Journal of Theoretical and Applied Information Technology
31st May 2017. Vol.95. No 10
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MDAASI: MODEL DRIVEN ARCHITECTURE APPROACH FOR APPLICATION SECURITY INTEGRATION

[1]LASBAHANI ABDELLATIF, [2]MOSTAFA CHHIBA, [3]ABDELMOUMEN TABYAOUI, [4]OUSSAMA MJIHIL

[1] FST, Hassan 1st University, PSI Laboratory, Settat, Morocco
[2] FST, Hassan 1st University, PSI Laboratory, Settat, Morocco
[3] FST, Hassan 1st University, PSI Laboratory, Settat, Morocco
[4] FST, Hassan 1st University, CNMM Laboratory, Settat, Morocco

E-mail: [1]abbdellatif.lasbahani@gmail.com, [2] moschhiba@yahoo.fr, [3]atabyaoui@gmail.com, [4]o.mjihil@uhp.ac.ma

ABSTRACT

There have been many research works suggesting Model-driven Architecture (MDA) approaches for automatic application generation and personalization. MDA approach allows code generation from platform-specific models (PSMs) by the means of generators that automatically transform models into the source code for a chosen platform to automate software engineering process. Previous works have widely addressed code generation, but they are not considering nonfunctional aspects such as application security. In this current work, we are proposing some additional MDA mechanisms to generate secure applications based on a given set of security policies. In this context, this approach is used for integrating security properties, such as Authorization, Authentication, Communication encryption, Message Integrity, and Confidentiality of critical data, thus security properties will be incorporated in the generated software during the whole development process or in early abstraction stages. In other words, security models will be merged with the system models in different abstraction levels by applying a set of model-to-model transformation. As a result of this process, the system's source code and configuration files will be generated automatically from communication diagrams by applying a model-to-code transformation.

Keywords: *Model Driven Architecture, Code Generation, Application Security, Communication Diagram.*

1. INTRODUCTION

Nowadays, applications are gaining increasing importance according to their broad and diverse utilization. Software tools are currently used in different areas such as banking, health-care, telecommunications, e-commerce, e-learning, and other domains. Consequently, application security becomes a determinant factor, thus any interaction between the application and its external actors shall be done according to the security policies already established by the application designers.

The security aspect is one of the most important no-functional specifications, which can be defined through five dimensions: Availability, Integrity, Traceability, No-Repudiation, and Confidentiality of critical data called (CAITN) standard. But we have limited the area of this work only to the following security properties: Authorization, Authentication, Communication encryption, and Integrity. The others aspects will be treated in future work.

Currently, there is a massive use of mobile and web applications. In order to address this increasing use, and respond to customers' needs, we have proposed a model-driven methodology to reduce building time of applications, and resolving security vulnerability problems. This approach also allows generating secure applications by taking into account no-functional aspect during system design and not during the implementation phase. So software building process should be improved for automating security policies integration at software design and generating security infrastructures. This integration will allow us to check user permissions that have been guaranteed to this user.

In fact, software Building tools should follow a well-defined software development process for getting a high-quality product, because the previous processes are limited only to establish analysis and design process of software without taking into account code generation and security integration. However, these approaches have been radically changed because of customer's needs and information technology evolution. For adapting this process with this continued growth, we need to reform, adopt, and modernize this process to find a revitalized strategy to automate generating secure software for reducing production lead time and improving planning quality.

In software development process, software can be represented by a set of models: functional models which are a structured representation of the system functionality, and no-functional models as security aspects, robustness, availability, and performance. So the both aspects can be merged to improve the quality of service. Specifications models will be enriched with even more powerful models including security policies which will be designed by security modeling language based on RBAC models extended with object constraint language (OCL)[1] during system design. In other word, generating automatically security policies infrastructure needs to incorporate security requirements in all system abstractions during system design process in the form of security models; functionality models will be extended to cover full security concerns by using OCL together with UML profile. So as to generate automatically secure applications respecting integrity constraints, structural integrity modeling was carried out in parallel with design system models to ensure the integrity of operations and data by means of OCL.we also incorporate authorization policies constraints models to ensure secure access to applications.

In this work, authorization policies are modeled using a modeling language based on Role Based Access Control (RBAC) extended with OCL and UML profile technology in order to enhance functionalities models with authorization policies concepts such as permissions, roles, and privileges. Consequently, authorization policies constraints should be verified and respected during customer interaction after implementation phase. In addition, security policies didn't limit on authorization, authentication, and integrity. But, it combines application access control with encryption of data flows and exchanges of sensitive data; data encryption has been also integrated with business models. Security integration is based on model transformation: model-to-model (M2M) and model-to-text (M2T); The models are enriched with further information during the design phase and transformed from more abstract models into more secure concrete models closer to reality by applying a set of transformation to obtain a significant gain in productivity.

To do so, we have given more attention to modeling rather than programming by using an approach based on models called model driven architecture (MDA) [2] which is an implementation of the Model Driven engineering (MDE), proposed by the Object Management Group (OMG) [3] initiated in 2000. The main objective of MDA is to develop suitable models and more productive. It also allows code generation from models. Therefore, functionality models will be refined to incorporate security information and all no functional aspects through applying a set of model transition. Then, the final products are automatically generated from these high-level specifications with a domain-specific code generator which automatically transforms PSMs models into the source code.

The main objectives of this paper are summarized as follows:

- ✓ MDAASI automates software development process (SDP) which is beginning by analysis and design phase and ending by implementation phase and reducing design mistakes.
- ✓ MDAASI proposes an approach based on models allowing security integration.
- ✓ MDAASI develops a code generator that include model-driven security and allows generate application security infrastructure.
- ✓ MDAASI allows automatically code generation for chosen platforms (JEE, .Nets, PHP, etc.) from communication diagram (CD) which is used like platform independent model called in terminology MDA (PIM).
- ✓ MDAASI allows generating maintainable, less costly and reusable software systems.

This paper is organized as follows. Section 2 summarizes related works. An Overview MDA is presented in Section 3. We present how do to generating secure applications together with MDA approach in Section 3. In this section we present utility of models in our approach. We discuss the proposed approach and an example of code generator in Section 4. In Section 5 we apply security constraints on system models. In Section 6 we discuss on tools and technicals. We present results in Section 7. And finally, we briefly discuss future works and conclude our paper in Section 8.

2. RELATED WORKS

Various works and many suggestions have been proposed in the domain of code generation and data security during the last few years. These works are very interesting and play an important role in technology evolution. These propositions have been deployed in numerous areas, more precisely for industrial communities, banking, health-care, telecommunications, e-commerce, e-learning, scientific organization, and other domains. In this work, we present some projects and suggestions of the high relevance and they will be considered as the knowledge base related to code generation and security integration.

The first type of code generator has been investigated in traditional paradigm as the study of code generation represented in many works based on code generation from Petri-Nets which has a long tradition and used only to add further semantic definition because the model UML is still largely undefined from a semantic point of view. Petri-Nets give several solutions allowing automatic generation of code. One among these solutions is presented in [4] which give an overview of different strategies to generate code from high-level Petri-Nets. However, in some review, the most of the suggestions have been focused on automatic code generation from low-level Petri-Nets as [5,6] which automates generating of controllers code. But the weakness or vulnerabilities of these approaches that automatically code generation from low-level Petri-Nets can't produce complex systems based on object-oriented principles. In addition, these approaches are used frequently in validation and verification of requirements during design system process without analyzing no-functional details.

