QuomBat 2010

Language-specific vs. language-independent approaches: embedding semantics on a metamodel for testing and verifying access control policies

Yves Le Traon
Tejeddine Mouelhi
Franck Fleurey
Benoit Baudry

Context and motivations

Quality of a Metamodel for verification and testing purpose

- MDE approach vs. language specific one
- The today issue: Metamodel loses information (and semantics)
  - Loss : quality/abstraction ?
  - Compared with language-specific techniques
- Problem tackled in the context of (MODELS'08)
  - Access control policies (RBAC, OrBAC, MAC, DAC, …
  - Need to validate security policies
  - Technology independent qualification process

Outline

Using MDE for security

Access control meta-model

Verification checks of the model

Mutation testing for security

Case studies and results

Summary and conclusions

Metamodelling ???

MMFruit

Semantics

Is-not-rotten

check-colour

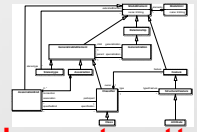
cut

MetaModel

Model

Instance

MBlueBerry



Is-not-rotten

cut

MApple

Is-not-rotten

cut

MBanana

Is-not-rotten

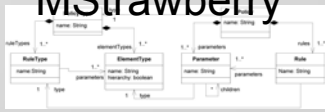
cut

MOrange

Is-not-rotten

cut

MStrawberry



Is-not-rotten

cut

Language-specific vs. language-independent approaches

Metamodelling ???

MetaModel

Model

Instance

MMFruit

Semantics

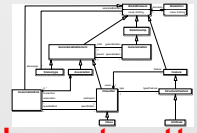
Is-not-rotten

seems-better(a,b)

cut

check-colour

MChicken



Is-not-rotten

cut

MRabbit

Is-not-rotten

cut

MLamb

Is-not-rotten

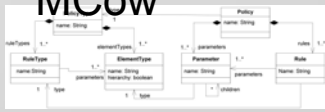
cut

MPig

Is-not-rotten

cut

MCow



Is-not-rotten

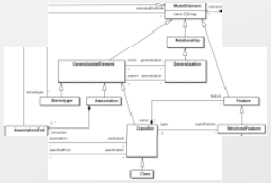
cut

Verifying and Testing Access Control Policies

MMSecurityPolicy + Semantics

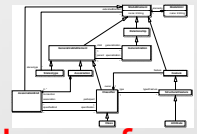
Is-conform() V&V
No-redundancies()
Mutate()

MetaModel



RBAC | MOrBAC | MDAC

Model



Is-conform()
Mutate()

Is-conform()
Mutate()

Is-conform()
Mutate()

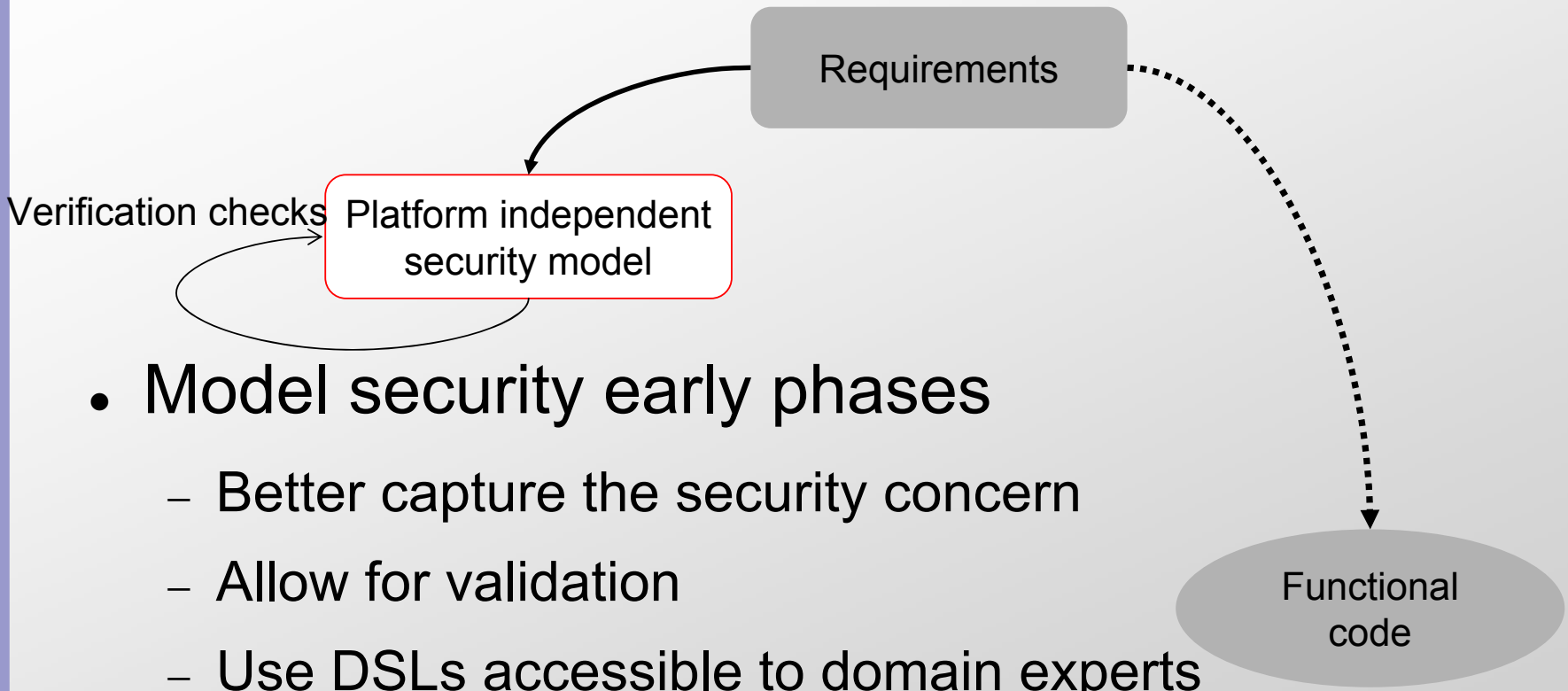
Instance

R1 -> UserRole(romain Student)
R2 -> UserRole(yves Director)
R3 -> UserRole(alice Secretary)
R4 -> RolePermission(Student BorrowBook WorkingDays)
R5 -> RolePermission(Personnel ModifyAccnt WorkingDays)
R6 -> RolePermission(Director CreateAccount AllTime)

