Secrecy UML Method for Model Transformations*

Waël Hassan[1], Nadera Slimani[2], Kamel Adi[2], and Luigi Logrippo[2]

[1] University of Ottawa, 4051D-800 King Edward, Ottawa, Ontario, K1N-6N5
[2] Universit du Qubec en Outaouais, 101 Rue St-Jean-Bosco, Gatineau, Qubec, J8X 3X7
wael@acm.org, {slin02, Kamel.Adi, luigi}@uqo.ca

Abstract. This paper introduces the subject of secrecy models development by transformation, with formal validation. In an enterprise, constructing a secrecy model is a participatory exercise involving policy makers and implementers. Policy makers iteratively provide business governance requirements, while policy implementers formulate rules of access in computer-executable terms. The process is error prone and may lead to undesirable situations thus threatening the security of the enterprise. At each iteration, a security officer (SO) needs to guarantee business continuity by ensuring property preservation; as well, the SO needs to check for potential threats due to policy changes. This paper proposes a method that is meant to address both aspects. The goal is to allow not only the formal analysis of the results of transformations, but also the formal proof that transformations are property preserving. UML is used for expressing and transforming models [1], and the Alloy analyzer is used to perform integrity checks [6].

Keywords: Model transformation, Property preservation, SUM, Alloy, UML.

1 Introduction

Governance requirements dictate a security policy that regulates access to information. This policy is implemented by means of secrecy models[3] that establish the mandatory secrecy rules for the enterprise. For example, a secrecy rule may state: *higher-ranking officers have read rights to information at lower ranks.* In addition, *Business policies* may specify instances such as: *user A has access to department M.* Hence, an enterprise governance system is composed of a combination of *secrecy model rules* and *business policies.*

Automation helps reduce design errors of combined and complex secrecy models [3]. However, current industry practices do not include precise methods for constructing and validating enterprise governance models. Our research

* This work has been funded in part from grants of the Natural Sciences and Engineering Research Council of Canada and CA Labs.

[3] We concentrate in this paper on secrecy, which is one of several aspects of security.

proposes a formal transformation method to construct secrecy models by way of applying transformations to a base UML model. It provides the advantage of formal validation of constructed models. For example, starting from an initial model called Base Model (BM), with only three primitives: Subject/Verb/Object, we can generate –by transformation functions– RBAC0 (Role Based Access Control model) in addition to a SecureUML model.

By way of examples we intend to show that our method is potentially useful for building different types of secrecy models. By means of formal analysis we intend to show that a SO will be able to validate a resultant model for consistency in addition to detecting scenarios resulting from unpreserved properties.

In this paper, we present our method in section 2. In section 3, we show examples that illustrate our approach with application results. Finally, we conclude this paper, in section 4, and discuss the future work and perspectives.

2 Secrecy UML Method (SUM)

SUM serves as a systematic method to construct secrecy properties for enterprise governance. Starting from a generic UML model, that we call *base model* (BM) and a set of transforming operations (TOs) (see Fig. 1). The operations are conjectured to be *property-enriching* as well as *property-preserving* and are applied to achieve a *resultant model* (RM). In Fig. 1, several rectangles labelled: Specialize, Aggregate, Compose, Split Right, Split Left, Reflex, Tree Macro, and Graph Macro. These rectangles represent the transformation operations. Each of the transformation operations takes as input a class labelled 'input' –shown using a grey shading– and produces the respective output –as shown. The TOs modify the base model iteratively in a way that, in practice, the resultant model is customized and used to govern security or privacy properties of a particular enterprise.

We use a first order logic formalism to: represent the base model, the transformation operations, and the resultant model. We show that it is possible to validate the resultant model for consistency using logic analysers. The translation from the UML language to a logic analyser language can be done through the use of a specialised secrecy modelling language called SML [4]. SML provides a component view of Alloy code that is conjectured to facilitate the representation of secrecy models. Alternatively it can be done directly through a UML to Alloy translator [6]. In all cases a transformation tool can be programmed to map a model to another. Alloy will validate the models for properties of model preservation and consistency.

2.1 Base-model (BM).

The base model proposed in this paper includes three primitives components: S, V and O. A subject S is a subject in the enterprise. A verb V denotes the fact that an action or right is given or denied to the subject. An object O is the data item or object to which the action or right refers.