Moreover, the second category of code generation is discovered for solving design errors, mistakes, and accidents which have appeared in the first category of code generation. So code generation in the second category is based on model transformation by using the model as a productive element promises a number of benefits including development of code with high quality, improving productivity performance, improving maintainability, reducing design errors, keep traceability between customer needs and the final code, integrating chosen platform description during design phase, and keeping a consistency between final code and design. Consequently, this category address the generated code as a models which can be transformed to other models more specific by applying a model to model transformation so as to extended the target language with other semantics and enhanced generated code with further features such as methods, interfaces classes, partial classes which allow for a single class's members to be divided among multiple sources code files, and partial method, security requirements, supported platforms, authorization policies constraints, application security files, and configuration files.

However, model-driven engineering approaches (MDE) based only on model-to-code transformation or translation from model to plain text without going on the intermediate model. So, the generated models through this process can't be enhanced and enriched with others features, so they stay limited on producing plain text based on text-based notation. According to this approach, several kinds of research have been proposed as [7] which describe how UML models of a system can be transformed into a code of an object-oriented imperative programming language or executable models by defining transformation rules which are based on the reconciliation of the differences between UML meta-model and meta-model of the target language. Other studies have been developed in this context like [8] which provides a method allowing generating operations specifications from domain class diagram using transition state diagram. While [9], addresses code generation from sequence diagram of system's internal behavior. In the both last works, they have concentrated on code generation of the operations signatures with their bodies without talked about the security properties which are a key success factor for these generated applications. So, code generation should be performed to take into account security infrastructures or rather authorization policies infrastructures and all no functional aspects. In addition, [10] present a case study of code generation based on model transformation together with stratego which is based on rewrite rules with programmable strategies for integrating model-to-model, model-to-code, and code-to-code transformation. These strategies were supported by tow dimension of transformation modularity: vertical and horizontal modularity. But this work doesn't support the entire development process, object-oriented systems, and security integration.

Furthermore, the complex tasks during the specification of conceptual schemas (CS) is system Behavior modeling, which is represented by a set of operations that are used for executing application uses cases. For that, some important approaches have been intervened for solving modeling behavior errors by simplifying the specification of conceptual schemas. [11] Provides a solution for modeling system operations and describing the system behavior, and a method allowing generating system Behavior by completing the static aspect of the conceptual schemas, and suffice to perform all typical life-cycle create/remove/update/delete (CRUD) operation. This contribution takes as input

CS expressed as a UML class diagram, which is enriched with the necessary specification, such as specification of association. Then, the new CS is generated that contains all necessary operations to start operating the system. In contrast, there are some approaches basing on a set of tools for simplifying modeling behavior.

In the security integration domain, there are some very important contributions about security integration during the design phase. According to these solutions, some technical improvements need to be made in order better to achieve secure software and enhancing software development process with other iteration allowing security policies integration. [12] Have been addressed security integration into distributed systems by providing an approach for developing secure distributed systems based on UML and additional support for specifying authorization constraints. On the other hand, others approaches have been integrated security aspects in software engineering process using a modeling language like UML which is used for modeling role-based access control policies to restrict system access to authorized users as shown in [13, 14], While some works have extended UML for solving security concerns that have an effect on productivity, quality of the software, and data confidentiality. In these works, security concerns not have been solved yet. In addition, we found also UMLSEC contribution which addresses security integration in the design phase by using UML.

In [15, 16], David basin and al. have combined secureUML with the design modeling language basing on class diagrams called componentUML, and conttrolerUML, which is also based on states diagrams in order to facilitate security integration and getting security architectures for distributed systems from models. While, [17] Provides some technical's allowing annotated UML models with authorization policies based on RBAC to authorize signers. In this context, [18] develops RBAC using MDA approach to benefit from MDA advantages for reducing systems vulnerabilities by providing a tool-supported framework which use the MDA approach with UML profile to build RBAC applications, and security specifications for generating systems security specifications in eXtensible Access Control Markup Language (XACML) format for distributed systems. There are also several works putting the accent on generating access control infrastructures for server-based applications as [19]. In addition, Model-driven security has been extended to cover database security as [20] which provides a methodology to develop secure XML databases.

Finally, [21] defines a methodology to refine application models with the appropriate security policies which have already proposed by a security administrator. In addition, [22] presents an approach for developing secure data warehouses independent of the target platform basing on UML for specifying security constraints in conceptual multidimensional database modeling. [23] Is similar to [22]. But in [23], the proposed approach is based on MDA.

This paper subscribes in the second category of code generation. Our proposal aims to generate secure applications via a code generator by applying a set of model transformation or model transition. Through these transformations and code generator, implementation phase will be performed automatically. In this work, we have generated an intermediate model (IM) for chosen platform or the target platform (PSM) to enable its extensibility by extending generated IM (structural model) for introducing others further improvements like non-functional aspects. Notwithstanding the diversity of these code generators and according to our best research, there is no complete code generator allowing generating secure applications from communication diagram. In other words, there is no complete code generator allowing security policies integration, which have already mentioned by application designers basing on solutions provided by the security expert. According to our comparison, the both categories of code generation are concentrating just on functional aspects like validation and verification of requirements, and generation of CRUD operations like [9] and al. However, these works still far to be the appropriate tools for generating automatically secure applications and they have a need to further improvements.

The motivation of this work is to complete the previous works that deal with code generation without security integration into system design phase by automating software development process based on gait (Unified Process-eXtreme Programming) (UP-XP).our proposal proposes a code generator that supports the entire development process together with security policies integration into design phase for generating secure applications with its security infrastructures.

3. OVERVIEW MDA

3.1 Mda Description

The model Driven architecture - (MDA) [24] is an implementation of Model-driven engineering (MDE) proposed by Object Management Group (OMG) in 2001, and it is used extensively for designing and building software basing on the UML standard. Furthermore, MDA provides many significant advantages. Among the benefits offered by MDA we found: the separation of concerns between the business logic of applications and used platforms. According to this separation, MDA provides three essential advantages:

- ✓ Develop sustainable models; presenting the business domain system without considering the architecture of the used platform.
- ✓ Improving productivity gain; business logic applications becomes more productive and competitive through the transformation of models and its extensibility.
- ✓ Integrating platform architecture during design system via the transformation of models; integrating of the technical details of the execution platforms through the model transformation in order to get a platform specific model (PSM).

Basing on MDA approach, the software development process becomes more modular rather than the old methodologies which are based only on UML.MDA approach defines three levels of model's abstraction for representing system model's, and for elaborating advanced design system:

The first level so-called Computational Independent Model (CIM) that gives a requirements view, and describes the situation and environment technical in which the produced system will be used.

The second level called Platform Independent Model (PIM) which represents an analysis and design view. At this level, system specifications or business logic of system will be represented without considering used platform in which the system will be deployed.

The third level called PSM which is obtained from PIM and gives a code view.PSM combines the system functionalities with a chosen platform-specific in which the system will be used.

