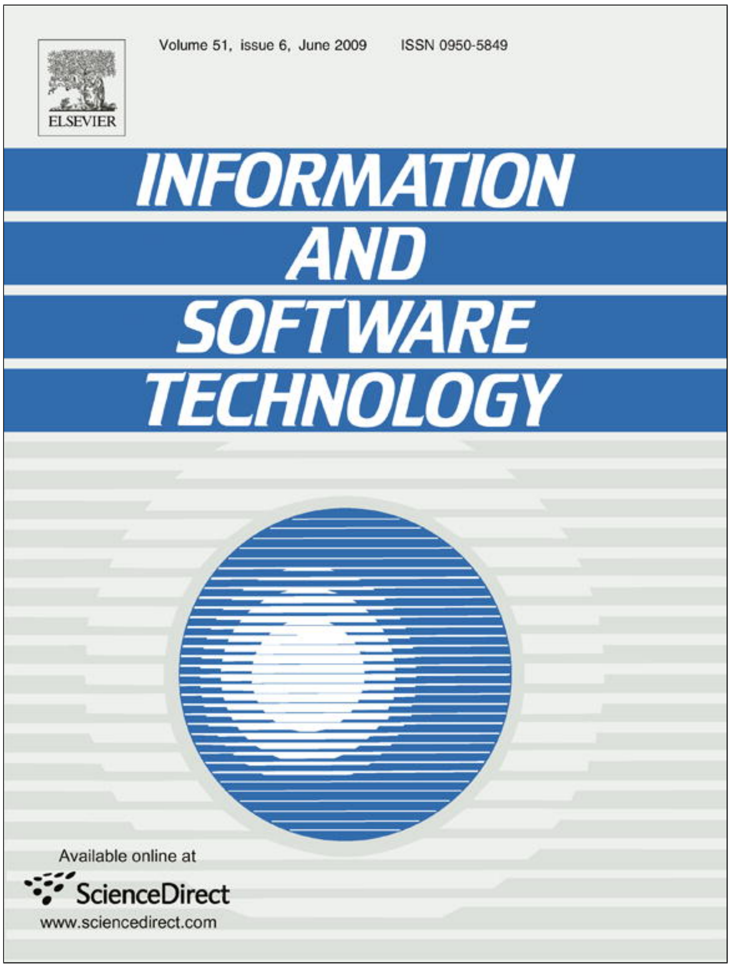


Volume 51, issue 6, June 2009 ISSN 0950-5849

ELSEVIER

INFORMATION AND SOFTWARE TECHNOLOGY

Available online at
ScienceDirect
www.sciencedirect.com

Information and Software Technology 51 (2009) 1033–1051

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

# An engineering process for developing Secure Data Warehouses

Juan Trujillo [a], Emilio Soler [b], Eduardo Fernández-Medina [c,*], Mario Piattini [c]

[a] LUCENTIA Research Group, Department of Software and Computing Systems, University of Alicante, C/San Vicente S/N, 03690 Alicante, Spain
[b] Department of Computer Science, University of Matanzas, Autopista de Varadero km 3, Matanzas, Cuba
[c] ALARCOS Research Group, Information Systems and Technologies Department, UCLM Research and Development Institute, University of Castilla-La Mancha, Paseo de la Universidad 4, 13071 Ciudad Real, Spain

ARTICLE INFO

*Article history:*
Received 4 February 2008
Received in revised form 5 December 2008
Accepted 10 December 2008
Available online 25 February 2009

*Keywords:*
Data Warehouses
Security requirements engineering
Access control
*i*^*

ABSTRACT

We present a new approach for the elicitation and development security requirements in the entire Data Warehouse (DWs) life cycle, which we have called a Secure Engineering process for DAta WArehouses (SEDAWA). Whilst many methods for the requirements analysis phase of the DWs have been proposed, the elicitation of security requirements as non-functional requirements has not received sufficient attention. Hence, in this paper we propose a methodology for the DW design based on *Model Driven Architecture* (MDA) and the standard Software Process Engineering Metamodel Specification (SPEM) from the Object Management Group (OMG). We define four phases comprising of several activities and steps, an d five disciplines which cover the whole DW design. Our methodology adapts the *i*^* framework to be used under MDA and the SPEM approaches in order to elicit and develop security requirements for DWs. The benefits of our proposal are shown through an example related to the management of the pharmacies consortium business.

## 1. Introduction

Data Warehouses (DW) systems represent a single source of information through which to analyze the status and the development of an organization [17]. This fact shows the need to define and enforce security measures throughout the entire DW development process. A DW is usually designed by following the phase requirements analysis along with conceptual, logical and physical design.

In recent years, many proposals concerning how DWs should be designed have been proposed [1]. Some authors suggest that there should be a phase dedicated to requirements analysis [11,17,21,36]. But few of them take into account both, functional and security aspects as non-functional requirements for the DW design.

Normally, within DWs projects the security aspects are implemented in the final phases of design [42]. However, some authors have noted that software engineering projects are critically vulnerable when security requirements are performed poorly during the earlier phases of the development process [4,30,9].

Security requirements are those requirements associated with the protection of valuable assets in the system. These security requirements describe how access is managed, what information can be accessed by whom, and under what conditions that information can be accessed, and they are thus often called Access Control Policies (ACP) [8]. The relevant literature comprises many proposals whose aim is to protect DWs. However, we put forward the works [7,6,51] in which the authors call for the design of security in the entire DW life cycle. These works constitute a sole proposal which allows us to establish security and audit measures for multidimensional conceptual modeling by means of a semantical model. The approach is based on the Access Control and Audit (ACA) model, which constitutes an ACP tailored for secure DW design. ACP specifications are often conducted without prescriptive guidance [3,46], thus leaving systems vulnerable to breaches in security and privacy [13]. Many researchers have, therefore, recognized the need to bridge the gap between requirements analysis and access control specification [3,12].

On the other hand, the proposals of [7,6,51] have been related with *Model Driven Architecture* (MDA) [31] an Object Management Group (OMG) standard which addresses the complete life cycle of designing, deploying, integrating, and managing applications. MDA is based on the creation of design models at different levels, and the transformations between them. Fig. 1 describes our framework based on the MDA for the development of secure DWs by [47]. Previous works show that the conceptual, logical and physical levels, along with the necessary transformations through the use of *Query Views and Transformation* (QVT) [34] have been defined and studied (see [51,48–50] for more details). However, a systematic engineering process with which to develop secure DWs that couples with the framework proposed in Fig. 1, i.e., that allows us to build and transform the Computation Independent Model (CIM) to obtain secure code is still lacking.

[*] Corresponding author. Tel.: +34 926295300; fax: +34 926295354.
*E-mail address:* eduardo.fdezmedina@uclm.es (E. Fernández-Medina).

0950-5849/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2008.12.003

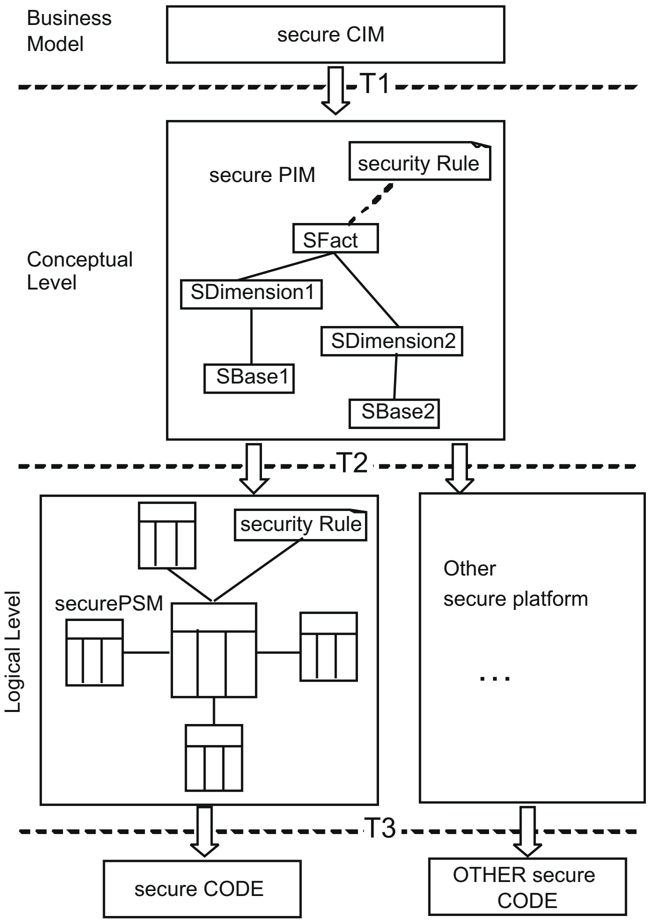


**Fig. 1.** A framework for the design of secure DWs.

The elaboration of CIM is especially important and critical, since this model collects the majority of the user's requirements which, once transformed, will be given to other models from the MDA framework, i.e., the Platform Independent Model (PIM) is obtained from the CIM by using a set of guidelines that we have defined (see Section 4.4.1), and the Platform Specific Model (PSM) and the code for a specific Database Management System (DBMS) are also defined by applying both the QVT [34] and *MOF2Text* [35] proposals, respectively. These models are ideally obtained in an automatic manner, together with different enrichments at different points of the MDA process. Hence, we call for the definition of an engineering process that reuses and couples our previous results under the MDA prism and facilitate the implementation of a CASE tool that will generate the secure DW repository.

In this paper we propose a comprehensive methodology with which to develop secure DWs based on the MDA framework which we will call Secure Engineering process for DAta WArehouses (SEDAWA). The process allows us to define security requirements from the business level which are transformed throughout the entire DW life cycle. The approach is coupled with both, the MDA framework and with our previously developed works. Our approach is based on the standard Software Process Engineering Metamodel Specification (SPEM) [32] from the Object Management Group (OMG). SPEM is a process metamodel that allows us to describe process engineering taking into account the life cycles of the projects. The major benefits of our proposal are: (i) the adaptation of the $i^*$ framework [53] in order to elicit and represent both, functional requirements and security as non-functional requirements into a secure CIM for the requirements analysis stage of the DWs design, (ii) that the elicited requirements be transformed and developed throughout the entire DWs life cycle by using the MDA framework, (iii) that the methodology be based on SPEM, which assures a standard notation that couples with the Unified Modeling Language (UML), and (iv) that our approach offers a guidance through which to develop an effective automated support to build secure DWs. It therefore reduces the risk, the development time and effort in the development of secure DWs.

Moreover, the methodological approaches are best suited to making the software process more systematic and predictable if they take into account quality control aspects. Therefore, the final software product is more suitable to the user's requirements, providing greater quality, less cost and greater ease of maintenance, etc., to the applications. The remainder of this paper is structured as follow. After this introduction, Section 2 describes the related work. The security modeling for DWs and its relationship with MDA is shown in Section 3. Section 4 presents our secure engineering process by means of a methodology which allows us to elicit and develop security requirements in the entire DW life cycle. Our methodology is illustrated with an example related to the management of the pharmacies consortium business in Section 5, which have been implemented as a prof. of concept in a prototype CASE tool that we are developing. The limitations of our proposal are treated in Section 6. Finally, Section 7 draws our main conclusions and outlines our immediate future work.

## 2. Related work

In this section, we describe work related to our approach from three perspectives: (1) requirements for DWs, (2) the engineering process for DWs and (3) proposals for security modeling in DWs.

### 2.1. Requirements for DWs

Three kinds of DW development approaches exist: data-driven (from an operational system into DWs), requirement-driven (an attempt to identify the information needs for the DWs, i.e., based on an explicit requirement stage) and mixed (which considers both data sources and requirements in the early stages of development). Here, we have not classified the different approaches according to the three kinds of DW development mentioned above. In general, the main approaches for requirement analysis in the context of DWs are [10,24,26,52,36,38]. However, the proposal of [36] alone considers security as non-functional requirements (NFRs) for the Requirement Analysis stage. The remainder of the proposals deal with functional requirements. However, the security requirements elicited are not developed and transformed in the entire DW life cycle. Moreover, none of the aforementioned approaches are part of the general engineering process that allows us to obtain secure DWs. We, however, take as our starting point the work [26] in which the authors proposed a novel and promising approach towards the definition of informational requirements.

### 2.2. Engineering process for DWs

Several methodologies with regard to how DWs must be built have been proposed in literature. In [18] different case studies of Data Mart (DM) are presented which integrate the design of several DMs by means of the BUS matrix architecture. The most relevant phases within the DW life cycle is presented, but a method for the entire process is not proposed. In [11] the authors propose a particular notation for the DW conceptual design and how a DW schema can be derived from data sources described by Entity Relationship (ER) schemas. Here, the authors assume a relational implementation of the DWs and the existence of ER schemas, which is often impossible. [29] shows how to build a star schema (and its different variations) from the conceptual schemas of the operational data sources. The approach supposes that the data sources are defined by means of ER schemas. MIDEA [2] constitutes a method based on an MD model. A set of steps that address the conceptual, logical and physical design of the DWs is proposed. The operational data sources, together with the final user's requirements, are considered in the design. In [22] a DWs development method, based on UML and the Unified Process (UP) is proposed, which addresses the design and development of the entire DW design. The approach is based on formal UML profiles and allows us to define various DWs schemas that can be integrated in order to define automatic transformation between them.

All of the aforementioned approaches are affected by some of the following problems: they do not include security issues in the DWs design, some of them do not cover the entire DW life cycle, some of them are based on specific implementation (the star schema in relational databases), or they are based on a set of steps or phases that permit the automatization of the entire process of the secure DW design.

### 2.3. Security in Data Warehouses

The security modeling for DWs comprises several initiatives to include security in the DW design. In [15] the authors describe a prototype model for DW security based on metadata, whose main goal is to reduce user queries to only those data which are to be seen by that user. However, this does not permit the specification of complex restrictions of confidentiality such as deny-allow access to a special user combining groups and security constraints. Rosenthal and Sciore [41] extend SQL grants and create a mechanism of inferences through which to establish DW security, which derives permission on tables and views of the system, thus establishing easy administration. A further attempt is that of the architecture for both Federated Information Systems (FIS) and DWs which preserves MultiLevel security integration between FIS and DWs [43]. The authorization of the DW scheme built takes into account the security policy of the federation itself. Kirkgöze et al. [16] defines a model based on the Discretionary Access Model (DAC) which propose a security concept for OLAP, a role based security model for DWs. According to these security rules, a derived data cube is defined for each role. Essmayr et al. [5] shows how access privileges for DWs and OLAP can be expressed more intuitively than by using SQL's grant statements. This access control model focuses specifically on expressiveness and usability. These approaches [5,15,16,41,43] are attractive but only focus on practical issues such as acquisition, storage and access control on the OLAP side. None of them examine the representation of security at the Requirement Analysis stage nor do they propose a method to develop it throughout the DW life cycle.

More elaborate initiatives which propose authorization models for DWs design also exist. For example, Priebe and Pernul [39] propose a security design methodology similar to the classical database methodology (requirement analysis, conceptual, logical, and physical design) which covers requirements and concrete implementations in commercial systems. The same authors extend the ADAPTed UML (which uses ADAPT symbols as UML stereotypes) model for the aforementioned conceptual phase [40], and specify a methodology and an MD security constraint language for the conceptual modeling of OLAP security. These approaches [39,40] offer security models at the conceptual level by means of security constraints, but basically deal with OLAP operations. In short, these works implement the security rules considered in their conceptual approach to commercial database systems. The proposed methodology focuses solely on the conceptual stage of the DW life cycle, other stages are not taken into account and no method exists with which to perform it.

In conclusion, none of the existing approaches model security requirements which can be developed throughout the DW life cycle. None of them consider a formal access control that can be defined in the requirement analysis stage, and transformed and enriched throughout the DWs life cycle. We believe that none of the aforementioned approaches constitute a general method that offers an effective guidance for the development of an automatic support with which to build secure DWs. We therefore base our approach on the work of [7] in which the authors propose a novel model for security and audit at the conceptual level for DWs. This model will be adapted and transformed to take on security and audit measures in the entire DW life cycle (i.e., from the requirement analysis phase to the final implementation in a specific DBMS) following the standard SPEM metamodel [32].

## 3. Modeling security for Data Warehouses

One of the main concerns in DW design is data security, which is usually seen as a non-information requirement [36]. In this section, we focus on explaining how to model security requirements for DWs by means of the Access Control and Audit (ACA) model. Moreover, we reuse a previous extension of the $i^*$ framework [26] in order to elicit security requirements and to define an ACA model for DWs at the business level.

