will allow another one to do with regard to the resources and objects mediated by the former. Access control usually requires authentication as a prerequisite. The Audit process gathers the data related to the activities in the system and analyzes them in order to discover security violations or to diagnose their cause. However, authentication is a mechanism that is design-independent and relies more on the company's policies and is, therefore, beyond the scope of this research. Nevertheless, access control and audit have an important design component, and these are the types of security requirements that we integrate into the data warehouse design approach.

6 Conclusion and Future Work

This paper introduces a new framework for the design of secure DWs based on MDA and QVT. The application of QVT transformation rules to the SMD PIM permits the development of different SMD PSMs, thus facilitating the representation of all security and audit requirements, captured during the earlier stages of the DW design, at a logical level. The greatest contribution of this work is that all the security and audit requirements are modeled at a conceptual level from the early stages of development. The use of QVT thus makes them more suitable for the end user, and the time and effort invested in the development of DWs are therefore shortened, and the security rules are closer to the end user and allow him/her to obtain the corresponding code for a relational platform. Secondly, the transition between different models and the final implementation is guaranteed, and we attain interoperability, portability, adaptability and reusability by employing MDA technology.

Our immediate future work consists of improving the early version of the secure Computation Independent Model (CIM) to represent security requirements of DWs presented in [Soler et al., (2008a)] by using the $i*$ framework and defining a secure CIM - secure PIM transformation. Our long term intentions are to study the possibility of implementing a tool which includes the SMD PIM, the SMD PSM, the QVT transformations and the code generation process.

After concluding our complete direct and reverse engineering approach, we shall define a quality framework in order to evaluate the quality of the model transformations. This will allow us to discover the semantic loss which each transformation in our architecture provokes, and therefore permit us to chose the path which best respects the conceptual requirements.

Acknowledgments

This work has been partially supported by the METASIGN project (TIN2004-00779) from the Spanish Ministry of Education and Science, of the Regional Government of Valencia, and by the QUASIMODO and MISTICO projects of the Regional Science and Technology Ministry of Castilla - La Mancha (Spain).

References

[Abelló et al., (2006)] Abelló, A., Samos, J., Saltor, F.: "YAM^2: a multidimensional conceptual model extending UML", Information Systems, 31, 6 (2006), 541-567.
[Basin and Doser (2006)] Basin D., Doser, J.: "Model Driven Security: From UML Models to Access Control Infrastructures", ACM Transactions on Software Engineering and Methodology, 15, 1 (2006), 39-91.
[Best et al., (2007)] Best, B., Jürjens, J., Nuseibeh, B.: "Model-Based Security Engineering of Distributed Information Systems Using UMLsec", 29th International Conference on Software Engineering, Minneapolis, USA (2007), 581-590.
[Binh et al., (2000)] Binh, N. T., Tjoa, A. M., Wagner, R.: "An Object Oriented Multidimensional Data Model for OLAP. Web-Age Information Management", 1st International Conference of Web-Age Information Management (WAIM'00), Shanghai, China (2000), 69-82.
[Blanco et al., (2009)] Blanco, C., Garca-Rodrguez de Guzmn, I., Fernández-Medina, E., Trujillo, J., Piattini, M.: "Applying QVT in order to implement Secure Data Warehouses in SQL Server Analysis Services", Journal of Research and Practice in Information Technology, 41, 2 (2009), 119-138.
[Borland (2007)] Eclipse, http://www.eclipse.org
[Burt et al., (2003)] Burt, C. C., Bryant, B. R., Raje, R. R., Olson, A. M., Auguston, M.: "Model Driven Security: Unification of Authorization Models for Fine-Grain Access Control", Proc. 7th IEEE Int. Enterprise Distributed Object Computing Conf., Brisbane, Australia (2003), 159-171.
[Canfora and Penta (2007)] Canfora, G., Penta, M. D.: "New Frontiers of Reverse Engineering", 29th International Conference on Software Engineering, Workshop on the Future of Software Engineering, FOSE'07, Minneapolis, USA (2007), 326-341.
[Devanbu and Stubblebine (2000)] Devanbu P., Stubblebine, S.: "Software Engineering for Security: a Roadmap", Proc. Conf. on The Future of Software Eng., Limerick, Ireland (2000), 227-239.
[Dhillon and Backhouse (2000)] Dhillon G., Backhouse, J.: "Information Systems Security Management in the New Millenium", Communications of the ACM, 43, 7 (2000), 125-128.
[Fernández-Medina and Piattini (2004)] Fernández-Medina, E., Piattini, M.: "Extending OCL for Secure Database Design", Proc. 7th Int. Conference on the Unified Modeling Languages and Applications, Lisboa, Potugal (2004), 380-394.
[Fernández-Medina et al., (2006)] Fernández-Medina, E., Trujillo, J., Villarroel, R., Piattini, M.: "Access Control and Audit Model for the Multidimensional Modeling of Data Warehouses", Decision Support Systems, 42, 3 (2006), 1270-1289.