R1 -> Permission(Library Student Borrow Book WorkingDays)
R2 -> Prohibition(Library Student Borrow Book Holidays)
R3 -> Prohibition(Library Secretary Borrow Book Default)
R4 -> Permission(Library Personnel ModifyAccount UserAccount WorkingDays)
R5 -> Permission(Library Director CreateAccount UserAccount WorkingDays)

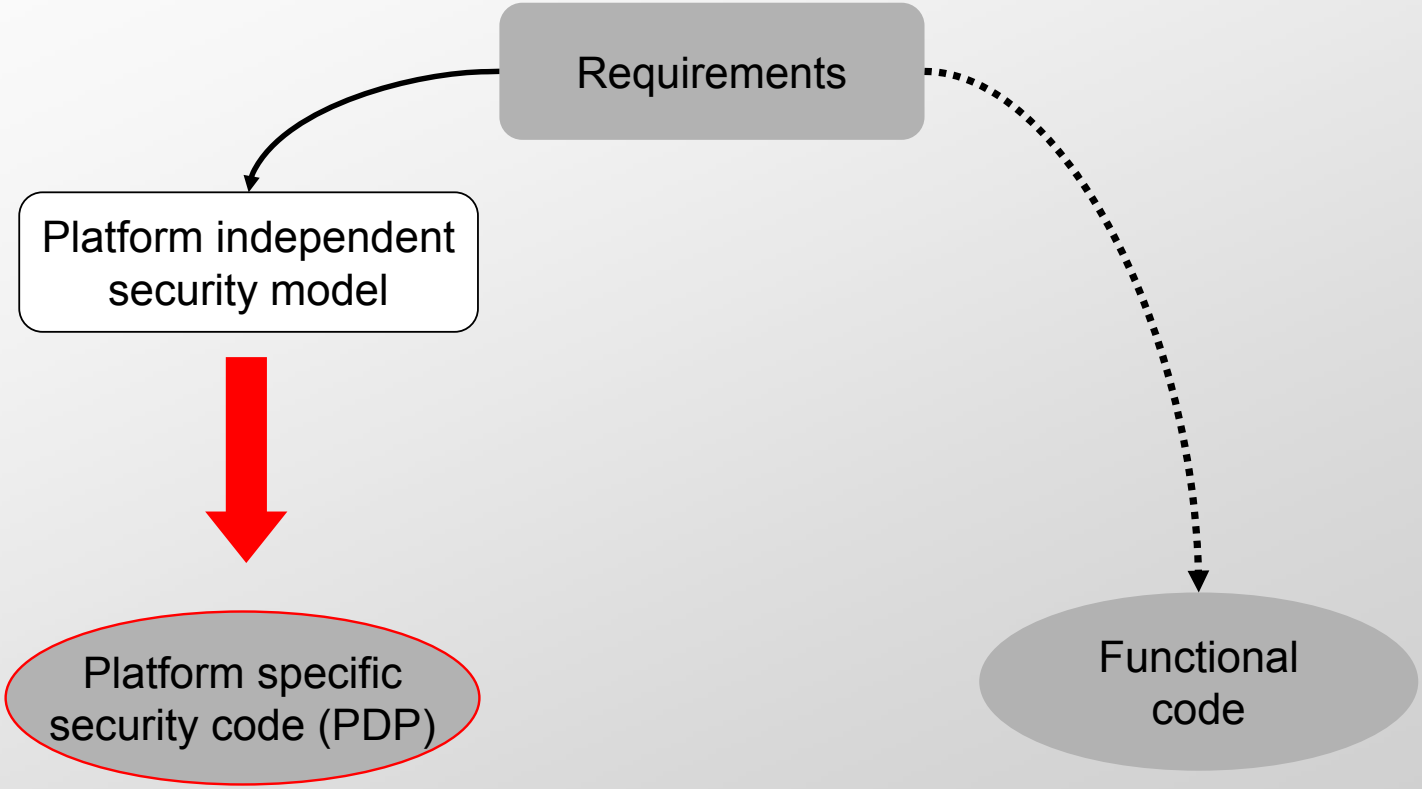
POLICY systemDAC (DAC)
R1 -> DACRule(Tim r file1)
R2 -> DACRule(Tim x file1)
R3 -> DACRule(Admin cp file1)
R4 -> DACRule(Admin r file1)
R5 -> DACRule(Admin w file1)
R6 -> DACRule(Admin x file1)

Overview of the approach (1)



