Validation of a Security Policy by the Test of its Formal B Specification a Case Study

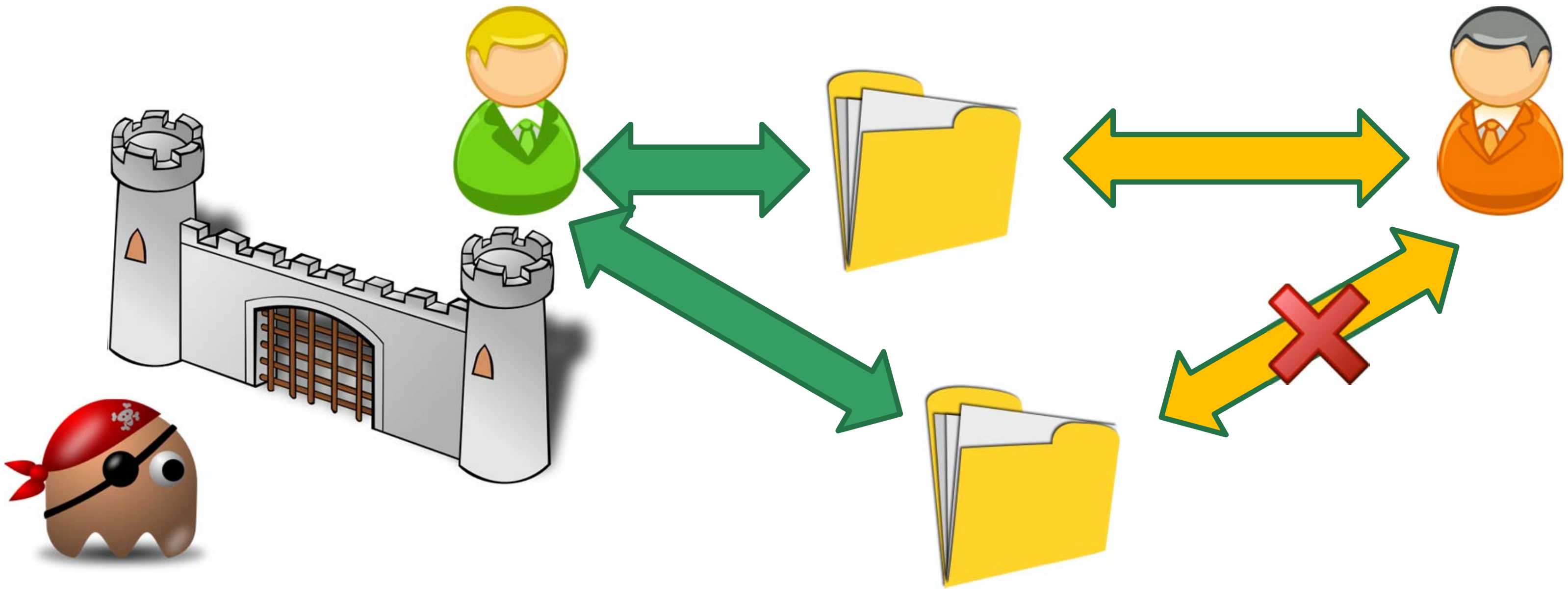
Yves Ledru, Akram Idani, Jean-Luc Richier

