The Computer Journal Advance Access published August 11, 2014

doi:10.1093/comjnl/bxu070

Modernizing Secure OLAP Applications with a Model-Driven Approach

CARLOS BLANCO[1]*, EDUARDO FERNÁNDEZ-MEDINA[2] AND JUAN TRUJILLO[3]

[1]*GSyA Research Group, Department of Computer Science and Electronics, Faculty of Sciences, University of Cantabria, Av. de los Castros s/n, 39071 Santander, Spain*
[2]*GSyA Research Group, Institute of Information Technologies and Systems, Department of Information Technologies and Systems, Escuela Superior de Informática, University of Castilla-La Mancha, Paseo de la Universidad, 4, 13071 Ciudad Real, Spain*
[3]*Lucentia Research Group, Department of Information Languages and Systems, Facultad de Informática, University of Alicante, San Vicente s/n, 03690 Alicante, Spain*
**Corresponding author: carlos.blanco@unican.es*

The majority of the organizations store their historical business information in data warehouses which are queried to make strategic decisions by using online analytical processing (OLAP) tools. This information has to be correctly assured against unauthorized accesses, but nevertheless there are a great amount of legacy OLAP applications that have been developed without considering security aspects or these have been incorporated once the system was implemented. This work defines a reverse engineering process that allows us to obtain the conceptual model corresponding to a legacy OLAP application, and also analyses and represents the security aspects that could have established. This process has been aligned with a model-driven architecture for developing secure OLAP applications by defining the transformations needed to automatically apply it. Once the conceptual model has been extracted, it can be easily modified and improved with security, and automatically transformed to generate the new implementation.

Keywords: data warehouses; OLAP; model-driven development; modernization; reverse engineering; security; confidentiality

Received 30 December 2013; revised 12 June 2014
Handling editor: David Rosado

1. INTRODUCTION

The information stored in data warehouses (DWs) is organized by following a multidimensional model which improves its further analysis, usually carried out by using online analytical processing (OLAP) tools. In this way, the information is organized in facts and measures which can be analyzed by different subjects called dimensions and in different detail levels.

This information has a strategic value for the organization for making strategic decisions and furthermore, it is used to include private data of individuals. It has to be assured, especially focusing on information confidentiality because final users solely will query DWs information [1–3]. The security aspects have been traditionally added to the final solution once the system has been built. Nevertheless, in order to improve the quality and security of any information system, it is needed to identify and incorporate security constraints from the beginning and consider them in all stages of the development process [4, 5].

On the other hand, DWs and OLAP applications can be developed by following a model-driven approach by using different models in the development process, separating the system functionality from details of specific technologies and implementations. This approach allows us to define transformations which are able to automatically generate the intermediate models and the final implementation, saving then on development costs and efforts. The different development stages of a DW can be aligned with the different models of a model-driven architecture [6]: business models for systems requirements (computational independent model, CIM); conceptual models (platform-independent model, PIM); and logical models focused on a concrete technology (platform-specific model, PSM).

There are contributions on the development of secure DWs and OLAP that propose interesting ideas, but they do not deal with reverse engineering that is very useful in the development of information systems, since it allows us to analyze legacy systems and to obtain their models at a higher abstraction level. Reverse engineering provides us a mechanism for re-documentation, model migration, restructuring, maintenance or improvement, tentative requirements, integration, conversion of legacy data, etc. The model obtained by applying a reverse engineering process is easier to understand than the implementation of the legacy system and can be used into a modernization process in which we can modify systems characteristics into the models, whereas modifying the implementation. For instance, new aspects not considered into the initial development, such as security, could be added [7, 8]. Although data reverse engineering field has been widely studied in literature [9–12], there is little research on re-engineering of DWs and OLAP applications and there are not any approaches that consider security and apply a model-driven approach to automate the process.

This paper defines a reverse engineering process for legacy OLAP applications that considers both structural and security aspects. This proposal has been included in a previously defined architecture for developing secure DWs and OLAP applications [13]. Then, the contribution of this paper is the definition of this reverse engineering process that is composed of two stages: the generation of logical models from legacy OLAP implementations (considering SQL Server Analysis Services (SSAS) as the source OLAP tool); and the generation of the conceptual model corresponding to the logical model. In this way, legacy OLAP applications can be re-documented and improved with security by modifying the conceptual model obtained. Then, the improved system can be automatically re-implemented or migrated to other platforms by using our model-driven architecture. As a contribution of this paper, the transformations needed for automatically obtain conceptual models from legacy OLAP applications have been implemented.

The rest of this paper is organized as follows: Section 2 will present related work; Section 3 will briefly describe our previously defined architecture for developing secure DWs and OLAP applications; Section 4 will present the reverse engineering process proposed in this paper; Section 5 will show an application example to validate our proposal; and Section 6 will finally present our conclusions and future work.

2. RELATED WORK

This section presents the related work organized according to: (i) proposals for developing secure information systems; (ii) works focused on secure DWs and OLAP; and (iii) contributions that deal with reverse engineering.

2.1. Secure information systems

There are relevant contributions concerning with a complete secure development of information systems.

UMLsec [14] defines and evaluates security specifications using formal semantics (labels, stereotypes, etc.). It is mainly focused on access control policies and in the specification of confidentiality and integrity requirements. UMsec uses the majority of UML diagrams and has been recently adapted to UML2 [15].

Model-driven security (MDS) [16] applies the model-driven approach to include security properties in high-level system models and to automatically generate secure system architectures. For modeling the system they propose an UML extension called SecureUML [17] that permits the inclusion of access control aspects into the models. MDS has been applied to UMLsec [18] defining three abstraction levels (requirements, modeling and implementation) and providing tools for code generation, reverse engineering, verification and configuration [19]. Furthermore, some works are developing transformations between SecureUML and UMLsec [20].

TROPOS is a methodology for software development based on the intentional goals of agents which provides an extension called Secure TROPOS [21–23]. They include security concepts (constraints, secure goals, delegation of permissions, etc.) and activities (trust of permission modeling, delegation of permission modelling, etc.), presenting a framework that allows us to model and analyze security requirements within functional requirement. Furthermore, they provide a CASE tool (ST-Tool) that supports requirements' analysis and verification.

Mokum [24] which is an active object-oriented knowledge-based system for modeling that permits the specification of security and integrity constraints, and the automatic code generation.

Some works propose processes based on security models and standards for building security systems. For instance, the process Process to Support Software Security (PSSS) [25] which is based on the activities derived from SSE-CMM, ISO/IEC 15408, ISO/IEC 27002 and OCTAVE.

2.2. Secure DWs and OLAP

Although the contributions above presented are relevant for the development of secure information systems they are not specifically focused on DWs and their specific security problems.

A typical DWs architecture is composed of several layers that present specific security concerns: heterogeneous data sources; ETL (extraction/transformation/load) processes which extract and transform data from these data sources and load the information into the DW; the repository of the DW, where data are stored; and DBMS and OLAP tools which analyze data. Since DWs mainly dealt with read operations over sensitive information used for the decision-making, the main security

problem related with DWs is information confidentiality and should be taken into account in all layers and operations of the DW [3].

Concerning with a complete secure DWs development, we solely found the methodology of Priebe and Pernul [1] in which the authors analyze security requirements and their implementation into commercial tools by hiding multidimensional elements such as cubes, measures, slices and levels. They extend their proposal with a DWs representation at conceptual level with ADAPTed UML, but do not establish the connection between models in order to allow automatic transformations.

On the other hand, there are several works focused on the secure modeling for DWs at certain abstraction levels. At business level there are proposals based on ontologies, business process, UML, etc. but solely Paim and Castro [26] include security requirements, however, they do not offer any formal metamodel.

At the conceptual level there are interesting works for modeling DWs considering their special characteristics by using extensions of the ER model, UML or an own notation, but they do not include security capabilities [27–31]. The conceptual modeling of security issues is solely considered by the AdaptedUML of Priebe and Pernul [1].

Traditionally, the multidimensional modeling at logical level has depended of the DBMS used and, in this way can be mainly classified in online analytical processing over a relational (ROLAP), multidimensional (MOLAP) and hybrid (HOLAP) approaches. There are many modeling proposals which do not consider security but solely CWM [32] provides a formal metamodel with relational and multidimensional packages.

Final tools have also to consider security constraints in order to avoid unauthorized accesses. Research efforts have been traditionally carried out in this way but focused on the final stage of development without including security issues in the whole development process. For instance, Kirkgoze *et al.* propose to define a virtual cube for each subject [33], or Weippl *et al.* which define an access control model for DWs and OLAP which allows us to define the OLAP operations authorized for each user [2]. On the other hand, the inference problem is another important research branch [3] that is being studied proposing query control systems [34, 35].

2.3. Reverse engineering

The concept of reverse engineering applied to DWs has the objective of understand the legacy system and generate specifications of the DW's structure. It allows us to re-document, improve or migrate legacy DWs managed by enterprise applications [10, 11, 36].

There are interesting contributions that apply reverse engineering concepts and procedures to legacy data bases and DWs to obtain models, migrate, etc. Some works propose algorithms for extracting and modeling metadata from legacy data bases or DWs [37–39]. Other works are focused on design models such as [12] that extract the design model from the implementation, [40] that deals with model migration, [41] that integrates data bases based in different models or [42, 43] that build applications guided by the information extracted from legacy data bases.

