Towards a UML 2.0/OCL extension for designing Secure Data Warehouses

Rodolfo Villarroel¹, Eduardo Fernández-Medina², Juan Trujillo³, and Mario Piattini²

¹ Departamento de Computación e Informática. Universidad Católica del Maule (Chile)
² Departamento de Informática. Universidad de Castilla-La Mancha (Spain)
³ Departamento de Lenguajes y Sistemas Informáticos. Universidad de Alicante (Spain)

Abstract. At present, it is very difficult to develop a methodology that fulfills all criteria and comprises all security constraints in terms of confidentiality, integrity and availability, to successfully design data warehouses. If that methodology was developed, its complexity would avoid its success. Therefore, the solution would be an approach in which techniques and models defined by the most accepted model standards (such as UML) were extended by integrating the necessary security aspects that, at present, are not covered by the existing methodologies. In this paper, we will focus on solving confidentiality problems in data warehouses conceptual modeling by defining a profile using the UML 2.0 extensibility mechanisms. In addition, we will define an OCL extension that allows us to specify the static and dynamic security constraints of the elements of data warehouses conceptual modeling, and we will show the benefit of our approach by applying this profile to an example.

1 Introduction

Security and specifically confidentiality is a very important aspect for data warehouses due to the fact that the constant changes of users requests and data sources force them not only to be more flexible but also to control more effectively information confidentiality. A very important aspect to be considered of data warehouses that make them different from operational systems is that information is not statically treated but the evolution of it becomes more important as time goes by, in other words, its history [10]. For this reason, mechanisms allowing confidentiality of such quantity of information must be established. Indeed, the very survival of the organizations depends on the correct management, security and confidentiality of information [5]. In fact, as some authors remarked [4, 6], information security is a serious requirement which must be carefully considered, not as an isolated aspect, but as an element present in all stages of the development lifecycle, from the requirement analysis to implementation and maintenance. Chung et al. also insist on integrating security requirements into design, by providing designers with models specifying security aspects, but they do not deal with data warehouses issues [2].
In the past few years, various approaches have been proposed to represent the main multidimensional (MD) properties at the conceptual level [1, 8, 9, 17-19]. However, none of these approaches for MD modeling considers security as an important issue of their conceptual models, so they do not solve the problem of security in these kinds of systems. Moreover, in the literature, we can find several initiatives to include security in data warehouses [11, 12, 15, 16]. Many of them are focused on interesting aspects related to access control, multilevel security, its applications to federated databases, applications using commercial tools and so on. However, neither of them considers the security aspects comprising all stages of the system development cycle nor the introduction of security into MD conceptual design.

We think that our solution would be an approach in which techniques and models defined by the most accepted model standards were extended by integrating the necessary security aspects that, at present, are not covered by the existing methodologies. Taking this viewpoint into account, the UML offers us with two different approaches to extend its metamodel [7]. The first one provides us with the possibility of defining a new modeling language by using MOF (Meta Object Facility) in which there are no restrictions regarding what can be done with a metamodel. For example, metaclasses and relationships can be added and removed according to our needs. We have not chosen this option because the new language will not respect the UML semantics and as a consequence, we will not be able to use commercial tools based on UML. Moreover, the purpose of our proposal is to be able to precisely and easily generate a secure conceptual modeling applied to a specific dominion, in this case, to data warehouses. This fact perfectly fits with the concept of profile.

A UML 2.0 profile is defined as a UML package stereotyped “profile”, that can extend either a metamodel or another profile [14]. A profile is used to extend an existing metamodel by using three basic mechanisms provided by the UML: stereotypes, tagged values and constraints to adapt it to a dominion, platform or specific method. In our case, we will use the indicated mechanisms to incorporate security aspects into data warehouses conceptual modeling.

The remainder of this paper is structured as follows. Section 2 will present the UML 2.0/OCL profile for designing secure data warehouses. In section 3, an example of modeling using the proposed extensibility mechanisms will be stated. Finally, Section 4 will put forward our main conclusions and will introduce our immediate future work.

