A Framework based on Security Patterns for Transformations

Fábio Sarturi Prass[1], Lisandra Mazoni Fontoura[1], Osmar Marchi dos Santos[1]

[1]Programa de Pós-graduação em Informática, Universidade Federal de Santa Maria (UFSM)
Avenida Roraima, 1000 - Cidade Universitária
97105-900 - Bairro Camobi - Santa Maria - RS, Brasil

fabioprass@gmail.com, {lisandra, osmar}@inf.ufsm.br)

***Abstract**: Security Patterns can be used in systems to protect shared data and information. They use the security specifications for controlling access to resources and prevent security violations. The increasing complexity of systems and the natural growth in the cost required to develop software, make the search for alternatives that can shorten the development effort becomes increasingly important. One of these initiatives is the MDA approach that allows modeling and application of transformations on the models in order to obtain the software in an automated way. Therefore, we propose a framework based on security patterns oriented model, providing guidelines for implementation of the application model, the validation of the correct use of patterns and the automatic generation for a specific platform. Security is implicitly inserted in the system by means of the transformation between models and automatic encoding, ensuring that security will not be violated at any level and will not be susceptible to errors or alterations in the code.*

1. Introduction

With the intention to obtain greater agility in developing systems and achieve the inherent benefits code generation tools have been emerging, which use templates, wizards, declarative development, among other features. The use of these tools in the development of applications require from the developer an extremely high value, which is the dependence of the solutions generated, making it more complex to perform changes in automated source codes for various reasons, among which, the algorithms may have generated high level of complexity (Brown 2004).

Several frameworks and other tools that use code completion have come up with the same purpose, to accelerate the processes of coding applications (Rosado et al 2003). Although the code complementation is extremely satisfactory, the essence and responsibility of the correct encoding remain assigned to the developer. While these tools do constant syntax checks, they cannot guarantee in development time, the correct use of objects and not even of the completeness and operating consistency of algorithms (Fernandez and Pam 2001).

The Object Management Group (OMG), seeking an alternative to the automation of development, created the approach Model Driven Architecture (MDA) as an alternative for the use of documentation and code generation in an automated form, in order to provide an evolutionary step in the process of software development. The MDA uses the notions of models generated in the steps of modeling for software development. This architecture also seeks to allow the specification of a system independently of the

platform and provide means of transformation of this specification for specific platforms (OMG 2003).

Having in view the development of secure systems, there are the security patterns that describe good solutions for the problems of security and can be reused in different contexts. However, to achieve the benefits of using these patterns is necessary to know when, where and how to use them, from their definitions, because, security faults can be introduced by incorrect implementation of security patterns (Schumacher 2003).

Security patterns in combination with MDA allow security to be integrated in the systems modeling as the project templates are integrated to the security models and therefore are used to automate the construction of systems from these models, assisting the creation of flexible structures and ensuring that security requirements are considered during all system development phases. Therewith, security failures can be prevented from modeling and implementation is maintained consistent with the security policy, beyond allowing migration to new platforms (Yoshioka, Honiden and Finkelstein 2004).

Applications such as software, electronic commerce and other similar within the IT, have security as essential requirements and despite the wide variety of structures available, constantly there are headlines about security faults and vulnerabilities in software systems (Fink, Koch and Pauls 2006). As the security structures appear, they are quickly overcome by different reasons. Most of the time do not provide an architectural model flexible enough to follow the needs of the business, which is under development, or even because they limit the scope of the architecture that the business needs. Due to the diversity of business requirements of each application, software architects and analysts increasingly need to create their own structures and keep their integrity and coherence in relation to the business needs to which they meet (Cunha 2007).

Knowledge of these developers can be incorporated into patterns, which describe proven solutions to recurring problems. Patterns, therefore, facilitate the generation and documentation of known and established solutions. These encapsulated solutions are very valuable in developing new systems, evaluating existing systems and can also improve the process of communication and learning (Hafiz, Adamczyk and Johnson 2007).