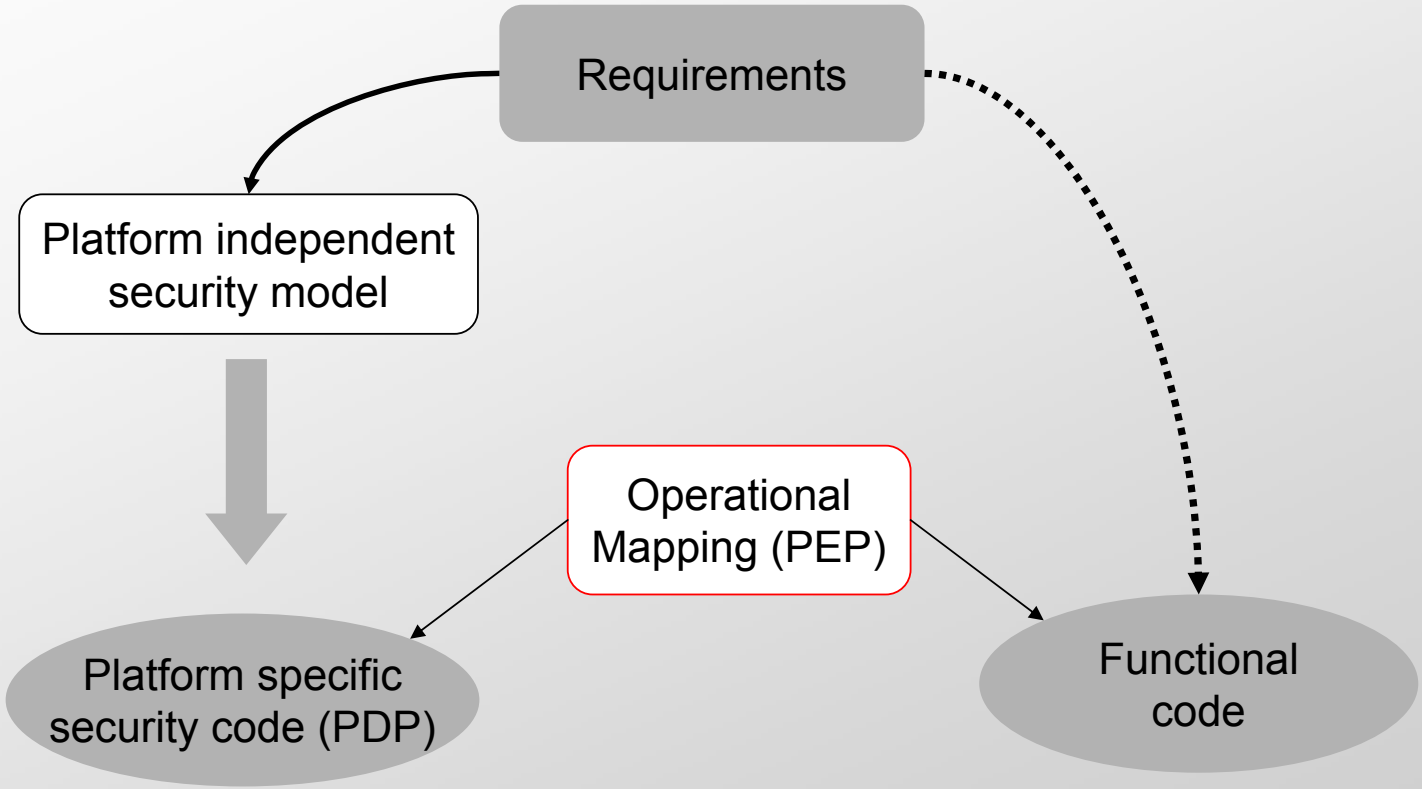
- Model security early phases
  - Better capture the security concern
  - Allow for validation
  - Use DSLs accessible to domain experts
  - Independent of the security platform
- Few assumptions on the functional code

Overview of the approach (2)



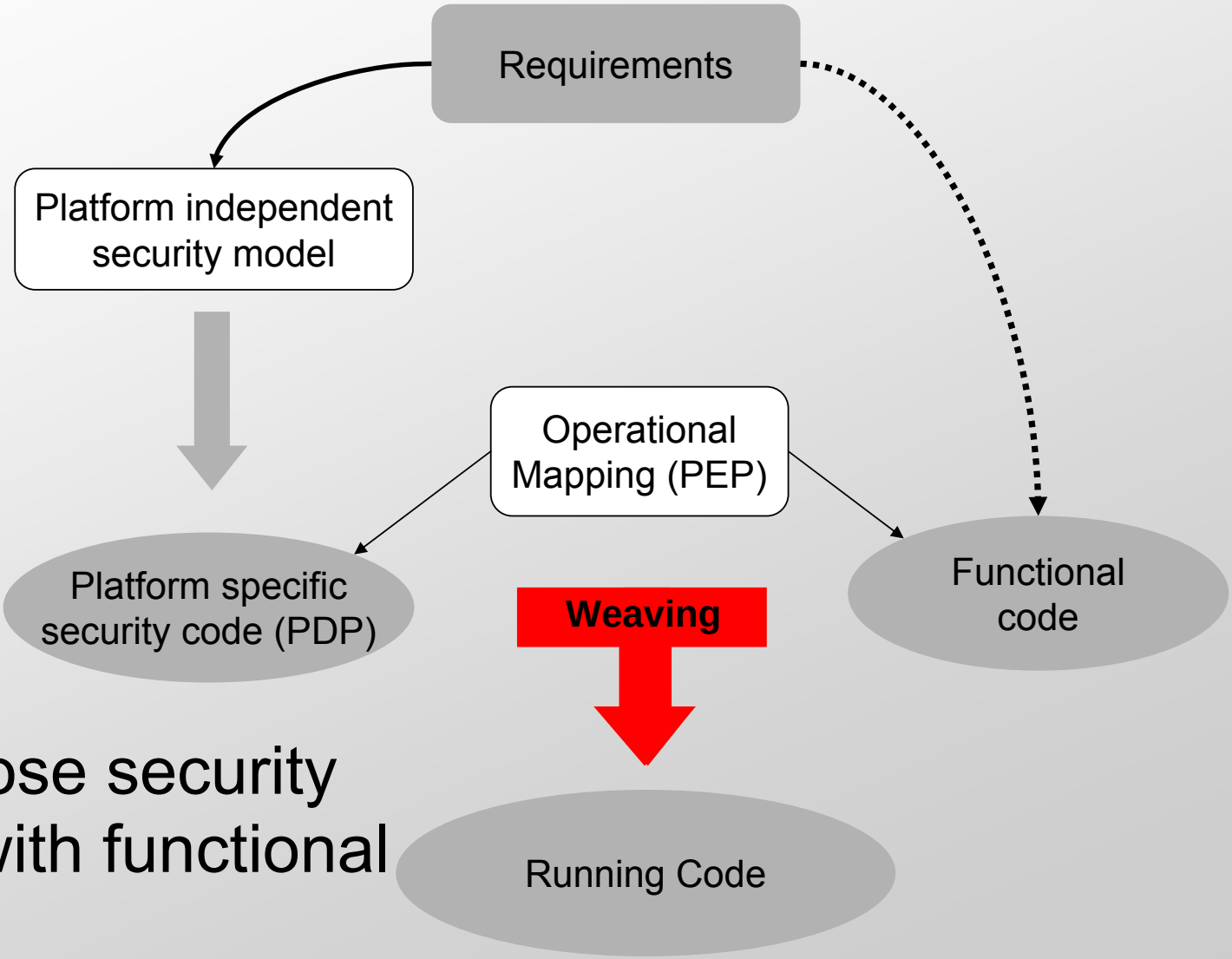
- Generate platform specific security policy
  - Using model transformation / code generation

