B for Modeling Secure Information Systems
The B4MSecure platform

Akram Idani and Yves Ledru

Univ. of Grenoble Alpes, LIG, F-38000 Grenoble, France
CNRS, LIG, F-38000 Grenoble, France
{Akram.Idani, Yves.Ledru}@imag.fr

Abstract. Several approaches dedicated to model access control policies (*e.g.* MDA-Security, SecureUML, UMLSec, etc) have used the Model Driven Engineering paradigm in order to ensure a clear separation of business rules and constraints specific to a target technology. Their supporting techniques mainly focus on modeling and verification of security rules without taking into account the functional model of the application and its interaction with the security model. In order to take into account both models, we developed the B4MSecure platform. It is a Model Driven Engineering platform that allows to graphically model and formally reason on both functional and security models. It translates a UML class diagram associated to a SecureUML model into formal B specifications. The resulting B specifications follow the separation of concerns principles in order to be able to validate both models separately and then validate their interactions. This paper gives an overview of our platform.
keywords: Formal methods, Security, RBAC, SecureUML

1 Introduction

In software engineering, method integration has been a challenge since several years. The objective is to link formal and graphical paradigms in order to guarantee the quality of specifications. Indeed, on the one hand, graphical languages (such as UML) have been widely used for specifying, visualizing, understanding and documenting software systems, but they suffer from the lack of precise semantical basis. On the other hand, formal methods (such as B [1]) are specifically used for safety critical systems in order to rigorously check their correctness but they lead to complex models which may be difficult to read and understand. These complementarities between formal and graphical languages motivate a lot of research teams to develop tools which combine both languages.

As far as secure information systems are concerned, existing research works in this context mainly focus, on the one hand, on modeling and verification of functional models without taking into account useful non-functional rules like access control policies. On the other hand, works dedicated to model access control policies (*e.g.* MDA-Security, SecureUML, UMLSec, etc) do not address the fact that the security model also refers to information of the functional model. Hence, evolutions of the functional state may influence the security behaviour and then open security flaws. Some well known insider attacks which were possible due to evolutions of the functional state can be cited:

- The account manager who creates a spurious account and adds himself to the system as a normal customer in order to transfer money into (or from) this spurious account. The access control policy should then forbid the account manager to evolve the functional entities Customer and Account whereby this malicious scenario could not take place.
- The attack of "Société Générale" in which the insider, through authorized actions, was able to conceal operations he has made on the market by introducing into the functional system fictive offsetting inverse operations.

2 B4MSecure overview

The B4MSecure tool is intended to model the information system as a whole by covering its functional description, and its security policy. The supporting models are based on UML for the functional description and SecureUML for the access control rules. A formal B specification is generated automatically from these models (figure 1). The resulting formal B specification allows then to formally reason on the whole system: functional and security models can be first validated separately, and then integrated in order to verify their interactions.

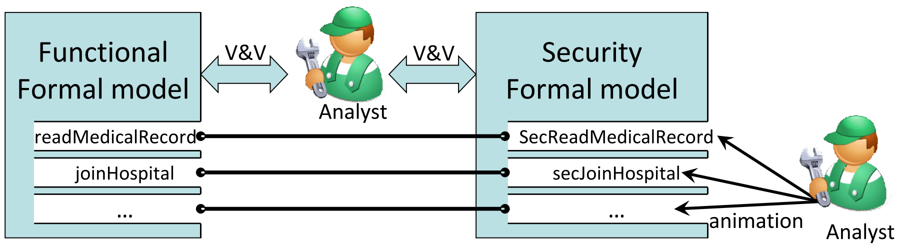


Fig. 1. Formal V&V activities of functional and security models

B4MSecure applies the model driven engineering (MDE) approach [11] in order to ensure a clear separation of business rules and their corresponding models. Semantics of the various models are defined on basis of UML, SecureUML and B methods meta-models and the various transformation rules are encoded by a set of mappings between these meta-models. The tool has several advantages:

(*i*) it clarifies and delineates a subset of source model structures for which the transformations are applicable;
(*ii*) it has a catalog of transformation rules expressed in a single transformation language; and
(*iii*) it provides a structured framework, based on meta-models, which clearly describes the semantics of the various models.

The tool is open source and available at http://b4msecure.forge.imag.fr. A video demo is also provided.