Security requirements are requirements which are associated with the protection of valuable assets in the system. This protection requires that every access to a system and its resources be controlled and that only authorized access can take place, and is thus called Access Control (AC) [44]. The development of an access control system is usually carried out by access control policies (ACP), access control models and an access control mechanism [44], which constitute different levels of abstraction. ACP are security requirements which defines high-level rules. Access control models provide a formal representation of the access control security policy, whereas the access control mechanism defines the low-level (software and hardware) functions that implement the controls imposed by the policies and are formally stated in the model [44].

ACP are grouped into three main classes: Discretionary Access Control (DAC), Mandatory Access Control (MAC) and Role-Based Access Control (RBAC) [44]. DAC policies control access based on the identity of the requestor and on access rules which state what requestors are allowed to do. MAC policies control access based on mandated regulations determined by a central authority. RBAC policies control access depending on the roles that users have within the system and on rules stating what accesses are permitted to users in given roles.

In previous work we have defined an Access Control and Audit (ACA) model for DWs by specifying security rules at the conceptual level [7]. This approach is based on access control to guarantee confidentiality and audit, which are essential components in the DW design. However, security includes other characteristics such as authentication, integrity, repudiation and availability, which constitute mechanisms that are design-independent and rely to a greater extent on company policies. They are not therefore taken into account by the ACA model. The ACA model allows us to represent the confidentiality and audit measures of DWs by classifying subjects and objects in the system.[1] The classification uses access classes on the basis of three different but compatible ways of classifying users: by their *security level*, by their *role*, and by the *compartments* to which they belong. The access class is one element of a partially ordered set of classes, in which an access class $c_1$

[1] The ACA model also allows us to define Sensitive Information Assignment Rules (SIARs) in order to specify the security information of each DW element, rules for representing authorization rules (AURs), which work together with SIARs, and rules which allow us to specify audit requirements (ARs).

dominates an access class $c_2$ *if and only if* the security level of $c_1$ is greater than or equal to that of $c_2$, the compartments of $c_1$ include those of $c_2$, and at least one of the user roles of $c_1$ (or one of its ancestors) is defined for $c_2$ [7].

The following classes are described in order for us to be able to specify the ACA model:

*Security user roles* are used by a company to organize users into a hierarchical role structure, according to the responsibilities of each type of work. Each user can play more than one role.

*Security levels* indicate the clearance level of the user. This is usually an element of a hierarchically ordered set, such as Top Secret (TS), Secret (S), Confidential (C), and Unclassified (U), where TS > S > C > U.

*Security user compartments* are also used by an organization to classify users into a set of horizontal compartments or groups, such as geographical location, area of work, etc. Each user can belong to one or more compartments.

In order to illustrate the previous concept we need to consider within the organization the Security levels *TopSecret* (TS) and *Secret* (S) and the Security Compartments *Asia* (SC = A), *Europe* (SC = E), *South Europe* (SC = SE), and *North Europe* (SC= N E). Moreover, we need the objects $O_1$, $O_2$ and $O_3$ with the following security information: $O_1$\{SL = S; SC = NE, SE\}, $O_2$\{SL = TS; SC = SE\} and $O_3$\{SL = TS; SC = A, NE\}. A user of the system is denoted by U with the following associated security information: U\{SL = TS; SC = E\}. According to the aforementioned definition the U access class dominates the access class of $O_1$ and $O_2$, but it does not dominates the access class of $O_3$, because there is a compartment of $O_3$ that does not belong to the compartments of U.

As was previously explained, the ACA model uses the classification for users and object based levels, roles and compartments. Therefore, the ACA model combines the MAC and RBAC models. MAC models have been widely studied, and many vulnerabilities have been detected, such as their lack of flexibility, their polyinstantiation [14], etc. Nevertheless, most of these problems arise from the necessity of taking into consideration both read and write operations in the system. Fortunately, we consider that the sole operation that will be used by the final users in decision-support systems is read, so the MAC model is absolutely appropriate. In contrast to the MAC model, the RBAC model represents a promising direction and a useful paradigm for many commercial and governmental organizations [44].

The ACA model has been integrated with the UML profile of [23] in order to conform a profile with which to classify both security information and user for the design of secure DWs. The profile allows us to classify information that will be used to represent the main security issues in the conceptual modeling of DWs.

### 3.1. Aligning secure Data Warehouse design with MDA

Model Driven Architecture (MDA) is a standard from the Object Management Group (OMG) which addresses the complete life cycle of developing applications by using models in software development. MDA relies on the idea of separating the specification of a system operation from the details of its platform [31]. MDA proposes several models at different levels: the Computation Independent Model (CIM), the Platform Independent Model (PIM), the Platform Specific Model (PSM) and Code. In the MDA framework the standard for defining transformations between previous models is Query/Views/Transformation (QVT) [34].

Fig. 2 shows the extensions proposed in order to establish a relationship between the DW life cycle and the MDA framework. The secure CIM model takes into account the work by [26], which is based on an extension of the $i^*$ framework [53]. In the following section we adapt this extension to represent both functional and non-functional requirements for DWs. Secure PIM corresponds to an extension of the Unified Modeling Language (UML) profile by [23] presented in [51]. This profile allows us to consider the main properties of secure MD modeling at the conceptual level. The Secure PSM corresponds with an extension of the Common Warehouse Metamodel (CWM) at the logical level [48] and secure Code corresponds with implementation at the physical level, i.e., with a DBMS that implements security issues. We have recently used the metamodels of [51,48] to define QVT relations in order to transform secure PIM into secure PSM in the design of secure DWs [49]. This set of QVT relations has been validated by means of the development of the case study of [50]. In the following section, we shall use the application of the modeling phase to explain how to map secure CIM into secure PIM by applying a set of guidelines shown in Section 4.4.1.

## 4. SEDAWA: Secure Engineering process for DAta WArehouses

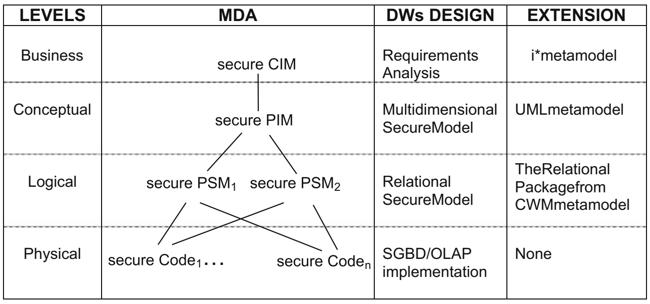
The secure engineering process is proposed with the purpose of defining secure requirements and transforming them in order to develop secure Data Warehouses. The proposal integrates the entire DWs life cycle taking into account its relationship with the MDA framework depicted in Fig. 2. This section describes our methodology, which is based on the standard Software Process Engineering Metamodel Specification (SPEM) [32] from the Object Management Group (OMG).

### 4.1. SPEM metamodel: description and notation

SPEM is a process metamodel used to describe a concrete software development process or a family of related software development process. The SPEM specification is structured as a UML profile, and provides a complete MOF-based metamodel [32].

The SPEM metamodel offers the constructs and semantics required for the software development process, which involve or require the use of Unified Modeling Language (UML), such as the Rational Unified Process (RUP). RUP is therefore a software engineering process that is iterative, architecture-centric, and use-case-driven [20]. RUP currently captures many of the best practices in modern software development, since it is highly suitable for a wide range of projects and organizations. However, we do not instantiate RUP since it has some disadvantages for our purposes. For example, requirements in RUP are captured in a textual document called the Vision Document, unlike our proposal which uses the goal/softgoal modeling diagram. Moreover, we describe how security requirements are elicited and modeled in early activities and developed in later phases of the DWs development process throughout a methodology. RUP, however, only presents a set of guidelines to capture and employ security requirements.

The SPEM stand-alone metamodel is built by extending a subset of the UML metamodel. Fig. 3 depicts part of the SPEM metamodel



| LEVELS | MDA | DWs DESIGN | EXTENSION |
|---|---|---|---|
| Business | secure CIM | Requirements Analysis | i*metamodel |
| Conceptual | secure PIM | Multidimensional SecureModel | UMLmetamodel |
| Logical | secure PSM$_1$ secure PSM$_2$ | Relational SecureModel | TheRelational Packagefrom CWMmetamodel |
| Physical | secure Code$_1$... secure Code$_n$ | SGBD/OLAP implementation | None |

**Fig. 2.** Aligning the design of secure DWs with MDA.

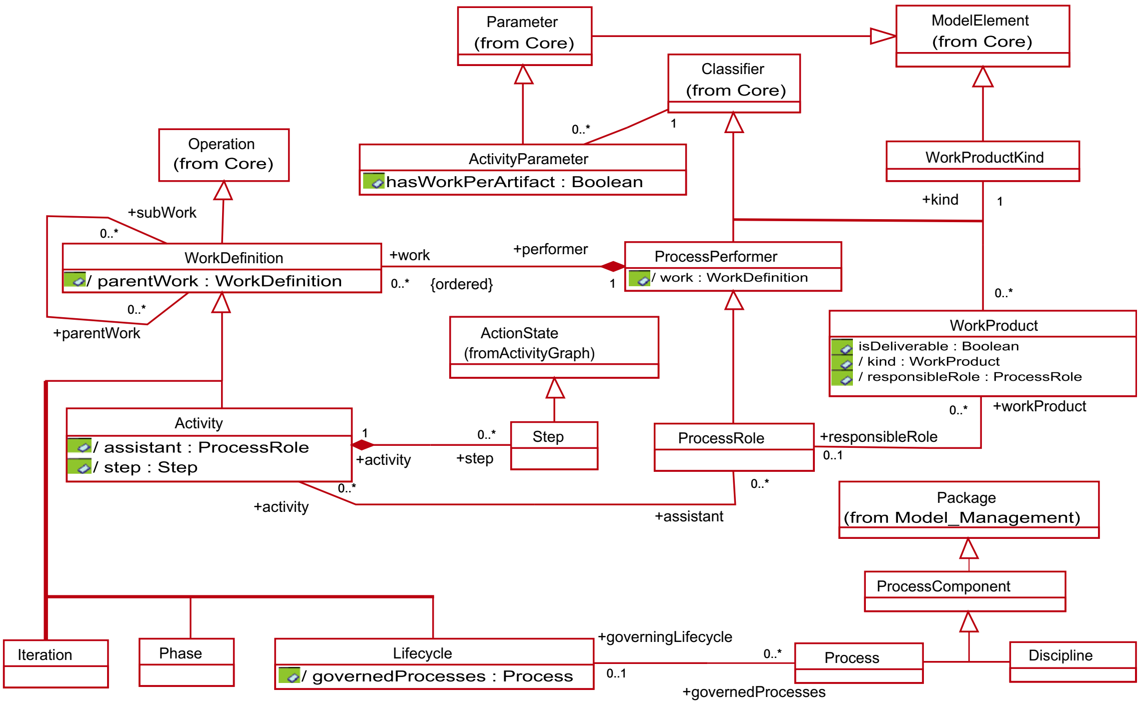


**Fig. 3.** Fragment of SPEM metamodel employed.

that we will use in our engineering process, which is supported by the *Core* and *ProcessComponent* packages.

The main SPEM classes that inherit from the *Core* package's classes are: *WorkDefinition* (∞), which describes the work performed in the process. Its main subclass is *Activity*, but *Phase*, *Iteration*, and *Lifecycle* are also subclasses of *WorkDefinition*. *Activity* (〉) describes a piece of work performed by one *ProcessRole*, which may consist of atomic elements called *Steps*. *ProcessRole* (☺) defines responsibilities and roles over specific *WorkProducts* that perform and assist in specific activities. *WorkProduct* (▲■) or artifact is anything produced, consumed, or modified by a process (a piece of information, a document, a model, source code, etc.). A *Phase* (»») is a specialization of *WorkDefinition* such that its precondition defines the phase entry criteria and its goal (often called a "milestone") defines the phase exit criteria. A process *Lifecycle* is defined as a sequence of *Phases* that achieve a specific goal. An *Iteration* is a composite *WorkDefinition* that represents a set of *Activities* focusing on a portion of the system development that results in a release (internal or external) of the software product. See Fig. 3 for more details.

The main SPEM packages that inherit from the *ProcessComponents* package are: *Package* (just as in UML), which is a container that can both own and import process definition elements. A *Process* (◎) is a *ProcessComponent* which is intended to stand alone as a complete, end-to-end process. *Discipline* is a particular specialization of *Package* that partitions the *Activities* within a process according to a common "theme" (see Fig. 3).

In the sequel we present an overview of the SEDAWA methodology. The following sections give a detailed description of the four phases that we have considered in our methodology. In each section we define activities, steps, and work products, which will be characterized according to the discipline that they belong to.

### 4.2. An overview of SEDAWA

SEDAWA is structured into four consecutive phases: elicitation, modeling, implementation and test–delivery. The iterative style should be applied to the phases of our methodology. We define five disciplines: requirements analysis, conceptual design, logical design, physical design and post-development review, a new discipline introduced by Luján and Trujillo [22].

Fig. 4 illustrates our SEDAWA methodology. We use the standard icons from SPEM [32], i.e., *Phase* (»»), *WorkProduct*[2] (▲■) and *Activity* (〉). We omit certain elements from the SPEM metamodel in the figure for reasons of better understanding. The engineering process is described along the two axis. The horizontal axis represents time and the dynamic aspect of the process expressed in terms of phases and iterations. The vertical axis represents the static aspect of the process described by disciplines which cover the entire DW life cycle, described in terms of *Activities*, *WorkProducts*, *Steps*, etc.

The engineering process begins with the Enterprise Architecture *WorkProduct* as input for activity A1.1. The Enterprise Architecture contains designs of the business processes, organizational structures, components, physical resources, products and services from the organization. This *WorkProduct* can be used, by applying activities A1.1, A1.2 and A1.3 from the Elicitation phase to obtain three models: (1) *GOModel* which contains informational requirements for DWs, i.e., functional requirements; (2) *SOModel* which contains security requirements for DWs; and (3) *GSAModel* which merges the above models and constitutes a secure CIM for DWs. The Modeling phase is conducted by activities A2.1, A2.2. Activity A2.1 receives as input the *WorkProducts GSAModel* and Secure MD metamodel (whose instance will be produced, see the metamodel shown in Appendix A). Activity A2.2 receives as input the MD model *WorkProduct* obtained from activity A2.1 and the operational sources *WorkProduct* that will serve to populate the secure DWs repository. The implementation phase is carried out by activity A3.1, which receives as input the enriched secure MD model *WorkProduct* obtained from activity A2.2. In addition A3.1 receives the SECure Relational Data Warehouses (SECRDW) metamodel (whose

[2] A WorkProduct icon will be represented by one or several WorkProducts.