Overview of the approach (3)



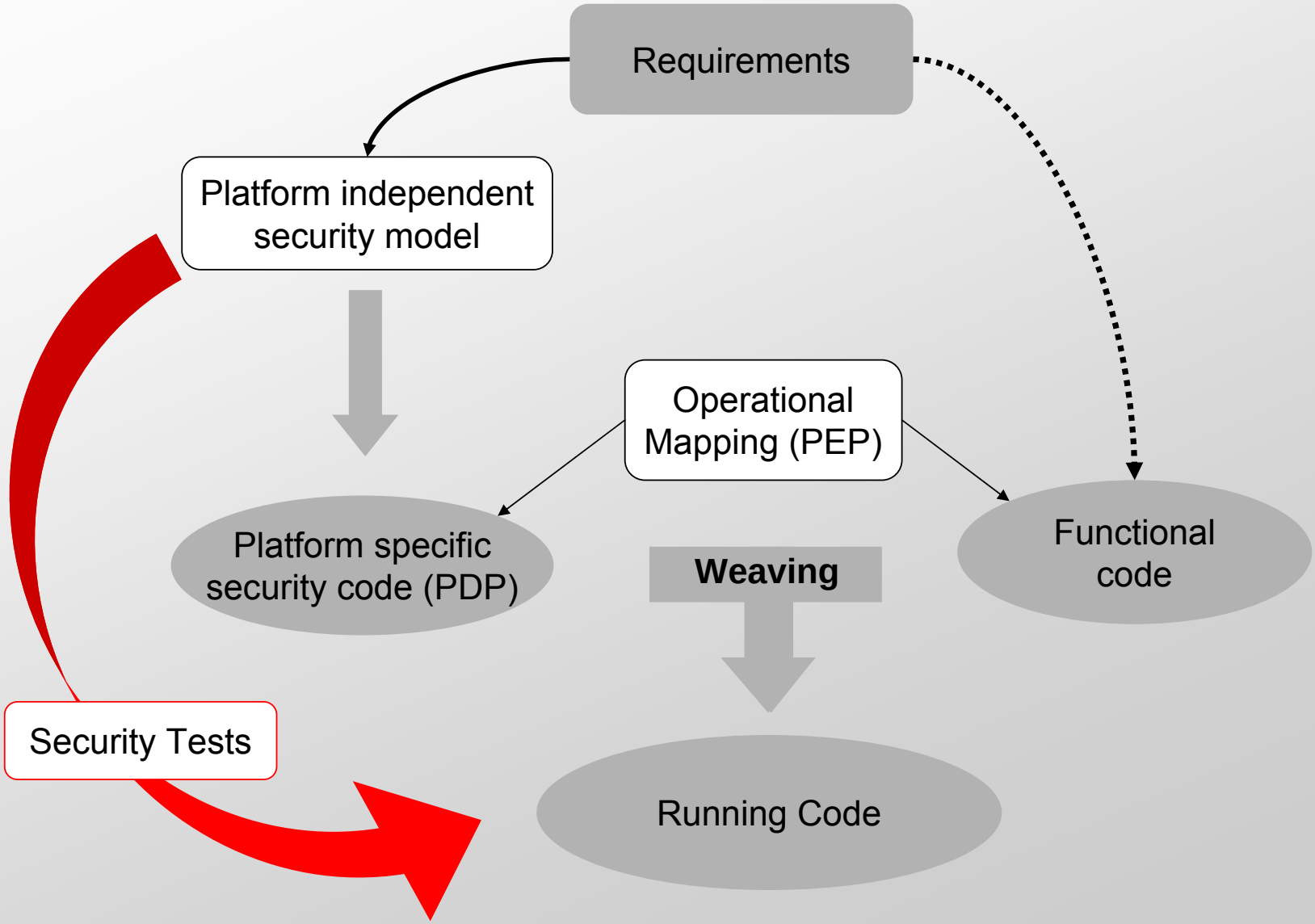
- Provide operational mapping

Overview of the approach (4)

