A Model Driven Approach for Generating Code From Security Requirements ⋆

Óscar Sánchez, Fernando Molina, Jesús García Molina and Ambrosio Toval

Departament of Informatic and Systems, University of Murcia, Spain

Abstract. Nowadays, Information Systems are present in numerous areas and they usually contain data with special security requirements. However, these requirements do not often receive the attention that they deserve and, on many occasions, they are not considered or are only considered when the system development has finished. On the other hand, the use of model driven approaches has recently demonstrated to offer numerous benefits. This paper tries to align the use of a model driven development paradigm with the consideration of security requirements from early stages of software development (such as requirements elicitation). With this aim, a security requirements metamodel that formalizes the definition of this kind of requirements is proposed. Based on this metamodel, a Domain Specific Language (DSL) has been built which allows both the construction of requirements models with security features and the automatic generation of other software artefacts from them. An application example that illustrates the approach is also shown.

1 Introduction

Nowadays, most organizations are depending increasingly on the use of Information Systems. On many occasions, these systems contain data with special security requirements. The suitable management of that information is critical for the survival of these organizations. However, these security requirements are frequently not considered or they are only considered when the systems have been completely developed [1] but, on the contrary, information security is a feature that must be considered in all the stages of the lifecycle of a system, from requirements elicitation to implementation and maintenance. On the other hand, the Model Driven Engineering (MDE) approaches are gaining in popularity among practitioners as an emerging approach that provides a cost-effective, reliable and rapid application development to get products faster and with more quality [2]. They propose that models with different abstraction levels should be used to drive the entire software development process. These models are built by means of Domain Specific Languages (DSL) [3] and they make possible automatic code generation. This development paradigm, together with the need for considering security

⋆ Partially supported by the projects DEDALO (TIN2006-15175-C05-03) from the Spanish Ministry of Science and Technology, and MELISA-GREIS (PAC08-0142-335). Fernando Molina is partially funded by the Fundación Séneca (Reg. de Murcia).

Sánchez Ó., Molina F., Molina J. and Toval A. (2009).
A Model Driven Approach for Generating Code from Security Requirements.
In *Proceedings of the 7th International Workshop on Security in Information Systems*, pages 119-126
DOI: 10.5220/0002199601190126

requirements in all the stages of software lifecycle lead us to consider that an MDE approach can be useful to meet this need.

With this aim, firstly we have designed a security requirements metamodel, which can be used as a mechanism for formalizing the elicitation of security requirements. Based on this metamodel, a DSL that allows developers to define security requirements models that can be used to generate other software artefacts has been devised. This automatic generation differentiates our approach from other requirements metamodelling approaches. The generative approach will be illustrated by means of an application example which shows how code for Oracle Label Security [4] and XACML security policies [5] can be generated.

This paper is organized as follows. First we present the security requirements metamodel, then the designed DSL will be presented along with an application example. Finally we summarize the proposal.

2 Using Metamodelling to Capture Security Requirements

2.1 Requirements Metamodelling

A great number of approaches dedicated to dealing with requirements have appeared. They often use textual descriptions for the requirements specification, which are gathered in non-formal models or organized in requirements documents which are hardly ever formally structured [6]. However, MDE approaches have introduced a new perspective for dealing with requirements which is based on the use of requirements metamodels that formally define the concepts and relationships involved in the RE process [6]. Several approaches for requirements metamodelling have recently appeared[6–8] but a reference model has not been established yet [6]. In our approach, the core concepts from these requirements metamodelling proposals have been extracted and, from them, the requirements metamodel shown in Figure 1 has been designed, which includes new relationships and concepts considered relevant.

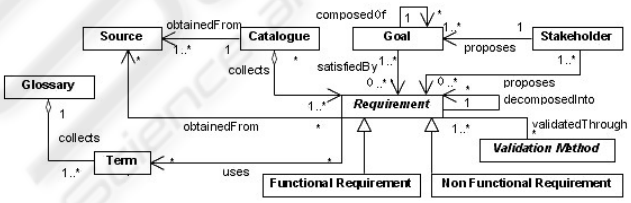

Fig. 1. Proposed requirements metamodel.