3. OUR MODEL-DRIVEN ARCHITECTURE FOR SECURE OLAP APPLICATIONS

This section briefly describes the different layers of our architecture for developing secure DWs and OLAP applications. This architecture has been aligned with an MDA architecture [13] providing security models at different abstraction levels (CIM, PIM, PSM) and automatic transformations between models (Fig. 1). We have defined in previous works security models for each development stage of the secure DW and OLAP application and, in recent works, we have applied the complete proposal to an application example [44].

For the requirement stage (business level (CIM)), a UML profile allows us to define security requirements associated with the DW. This profile has been defined [45] based on the i* framework [46]. Then, for the conceptual modeling stage (PIM), another UML profile called SECDW [13] allows us to achieve the multidimensional modeling of the structural aspects of the DW (facts, dimensions, bases, hierarchies, attributes, etc.) within its security constraints defined by using an access control and audit model focused on DW confidentiality [47].

SECDW is shown in Fig. 2. It allows us to define DW's structure with secure classes for facts, dimensions, bases (aggregation levels) and properties. The security configuration of the DW can be defined by using a classification of subjects and objects into a hierarchy of security roles, a sequence of security levels and a set of security compartments. These elements will be used to establish the security privileges needed to access each structural element (secure information class). Furthermore, SECDW permits to define more complex security constraints by using different kinds of security rules (for sensitive information, information combinations, authorizations and auditing).

In order to achieve the multidimensional modeling at the logical level (PSM) a metamodel called SECMDDW focused on the OLAP technology has been defined [48]. It extends the Common Warehouse Metamodel [32] to permit the inclusion of security constraints and incorporates all the details needed for a further implementation of the system into a OLAP tool.

Figure 3 shows SECMDDW metamodel. The main class is the DW's schema that has attached all the elements of the

FIGURE 1. Model-driven architecture for secure OLAP applications.

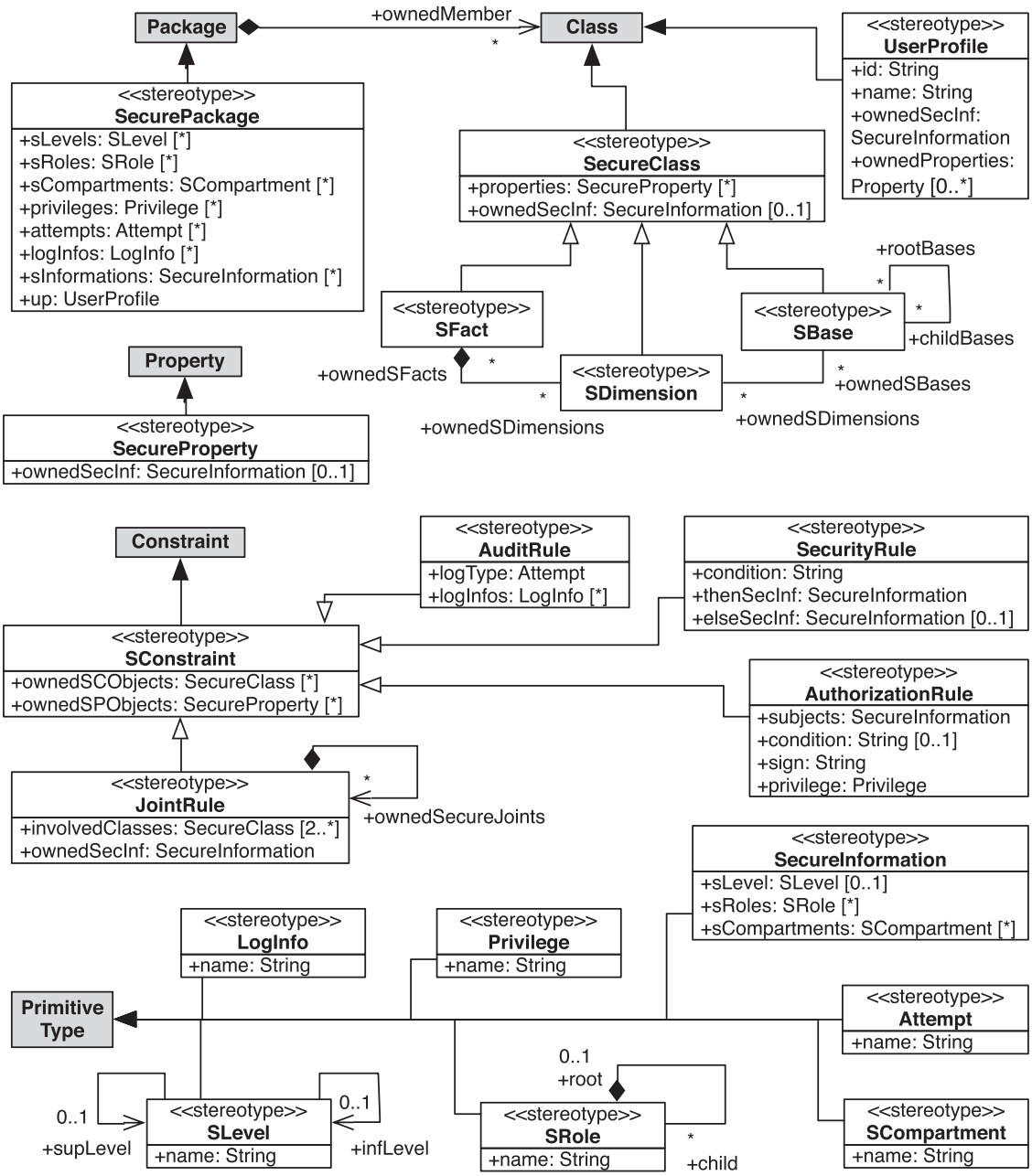

FIGURE 2. Conceptual modeling (SECDW).

DW: security configuration, cubes and dimensions. The security configuration is defined by using an role-based access control (RBAC) policy. It allows us to establish a hierarchy of user roles that will be used to classify users and objects and by the security constraints. Cubes represent facts at the logical level within their measures and security constraints attached to cubes or measures. Dimensions are the different perspectives in which the information can be classified. At logical level these dimensions are defined within their attributes, classification hierarchies and aggregation levels, and security constraints associated with dimensions or attributes.

The development process of the secure OLAP application has been also automated by defining sets of transformations that allows us to generate the eventually implementation from models [48, 49]. The code generation has two stages: the transformation between conceptual and logical models (defining query view transformation specification (QVT) rules) and from the logical model toward the final secure implementation of

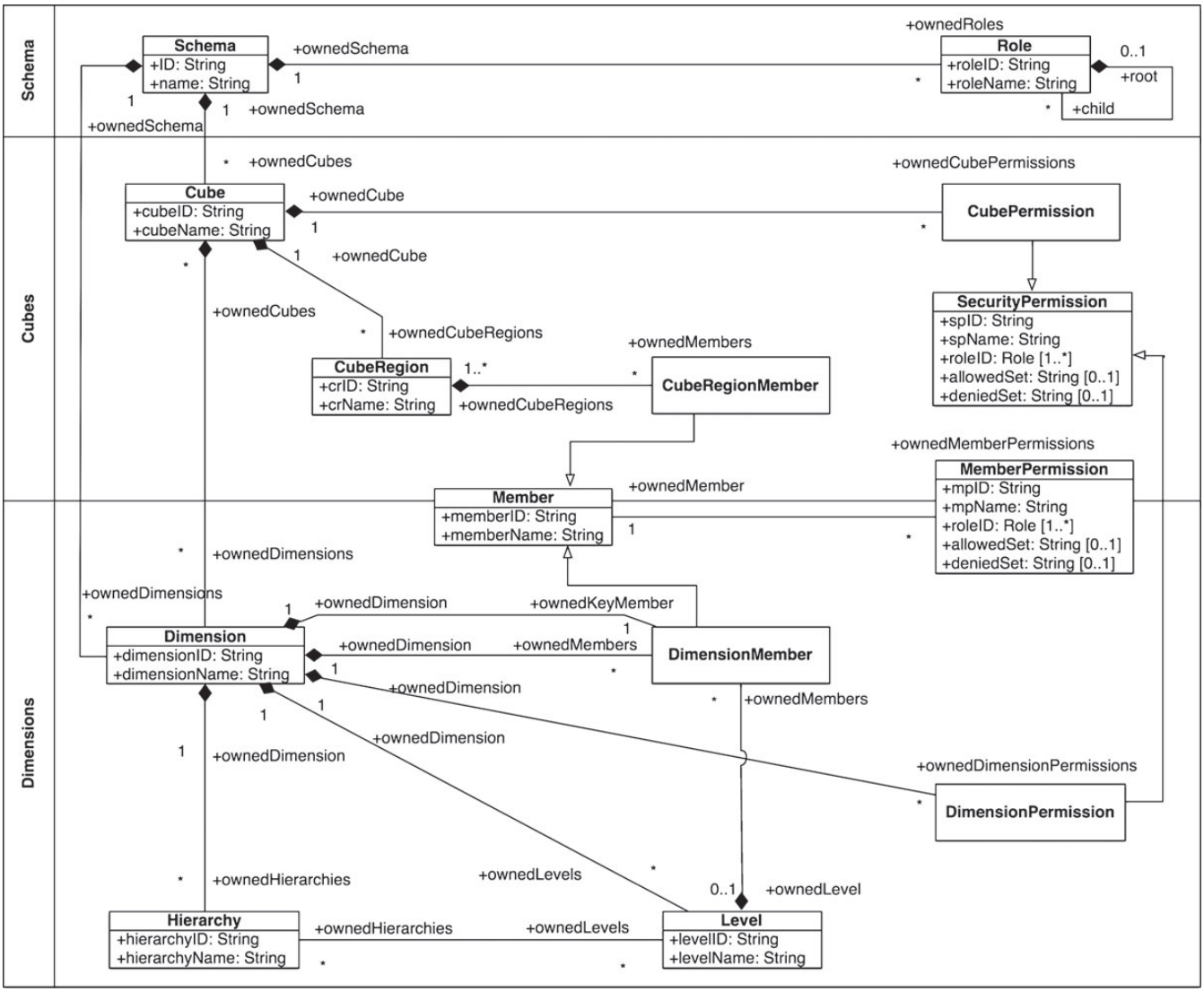

FIGURE 3. Logical modeling for OLAP (SECMDDW).

the OLAP application (defining MOFScript rules). In this way, we have consider SSAS as the target OLAP tool.