VASCO team

Univ. Grenoble Alpes/CNRS, LIG, UMR5217, 38000, Grenoble, France

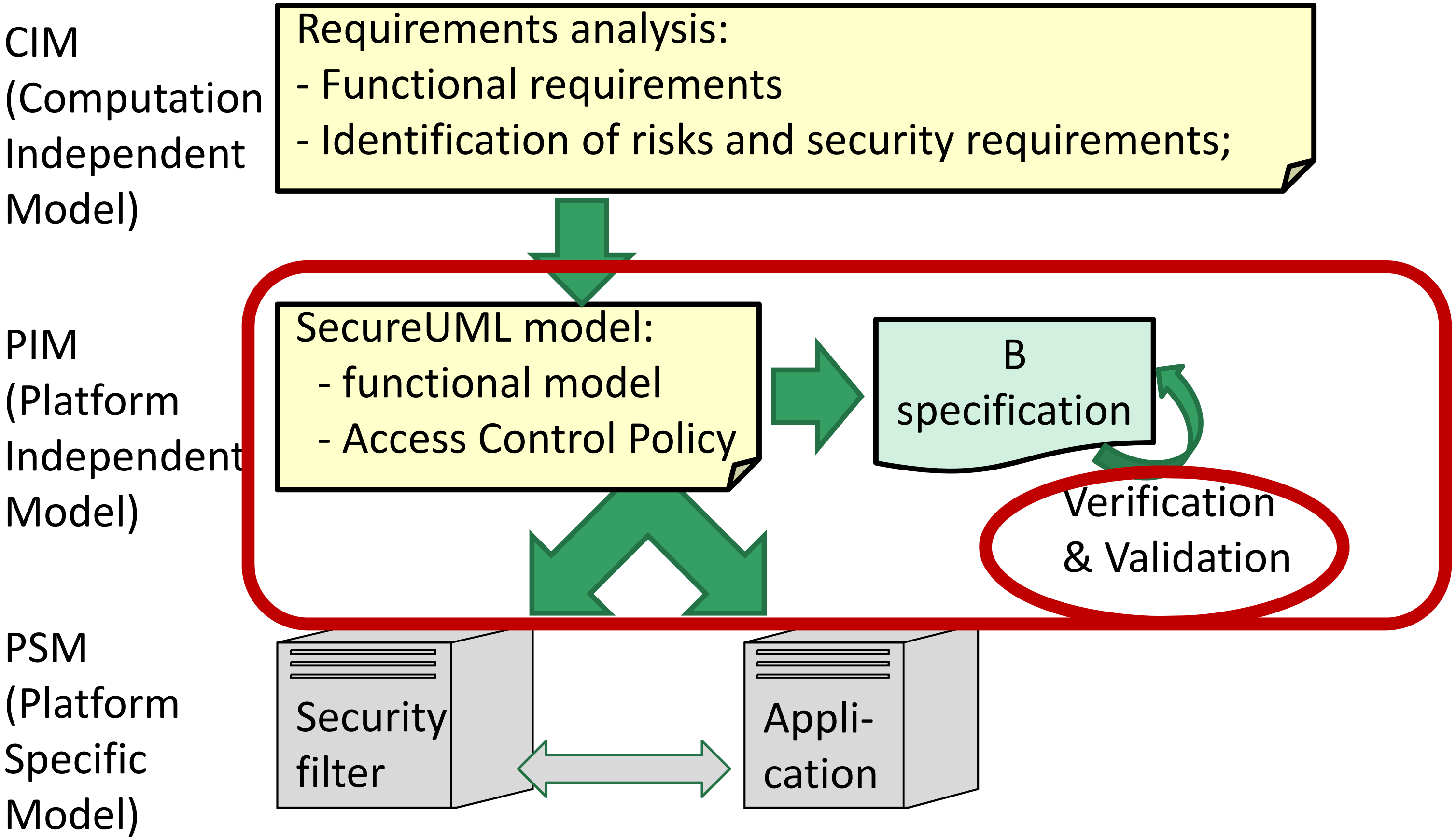
Yves.Ledru@imag.fr

Secure Information Systems



- Information systems : data and functions to coordinate people
- Secure information systems : protect the access to data and functions
  - From outsiders
  - From insiders
- Insider attacks : performed by legitimate users who abuse their privileges.

The Selkis approach



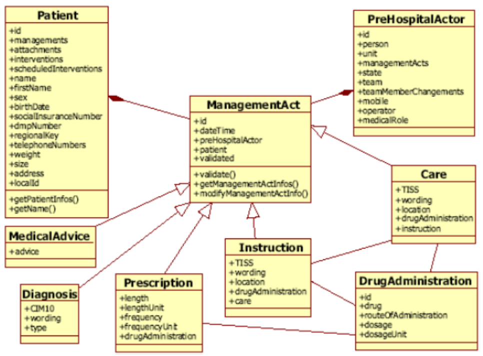
This paper : V&V of the PIM by its translation into a B specification.
Focus on the policy (PIM),
not on detailed underlying security mechanisms (PSM).

Res@mu Case Study

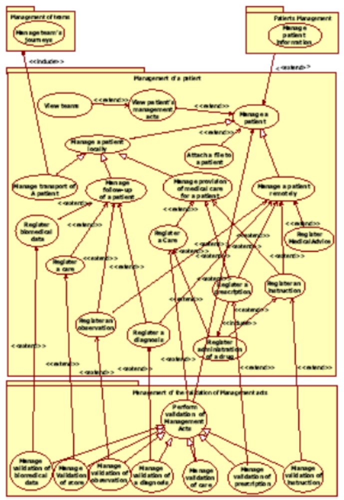
- Information system for an urgency medical help service (SAMU)

SAMU

- Developed by IFREMMONT, a french association for e-medecine.



- Functional model : 77 classes, 100 use cases developed before this study.