The key element in Figure 1 is that of `requirement`, which can be described by using a set of attributes (hidden in the figure) such as an `identifier` and its textual `description`. Requirements can be classified as either `functional` or `non-functional requirements`. Due to the need for traceability with business objectives, each requirement is connected to the set of `goals` that contributes to satisfy. Every goal can be composed of other goals, and captures a high-level objective from

one or more stakeholders that propose it. Another concern is related to the reuse of requirements, which is tackled by including a catalogue concept. This concept serves to gather a set of requirements extracted from one or more sources (*i.e.*, a law, an organization policy, a particular domain,...) and that can be reused in all the projects in which these sources are applicable. Still another concern is that of vocabulary disambiguation. Since requirements are usually described by using natural language, which can be imprecise and ambiguous, the metamodel considers the possibility of describing requirements using a set of well-defined terms gathered into a glossary. Finally, possible methods for assuring the fulfilment of requirements can be modelled through the concept of Validation Method.

2.2 Extending the Requirements Metamodel with Security Concepts

Once the requirements metamodel of Figure 1 has been defined, the next aim is to extend it with specific security concepts (see Figure 2). In order to ease the explanation, we have classified them in several categories: basic security concepts, security requirements, and access control methods.

The basic security concepts are: Asset, Threat, Safeguard and Contingency Plan. These terms have been extracted from MAGERIT [9], which conforms to the standard ISO/IEC 15408-1999 (also known as the Common Criteria Framework [10]). An Asset is a physical or logical object that has value itself and deserves to keep some guarantees over it. Assets can have different types, for instance, documents, data tables, and so forth, and they are important for a business, which is measured with an impact index. An Asset can be damaged by a Threat, which has properties such as its type, frequency (modelled as an annual rate), a concrete success probability and a degradation (that is, the level of damage caused in an Asset if the threat achieves its goal). Safeguards serve as a crackdown on a risk in order to reduce it. As shown in the type attribute, we will distinguish between Safeguard Functions and Safeguard Measures. The former are actions which reduce the risk whereas the latter are physical or logical devices or processes that reduce the risk. Two operation modes are distinguished for the safeguards: preventive if they act before a threat had taken place and curative if they act on damaged assets. For the sake of softening a threat that can give rise to damage, a detailed Contingency Plan composed of a set of safeguards is recommended.

There not exists an standard classification for Security Requirements, so based on [11, 12], five categories of them have been considered, which tackle five categories of threats, according to the characteristics that give value to the assets. These categories are: privacy, integrity, authentication, availability and accountability. Frequently there exist sets of requirements which are related to the same asset, soften the same attack and achieve the same security objective. This concept, which has been extracted from [13], is introduced in our metamodel as a Security Requirements Cluster. Regarding to Privacy and Integrity requirements, they are directly associated with an Access Control Method which has a validity period. The different methods considered are Permissions (DAC), Security Levels (MAC)

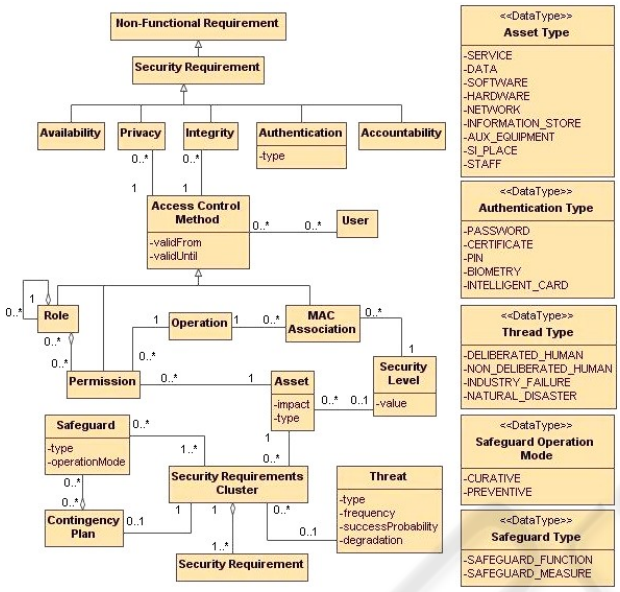

Fig. 2. Extending the requirements metamodel of Fig. 1 with security concepts.

or `Roles` (HRBAC) [14]. The concept of `MAC Association` has been introduced for associating a security level and an operation to an user.