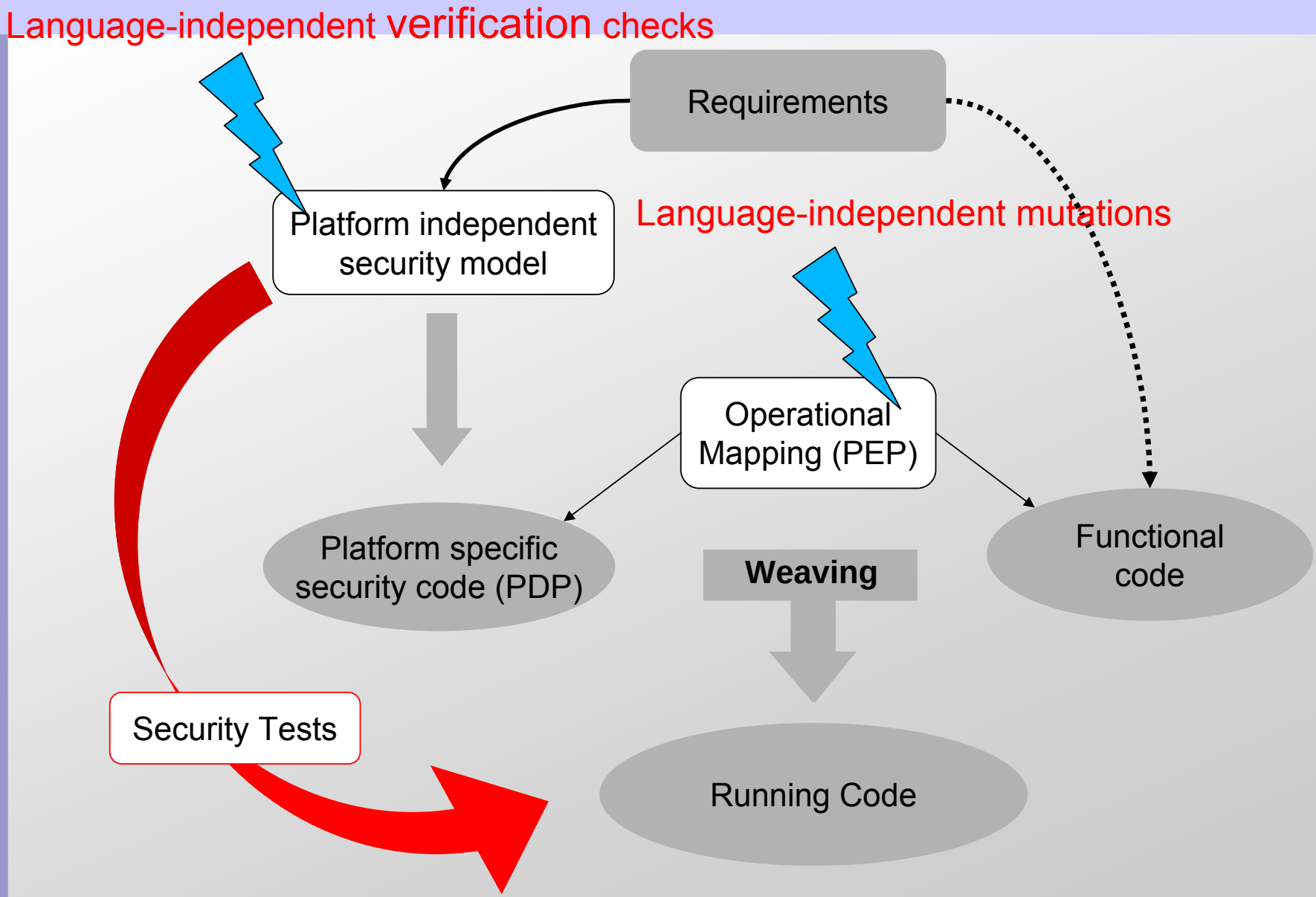


- Compose security code with functional code

Overview of the approach (5)



Overview of the approach (5)