Practically, PIMs can be represented by the domain class diagram, sequence diagram of system’s internal or external behavior, collaboration diagram, transition state diagram, or communication diagram. In this contribution, we have used communication diagram (CD) as PIM Because CD more relevant to design phase rather than others. Then, this PIM can be translated to one or more platform-specific models (PSMs) including CORBA, .NET, J2EE, etc...

In addition, success secret of MDA resides in model transformation strategy which allows automating code generation and to obtain a significant gain in productivity. In MDA approach, to go from PIM to PIM or PSM, or PSM to PSM or final code, a model transition is the obligatory stage. For that, these transitions involve some mechanisms for model transformation. Regarding model transformation, OMG has proposed a set of the tools and transformation languages in order to cover these transitions between different levels of model's abstraction. We quote Atlas transformation Language (ALT) [25] and Query, View, and Transformation (QVT) which may be considered as the most appropriate model's transformation languages. There are also others tools.

In MDA, the language used to create and validate different MDA models is called meta-model or meta-modeling language which defines the structure of the models; UML technology is the appropriate meta-model in MDA methodology. This meta-model is also represented in the form of model, and collects a set of classes. Meta-modeling language has also its meta-model called meta-meta-model which is used to describe a meta-model structure, and define a semantic to describe meta-model architecture. In MDA context, this model known as Meta-Object Facility (MOF) [26], and which has the ability to describes itself.

Consequently, the final code will be generated semi-automatically from PSM models through models transformation. Figure 1 describes the MDA architecture.

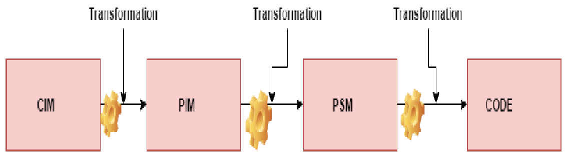


Figure 1: MDA architecture

3.2 MDA and Proposed Methodology

In this work, we have proposed a new MDA methodology allowing security integration and code generation by including data encryption, secure communication, authentication, Message integrity, and authorization concepts during system design phase. In addition, we will improve this approach to consider views generation from uses cases in the future works. Figure 2 describes our methodology in detail.

3.3 MDA for Application Security

As already mentioned, MDA has been proposed by OMG to automate software building and allowing code generation for chosen platform such as PHP, CORBA, NET, or J2EE basing on MDA rules. Code generation was performed through separation of the concerns, and use massive of the models, and model transformation. In this approach, MDA methodology has been focused only on the functional aspect and code generation only for this aspect without talked on security aspect like allowing code generation for gutter, setter, constructor, and same methods signatures. up until now, generated code hasn't been fully explored and there were other important further improvements which represent the determinant factor and success key. But they weren't integrated yet as generating code correspond to calculate methods, complex methods, and security infrastructures.

In this work, we have focused our entire concentration on non-functional aspect by automating software development process, and code generation. The new methodology of system design will be enhanced by security aspect integration during design phase and not after

Analysis and design phase. In previous works, security policies are negotiated in an ad-hoc manner after design phase by the system administrator which has a lack of knowledge on system architecture. This security integration technique gives a divergence between the design system and proposed security requirements, and increasing vulnerabilities. Thus, we have used MDA to improve software development process by injecting security policies into design system process to improve productivity gain and obtain secure applications which are easy to maintain, reusable, and scalability. In addition, our proposal has also many advantages including reducing implementation time, reducing design mistakes, and increases the quality of service.

As we know, a secure system should be used in a private manner and respects security policies proposed by a security expert during security requirements definition. Security criteria can be divided into five important groups: Confidentiality, Availability, Integrity, Traceability, and no-repudiation of the crucial data of information system (CAITN) standard. but in this work, we have focused only on the integration of Authorization, Authentication, Communication encryption, and Integrity without discussing others criteria. For that, we will enrich system specifications models with security requirements in order to automate security integration in the design system, and allowing secure application's generation which respect security policies already established. To do this integration, we have tried to transform security policies already proposed by the security expert to models which give an abstraction on security requirements.

Practically, we have used UML profile technology to adapt UML concepts with security policies and transform them into tagged values and stereotypes in order to enrich system models with proposed security policies or rather java models. Security policies determine the necessary preconditions and post-condition to achieve an operation; UML profile is a technology used to tag a design with information that is not captured in UML.

In our study, we'll generate secure applications from CD that is considered as PIM models obtained from CIM models by applying a set of model-to-model transition. To do so, a structural or intermediate model (IM) of java platform was generated from platform models (PM) and business domain system (PIM).this IM will be used to annotate PSM with others technical improvements; we have used java as the target platform to put into practice our proposed contribution. in this work, we have injected security policies into design system through enriching system models with security policies constraints by using this generated intermediate model that will extend to introduce further improvements. This enhancement is performed by applying UML profile together with OCL which is used to define informal specifications. For example, we can use OCL to make authorization policies constraints on the system operations and to manage resources access control. Consequently, the final code of the chosen platform will be generated automatically after applying a model to code transformation.

3.4 Models Overview and its Utility in Software Engineering

MDA gives more time to modeling rather than programming by using models at different phase of software engineering. Indeed, we have applied this philosophy to no-functional aspect by integrating the security policies and further improvements in the form of models at different abstraction levels of the system as shown in Figure3.this integration allows design engineer to annotate system models with security policies and allowing code generation by performing a set of model transformation model-to-model (M2M) and model-to-code (M2C).

In this work, the models can be used for the following four activities in the development process for developing secure applications.

- ✓ Documenting security requirements together with functionality system.
- ✓ Checking and analyzing security constraints.
- ✓ Model-based transformation, we can transform CIM models to PIM models by using the models.
- ✓ Generating final code, including complete, application security.