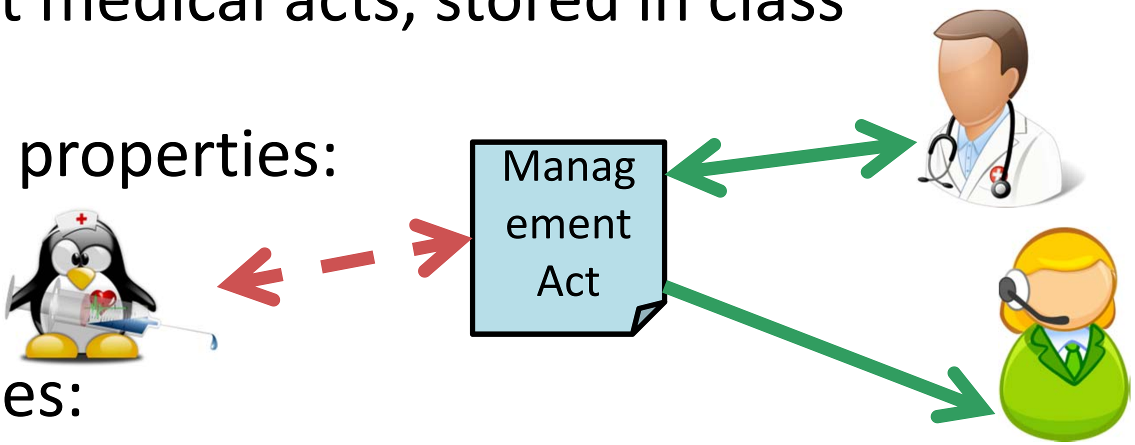


The need for security in Res@mu

- Access to the information system must be restricted to authorized personal
- The authorized personal are numerous and evolve over the life-time of the information system => need for a role-based approach
- Medical data
  - Are confidential
  - Must be available to the rescue teams
  - Must be protected against unauthorized modifications (integrity)