Outline

Using MDE for security

Access control meta-model

Verification checks of the model

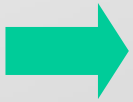
Mutation testing for security

Case studies and results

Summary and conclusions

Modeling access control policies

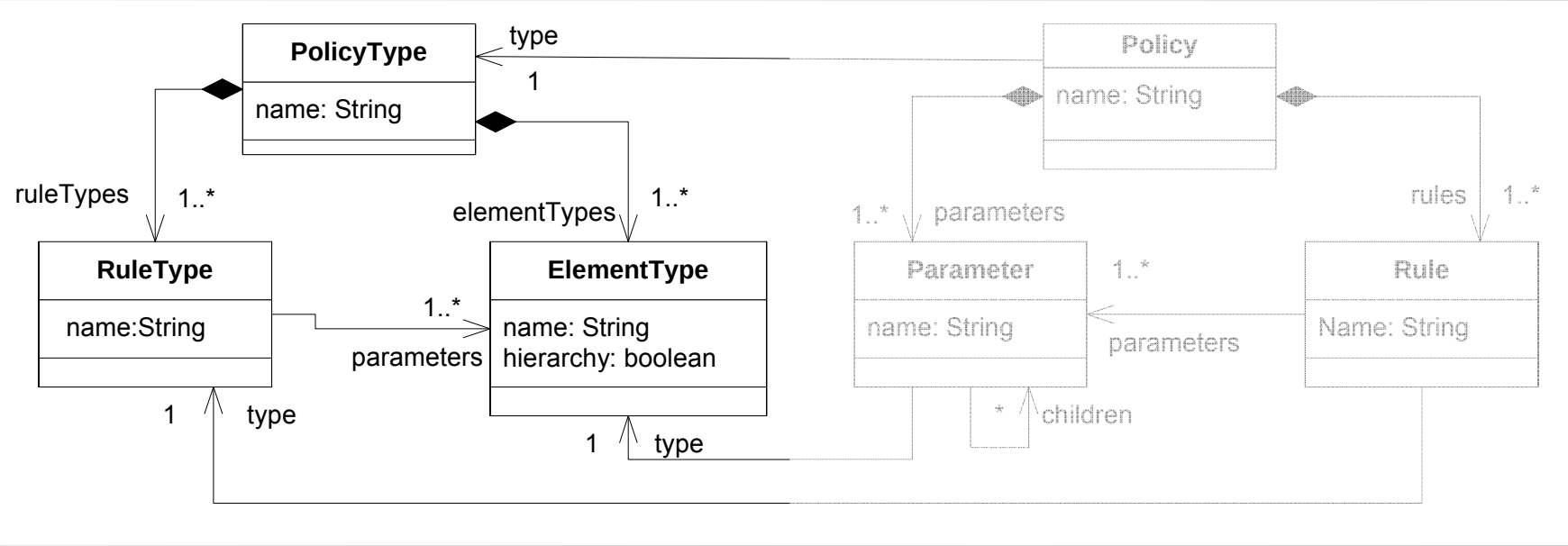
- Most formalisms are rule based
  - Different types of rules
  - Different types of parameters
- Examples
  - OrBAC
    - R1 - Permission (Library Student Borrow Book WorkingDays)
    - R2 - Prohibition (Library Student Borrow Book Holidays)
  - RBAC
    - R3 – UserRole (Alice Student)
    - R4 – RolePermission (Student BorrowBook WorkingDays)



Need to model both rules and rule types

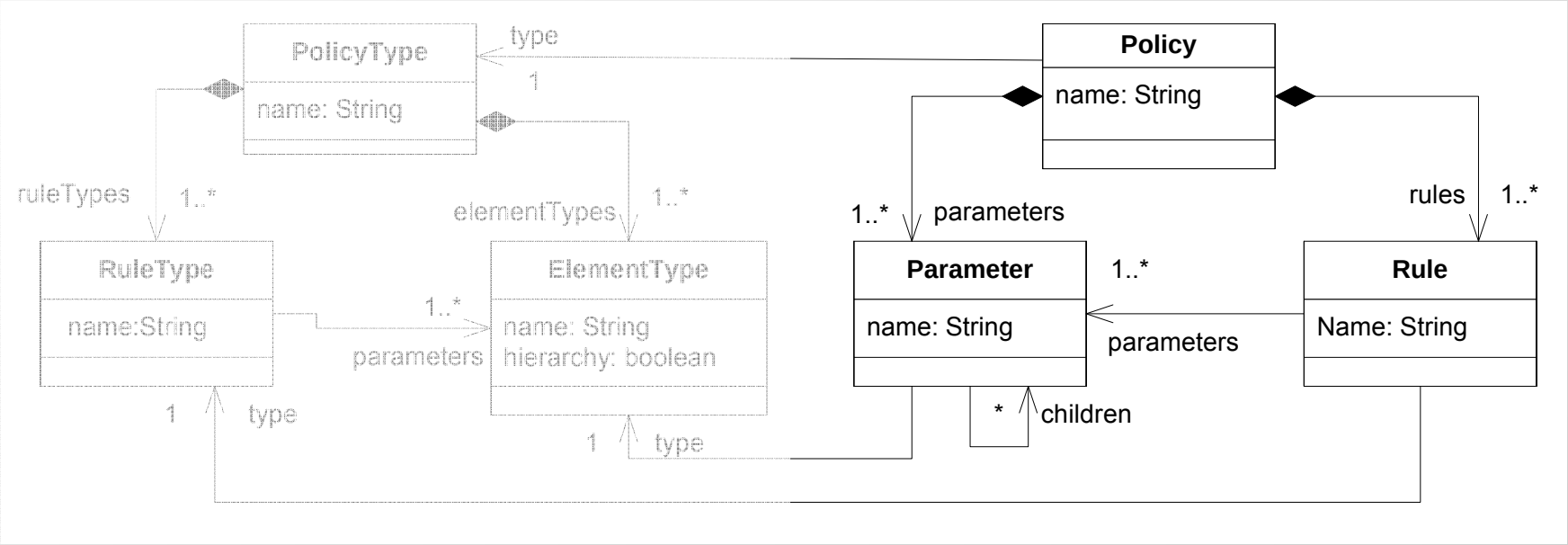
Modeling rule types (1)