2 UML 2.0/OCL profile for designing Secure Data Warehouses

In this section, we will present the main aspects of our profile for designing secure data warehouses. According to [3], an extension to the UML begins with a brief description and then lists and describes all the stereotypes, tagged values, and constraints of this extension. Basically, we have reused the previous profile defined in [13], which allows us to design data warehouses from a conceptual perspective, and we have added the required elements to generate the profile (a set of tagged values, stereotypes, and constraints), which enables us to create secure MD models.
Furthermore, an extension is formed by a set of well-formedness rules that will ensure a correct static semantics of the multidimensional model.

The goal of this UML profile is to be able to design MD conceptual model, but classifying information in order to define which properties has to own the user to be entitled to access information. Therefore, our aim is to classify the security information that will be used in our data warehouses conceptual modeling. We can define, for each element of the model (fact class, dimension class, fact attribute, etc.), its security information, specifying a sequence of security levels, a set of user compartments, and a set of user roles. We can also specify security constraints considering these security attributes. The security information and these constraints indicate the security properties that users have to own to be able to access information. We have adapted OCL [20] to be coherent with our UML 2.0 profile.

### 2.1 General Description

Our profile will be called SECDW (Secure Data Warehouses) and will be represented as a UML package. This profile will not only inherit all properties from UML metamodel but also it will incorporate new data types, stereotypes, tagged values and constraints. In Figure 1, a high level view of our SECDW profile is provided. The package SECDW and OCL are imported from SECDW profile. Therefore, SECDW data types and OCL types will be used as valid types for stereotypes of our profile.

![Fig. 1. High level view of our SECDW profile](image)

### 2.2 Data Types

We need the definition of some new data types to be used in the tagged values definitions of the new stereotypes. In Table 1, we will provide the new data types definitions we have specified. All the information considered in these new data types has to be defined for each specific secure conceptual database model depending on its confidentiality properties, and on the number of users and complexity of the organization in which the data warehouse will be operative.
Table 1. New Data Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Base class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level</td>
<td>Enumeration</td>
<td>The type Level will be an ordered enumeration composed by all security levels that have been considered.</td>
</tr>
<tr>
<td>Levels</td>
<td>Primitive</td>
<td>The type Levels will be an interval of levels composed by a lower level and an upper level.</td>
</tr>
<tr>
<td>Role</td>
<td>Primitive</td>
<td>The type Role will represent the hierarchy of user roles that can be defined for the organization.</td>
</tr>
<tr>
<td>Compartment</td>
<td>Enumeration</td>
<td>The type Compartment is the enumeration composed by all user compartments that have been considered for the organization.</td>
</tr>
<tr>
<td>Privilege</td>
<td>Enumeration</td>
<td>The type Privilege will be an ordered enumeration composed by all different privileges that have been considered.</td>
</tr>
<tr>
<td>AccessAttempt</td>
<td>Enumeration</td>
<td>The type AccessAttempt will be an ordered enumeration composed by all different access attempts that have been considered.</td>
</tr>
</tbody>
</table>

In figure 2, we can observe the values associated to each one of the necessary types. Security levels, roles and organizational compartments can be defined according to the needs of the organization. However, for this figure to be better understood, we have considered within the “Level” data type, the typical values associated to security levels.

![Types SECDW Diagram]

Fig. 2. Values associated to new data types
2.3 Stereotypes

We have defined a package that includes all the stereotypes that will be necessary in our profile (see Figure 3). This profile contains four types of stereotypes:

- Secure class and secure data warehouses stereotypes (and stereotypes inheriting information from them) that contain tagged values associated to attributes (model or class attributes), security levels, user roles and organizational compartments.
- Attribute stereotypes (and stereotypes inheriting information from attributes) and instances, that have tagged values associated to security levels, user roles and organizational compartments.
- Stereotypes that allow us to represent security constraints, authorizations rules and audit rules.
- User Profile stereotype, which is necessary to specify constraints depending on particular information of a user or a group of users.

In figure 3, we can see the tagged values associated to each one of the stereotypes. For example, ‘SecureDW’ stereotype has the following values associated: Classes, SecurityLevels, SecurityRoles and SecurityCompartments. In Table 2, we will show the description of each one of the stereotypes.