3 Translation of functional models

The translation of a UML diagram into a B specification is intended to take advantage of the B method tools in order to validate, by proofs and animations, the initial UML diagram. Several teams worked on this research problem and defined different kinds of mappings from UML into B: UML2B [3], UML2SQL [4], U2B [13] and ArgoUML+B [9]. Analysis of these various approaches show that each kind of UML-to-B mapping has its own objectives and characteristics:

- UML2SQL [4, 5]: B specifications are obtained by translating a set of UML diagrams which describe a database application.
- U2B [13, 14]: this work was intended to produce a B specification (so-called "natural") exempt from constructions related to the translation mechanism and which can complicate formal proofs.
- ArgoUML+B [6, 9]: this work tried to take into account complex UML features. It proposed, on the one hand, solutions for the translation of the UML inheritance mechanism, and on the other hand, a new formalization of state/transition diagrams.

Most of these tools except U2B are not publicly available despite the interest of their contributions. The B4MSecure tool encodes these mappings and then allows to reuse and combine rules issued from different UML-to-B approaches. In order to fit this need we have reimplemented existing translations using the java language. The advantage of our MDE architecture compared with existing UML-to-B tools (U2B, ArgoUML+B, UML2SQL) is its extensibility. In fact, in order to cover the transformation of UML constructs that have not been considered by the existing approaches, the rule-writer simply encodes in Java new rules and adds them to the catalog of transformations that we have implemented. Transformation from UML class diagrams into B is guided by their respective meta-models. The various java rules provided in the platform get UML concepts issued from the UML meta-model and produce instances of concepts issued from the B meta-model.

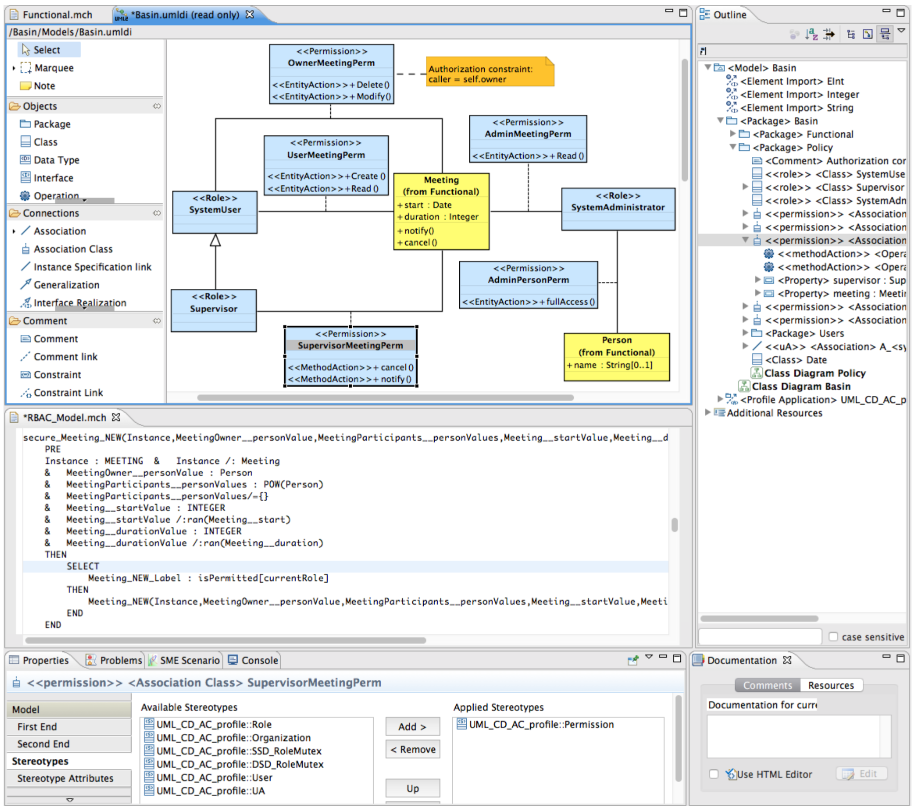
The tool produces one B model, from the functional UML class diagram, gathering its structural properties and all basic operations (constructors, destructors, getters and setters). By construction, the B proof obligations are true on the generated model. The resulting functional B machine covers a wide range of UML constructs: inheritance, mandatory and optional attributes, initial attribute value, navigation, read-only associations, unique values, etc. The analyst can then introduce, manually, additional invariants and user-defined operations and take benefit of a proof tool like AtelierB in order to validate the consistency of the functional specification. The platform provides an annotation mechanism allowing to integrate B invariants and specification of operations in the graphical model. This functionality is useful to avoid inconsistent evolutions of the graphical and formal methods.

4 Extraction of an RBAC filter from the security model

4.1 A brief overview

In order to express a security policy, the Role-Based Access Control (RBAC) model provides a powerful mechanism for reducing the complexity, cost, and potential for error of assigning and managing users and permissions. It is supported by several software products like popular commercial database management systems (*e.g.* Oracle, Sybase, ...) or webservers (*e.g.* JBoss). The available implementations of this model act like a filter which intercepts a user request to a resource in order to permit or deny the access to associated functional actions (*e.g.* transactions on databases, file operations, ...). Our tool is based on the same principle at a modeling level. Each functional operation is encapsulated in a secure operation which checks that the current user is authorized to call it.