Bearing the above security concepts in mind, we have designed a security requirements metamodel that is obtained by extending the requirements metamodel shown in Figure 1 with the specific security concepts shown in Figure 2. The reader must take into account that for lack of space solely the most important attributes are shown, and that the data types depicted are not all the possible values but some of them. As can be noticed, the link between both metamodels is the `NonFunctional Requirement` metaclass. `Security Requirement` is a specialization of `NonFunctional Requirement`, with the aim of dealing with requirements in a more suitable way.

3 A DSL for Security Requirements Management

From the defined security requirements metamodel, we have devised a generative approach based on a graphical DSL specifically built to specify security requirements models. This DSL [3] allows users to create models in a handy and easy way, as well as get software artefacts from these models. The Eclipse platform has been chosen for building this graphical DSL. Specifically, the abstract syntax has been defined using Ecore, the concrete syntax has been established with the Graphical Modelling Framework (GMF [15]) and the semantics has been provided by means of model-to-text transformations expressed in MOFScript templates [16].

The process of building our DSL and enable software artefacts generation is as follows: (i) define the security requirements metamodel we have designed by using Ecore (see Figures 1 and 2), (ii) define a graphical notation for the DSL through GMF, (iii)

specify the rest of elements that GMF needs to work, such as the definition of the elements in the DSL toolbox and (iv) define MOFScript templates in order to generate code from models created with the DSL. To illustrate our approach, templates for generating SQL and XACML code have been implemented, which will be explained in Section 4.

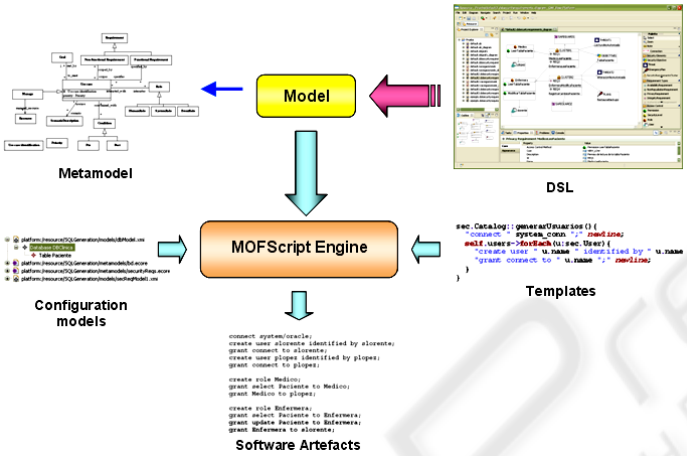

Fig. 3. Overview of the generative architecture.

The designed DSL has been splitted into two connected views. The first one deals with the general requirements concepts (see Figure 1) while the second one deals with the specific security concepts that appear in Figure 2. This separation of concerns in two views eases the modelling task and makes the tool handy.

The process of using the DSL is depicted in the Figure 3. Firstly, the developer uses the DSL to create security requirements models that conform to the metamodel. Secondly, the developer sets those features which are too specific for being included in the requirements model (e.g. details regarding to a specific platform, DBMS or security protocol). Following the MDE philosophy, we propose the use of models with this aim. In our case, two simple metamodels have been built: a database metamodel in order to model the DBMS specific features and a policy metamodel to specify the roles or users allowed. The developer only needs to instantiate these metamodels to configure the generation process. These configuration models provide a mechanism to parameterize the MOFScript transformation in an easy way.

