Formal Model Driven Engineering (FM & MDE)

Habilitation à Diriger des Recherches

Akram Idani

26 Mai 2023
“It is easier to perceive error than to find truth, for the former lies on the surface and is easily seen, while the latter lies in the depth, where few are willing to search for it.” – Johann Wolfgang von Goethe

Overview

- MDE
  - Communication
  - Standard

- Proofs / Refinements

- FM
  - Automated reasoning
  - Safety-critical systems

« The picture changes dramatically for safety-critical, high-assurance software. Here, validation by testing reaches its limits and needs to be complemented or even replaced by the use of formal methods such as model checking, static analysis, and program proof. »

“It is easier to perceive error than to find truth, for the former lies on the surface and is easily seen, while the latter lies in the depth, where few are willing to search for it.” – Johann Wolfgang von Goethe

Outline

• Introduction
• Model Driven Security
  – UML / SecureUML
  – Insider attacks
• Executable DSLs
  – Motivations and results
  – Applications
• Conclusion
• Selkis (projet ANR, 2008/2012) : A development method of secure health care networks information systems : from requirements engineering to implementation


• NextRegio (IRT Railenium, 2015/2019) : solutions d'exploitation pour les lignes de desserte fine du territoire.

« Members of the FST and the MDE communities need to collaborate. »

« Modeling languages must have formally defined semantics if they are to be used to create analyzable models. »
Introduction

• Two schools
  – The Extensible General-Purpose Modeling Language School
    • e.g. SysML, OntoML, UML-RT
    • Provide a language with extension mechanisms
    • **B4MSecure: Model-Driven Security**
  – The Domain Specific Modeling Language School
    • e.g. Capella, OWL, Lustre
    • Tool support for engineering modeling languages
    • **Meeuse: grammars and Meta(-meta)-modeling**


« […] can both play vital roles in an MDE environment. We envisage that research in both schools will provide valuable insights and research results that will lead to a convergence of ideas. »
Outline

• Introduction

• Model Driven Security
  – UML / SecureUML
  – Insider attacks

• Executable DSLs
  – Motivations and results
  – Applications

• Conclusion
Access control

- Access control is horizontal:
  - Ad-hoc integration is error prone and costly
- High-level of abstraction:
  - Modeling structure and behavior
- Correctness:
  - Testing / model-checking / Proofs
- Prevention against attacks

Information System

![Chart showing the percentage of insider and outsider attacks in 2014 and 2015.]
Role-Based Access Control

- **RBAC**: 1992 by D. Ferraiolo and R. Kuhn 2000

- **Constraints**
  - Static Separation of duties
  - Dynamic Separation of duties
  - Contextual
  - **Data-centric too!**

- **ABAC**: 2011
MDE for Security

- Separation of concerns
- e.g. SecureUML
MDE for Security

- Separation of concerns
- e.g. SecureUML
- Contribution:
  - Formal MDS
    - Proof
    - Test/Animation
    - Model-checking
MDE for Security

- Separation of concerns
- e.g. SecureUML
- Contribution:
  - Formal MDS
    - Proof
    - Test/Animation
    - Model-checking
A Simple Example
Verification & Validation (1/2)

- **Static V&V**
  - Evaluation is done in a given state
    - Given an atomic action, which roles can perform this action?
    - Given a role and an atomic action, under which circumstances can a user in this role perform this action?
    - Do two permissions overlap?

Is the manager able to transfer funds from a customer account?

A static query: NO
Dynamic V&V

- **Jaza/animation [PhD of N. Qamar (50%)]: static queries + animation**
- USE/OCLE [Yu et al. 2009], generation of execution schemas.
- RW [Zhang et al. 2008], model-checking
- SMP (Logic for State Modifying Policies) [Becker et Nana 2010]
- Theorem proving [A. Mammar et al., FAC’15]

Is there a sequence of operations that can be executed by a manager in order to transfer funds from a customer’s account?
Research directions

• Modeling/Testing
  – **Configurable transformations**
  – Abstraction levels: M1 ; M2 and M1/M2
  – PhD M.-A. Labiadh (50%) + 1 M2 + 1 PFE

• Attacks
  – Forward search
    • **Guided model-checking** (2 M2)
    • Ant-colony optimisation (2 IRL-ENSIMAG)
  – Backward search
    • **Proof** and constraint solving
    • PhD A. Radhouani (33%) + 1 M2

• Business processes
  – Task-based access control: PhD S. Chehida (25%)
  – BPMN: 2 M2, collaboration with SIGMA/LIG
Modeling/Testing
Modeling/Testing (1/3)

- Various tools and approaches, but
  - Most of them are unavailable
  - Should be revisited/updated

(Snook et al., 2004) (Laleau, 2002) (Meyer, 2001)
## Modeling/Testing (2/3)

<table>
<thead>
<tr>
<th>Feature</th>
<th>R. Laleau</th>
<th>E. Meyer</th>
<th>C. Snook</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Classes (fixed instances)</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Attributes</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>(Single/Multi)-valued attributes</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Inheritance</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Multiplicities</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Navigation</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Roles</td>
<td>X</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Association constraints</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Fixed/Non-fixed associations</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Association classes</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Association classes with inheritance or other relationships</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Parameterized classes</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>
Modeling/Testing (3/3)

- Access Control
- MDE for security
- V&V: static vs dynamic
- Modeling/Testing
- Insider attacks

Introduction
Model Driven Security
Executable DSLs
Conclusion & Perspectives
Attacks
Insider attacks: forward search (1/3)

• Guided model checking (CSP | | B)

Is there a sequence of operations that can be executed by a manager in order to transfer funds from a customer’s account?