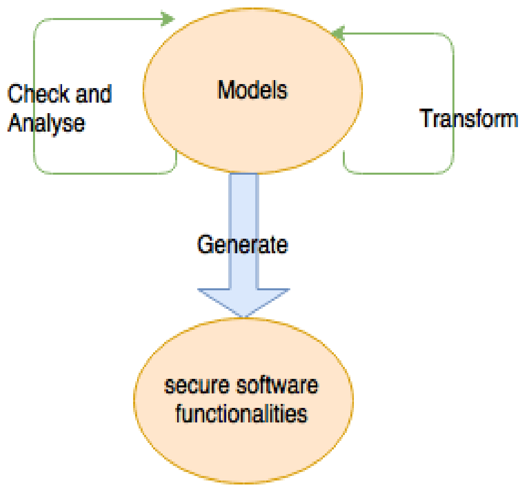


Figure 3: Use of models in MDA approach

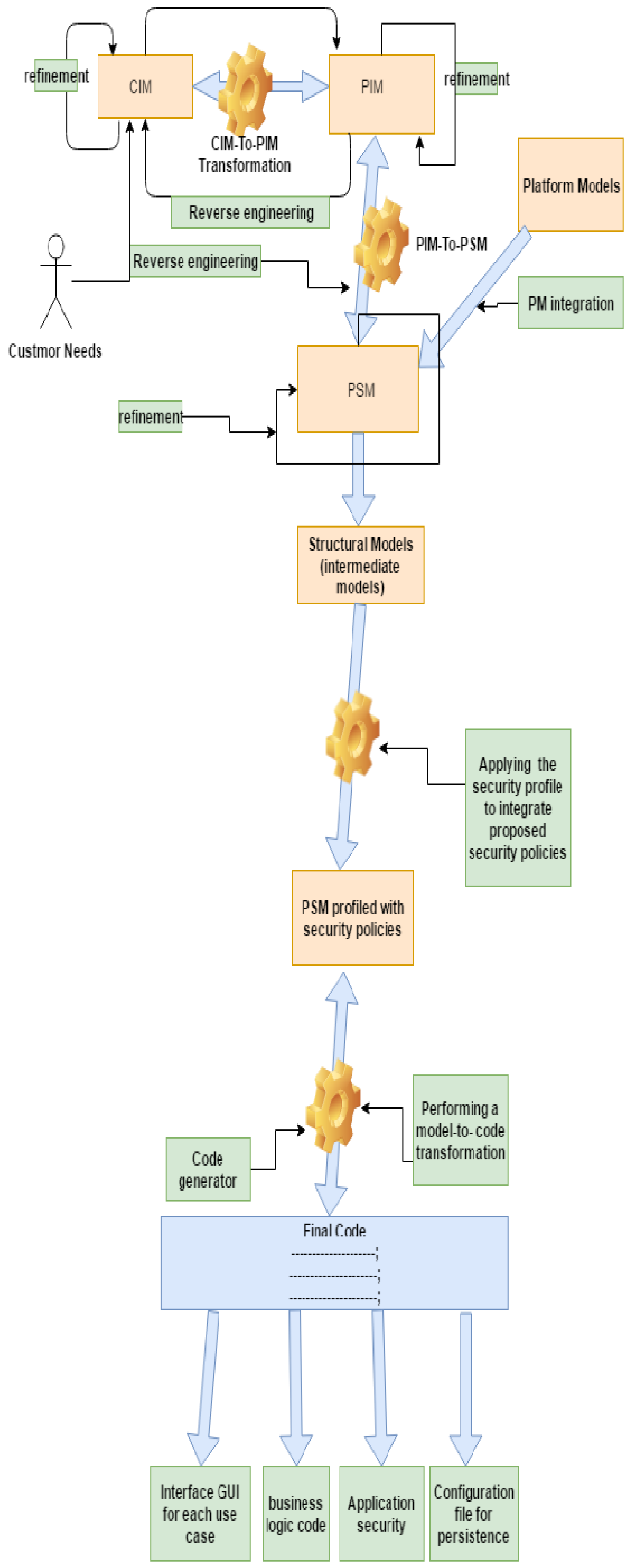

refinement
CIM
CIM-To-PIM Transformation
PIM
refinement
Reverse engineering
Reverse engineering
Platform Models
PIM-To-PSM
PM integration
Custmor Needs
PSM
refinement
Structural Models (intermediate models)
Applying the security profile to integrate proposed security policies
PSM profiled with security policies
Performing a model-to- code transformation
Code generator
Final Code
Interface GUI for each use case
business logic code
Application security
Configuration file for persistence

Figure 2: Architecture of the proposed approach

4. PROPOSED APPROACH

4.1 Proposed Approach Description

In this work, we have proposed a methodology for the automation of the entire proposed software development process by providing an efficient solution allowing security policies integration during software development process, and take them into account in the code generation by generating security infrastructures through a specific code generator.

To do so, we have proposed a model-driven code generator allowing code generation from models. In this case, our proposed code generator takes in input an intermediate model, which is profiled with the further improvements, then transform it towards a final code correspond to chosen platform. Moreover, we have proposed communication diagram meta-model (CDMM) that will be employed as a validator of communication diagram, and will be used as a semantic or modeling language to define CD structure, because it is used as a PIM obtained from CIM models. Otherwise, we have chosen CD as PIM, because it is the most relevant and appropriate diagram to draw complete and secure interaction between objects during execution of the operation. In addition, CD determines correctly the objects participant within the interaction and showing spatially the objects participant in the interactions.

Practically, by performing a model to model transformation, an intermediate model has been generated from chosen platform PSM and PIM. Generated IM will be used to improve java models with security policies which are already negotiated by security expert and design engineer at the beginning of the analysis and design phase CIM. To do so, IM will be extended to cover up the further improvements by using UML profile technology, and OCL [24] which is used to enrich systems models with the constraints which couldn't be formulated by UML models. In our work, we have used OCL to improve system models with security invariants such as data integrity, data encryption, and authorization policies based on RBAC. In addition, we have used OCL to extend precondition and post-condition of LARMAN operations contract (LOC) which is used to describe system's statue before and after operation execution. For that, we have extended the pre and post conditions to support security rules during interaction between objects, and resolve the shortcomings related to the assignment of the responsibilities to the objects.

By the new definition of LOC, design mistakes and security vulnerabilities will be resolved by presenting a new semantic of LOC called extended pre-condition and post-condition matrix. In this EPPM, we have introduced General responsibilities Assignment Software patterns GRASP PATTERN for automating assign responsibilities to the object responsible charge to achieve pre-condition and post-condition of an operation, and not manually. We have also enhanced this pre-condition and post-condition to take in account security integration during interaction between objects by providing a secure interaction basing on OCL. Then, operation body, signature, and object responsible will be deducted automatically through the tool EPPM with the code generator. For example, to create a new contact, staff manager should have a creation role, and must be authenticated the user, and must inform valid data, and date encryption as a password or sensitive data. For checking these constraints, we have applied EPPM tool by deducing the objects responsible and participant to achieve create contact operation.

Finally, by applying a model to code transformation, the final code of the business logic, and GUI, and security infrastructures will be generated automatically from IM by the way of the proposed code generator that takes in input a PSM models. Then transform them to code correspond to chosen platform.Figure4 below describe this approach in detail.