A variety of different types of patterns has been considered in the literature: patterns are developed at different levels of abstraction, ranging from fundamental paradigms on the organization software of systems or management patterns for concrete implementations of particular project decisions (Weiss and Mouratidis 2008). Therefore, a model driven approach can help the development of secure systems, providing an easier implementation of security patterns and, therefore, ensuring adherence to architectural guidelines of the organization and security issues. This paper proposes a framework for developing secure software using security patterns and MDA-based approach applied to model transformations

The sections of this paper are structured as follows: In Section 2 are discussed related works, in Section 3 we present the transformations between models and how these changes will be made, in Section 4 it is presented the metamodel generated from the models of security patterns selected, in Section 5 are shown the steps of the use of the metamodel with the rules of transformation aggregated model at each stage of the

process by creating a development framework, in Section 6 is proposed a case study with the application of the framework and in Section 7 are described the conclusions, identifying open research issues about modeling security patterns, listing still some possibilities for future work.

2. Related works

Some approaches use MDA to develop security structures. Kienzle, Elder, Tyree and Edwards-Hewitt (2002) present an MDA approach to security engineering that uses a modeling language based on the Unified Modeling Language (UML), which integrates the Role-Based Access Control (RBAC) in the development process of model driven software. Security information integrated into the UML models is used to generate access control infrastructures. Security addressed to model provides methods and tools to integrate security into the development process. The main idea is the use of abstraction, constructing visual models that integrate the project of the system with the security project and uses techniques of automatic code generation to automate the construction of systems from these projects. The project model is combined with the security model, creating a new type of model, called of Security Design Model. In this model, security policies refer to elements of the system model, such as components, business objects, methods, attributes, etc..

The work of Kienzle, Elder, Tyree and Edwards-Hewitt (2002) is specifically conceived for access control and authorization properties. One difference is that the work proposed in this paper is more general to check variable types of security property, since it can be expressed in first order. Finally, we are interested in the security of the validation of the property in abstract specifications of models and not in investigating the generation of code from these models. It may be possible to use model transformations to refine the Platform Independent Model (PIM) to Platform Specific Model (PSM) and partly generate the code for the system.

Fernandez and Sorgente (2005) propose the use of three patterns that correspond to security models: Authorization, RBAC and Multilevel Security. The methods allow users to define models using the UML notation and use the Object Constraint Language (OCL) to specify restrictions. Authorization represents the matrix model of access and aims to describe authorizations by active computational entities (subjects) to passive resources (objects). The RBAC pattern is intended to describe the assignment of permissions to users according to their roles in an institution. Now the pattern Multilevel Security aims to help in the decision to access in an environment with security classifications, this is, environments where information and documents have security levels, such as secret, confidential, etc.

Cunha (2007) presents an outline for control oriented access model in which security is introduced into the system through the transformation between models and automatic codifications. So, the security requirements of applications generated by the outline become less susceptible to errors or changing of the code, ensuring the quality of the generated system. The security outline called of ArchiMDAs is divided into two modules, one responsible for the security rules, which is called the security rules module and the other responsible for managing of the specific security of each application, called security administrative module. The security rules module is responsible for security authentication and authorization rules of the artifacts of the application where the modeling process of business rules does not suffer change. The

administrative module of security presents the security model of the application, which depends on the access control policy of the organization.

Despite the work of Fernandez and Sorgente (2005) and Cunha (2007) allow the definition of policies to control the access using UML models, when using models of permissions, system security is enabled through the modeling of access control policy performed manually. In addition, the junction of the two models, the business one and the security one, may cause an overload of assignments to the system developer, raising the complexity of development and thereby increasing the risk of incidence of security failures. Another downside is the pollution of the business model, making it difficult to understand and maintain.

In the work presented in this article, the project model is combined with the security model, this is, the PIM of business is enriched with new modeling elements representing the access control policy for an application. Thus, the PIM is replaced by modeling elements representing roles, permissions, etc.. In Table 1 can be seen the comparisons and divergences of these works.