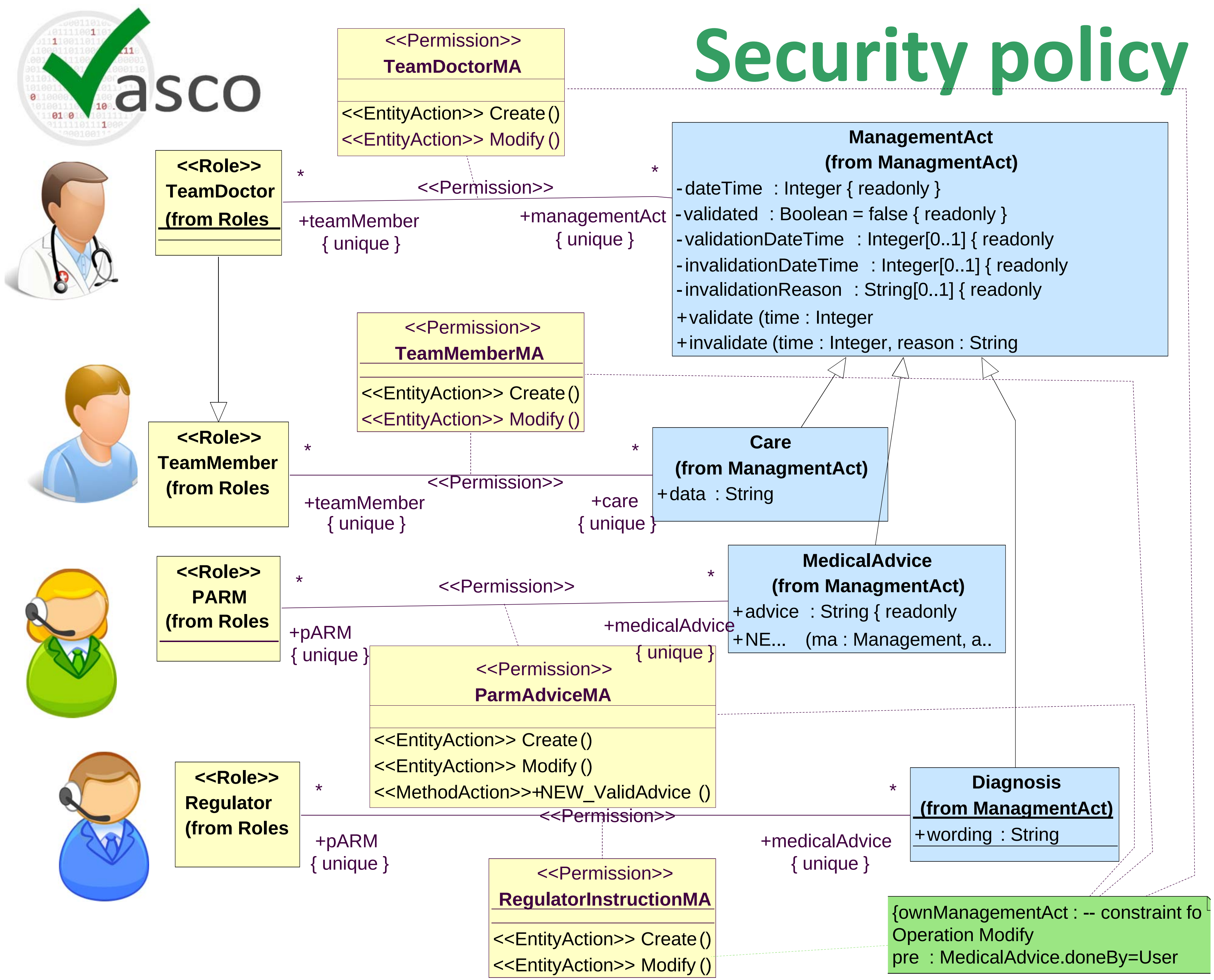
Security target

- Instead of protecting everything…
- … we focused on a single security target: Information about medical acts, stored in class ManagementAct
- Expected security properties:
  - Confidentiality
  - Integrity



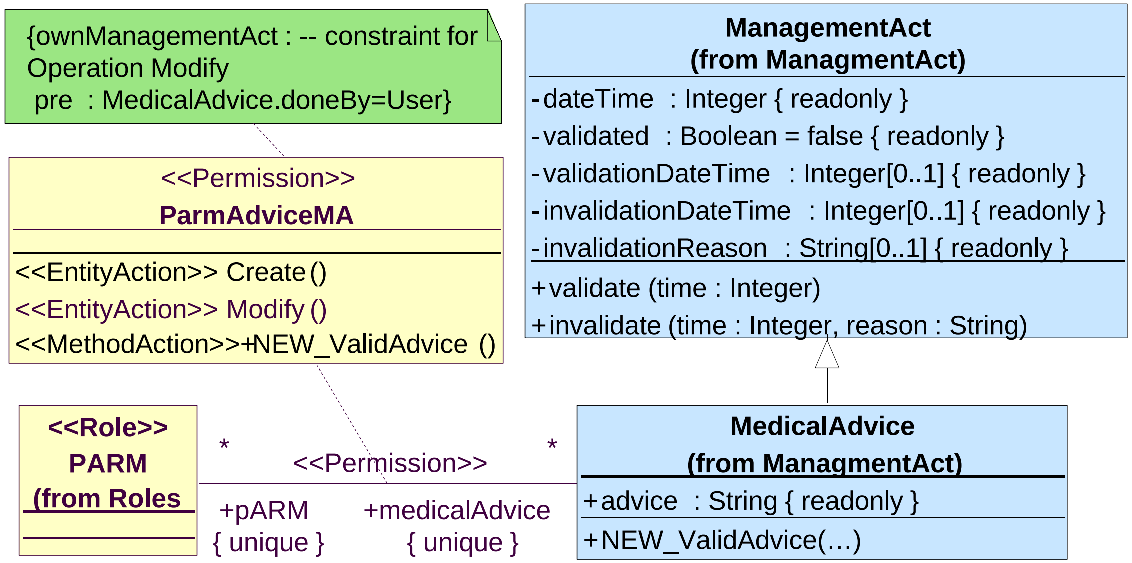
- Access control rules:
  - Read access for the members of the teams in charge of the patient
  - Write access to the qualified person performing the act
  - No access for other users