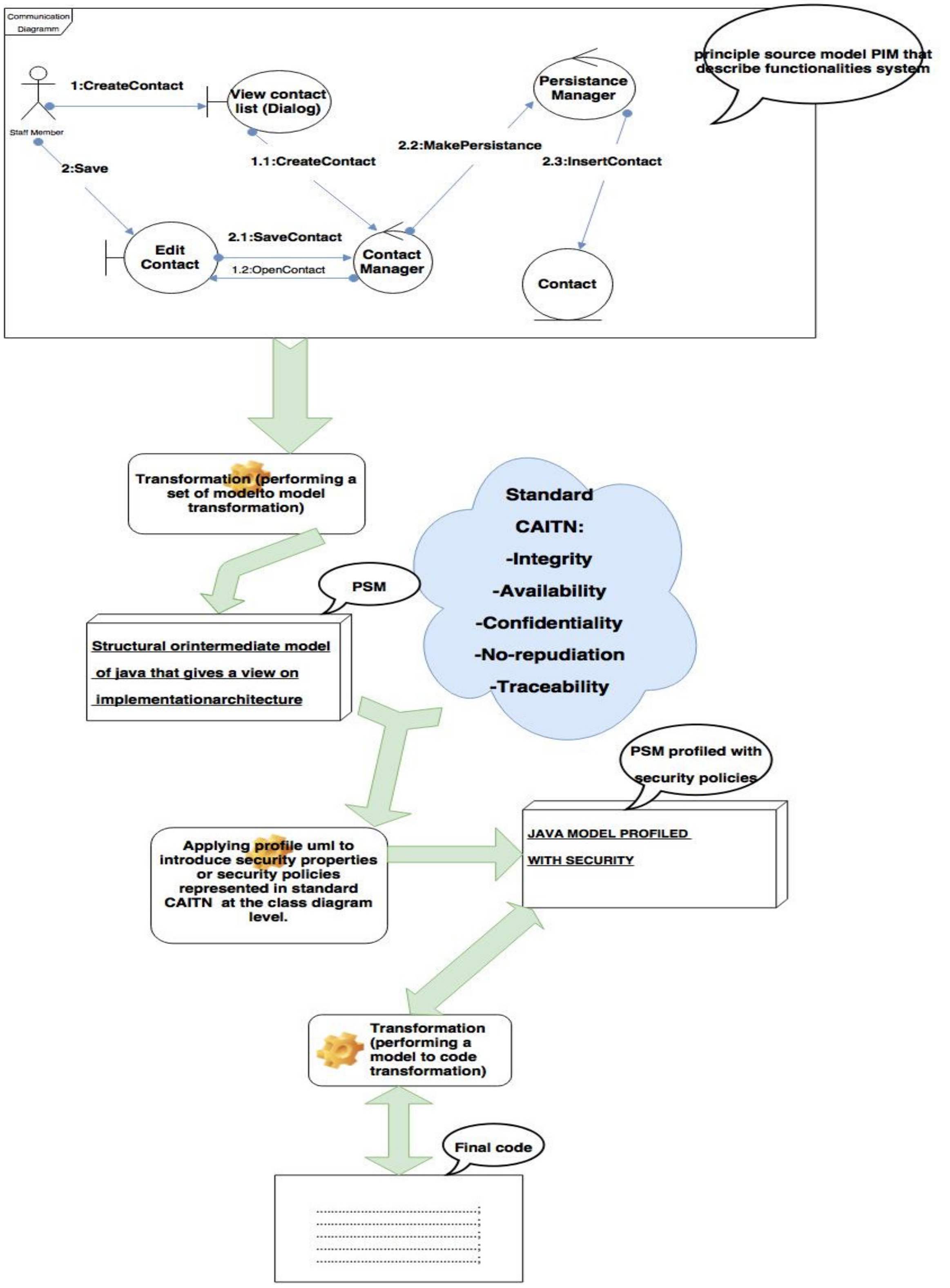

Communication Diagramm
Staff Member
1:CreateContact
View contact list (Dialog)
2:Save
1.1:CreateContact
2.2:MakePersistance
Persistance Manager
2.3:InsertContact
principle source model PIM that describe functionalities system
Edit Contact
2.1:SaveContact
1.2:OpenContact
Contact Manager
Contact
Transformation (performing a set of modelto model transformation)
Standard CAITN:
-Integrity
-Availability
-Confidentiality
-No-repudiation
-Traceability
PSM
Structural orintermediate model of java that gives a view on implementationarchitecture
PSM profiled with security policies
JAVA MODEL PROFILED WITH SECURITY
Applying profile uml to introduce security properties or security policies represented in standard CAITN at the class diagram level.
Transformation (performing a model to code transformation)
Final code

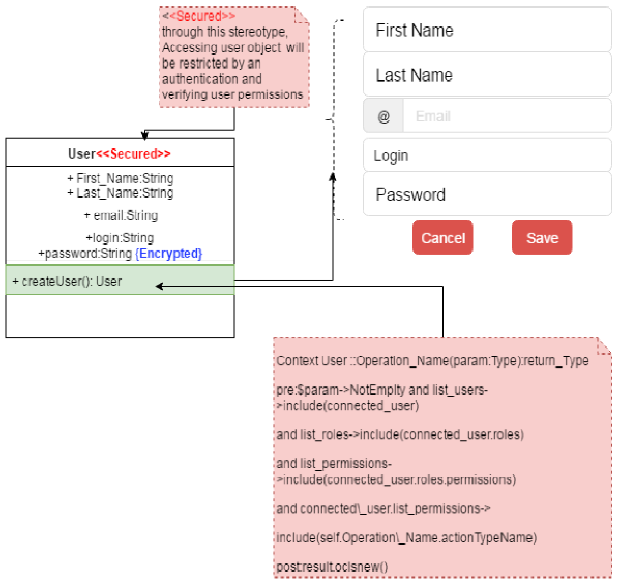
Figure 4: Different Transformation Performed To Generate Secure Application Starting From CD.

4.2 Enhancing system model with security policies and code generation

In this section, we propose a methodology to enrich structural model of the java model by security policies about data encryption, message integrity, authorization, and authentication concepts into design system phase by applying OCL and UML Profile. According to the syntax given below, software engineers can enrich their system models or system functionalities models with security constraints or authorization constraints such as data encryption, Access control based on RBAC extend with constraints, assignment of write/read privileges, data validation, Message Integrity, and all informal specifications which can't be formulated with UML models. Consequently, the code source according to chosen platform will be generated automatically from input models by respecting the quality of services such as Scalability, reusability, reliability, and data security by the means of proposed code generator.

In another word, the Structural model has been enriched by the security constraints basing on security profile that is expressed according to UML and OCL so that the system objects and resources will be able to interact securely.

The figure 5 below shows a UML model allowing enhanced with required security policies. This figure describes also the future work that will be focused on an approach allowing generating GUI for each use case basing on the same code generator.



```
Context User ::Operation_Name(param:Type):return_Type
pre:$param->NotEmplty and list_users-
>include(connected_user)
and list_roles->include(connected_user.roles)
and list_permissions-
>include(connected_user.roles.permissions)
and connected\_user.list_permissions->
include(self.Operation\_Name.actionTypeName)
post:result.oclsnew()
```

Figure 5: Example of applying the security constraints on systems operations and generating GUI.

4.3 Applying EPPM Tool

In this section, we show an example of creating the new user by describing the system's statue before and after achievement of the use case "CreateUser" by using OCL for verifying security constraint and permissions which are granted to protected resources, with profile UML to apply security properties on class diagram during system design phase. We have also addressed code generation.