Table 1. Comparison among related work

Paper	Patterns	Models	MDA	Rules	Focus
A Pattern Language for Secure Operating System Architectures (Fernandez and Sorgente 2005)	3 patterns: Authorization, Control of Access Based in Papers and Multi-level Security.	No	No	No	Operational System
ArchiMDAs: An outline of security based on transformations of model (Cunha 2007)	RBAC extended with the creation of 10 security patterns	Yes	Yes	Yes	Administrative Areas
Security Patterns Repository, version 1.0 (Kienzle, Elder, Tyree and Edwards-Hewitt 2002)	26 patterns and 3 mini patterns	No	No	No	Web Applications
Proposed work	28 patterns	Yes	Yes	Yes	Any Application

3. Model-Model Transformations

MDA defines a sequence of activities for software development. These activities are divided into stages, beginning with a survey of system requirements and elaboration of the Computation Independent Model (CIM), based on these requirements. From the CIM it is generated the PIM which is a model with a high level of abstraction, independent of any implementation technology, where all the system functionality and constraints of the business will be modeled (Basin and Doser 2005). A transformation must be applied to the PIM so that a PSM can be generated. This model combines the specifications in the PIM with the details that specify how the system uses a particular type of platform (OMG 2003). As PIM is an initial model, a complete MDA

specification consists of a basic PIM, besides one or more PSMs and sets of interface definitions, each describing how the base model is implemented in a different middleware platform (Opengroup 2007).

These models and their improvements are a critical part of the development methodology for situations that include refinements among models representing different aspects of the system, besides details of the model, or conversion between different types of models. This perspective is viewed as a transformation that is the activity to generate a target model from a source model, according to a mapping, consisting of a set of rules that together describe how a model in the source language can be transformed into a model in the target language.

The OMG (2003) presents five approaches to the transformations: brands, metamodel, models using types, application templates and union of models.

- Tags - after choosing the platform, it is created, if not already available, a mapping for this platform, which has a set of tags that should be used to mark elements of the PIM in order to be transformed into a PSM.
- Metamodel - PIM is constituted using the platform-independent language specified by a metamodel. A platform is chosen and is build, if not already available, a specification of a transformation mapping the PIM metamodels in the PSM.
- Models using types - follows the same model of transformation as the previous, but uses independent types of platforms. The elements of the PIM are subtypes of these independent types of platform.
- Applying of templates - can be used in transformations of frameworks and models transformations to map the PIM for types and templates in the PSM.
- Union of models - these transformations are based on the union of models, as, for example, the use of patterns.

The MDA transformations used in this paper are based on mappings with metamodels, since it brings as advantage to the process the low cost because it is a free tool, and high flexibility for facilitating major changes in the transformation directly through the interface of the editor of rules Atlas Transformation Language (ATL).

4. Modeling Security Patterns

The creation of a metamodel in the PIM level has a level of detail abstract to the platform. Each model must be instantiated from another model that represents a domain, an area of knowledge under which many different models can be built. Based on the elicited requirements, the first step was the development of a set of models of security patterns.

In the development of the proposed metamodel, it is adopted security patterns classified by Schumacher (2005) as Access Control Models, Access Control System Architecture and Operating System Access Control. This classification contemplates patterns that specify models of access control oriented to objects, declaratives that can be used as guidelines for building secure systems, forms used to collect the fundamental requirements of a system considered from a generic set of requirements of access and mechanisms directed to describe how the operating system controls access to resources.

These patterns were adopted because they are aimed at intermediate users' access to information based on activities that are developed by them in the system.

In Figure 1, is possible to visualize a range of security patterns described in a class diagram. Initially, each pattern is conventionally specified in declaratively form. Its specifications are generic in the sense that they capture good design practices in a form independent of domain. These declarative representations, which are models of security patterns, then are instantiated in concrete representations of specific domain (Rosado et al 2006).