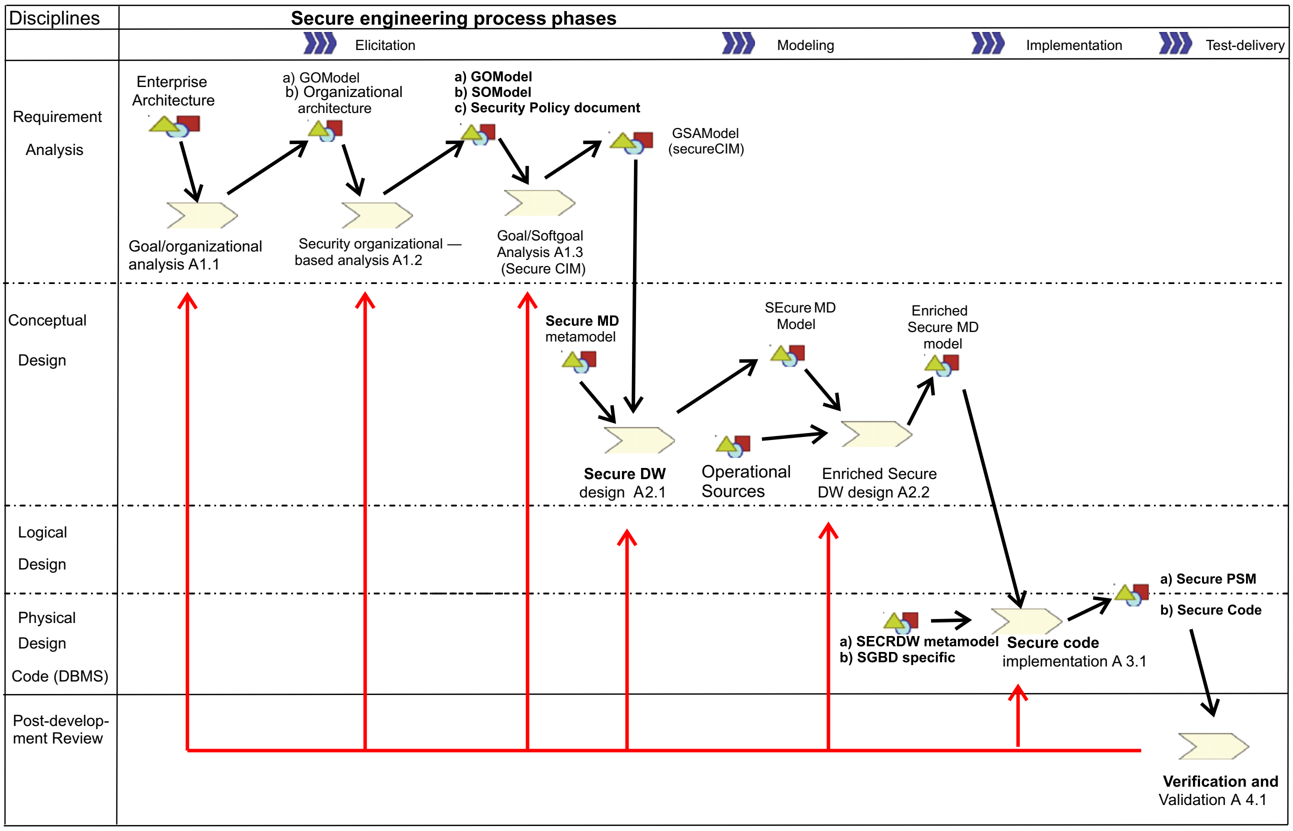


**Fig. 4.** Secure engineering process for DWs.

instance will be produced, see the metamodel shown in Appendix B) and the DBMS specific (which will implement the secure DWs repository) *WorkProducts*. Finally, the test–delivery phase contains the activity A4.1 in order to verify, test and deliver the secure DW repository.

According to SPEM our methodology is described by using the structure shown in Fig. 5 (see class description from SPEM explained in Section 4.1). Each activity specifies *WorkProduct* as both input and output, respectively. A more detailed explanation of each phase is described in the following subsections.

### 4.3. Elicitation phase

The elicitation phase comprises three activities: Goal/organizational analysis (A1.1), which receives as input the *WorkProduct* Enterprise Architecture. Security organizational-based analysis (A1.2) receives as input the outputs from activity A1.1, i.e., the Goal organization model (*GOModel*) and Organizational architecture *WorkProducts*. Finally, Goal/Softgoal analysis (A1.3) has as input *GOModel*, *SOModel* and the security policy document, which constitutes the *WorkProducts* output from A1.2. The output of activity A1.3 is the Goal/Softgoal analysis model (*GSAModel*), i.e., the secure CIM (see details in Fig. 4). In order to carry out the above activities several steps are performed for each activity, as we can see in Figs. 6, 8 and 10. During this phase the iterative style has strong emphasis on the requirements analysis discipline. See the *GOModel*, *SOModel* and *GSAModel* *WorkProducts* in Fig. 4. We must clarify that

Lifecycle: **Secure Data Warehouse**
Phase:
Process: **SEDAWA**
Subactivities
Iteration:
Subactivities
Activity:
ProcessRole:
ActivityParameters {kind:input}
WorkProduct:
ActivityParameters {kind:output}
WorkProduct:
Steps
Step:
Step:

**Fig. 5.** SEDAWA phases structure.

Phase: **Eliciting**
Process: **SEDAWA**
Subactivities
Iteration: **First iteration**
Subactivities
Activity: **Goal/organization alanalysis**
ProcessRole: **Requirement engineer**
ActivityParameters {kind:input}
WorkProduct: **Enterprise Architecture**
ActivityParameters {kind:output}
WorkProduct: **Goal organizational model (GOModel)**
WorkProduct: **Organizational architecture**
Steps
Step: **Determine DWs context**
Step: **Find the intentional actors**
Step: **Capturing the high-level enterprise goals**
Step: **Classify and define goals according to their level of abstraction**
Step: **Derive information alrequirement from information goals**
Step: **Obtain MD elements from information requirements**
Step: **Related context analysis with aggregations**

**Fig. 6.** Goal/organizational analysis activity (1.1).

an explanation of the goals that are elicited is not within the scope of this paper.

In order to define security requirements for the requirements analysis discipline we have adapted the $i^*$ framework which permits the modeling and reasoning of the organizational environment and its information system by using [53]. This establishes two main modeling components. The Strategic Dependency (SD) model is used to describe the dependency relationships among various actors in an organizational context. The Strategic Rationale (SR) model is used to describe stakeholder interests and concerns, and how they might be addressed by various configurations of systems and environments. However, for our purposes we only need SR models, which are depicted in Fig. 4 as the *GOModel*, *SOModel* and *GSAModel WorkProducts*.

### 4.3.1. Goal/organizational analysis (A1.1)

Activity Goal/organizational analysis (A1.1) (see Fig. 6) is performed by the *ProcessRole* requirement engineer. The input is the enterprise architecture (to discover the domain infrastructure). Several steps are necessary to achieve its outputs (*WorkProducts GOModel* and Organizational architecture). *GOModel* represents the informational requirements for DWs, i.e., the functional requirements.

Activity A1.1 is based on the work by [26], in which the authors adapted the $i^*$ framework [53] to the modeling of goals and information requirements for DWs. The approach is supported by a UML profile [33] which is integrated with the Model Driven Architecture (MDA) framework of [25]. The adaptation of the $i^*$ framework is based on two extensions of UML [33]: (i) a profile for $i^*$ and (ii) a profile which adapts $i^*$ to the DW domain. In accordance with the UML specification [33], in Fig. 7 we show the packages which resume the elements contained in proposal [26]. The profiles use two kinds of extending relationships: the *Extension* relationship (whose arrowhead is shown as a filled triangle) which points from stereotypes (the extending elements, labeled as ≪stereotype≫) to metaclasses (the UML extended elements, labeled as ≪metaclass≫), and the *Generalization* relationship (an arrowhead with a hollow triangle) between stereotypes. On the left hand side of Fig. 7 we have represented the $i^*$ profile by means of various UML metaclasses (i.e., *Package*, *Class*, *AssociationClass*, and *Association*) and stereotypes (the *IElement*, *Argumentable*, and *IRelationship* stereotypes). These stereotypes permit the representation of SR and SD models belonging to the $i^*$ framework.

On the right hand side of Fig. 7 we show the $i^*$ profile for DWs, which is based on a classification of the different kinds of goals that decision makers expect to fulfill with the DWs: (i) *Strategic goals* represent the highest level of abstraction. These are the main objectives of the business process (for example, "*increase sales*"); (ii) **Decision goals** represent the medium level of abstraction. They attempt to answer the question: "*how can a strategic goal be achieved*?" (for example, "*determine some kind of promotion*"); (iii) **Information goals** represent the lowest level of abstraction. They attempt to answer the question: "*how can decision goals be achieved in terms of information required*" (for example, "*analyze customer purchases*" or "*examine stocks*"). The profile reuses the previous stereotype *Goal*, as we can see in Fig. 7.

For decision makers, every goal must be specified according to the classification of goals in terms of the strategic-decision-information hierarchy. Information requirements (*Requirement* as *Task* on the right hand side of Fig. 7) for decision makers are derived from information goals. The profile has added three MD elements as resources: the business process to be analyzed (*BusinessProcess* stereotype), process measures under analysis (*Measure stereotype*), and context of analysis (*Context* stereotype). These stereotypes are, therefore, derived from Resource (see right hand side of Fig. 7).

The $i^*$ profile for DWs provides a mechanism with which to represent actors (*IActor*, ○) and their goals (*Goal*, ⬭). The information requirements of decision makers are considered as tasks (*Task*, ⬡), and the elements needed in the DW to provide such information are considered as resources (*Resource*, ▭). According to the type of DW element, these resources can be labeled as ≪BusinessProcess≫, ≪Context≫, or ≪Measure≫. We furthermore model relationships such as means-end (*MeansEnd*, —▷—) thus representing alternative means to fulfill goals, or tasks, i.e., the possible relationships are Goal–Goal and Goal–Task. Decomposition (*Decomposition*, —┼—) represents the elements which are necessary if a task is to be performed. Additionally, the profile allows us to define aggregation relationships between context of analysis (for instance, the city context can be aggregated by the country context). In order to model these relationships, we have used the (shared) aggregation relationship of UML (*Association* UML metaclass, represented as —◇).

### 4.3.2. Security organizational-based analysis (A1.2)

Once the functional requirements have been identified we need to define security requirements for DWs by means of the following activities. Fig. 8 depicts the Security organizational-based analysis activity, which is performed by the *SecurityExpert ProcessRole* and has as output the Softgoal Organizational Model (*SOModel*) and the security policy document.

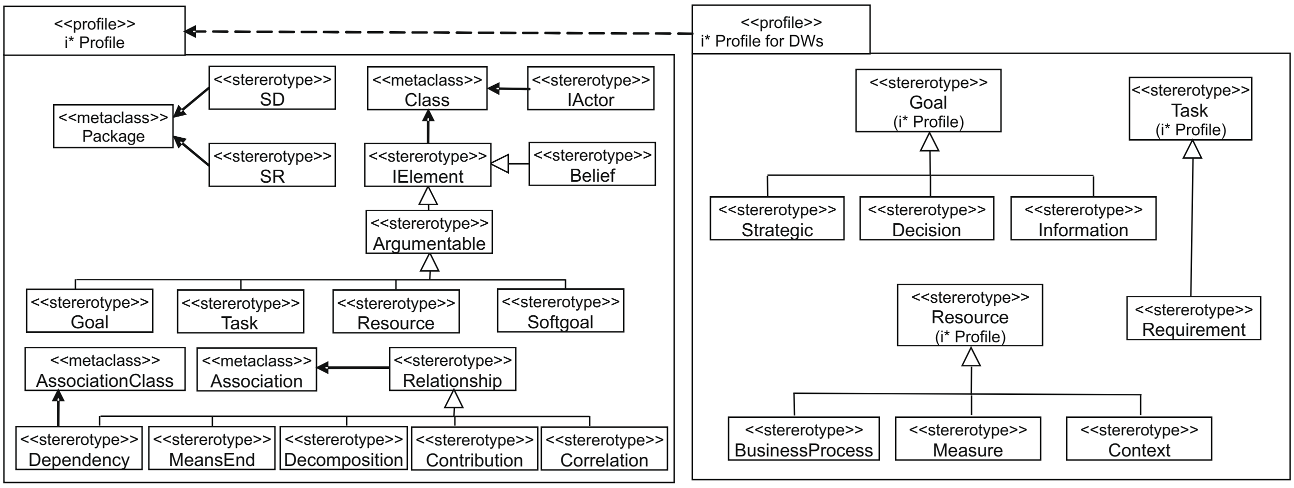


**Fig. 7.** UML profile for $i^*$ in the context of DWs.

*SOModel* is built with an adaptation of the *i** framework in order to represent security requirements for DWs based on the Access Control and Audit (ACA) model. Fig. 9 depicts how the adapted profile reuses stereotypes from the *i** for DWs.

The proposed extension for *i** offers mechanisms with which to represent a special security manager actor (*SecurityManager*, ◯), who is the person in charge of the security within the organization. Softgoals (*SSoftgoals*, ⬭) are introduced to represent and refine the organization's security policy. The elements with which it is necessary to represent the ACA model are considered as resources (*Resource*, ▭) labeled as ≪`SCompartment`≫ (to represent the horizontal compartment or groups of users from the organization such as geographical localization ), ≪`SLevel`≫ (to define the clearance level, usually named *TopSecret*, *Secret*, *Confidential* and *Unclassified*) and ≪`SRole`≫ (to represent a role from the hierarchical roles defined within the organization). See right hand side of Fig. 9. Moreover, in order to specify constraints for resources, we introduce several tasks (*Task*, ⬡), which are labeled as ≪`SConstraintRule`≫ (to indicate additional constraints with regard to the multilevel security policies), ≪`SConstraintAudit`≫ (to indicate future audit) and ≪`SConstraintAuthorization`≫ (to indicate additional more elaborate constraints with regard to the access). These constraints represent rules that contribute to the fulfillment of *SSoftgoals* through the contribution link (*Contribution*, →). We model the refinement process of softgoals by means of the means-end link (*MeansEnd*, ▷). Finally, each of the softgoals which has been refined is detailed into resources (i.e., *BusinessProcess*, *Context*, or *Measure*) by means of *Decomposition* link (—+). We also have three packages (*GOModel*, *SOModel* and *GSAModel*). As was previously stated, *GOModel* contains the *i** model for DWs (i.e., functional requirements obtained from activity 1.1). *SOModel* contains the secure *i** model for DWs (i.e., security requirements) and the *GSAModel* contains *GOModel*, *SOModel* packages and their relationships.

### 4.3.3. Goal/Softgoal analysis (A1.3)

It is now necessary to mix the information requirements obtained from activity 1.1 with the *SSoftgoals* obtained from activity 1.2 in the Goal/Softgoal analysis activity (1.3). Fig. 10 shows the activity, which is defined through several steps. The output of this activity is the Goal/Softgoal analysis model (*GSAModel*), i.e., a model that mixes previous models. The activity needs the three *WorkProducts* obtained previously: the Goal organizational model, the Softgoal organization model, and the Security policy document.

- Phase: **Eliciting**
  - Process: **SEDAWA**
  - Subactivities
    - Iteration: **First iteration**
      - Subactivities
        - Activity: **Security organizational-based analysis**
          - ProcessRole: **Security expert**
          - ActivityParameters {kind:input}
            - WorkProduct: **Goal organization model**
            - WorkProduct: **Organizational architecture**
          - ActivityParameters {kind:output}
            - WorkProduct: **Soft goal organizational model**
            - WorkProduct: **Security policy document**
          - Steps
            - Step: **Examine policies, laws, rules and regulations**
            - Step: **Consider“organizational security policy”document**
            - Step: **Define security policy**
            - Step: **Determine the most relevant assets to be protected and vulnerabilities**
            - Step: **Deriving soft goals from organizational structures, main assetsandsecuritypolicy**
            - Step: **Identify and associate security soft goals with the Security Manager actor**
            - Step: **Refine soft goal into lower-level,i.e., more specific softgoals**
            - Step: **Discover responsabilities, task and levels that we use**
            - Step: **Define levels, roles and compartments as resources**
            - Step: **Associate soft goals refined with levels, roles and compartments according to SGModel**

**Fig. 8.** Security organizational-based analysis activity (1.2).

- Phase: **Eliciting**
  - Process: **SEDAWA**
  - Subactivities
    - Iteration: **First iteration**
      - Subactivities
        - Activity: **Goal/Soft goal analysis**
          - ProcessRole: **Requirement engineer**
          - ProcessRole: **Security expert**
          - ActivityParameters {kind:input}
            - WorkProduct: **Goal organizational model**
            - WorkProduct: **Soft goal organization model**
            - WorkProduct: **Security policy document**
          - ActivityParameters {kind:output}
            - WorkProduct: **Goal/Softgoal analysis model (secureCIM)**
          - Steps
            - Step: **Analyze the functionality of the main resources from GOModel**
            - Step: **Identify resources from GOModel and soft goals refined**
            - Step: **Define dependencies between actors in order to achieve softgoals**
            - Step: **Associate softgoals with resources obtained from GOModel (to assign Levels, Compartments and Roles)**
            - Step: **Detect new vulnerabilities from resources**
            - Step: **Determine additional constraints that resources must be need**
            - Step: **Identify and associate new constraints to softgoals refined**

**Fig. 10.** Goal/Softgoal analysis activity (1.3).

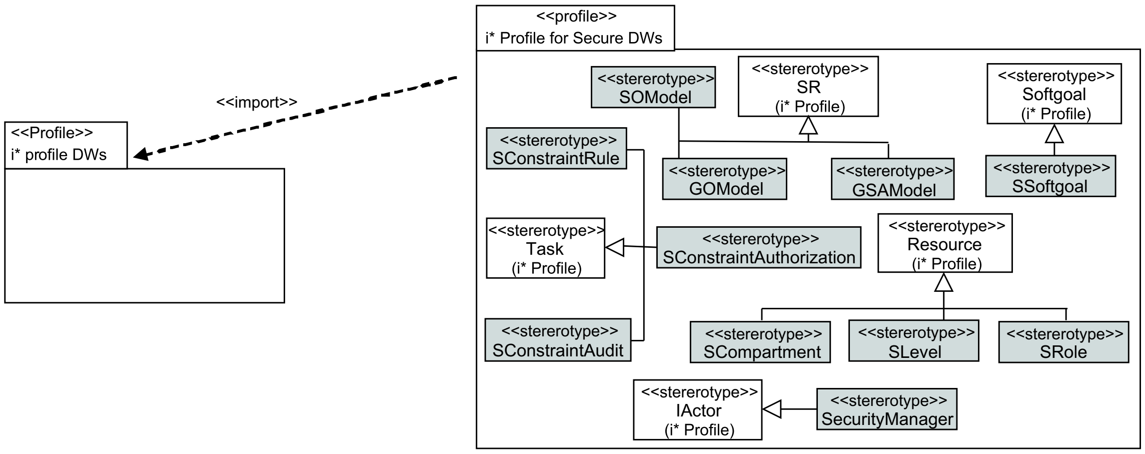


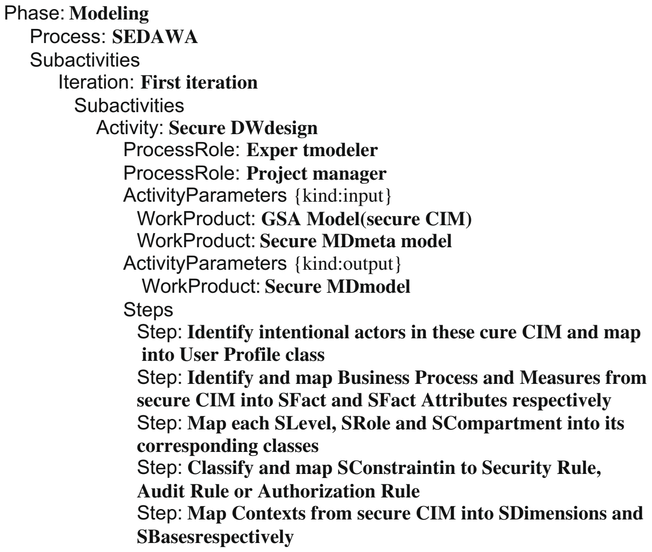
**Fig. 9.** Profile stereotypes with which to define security requirements.