4.3.1 Pre-condition and its OCL expression

In the first place, the staff Manager should have a creation role and all privileges which allow him to create a new user and must be authenticated and authorized before invoking off the operation concerned. The figure 7 bellow gives a syntax allowing enhancing the system operations with the security constraints about authentication and permissions constraints.

```
context user:: createUser():User
pre: list_user->include(connected_User)
 and rules->include(connectedUser.role)
 and
self.createUser.actionTypePermission>include
(connectedUser.role.permission)
```

Figure7: OCL expression allowing enhancing method with security policies constraints

4.3.2 Verifying data integrity

In this section, we give syntax OCL allowing enriching data models UML or more precisely java models including java methods, fields, interface, and classes with the confidentiality and data integrity constraint. For example, accessing to an operation should be restricted by permissions and constraints which are formulated by fallowing the below syntax. By the way, this syntax is applicable to all java models.

```
context user:: createUser():User
pre: self.firstName->notEmpty
and self.latsName->notEmpty
 and self.login->notEmpty and
self.password->notEmpty
```

Figure8: OCL expression allowing enhancing method with confidentiality and data integrity constraints

4.3.3 Data encryption and its OCL expression

Data encryption becomes an obligation for some sensible information like password, the balance of accounts. Therefore, we have incorporated the data encryption during system design to indicate that an information or object should be encrypted during the communication between system objects, or between system and external components. The figure 9 describes a OCL syntax allowing to indicate that an element of models is encrypted.

```
context user inv:pre: self.password.format->
include(user.password.applied_encryption_Type)
```

Figure9: OCL expression allowing enhancing data models with encryption constraints

4.3.4 Post-condition description

In the post-condition, we describe system's statue after the execution of an operation (createUser) by taking into account the new tool EPPM to assign automatically responsibilities to the specific objects in charge to complete the execution of the operation, and verifying data integrity, and verifying identity of objects participate in the achievement of the operation before finishing the execution of the operation as shown in the following example. Security Manager will be an XML file generated automatically from system models or than PSM (Platform Specific Model) after their enhancement with requested security properties.

1) The authenticated user sends a request from (layer dialog) to the controller to create a new user.
2) The controller sends a message to security validator or security Manager which is employed for verifying data integrity (Data integrity and confidentiality), and the identity of the user (Authentication and authorization) basing on GRASP PATTERNS low coupling and high cohesion.
3) The controller sends a message to responsible class to create an instance from concerned subject class. By applying GRASP PATTERN Creator, controller, and Expert.
4) The responsible class sends a message to subject class (user) to call its constructor which has the same name as its subject class to create a new instance based on proposed security rules.
5) Finally, the responsible class sends a message to Data Access Object (DAO) in order to persist created object into the database. We have applied GRASP PATTERNS pure fabrication and Expert.

In this example, there are five main objects: connected user, Controller, Security Manager, Subject class, and DAO.

4.3.5 Operation signature and its body

After applying the new tool EPPM and the security properties at Communication diagram elaboration, we are getting a complete PSM from PIM which is enhanced with new features. Then by applying a specific code generator, the final code will be generated automatically from PSM. Figure 10 gives an example of code generated from User class.

```
Public class User{
//gutters and setters
//create user method
Public user  createUser(){
return new User();
}}
```

Figure10: User Class code

Consequently, we have applied this methodology to the java platform by enriching the structure model of the java platform with the proposed security policies such as privacy policies based on RBAC, data integrity, data encryption, and secure communication basing on OCL and Profile UML. By the means of this enhancement of the structural model, the system models PSMS will be refined by security constraints which have been performed through security profile so that to add all constraints which cannot be formulated in system models by using only UML meta-model.

4.4 Enhancing System Models with Security Policies at Class Diagram Level

In this section, we propose a syntax allowing improving the structural model of the java model by security policies into design system phase by applying OCL and UML Profile. According to the syntax given below, software engineers can enrich their system models with security constraints such as data encryption, Access control based on RBAC extend with constraints, assignment of write/read privileges, data validation, and all informal specifications which can't be formulated with UML models. Consequently, the code source according to chosen platform will be generated automatically from input models PSM by respecting the proposed quality of services such as Scalability, reusability, reliability, and data security.

In other word, Structural model has been enriched by the security constraints basing on security profile which is expressed according to

UML, and OCL. As a result, system objects and resources will be able to interact securely.

Figure 11 describes a Syntax allowing enhancing system operation by security constraints so that to restrict unauthorized access. This improvement was done at UML profile level.

```
Context X :: Operation_Name(param:Type):return_Type
pre: $param->NotEmplty
and list_users->include(connected_user)
and list_roles->include(connected_user.roles)
and list_permissions->
include(connected_user.roles.permissions)
and connected_user.list_permissions->
include(self.Operation_Name.actionTypeName)
post:  result.oclsnew()
```

Figure11: Syntax allowing restricting unauthorized access to system operation

4.5 Security Profile Description

In this section, we proposed a security profile that includes the security policies or security requirements which are obtained from the security expert. By means of this profile, we have the ability to integrate easy all shortcomings about security concerns so that for keeping a good data consistency check. The figure 12 below shows the proposed security profile that has been used to integrate the security properties whose we have talked in the previous section. In this work, we have concentrated only on authorization, authentication, Message integrity, and data encryption security policies.

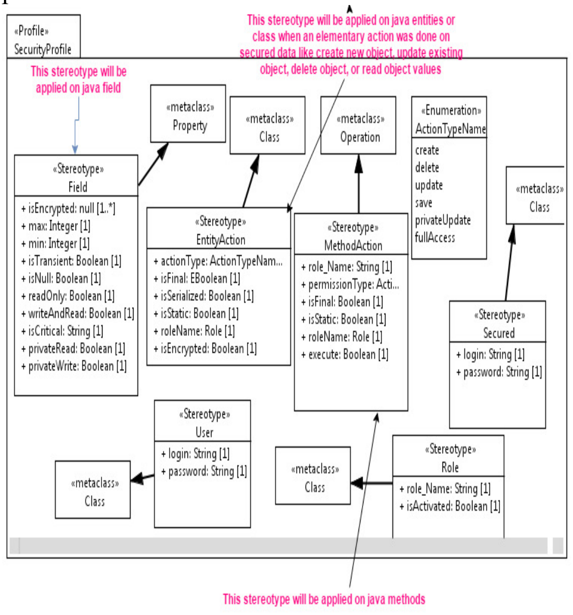

«Profile» SecurityProfile
This stereotype will be applied on java field
This stereotype will be applied on java entities or class when an elementary action was done on secured data like create new object, update existing object, delete object, or read object values
«metaclass» Property
«metaclass» Class
«metaclass» Operation
«Enumeration» ActionTypeName
create
delete
update
save
privateUpdate
fullAccess
«metaclass» Class
«Stereotype» Field
+ isEncrypted: null [1..*]
+ max: Integer [1]
+ min: Integer [1]
+ isTransient: Boolean [1]
+ isNull: Boolean [1]
+ readOnly: Boolean [1]
+ writeAndRead: Boolean [1]
+ isCritical: String [1]
+ privateRead: Boolean [1]
+ privateWrite: Boolean [1]
«Stereotype» EntityAction
+ actionType: ActionTypeNam...
+ isFinal: EBoolean [1]
+ isSerialized: Boolean [1]
+ isStatic: Boolean [1]
+ roleName: Role [1]
+ isEncrypted: Boolean [1]
«Stereotype» MethodAction
+ role_Name: String [1]
+ permissionType: Acti...
+ isFinal: Boolean [1]
+ isStatic: Boolean [1]
+ roleName: Role [1]
+ execute: Boolean [1]
«Stereotype» Secured
+ login: String [1]
+ password: String [1]
«Stereotype» User
+ login: String [1]
+ password: String [1]
«metaclass» Class
«metaclass» Class
«Stereotype» Role
+ role_Name: String [1]
+ isActivated: Boolean [1]
This stereotype will be applied on java methods

Figure 12: Proposed security profile

4.5.1 Applying security profile on class diagram

After applying security profile on the structure java model, deduced models will be profiled automatically according to the security requirements by following the new software engineering process. Figure 13 show a UML class enhanced with security policies by the means of the proposed profile.