In this paper, we complete this architecture by defining the transformations needed to automate a reverse engineering process that obtains a conceptual model from a legacy implementation. This process is composed of the generation of the logical model corresponding to the legacy implementation and the transformation of the logical model to the conceptual model. This process is further detailed in Section 4.

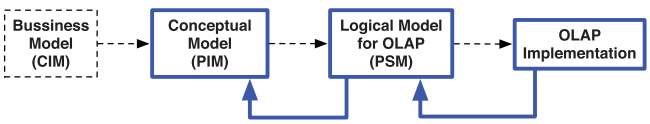

FIGURE 4. A reverse engineering process for legacy OLAP applications.

4. A REVERSE ENGINEERING PROCESS FOR LEGACY OLAP APPLICATIONS

The reverse engineering process presented in this paper starts from the legacy implementation of the OLAP application and obtains its logical and conceptual models (Fig. 4). These models re-document the system and permit the migration toward other platforms or technologies by following our model driven approach.

It has been integrated with our model-driven architecture for developing secure DWs and OLAP applications (defined in previous works). That is, it uses the models that have

been defined for the different development stages and includes the transformations needed to obtain the conceptual model corresponding to the legacy OLAP implementation. This process is composed of two stages: (1) the transformation of the legacy OLAP implementation into a logical model and (2) the transformation of the logical model into a conceptual model.

As can be observed, our process does not finish into the requirements level (CIM). To completely automate the transformations from or toward the requirements level is complex, since there is a considerable semantic gap between these abstraction levels and few works deal with this transformation in an automatic way. Nevertheless, in order to complete our architecture we plan to address the connexion between these levels in future works. In previous published works, we dealt with the transformation from CIM to PIM, but describing a process that does not provide an automatic solution [50].

4.1. Obtaining logical models

First, the legacy OLAP implementation has to be analyzed in order to detect the structural and security aspects of the OLAP application and to generate its corresponding logical model.

There are a great amount of OLAP platforms such as the solutions identified in Gartners Magic Quadrant for business intelligence and analytic platforms (www.gartner.com): Microsoft, Oracle, IBM, Microstrategy, Pentaho, Jaspersoft, etc. In this work, we have considered that the legacy OLAP application has been implemented into one of the platforms identified in the leaders quadrant, Microsoft SSAS.

The majority of OLAP tools represent the structural and security information as metainformation stored in XML files: cubes, dimensions, attributes, measures, hierarchies, roles, security permissions, etc. Nevertheless, each OLAP tool uses its own syntax and thus has to be specifically processed. In this case, SSAS organizes OLAP metainformation in three kind of files:

> *Role files* that represent the access control policy by using an RBAC strategy. These roles are used in the definition of security permissions.
> *Cube files* cube, measure groups, related dimensions, classification hierarchies and security permissions established over the cube or its measures.
> *Dimension files* dimension, attributes, hierarchies, aggregation levels and security permissions over the dimension or its attributes.

For this stage, we have implemented a parser that receives these three kind of XML files as input, processes them by using XPath expressions and generates its corresponding logical model according to our logical metamodel. This transformation also processes the information in order to improve the target logical model, for instance by grouping information that can be represented in the model together.

4.2. Obtaining conceptual models

In a second stage, the logical model previously generated is transformed into a conceptual model. In order to achieve this goal a set of QVT transformations has been defined and integrated into our MDA architecture.

Figure 5 shows the main elements of the transformation defined, called SECMDDW2SECDW. It is composed for several QVT relations grouped by its purpose into relations for roles, cubes and dimensions.

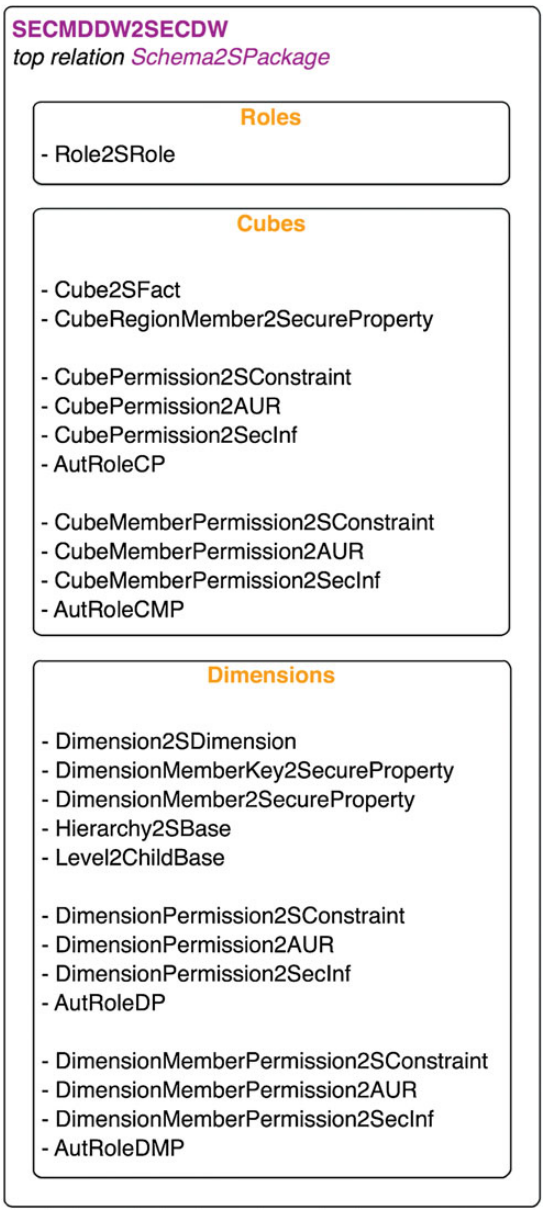

FIGURE 5. Transformation overview.

4.2.1. *Roles*

The first group of transformations generates the security configuration structure that will be used to classify subjects and objects according to their security privileges. The relation 'Role2SRole' establishes the security configuration by following a RBAC policy by generating the hierarchy of security roles needed to represent the security configuration in the conceptual model.

4.2.2. *Cubes*

Cubes are transformed into facts in the conceptual model, processing also the remainder structural aspects such as cube measures and the dimensions that allow us to classify the information by different perspectives. Once these structural elements have been created, the security constraints defined over cubes or its measures are transformed into the conceptual model as security rules.

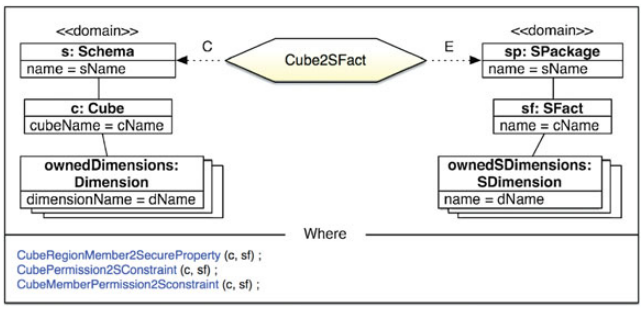

FIGURE 6. Cube2SFact relation.

The relation Cube2SFact (Fig. 6) is the top relation that firstly defines the structural aspects generating facts and dimensions and then, it serves from auxiliary relations to process the security constraints.

Next, several of these auxiliary relations are described as examples. The relation 'CubePermission2SConstraint' (Fig. 7) processes security permissions defined over cubes (CubePermission) and transforms them into security constraints attached to facts in the conceptual model.

In this case, the source security information can be transformed into four different security elements that represent different kinds of security rules. Depending of the information included in the source element will be better to generate one or other kind of security rule. The heuristic needed to automatically choose the best target element has been implemented in the where clause of the relation and depending on the source characteristics calls the auxiliary relations needed to create the target element.