### 4.4. Modeling phase

This subsection comprises two activities which cover the conceptual stage for the DW design. The modeling phase is carried out by the use of two activities. Both the information and the security requirements defined at the business level (i.e., secure CIM) must be transformed into the conceptual level, i.e., into the corresponding secure PIM.

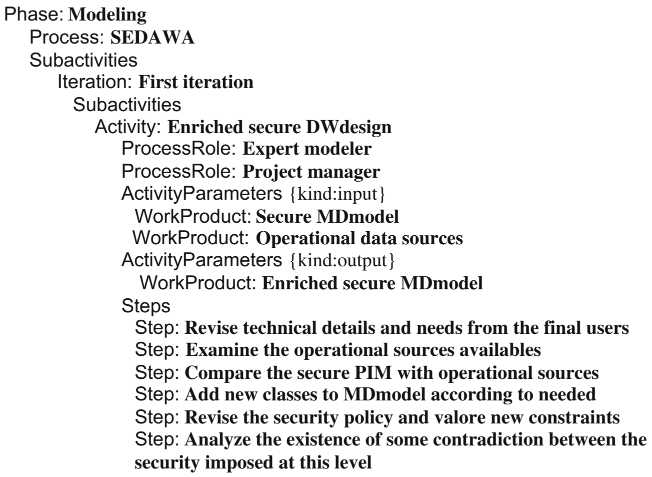
As was mentioned in the previous section, the PIM corresponds with an extension of the UML presented in [51], which the information is clearly organized into secure facts and secure dimensions. These secure facts and dimensions are modeled by *SFact* (represented as S) and *SDimension* (↗) stereotypes, respectively. *SFact* and *SDimension* are related through shared aggregation relationships (the *Association* UML metaclass) in class diagrams. While an *SFact* is composed of measures or secure fact attributes (*SFactAttribute* stereotype, **SFA**), with regard to *SDimensions*, each aggregation level of a hierarchy is specified by classes stereotypes such as *SBase* (B). Each *SBase* class can contain several secure dimension attributes (*SDimensionAttribute*, **SDA**) and must also contain a secure descriptor attribute (*SDescriptor* attribute, **SD**). An association stereotyped as *Rolls-upTo* (≪Rolls-UpTo≫) between *SBase* classes specifies the relationship between two levels of a classification hierarchy. Within this, role R represents the direction in which the hierarchy rolls up, whereas role D represents the direction in which the hierarchy drills down. The information about all users who are entitled to access the MD model are represented as instances of the *UserProfile* class (stereotype *UserProfile*, 👤). The UML metamodel which supports the secure PIM is shown in Appendix A.

This proposal allows us to classify both, information and users in order to represent the main security aspects in the conceptual modeling of DWs. Security information is defined for each element of the model (*SFact*, *SDimension*, *SFactAttribute*, etc.) specifying a sequence of security levels a set of user compartments and a set of roles. Moreover, the constraints (*AuditRule*, *AuthorizationRule* and *SecurityRule*) are modeled through the UML notes. These constraints are defined following the syntax of the ARs, AURs and SIARs rules from the ACA model (more details in [7,6,51]). We shall now present activities A2.1 and A2.2.



- Phase: **Modeling**
  - Process: **SEDAWA**
  - Subactivities
    - Iteration: **First iteration**
      - Subactivities
        - Activity: **Secure DWdesign**
          - ProcessRole: **Exper tmodeler**
          - ProcessRole: **Project manager**
          - ActivityParameters {kind:input}
            - WorkProduct: **GSA Model(secure CIM)**
            - WorkProduct: **Secure MDmeta model**
          - ActivityParameters {kind:output}
            - WorkProduct: **Secure MDmodel**
          - Steps
            - Step: **Identify intentional actors in these cure CIM and map into User Profile class**
            - Step: **Identify and map Business Process and Measures from secure CIM into SFact and SFact Attributes respectively**
            - Step: **Map each SLevel, SRole and SCompartment into its corresponding classes**
            - Step: **Classify and map SConstraintin to Security Rule, Audit Rule or Authorization Rule**
            - Step: **Map Contexts from secure CIM into SDimensions and SBasesrespectively**

**Fig. 11.** Secure Data Warehouse design activity (2.1).



- Phase: **Modeling**
  - Process: **SEDAWA**
  - Subactivities
    - Iteration: **First iteration**
      - Subactivities
        - Activity: **Enriched secure DWdesign**
          - ProcessRole: **Expert modeler**
          - ProcessRole: **Project manager**
          - ActivityParameters {kind:input}
            - WorkProduct: **Secure MDmodel**
            - WorkProduct: **Operational data sources**
          - ActivityParameters {kind:output}
            - WorkProduct: **Enriched secure MDmodel**
          - Steps
            - Step: **Revise technical details and needs from the final users**
            - Step: **Examine the operational sources availables**
            - Step: **Compare the secure PIM with operational sources**
            - Step: **Add new classes to MDmodel according to needed**
            - Step: **Revise the security policy and valore new constraints**
            - Step: **Analyze the existence of some contradiction between the security imposed at this level**

**Fig. 12.** Enriched Secure Data Warehouse design activity (2.2).

#### 4.4.1. Secure Data Warehouse design (A2.1)

Fig. 11 shows activity 2.1 which is called the Secure Data Warehouse design. The goal of this activity is to map the secure CIM obtained during the elicitation phase onto the secure MD model, i.e., the output is the secure MD model (secure PIM). As input the activity has the *WorkProducts GSAModel* (*secureCIM*) and the Secure MD metamodel (see Appendix A). The activity takes place in several steps.

In order to map secure CIM onto secure PIM we need to apply a set of QVT relations, which is part of our future work. We shall limit our efforts to defining a manual transformation between secure CIM (*GSAModel*) and secure PIM based on the guidelines presented below.

In the sequel we suggest several guidelines for transforming the secure CIM into the secure PIM:

*Guideline G1:* Related to actors.
*Guideline G2:* Related to *BusinessProcesses*.
*Guideline G3:* Related to *Measures*.
*Guideline G4:* Related to *Contexts*.

*Guideline G1:* Actors in the *GSAModel* (secure CIM) are mapped onto the *userProfile* class of the MD model. By default the *userProfile* class will contains three attributes: *securityLevel* (SL), *securityRole* (SR) and *securityCompartment* (SC). According to the ACA model, these attributes allow us to represent the security information for each of the system's users.

*Guideline G2:* Create an *SFact* class for each *BusinessProcess* in the *GSAModel*. The name of the *SFact* in the MD model will be the name of the *BusinessProcess* in the *GSAModel*. Several guidelines are given to obtain the security information associated with the *SFact* in the MD model.

*Guideline G2.1:* *SLevel*, *SRole* and *SCompartment* decomposition associated with the *BusinessProcess* resource through an *SSoftgoal* dependency in *GSAModel* are mapped as SL, SR and SC classes associated with the *SFact* that represents the corresponding *BusinessProcess*.

*Guideline G2.2:* Each *SConstraintRule* task that makes a positive contribution to an *SSoftgoal*, which constitutes an *SSoftgoal* dependency for the *BusinessProcess* in the *GSAModel* is mapped as a *SecurityRule* class associated with the *SFact* in the MD model. *SConstraintAudit* and *SConstraintAuthorization* tasks in the *GSAModel* are dealt with in an analogous manner.

*Guideline G3:* Each resource labeled with the stereotype ≪Measure≫ associated through the Strategic, Decision and Information goal with the *BusinessProcess* detected in guideline G2 is mapped as *SFactAttribute* for the *SFact* that corresponds with the *BusinessProcess*.

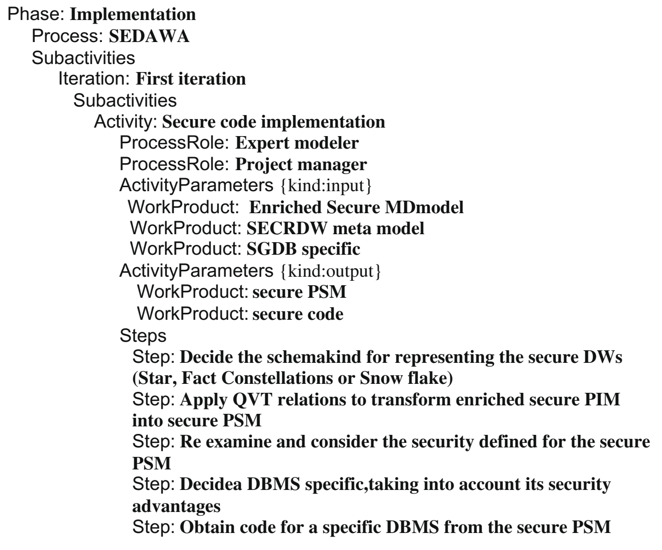
*Guideline G3.1: SLevel*, *SRole* and *SCompartment* decomposition associated with the *Measure* resource through a *SSoftgoal* dependency in *GSAModel* are mapped as SL, SR and SC classes associated with the *SFactAttribute* which represents the corresponding *Measure*.

*Guideline G3.2:* Each *SConstraintRule* task that makes a positive contribution to an *SSoftgoal*, which constitutes an *SSoftgoal* dependency for the *Measure* in the *GSAModel* is mapped as a *SecurityRule* class associated with the *SFact* that contains the *SFactAttribute* corresponding to the *Measure*. *SConstraintAudit* and *SConstraintAuthorization* tasks associated with *Measure* in the *GSAModel* are dealt with in an analogous manner.

#### 4.4.2. Enriched Secure Data Warehouse design (A2.2)

Activity 2.1 has been used to obtain a secure PIM, whose elements are *BusinessProcess*, *Context* and *Measure*. Logically, we cannot identify the whole MD model, because the level of granularity is very low (observe in Fig. 18 that certain classes do not have any attributes). Hence, we need to enrich this secure PIM with operational data sources that will populate the DW repository. Fig. 12 shows activity 2.2, which guarantees as output an enriched secure MD model, i.e., an enriched secure PIM. This activity receives as input two *WorkProducts*: the secure MD model and operational data sources. The goal of this activity is to revise other detailed technical aspects and constructors which do not belong to the organizational model, taking into account the operational data sources available.

During this activity the expert modeler and the project manager *ProcessRoles* need to increase the descriptive level of the secure MD model *WorkProduct*. The needs of the final users of the DWs are revised again in order to match technical details not contained in the secure MD model, i.e., to match complex end user's queries with the secure MD model. User needs are not only needs which are taken into account in order to define a secure DWs. The secure DWs repository will be populated with operational sources (data sources) such as Online Transaction Processing (OLTP) systems, external data sources (syndicated data, census data), etc. Hence, we need to examine the available operational sources and compare them with the secure MD model (secure PIM) *WorkProduct*. As result, new classes, new associations between them and new attributes can be added to the secure MD model. Therefore, we need to value the security policy according to the new valuable assets, and to establish new security information (SL, SR and SC) and additional constraints to the secure MD model. Once executed the steps included in activity 2.2 have been executed, we have as output the enriched secure MD model as output *WorkProduct*.



```
Phase: Implementation
   Process: SEDAWA
   Subactivities
       Iteration: First iteration
          Subactivities
             Activity: Secure code implementation
                 ProcessRole: Expert modeler
                 ProcessRole: Project manager
                 ActivityParameters {kind:input}
                   WorkProduct:  Enriched Secure MDmodel
                   WorkProduct: SECRDW meta model
                   WorkProduct: SGDB specific
                 ActivityParameters {kind:output}
                    WorkProduct: secure PSM
                    WorkProduct: secure code
                 Steps
                   Step: Decide the schemakind for representing the secure DWs
                   (Star, Fact Constellations or Snow flake)
                   Step: Apply QVT relations to transform enriched secure PIM
                   into secure PSM
                   Step: Re examine and consider the security defined for the secure
                   PSM
                   Step: Decidea DBMS specific,taking into account its security
                   advantages
                   Step: Obtain code for a specific DBMS from the secure PSM
```

**Fig. 13.** Secure code implementation activity (3.1).

### 4.5. Implementation phase

This phase is devoted to obtaining code for specific DBMS through activity 3.1, which is based on previous work related to MDA. The implementation phase covers the logical and physical stages of the DW design. As was explained in the previous subsection, the secure PIM is modeled by using the metamodel proposed by Villarroel et al. [51]. Depending upon the MDA, the secure PIM must be transformed into a secure PSM. In order to define the secure PSM for the logical level we use the metamodel presented in [48], in which we extended the relational package from the Common Warehouse Metamodel (CWM). The extended metamodel is called the SECRDW metamodel (see Appendix B) which permits the representation at the logical level of all the security and audit measures captured during the conceptual modeling stage of the DWs design.

The SECRDW metamodel defines a container *SSchema* which is inherited from Schema. *SSchema* is a collection of *STables* and *securityProperties* and is aimed at security at the model level. A *ColumnSet* represents any form of relational data. An *STable* and *UserProfile* are inherited from *Table*, which contains *Columns*. 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 (see metamodel shown in Appendix B). 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 *SecuritLevels*, *securityCompartments* and *securityRoles* classes. Furthermore, other classes are also used to represent security constraints, authorization rules and audit rules in the metamodel: the *AuditConstraint* class, the *ARConstraint* class and the *AURConstraint* class, which inherit from the *SecurityConstraint* metaclass. The aforementioned classes, along with the associations between them, can be observed in Appendix B.

Fig. 13 defines activity 3.1, which is called secure code implementation. The activity comprises several steps, which guarantee the transformation of the MDA between the enriched secure PIM and the secure PSM, and between the secure PSM and the secure code. Hence, the activity has as output the secure PSM and the secure code has as output *WorkProducts*. The Secure MD metamodel, the secure MD model, the SECRDW metamodel and, the DBMS specific *WorkProducts* are the inputs of this activity.

The secure PSM corresponds with the logical level, which is designed according to the specific properties of the DBMS such as Relational Online Analytical Processing (ROLAP), Multidimensional Online Analytical Processing (MOLAP) or Hybrid Online Analytical Processing (HOLAP). Nevertheless, Kimball and Ross [17] assures us that the most common representation is through the relational platform (i.e., ROLAP systems). Unfortunately, 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 (see future work in Section 7). In relational systems the main schema types with which to represent DWs are star, snowflake and fact constellation schemas. Therefore the first step is to decide the schema type with which to represent the secure DWs. In our context we have decide to use a star schema. The QVT relations applied to the enriched secure PIM will allow us to obtain the secure PSM automatically. At this point the revision of all the security measures obtained by using the QVT relations is appropriate, since new needs may appear. We shall now examine the security advantages for the specific DBMS that will store the secure DWs repository, i.e., Oracle, DB2, etc. Finally the secure code for a specific DBMS is automatically obtained by applying the proposal *Model2Text* [35], which is part of our future work (see Section 7).