The tool is based on the eclipse Topcased environment. Graphical modeling of a RBAC policy in B4MSecure is done using a UML profile inspired by SecureUML [2]. Figure 2 gives a screenshot of the graphical modeler applying this security profile.



Functional.mch
*Basin.umldi (read only)
/Basin/Models/Basin.umldi
<<Permission>> OwnerMeetingPerm
<<EntityAction>>+Delete()
<<EntityAction>>+Modify()
Authorization constraint: caller = self.owner
<<Permission>> AdminMeetingPerm
<<EntityAction>>+Read()
<<Permission>> UserMeetingPerm
<<EntityAction>>+Create()
<<EntityAction>>+Read()
<<Role>> SystemUser
Meeting (from Functional)
+start : Date
+duration : Integer
+notify()
+cancel()
<<Role>> SystemAdministrator
<<Permission>> AdminPersonPerm
<<EntityAction>>+fullAccess()
<<Role>> Supervisor
<<Permission>> SupervisorMeetingPerm
<<MethodAction>>+cancel()
<<MethodAction>>+notify()
Person (from Functional)
+name : String[0..1]
*RBAC_Model.mch

```
secure_Meeting_NEW(Instance,MeetingOwner__personValue,MeetingParticipants__personValues,Meeting__startValue,Meeting__d
   PRE
   Instance : MEETING  &   Instance /: Meeting
   &   MeetingOwner__personValue : Person
   &   MeetingParticipants__personValues : POW(Person)
   &   MeetingParticipants__personValues/={}
   &   Meeting__startValue : INTEGER
   &   Meeting__startValue /:ran(Meeting__start)
   &   Meeting__durationValue : INTEGER
   &   Meeting__durationValue /:ran(Meeting__duration)
   THEN
      SELECT
         Meeting_NEW_Label : isPermitted[currentRole]
      THEN
         Meeting_NEW(Instance,MeetingOwner__personValue,MeetingParticipants__personValues,Meeting__startValue,Meeti
      END
   END
```

Fig. 2. B4MSecure screenshot

The tool produces a B specification gathering a set of secured operations which filter the access to the functional model. Proof-based validation of this specification should automatically succeed provided the SecureUML model is conformant to structural invariants: no cycles in role-hierarchy, no static separation of duties violation, etc. We note that the tool does not produce administrative operations allowing to modify dynamically the policy and hence validation of this model is only based on well-formedness properties.

Animation of secured operations gives access only to the authorized functional operations for the user who is trying to execute them. This approach allows to validate the functional model as well as the security policy. In fact, animation of an authorized operation changes the state of the functional model and hence allows the analyst to validate both models.

4.2 Case studies

In [10] we discussed the benefits of the resulting B specifications and how the expected validation activities can be done. These validation activities include classical proof obligations, but also the animation and test of functional and security models, using ProB [8]. Formally taking into account links between both functional and security models allows to address challenging vulnerabilities [7]. Actually our security model, based on SecureUML, allows to associate authorisation constraints to the permissions. Authorisation constraints express conditions on the functional state in order to grant permission. This enables insiders attacks where the attacker modifies the functional state in order to get illegal permissions. In [12] we showed how the B specifications produced by our platform can be useful in order to exhibit malicious scenarios leading to insiders attacks.

The platform was experimented on several examples of various sizes and was applied on a real case study[1] proposed by the French Institute of Mountain Medicine Research (Ifremmont[2]). It is a pre-hospital information system managing medical patient data. The associated conceptual model is composed of fifteen functional classes and several control rules. The resulting B specification counts 1730 lines for the functional model, 2652 lines for the access control filter and allowed several validation activities:

- acceptance validation guided by use-cases : each functional use case identified during requirements analysis was animated on the secured specification, showing that the security filter does not prevent the normal use of the system.
- systematic testing of access control rules : each access control rule was tested to show that it was able to both grant and deny permission to access the associated resource. This animation aims at detecting errors in the expression of these access control rules.

[1] http://telemedecine.ifremmont.com/ifrelab/index.php?Wwwresamuorg
[2] http://www.ifremmont.com

– search for malicious insider scenarios : several attack scenarios were designed manually and animated on the formal model, showing its robustness against these attacks. Current work in our team tries to automatically synthesize such attacks by exploration of the specification.