![Fig. 3. New stereotypes](image-url)
## Table 2. Stereotypes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecureDW</td>
<td>Instances of this data warehouse model will allow us to define security information and constraints regarding its elements.</td>
</tr>
<tr>
<td>UserProfile</td>
<td>Classes of this stereotype contain all the properties that the systems manage from users.</td>
</tr>
<tr>
<td>Secure Class</td>
<td>This type of class can have sensitivity information associated. We can therefore classify these classes according to their own confidentiality properties.</td>
</tr>
<tr>
<td>SecureFact</td>
<td>They represent facts within a multidimensional model. They inherit tagged values from SecureClass.</td>
</tr>
<tr>
<td>SecureDimension</td>
<td>They represent dimensions within a multidimensional model. They inherit tagged values from SecureClass.</td>
</tr>
<tr>
<td>SecureBase</td>
<td>They represent dimension hierarchy levels within a multidimensional model. They inherit tagged values from SecureClass.</td>
</tr>
<tr>
<td>SecureAttribute</td>
<td>This type of attributes can have sensitivity information associated. We can therefore classify these attributes according to its own confidentiality properties.</td>
</tr>
<tr>
<td>SecureFactAttribute</td>
<td>They represent Fact class attributes within a multidimensional model and inherit tagged values from SecureAttribute.</td>
</tr>
<tr>
<td>SecureDimensionAttribute</td>
<td>They represent Dimension or Base class attributes within a multidimensional model and inherit tagged values from SecureAttribute.</td>
</tr>
<tr>
<td>SecureOID</td>
<td>They represent OID attributes (Identifier attribute) of Fact, Dimension or Base classes within a multidimensional model and inherit security aspects from SecureAttribute.</td>
</tr>
<tr>
<td>SecureDescriptor</td>
<td>They represent descriptor attributes of Dimension or Base classes within a multidimensional model and inherit security aspects from SecureAttribute.</td>
</tr>
<tr>
<td>SecureInstance</td>
<td>This type of instances can have sensitivity information associated. We can therefore classify these instances according to their own confidentiality properties.</td>
</tr>
<tr>
<td>AuditRule</td>
<td>This type of rules can contain information to analyze the user behaviour when using the system. Therefore, they will specify whether access must be registered.</td>
</tr>
<tr>
<td>AuthorizationRule</td>
<td>This type of rules can contain information to permit or deny access. Therefore, they will specify if authorization is positive or negative and the necessary privileges to access.</td>
</tr>
<tr>
<td>SecurityRule</td>
<td>This type of rules can have sensitivity information associated. Therefore, they will specify if security information is necessary.</td>
</tr>
</tbody>
</table>
2.4 Tagged Values

The tagged values we have defined are applied to certain components that are especially particular to MD modeling, allowing us to represent them in the same model and in the same diagrams that describe the rest of the system. In Table 3, the necessary tagged values in our profile are shown. These tagged values will represent the sensitivity information of the different elements of the MD modeling (fact class, dimension class, base class, attributes, etc.), and they will allow us to specify security constraints depending on this security information and on the value of attributes of the model.

Table 3. Tagged values

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>Set(OclType)</td>
<td>It specifies all classes of the model. This new tagged value is useful in order to navigate through all classes of the model.</td>
<td>Empty set</td>
</tr>
<tr>
<td>Attributes</td>
<td>Set(OclType)</td>
<td>It specifies all attributes of the class. This new tagged value is useful in order to navigate through all attributes of the model.</td>
<td>Empty set</td>
</tr>
<tr>
<td>SecurityLevels</td>
<td>Levels</td>
<td>It specifies the interval of possible security level values that an instance of this class can receive.</td>
<td>The lowest level (if we consider traditional levels, should be 'Unclassified')</td>
</tr>
<tr>
<td>SecurityRoles</td>
<td>Set(Role)</td>
<td>It specifies a set of user roles. Each role is the root of a subtree of the general user role hierarchy defined for the organization.</td>
<td>The set composed by one role that is the role hierarchy defined for the model</td>
</tr>
<tr>
<td>Security-Compartments</td>
<td>Set (Compartment)</td>
<td>It specifies a set of compartments. All instances of this class can have the same user compartments, or a subset of them.</td>
<td>Empty set of compartments</td>
</tr>
<tr>
<td>LogType</td>
<td>AccessAttempt</td>
<td>It specifies whether the access has to be recorded: none, all access, only frustrated accesses, or only successful accesses.</td>
<td>None</td>
</tr>
<tr>
<td>Involved-Classes</td>
<td>Set(OclType)</td>
<td>It specifies the classes that have to be involved in a query to be enforced in an exception.</td>
<td>Empty</td>
</tr>
<tr>
<td>ExceptSign</td>
<td>{'+', '-'}</td>
<td>It specifies if an exception permits (+) or denies (-) access to instances of this class to a user or a group of users.</td>
<td>+</td>
</tr>
<tr>
<td>Except-Privilege</td>
<td>Set(Privilege)</td>
<td>It specifies the privileges the user can receive or remove.</td>
<td>Read</td>
</tr>
<tr>
<td>isTime</td>
<td>Boolean</td>
<td>It indicates whether dimension represents a time dimension or not.</td>
<td>False</td>
</tr>
<tr>
<td>derivationRule</td>
<td>String</td>
<td>If the attribute is derived, this tagged value represents the derivation rule.</td>
<td>Empty</td>
</tr>
</tbody>
</table>
2.5 Well-Formedness Rules