### 4.6. Test–delivery phase

This phase is comprised solely of the activity verification and validation performed by the *ProcessRoles* expert modeler and project manager. The activity does not produce new *WorkProducts* as output. The work carried out during the test–delivery phase has a direct relationship with the post-development review discipline. This discipline allows us to look back at the development of DWs, revise the documentation created, and attempt to identify both opportunities for improvement and major successes that should be taken into account [22].

Fig. 14 defines activity 4.1 through the various steps carried out by the *ProcessRoles* expert modeler and project manager. Numerous testing techniques in the specialized literature have been proposed but it is accepted that they can be carried out the use of verification and validation methods [37]. The objective of verification testing is primarily to ensure that the secure DWs has been correctly built. The defects found must be corrected. The objective of validation testing is to determine whether the secure DWs has been built correctly. In other words, does the secure DW perform as was expected? During validation testing, test data and so on will be created [37].

Several steps are undertaken in order to carry out activity 4.1. Internal validation and external verification as expressed in [28] are the main issue when verifying that all requirements have been correctly implemented. Internal verification must identify potential conflict among security requirements and the remaining requirements, and detect incomplete, inconsistent, incomprehensible, or ambiguous requirements specification. Several techniques can be used when performing the verification process such as peer reviews, checklists or Fagan's methods [27]. Moreover, conflicts can be detected and solved by using the proposal of [19] through the graph-based approach for the specification of Access Control policies. Vulnerabilities are discovered by analyzing threats to and attacks on both the requirements and the DW repository (i.e., at both, business and application-levels). One of the best known techniques through which to model threats/attacks are attack trees, which contain threats, and their possible attacks [45]. This issue will be studied (see section dealing with Future work). Once the vulnerabilities have been found and emended we verify the quality acceptance level by testing the DWs with users. If the quality is assured and the users and maintainers have received training then we decide to release and deliver the final secure DWs.

## 5. A pharmaceutical consortium: an example of application

In this section, we apply our secure engineering process (SEDAWA) to the context of a pharmaceutical consortium. This consortium manages several pharmacies which offer various types of services to the community, and wishes to control all aspects related to the sales of medicines through medical prescriptions. A patient's prescription contains the patient's data, information related to its illness and the medicine that should acquire. In order to satisfy the demands that presupposes the previous problem is required a secure DWs.

In the following subsections we apply the four phases that comprises the process. Our process is iterative and incremental thereby is necessary to plan the iterations that will be executed. In this section we consider only one iteration in order to make more understandable the example.

### 5.1. Elicitation phase: example

This phase comprises three activities (A1.1, A1.2 and A1.3). The *WorkProduct* output is *GSAModel* (i.e., secure CIM).

#### 5.1.1. Activity A1.1

Fig. 15 shows the *GOModel* obtained as *WorkProduct* output from activity A1.1. The business process is related to one main actor, the marketing manager via the strategic goal "*increase prescription sales*". Two different decision goals are derived from this strategic goal: "*decrease prescription price*" and "*give incentive to pharmacist*". The following information goals have been obtained

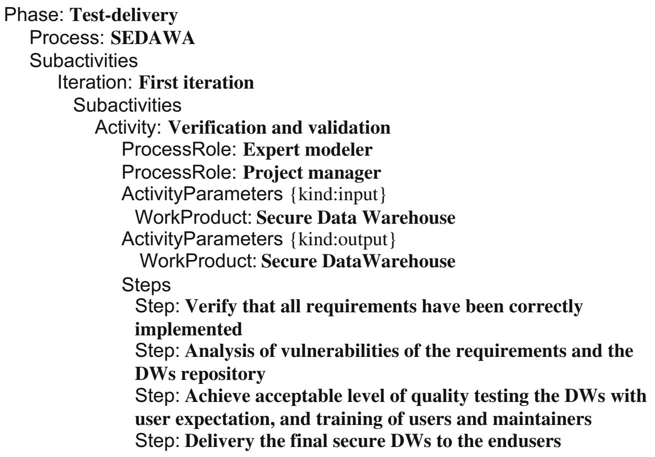

Phase: **Test-delivery**
Process: **SEDAWA**
Subactivities
Iteration: **First iteration**
Subactivities
Activity: **Verification and validation**
ProcessRole: **Expert modeler**
ProcessRole: **Project manager**
ActivityParameters {kind:input}
WorkProduct: **Secure Data Warehouse**
ActivityParameters {kind:output}
WorkProduct: **Secure DataWarehouse**
Steps
Step: **Verify that all requirements have been correctly implemented**
Step: **Analysis of vulnerabilities of the requirements and the DWs repository**
Step: **Achieve acceptable level of quality testing the DWs with user expectation, and training of users and maintainers**
Step: **Delivery the final secure DWs to the endusers**

**Fig. 14.** Verification and validation activity (4.1).

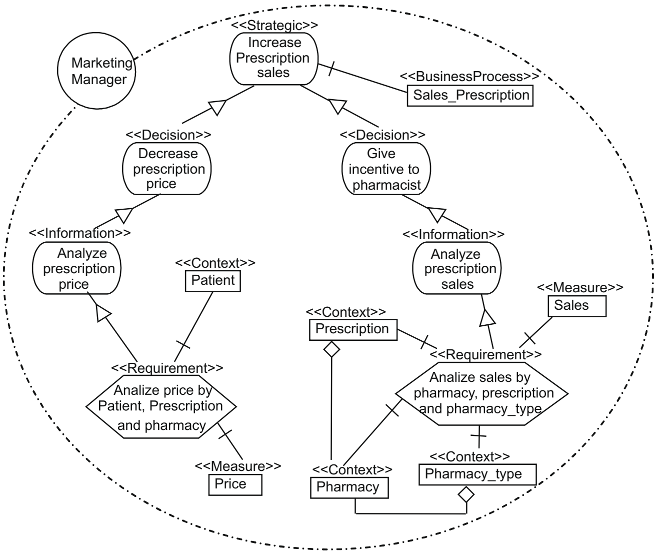


**Fig. 15.** Goal organization model obtained from activity 1.1.

from each of these decision goals: "*decrease prescription price*" and "*analyze prescription price*". The derived information requirements are as follows: "*analyze price by patient*, *prescription and pharmacy*, and *analyze sales by pharmacy*, *prescription and pharmacy_type*". In Fig. 15, each of these elements are defined as goals (strategic, decision and information goals) or tasks (information requirements). Furthermore, several resources are associated with the information requirements where necessary, such as measures and context of analysis. The measures are "*Sales* and *Price*". The elements that represent context of analysis are "*Prescription*, *Pharmacy* and *Pharmacy_Type*", but these are related to each other, since they represent means of aggregating the "*Pharmacy*" data. *Patient* is also a context of analysis, but has no relationship to the other *Contexts*. Fig. 15 represents the functional requirements for DWs. According to MDA it represents a CIM without security.

### 5.1.2. Activity A1.2

The security requirements for DWs are obtained by applying activity A1.2. Hence, we focus on the sales prescription process as a security policy, which is performed by the *SecurityManager* actor via the "*guarantee the security for the sales prescription process*" *SSoftgoal*. By using a refinement process, three new softgoals are obtained: "*guard the security of use of certain medication and consumer's rights*, *maintain privacy of sales*, *price and patient's data* and *impose a clearance level on prescription process*" (see Fig. 16). Various responsibilities are discovered in this process. Hierarchical relations are therefore defined, of which the most general is *PharmacyEmployee*, which is then specialized into the *Pharmacist (Pharma)* and *Administrative (Admin)* roles. Horizontal groups (compartments) within the organization are detected: *pharmacovigilanceCenter* (*pharmaC*), *which is responsible for the security of the use of certain medications* and *commercialManagerCenter* (*commercialC*), which is responsible for commercialization and supply. Restriction levels are established by means of *TopSecret* (TS) and *Secret* (S). Note in Fig. 16 how the security resources are associated with their corresponding *SSoftgoals*. Fig. 16 represents the *WorkProduct* output (*SOModel*) from activity A1.2.

### 5.1.3. Activity A1.3

By means of activity A1.3 the *WorkProducts* output from activities A1.1 and A1.2 are merged in the *WorkProduct GSAModel*. Fig. 17 shows how *GOModel* and *SOModel* are merged by means of the *Dependency* association (–D–). The requirements shown in Fig. 15 are associated with the *SSoftgoals* contained in Fig. 16. If we are to fulfill previous *SSoftgoals* we need to associate resources contained in *GOModel* (i.e., *Sales_Prescription*, *Patient*, *Price*, *Prescription*, *Pharmacy_Type* and *Sales*) with the *SSoftgoals* contained in *SOModel*. For example, "*impose maximum level of restriction on the sales prescription*" (marked in Fig. 17 with the number 1) and "*Guarantee the secure use of medication norms*" (marked in Fig. 17 with the number 5). The remaining *SSofgoals* which establish associations with resources from *GOModel* are dealt with analogously (see the *SSoftgoals* marked with the numbers 2, 3 and 4 in Fig. 17). The aforementioned *SSoftgoals* are, therefore, achieved through a *Dependency* association between the *SecurityManager* and the *MarketingManager*. *Sales_Prescription* is associated with the "*Impose maximum level of restriction on the sales prescription*" *SSoftgoals* whose *SLevel* is *TopSecret*.

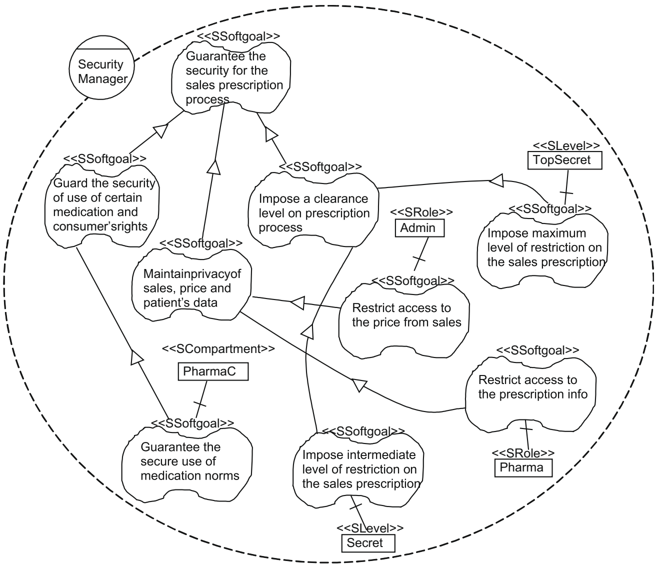

Security Manager
<<SSoftgoal>>
Guarantee the security for the sales prescription process
<<SSoftgoal>>
Guard the security of use of certain medication and consumer'srights
<<SSoftgoal>>
Impose a clearance level on prescription process
<<SLevel>>
TopSecret
<<SRole>>
Admin
<<SSoftgoal>>
Impose maximum level of restriction on the sales prescription
<<SSoftgoal>>
Maintainprivacyof sales, price and patient's data
<<SSoftgoal>>
Restrict access to the price from sales
<<SCompartment>>
PharmaC
<<SSoftgoal>>
Restrict access to the prescription info
<<SSoftgoal>>
Guarantee the secure use of medication norms
<<SSoftgoal>>
Impose intermediate level of restriction on the sales prescription
<<SRole>>
Pharma
<<SLevel>>
Secret

**Fig. 16.** Softgoal organization model obtained from activity 1.2.

Other *SSoftgoals* are, moreover, associated with resources (*Patient*, *Price*, *Prescription*, *Pharmacy_Type* and *Sales*). *Sales_Prescription* and *Prescription* are very valuable assets, and therefore, need additional restrictions. Fig. 17 shows how the *SOModel* has been modified with the *SRule* and *Audit* constraints, which are labeled as *SConstraintRule*, *SConstraintAudit*, respectively. *SRule* contributes to the fulfillment of the *SSoftgoal* "*impose maximum level of restriction on the sales prescription*", so according to the dependency association defined, it is related to both the *BusinessProcess Sales_Prescription* and *Context Prescription*, respectively. The same reasoning assures that the *Context Prescription* will be related to the *Audit* constraint. Moreover, other *SSoftgoals* are associated with resources (*Patient*, *Price*, *Prescription*, *Pharmacy_Type* and *Sales*). These are dealt with in an analogous manner.

According to MDA Fig. 17 can be seen as a secure CIM because its represents both functional and non-functional requirements in the same model. In accordance with the classification for users of the ACA model introduced in Section 3, each of the system's user will have *securityLevel*, *securityRole* and *securityCompartment*. Hence, we can conclude that a user has access to *Sales_Prescription* if his/her access class dominates the access class of *Sales_Prescription*, i.e., his/her security level is *TopSecret* (in this restricted case).

We have developed a prototype case tool in order to assessment the secure engineering process by using Eclipse development platform framework.

### 5.2. Modeling phase: example

The modeling phase guarantees both, map secure CIM (*WorkProduct GSAModel* obtained from A1.3) onto secure PIM and enrich it with the operational data sources that will populate the DW repository.

### 5.2.1. Activity A2.1

In order to define a map between a secure CIM (represented in Fig. 17) and a secure MD model (secure PIM), we apply a set of guidelines which correspond with the steps from activity 2.1. As we can see in Fig. 17 we have only one actor, denoted as *MarketingManager*, which will be an instance of the *UserProfile* class. The values of SL, SR and SC for each actor are (for the moment) unknown due to the granularity at this level.

We also have only one *BusinessProcess* (see the *GSAModel* depicted in Fig. 17). According to G2 the *Sales_Prescription BusinessProcess* should be mapped onto the *Sales_Prescription SFact* (see Fig. 18). Note in Fig. 17 how the *SecurityManager* depends on the *MarketingManager* to achieve the *SSoftgoals* marked with the numbers 1 and 2. According to G2.1 the *Sales_Prescription SFact* is associated with the SL *Secret* (S) and *TopSecret* (TS), which are represented in its heading (see Fig. 18). According to G2.2 the *Sales_Prescription SFact* is associated with the *SecurityRule* 1, which is modeled in Fig. 18 by using a UML note.