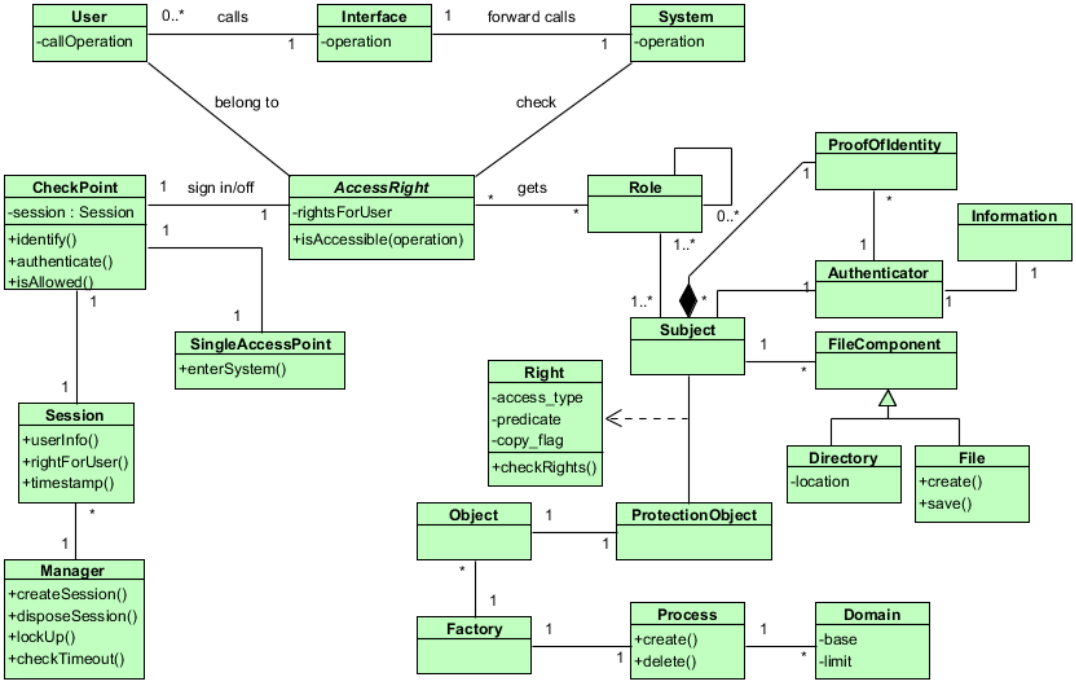

Figure 1. Class Diagram of security patterns

The central idea of this modeling is that the user performs different roles in a system. A role can be defined as a set of activities and responsibilities associated with a particular position or function within an organization. Therefore, permissions are granted to roles and users are allowed to exercise them. The access control facilitates management of authorization, because when the user´s assignments change – being, therefore, disassociated from one role to another – the maintenance of the permissions of the roles does not suffer changes. In general, one works with the principle of least privilege, in which a user activates only the subset of papers that needs to perform an operation, and this activation may or may not be subject to restrictions.

The mechanisms that are part of the access control implement policies that allow the user to assign access rights to their computing resources according to their need. Such mechanisms are based on the idea that the owner of the information should determine who will have access to this information. The control allows data to be freely copied from object to object, so that even if access is denied to the original data, one can gain access to a copy. However, if the user does not correctly assign these rights, or even if the user does it but allowing access of copy to other subjects, the dissemination of this information in the system cannot be controlled. The access control imposes no restriction on the dissemination of rights and the own evolution of the matrix of access.

In this context, one can see that the big change occurs when security is enabled because at that time on the transformation of the PIM to PSM the new security requirements require that a new PSM is generated including security artifacts. Consequently, the transformation from PSM to code is also changed due to the fact of the creation of all the security infrastructure of the system. So this metamodel contributes to security integration during development of the system, simplifying the development, increasing productivity and quality of security specifications, thereby adding considerably the quality and maintainability of the systems built. The separation between the implementation of the business and security solution facilitates the evolution of the solution so that it does not become obsolete, mainly due the fact that security can be generated automatically.

5. Application of the metamodel to transformations

To apply the developed metamodel it is adopted model to model transformations of the MDA process. This category performs the transformation of an input model (UML) in an output model (XMI - XML Metadata Interchange). An analogy can be made between general transformations and generic classes (or templates) used in object-oriented languages. Therefore, while the parameters of generic classes are connected in time of design, the variable types in transformation rules are replaced only at runtime (Yoshioka, Honiden and Finkelstein 2004).

The goal is to demonstrate how the transformation rules of the MDA approach can be specialized to be called, in this paper, framework. In this approach, the languages for general-purpose modeling and transformation functions are extended by modeling and transformation rules to integrate security into the development process (Selic 2005). These concepts are used to specify the security properties of the target system and generate the corresponding security mechanisms.