[Fernández-Medina et al., (2006a)] Fernández-Medina, E., Trujillo, J., Villarroel, R., Piattini M.: "Developing Secure Data Warehouses with a UML Extension", Information Systems, 32, 6 (2007), 826-856.

[Giorgini et al., (2006)] Giorgini, P., Mouratidis, H., Zannone, Z.: "Modelling Security and Trust with Secure Tropos", Integrating Security and Software Engineering: Advances and Future Visions, Idea Group Publishing (2006).

[Golfarelli et al., (1998)] Golfarelli, M., Maio, D., Rizzi, S.: "The Dimensional Fact Model: A Conceptual Model for Data Warehouses", International Journal of Cooperative Information Systems, 7, (2-3) (1998), 215-247.

[Grabosky (2007)] Grabosky, P.: "Security in the 21st Century", Security Journal, 20, 1 (2007), 9-11.

[Hafner et al., (2006)] Hafner, M., Alam, M., Breu, R.: "Towards a MOF/QVT-Based Domain Architecture for Model Driven Security", Proc. Model Driven Engineering Languages and Systems, Genova, Italy (2004), 230-244.

[Jajodia and Wijesekera (2001)] Jajodia, S., Wijesekera, D.: "Security in Federated Database Systems", Information Security Technical Report, 6, 2 (2001), 69-79.

[Jeloka (2007)] Jeloka, S.: "Oracle Label Security Administrator's Guide, 11g" (2007).

[Jürjens (2002)] Jürjens, J.: "UMLsec: Extending UML for secure systems development", 7th Int. Conference on the Unified Modeling Languages and Applications (UML'02), Dresden, Germany (2002), 412-425.

[Jürjens (2004)] Jürjens, J.: "Secure Systems Development with UML", Springer Academic Publishers (2004).

[Jürjens and Shabalin (2007)] Jürjens, J., Shabalin, P.: "Tools for Secure Systems Development with UML", International Journal on Software Tools for Technology Transfer, 9, (5-6) (2007), 527-544.

[Jürjens et al., (2008)] Jürjens, J., Schreck, J., Bartmann, P.: "Model-based Security Analysis for Mobile Communications", 30th International Conference on Software Engineering, Leipzig, Germany (2008), 683-692.

[Jürjens (2009)] Jürjens, J.: "A Domain-specific Language for Cryptographic Protocols based on Streams", Journal of Logic and Algebraic Programming, Special issue on Streams and Algebra, 78, 2 (2009), 54-73.

[Katic et al., (1998)] Katic, N., Quirchmayr, G., Schiefer, J., Stolba, M., Tjoa, A. M.: "A Prototype Model for DW Security Based on Metadata", Proc. 9th Int. Workshop on DB and Expert Systems Applications, Vienna, Austria (1998), 300-308.

[Khusidman and Ulrich (2007)] Khusidman, V., Ulrich, W.: "Architecture-Driven Modernization: Transforming the Enterprise", DRAFT V.5, http://www.omg.org/docs/admtf/07-12-01.pdf, OMG: 7, (2007).