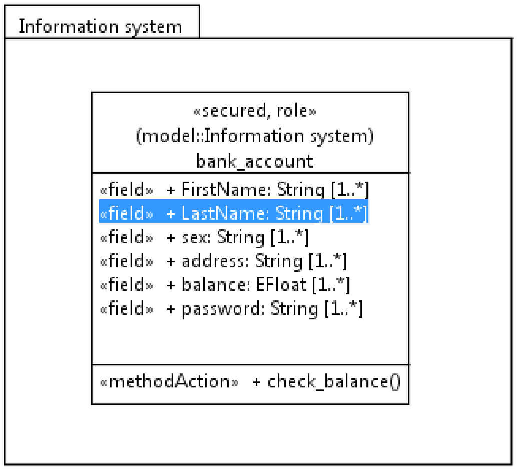

Information system
«secured, role»
(model::Information system)
bank_account
«field» + FirstName: String [1..*]
«field» + LastName: String [1..*]
«field» + sex: String [1..*]
«field» + address: String [1..*]
«field» + balance: EFloat [1..*]
«field» + password: String [1..*]
«methodAction» + check_balance()

Figure13: A UML class profiled by security profile

5. TOOLS AND TECHNICALS

Practically, for allowing to software engineers getting secure application from profiled models (extended PSM with no-functional aspect) automatically. It's very important to deals with this technical's improvements by providing a specific code generator dedicated to this mission or update the existing code generators for supporting these changes which were performed at the system design process level. Therefore, the final code corresponding to functional and no-functional models will be generated automatically from secured models which are designed basing on MDA approach.

For that, we have performed some comparison study stuck between different well-known existing code generators by recounting the strong and weak point for each code generator. This comparison is based on criteria of the generated code technical's details, and the structure of the code generator on another side. Although there are several code generators, they are still now incapable of generating secure applications and immature to take into account the both aspects: functional and no-functional at the same time for generating the code complete of an application.

For example, the first tool is startigo/XT which is frequently deployed as a creator of code generators. In spite of the diverging structure, this generator stays incapable of producing secure applications which incorporate at the same time functional and no-functional aspects. In addition, startigo/XT hasn't a concrete syntax and doesn't cover up the complete development process of software. So it remains incapable of supporting the object-oriented and complex system. Therefore, its syntax need to some modifications allowing secure application generation.

In the second kind, we found the classical code generators which are based on Petri nets theory. Really, this type was specifically intended for requirements validation because they have not standardized yet to support the entire development process, and not yet adapted for supporting the complex system object-oriented system. In addition, those generators don't have a concrete syntax and can't allow secure application generation.

The third code generator is [27], this category has been anticipated to determine the shortcomings resulting from startigo/XT and Petri nets solutions. This category provides a code generator that supports the entire development process of software, and support object-oriented system. In addition, it takes into account the structural model. But it doesn't generate CRUD operations, and GUI interfaces, and controllers, and security infrastructure files. So this kind also needs to some modification so that it can deploy as a code generator.

The fourth code generator is WebML which is designed for the web application like WebDSL. This generator provides many advantages such as generation of CRUD operation, method bodies, and GUI interfaces. In addition, it uses a concrete syntax. But at the same, it has a few weaknesses like:

- ✓ WebML doesn't integrate the structural model which is essential elements of this approach.
- ✓ WebML doesn't support the entire development process.
- ✓ WebML doesn't generate applications controller.
- ✓ WebML doesn't generate security infrastructure files like security Manager.

Finally, the fifth code generator is Acceleo which is more similar to WebML. But, the difference that Acceleo supports the entire development process while WebML not doing it. Additionally, Acceleo stays still immature to generate secure applications supporting code generation of the security infrastructure files.

At the beginning, we have used Acceleo as a code generator to start the unit testing of the suggestion. While in the same time, we have built in parallel our code generator for automating the key phases of software development by covering the entire development process from specifications phase (CIM) to implementation phase (PSM) as shown in the figure 2 by allowing code generation of the security Manager file, GUI interfaces, Controller, operations bodies. This generator will be available in the next paper. In addition, our proposed code generator is going to provide many advantages:

- ✓ The proposal supports the structural model.
- ✓ The proposal covers the entire development process and object-oriented system.
- ✓ The proposal allows applications controller generation.
- ✓ The proposal generates security infrastructure files, and method signatures and their bodies.
- ✓ The proposal generates security infrastructure files and CRUD operations.

6. RESULTS

As results, we have proposed a new model-driven methodology allowing automating the entire software development process which is beginning from specification requirements phase and finishing by implementation. In this methodology, we have applied a radical change on a MDA approach by adding the new iterations to MDA process like the enhancements of the PSM with authorizations policies, generating secure application, and taking into account models refinements to as well separate functional and no-functional models. To put into practice the new methodology and security integration during design system phase, we have proposed a new specific tool called Extended Pre-conditions Post-conditions Matrix (EPPM), which is based on LARMAN operations contract, security profile, and OCL.

In this tool, we have extended the old LARMAN operations contract to a new version more detailed description than its original version to support securities policies integration during software design phase, and more precisely during interaction between systems objects. To do so, the generated structural model of the java platform has been enriched by applying the security profile in order to apply the securities policies rules which are defined in four essential elements: authorization; authentication; data encryption; and data integrity. So by means of this extension, specifications models will be improved by security policies which can be formulated in the form of the models via UML profile. The security models will be merged with specifications models through the model's transformation at different abstractions of the MDA. In other words, from now on any access to an instance of Java Class, class Methods, relations, and java Fields will be controlled by security policies which are already addressed by the security expert and transformed in the form of the security profile.

By applying security profile on the generated structural model, access to java class becomes restricted by login, password, and privileges. Therefore, access the instance of a class includes a previously authentication and authorization. Thereby a user must have prior access permission with required roles that allow him to interact with these instances of a class and its content. On the other hand, methods and attributes of a class will be restricted by access controls, including assignment of reading/writing privileges, data encryption, and data validation by applying security profile with OCL for verifying security policies constraints, and verifying the identity of a user. By the means of this methodology, the final code will be generated with its security infrastructures including security controller and security manager.

In this work, we have also proposed a communication diagram meta-model allowing describing the communication diagram structure which will be used as a PIM obtained from CIM. This PIM has been transformed into PSM by performing a set of model-to-model transformation. Then, the structural model of the java platform has been generated from PSM after their enhancement with the security needs basing on customers need.

Finally, by applying a model-to-code transformation, the code source according to the JAVA platform will be obtained from PSM which has already improved with security policies. This generation includes also applications security files like security manager and the security controller, which contains all authorization policies described in the form of the XML file

7. CONCLUSIONS AND FUTURE WORK

In this work, we introduced the Model Driven Engineering (MDE) and its implementation Model Driven Architecture (MDA). We have also concentrated on the application security integration during MDA process at the same time by proposing a model-driven oriented-object methodology that combine the functional aspects and no-functional aspects at the same time to get a tailored solution which cover all aspects basing on the potential offered by MDA approach, and its utility in relation to the productivity of the models, sustainability, and platform description integration trough the model transformations.