In order to describe the necessary rules for the generation of the proposed framework, is used as a development language the ATL, which is a language for performing model transformation for models in the context of MDA. Figure 2 illustrates the stages of conception and use of the framework with the requirements of the model of security patterns, which has the following steps:

1. Detailed in Figure 1, proposes a general scheme for integration of the requirements of security patterns in class models. The models should be established so that use patterns of understanding between all parties involved. At this stage the source model is more abstract than the target model. At the stage of creating models from existing artifacts, the source model is more refined than the target model. Abstraction can be defined by removing details irrelevant to the context and refinement is an antonym for abstraction.
2. Presents a class model of a system that will suffer the inference of the item described above.

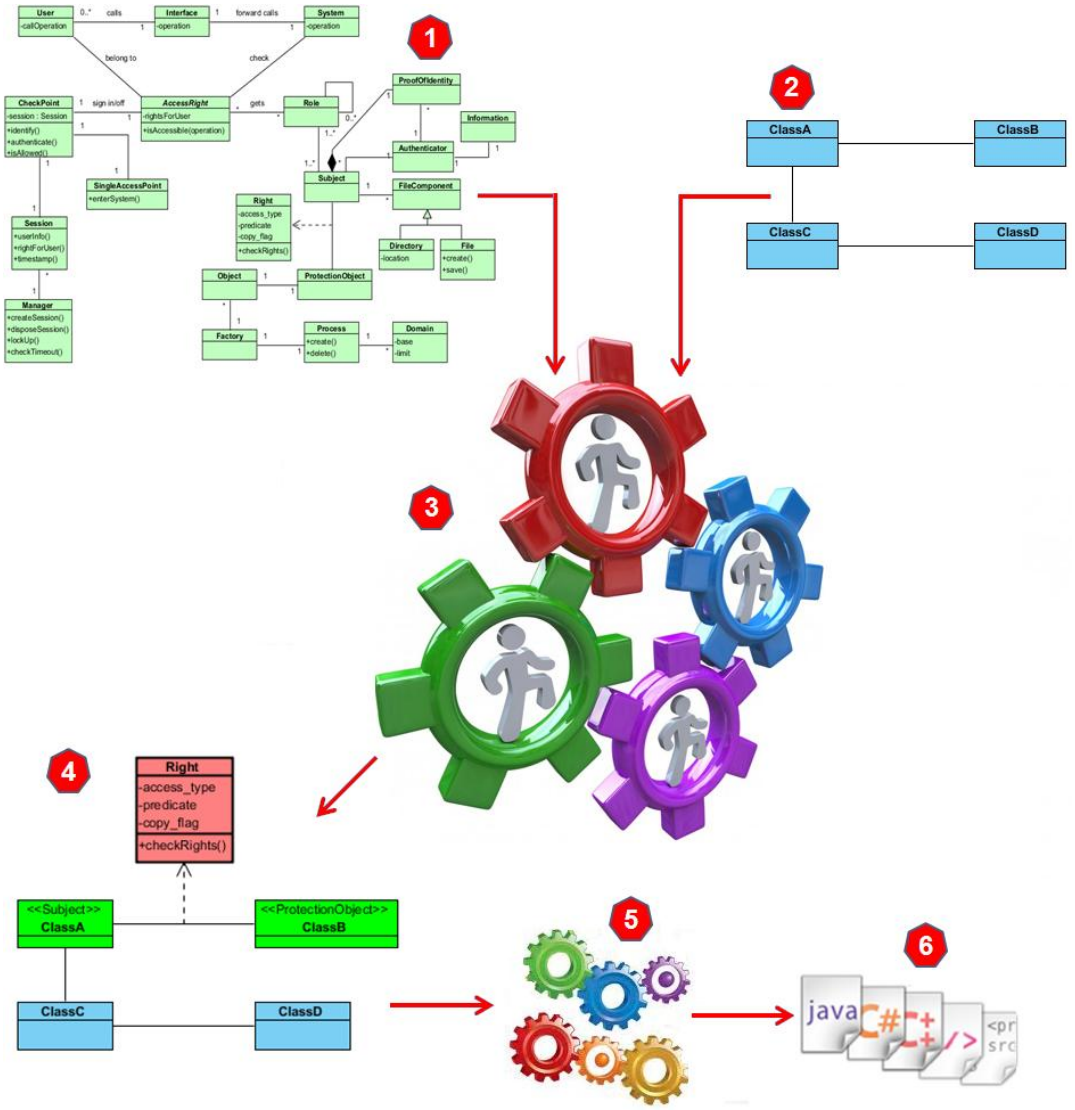

Figure 2. Steps for applying the framework

3. The main idea is to add security modeling languages that are specific to the resources that should be protected. At this stage, security experts must determine the dependencies between the patterns referring to the section of the metamodel of security patterns. Moreover, the possibility exists that dependence could be verified, taking in account the problem and the context of the pattern applied, and also to the context, the solution, and the consequences of all other patterns. So, the transformation system (in ATL) uses an input model (UML) and generates an output model (XMI), and the security pattern is applied, causing a transformation in the model, this is, the developer selects the security pattern and the model inputs and parameters and the system transforms the model and the model outputs with the security pattern chosen. To make the transformation from PIM to PSM should be taken design decisions. Such decisions may be taken during the design development, which sets the requirements in the implementation. This is a

useful way, because these decisions are taken in the context of a specific implementation design.

4. The model generated has the same properties, attributes, functions, etc, than Item 2, aggregated by the selected security patterns. For each relationship created, there is the targeting of a stereotype of the chosen pattern. As an example we take the pattern Authorization in which is added the stereotype ProtectionObject for the class that corresponds to the role of protective object to be used by the developer as an input argument. It is added, also, a stereotype for the class that corresponds to the role Right and the relationship among the classes Subject and Right. At this stage the model is complete regarding the classification, structure, invariants and pre / post conditions. The developer can specify requirements directly in the model, using an action language. This makes the PIM computationally complete, this is, the PIM contains all the information needed to produce a program code. In this context it is not necessary that the developer see the PSM, nor is it necessary that information be added to the PIM.