[Kimball and Ross (2002)] Kimball, R., Ross, M.: "The Data Warehousing Toolkit", Second Edition; John Wiley & Sons, Inc./ New York (2002).

[Kirkgöze et al., (1997)] Kirkgöze, R., Katic, N., Stolda, N., Tjoa, A. M.: "A Security Concept for OLAP", Proc. 8th Int. Workshop on Database and Expert System Applications, Toulouse, France (1997), 619-626.

[Lang and Schreiner (2004)] Lang, U., Schreiner, R.: "OpenPMF: a Model-Driven Security Framework for Distributed Systems", Proc. of the Information Security Solutions Europe, Berlin, Germany (2004).

[Lodderstedt et al., (2002)] Lodderstedt, T., Basin, D., Doser, J.: "SecureUML: A UML-Based Modeling Language for Model-Driven Security", Proc. 5th Int. Conf. of UML, Dresden, Germany (2002), 426-441.

[Luján et al., (2006)] Luján-Mora, S., Trujillo, J., Song, I.-Y.: "A UML profile for multidimensional modeling in data warehouses", DKE (Data & Knowledge Engineering), 59, 3 (2006), 725-769.

[Mazón et al., (2007)] Mazón, J.-N., Trujillo, J.: "A Model Driven Modernization Approach for Automatically Deriving Multidimensional Models in Data Warehouses", 26th International Conference on Conceptual Modeling (ER'07), Auckland, New Zealand (2007), 56-71.

[Mazón et al., (2008)] Mazón, J.-N., Trujillo, J., Serrano, M., Piattini, M.: "A MDA Approach for the Development of Data Warehouses", Decision Support Systems, 45, 1 (2008), 41-58.

[McGraw (1999)] McGraw, G.: "Software Assurance for Security", IEEE Computer, 32, 4 (1999), 103-105.

[McGraw (2002)] McGraw, G.: "Building Secure Software: Better than Protecting Bad Software", IEEE Software, 19, 6 (2002), 57-59.

[Miller and Mukerji (2003)] Miller J., Mukerji, J.: "MDA Guide Version 1.0.1", 2003.

[Nakamura et al., (2005)] Nakamura, Y., Tatsubori, M., Imamura, T., Ono, K.: "Model-driven security based on a Web services security architecture", Proc. IEEE Int. Conference on Services Computing, Orlando, USA (2005), 7-15.

[OMG (2003)] OMG,"Common Warehouse Metamodel Specification 1.1". http://www.omg.org/technology/documents/formal/ cwm.htm (2003).

[OMG (2005)] OMG, "MOF 2.0 QVT Final Adopted Specification", (2005).

[OMG (2006)] OMG, "ADM Glossary of Definitions and Terms", http://adm.omg.org/ADM_Glossary_Spreadsheet_pdf.pdf, OMG: 34, (2006).

[Paim and Castro (2003)] Paim F. R. S., Castro, J.: "DWARF: An Approach for Requirements Definition and Management of Data Warehouse Systems", IEEE International Conference on Requirements Engineering, Monterey Bay, California, USA (2003), 75-84.

[Prat et al., (2006)] Prat, N., Akoka, J., Comyn-Wattiau, I.: "A UML-based data warehouse design method", Decision Support Systems, 42, 3 (2006), 1449-1473.

[Priebe and Pernul (2001)] Priebe T., Pernul, G.: "A Pragmatic Approach to Conceptual Modeling of OLAP Security", Proc. 20th Int. Conference on Conceptual Modeling (ER'01), Yokohama, Japan (2001), 311-324.

[Rosenthal et al., (2000)] Rosenthal A., Sciore, E.: "View Security as the Basic for DW Security", Proc. 2nd Workshop Design and Management of DW, Stockholm, Sweden (2000), 8.1-8.8.

[Saltor et al., (2002)] Saltor, F., Oliva, M., Abell, A., Samos, J.: "Building Secure Data Warehouse Schemas from Federated Information Systems", in Heterogeneous Inf. Exchange and Organizational Hubs., D. T. Bestougeff, Kluwer Academic (2002).