Security policy



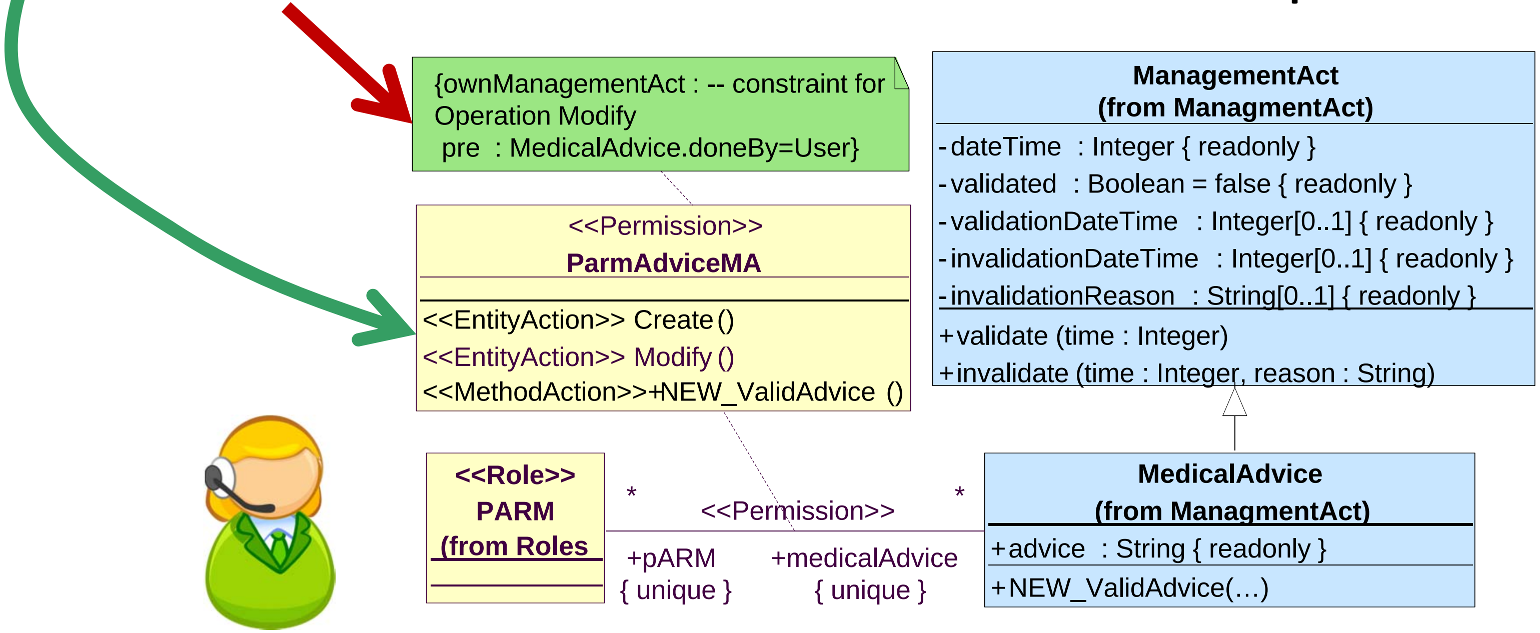
Separation of concerns

- UML classes => functional model
- Roles and permissions => security model
- Autorisation constraints : part of the security model but referring to the functional model



A permission rule

- Expressed in SecureUML
- Relates a role to the associated class
- Lists the operations permitted for this role
- An autorisation constraint restricts the permission



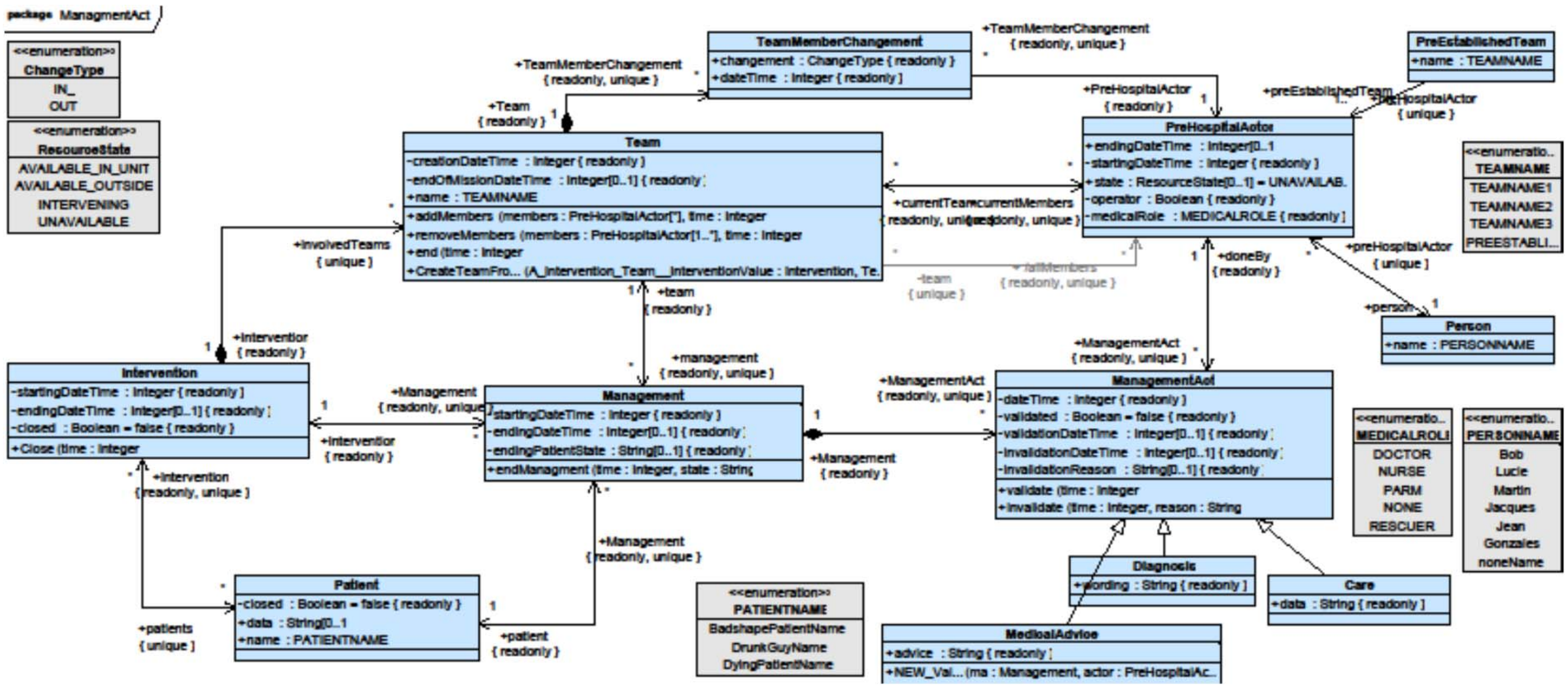
Evolutions of the functional state may influence the constraint!