A set of inherent constraints are specified in order to define well-formedness rules. The correct use of our extension is assured by the definition of constraints in both natural language and Object Constraint Language (OCL). We will identify and specify some well-formedness rules needed for the correct use of the new elements specified in this profile. These rules are grouped as follows:

- Correct value of tagged values. For example; the security levels defined for each class of the model and for each attribute of each class has to belong to the sequence of security levels that has been defined for the model.
- Security information of instances. For example, the security level of the instance of a class has to be included in the ranking of security levels that has been defined for the class.
- Relationship between security information of classes and their attributes. The security levels defined for an attribute have to be equal or more restrictive than the security levels defined for its class.
- Categorization of dimensions. When a dimension class is specialized in several base classes, the security levels of the subclasses have to be equal or more restrictive than the security levels of the superclass.
- Classification hierarchies. As a general rule, we can consider that the more specific the information is, the more restrictive its access is.
- Derived Attribute. The security levels of a derived attribute have to be equal or more restrictive than the attributes which this attribute is based on.
- Combination of dimensions. For example, a query that involves the combination of several dimensions class, and the fact class, has to consider the combination of the security information of all classes. The security levels of the combination will be the most restrictive in the security levels of all classes.

For example, we can consider the following rule, related to the correct value of the tagged values, and express it using OCL: ‘The set of user roles defined for each class and attribute of the model has to be a subtree of the roles tree that has been defined for the model’.

```ocl
context Model
classifier inv self.classes-> forAll(c | c.Roles-> forAll(r | self.Role-> includesAll(r)))
classifier inv self.classes-> forAll(c | c.attributes-> forAll(a | a.Roles-> forAll(r | self.Role-> includesAll(r))))
```

2.6 OCL Extension

We will need some syntactic definitions that are not considered in standard OCL. Besides Set, OrderedSet, Bag and Sequence, we will need the `Tree` type. `Tree` type will be defined as a collection containing a root and a tree sequence. This type will be necessary to represent the user roles hierarchy. Consequently, the tree type will be able to use the operations of this collection defined by OCL and also the two new operations that are described below:

- Root: It will indicate the tree root.
Subtree(n): It will indicate the n subtree (starting from the left side) of the sequence of subtrees of a tree.

Trees can be described using complex OCL structures. However, we consider that there is a simpler representation way to define a new type of data collection. The new data type tree will not be used for modeling but it will be necessary later during the implementation of an automated tool that allows us to check OCL sentences.

This profile provides us with a series of aspects that will facilitate the use of our OCL extension. For example, it will be possible:
- To navigate, using the tagged values, in an intuitive way. This is possible due to the fact that tagged values are considered as attributes.
- To establish constraints by using UserProfile stereotype attributes. In this way, we will not only be able to refer to a contextual instance (writing “Self” first) but also to a contextual user (writing “UserProfile” first) thus limiting information depending on the characteristics of the user that is requesting that information.
- To model dynamic constraints, using security rules, authorization rules and audit rules. The context keyword will introduce the context of the expression, and the keywords secRule, auditRule and authRule denote respectively the stereotype «securityRule», «AuditRule», and «AuthorizationRule» of the constraint.