- Define elements types
- Define rules types



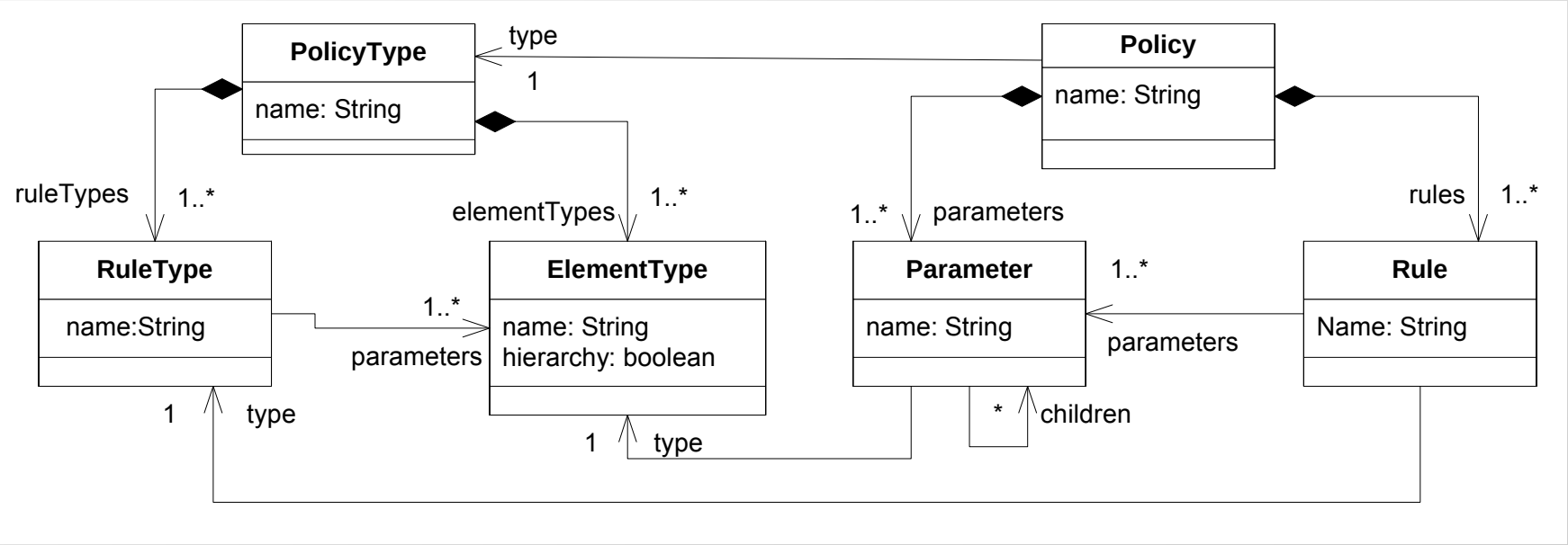
Modeling rules (1)

- Model rules and their parameters



Modeling rules (1)

- Model rules and their parameters



Outline

Using MDE for security

Access control meta-model

Verification checks of the model

Mutation testing for security

Case studies and results

Summary and conclusions

Simple verification checks

- no_conflicts(): checks the absence of conflicts. It essentially involves checking that there are no rules having the same parameters and having different types.
- no_redundancies(): checks that the security policy is minimal, which means that no rule appears more than once.

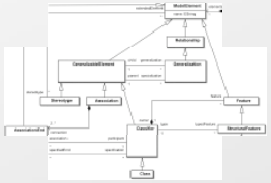
Verification checks

Some checks are not relevant for all access control languages: no_conflict

MMSecurityPolicy + Semantics

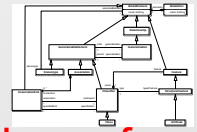
Is-conform() **V&V**
No-redundancies()
Mutate()

MetaModel



Model

HRBAC



Is-conform()
Mutate()

MOrBAC

Is-conform()
Mutate()

MDAC

Is-conform()
Mutate()

Instance

R1 -> UserRole(romain Student)
R2 -> UserRole(yves Director)
R3 -> UserRole(alice Secretary)
R4 -> RolePermission(Student BorrowBook WorkingDays)
R5 -> RolePermission(Personnel ModifyAccnt WorkingDays)
R6 -> RolePermission(Director CreateAccount AllTime)

R1 -> Permission(Library Student Borrow Book WorkingDays)
R2 -> Prohibition(Library Student Borrow Book Holidays)
R3 -> Prohibition(Library Secretary Borrow Book Default)
R4 -> Permission(Library Personnel ModifyAccount UserAccount WorkingDays)
R5 -> Permission(Library Director CreateAccount UserAccount WorkingDays)

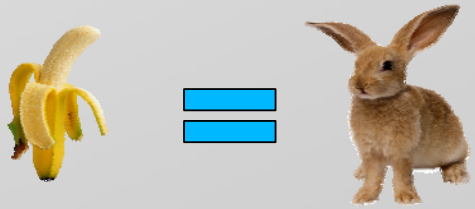
POLICY systemDAC (DAC)
R1 -> DACRule(Tim r file1)
R2 -> DACRule(Tim x file1)
R3 -> DACRule(Admin cp file1)
R4 -> DACRule(Admin r file1)
R5 -> DACRule(Admin w file1)
R6 -> DACRule(Admin x file1)

Language-specific vs. language-independent approaches

Advantages and limitations

- It works
  - Same verification for any rule-based access control languages

- But
  - The verification checks
    - Are very rough **Is-not-rotten**
    - Need complementary specific checks check-colour
  - Not specific to access control
    - rule-based system
    - Too generic ?



Disappointing ?