5. The transformation tool interprets the model directly or transforms the model directly in the source code of the program. It is from the PSM that will be generated the source code, where the work of generating it is in charge of a tool that reads the PSM and writes the code. There are several tools, available to generate code using MDA approach.
6. At this stage we have the source code for the language chosen in advance.

At the end of these steps, the application model aggregated with security patterns, can be used generic transformation rules to generate the source code, as shown in item 5, in which it is received an input model (for example: model of classes of UML from item 4), and performed the generation of source code for a specific language (for example: Java, C + +) referring to item 6, called transformations of type model-text that can be considered special cases of transformations type model-model. Therefore, the source code generated by the transformation is represented by a model that is subsequently serialized in text file.

6. Example of implementation

Declarative form is used to write a transformation, which means that the mappings can be expressed in a simple form. However, the imperative construction is provided so that some mappings too complex to be treated in a declarative form can still be specified. An ATL transformation software is composed of rules that define how source model elements are combined to create and initialize the elements of the target models.

To illustrate the use of the framework, considers the diagram shown in Figure 3 which describes the movie rental at a video store. The model faces two problems regarding security requirements. The first problem is that there is no way to assess whether the user is an "Employee" or not. A third person could pass for an "Employee" in order to abstract information from the user. The second problem is that all related to the rental company have permission to read and write information of the "Client". Even if the first problem is solved, the second problem remains. A potential problem is that someone could illegally rent a movie to any one customer.

In order to apply security to the example described, have again the steps to its utilization:

1. The developer selects the security pattern. To illustrate the use, its take as example the pattern Authenticator.
2. The developer defines the model inputs and parameters for the proposed system.
3. The system transforms the model and the model outputs with the security patterns applied, this is, the framework transforms a UML model in XMI format inserted by developers, and the security pattern is applied, causing a transformation of the model. The system deals with two models as input and output: class diagram and sequence diagram. These are described in XMI format. The transformation rules, once described, can be reused. The consecutive application of security patterns, taking into account the dependencies among the patterns, becomes possible using the result model as the model of input for the further transformation.
4. After selecting the ATL file (aggregated to the framework) in which the pattern Authenticator is described, the developer defines the inputs of the model and parameters for the system. If the developer informs "Name=Employee", the system decides that the class "Employee" corresponds to the role "Subject" and applies the default. A stereotype is added to indicate such a relationship. Figure 4 shows the class diagram after the Authenticator pattern has been applied.