This case study showed the usefulness of the tool while designing a security policy. It supports this design at a level which abstracts away implementation details such as authentification mechanisms and cryptography, and concentrates on the consistency of the set of access control rules.

5 Conclusion

In most existing Information Systems, functional and security requirements are mixed in the application code. It is, therefore, difficult to understand these systems and modify them in order to maintain, evolve and correct the security policy. In order to master complexity of systems, the MDE paradigm advocates for a separation of concerns and the use of models throughout the development process. Therefore, Information Systems security is a domain where the potential of the MDE approach is highly useful. Indeed, modeling separately functional and security models allows to better understand, validate and maintain these models.

Although it is useful to analyse and validate both models in isolation, which is addressed by several works, interactions between these models must also be taken into account. Such interactions result from the fact that constraints expressed in the security model also refer to information of the functional model. Hence, evolutions of the functional state have an impact on the security behaviour.

Conversely, security constraints often depend on the functional behaviour. The B4MSecure platform allows, on the one hand, this separation of concerns, and on the other hand, the investigation of links between both functional and security models. The platform allows: a graphical modeling of a functional UML class diagram, a graphical modeling of an access control policy using a UML profile for RBAC (Role Based Access Control) and which is inspired by SecureUML, and the translation of both models into B specifications in order to formally reason about them.

The various B specifications extracted from functional and security models allow several kinds of validation. This paper addressed mainly the principles of the tool and discussed some validation activities that we have done on several case studies. Other kinds of validation can be addressed such as the use of a constraint solver, or symbolic animation, etc.

In the future, we plan to take into account translation of other UML diagrams like state/transition and activity diagrams. Currently, the platform only deals with structural aspects of an Information System. We also plan to cover translation of other security models. Our interest is directed to attribute-based access control models.

References

1. Jean-Raymond Abrial. *The B-book: assigning programs to meanings*. Cambridge University Press, 1996.
2. David A. Basin, Manuel Clavel, Jürgen Doser, and Marina Egea. Automated analysis of security-design models. *Information & Software Technology*, 51(5):815–831, 2009.
3. Lotfi Hazem, Nicole Levy, and Rafael Marcano-Kamenoff. UML2B : un outil pour la génération de modèles formels. In Jacques Julliand, editor, *AFADL'2004 - Session Outils*, 2004.
4. Régine Laleau and Amel Mammar. An Overview of a Method and Its Support Tool for Generating B Specifications from UML Notations. In *15th IEEE International Conference on Automated Software Engineering*, pages 269–272, 2000. IEEE Computer Society Press.
5. Regine Laleau and Fiona Polack. Coming and Going from UML to B: A Proposal to Support Traceability in Rigorous IS Development. In *2nd International Conference of B and Z Users*, volume 2272 of *LNCS*, pages 517–534. Springer, 2002.
6. Hung Ledang. Automatic Translation from UML Specifications to B. In *Automated Software Engineering*, page 436. IEEE Computer Society, 2001.
7. Yves Ledru, Akram Idani, Jérémy Milhau, Nafees Qamar, Régine Laleau, Jean-Luc Richier, and Mohamed-Amine Labiadh. Taking into account functional models in the validation of IS security policies. In *1st International Workshop on Information Systems Security Engineering, CAiSE 2011*, volume 83 of *LNBP*, pages 592–606. Springer, 2011.
8. M. Leuschel and M. Butler. ProB: A Model Checker for B. In *FME 2003: Formal Methods Europe*, volume 2805 of *LNCS*, pages 855–874. Springer-Verlag, 2003.
9. Eric Meyer. *Développements formels par objets : utilisation conjointe de B et d'UML*. PhD thesis, Université de Nancy 2, Mars 2001.
10. Jérémy Milhau, Akram Idani, Régine Laleau, Mohamed-Amine Labiadh, Yves Ledru, and Marc Frappier. Combining UML, ASTD and B for the formal specification of an access control filter. *ISSE*, 7(4):303–313, 2011.
11. OMG. Mda guide version 1.0.1. http://www.omg.org/mda/, June 2003.
12. Amira Radhouani, Akram Idani, Yves Ledru, and Narjes Ben Rajeb. Extraction of insider attack scenarios from a formal information system modeling. In *Proceedings of the Formal Methods for Security Workshop co-located with the PetriNets-2014 Conference*, volume 1158 of *CEUR Workshop Proceedings*, pages 5–19. CEUR-WS.org, 2014.
13. Colin Snook and Michael Butler. U2B-A tool for translating UML-B models into B. In J. Mermet, editor, *UML-B Specification for Proven Embedded Systems Design*, 2004.
14. Colin Snook and Michael Butler. UML-B: Formal modeling and design aided by UML. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 15(1):92–122, 2006.