Within this context, we have contributed by proposing an enhanced MDA approach for generating secure applications from communication diagram which is described by its meta-model, which gives the structure or semantic of the communication diagram (communication diagram meta-model). So the software development process which is based on traditional software engineering methods becomes more automatic by automating software development process basing on second code generation which is based on model transition: model-to-model, and model-to-code transformations, and MDA approach. By means of this transformation, the security concerns will be incorporated into models. In other words, the core idea behind automating software process is allowing code generation for chosen platform from communication diagram by taking into account the security policies requirements integration into the software design phase and automating code generation corresponding to application security, more precisely during interaction between system objects. We have also obtained PIM (communication diagram) from CIM by applying model transformation mechanisms and a set of models refinements. In addition, getting PIM has been performed by using design patterns GRASP PATTERNS and the new semantic of LARMAN operations contract (EPPM) in which we integrated security constraints basing on OCL pre and post-conditions for checking authorization policies, and data integrity, and data encryption during interaction between system objects.

The main objective of this integration is enriching the structural model of the chosen platform with the security policies proposed by the security expert in order to obtain a secure application from models by means of the code generator and the new approach of MDA. Therefore this methodology allows generating the complete methods signatures and their body. all the generated methods will be generated with its security properties and the permissions needed to do the method. It allow also obtaining security controller that is used to verify the authorization policies like authentication and authorization of the users before accessing to a method or resources. So this approach is very useful in the case of an object-oriented and complex system.

In addition, the future version of code generator will be enhanced to include additional information on code generation of the embedded system, improving existing code generator to support GUI interfaces generation for different uses cases according to chosen technology, and finishing the proposed code generator to take into account all these details within the system design and code generation phase.

REFRENCES:

[1] OMG, « Object Constraint Language (OCL) Specification, version 2.0 », 2006. http ://www.omg.org/spec/OCL/2.0/.

[2] OMG: MDA GUIDE, Version 1.0.1 Object Management Group document number omg/2003-06-01 available at http://www.omg.org/docs/omg/03-06-01.pdf.

[3] OMG: Object Management Group. www.omg.org http://www.omg.org/docs/omg/03-06-01.pdf.

[4] C. Girault, R. Valk, Petri-nets for systems engineering,Springer, 2003, berlin.

[5] S.S. Huang, Y. Smaragdakis, "Easy language extension with Meta- AspectJ," In ICSE 06 Proceeding of the 28th International Conference on Software Engineering, ACM, New York 2006 , pp. 865-868.

[6] D. Zook, S.S Huang, Y. Smaragdakis, "Generating AspectJ Programs with Meta-AspectJ, " In Generative Programming and Component Engineering Conference, GPCE 2004, Vancouver, Canada, October, 2004, vol. 3286, pp. 1-18.

[7] Code generation through model transformation http://alexandria.tue.nl/extra2/afstversl/wsk-i/verstraeten2008.pdf.

[8] B. Bouseta, O. El Beggar, T. Gadi, " Generating operations specifications from domain class diagram using transition state diagram, " international journal of computer and information technology, January, 2013, vol. 02 , pp. 29-36.

[9] O. El Beggar,B. Bouseta, T. Gadi Taoufiq, " automatic code generation by model transformation from sequence diagram of system's internal behavior," international journal of computer and information technology, November, 2013, vol. 02 , pp. 129-146.

[10] Z. Hemel, L.C.L Kats, E. Visser, "Code Generation by Model Transformation A Case Study in Transformation Modularity, Chapter Theory and Practice of Model Transformations," series Lecture Notes in Computer Science, vol. 5063, pp 183-198.

[11] A. Manoli, J. Cabot, C. Gómez , V. Pelechano , "Generating operation specifications from UML class diagrams: A model transformation approach, " Data & Knowledge Engineering , April, 2011, vol. 70, pp. 365-389.

[12] E.B Fernardez, M.M Larondo-Petrie, T.Sorgente, M.Vanhilst, a Methodology to develop secure systems using patterns, 2008.

[13] G.-J. Ahn, M. E. Shin, "UML-based representation of role-based access control," In Proceedings of the 9th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'00), IEEE Computer Society, Jun, 2000, pp. 195-200.

[14] D. Basin, J. Doser, T. Lodderstedt, "Model driven security for process-oriented systems," In Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT '03), ACM Digital Library, Jun, 2OO3, pp.100-109.

[15] J. Jürjens, "UMLsec: Extending UML for secure systems development," In Proceedings of the 5th International Conference on the Unifed Modeling Language (UML'02), LNCS, October, 2002, vol. 2460, pp. 412-425.

[16] D. Basin, J. Doser, T. Lodderstedt, "Model driven security for process-oriented systems," In Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT '03), ACM Press , Jun, 2003, pp. 100-109.

[17] X. Jin, Applying model driven architecture approach to model role based access control system (Doctoral dissertation, University of Ottawa).

[18] D. Basin, J. Doser, T. Lodderstedt, "SecureUML: A UML-Based Modeling Language for Model-Driven Security," In Proceedings of the 5th International Conference on the Unifed Modeling Language (UML'02), LNCS, October, 2002, vol. 2460, pp. 426-441.

[19] D. Basin, J. Doser, T. Lodderstedt, "Model driven security: From UML models to access control infrastructures," ACM Transactions on Software Engineering and Methodology (TOSEM), Jun, 2006, vol. 15, pp. 39-91.

[20] E. Fernandez--Medina, J. Trujillo, R. Villarroel, M. Piattini, "Developing secure data warehouses with a UML extension," Information Systems, September, 2007, vol. 32, pp. 826-856.

[21] J. Reznik, T. Ritter, "Model Driven Development of Security Aspects," In Proceedings of the Second International Workshop on Aspect-Based and Model-Based Separation of Concerns in Software Systems (ABMB 2006), Electronic Notes in Theoretical Computer Science , April, 2007, vol. 163, pp. 65-79.

[22] J. Trujillo, E. Soler, E. Fernández-Medina, M. Piattini, "An engineering process for developing Secure Data Warehouses,"Information and Software Technology, Jun, 2009, vol. 51, pp. 1033-1051,.

[23] C. Blanco, I. García-Rodríguez de Guzmán, E. Fernández-Medina, J. Trujillo,M. Piattini, "Applying an MDA-Based Approach to Consider Security Rules in the Development of Secure DWs," IEEE Xplore digital library , Jun, 2009, vol. 51, pp. 1-25.

[24] J. Miller, J. Mukerji, MDA Guide Version 1.0.1. Technical report, Object Management Group (OMG), 2003.

[25] F. Allilaire , J. Bézivin , F. Jouault , I. Kurtev, ATL–Eclipse Support for Model Transformation (2006) : Proc. of the Eclipse Technology eXchange Workshop (eTX) at ECOOP.

[26] Object Management Group, Inc. Meta Object Facility (MOF) 2.0 Core Specification, Final Adopted Specification, January 2006.

[27] S. Philippi, "Automatic code generation from high-level Petri-Nets for model driven systems engineering," The Journal of Systems and Software, 2006, vol. 79, pp. 1444-1455.