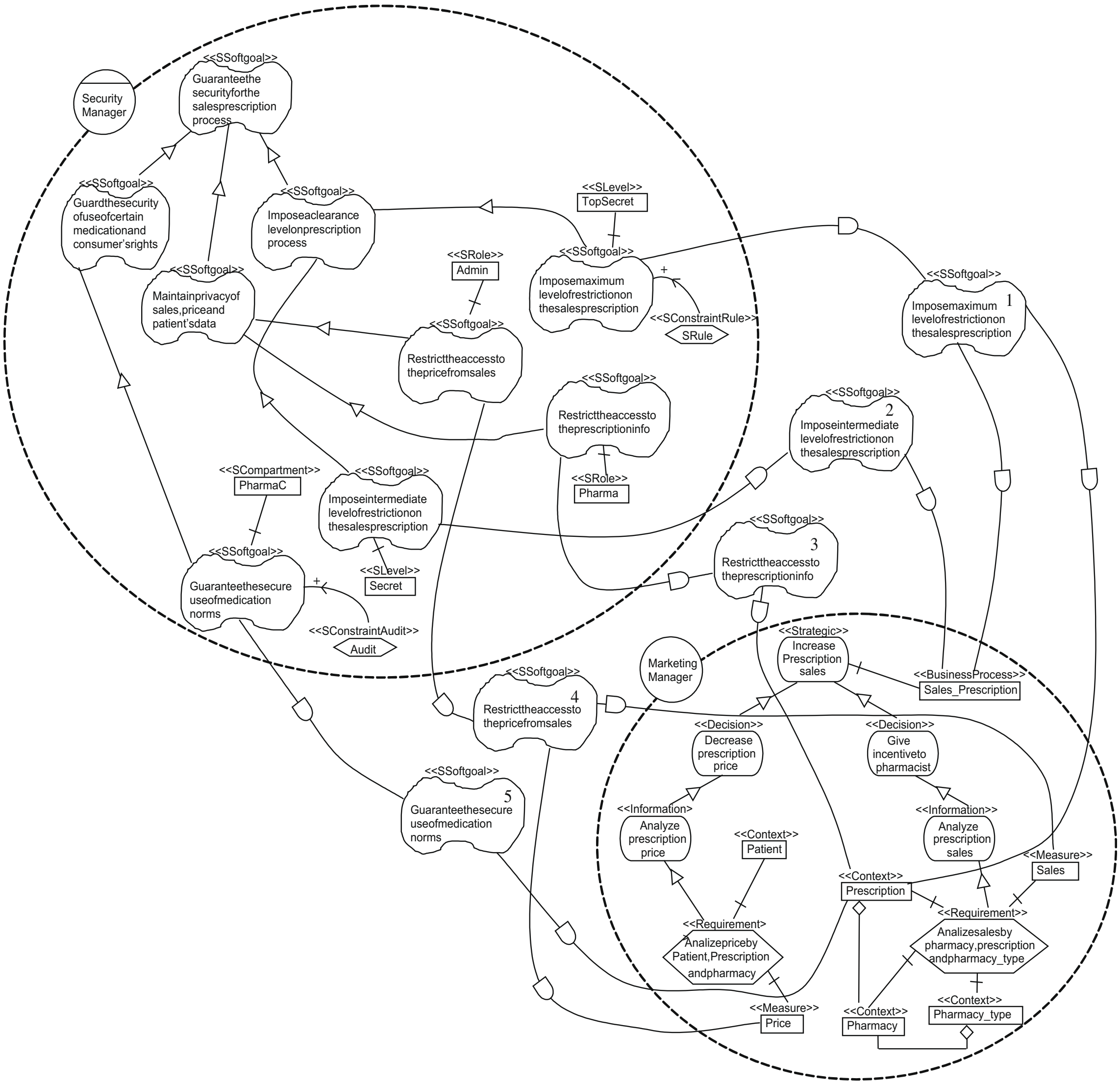

Security Manager
<<SSoftgoal>>
Guaranteethe securityforthe salesprescription process
<<SSoftgoal>>
Guardthesecurity ofuseofcertain medicationand consumer'srights
<<SSoftgoal>>
Imposeaclearance levelonprescription process
<<SLevel>>
TopSecret
<<SRole>>
Admin
<<SSoftgoal>>
Imposemaximum levelofrestrictionon thesalesprescription
+
<<SConstraintRule>>
SRule
<<SSoftgoal>>
Maintainprivacyof sales,priceand patient'sdata
<<SSoftgoal>>
Restricttheaccessto thepricefromsales
<<SSoftgoal>>
Restricttheaccessto theprescriptioninfo
<<SRole>>
Pharma
<<SCompartment>>
PharmaC
<<SSoftgoal>>
Imposeintermediate levelofrestrictionon thesalesprescription
<<SLevel>>
Secret
<<SSoftgoal>>
Guaranteethesecure useofmedication norms
+
<<SConstraintAudit>>
Audit
<<SSoftgoal>>
Imposemaximum levelofrestrictionon thesalesprescription
1
<<SSoftgoal>>
2
Imposeintermediate levelofrestrictionon thesalesprescription
<<SSoftgoal>>
3
Restricttheaccessto theprescriptioninfo
Marketing Manager
<<Strategic>>
Increase Prescription sales
<<BusinessProcess>>
Sales_Prescription
<<SSoftgoal>>
4
Restricttheaccessto thepricefromsales
<<Decision>>
Decrease prescription price
<<Decision>>
Give incentiveto pharmacist
<<SSoftgoal>>
5
Guaranteethesecure useofmedication norms
<<Information>>
Analyze prescription price
<<Context>>
Patient
<<Information>>
Analyze prescription sales
<<Measure>>
Sales
<<Context>>
Prescription
<<Requirement>>
Analizepriceby Patient,Prescription andpharmacy
<<Requirement>>
Analizesalesby pharmacy,prescription andpharmacy_type
<<Measure>>
Price
<<Context>>
Pharmacy
<<Context>>
Pharmacy_type

**Fig. 17.** Goal/Softgoal analysis model obtained from activity 1.3.

With regard to *Measures*, we have the *Price* and *Sales* (see *GSA-Model* in Fig. 17) which are mapped as *SFactAttributes* in the MD model, as Fig. 18 shows. In Fig. 17 we can see that *SecurityManager* depends on *MarketingManager* to achieve the *SSoftgoals* marked with the number 4. Hence, according to G3.1 these attributes are associated with the SR *Admin* (see Fig. 18). These *Measures* do not have associated security constraints (*SConstraintRule*, *SConstraintAudit* and *SConstraintAuthorization*).

In Fig. 17 we have two *Contexts* of analysis: (i) the *Patient Context* is transformed into the *Patient SDimension* in the MD model (see Fig. 18) and (ii) the *Pharmacy*, *Pharmacy_Type* and *Prescription Contexts* represent the *SDimension Pharmacy*. According to G4.1 the *Pharmacy Context* represents the *SBase* root of the *SDimension Pharmacy* (see Fig. 17). If we apply G4.2, the *Prescription* and *Pharmacy_Type Contexts* are mapped as *SBases* in the MD model. The UML aggregations between them and the *Pharmacy Context* are mapped as a *Rolls-upTo* association between the corresponding *SBase* classes in the MD model (see Fig. 18).

In Fig. 17 we can see that *SecurityManager* depends on *Marketing-Manager* to achieve the *SSoftgoals* marked with the numbers 1, 3 and 5. Hence, according to G4.3, the *Prescription Context* is associated with the SL *TopSecret* (TS), the SR *Pharma* and the SC *PharmaC* (see Fig. 18). According to G4.4 the *Prescription Context* is associated with the *AuditRule* 2 and the *SecurityRule* 3, which are represented in the MD model shown in Fig. 18 by using UML notes. These constraints

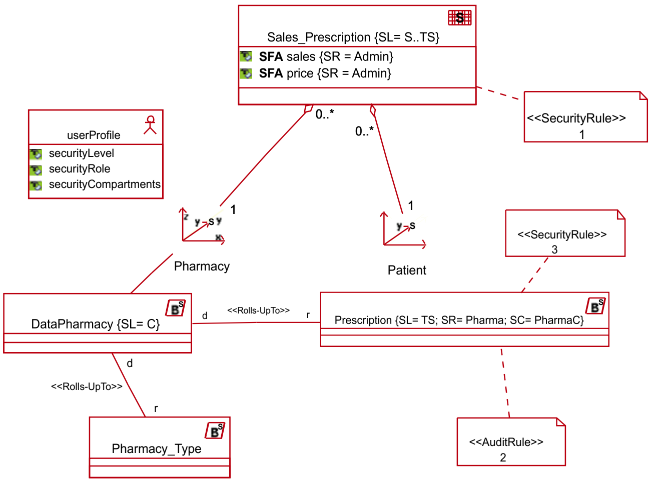


**Fig. 18.** Secure PIM obtained from activity 2.1.

are obtained by taking into account the *SSoftgoals* dependency marked in the *GSAModel* (see Fig. 17) with the numbers 1 and 5, between the *SecurityManager* and *MarketingManager*.

### *5.2.2. Activity A2.2*

As was previously stated, the activity 2.2 contrasts the secure PIM (obtained from activity A2.1) with the operational data sources available. The *WorkProduct* output from activity A2.2 is the Enriched secure MD model (enriched secure PIM). This activity is carried out manually, with which the designer can retouch the incipient secure MD model (secure CIM) obtained from activity A2.1.

If we continue with the example that we have been developing, it now follows that we apply activity A2.2. Fig. 19 shows an instance of our enriched secure PIM (see metamodel shown in paragraph A), which makes part of the DW that is required for the previous problem more complete. The *SFact Sales_Prescription* (stereotype *SFact*) contains all the sales information in one or more pharmacies, and can be accessed by users who have *Secret* or *topSecret security levels*, play an *Administrative* or *Pharmacist* role and belong to *pharmacovigilanceCenter*, *healthOversightCenter* (the committee which guards the health of the company's clients) and *commercialManagerCenter* compartments. The sales attribute can only be accessed by users who perform the administrative role (tagged values SR of sales attribute) and belong to the *commercialManagerCenter* compartment, and access to this attribute will therefore be forbidden to other users who are (*pharmacist* and *maintenance* employees or belong to other different *commercialManagerCenter* compartments). The *income* attributes can only be accessed by users who perform the *administrative* role (tagged value SR of *income* attribute). Other static user classifications for the conceptual model classes defined in Fig. 19 are:

The *SFact Sales_Prescription* which contains three *SDimensions* (*Pharmacy*, *Patient* and *Medication*), which contains *SBase* hierarchies. Access to these *SBase* hierarchies is established in the same way as was done with the *SFact*. The *UserProfile* has been completed in order to store information about all users who will have access to this secure MD model.

Several security constraints have been specified by using the previously defined constraints, stereotypes and tagged values. The following paragraphs correspond to notes 1, 2 and 3 in Fig. 19:

1. For each instance of the *SFact* class *Sales_Prescription*, if the type of payment is through insurance then the security compartment will be *commercialManagerCenter* (*commercialC*, tagged value SC). This constraint is only applied if the user makes a query whose information comes from *DataPharmacy*.
2. We would like to record, for future audit, the *subject*, *object* and *time* of every frustrated access attempt upon *Prescription*.
3. For each instance of the *SBase* class *Prescription*, if the prescription is of the type "*socialSecurity*", then the security level will be *Secret* (*Secret*, tagged value SL).

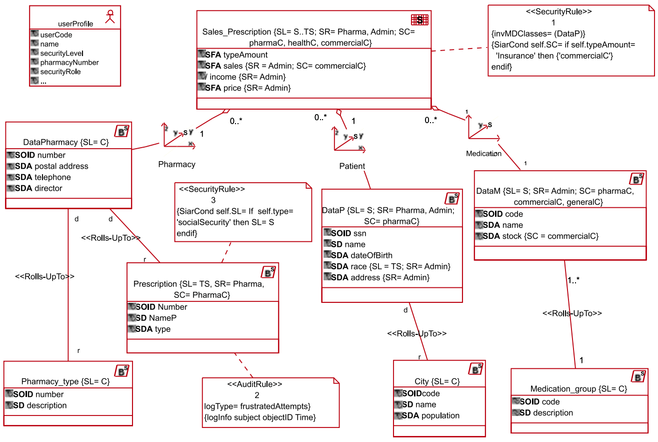


**Fig. 19.** Enriched secure PIM obtained from activity 2.2.

### *5.3. Implementation phase: example*

By means of this phase is obtained the secure PSM (relational) and the corresponding code for specific DBMS. The Implementation phase comprises only the activity A3.1.

### *5.3.1. Activity A3.1*

By using the *PIM* in Fig. 19 as a starting point, we apply a set of *QVT* relations [49] (contained as step within the activity A3.1) through which to achieve an instance of the *secure PSM*. The transformation ensures that *SFact* and *SDimensions* are transformed into *STables* with their associated security information. The *UserProfile* class is transformed into a classical *Table* from *CWM*. Fig. 20 represents a star schema at the logical level, which corresponds with an instance of the relational metamodel from the CWM extended in [48].

The *SFact Sales_Prescription* is represented in Fig. 20 by means of the *STable Sales_ Prescription*. All of its columns are represented in this table along with all the associated security information, which restricts access both to the table itself and to its columns. All the hierarchy that conforms to an *SDimension* must be represented by means of a single *STable*. Observe in Fig. 20 that the *Pharmacy STable* contains as *SColumn* the attributes from the *SBases DataPharmacy*, *Pharmacy_Type* and *Prescription* classes from Fig. 19. This occurs in an analogous manner with the *Patient* and *Medication SBases* classes. In order to build a star scheme the *Sales_Prescription* table must contain columns such as *Foreign Key*(*FK*) which represent *Primary Key* (PK) in the tables that correspond with *SDimensions* at the PIM level.

The security information (SL, SR and SC) represented in the classes from Fig. 19 is modeled at the logical level in the title of the table itself (See Fig. 20).

The *SecurityRule1*, *AuditRule2* and *SecurityRule3* security constraints that appear in Fig. 19 are transformed into instances of the *SecurityConstraint* from the extended relational metamodel. These instances are modeled in Fig. 20 by means of UML notes with

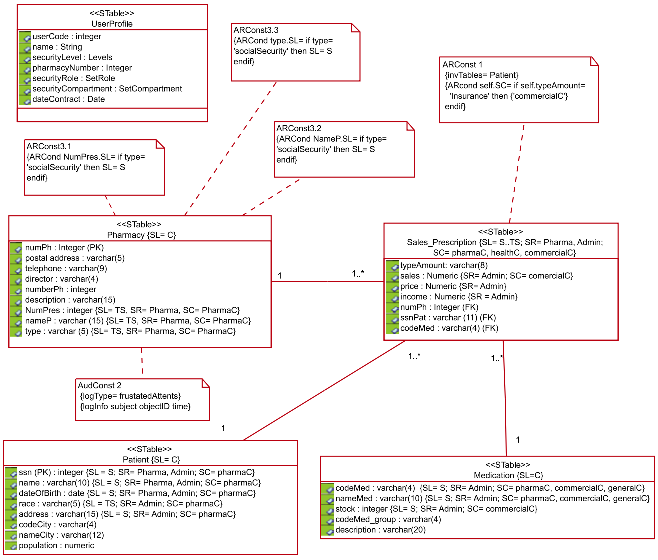
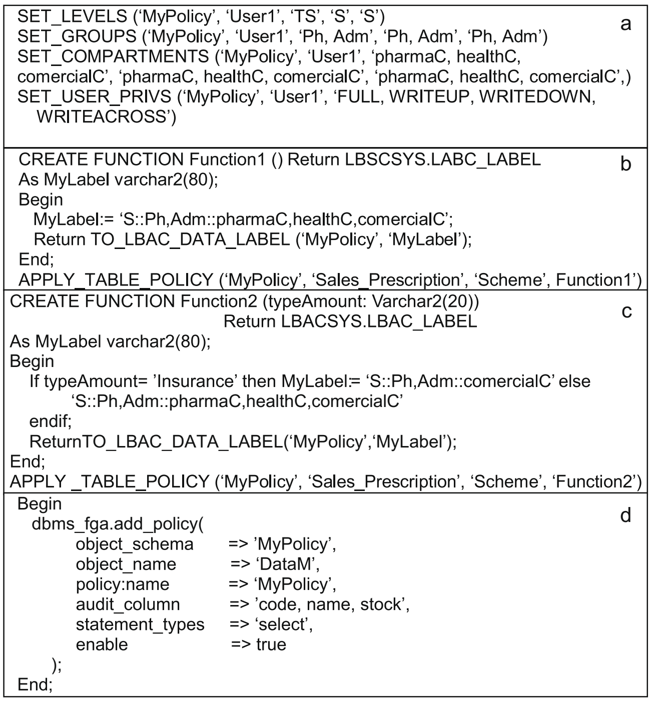

<<STable>>
UserProfile
userCode : integer
name : String
securityLevel : Levels
pharmacyNumber : Integer
securityRole : SetRole
securityCompartment : SetCompartment
dateContract : Date
ARConst3.3
{ARCond type.SL= if type= 'socialSecurity' then SL= S
endif}
ARConst 1
{invTables= Patient}
{ARcond self.SC= if self.typeAmount= 'Insurance' then {'commercialC'}
endif}
ARConst3.1
{ARCond NumPres.SL= if type= 'socialSecurity' then SL= S
endif}
ARConst3.2
{ARCond NameP.SL= if type= 'socialSecurity' then SL= S
endif}
<<STable>>
Pharmacy {SL= C}
numPh : Integer (PK)
postal address : varchar(5)
telephone : varchar(9)
director : varchar(4)
numberPh : integer
description : varchar(15)
NumPres : integer {SL= TS, SR= Pharma, SC= PharmaC}
nameP : varchar (15) {SL= TS, SR= Pharma, SC= PharmaC}
type : varchar (5) {SL= TS, SR= Pharma, SC= PharmaC}
1
1..*
<<STable>>
Sales_Prescription {SL= S..TS; SR= Pharma, Admin; SC= pharmaC, healthC, commercialC}
typeAmount: varchar(8)
sales : Numeric {SR= Admin; SC= comercialC}
price : Numeric {SR= Admin}
income : Numeric {SR = Admin}
numPh : Integer (FK)
ssnPat : varchar (11) (FK)
codeMed : varchar(4) (FK)
1..*
1..*
AudConst 2
{logType= frustatedAttents}
{logInfo subject objectID time}
1
1
<<STable>>
Patient {SL= C}
ssn (PK) : integer {SL = S; SR= Pharma, Admin; SC= pharmaC}
name : varchar(10) {SL = S; SR= Pharma, Admin; SC= pharmaC}
dateOfBirth : date {SL = S; SR= Pharma, Admin; SC= pharmaC}
race : varchar(5) {SL = TS; SR= Admin; SC= pharmaC}
address : varchar(15) {SL = S; SR= Admin; SC= pharmaC}
codeCity : varchar(4)
nameCity : varchar(12)
population : numeric
<<STable>>
Medication {SL=C}
codeMed : varchar(4) {SL= S; SR= Admin; SC= pharmaC, commercialC, generalC}
nameMed : varchar(10) {SL= S; SR= Admin; SC= pharmaC, commercialC, generalC}
stock : integer {SL= S; SR= Admin; SC= commercialC}
codeMed_group : varchar(4)
description : varchar(20)

**Fig. 20.** An instance of the secure PSM obtained from activity 3.1.

the names *ARConst1* and *AURConst2*, respectively. The *securityRule3* attempts to change the security for the *SBase Prescription* class, thus establishing new values for *securityLevel (SL)*. As we observed in Fig. 19, the security of the *SBase Pharmacy* class has been assigned to the *SColumns NumPres*, *nameP* and *type*. Hence, the constraint is transformed and applied to *SColumns NumPres*, *nameP* and *type*. Consequently, the *SecurityRule3* is transformed into three *ARConstraints*, which appear in Fig. 20 under the names of *ARConst3.1*, *ARConst3.2* and *ARConst3.3*, which are associated with the *SColumns NumPres*, *nameP* and *type*, respectively.