[Sapia et al., (1998)] Sapia, C., Blaschka, M., Höfling G., Dinter, B.: Extending the E/R Model for the Multidimensional Paradigm, 1st Int. Conference on Conceptual Modeling (ER'98), Singapore (1998), 105-116.

[Shoshani (1997)] Shoshani, A.: "OLAP and Statistical Databases: Similarities and Differences", Proc. of the 16th ACM Symposium on Principles of Database Systems (PODS'97), Tucson, Arizona (1997), 185-196.

[Simitsis and Vassiliadis (2008)] Simitsis, A., Vassiliadis, P.: "A method for the mapping of conceptual designs to logical blueprints for ETL processes", Decision Support Systems, 45, 1 (2008), 22-40.

[Sivanandam and Karpagam (2004)] Sivanandam S. N., Karpagam, G. R.: "A Novel Approach for Implementing Security Services", Academic Open Internet Journal, 13, 13 (2004).

[SmartQVT (2007)] SmartQVT, http://smartqvt.elibel.tm.fr, (2007).

[Soler et al., (2007)] Soler, E., Trujillo, J., Fernández-Medina, E., Piattini, M.: "A set of QVT relations to transform PIM to PSM in the Design of Secure Data Warehouses", Second International Conference on Availability, Reliability and Security (ARES'07), Vienna, Austria (2007), 644-654.

[Soler et al., (2007a)] Soler, E., Trujillo, J., Fernández-Medina, E., Piattini, M.: "Application of QVT for the Development of Secure Data Warehouses: A case study", Second International Conference on Availability, Reliability and Security (ARES'07), Vienna, Austria (2007), 829-836.

[Soler et al., (2007b)] Soler, E., Trujillo, J., Fernández-Medina, E., Piattini, M.: "A Framework for the Development of Secure Data Warehouses based on MDA and

QVT", Second International Conference on Availability, Reliability and Security (ARES'07), Vienna, Austria (2007), 294-300.

[Soler et al., (2008)] Soler, E., Trujillo, J., Fernández-Medina, E., Piattini, M.: "Building a secure star schema in data warehouses by an extension of the Relational Package from CWM", Computer Standards & Interfaces, 30, 6 (2008), 341-350.

[Soler et al., (2008a)] Soler, E., Stefanov, V., Mazón, J.-N., Trujillo, J., Fernández-Medina, E., Piattini, M.: "Modelado de Requisitos de Seguridad para Almacenes de Datos", XI Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS'08), Pernambuco, Brasil (2008), 281-294.

[Thuraisingham (1994)] Thuraisingham, B.: " Security Issues for Federated Database Systems", Computer and Security, 13, 6 (1994), 509-525.

[Thuraisingham et al., (2007)] Thuraisingham, B., Kantarcioglu, M., Iyer, S.: "Extended RBAC-based design and implementation for a secure data warehouse", International Journal of Business Intelligence and Data Mining, 2, 4 (2007), 367-382.

[Trujillo and Luján-Mora (2003)] Trujillo, J., Luján-Mora, M.: "A UML Based Approach for Modeling ETL Processes in Data Warehouses", Proc. 22th Int. Conference on Conceptual Modeling (ER'03), Chicago, Illinois, USA (2003), 13-16.

[Tryfona et al., (1999)] Tryfona, N., Busborg, F., Borch, J. G.: "starER: A Conceptual Model for Data Warehouse Design", Second International Workshop on Data Warehousing and OLAP (DOLAP'99), Missouri, USA (1999), 3-8.

[Vela et al., (2006)] Vela. B., Fernández-Medina, E., Marcos, E., Piattini, M.: "Model Driven Development of Secure XML Databases", SIGMOD Record, 35, 3 (2006), 22-27.

[Villarroel et al., (2006)] Villarroel, V., Fernández-Medina, E., Piattini, M.: "A UML 2.0/OCL Extension for Designing Secure Data Warehouses", Journal of Research and Practice in Information Technology, 38, 1 (2006), 31-43.