Validation of the policy

- Is it the right model? The right rules?
- Validation based on
  - The translation of the functional and security models into B
  - Animation or test of the models
- Functional model too big for validation tools
  => need for simplifications…

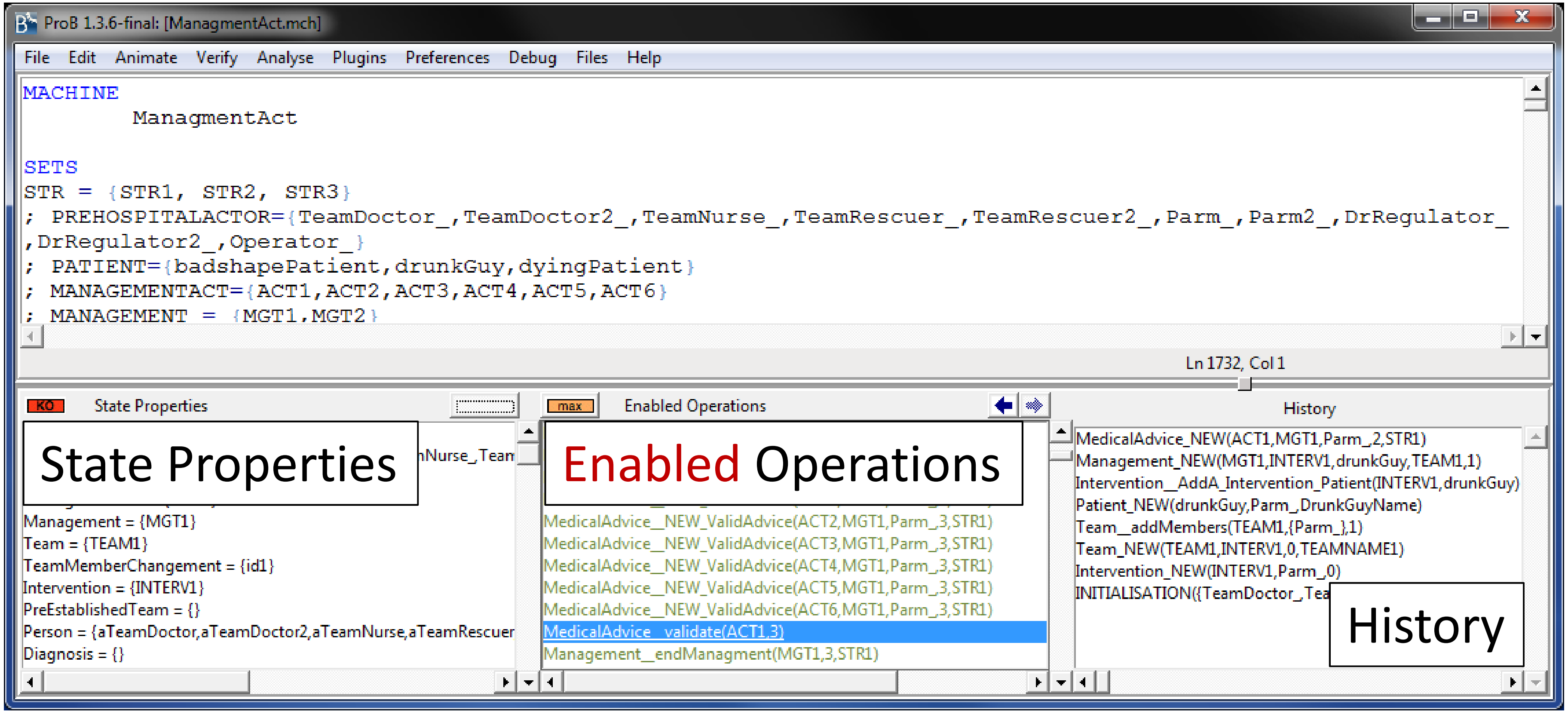
Simplified Class Diagram

- 12 classes selected amongst the 77 classes:
  - Directly related to Management Acts and autorisation constraints
  - Only relevant attributes are kept



Translation and animation

- UML models augmented with B annotations
- Translated into B specifications using B4MSecure tool
  1730 lines for functional model, 2652 lines for security model
- Animated with ProB, showing enabled operations at each step