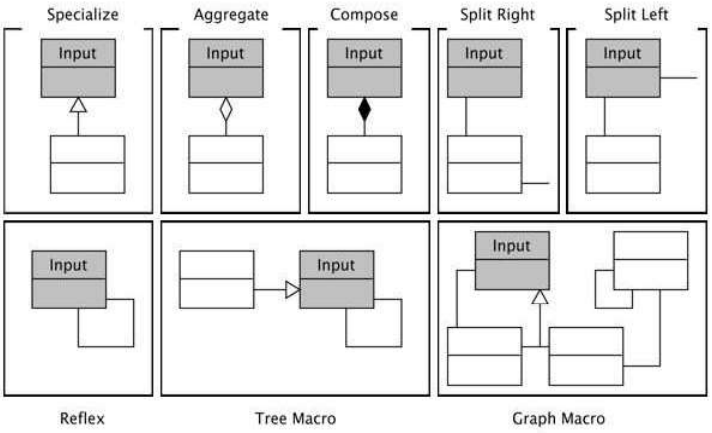

Fig. 1. Transformation operations.

2.2 Transformation Operation (TO).

A TO is an operation consuming an input and producing an output model. Our method defines the following TOs: Specialisation, Aggregation and Composition, Reflex, Split, Tree Macro, Graph Macro (See Fig. 1).

Specialisation: Following the UML definition [5] this operation extends a general class into a specific one with detailed features.

Aggregation and Composition: Aggregation and Composition describe the construction of a parent class from sub-classes, that are mandatory (sub-classes) in the case of Composition. Example of Composition: *An audit department is composed of financial and privacy audit sub-departments.* Both departments (privacy and financial audits) are necessary for the audit department to exist. On the other hand, *the set of employees consists of full-timers, part-timers, and consultants* is considered an Aggregation.

Reflex operation: A reflex transformation adds a relation to the input class. This operation is frequently used, mostly to represent a structural relation. e.g. *a node is a sibling of another node.*

Split operation: A split is often used to transform a component into a relation between two components. For example, an object can be split *into a session controlling an object.* A split can preserve all or some of the original relations of the input component. In Fig. 1, we show two kinds of split left, right, which we will detail in future work.

Tree Macro: The tree macro is useful for the construction of several secrecy models. For instance, *it can be used to represent a relation between a subject and its department or Group.*

Graph Macro: A graph macro takes an input class and creates a graph of classes. For example, *it can be used for building business processes.*

3 Examples

3.1 Transforming BM to RBAC

In this first example of transformation, we apply a set of operations so that the resultant model is similar to RBAC (Role Based Access Control model) in [2]. Fig. 2 shows the syntax representation of the RM components at each iteration.

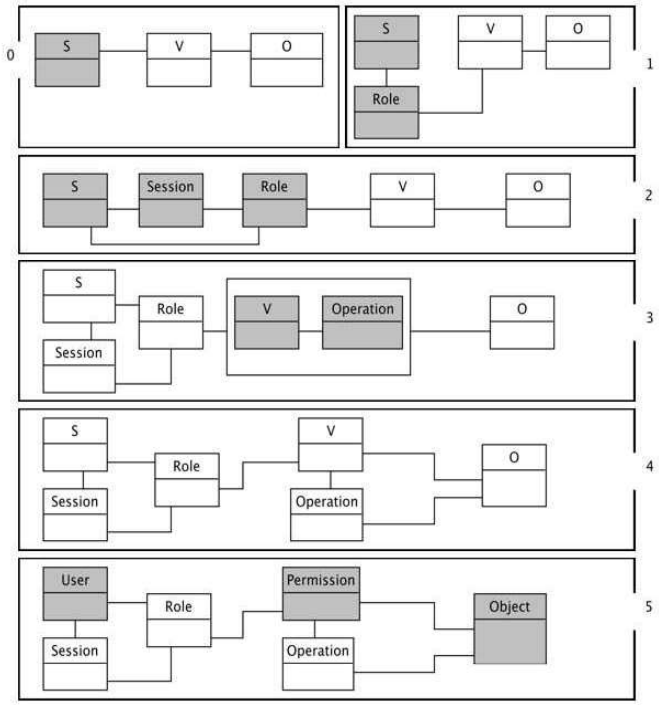

Fig. 2. Transformation steps from BM to RBAC model.

In this case, we simply apply three Split operations on both primitive components: Subject and Verb, followed by the Renaming operation, e.g. Verb becomes

Operation. Here is a list of the used successive operations, so that in the left side (the function result) we have the set of components forming the new model:

- Split(S)={S, Role}
- Split(S)={S, Session}
- Split(V)={Verb, Operation}
- Rename(S, V)={User, Permission}

3.2 Transforming BM to SecureUML

SecureUML is a security modeling language based on RBAC with refinement [3]. Fig. 3, shows the transformations needed to develop the meta-model of SecureUML.