One of these cases is the generation of several authorization rules (AUR) in the conceptual model to represent a cube permission. The auxiliary rule 'CubePermissions2AUR' (Fig. 8) is the rule dedicated to create these AUR. It generates an authorization rule based on the parameters indicated in its invocation (fact name, permissions involved, if the authorization is a positive or a negative permission, etc.). To complete the

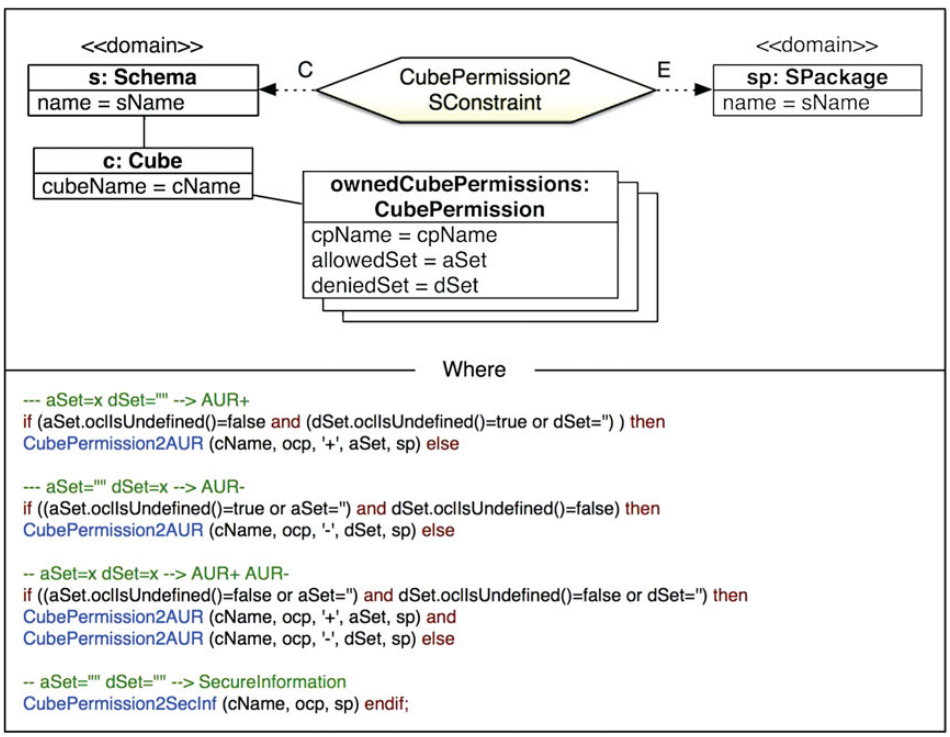

FIGURE 7. CubePermission2SConstraint relation.

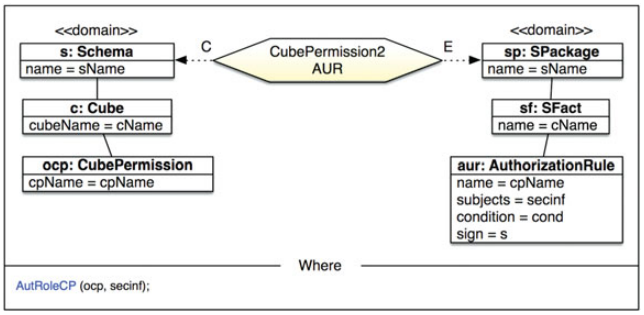

FIGURE 8. CubePermission2AUR relation.

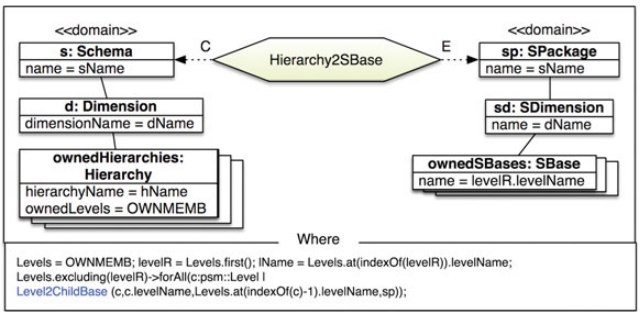

FIGURE 9. Hierarchy2SBase relation.

AUR, the subjects (roles) affected by the authorization are next added by other auxiliary rule called 'AutRoleCP'.

4.2.3. Dimensions

The dimensions defined in the logical model associated with cubes, are transformed into dimensions associated with fact classes, generating also the remainder structural and security elements needed. The top relation called 'Dimension2SDimension' serves from auxiliary relations to create these elements into the target model (conceptual model). Next, some examples of these relations are provided.

The first example is related with an structural aspect, the generation of classification hierarchies. Dimensions used to include hierarchies to classify the information in different aggregation levels (that represent different detail levels). The relation Hierarchy2SBase (shown in Fig. 9) analyses the classification hierarchies (Hierarchy) associated with each dimension (ownedHierarchies), and for each one creates in the conceptual model a base class (SBase) associated with the dimension implied (SDimension). It represents a classification hierarchy composed of one aggregation level and then, the Level2ChilBase is called in order to add the remainder aggregation levels of the hierarchy.

Once the structural elements have been processed, several relations analyze the security constraints defined over dimensions and their attributes. Next, the relation that processes fine grain security constraints defined over attributes is shown in Fig. 10.

The fine grain security constraints (MemberPermission) are analyzed by the relation DimensionMemberPermission2SConstraint. Depending on the information included in the source element it decides which kind of security rule to create into the target model (authorization rules or secure information). The best target element is automatically chosen with the heuristic included in the where clause of the relation and then, auxiliary relations are invoked to generate the security rules needed.

The relation 'DimensionMemberPermission2SecInf' transforms these member permissions into secure information elements attached to the dimension attributes affected by the security constraint. This secure information indicates which roles can access to the attributes. On the other hand, authorization rules are created by the relation 'DimensionMemberPermission2AUR' that receives all the parameters needed to define the authorization (subjects, objects, authorization sign, conditions, etc.).

4.2.4. Heuristics

Logical models are more concrete than conceptual models, for instance, our logical model is focused on OLAP and manages concepts related with this concrete technology. On the other hand, conceptual models are richer in expressiveness and contain information independent of the platform used.

When we define a reverse engineering process, we have to take into account that the logical model does not include enough information 'independent of the platform' for rebuild all the aspects of the conceptual model. Then, in some occasions there are different choices for transforming certain elements and we have to decide the best one. In order to automate the entire process these special situations have been analyzed and some heuristics have been implemented inside the transformations.

Next, the heuristics defined and implemented in this proposal are described:

Security configuration. Our logical model represents this information by using a RBAC policy, whereas our conceptual model is richer and allows us to define security roles, levels and compartments. Nevertheless, security levels and compartments cannot be rebuild from the source information (the logical model) since it solely has security roles defined. In our proposal, we have solved this semantic gap by transforming all the source security configuration roles into a equivalent security role hierarchy built in the conceptual model.

Security constraints for cubes and dimensions. These constraints are represented in the logical model as security permissions that affect a whole cube or dimension. They can include expressions that specify in a more detailed level which elements can be shown. These permissions can be modeled at the conceptual level by using several alternatives: security rules (that can include conditions or not) or authorization rules that grant or revoke the read privilege over certain elements.

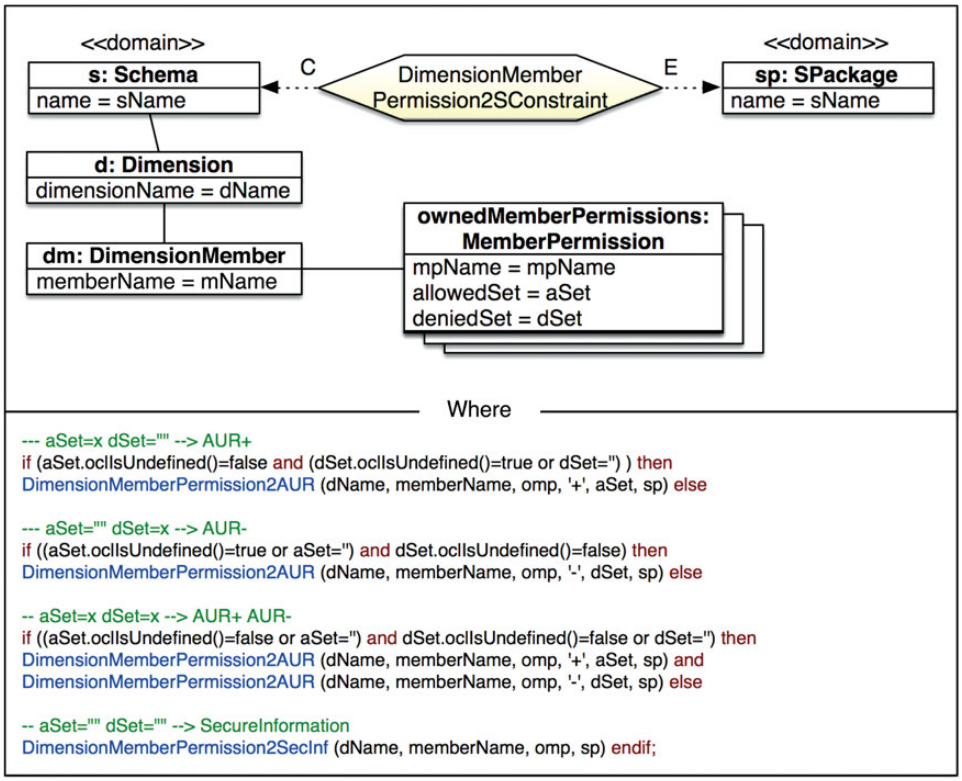

FIGURE 10. DimensionMemberPermission2SConstraint relation.

When each security constraint has to be transformed we have to choose one of these alternatives. In order to automatically support our model-driven approach, the heuristics needed to generate adequate target models in the majority of situations have been defined and implemented within the transformation rules. In this case, the heuristics that manage the security permissions defined over cubes and dimensions have been implemented into the 'CubePermission2SConstraint' (Fig. 7) and 'DimensionPermission2SConstrain' relations.

If the security permission that we want to transform solely indicates the cube or dimension and the security roles involved, this permission is transformed into a security information attached to the cube or dimension and roles indicated.