3 An Example applying our profile

We have considered a reduced example in order to focus our attention on security specifications. Our SecureModel, named ‘Hospital’ is based on a typical health-care system. Given SECDW profile, Figure 4 shows us how this profile has been applied to the package ‘Hospital’. Applying SECDW profile means that it is allowed, but not necessarily required, to apply the stereotypes that are defined as part of the profile.

Figure 5 shows us the secure multidimensional model Hospital whose patients admission is composed of a fact class named Admission, dimension classes called Diagnosis, Patient and Time, and base classes named Diagnosis_group of Patient Dimension. Additionally, in this modeling, an additional class called UserProfile is considered (stereotype UserProfile), that will contain information of all users entitled to access to this multidimensional model (that will be possible to be used as a contextual user in the specification of our constraints with OCL).

We have used the following security levels: Confidential, Secret and topSecret. User roles Health (including Doctor and Nurse subroles) and NonHealth (including Maintenance and Administrative subroles) have been defined. The root of this
hierarchical roles tree is HospitalEmployee. In this example, we have not considered organizational compartments.

In figure 5, we can see that, in our model, we use the classes stereotypes inherited from the proposal stated in [13], which we have added security aspects into (secureFact, secureDimension, secureBase representing them with the same icons but adding them a letter “S” indicating that is a secure class). At the same time, all our constraints (AuditRule, AuthorizationRule and SecurityRule) will be modeled using UML notes.

**Fig. 5.** Example of secure multidimensional modeling

1. The security level of each instance of Admission is defined by a security constraint specified in the model. If the value of the description attribute of the Diagnosis_group to which diagnosis belongs is cancer or AIDS, the security level –tagged value SL- of this admission will be top secret, otherwise secret. This constraint is only applied if the user makes a query whose information comes from Diagnosis dimension or Diagnosis_group base classes, together with Patient dimension –tagged value involvedClasses-. Therefore, a user who has secret security level could obtain the number of patients with cancer for each city, but never if information of Patient dimension appears in the query.

2. For confidentiality reasons, we could deny access to admission information to users whose working area is different than the area of a particular admission instance. This is specified by another exception in Admission fact class, considering a condition and the tagged values involvedClasses, exceptSign.

3. The tagged value logType has been defined for Admission class, specifying the value frustratedAttempts. This stereotype specifies that the system has to record,
for future audit, the situation in which a user tries to access to information whose type is ‘primary diagnosis’ of this fact class, and the system denies it because of lack of permissions.

4. The security level –tagged value $SL-$ of each instance of $Admission$ can also depend on the value of $cost$ attribute, which indicates the price of the admission service. In this case, the constraint is only applicable to queries that contain information of the $Patient$ dimension –tagged value $involvedClasses$.

5. Users can be denied access to patients data that have been treated before the date of contract of the staff in the health area. This stereotype is specified with an exception in the Admission class, considering a condition and $InvolvedClasses$ and $ExceptSign$ tagged values.

6. Patients could be special users of the system. In this case, it could be possible that patients access to their own information as patients (for instance, for querying their personal data). This constraint is specified by using the $exceptSign$ tagged value in the $Patient$ class.

4 Conclusions and Future Work

In this paper, we have presented a UML 2.0/OCL profile that allows us to represent the main security aspects in the conceptual modeling of data warehouses. This extension contains the necessary stereotypes, tagged values and constraints for a complete and powerful secure MD modeling. These new elements allow us to specify security aspects such as security levels on data, compartments and user roles on the main elements of a MD modeling such as facts, dimensions and classification hierarchies. We have used the OCL to specify the constraints attached to these new defined elements, thereby avoiding an arbitrary use of them.

Taking into account that data warehouses are used for discovering crucial business information in the strategic decision making process, this proposal provides as with interesting advances to improve security in decision support systems as well as sensitivity information protection that these systems generally manage.

Our immediate future work consists of developing an automated tool that allows us not only to model data warehouses in a secure way using our profile but also to translate as well validate all our OCL sentences specified in our modeling. Furthermore, our proposal will be tested in a real environment in order to get experience and get results of his efficiency.

Acknowledgements

This research is part of the RETISTIC (TIC2002-12487-E) and the MESSENGER (PCC-03-003-1) projects, supported by the Dirección General de Investigación of the Ministerio de Ciencia y Tecnología, and the network VII-J.RITOS2 financed by CYTED respectively, and the METASIGN project (TIN2004-00799) supported by the CICYT.
References