To illustrate the step related to obtain code for a specific DBMS (contained as step within the activity A3.1), we shall briefly show the possibilities that Oracle 11g DBMS offers in order to implement security and audit measures by means of Oracle Label Security (OLS11g), Virtual Private Databases (VPD) and Oracle Fine-Grained Auditing (FGA). We shall only explain the security aspects that our extension contemplates, and to do this we have first created a security policy named "*MyPolicy*" along with valid levels, compartments and hierarchical groups. See Fig. 21.

In Fig. 21a we show how *User1* satisfies the security properties for the *Sales_Prescription STable*. Fig. 21b shows how we define and establish the security information for the *Sales_Prescription* table by labeling functions from OLS, although it is not possible to consider security at the column level. The *ARConst 1* is implemented by means of the labeling function represented in Fig. 21c. The FGA allows us to define and implement the *AudConst 2* (see Fig. 21d). In *AudConst 2* we cannot implement the *logType* because FGA does not allow us to choose it.



```
SET_LEVELS ('MyPolicy', 'User1', 'TS', 'S', 'S')                                         a
SET_GROUPS ('MyPolicy', 'User1', 'Ph, Adm', 'Ph, Adm', 'Ph, Adm')
SET_COMPARTMENTS ('MyPolicy', 'User1', 'pharmaC, healthC,
comercialC', 'pharmaC, healthC, comercialC', 'pharmaC, healthC, comercialC',)
SET_USER_PRIVS ('MyPolicy', 'User1', 'FULL, WRITEUP, WRITEDOWN,
   WRITEACROSS')
```

```
CREATE FUNCTION Function1 () Return LBSCSYS.LABC_LABEL                                    b
As MyLabel varchar2(80);
Begin
  MyLabel:= 'S::Ph,Adm::pharmaC,healthC,comercialC';
  Return TO_LBAC_DATA_LABEL ('MyPolicy', 'MyLabel');
End;
APPLY_TABLE_POLICY ('MyPolicy', 'Sales_Prescription', 'Scheme', Function1')
```

```
CREATE FUNCTION Function2 (typeAmount: Varchar2(20))                                      c
                              Return LBACSYS.LBAC_LABEL
As MyLabel varchar2(80);
Begin
  If typeAmount= 'Insurance' then MyLabel:= 'S::Ph,Adm::comercialC' else
       'S::Ph,Adm::pharmaC,healthC,comercialC'
  endif;
  ReturnTO_LBAC_DATA_LABEL('MyPolicy','MyLabel');
End;
APPLY _TABLE_POLICY ('MyPolicy', 'Sales_Prescription', 'Scheme', 'Function2')
```

```
Begin                                                                                     d
  dbms_fga.add_policy(
        object_schema      => 'MyPolicy',
        object_name        => 'DataM',
        policy:name        => 'MyPolicy',
        audit_column       => 'code, name, stock',
        statement_types    => 'select',
        enable             => true
     );
End;
```

**Fig. 21.** Implementing our constraints in Oracle 11g.

*5.4. Test–delivery phase: example*

This phase comprises only one activity. As we said above, we have only developed one iteration to make more understandable the example. Hence, we assume that all steps belonging to activity A4.1 have been executed, i.e., all informational and security requirements have been correctly implemented, both the repository and the requirements do not have vulnerabilities, we achieve

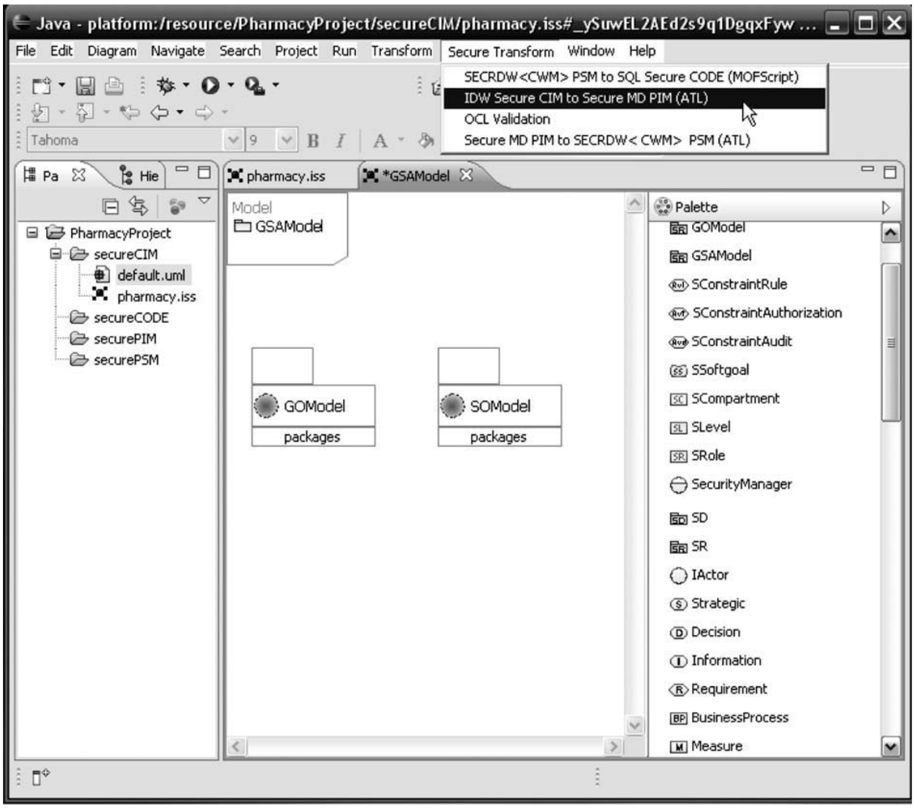


**Fig. 22.** Defining *GOModel* and *SOModel* in our IDE based on the Eclipse platform.

good acceptance level of quality and we delivery the final secure DWs to the end users.

With the aforementioned comment we finish the development of the secure DWs corresponding to the pharmaceutical consortium example. In the next section we show how the prototype CASE tool that we are developing is applied to the example.

### 5.5. Applying a prototype case tool

Currently we are developing a prototype case tool based on the Eclipse[3] development platform (see Fig. 22). We employ several of its plug-ins implementing the MDA standard: for instance, the "model development tools" (MDT) for supporting UML and UML profile, the "eclipse modeling framework" (EMF) for specifying the CWM extended [48], "ATLAS Transformation Language" (ATL)[4] project in order to specify the secure CIM-secure PIM and secure PIM-secure PSM transformations and *MOFScript* to design model-to-code *MOF2Text* [35] mappings to automatically implement the final secure DWs. We have combined the aforementioned defined plug-ins to provide an "integrated development environment" (IDE) to design secure DWs projects based on the secure engineering process proposed in Section 4.

Fig. 22 shows the tools that we have implemented as proof of concepts of our secure engineering process. On the left-hand side of the figure, the Pharmacy Secure Data Warehouse project have been initiated, which automatically creates the *secureCIM, securePIM, securePSM* and *secureCODE* folders within the Eclipse *Workspace*. If we create the pharmacy project for defining the secure CIM, then within the *secureCIM* folder appears the default.uml and pharmacy.iss files (see left hand side of the figure). The pharmacy.iss file allows us to define the *GSAModel, GOModel* and *SOModel* diagrams (see the middle part of the Fig. 22) by using the corresponding tools, which appear on the right hand side of the own figure. The default.uml file allows us to editing the created diagram by using the UML model editor from Eclipse. The figure also, shows on the right hand side the necessary tools in order to define *GOModel* and *SOModel*. In the upper part of the figure appears the classical menu bar from Eclipse, which have been adapted to support the Transform and Secure Transform menu options. The Transformation (Secure Transformation) menu option corresponds with the DWs development without security (with security). The Secure Transformation menu option includes the **IDW Secure CIM** to **Secure MD PIM (ATL)**, **Secure MD PIM** to **SECRDW** <CWM> **PSM (ATL)** and **SECRDW** <CWM> **PSM to SQL Secure CODE (MOFScript)** in order to transform the secure CIM into the secure PIM, the secure PIM into the secure PSM and the secure PSM into the secure CODE, respectively. The OCL Validation menu option implements and checks the associated constraints to the metamodels employed.

## 6. Limitations of SEDAWA

Our secure engineering process contributes to automatize the development of secure DWs projects. Nevertheless, our proposal has some limitations:

- The step related to the transformation of secure CIM to secure PIM requires a manual retouch in order to contrast the secure PIM with the operational data sources obtained. This is because the CIM model has a very high abstraction level.
- The architecture can be completed with other secure relational paradigms as secure PSM (for instance, MOLAP or HOLAP systems) and the corresponding MDA transformation.
- The prototype CASE tool which supports our process needs to be completed and validated with real projects.
- Our process is only based on direct engineering methods. The proposal could be enriched by developing methods in order to offer direct and reverse engineering methods.
- In our approach, security is based on access control to guarantee confidentiality and audit in the DWs design. However, other security aspects, such as integrity, reliability and availability could be taken into account out of the Data Warehouse design.
- Other kind of non-functional requirements such as cost-benefit and performance are not included within our process.

In the following section we describe as future work some of the aforementioned limitations of our secure engineering process (SEDAWA).

## 7. Conclusion and future work

In this paper we have proposed a secure engineering process for DWs, by eliciting and developing both functional and security aspects as non-functional requirements at the business level. This approach is outlined as follows. First a secure CIM is built by using the three activities supported by an adaptation of the $i^*$ framework. Second, the secure CIM is transformed and developed by using QVT transformations throughout the DW life cycle. Our methodology is MOF-compliant as a result of the application of SPEM, i.e., according to the four layer architecture from OMG, it belongs to the M1 layer. The greatest contribution of this work is that all the security and audit requirements elicited during the early phases are modeled, developed and defined throughout the entire DW life cycle. We believe that both the time and effort invested in the development of DWs are lessened, the transition between different models and the final implementation is guaranteed, and that it is possible to attain interoperability, portability, adaptability and reusability by employing MDA technology.

Our immediate future work consists of several tasks: defining a formal MDA transformation by using QVT between secure CIM and secure PIM, defining several secure PSMs, such as secure Multidimensional Online Analytical Processing (MOLAP) and secure Hybrid Online Analytical Processing (HOLAP), adapting the *Model2Text* approach in order to transform models into code for specific DBMS such as Oracle, SQL Server or MySQL, which may be exploited by using Pentaho or SQL Server Analysis Services, and complete the CASE tool developed in order to automatically implement secure DWs. We shall also propose new methods with which to detect vulnerabilities and contradictions between security rules defined by the ACA model at different levels of design.

## Acknowledgements

This work has been partially supported by the META-SIGN Project (TIN2004-00779) from the Spanish Ministry of Education and Science, by the QUASIMODO and MISTICO Projects from the Regional Science and Technology Ministry of Castilla-La Mancha and by the ESFINGE Project (TIN2006-15175-C05-05) of the Ministry of Science and Innovation (Spain).

## Appendix A. UML Metamodel for the secure PIM

Fig. B.1 shows the UML metamodel whose instances serve as secure PIM in activities A1.2 and A1.3 from the modeling phase (see Section 4.4). The metamodel allows us to represent the main secu-

[3] URL: <http://www.eclipse.org>.

[4] URL: <http://www.eclipse.org/m2m/atl/>.

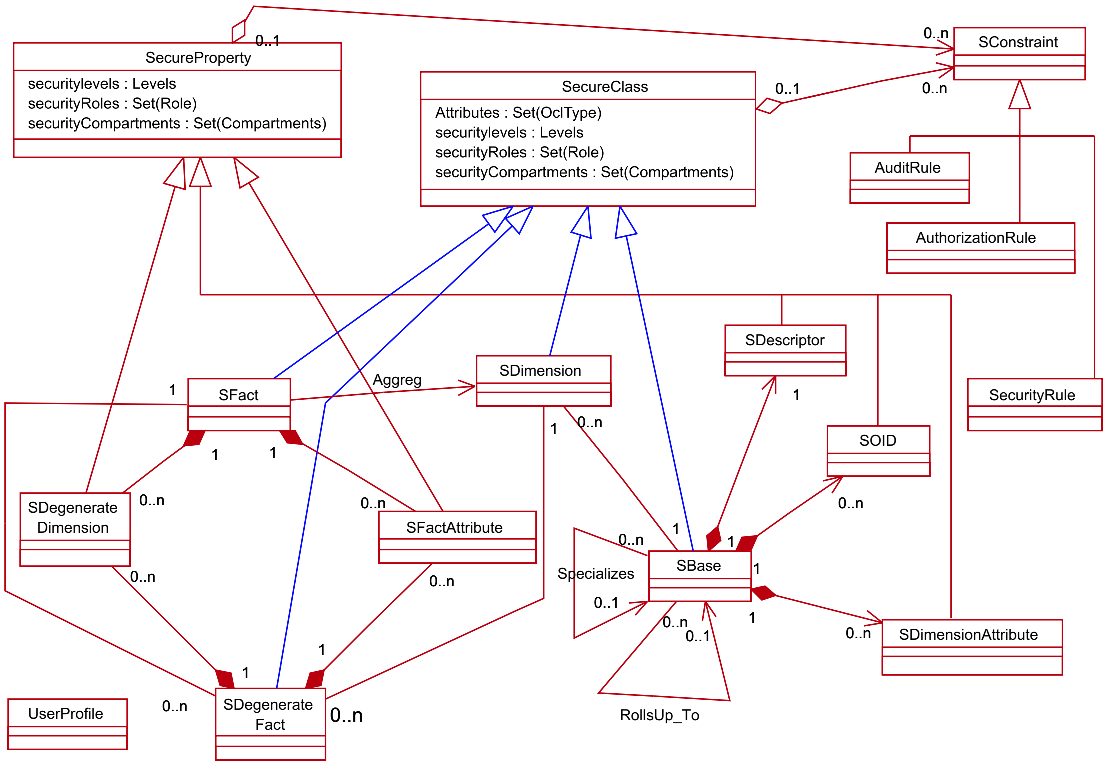


**Fig. B.1.** Metamodel used in the design of secure PSM.

rity information, constraints and security rules associated with *SFact*, *SBase*, *SFactAttribute*, etc., classes in the conceptual modeling of secure DWs.