Finally, a template is executed with the MOFScript engine which also takes the requirements model and the configuration model as an input, with the aim of generating software artefacts (code in this case) that implements the requirements. Although we obtain database and XACML policies, any sort of software artefacts can be generated such as security reports, Java code for interacting with a LDAP server or XML configuration files.

4 Application Example

A web application for the management of medical patients adapted from [17] is proposed as an example of the use of our security requirements DSL. The DSL editor with a simplified part of the example model is shown in Figure 4. It is important to highlight that the Figure 4 focuses on the security concepts and does not show the view related to the concepts in Figure 1 due to lack of space.

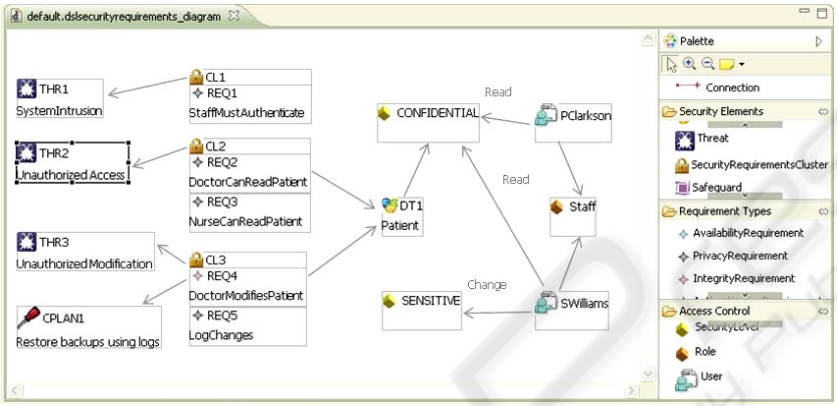

Fig. 4. A fragment of the model for the application example.

As can be seen in Figure 4, the `Patient` asset needs protecting. The `CL1` cluster tries to restrict the access to the system to the medical staff, which leads the modeller to create a role `Staff` and assign this role to doctors and nurses. The `CL2` cluster contains security requirements which try to avoid that unauthorized staff can access to information of `Patients`. In this case, the modeller grants read access for the `Confidential` level to the users `PClarkson` (nurse) and `SWilliams` (doctor). The `CL3` Cluster requirements attempts to avoid unauthorized modification of the `Patient` data, so the modeller allows `SWilliams` (doctor) to update the `Patients` table. `Sensitive` level is expected to be higher than `Confidential` level, so the result is that `PClarkson` can read the `Patients` information, and `SWilliams` can read and update this data.

After that, model-to-text transformations are automatically applied to the model to take advantage of the DSL, making use of the transformation templates previously defined. On the other hand, the DSL is, on the one hand, able to generate code for Oracle Label Security, which can be used for example in the persistence layer of the application. An excerpt of the generated code is shown in Figure 5(a). This code creates a policy, defines `Confidential` and `Sensitive` levels, creates the security labels, sets the user levels, and applies the policy to the table `Patient`. Besides the security levels modelled in the DSL, with the configuration model the developer can indicate that the generated code will include a default security level named 'public' level. On the other hand, the DSL is also able to generate XACML access policies for the web application. A short excerpt of this code is shown in Figure 5(b). Within the code, a rule