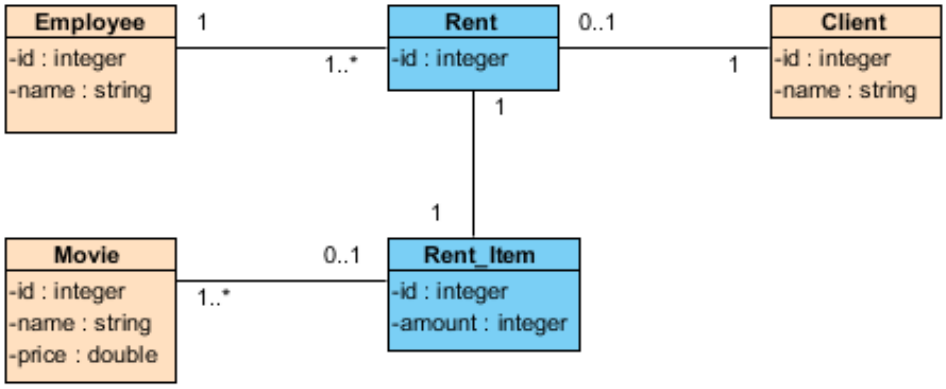

Figure 3. Diagram of product request by a client

5. The specification of the transformation can be divided into two logical parts. The first part performs the adaptation of the source model and the second part contains the transformation rules. Each class generates a sequence of relationships derived from their attributes. Every relationship is a sequence of attributes and will be transformed into a column. If an attribute is a primitive type then the relationship is the attribute itself. If an attribute is a non-primitive type, it results in a set of relationships derived from remnants of its type, preceding the attribute for each relationship. Relationships represent the paths to the primitive attributes for a given class after applying leveling.
6. The marking by stereotypes chosen as the shape of the model transformation

is used to show where a security pattern is applied. Therewith one have the security patterns applied directly on the source code of the application generated.

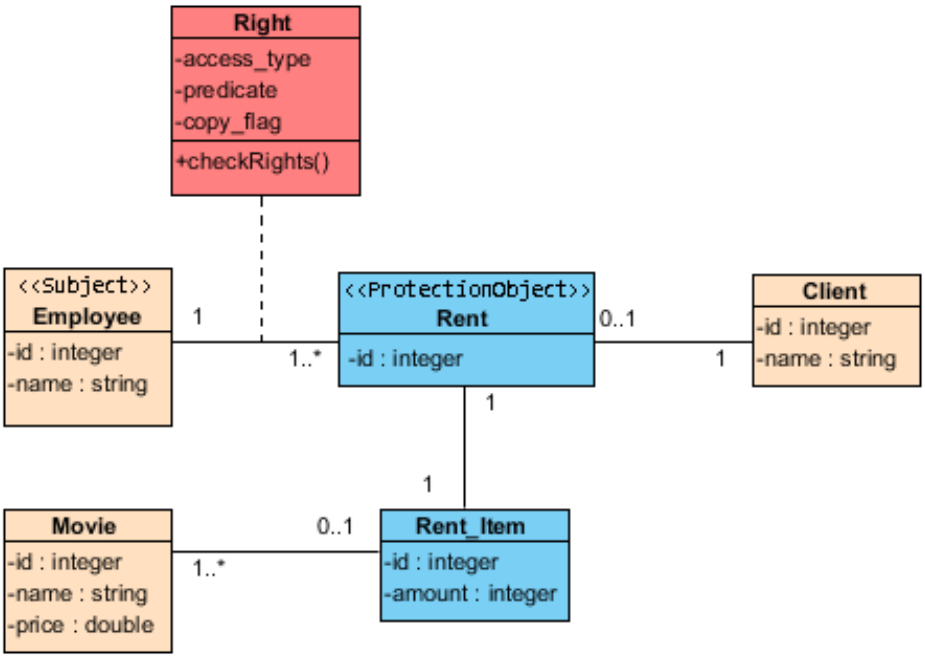

Figure 4. Diagram of product request by a client with stereotypes of the framework

7. Conclusions

This paper proposes a framework that uses security patterns to reinforce security in applications. Some patterns are dependent of others and so are associated. However, others do not have a sequence of implementation and can be implemented according to the security plan of each organization.