Nevertheless, the security permission can also specifies an MDX expression that has to be evaluated at running time to know the set of elements that can be shown. In this case, it is transformed into a positive authorization rule that grants to the affected roles the read privilege over the objects indicated in the security permission, keeping also the MDX expression that has to be evaluated.

On the other hand, the MDX expression could indicate elements that cannot be shown by the role. In this case, the authorization rule generated has to be a negative rule that revokes the read permission over the elements specified into the MDX expression.

Finally, if the security permission describes both expressions (for elements that can and cannot be shown), the heuristic defied generates two authorization rules with the implied roles and elements (MDX expression), one of them a positive rule and the other one negative.

Fine grain security constraints for attributes. The security constraints defined over cube measures or dimension attributes are processed in a similar way that cube and dimension permissions. Depending on the characteristics of the security constraints that will be transformed, the heuristics defined generate security permissions or authorization rules considering the four cases above described for the heuristics of cube and dimension permissions.

In this case, the security permissions and authorization rules generated are attached to the cube measures or dimension attributes affected for the source permission. If the source permission does not establish conditions to be evaluated (MDX expressions) a security information attached to the measure or attribute is generated. Whereas if these conditions are defined, the transformation generates the authorization rules needed to grant or revoke the accesses to the measures or attributes involved.

These heuristics for fine grain security constraints have been implemented in the relations 'CubeMemberPermission2-SConstraint' and 'DimensionMemberPermission2SConstraint' (Fig. 10).

5. APPLICATION EXAMPLE

This section shows our proposal applied to a legacy OLAP application implemented into SSAS. The DW used in this example manages information of an airport and has defined several Data Marts for different purposes: trips, flights, incidences, multimedia information, etc. The example presented in this paper is focused on the Data Mart that manages trips and information related such as passengers, baggages, flights and arrival and departure dates and places. This information can be analyzed to make decisions about adding flights for the more requested destinations, reinforcing the security of trips with more incidences, etc.

Next, each stage of the reverse engineering presented in this paper is detailed. First, the legacy implementation of the DW into SSAS will be described. This implementation is the source for obtaining its logical model for the secure OLAP application (according to the SECMDDW metamodel). Then, the model-to-model transformations presented in the previous section are applied in order to obtain the target model, that is, a conceptual model defined according to the SECDW profile. This model includes both structural and security aspects of the DW and is easier to understand and modify than the implementation. Thus, a complete modernization process can be carried out by modifying this model and finally, obtaining its secure implementation by applying the set of transformations for code generation defined in previous works [48, 49].

5.1. Legacy implementation

This section shows pieces of code of the legacy OLAP application for the airport DW used in this example. It has been implemented into SSAS, that uses three kind of XML files to define metadata related with roles, cubes and dimensions (one XML file per each role, cube or dimension). The complete implementation has been omitted and solely a piece of code of the cube 'Trip' and the dimension 'Flight' are shown.

Table 1 partially shows the implementation of the 'Trip' cube. It has defined a measure group composed of the measures price, purpose, etc. and several dimensions that classify the information according to different point of view: Passenger, Baggage, Place, Date and Flight. Finally, it indicates the security constraints attached to the cube and their measures. This table shows a security permission that authorize users with the role 'Staff' to access information of the cube 'Trip'.

Next, Table 2 shows an example of an XML file for dimensions. It represents a piece of code for the dimension 'Flight'. It defines structural aspects such as the identification attribute (flightCode), the remainder attributes (planeCode, etc.) and the classification hierarchies with their aggregation levels (Plane, etc.). On the other hand, it establishes the security permissions that affects dimensions or their attributes. In this table, a permission to grant accesses for the role 'Flight' can be observed as an example.

TABLE 1. Legacy implementation (cube).

```
<Cube><ID>Trip</ID><Name>Trip</Name>
<!--Measures-->
 <MeasureGroups><MeasureGroup><ID>Trip</ID><Name>Trip</Name>
   <Measures>
    <Measure><ID>price</ID><Name>price</Name></Measure>
    <Measure><ID>purpose</ID><Name>purpose</Name></Measure>
    ...
   </Measures></MeasureGroup></MeasureGroups>
<!--Dimensions-->
 <Dimensions>
  <Dimension><ID>Passenger</ID><Name>Passenger</Name>
   <DimensionID>Passenger</DimensionID>
   <Attributes><Attribute><AttributeID>passengerCode</AttributeID></Attribute>
    ...</Attributes>
<!--Security permissions-->
<CubePermissions>
   <CubePermission>
      <ID>CPTripStaff</ID><Name>CPTripStaff</Name><RoleID>Staff</RoleID>
      <Process>true</Process><Read>Allowed</Read>
      <DimensionPermissions><DimensionPermission>
          <CubeDimensionID>Trip</CubeDimensionID><Read>Allowed</Read>
      </DimensionPermission></DimensionPermissions></CubePermission>
    ...</CubePermissions></Cube>
```

TABLE 2. Legacy implementation (dimension flight).

```
<Dimension><ID>Flight</ID><Name>Flight</Name>
<!--Attributes-->
 <Attributes>
  <Attribute><ID>flightCode</ID><Name>flightCode</Name>
   <Usage>Key</Usage>
   <KeyColumns><KeyColumn><DataType>WChar</DataType></KeyColumn></KeyColumns>
  </Attribute>
  <Attribute><ID>planeCode</ID><Name>planeCode</Name>
   <KeyColumns><KeyColumn><DataType>WChar</DataType></KeyColumn></KeyColumns></Attribute>
  ...</Attributes>
<!--Hierarchies-->
  <Hierarchies>
   <Hierarchy><ID>FlightHierarchy0</ID><Name>FlightPlane</Name>
    <Levels><Level><ID>Plane</ID><Name>Plane</Name>
     <SourceAttributeID>planeCode</SourceAttributeID></Level>
     ...</Levels></Hierarchy></Hierarchies>
<!--Security permissions-->
  <DimensionPermissions>
    <DimensionPermission>
      <ID>DPFlightFlight</ID><Name>DPFlightFlight</Name><RoleID>Flight</RoleID>
      <Process>true</Process><Read>Allowed</Read>
      <AllowedSet>Flight</AllowedSet><DeniedSet></DeniedSet>
      <AttributePermissions>...</AttributePermissions>
    </DimensionPermission>
    ...</DimensionPermissions></Dimension>
```

5.2. Logical model

This section presents de logical model obtained by the parser developed for processing the legacy implementation into SSAS. It analyses the metainformation described in XML files and generates the logical model according to the SECMDDW metamodel. In this section, the logical model obtained has been split in several diagrams to improve their description.

First, the set of XML files which represent the information of each security role is analyzed and transformed into the hierarchy of security roles shown in Fig. 11. There is a main role 'User' attached to the main schema that is specialized into 'Staff' and 'Passenger'. The role 'Staff' is deeply specialized according to their functions 'Security', 'Flight' ('Boarding' or 'Baggaging') or 'Administration'.

Next, the XML file with the metainformation for the cube 'Trip' is processed creating both all the structural and security constraints related with the cube. Figure 12 shows the cube part of the logical model obtained. The cube 'Trip' is created and associated with the main schema. Then, cube measures (price, etc.) are added and grouped into a measure group. Once structural elements have been defined the needed security constraints are added. These permissions can affect to the whole cube (CubePermission) such as the permission called 'CPTripSecurity' that allows role 'Security' to accesses the cube, or the permission 'CPTripStaff' that allows role staff to accesses solely the trips with a non-military purpose (by using a denied set condition). Finally, the fine grain security constraints that affect measures (MemberPermission) are defined, such as the permission 'MPpurpose' that solely permits to the role 'Security' to access the purpose of the trip (measure purpose).

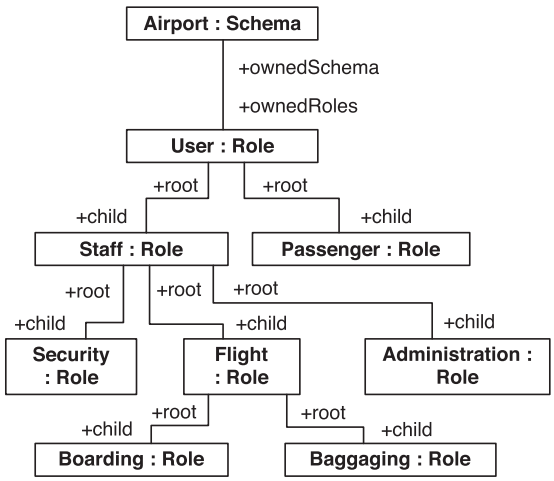

FIGURE 11. Logical model. Security roles.

Finally, dimensions are added to the logical model processing the set of dimension XML files. Since in this example there are five dimensions (Passenger, Baggage, Place, Date and Flight) and they have similar structural and security elements, we have

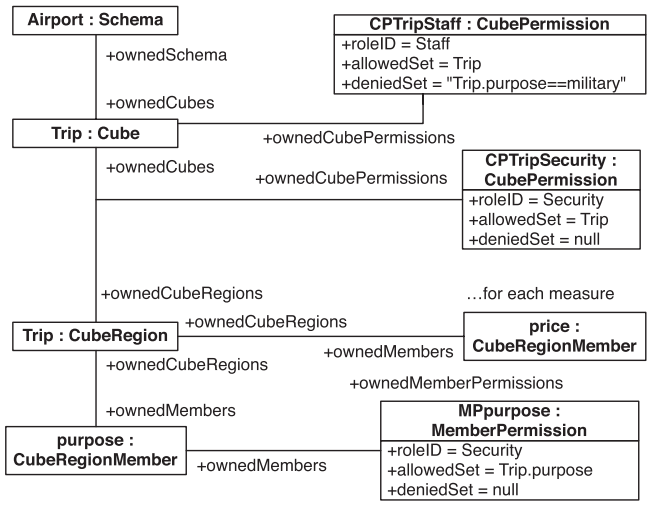

FIGURE 12. Logical model. Cube.

selected two of them (dimensions Flight and Passenger) to describe them. The logical models corresponding with the other dimensions have been omitted.