A secure operation

```
secure_MedicalAdvice__validate(Instance,time)=
PRE
Instance : MedicalAdvice & time : INTEGER
& ManagementAct__validated(Instance)=FALSE
& Instance /: dom(ManagementAct__invalidationDateTime)
& time > ManagementAct__dateTime(Instance) /* Precondition generated from annotation*/
THEN
SELECT
MedicalAdvice__validate_Label : isPermitted[currentRole]
& currentUser : A_preHospitalActor_person[
A_Team_PreHospitalActor~[A_Team_Management(A_Management_ManagementAct(Instance))]]
& A_PreHospitalActor_ManagementAct(Instance) : A_preHospitalActor_person~[currentUser]
THEN
MedicalAdvice__validate(Instance,time)
END
END;
```

Precondition taken from the functional precondition

Guard enforcing the security policy

Encapsulates the functional operation

Vasco

Validation activities

1. B proof obligations
2. Functional animation
3. Animation of secured operations
4. Systematic test of the permissions
5. Attacks

1. B Proof obligations

- Discharged using Atelier B tool.
- On the functional model :
  - Checks that invariant properties (added as annotations) are consistent with the operations
- On the security model
  - Useless because we use a generic security model, instanciated by the policy
  - The generic security model satisfies the proof obligations.

2. Functional animation

- Based on (functional) use cases
- Shows that the use case is feasible with the current functional specification.
- Helps finding missing operations or too strong preconditions.

```
Intervention_NEW(INTERV1,Parm_,0) ;
Team_NEW(TEAM1,INTERV1,0,TEAMNAME1) ;
Team__addMembers(TEAM1,{Parm_},1) ;
Patient_NEW(drunkGuy,Parm_,DrunkGuyName) ;
Intervention__AddA_Intervention_Patient
                    (INTERV1,drunkGuy) ;
Management_NEW(MGT1,INTERV1,drunkGuy,TEAM1,1) ;
MedicalAdvice_NEW(ACT1,MGT1,Parm_,2,STR1) ;
MedicalAdvice__validate(ACT1,3)
```

Initial sequence needed to perform the use case (sequence found with the help of ProB enabled ops)

Use case:

1. Create a medical advice
2. Validate it!

3. Animation of secured operations

- The same use cases can be played using the secured version of its operations + additional security related actions.
- This shows that the security policy does not block functional use cases.

```
setPermissions ;
Connect(aParm,{PARM}) ;
changeCurrentUser(aParm) ;
secure_Intervention_NEW(INTERV1,Parm_,0) ;
secure_Team_NEW(TEAM1,INTERV1,0,TEAMNAME1) ;
secure_Team__addMembers(TEAM1,{Parm_},1) ;
secure_Patient_NEW(drunkGuy,Parm_,DrunkGuyName) ;
secure_Intervention__AddA_Intervention_Patient
        (INTERV1,drunkGuy) ;
secure_Management_NEW
        (MGT1,INTERV1,drunkGuy,TEAM1,1) ;
secure_MedicalAdvice_NEW(ACT1,MGT1,Parm_,2,STR1) ;
secure_MedicalAdvice__validate(ACT1,3)
```

4. Systematic test of rules

- Positive and negative tests for each rule
- Test cases differ only by the nominal/robustness code

preamble

```
setPermissions ;
…
secure_MedicalAdvice_NEW(ACT1,MGT1,Parm_,2,STR1) ;
```

nominal

```
Skip
```

robustness1

```
Connect(aTeamRescuer,{Rescuer}) ;
changeCurrentUser(aTeamRescuer) ;
disconnectUser(aParm) ;
Connect(aParm,{Operator}) ;
changeCurrentUser(aParm)
```

robustness2

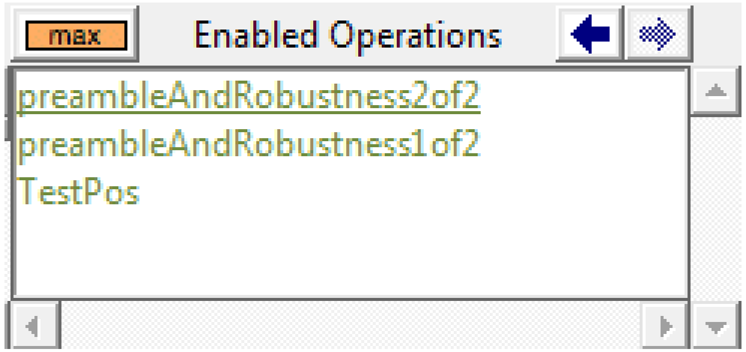
```
secure_Team__addMembers
        (TEAM1,{Parm2_},1) ;
Connect(aParm2,{PARM}) ;
changeCurrentUser(aParm2)
```

call

```
secure_MedicalAdvice__validate(ACT1,3)
```

4. Systematic test of the rules (2)

TestPos = **BEGIN** preamble ; nominal ; call **END** ;
TestNeg1of2 = **BEGIN** preamble ; robustness1 ; call **END** ;
TestNeg2of2 = **BEGIN** preamble ; robustness2 ; call **END** ;
preambleAndRobustness1of2 = **BEGIN** preamble ; robustness1 **END** ;
preambleAndRobustness2of2 = **BEGIN** preamble ; robustness2 **END**



- The positive test succeeds.
- The negative tests are not enabled by ProB.
- preambleAndRobustness operations are enabled,
  => it reveals that the guard of **call** is not satisfied!