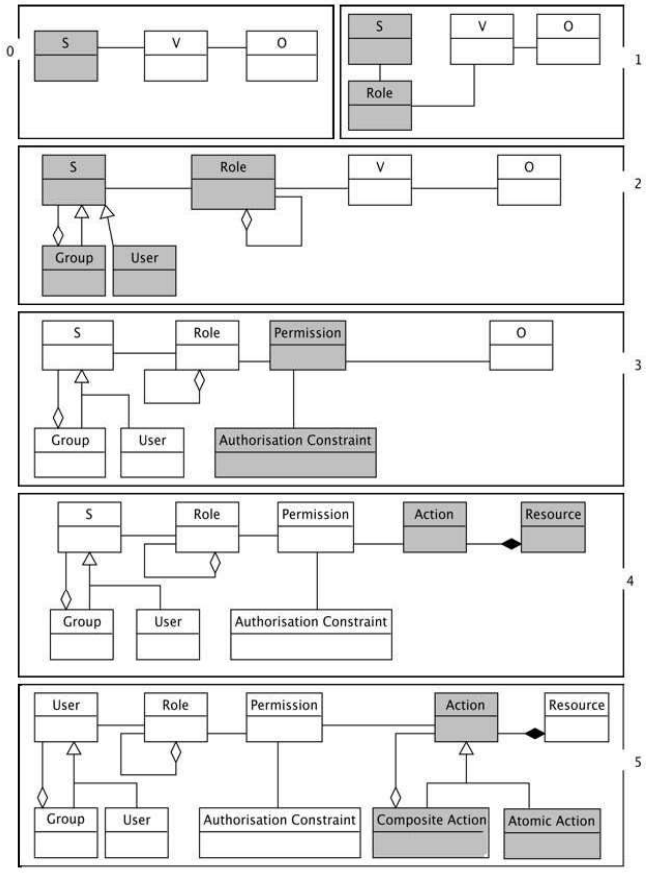

Fig. 3. Transformation steps from BM to SecureUML model.

The SecureUML is defined as an extension of RBAC. It supports: the policy constraint (using the Authorisation Constraint component), the Action hierarchy, the Specialisation of the Action in Atomic Action and Composite Action, etc. Since, SecureUML is more complicated. It requires using more than two types of TO, in the following operation list:

- Split(S)={S,Role}
- Specialise(S)={Group,User}
- Aggregate(Group)={S}
- Aggregate(Role)={Role}
- Split(V)={V, AuthorisationConstraint}
- Rename(V)={Permission}
- Compose(Resource)={Action}
- Specialise(Action)={AtomicAction, CompositeAction}
- Aggregate(AtomicAction)={Action}

4 Conclusion

In conclusion, the Secrecy UML method (SUM) supports the construction of complex secrecy models from a base-model by a disciplined transformation method. We believe that its application would be to assist a security officer in achieving the required enterprise security policy by model transformation. We will show in future publications that our technique, combining the use of UML and relational logic, allows verification of the final result using the Alloy analyser. Future work will strengthen this conjecture by proving the property-preserving characteristics of the transformations. There are several avenues for future work in this domain. We plan (i) to provide a detailed formal description of the transformation operations on a case study; (ii) to extend this paper to include the SML statements corresponding to the output model in each case; (iii) to show the ability to detect inconsistencies in the design. Finally, we foresee to create a graphical user interface module that allows a designer to transform a UML model, using SUM operations, and to create an automatic SUM to SML translator.

References

1. Berardi, D., Calvanese, D.,De Giacomo, G.: Reasoning on UML class diagrams. Artif. Intell. Essex, UK , 2005. 70-118, Elsevier Science Publ. Ltd.
2. Ray, I., Li, N., France, R., Kim, D.: Using UML to visualize role-based access control constraints. SACMAT'04. NY, USA, 2004. 115-124, ACM.
3. Basin, D., Doser, J. and Lodderstedt, T.: Model driven security: From UML models to access control infrastructures. Softw. Eng. Methodol. NY, USA, 2006. 39-91, ACM Press.
4. Hassan, W.: Secrecy Modelling Language. http://code.google.com/p/sml-silver/. Accessed Aug 2009.
5. Evans, A., France, R. B., Lano, K. and Rumpe, B.: The UML as a Formal modelling Notation. UML'98: London, UK, 336-348, 1999. Springer-Verlag.
6. Anastasakis, K., Bordbar, B., Georg, G., Ray I.: On Challenges of Model Transformation from UML to Alloy. MoDELS 2007. 436-450.