Figure 13 shows another example of a logical model for a dimension. In this case, the dimension modeled is 'Passenger' that includes an identification attribute (passengerCode) and several attributes (name, address, fingerprint, etc.). This dimension does not have classification hierarchies defined. Nevertheless, it has two security permissions defined. One of them grants accesses to 'Passenger' dimension to the role 'Security' ('DPPassengerSecurity'), whereas the other one ('DPPassengerBoarding') grants accesses to certain passengers to the role 'Boarding'. This security permission splits the set of passenger into suspicious and non-suspicious and allows users with a 'Boarding' role to solely access non-suspicious passengers. To consider a passenger as suspicious its attribute suspicious has to be setted as 'true' and its attribute risk index has to indicate a value greater than five.

Figure 14 models the dimension 'Flight', associated with the cube 'Trip'. This dimension has an identification attribute (flightCode) and several attributes (planeCode, planeName, etc.). Furthermore, its information can be grouped by following a classification hierarchy composed of different aggregation levels with certain detail levels (Plane, Aircraft, Company). On the other hand, the security constraints defined affecting the dimension 'Flight' or their attributes are processed creating dimension or member permissions. In this figure, several dimension permissions can be observed. They establish that information of the 'Flight' dimension can be accessed by the user role 'Flight' ('DPFlightFlight' dimension permission).

5.3. Conceptual model

Once the logical model has been obtained, the transformation (QVT rules) defined in this paper is applied in order to

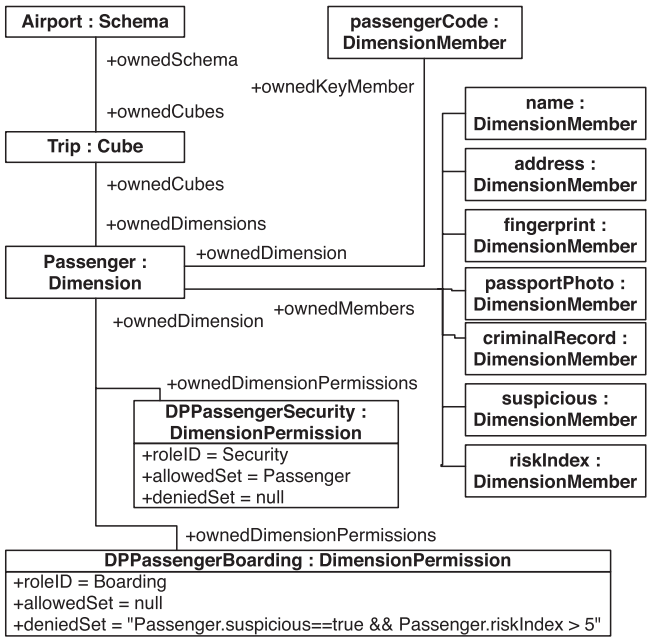

FIGURE 13. Logical model. Dimension passenger.

automatically generates the conceptual model. Since the conceptual model is richer than the logical model that is focused on the OLAP technology, a source element can have different choices to be transformed into the target model. This transformation includes heuristics to automatically decide which elements create depending of the source information.

Figure 15 shows the conceptual model obtained. At a first look, it can be observed how the conceptual model is smaller and easier to understand and to modify than the logical model, and than the implementation.

This model is composed of a central fact for trips (SFact 'Trip') that stores information of price, seat, distance, flight time, if check-in and boarding have been carried out and the purpose of the flight. Trips information can be classified according to different dimensions: Passenger, Baggage, Place, Data and Flight. Each dimension has associated attributes, for instance, the dimension 'Passenger' has the attributes 'passengerCode', 'name', 'address', etc. Furthermore, some dimensions present classification hierarchies such as the dimension 'Place' that can be grouped in different detail levels 'Place', 'Gate', 'Terminal' and 'Airport'. Furthermore, the set of security roles is defined keeping the relationships of the hierarchy established in the logical model.

Finally, the security constraints (cube, dimension and member permissions) are represented as tagged values in classes or attributes. For instance, the tagged value 'SR=Staff' of the class 'Trip' indicates that an user role of 'Staff' is needed to access its information. Fine grain security constraints that affect attributes are also defined as tagged values such as the constraint 'SR=Security' defined for the attribute 'purpose'. On the other

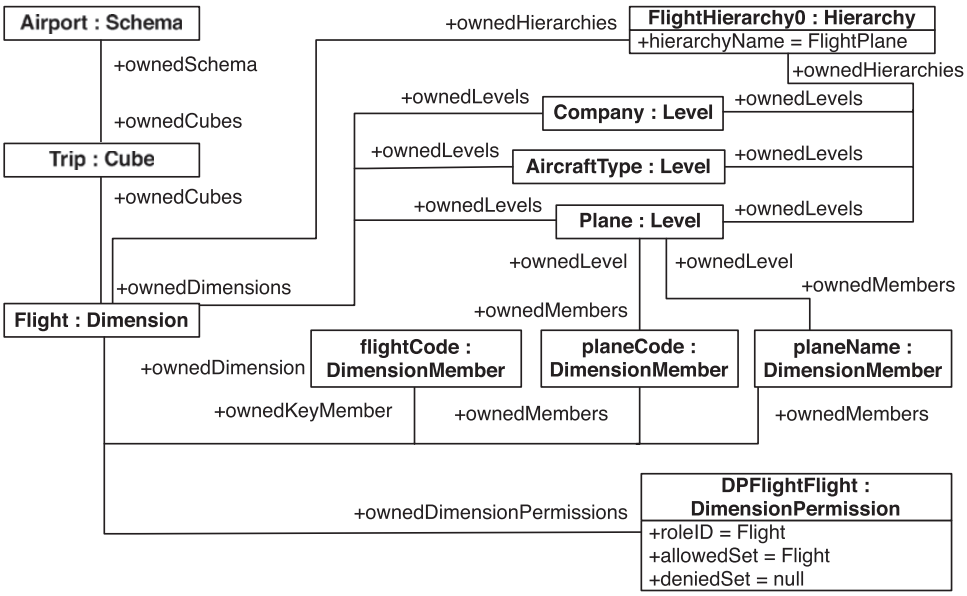

FIGURE 14. Logical model. Dimension Flight.

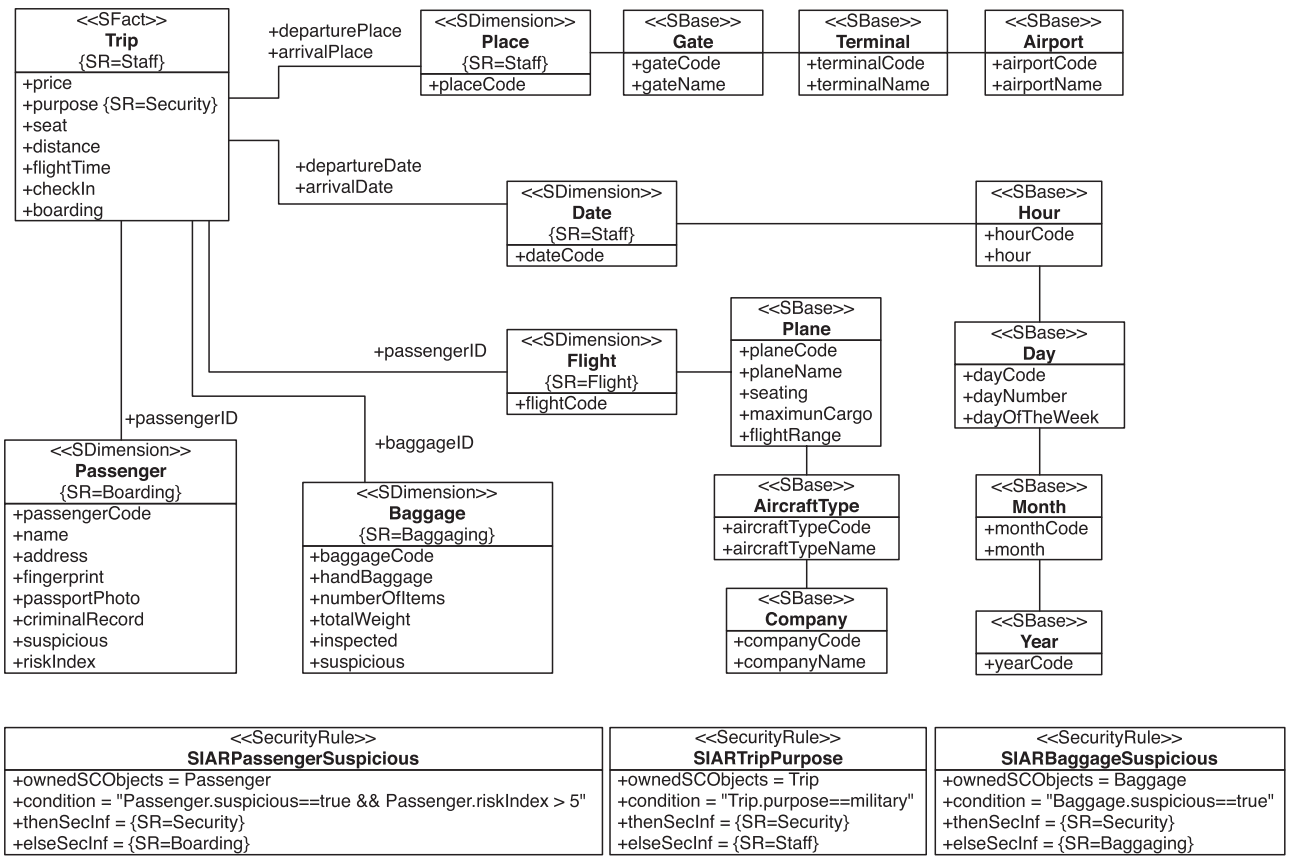

FIGURE 15. Conceptual model.