Vasco 4. Systematic test of the rules (3)

Permission rule	Positive tests	Negative tests
Intervention Perms	3	7
Patient Perms	3	7
Team Perms	5	10
Management Perms	3	6
TeamDoctorMA	4	12
TeamMemberMA	5	14
ParmAdviceMA	4	15
RegulatorInstructionMA	3	12
RegParmMAPerm	2	2
TeamMemberMAPerm	2	2
Total	**34**	**87**

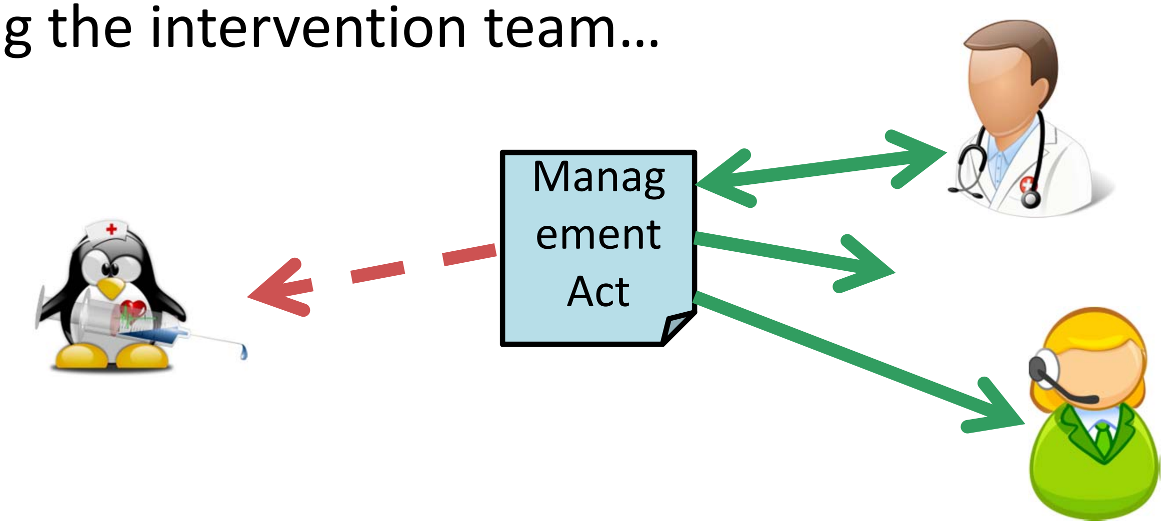
- The 10 permissions of our security model were tested by positive and negative test cases.
- Specifier and tester were distinct persons.
- Followed a TDD approach where test cases were written before the detailed security policy.

5. Attacks

- At this stage, we checked that:
  - The security rules don't prevent normal use
    (3. Animation of secure operations)
  - Each rule grants or denies access correctly
    (4. Systematic test of the rules)
- But, does the system prevent insider attacks,
  i.e. sequences of actions which would grant additional but undue rights to legitimate users?
- We don't have a tool to design such attacks. (on-going work)
- But, given an attack, we can test it!

5. Attacks

- For example, a nurse tries to get read access to confidential information by joining the intervention team…



- 13 attacks were tested
  (and 7 closely related nominal cases)
- Note that attacks are more complex than the previous tests, and had to be cut into smaller steps before being played with ProB.

Conclusion

- V&V of a PIM model of Secure Information System :
  - Proofs
  - Animation
  - Test
- 141 tests played against the Res@mu model
- Future work:
  - Automate the systematic generation of tests (combinatorial testing of roles and operations).
  - Automated synthesis of test cases and attacks using proof and model-checking techniques.

Questions?

- Photo Credits:
  - http://commons.wikimedia.org/wiki/File:Sala_de_Regulacion_del_Samu_de_Paris.jpg?uselang=fr
  - http://commons.wikimedia.org/wiki/File:Logo_Samu.gif?uselang=fr
  - http://commons.wikimedia.org/wiki/File:H%C3%B4pital_d%27Orl%C3%A9ans-la-Source_SAMU_1.jpg?uselang=fr
- Clipart from openclipart.org