Secure facts and secure dimensions are represented by *SFact* and *SDimension* classes, respectively. *SFact* classes are specified as composed classes by means of aggregation relationships of n *SDimension* classes. *SDimension* classes are composed of classification hierarchy levels; every classification hierarchy level is specified by a class called *SBase* class. *Roll-UpTo* association represents the relationship between two levels of a classification hierarchy. The categorization of *SDimensions* is represented by means of specialization associations between *SBases* classes. *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 a *SFact* class and a *SDimension* class, which can contains *SFactAttributes* and *SDegenerateDimensions*. 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 *UserProfile* class may contains attributes in order to specify security information associated to an system's user. Other tagged values (*LogType*, *LogInfo*, etc.) allow us to define security rules (*AuditRule*, *AuthorizationRule* and *SecurityRule*) which are associated with *SecureClass* and *SecurityProperty* to impose additional restriction in the system. See [51,6,7] for more details. Secure facts and secure dimensions are represented by *SFact* and *SDimension* classes, respectively. *SFact* classes are specified as composed classes by means of aggregation relationships of n *SDimension* classes. *SDimension* classes are composed of classification hierarchy levels; every classification hierarchy level is specified by a class called *SBase* class. *Roll-UpTo* association represents the relationship between two levels of a classification hierarchy. The categorization of *SDimensions* is represented by means of specialization associations between *SBases* classes. *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 a *SFact* class and a *SDimension* class, which can contains *SFactAttributes* and *SDegenerateDimensions*. 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 *UserProfile* class may contains attributes in order to specify security information associated to an system's user. Other tagged values (*LogType*, *LogInfo*, etc.) allow us to define security rules (*AuditRule*, *AuthorizationRule* and *SecurityRule*) which are associated with *SecureClass* and *SecurityProperty* to impose additional restriction in the system. See [51,6,7] for more details.

## Appendix B. Relational CWM Metamodel for the secure PSM

Fig. B.1 shows the extended Relational Metamodel from CWM whose instances serve as secure PSM (relational) in activity A3.1 from the Implementation phase (see Section 4.5). The metamodel allows us to represent, at the logical level, all the security and audit rules captured during the conceptual modeling stage of the DWs design.

The *SSchema* (*SCatalog*) classes specialize in the schema (catalog) classes to allow a secure schema (catalog). *STable* and the *UserProfile* specializes in the *Table* metaclass. The *SColumn* specializes in the *Column* metaclass. The *UserProfile* table is a special table that stores information about users who have access to the systems. Several data types are defined in order to define the classes inherited from the *SecurityProperty* (*SecurityLevels*, *SecurityRoles* and *SecurityCompartments*) and *SecurityConstraints* (*AuditConstraint*, *ARConstraint*, *AURConstraint*) metaclasses. The associations be-

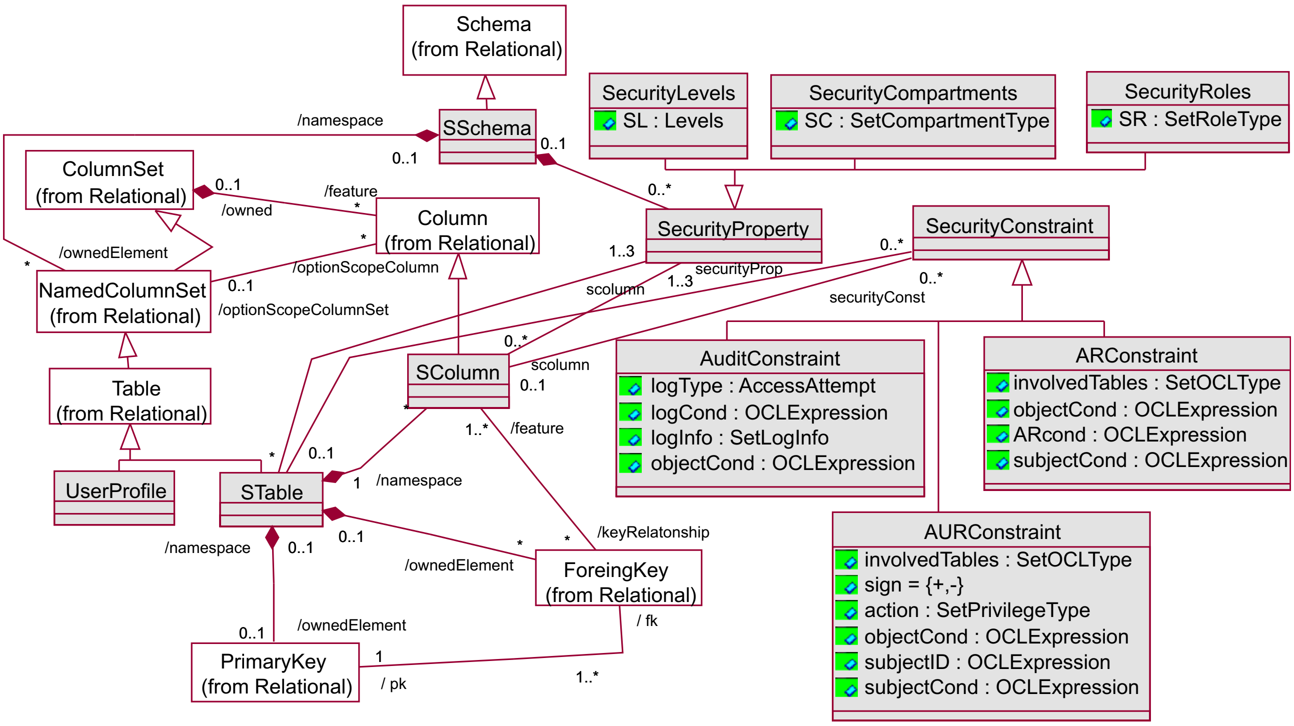


**Fig. B.2.** Metamodel used in the design of secure PIM.

tween *SecurityProperty* with *STable* and *SColumn* allow us to establish information security by means of *securityLevel*, *securityCompartment* and *securityRole*. *SecurityConstraint* (which inherits from *Constraints* metaclass) allows us to define *AuditConstraint*, *ARConstraint* and *AURConstraint*. *AuditConstraint* is useful both as a deterrent against misbehavior and as a means by which to analyze user behavior by employing the system to find out 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 for specifying multilevel security policies in tables and columns. *AURConstraint* enable 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*. See [48] for more details.

## References

[1] A. Abelló, J. Samos, F. Saltor, A framework for the classification and description of multidimensional data model, in: Proceedings of the 12th International Conference on Database and Expert Systems Aplications (DEXA'01), Munich, Germany, 2001, pp. 668–677.
[2] J.M. Cavero, M. Piattini, E. Marcos, MIDEA: a multidimensional Data Warehouse methodology, in: Proceedings of the 3rd International Conference on Enterprise Information Systems (ICEIS'01), Setubal, Portugal, 2001, pp. 138–144.
[3] R. Crook, D.C. Ince, B. Nuseibeh, Modelling access policies using roles in requirements engineering, Information & Software Technology 45 (2003) 979–991.
[4] P. Devanbu, S. Stubblebine, Software engineering for security: a roadmap, in: Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland, 2000, pp. 227–239.
[5] W. Essmayr, E. Weippl, F. Lichtenberger, W. Winiwarter, O. Mangisengi, An authorization model for Data Warehouses and OLAP, in: Workshop on Security in Distributed Data Warehousing, in Conjunction with 20th IEEE Symposium on Reliable Distributed Systems (SRDS'2001), USA, 2001, pp. 9–13.
[6] E. Fernández-Medina, J. Trujillo, R. Villarroel, M. Piattini, Developing Secure Data Warehouses with a UML extension, Information Systems 32 (2007) 826–856.
[7] E. Fernández-Medina, J. Trujillo, R. Villarroel, M. Piattini, Access control and audit model for the multidimensional modeling of DWs, Decision Support Systems 42 (2006) 1270–1289.
[8] D.F. Ferraiolo, D.R. Kuhn, R. Chandramouli, Role-Based Access Control, Artech House Inc., Norwood, MA, USA, 2003.
[9] J.F. Freeman, R.B. Neely, On security policy modeling, in: Proceedings of the 8th Annual Conference on Computer Assurance: Practical Paths to Assurance (COMPAS'93), Colorado Springs, 1993, pp. 61–69.
[10] P. Giorgini, S. Rizzi, M. Garzetti, Goal-oriented requirement analysis for Data Warehouse design, in: Proceedings of the 8th International Workshop on Data Warehousing and OLAP (DOLAP'05), Bremen, Germany, 2005, pp. 47–56.
[11] M. Golfarelli, S. Rizzi, A methodological framework for Data Warehouse design, in: Proceedings of the 1st International Workshop on Data Warehousing and OLAP (DOLAP'98), Bethesda, MD, USA, 1998, pp. 3–9.
[12] Q. He, A. Antón, A framework for modeling privacy requirements in role engineering, in: Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Klagenfurt/Velden, Austria, 2003, pp. 137–146.
[13] Q. He, Requirements-Based Access Control Analysis and Policy Specification, PhD Thesis, North Carolina State University, 2005.
[14] S. Jajodia, R. Sandhu, Polyinstantiation for cover stories, in: Proceedings of the 2nd European Symposium on Research in Computer Security (ESORICS'93), Toulouse, France, 1992, pp. 307–328.
[15] K. Katic, G. Quirchmayr, J. Schiefer, M. Stolba, A.M. Tjoa, A prototype model for Data Warehouse security based on metadata, in: Proceedings of the 9th International Workshop on Database and Expert Systems Applications (DEXA'98), Vienna, Austria, 1998, pp. 300–309.
[16] R. Kirkgöze, N. Katic, M. Stolba, A.M. Tjoa, A security concept for OLAP, in: Proceedings of the 8th International Workshop on Database and Expert System Applications (DEXA'97), Toulouse, France, 1997, pp. 619–626.
[17] R. Kimball, M. Ross, Data Warehouse Toolkit, John Wiley&Songs Publishing, USA, 2002.
[18] R. Kimball, L. Reeves, M. Ross, W. Thornthwaite, The Data Warehouse Lifecycle Toolkit, John Wiley&Sons Publishing, USA, 1998.
[19] M. Koch, L.V. Mancini, F. Parisi-Presicce, Conflict detection and resolution in access control policy specifications, in: Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'02), LNCS, vol. 2303, Grenoble, France, 2002, pp. 223–238.
[20] P. Kroll, P. Kruchten, The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP, Addison Wesley, 2003.
[21] S. Luján, J. Trujillo, A comprehensive method for Data Warehouse design, in: Proceedings of the 5th International Workshop on Design and Management of Data Warehouses (DMDW'2003), Berlin, Germany, 2003, pp. 1.11–1.14.
[22] S. Luján, J. Trujillo, A Data Warehouse engineering process, in: Proceedings of the 3rd International Conference on Advances in Information Systems (ADVIS'04), Izmir, Turkey, 2004, pp. 20–22.

[23] S. Luján-Mora, J. Trujillo, I.Y. Song, A UML profile for multidimensional modeling in Data Warehouses, Data & Knowledge Engineering (DKE) 59 (2006) 725–769.

[24] J.N. Mazón, J. Trujillo, M. Serrano, M. Piattini, Designing Data Warehouses: from business requirement analysis to multidimensional modeling, in: Proceedings of the 1st International Workshop on Requirements Engineering for Business Needs and IT Alignment (REBNITA'05), Paris, France, 2005, pp. 44–53.

[25] J.M. Mazón, J. Trujillo, An MDA approach for the development of Data Warehouses, Decision Support Systems 45 (2008) 41–58.

[26] J.N. Mazón, J. Pardillo, J. Trujillo, A model-driven goal-oriented engineering approach for Data Warehouses, in: Proceedings of the International Workshop on Requirements, Intentions and Goals in Conceptual Modelling (RIGiM'07), LNCS, vol. 4802, Auckland, New Zealand, 2007, pp. 255–264.

[27] N.R. Mead, E.D. Hough, T.R. Stehney, Security quality requirements engineering (SQUARE), TR CMU/SEI-2005-TR-009, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, 2005.

[28] J.D. Moffett, C.B. Haley, B. Nuseibeh, Core security requirements artefacts, Technical Report, vol. 23, Department of Computing, The Open University, Milton Keynes, UK, 2004, pp. 1–47.

[29] D. Moody, M. Kortink, From enterprise models to dimensional models: a methodology for Data Warehouse and Data Mart design, in: Proceedings of the 3rd International Workshop on Design and Management of Data Warehouses (DMDW'00), Stockholm, Sweden, 2000, p. 110.

[30] H. Mouratidis, P. Giorgini, G. Manson, Integrating security and systems engineering: towards the modeling of secure information systems, in: Proceedings of the 15th Conference on Advanced Information Systems Engineering (CAiSE'03), Klagenfurt, Austria, 2003, pp. 63–78.

[31] J.M. Mukerji, MDA Guide Version 1.0.1, OMG, 2003.

[32] OMG, Software & Systems Process Engineering Meta-Model Specification, ver. 1.1, 2005. <http://www.omg.org/docs/formal/05-01-06.pdf>.

[33] OMG, UML Infrastructure Specification, v2.0, 2006. <http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF/>.

[34] OMG, MOF 2.0 QVT Final Adopted Specification, 2005. <http://www.omg.org/cgi-bin/doc?ptc/2005-11-01>.

[35] OMG, MOF Model to Text Transformation Language, OMG Adopted Specification v1.0, 2008. http://www.omg.org/spec/MOFM2T/1.0/PDF.

[36] F. Paim, J. Castro, DWARF: an approach for requirements definition and management of Data Warehouse systems, in: Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE'03), Monterey Bay, CA, USA, 2003, pp. 75–84.

[37] W.E. Perry, Effective Methods for Software Testing, third ed., John Wiley& Sons Publishing, USA, 2006.

[38] N. Prakash, Y. Singh, A. Gosain, 2004. Informational scenarios for Data Warehouse requirements elicitation, in: Proceedings of the 23rd International Conference on Conceptual Modeling (ER'04), LNCS, vol. 3288, Shanghai, China, 2006, pp. 205–216.

[39] T. Priebe, G. Pernul, Towards OLAP security design – survey and research issues, in: Proceedings of the 3rd ACM International Workshop on Data Warehousing and OLAP (DOLAP'00), VA, USA, 2000, pp. 33–40.

[40] T. Priebe, G. Pernul, A pragmatic approach to conceptual modeling of OLAP security, in: Proceedings of the 20th International Conference on Conceptual Modeling (ER'01), Yokohama, Japan, LCCS, vol. 2224, 2001, pp. 311–324.

[41] A. Rosenthal, E. Sciore, View security as the basic for Data Warehouse security, in: Proceedings of the Second International Workshop on Design and Management of Data Warehouses (DMDW'00), Stockholm, Sweden, 2000, pp. 1–8.

[42] S. Rizzi, A. Abelló, J. Lechtenbörger, J. Trujillo, Research in Data Warehouse modeling and design: dead or alive? in: Proceedings of the 9th International Workshop on Data Warehousing and OLAP (DOLAP'06), Arlington, VA, USA, 2006, pp. 3–10.

[43] F. Saltor, M. Oliva, A. Abelló, J. Samos, Building secure data warehouse schemas from federated information systems, in: H. Bestougeff, J.E. Dubois, B. Thuraisingham (Eds.), Heterogeneous Information Exchange and Organizational Hubs, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002, pp. 123–134. ISBN: 1-4020-0649–7.

[44] P. Samarati, S. De Capitani di Vimercati, Access control: policies, models, and mechanisms, in: International School on Foundations of Security Analysis and Design (FOSAD'00), LNCS, vol. 2171, Bertinoro, Italy, 2000, pp. 137–196.

[45] B. Schneier, Attack Trees: Modeling Security Threats, Dr. Dobb's Journal 24 (1999) 21–29.

[46] M. Sloman, E. Lupu, Security and management policy specification, IEEE Network 16 (2002) 10–19.

[47] E. Soler, J. Trujillo, E. Fernandez-Medina, M. Piattini, A Framework for the development of Secure Data Warehouses based on MDA and QVT, in: Proceedings of the Second International Conference on Availability, Reliability and Security (ARES'07), Vienna, Austria, 2007, pp. 294–300.

[48] E. Soler, J. Trujillo, E. Fernandez-Medina, M. Piattini, Building a secure star schema in Data Warehouses by an extension of the relational package from CWM, Computer Standards & Interfaces 30 (2008) 341–350.

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

[50] E. Soler, J. Trujillo, E. Fernandez-Medina, M. Piattini, Application of QVT for the development of Secure Data Warehouses: a case study, in: Second International Conference on Availability, Reliability and Security (ARES'07), Vienna, Austria, 2007, pp. 829–836.

[51] R. Villarroel, E. Fernández-Medina, M. Piattini, A UML 2.0/OCL extension for designing Secure Data Warehouses, Journal of Research and Practice in Information Technology 38 (2006) 31–43.

[52] R. Winter, B. Strauch, A method for demand-driven information requirements analysis in Data Warehousing projects, in: Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)-Track 8, Big Island, HI, USA, IEEE Computer Society, 2003, pp. 231.1.

[53] E. Yu, Modelling Strategic Relationships for Process Reengineering, Ph.D. Thesis, University of Toronto, Canada, 1996.