hand, more complex security rules can be defined in the conceptual model. These security rules are able to group several security permissions (from the logical model) into a single rule easier to understand (in the conceptual model). In this example, three security rules have been created. For instance, the security rule 'SIARPassengerSuspicious' splits passengers into two groups (suspicious and non-suspicious) and establishes that the role 'Security' can access all passengers and the role 'Boarding' solely the passengers considered as non-suspicious (depending on the evaluation of the condition expressed in the rule).

6. CONCLUSIONS

The proposal presented in this paper has been conceived as a model-driven architecture [51, 52]. Model-driven development is based on the definition of models that separate the specification of system functionalities and its implementation by using a specific technology. Furthermore, the development process can be automated by defining transformations from models toward the final implementation. This approach improves the development process reducing times and costs and also improves the quality obtained in the final product [4, 5].

The model-driven approach has been successfully applied to different software development areas such as data bases [53–56], DWs [57–59], web services [60–63], product lines [64], critical applications [65] or real-time systems [66–68].

Moreover, the quality of the final solution obtained is also improved by using our approach which is focused on the security improvement. Our proposal allows designers to observe conceptual models including security constraints within structural aspects. At the conceptual abstraction level designers model security restrictions in an easier and more understandable way than establishing security directly into the final implementation. The use of a more understandable model mitigates possible mistakes derived from managing a vast amount of code.

On the other hand, the identification and inclusion of security aspects into the models corresponding to early development stages improve the final product quality. If security constraints are early modeled, these security constraints are considered for making design decisions when the system is automatically generated by using transformations. So, the security constraints are perfectly fitted into the final solution.

The reverse engineering process presented in this paper allows us to automatically obtain conceptual models from legacy OLAP applications. In order to achieve this goal we have develop: (i) a parser to analyze the legacy OLAP applications metadata and to generate the corresponding logical model for OLAP; and (ii) we have defined a set of QVT transformations to obtain conceptual models independent of the platform from logical models focused on the OLAP technology.

These models are useful for re-document existing applications and improve OLAP applications modifying the models obtained. Another contribution of our proposal is that the reverse engineering process also considers security aspects. That is, if the legacy system has security constraints defined, these are extracted and represented in the conceptual model. And once the conceptual model has been obtained, it can be used to incorporate or to modify the security configuration. Finally, the implementation for the modified conceptual model can be automatically obtained by using our model-driven architecture (by applying the set of transformation rules defined in previous works for automatically generating secure OLAP code from conceptual models).

The validation of our proposal has been planed in several stages. Firstly, small application examples helped us for an early evaluation. Next, this paper has presented an application example that includes a great variety of the structural and security elements allowing us to evaluate the applicability of the proposed models and transformations and to improve them. Nevertheless, as a future work we consider necessary to include a next stage to complete the evaluation of our architecture by applying it to industrial case studies with the participation of professional designers. We will also define a family of experiments in order to measure the improvement obtained in comparison to the traditional development process.

Furthermore, once the conceptual model for a legacy system has been obtained, we would like to offer different targets to migrate the improved systems. In order to achieve this goal we plan as future work to support more OLAP tools (such as Pentaho) and different technologies (such as NoSQL). To add this support into our model-driven approach, we will need to define the necessary models and transformations. For instance, if we want to include a new technology, we have to define a new logical model for this technology and the transformations needed to connect the legacy implementation with the new logical model and the logical model with the conceptual model. Finally, as it was commented before we plan to complete our approach by offering support with the requirements' specification stage. In this way, a CIM model has been proposed, but due to the great semantic gap between requirements and conceptual levels, the transformations provided are not completely automated. We need to define the transformations needed to completely automate both processes (code generation and reverse engineering) toward the requirements abstraction level.

FUNDING

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

REFERENCES

[1] Priebe, T. and Pernul, G. (2001) A Pragmatic Approach to Conceptual Modeling of OLAP Security. *20th Int. Conf. on Conceptual Modeling (ER 2001)*, Yokohama, Japan. Springer.

[2] Weippl, E., Mangisengi, O., Essmayr, W., Lichtenberger, F. and Winiwarter, W. (2001) An Authorization Model for Data Warehouses and olap. *Workshop on Security in Distributed Data Warehousing*, New Orleans, Louisiana, USA.

[3] Thuraisingham, B., Kantarcioglu, M. and Iyer, S. (2007) Extended rbac-based design and implementation for a secure data warehouse. *Int. J. Bus. Intell. Data Mining*, **2**, 367–382.

[4] Fernández-Medina, E., Jurjens, J., Trujillo, J. and Jajodia, S. (2009) Model-driven development for secure information systems. *Inf. Softw. Technol.*, **51**, 809–814.

[5] Mouratidis, H. (2011) *Software Engineering for Secure Systems: Industrial and Research Perspectives*. IGI Global.

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

[7] Muller, H.A., Jahnke, J.H., Smith, D.B., Storey, M.-A., Tilley, S.R. and Wong, K. (2000) Reverse engineering: A road map. *Proc. of the Conf. on the Future of Software Engineering (ICSE '00)*, pp. 47–60.

[8] Canfora, G. and Penta, M.D. (2007) *New Frontiers of Reverse Engineering*. Future of Software Engineering. IEEE Computer Society, pp. 326–341.

[9] Aiken, P.H. (1998) Reverse engineering of data. *IBM Syst. J.*, **37**, 246–269.

[10] Blaha, M. (2001) A Retrospective on Industrial Database Reverse Engineering Projects-Part 1. *Proc. 8th Working Conf. on Reverse Engineering (WCRE01)*, Suttgart, Germany, pp. 136–147. IEEE Computer Society.

[11] Cohen, Y. and Feldman, Y.A. (2003) Automatic high-quality re-engineering of database programs by abstraction, transformation and reimplementation. *ACM Trans. Softw. Eng. Methodol.*, **12**, 285–316.

[12] Hainaut, J.-L., Englebert, V., Henrard, J., Hick, J.-M. and Roland1, D. (2004) Database reverse engineering: From requirements to care tools. *Appl. Categ. Struct.*, **3**, pp. 9–45.

[13] Fernández-Medina, E., Trujillo, J. and Piattini, M. (2007) Model-driven multidimensional modeling of secure data warehouses. *Eur. J. Inf. Syst.*, **16**, 374–389.

[14] Jurjens, J. (2004) *Secure Systems Development with UML*. Springer.

[15] Jurjens, J. and Schmidt, H. (2011) Umlsec4uml2 – Adopting UMLsec to Support UML2. Technical Report. Technical Reports in Computer Science. Technische Universitat Dortmund. http://hdl.handle.net/2003/27602.

[16] Basin, D., Doser, J. and Lodderstedt, T. (2006) Model-driven security: from UML models to access control infrastructures. *ACM Trans. Softw. Eng. Methodol.*, **15**, 39–91.

[17] Lodderstedt, T., Basin, D. and Doser, J. (2002) Secureuml: A UML-Based Modeling Language for Model-Driven Security. *UML 2002. The Unified Modeling Language. Model Engineering, Languages Concepts, and Tools. 5th International Conference*, Dresden, Germany, pp. 426–441. Springer.

[18] Best, B., Jurjens, J. and Nuseibeh, B. (2007) Model-Based Security Engineering of Distributed Information Systems Using UMLsec. *Int. Conf. on Software Engineering*, Minneapolis, MN, USA, pp. 581–590. IEEE.

[19] Jurjens, J. and Shabalin, P. (2007) Tools for secure systems development with UML. *Int. J. Softw. Tools Technol. Transf.*, **9**, 527–544.

[20] Matulevicius, R. and Dumas, M. (2011) Towards Model Transformation Between Secureuml and UMLsec for Role-Based Access Control. *Proc. 2011 Conf. on Databases and Information Systems VI: Selected Papers from the 9th Int. Baltic Conf., DB&IS 2010*, Amsterdam, The Netherlands, pp. 339–352. IOS Press.

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

[22] Massacci, F., Mylopoulos, J. and Zannone, N. (2007) Computer-aided support for secure tropos. *Autom. Softw. Eng.*, **14**, 341–364.

[23] Compagna, L., Khoury, P., Krausová, A., Massacci, F. and Zannone, N. (2009) How to integrate legal requirements into a requirements engineering methodology for the development of security and privacy patterns. *Artif. Intell. Law*, **17**, 1–30.