is defined for permitting users with a role attribute equals to `Staff` and a default rule for denying all others. Both transformations also require the configuration model that includes technology-dependant parameters, which has not been shown due to lack of space.

```
execute sa_sysdba.create_policy ('Access_Patient','ols_col','read_control,label_default,hide');
execute sa_components.create_level('Access_Patient',1000,'PUB','PUBLIC');
execute sa_components.create_level('Access_Patient',2000,'CONF','CONFIDENTIAL');
execute sa_components.create_level('Access_Patient',3000,'SENS','SENSITIVE');
execute sa_label_admin.create_label('Access_Patient',1000,'PUB');
execute sa_label_admin.create_label('Access_Patient',2000,'CONF');
execute sa_label_admin.create_label('Access_Patient',3000,'SENS');
execute sa_user_admin.set_user_labels ('Access_Patient','PClarkson','CONF','CONF','PUB','CONF','CONF');
execute sa_user_admin.set_user_labels ('Access_Patient','SWilliams','SENS','SENS','PUB','SENS','SENS');
execute sa_policy_admin.apply_table_policy('ACCESS_PATIENT','Test','Patient');
(a)

<Rule RuleId="AllowStaff" Effect="Permit">
  <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
      <SubjectAttributeDesignator
        DataType="http://www.w3.org/2001/XMLSchema#string"AttributeId="role"/>
    </Apply>
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Staff</AttributeValue>
  </Condition>
</Rule>
<Rule RuleId="DenyAllOthers" Effect="Deny"/>
(b)
```

Fig. 5. Excerpts of the code automatically generated.

5 Conclusions and Further Work

In this paper, we have shown the viability of using metamodelling techniques aligned to the MDE approach for the elicitation of requirements, making special emphasis in the early consideration of security requirements. To achieve this goal, we have proposed a security requirements metamodel, which is used as a basis of a generative architecture that allows to obtain automatically code from requirements models. The metamodel comes accompanied by a DSL that facilitates modellers the task of building requirements models.

As further work, the requirements metamodel and the associated automatic support will be extended for considering new security features and additional capabilities for the generation of other software artefacts such as conceptual models. For example, the possibility of generating code related to authentication and access control policies using languages such as SAML and enriching the security metamodel to make it more dynamic are being considered.

References

1. Villarroel, R., Fernández-Medina, E., Piattini, M.: Secure information systems development - a survey and comparison. Computers & Security 24 (2005) 308–321
2. Selic, B.: Mda manifestations. The European Journal for the Informatics Professional IX (2008)
3. Kelly, S., Tolvanen, J.: Domain-Specific Modeling: Enabling Full Code Generation. Wiley-IEEE Computer Society Press (2008)

4. ORACLE: Oracle label security. http://www.oracle.com/technology/deploy/security/database-security/label-security/index.html (2008)
5. OASIS: extensible access control markup language. "http://www.oasis-open.org/" (2008)
6. Goknil, A., Kurtev, I., van den Berg, K.: A metamodeling approach for reasoning about requirements. In: ECMDA-FA. (2008) 310–325
7. Vicente-Chicote, C., Moros, B., Toval, A.: Remm-studio: an integrated model-driven environment for requirements specification, validation and formatting. Journal of Object Technology, Special Issue TOOLS EUROPE 2007,6 (2007) 437–454
8. Berre, A. J.: Comet (component and model based development methodology). http:// modelbased.net/comet/ (2006)
9. MAGERIT: Methodology for information systems risk analysis and management. Spanish Ministry for Public Administration. http://www.csae.map.es/csi/pg5m20.htm (2006)
10. I.S.O.: Iso/iec 15408 (common criteria v3.0): Information technology security techniques-evaluation criteria for it security. (2005)
11. Rodríguez, A., Fernández-Medina, E., Piattini, M.: A bpmn extension for the modeling of security requirements in business processes. IEICE Transactions 90-D (2007) 745–752
12. Standard:ECMA-271: Extended commercially oriented functionality class for security evaluation. (1999)
13. Mellado, D., Fernández-Medina, E., Piattini, M.: A common criteria based security requirements engineering process for the development of secure information systems. Comput. Stand. Interfaces 29 (2007) 244–253
14. Samarati, P., Capitani, S. D.: Access control: Policies, models, and mechanisms. In: FOSAD. (2000) 137–196
15. Eclipse: Eclipse graphical modeling framework. http://www.eclipse.org/gmf/ (2008)
16. Eclipse: Generative Modeling Technologies (GMT): MOFScript. http://www.eclipse.org/gmt/ (2008)
17. Fernández-Medina, E., Piattini, M.: Designing secure databases. Information & Software Technology 47 (2005) 463–477