MAIN = UI

UI = (Connect?user!{AccountManager} -> setCurrentUser(user) -> MANAGER_FUNC
[] Connect?user!{CustomerUser} -> setCurrentUser(user) -> CLIENT_FUNC)
; disconnectUser -> UI

MANAGER_FUNC =
    CREATE_ACCOUNT [] CREATE_CUSTOMER [] UPDATE_CUSTOMER [] SKIP

CREATE_ACCOUNT =
    secure_Account_NEW -> (CREATE_ACCOUNT [] MANAGERFUNC)

CREATE_CUSTOMER =
    secure_Customer_NEW?customer -> secure_Customer__SetName!customer
    -> (ADD_CUSTOMER_ACCOUNT(customer) [] CREATE_CUSTOMER [] MANAGERFUNC)
Insider attacks: forward search (2/3)

- Guided model checking (CSP | | B)

Is there a sequence of operations that can be executed by a manager in order to transfer funds from a customer’s account?

MAIN = UI [|| Connect, secure_Account_transferFunds ||] ATTACK

ATTACK = ATTACKER ||| secure_Account_transferFunds!cpt1 -> goal -> SKIP

ATTACKER = []role:Set(ROLES) @ Connect!Bob!role -> ATTACKER
Insider attacks: forward search (3/3)

• Guided model checking (CSP | | B)

Is there a sequence of operations that can be executed by a manager in order to transfer funds from a customer’s account?

Connect(Bob, {AccountManager})
setCurrentUser(Bob)
secure_Account.NEW(cpt3, 333)
secure_Customer.NEW(Bob, {cpt3})
secure_Customer.SetName(Bob, "...")
Insider attacks: forward search (3/3)

- Guided model checking (CSP | | B)

Is there a sequence of operations that can be executed by a manager in order to transfer funds from a customer’s account?

Connect(Bob, {AccountManager});
setCurrentUser(Bob);
secure_Account_NEW(cpt3, 333);
secure_Customer_New(cpt4, 444);
secure_Customer_AddAccount(Paul, {cpt4});
secure_Customer_RemoveAccount(Paul, {cpt1});
secure_Customer_AddAccount(Bob, {cpt1});
Insider attacks: forward search (3/3)

- Guided model checking (CSP | B)

Is there a sequence of operations that can be executed by a manager in order to transfer funds from a customer’s account?

```
Connect(Bob, {AccountManager});
setCurrentUser(Bob);
secure_Account.NEW(cpt3, 333);
secure_Customer.NEW(Bob, {cpt3});
secure_Customer.SetName(Bob, "..."),
secure_Customer.RemoveAccount(Paul, {cpt4});
secure_Customer.AddAccount(Paul, {cpt4}));
secure_Account.NEW(cpt4, 444);
disConnect(Bob);
Connect(Bob, {CustomerUser});
secure_Account.transferFunds(cpt1, 333, 500);
```
Insider attacks: backward search

States = Predicates

\[ (u, R, c_{auth}) \models \text{False} \]
\[ (u, R, c_{auth}) \models \text{True} \]

1. always enabled: \( \forall x. E \Rightarrow \text{Pre}(op) \)
2. never enabled: \( \forall x. E \Rightarrow \neg \text{Pre}(op) \)
3. possibly enabled (\( \neg (1) \land \neg (2) \)): \( \exists x. E \land \text{Pre}(op) \)
4. always reached: \( \forall x. E \land \text{Pre}(op) \Rightarrow [\text{Action}(op)]F \)
5. never reachable: \( \forall x. E \land \text{Pre}(op) \Rightarrow [\text{Action}(op)]\neg F \)
6. possibly reached (\( \neg (4) \land \neg (5) \)): \( \exists x. E \land \text{Pre}(op) \land \neg [\text{Action}(op)]\neg F \)
Insider attacks: backward search

\[- Pf \land \neg Pre\( \text{op}_n \) \land Pre\( \text{op}_{n-1} \)\]

\[- Pf \land \neg Pre\( \text{op}_n \) \land \neg Pre\( \text{op}_{n-1} \) \land Pre\( \text{op}_{n-2} \)\]

\[- Pf \land \neg Pre\( \text{op}_n \) \land \ldots \land \neg Pre\( \text{op}_{n-3} \)\]

\[- Pf \land \neg Pre\( \text{op}_n \) \land \ldots \land Pre\( \text{op}_{n-3} \)\]
Insider attacks: backward search

\[
\neg P_F \land \neg \text{pre}(op_n) \land \text{pre}(op_{n-1})
\]

\[
\neg P_F \land \neg \text{pre}(op_n) \land \neg \text{pre}(op_{n-1}) \land \text{pre}(op_{n-2})
\]

\[
\neg P_F \land \neg \text{pre}(op_n) \land \ldots \land \neg \text{pre}(op_{n-3})
\]

\[
\neg P_F \land \neg \text{pre}(op_n) \land \ldots \land \text{pre}(op_{n-3})
\]

\[
Q_{symb} = \langle \text{init}, op_{n-3}, op_{n-2}, op_{n-1}, op_n \rangle
\]
Outline