[24] van de Riet, R. (2008) Twenty-five years of mokum: for 25 years of data and knowledge engineering: Correctness by design in relation to mde and correct protocols in cyberspace. *Data Knowl. Eng.*, **67**, 293–329.

[25] Nunes, F.J.B., Belchior, A.D. and Albuquerque, A.B. (2010) Security engineering approach to support software security. *IEEE 6th World Congress on Services (SERVICES-1)*, pp. 48–55.

[26] Paim, F.R.S. and Castro, J. (2003) Dwarf: an approach for requirements definition and management of data warehouse systems. *11th IEEE Int. Requirements Engineering Conf., 2003. Proceedings*. pp. 75–84.

[27] Sapia, C., Blaschka, M., Hofling, G. and Dinter, B. (1998) Extending the e/r model for the multidimensional paradigm. *1st Int. Workshop on Data Warehouse and Data Mining (DWDM'98)*, Singapore, pp. 105–116. Springer. *11th IEEE Int. Requirements Engineering Conf., 2003. Proceedings*. pp. 75–84

[28] Tryfona, N., Busborg, F. and Christiansen, J. (1999) Starer: A Conceptual Model for Data Warehouse Design. *ACM 2nd Int. Workshop on Data Warehousing and OLAP (DOLAP'99)*, Missouri, USA, pp. 3–8. ACM.

[29] Binh, N.T., Tjoa, A.M. and Wagner, R. (2000) An object-oriented multidimensional data model for olap. *In Proc. of 1st Int. Conf. on Web-Age Information Management (WAIM)*. LNCS 1846. pp. 69–82.

[30] Abelló, A., Samos, J. and Saltor, F. (2006) Yam2: A multidimensional conceptual model extending uml. *Inf. Syst.*, **31**, 668–677.

[31] Golfarelli, M. and Rizzi, S. (2009) *Data Warehouse Design: Modern Principles and Methodologies*. McGraw-Hill Osborne Media.

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

[33] Kirkgoze, R., Katic, N., Stolda, M. and Min Tjoa, A. (1997) A Security Concept for olap. *8th Int. Workshop on Database and Expert System Applications (DEXA'97)*, Toulouse, France, pp. 619–626. IEEE Computer Society.

[34] Liu, Y., Sung, S. and Xiong, H. (2006) A cubic-wise balance approach for privacy preservation in data cubes. *Inf. Sci.*, **176**, 1215–1240.

[35] Sung, Y., Liu, Y., Xiong, H. and A. Xg, P. (2006) Privacy preservation for data cubes. *Knowl. Inf. Syst.*, **9**, 38–61.

[36] Henrard, J. and Hainaut, J.-L. (2001) Data dependency elicitation in database reverse engineering. *CSMR '01 Proc. of the Fifth European Conf. on Software Maintenance and Reengineering*, pp. 11–19.

[37] Hainaut, J.-L., Henrard, J., Ronald, D. and Englebert, V. (1996) Database design recovery. CAiSE, pp. 272–300.

[38] Soutou, C. (1998) Relational database reverse engineering: Algorithms to extract cardinality constraints. *Data Knowl. Eng.*, **28**, 161–207.

[39] Sousa, P., Pedro-de Jesus, L., Pereira, G. and Brito e Abreu, F. (2002) Clustering relations into abstract er schemas for database reverse engineering. *Sci. Comput. Program*, **45**, 137–153.

[40] Behm, A., Geppert, A. and Dittrich, K. (2000) Algebraic database migration to object technology. *ER'00 Proc. of the 19th int. conf. on Conceptual modeling*, pp. 440–453.

[41] McBrien, P. and Poulovassilis, A. (1999) Automatic migration and wrapping of database applications—a schema transformation approach. *ER '99 Proc. of the 18th Int. Conf. on Conceptual Modeling*, pp. 93–113.

[42] Polo, M., García Rodríguez de Guzmán, I. and Piattini, M. (2007) An mda-based approach for database re-engineering. *J. Softw. Maintance Evol.*, **19**, 383–417.

[43] Polo, M., Gómez, J., Piattini, M. and Ruiz, F. (2002) Generating three-tier applications from relational databases: a formal and practical approach. *Inf. Softw. Technol.*, **44**, 923–941.

[44] Blanco, C., de Guzmán, I.G.R., Fernández-Medina, E. and Trujillo, J. (2014) Showing the benefits of applying a model-driven architecture for developing secure olap applications. *J. UCS*, **20**, 79–106.

[45] Trujillo, J., Soler, E., Fernández-Medina, E. and Piattini, M. (2009) A uml 2.0 profile to define security requirements for data warehouses. *Comput. Stand. Interfaces*, **31**, 969–983.

[46] Yu, E. (1997) Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. *3rd IEEE International Symposium on Requirements Engineering (RE'97)*, Washington, DC, pp. 226–235. IEEE.

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

[48] Blanco, C., García Rodríguez de Guzmán, I., Fernández-Medina, E., Trujillo, J. and Piattini, M. (2010) Defining and transforming security rules in an mda approach for dws. *Int. J. Bus. Intell. Data Mining*, **5**, 116–133.

[49] Blanco, C., García-Rodríguez de Guzmán, I., G.Rosado, D., Fernández-Medina, E. and Trujillo, J. (2009) Applying qvt in order to implement secure data warehouses in sql server analysis services. *J. Res. Practice Inf. Technol.*, **41**, 135–154.

[50] Trujillo, J., Soler, E., Fernández-Medina, E. and Piattini, M. (2009) An engineering process for developing secure data warehouses. *Inf. Softw. Technol.*, **51**, 1033–1051.

[51] Bézivin, J. (2004) In search of a basic principle for model-driven engineering. *Upgrade*, **5**, 21–24.

[52] Mellor, Clark, and Futugami (2003) Model-driven development—guest editor's introduction. *IEEE Softw.*, **20**, 14–18.

[53] Vela, B., Fernandez-Medina, E., Marcos, E. and Piattini, M. (2006) Model-driven development of secure xml databases. *ACM Sigmod Record*, **35**, 22–27.

[54] Li, B., Liu, S. and Yu, Z. (2005) Applying mda in Traditional Database-Based Application Development. *Int. Conf. on Computer Supported Cooperative Work in Design*, Coventry, UK, pp. 1038–1041. IEEE.

[55] Dubielewicz, I., Hnatkowska, B., Huzar, Z. and Tuzinkiewicz, L. (2007) Evaluation of mda-psm Database Model Quality in the Context of Selected Non-Functional Requirements. *Int. Conf. on Dependability of Computer Systems*, Szklarska Poreba, Poland, pp. 19–26. IEEE.

[56] Vara, J., Vela, B., Cavero, J. and Marcos, E. (2007) Model Transformation for Object-Relational Database Development. *ACM Symposium on Applied Computing*, Seoul, Korea, pp. 1012–1019. ACM.

[57] Mazón, J.-N. and Trujillo, J. (2008) An mda approach for the development of data warehouses. *Decis. Support Syst.*, **45**, 41–58.

[58] Vela, B., Mazón, J., Blanco, C., Fernández-Medina, E., Trujillo, J., and Marcos, E. (2013) Automatic generation of secure xml data warehouses by using qvt within an mda framework. *Inf. Softw. Technol.*, **55**.

[59] Fernandes, L., Neto, B., Fagundes, V., Zimbrao, G., de Souza, J. and Salvador, R. (2010) Model-driven architecture approach for data warehouse. *6th Int. Conf. on Autonomic and Autonomous Systems (ICAS)*, pp. 156–161.

[60] Meliá, S. and Gomez, J. (2006) The websa approach: applying model-driven engineering to web applications. *J. Web Eng.*, **5**, 121–149.

[61] Tan, W., Ma, L., Li, J. and Xiao, Z. (2006) Application mda in a Conception Design Environment. *Int. Multi-Symposiums on Computer and Computational Sciences*, Hangzhou, China, pp. 702–704. IEEE Computer Society.

[62] Yu, B., Zhang, C. and Zhao, Y. (2006) Transform from Models to Service Description Based on mda. *Asia-Pacific Conference on Services Computing*, GuangZhou, China, pp. 605–608. IEEE.

[63] Kraus, A., Knapp, A. and Koch, N. (2007) Model-Driven Generation of Web Applications in uwe. *International Workshop on Model-Driven Web Engineering*, Como, Italy.

[64] Braganca, A. and Machado, R. (2007) Model-Driven Development of Software Product Lines. *Int. Conf. on the Quality of Information and Communications Technology*, Lisbon, Portugal, pp. 199–203. IEEE.

[65] Moebius, N., Stenzel, K. and Reif, W. (2009) Generating formal specifications for security-critical applications—a model-driven approach.

[66] Cuccuru, A., De Simone, R., Saunier, T., Siegel, G. and Sorel, Y. (2005) P2i: An Innovative mda Methodology for Embedded Real-Time Systems. *Euromicro Conference on Digital System Design*, Porto, Portugal, pp. 26–33. IEEE.

[67] Lu, S., Halang, W.A. and Zhang, L. (2005) A Component-Based uml Profile to Model Embedded Real-Time Systems Designed by the mda Approach. *Int. Conf. on Embedded and Real-Time Computing Systems and Applications*, Hong Kong, China, pp. 563–566. IEEE.

[68] Wang, Y., Zhou, X., Liang, L. and Peng, C. (2006) A mda-Based Soc Modeling Approach Using UML and SystemC. *Int. Conf. on Computer and Information Technology*, Baridhara, Bangladesh, pp. 245–251. IEEE.