This proposed allowed automatic application of security patterns that are dependent on each other by creating a method to describe the transformation rules, marking the point in the model which was transformed.

It has been verified an ease of creation and maintenance of transformation rules to monitor the MDA process using the strategy to implement transformations through ATL. This is justified because the developer does not need to become familiar with a new development environment, but to study the application and the patterns to be used. Moreover, at each need to modify the transformation chain of the process becomes easier, more flexible, and reusable to give maintenance in the rules in ATL. As future work is planned a more detailed case study involving a more complex application, in line with a higher security pattern mapping available in the framework defined.

8. References

Basin, D.; Doser, J. (2005) Model Driven Security: from UML Models to Access Control Infrastructures. In 5th International School on Foundations of Security

Analysis and Design, FOSAD.

Brown, A. W. (2004) Model driven architecture: Principles and practice. Software and Systems Modeling, v. 3, n. 4, p. 314–327.

Cunha, Milene Fiorio da. (2007) ArchiMDAs: Um arcabouço de segurança baseado em transformações de modelos em MDA. Rio de Janeiro, 2007. Dissertação (Mestrado em Informática) – Instituto de Matemática - Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

Fernandez, E., Pam, R. (2001) A Pattern Language for Security Models. Dept. of Computer Science & Engineering, Florida Atlantic University.

Fernandez, E., Sorgente, T. (2005) A pattern language for secure operating system architectures. Proceedings of the 5th Latin American Conference on Pattern Languages of Programs, Campos do Jordao, Brazil, August 16-19, 68-88.

Fink, T., Koch, M., Pauls, K. (2006) An MDA approach to Access Control Specifications Using MOF and UML Profiles. In Proceedings 1st International Workshop on Views On Designing Complex Architectures

Hafiz, Munawar; Adamczyk, Paul and Johnson, Ralph E.. (2007) Organizing Security Patterns. IEEE Software, July/August pp. 52 – 60.

Kleppe, A.; Warmer J.; Bast, W. (2003) MDA Explained - The Model Driven Architecture: Practice and Promise. Addison-Wesley.

Kienzle, Darrell M.; Elder, Matthew C.; TYREE, David; HEWITT, James Edwards. (2002) Security Patterns Repository Version 1.0. Disponível em http://www.scrypt.net/~celer/securitypatterns. Acessado em 15 de Maio de 2011.

Larman, C. (2004) Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process. ISBN: 0131489062.

OMG - Object Management Group. (2003) MDA Guide Version 1.0.1. Disponível em: http://www.omg.org/docs/formal/03-06-01.pdf

Open Group. (2007). http://www.opengroup.org/, acesso em junho de 2011.

Rosado, D. G., Gutierrez, C., Fernandez-Medina, E., Piattini, M. (2006) Security patterns related to security requirements. E. Fernandez-Medina e M. Inmaculada (Eds.) Security in Informaiton Systems: Proceedings of the 4th International Workshop on Security in Information Systems. Setúbal, Portugal: INSTICC Press.

Schumacher, M. (2003) Security Engineering With Patterns: Origins, Theoretical Models, and New Applications. Springer Berlin, Heidelber.

Schumacher, Markus; Fernandez, Eduardo B.; Hybertson, Duane; Buschmann, Frank; Sommerlad, Peter. (2005) Security Patterns - Integrating Security and Systems Engineering. ISBN: 0-470-85884-2. John Wiley & Sons.

Selic, B. (2003) The pragmatics of Model-Driven Development. IEEE Software, v. 20, n. 5, p. 19–25. Tariq N. A.; Akhter N. Comparison of Model Driven Architecture (MDA) based tools. 13th Nordic Baltic Conference (NBC).

Weiss, M.; Mouratidis, H. (2008) Selecting Security Patterns that Fulfill Security Requirements. In: 16th IEEE International Requirements Engineering Conference

pages 169–172. IEEE Computer Society.

Yoshioka, N., Honiden, S. and Finkelstein, A. (2004) Security Patterns: A Method for Constructing Secure and Efficient Inter-Company Coordination Systems. In: 8th IEEE Intl Enterprise Distributed Object Computing Conf (EDOC).