• Introduction
• **Model Driven Security**
  – UML / SecureUML
  – Insider attacks
• **Executable DSLs**
  – Motivations and results
  – Applications
• Conclusion
UML vs DSLs

- NextRegio (IRT Railenium, SNCF réseau)
- Undesirable scenarios + responsibilities

```plaintext
{{B}}
/* AuthConstraint : the train has a on signal or no signal is present on the next portion */
/* First check that there is a next portion otherwise, the train is not permitted to move */
TrainOfPortion(aTrain) : dom(next_portion(Train_direction(aTrain))) &
/* Identify the next portion */
LET new_portion BE new_portion = (next_portion(Train_direction(aTrain)))(TrainOfPortion(aTrain))
IN /* absence of signal or signal in the on state */
new_portion /: ran(LightOfPortion )
or
Light_direction(LightOfPortion~(new_portion)) /= Train_direction(aTrain)
or
Light_state(LightOfPortion~(new_portion)) = on
END
```
UML vs DSLs

• NextRegio (IRT Railenium, SNCF réseau)
• Undesirable scenarios + responsibilities
UML vs DSLs

- NextRegio (IRT Railenium, SNCF réseau)
- Undesirable scenarios + responsibilities
Positioning

• DSL $\rightarrow$ FM
  – Survey 1 (2006/2012):

  “there is an urgent need in DSL research for identifying the reasons for lack of using formal methods within domain analysis and possible solutions for improvement”


Refers to testing as “the” verification feature of Language Work-Benches
Personal opinion

• DSL → FM

  – Translational approaches:

    • Advantages:
      1. provide a well-defined and understood semantics, and
      2. the DSL can convey existing tools of the language into which it is translated.

    • Limitations:
      3. it is very challenging to correctly map the constructs of the DSL into the constructs of the target language, and
      4. the mapping of the verification results back into the DSL appears as a major issue of existing approaches.
Applications

- Railway systems - PhD A. Yar (33%)
Applications

- **NextRegio Project** (Railway systems)
  - V1 (POC)
- **TTC’19** (Awarded)
  - V2 (Beta)
  - Big models
- **MeeNET** (Powered by Meeduse)
  - V3 (Stable)
- **DomoSur Project** (New features)
- **xOWL Project**
- **Multi-targets**
  - Event-B
  - Z
  - Alloy

**Timeline**:
- **2018**: NextRegio Project
- **2019**: TTC’19 (Awarded), MeeNET (Powered by Meeduse)
- **2020**: Basic PTNets, PNML (XML), PNML Refinements, Transpilation Of PNML
- **2021**: @Runtime, SmartHome, M2M, Isomorphism, Propagation, Ontologies, Patterns, Evolution
- **2022**: isoMorphism Propagation Ontologies Patterns Evolution
- **2023**: Multi-targets
Outline

• Introduction

• **Model Driven Security**
  – UML / SecureUML
  – Insider attacks

• **Executable DSLs**
  – Motivations and results
  – Applications

• **Conclusion & Perspectives**
Conclusion & perspectives

• FM & MDE
  – MDS: Healthcare IS (Security)
  – xDSLs: Railway systems (Safety)

• B4MSecure / Meeduse

• FM and MDS: is (still) an active topic
  – Conformance PIM/PSM/Application
  – Monitor/Controller synthesis
  – Align (Data, Security, business)-Models

• FM and xDSLs: not a new topic but requires further works
  – Material already exists (iFM, MoDeVVa, UML&FM, VOLT, etc)
  – Investigate other languages (Maude, ASM, Z, Alloy, etc)
  – Provide a unifying framework
Facts

• B4MSecure: Teaching (master degree, MoSIG 2)
  – Topic: Software/Hardware: quality engineering, models of computation
  – Lecture: Information Security

• Meeduse: Recent papers
  – A. Idani., The B Method meets MDE.
    Research Challenges in Information Science (RCIS-2022)
  – A. Idani et al., Alliance of model-driven engineering with a proof-based formal approach.
  – A. Idani., Meeduse: A Tool to Build and Run Proved DSLs.
  – A. Idani et al., Incremental Development of a Safety Critical System Combining formal
    Methods and DSMLs - Application to a Railway System.
  – A. Idani et al., Towards a Tool-Based Domain Specific Approach for Railway Systems
    Modeling and Validation.