Outline

Using MDE for security

Access control meta-model

Verification checks of the model

Mutation testing for security

Case studies and results

Summary and conclusions

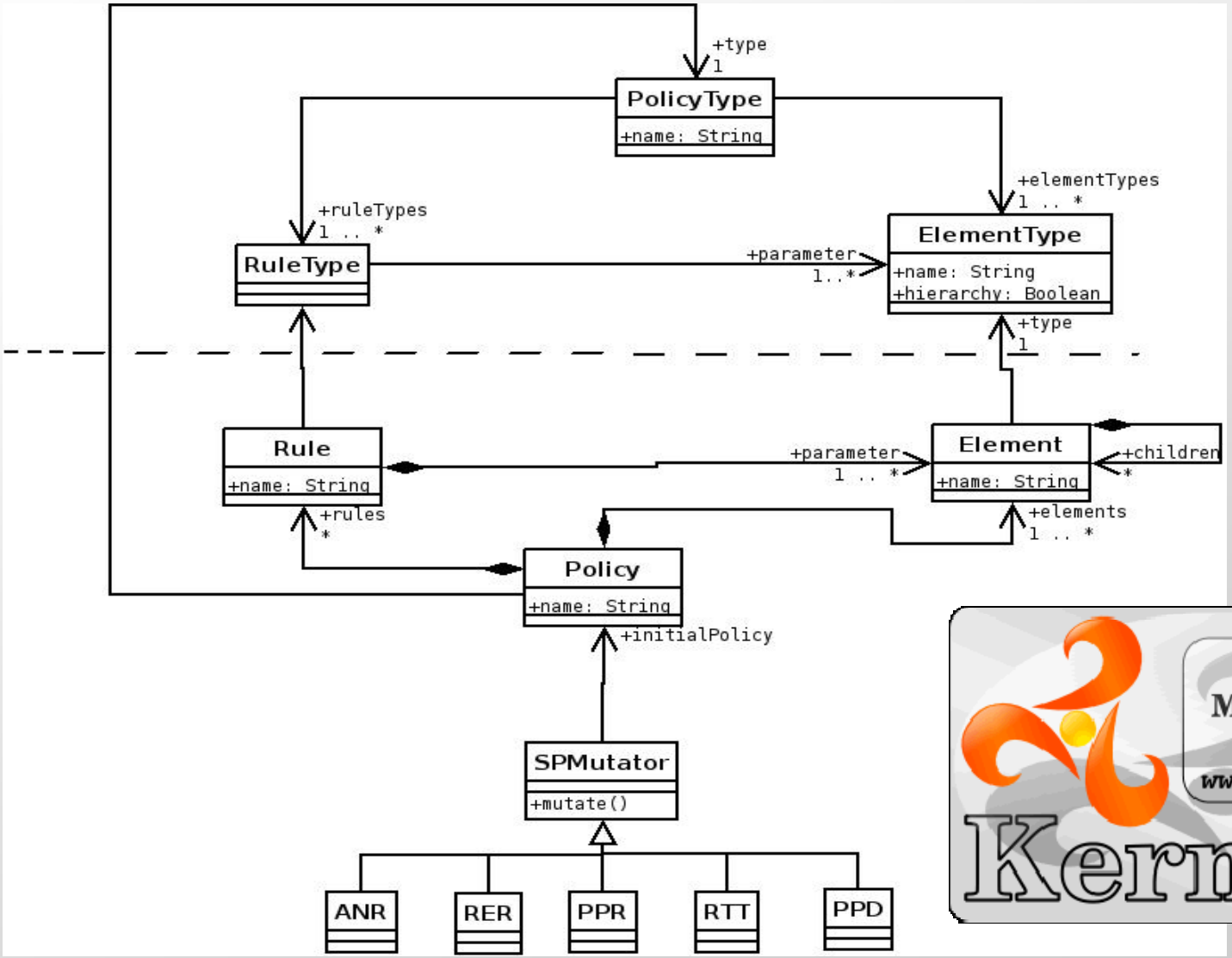
Mutation analysis: certifying tests with a common framework

- Objective
  - Qualify tests with respect to security
- Principle
  - Inject faults in the security model
  - Check that the tests can catch faults
- Defining faults models at a generic level
- Defining the analysis at the generic level
  - The analysis can be performed on any policy defined according to any policy model

Mutation operators

- 5 Generic operators
  - ANR → new rule
  - RER → Remove existing rule
  - PPR → Replace rule parameter
  - RTT → Modify rule type
  - PPD → A parameter replaced by a descendant

Operators in the Metamodel



Implementation of RER

```
/** Removes an existing rule */
class RER inherits SPMutator {
    method mutate(p : Policy) : set Policy[*] is do
     var mutant : Policy
     result := Set<Policy>.new
     p.rules.each{ r |
          // create mutated policy
          mutant := p.copy
          mutant.name := p.name + "-RER-" + r.name
          mutant.rules.remove(mutant.rules.detect{x | x.name == r.name})
          result.add(mutant)
     }
     end
}
```

Outline

Using MDE for security

Access control meta-model

Verification checks of the model

Mutation testing for security

Case studies and results

Summary and conclusions

Empirical validation

- Two questions
  - Is the approach feasible / practical ?
  - Are the generic mutation operator meaningful ?

- Three case studies

	# classes	# methods	LOC (executable statements)
LMS	**62**	**335**	**3204**
VMS	**134**	**581**	**6077**
ASMS	**122**	**797**	**10703**

Results

- Generic mutation operator implementation
- Generate more mutants
- Includes all specific mutants

System	generic mutants	specific mutants
LMS	1044	371
VMS	1572	1426
ASMS	3088	2056

Mutation scores generic/specific

Nominal test cases

Mutants	Basic Mutants (func. Tests)		
System	LMS	VMS	ASMS
Generic mutants	72%	61%	45%
Specific mutants	78%	69%	55%
Delta	-6%	-8%	-10%

Robustness test cases

Mutants	ANR mutants (sec. tests)		
System	LMS	VMS	ASMS
Generic mutants	13%	12%	28%
Specific mutants	17%	19%	33%
Delta	-4%	-7%	-4%

Outline

Using MDE for security

Access control meta-model

Generation of security components

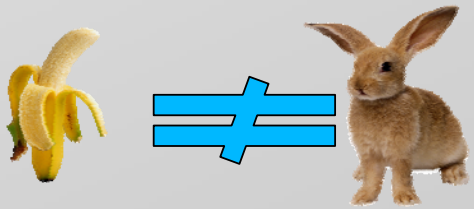
Mutation testing for security

Case studies and results

Summary and conclusions

Summary and Conclusion: what is a ‘good meta-model’ for V&V ?

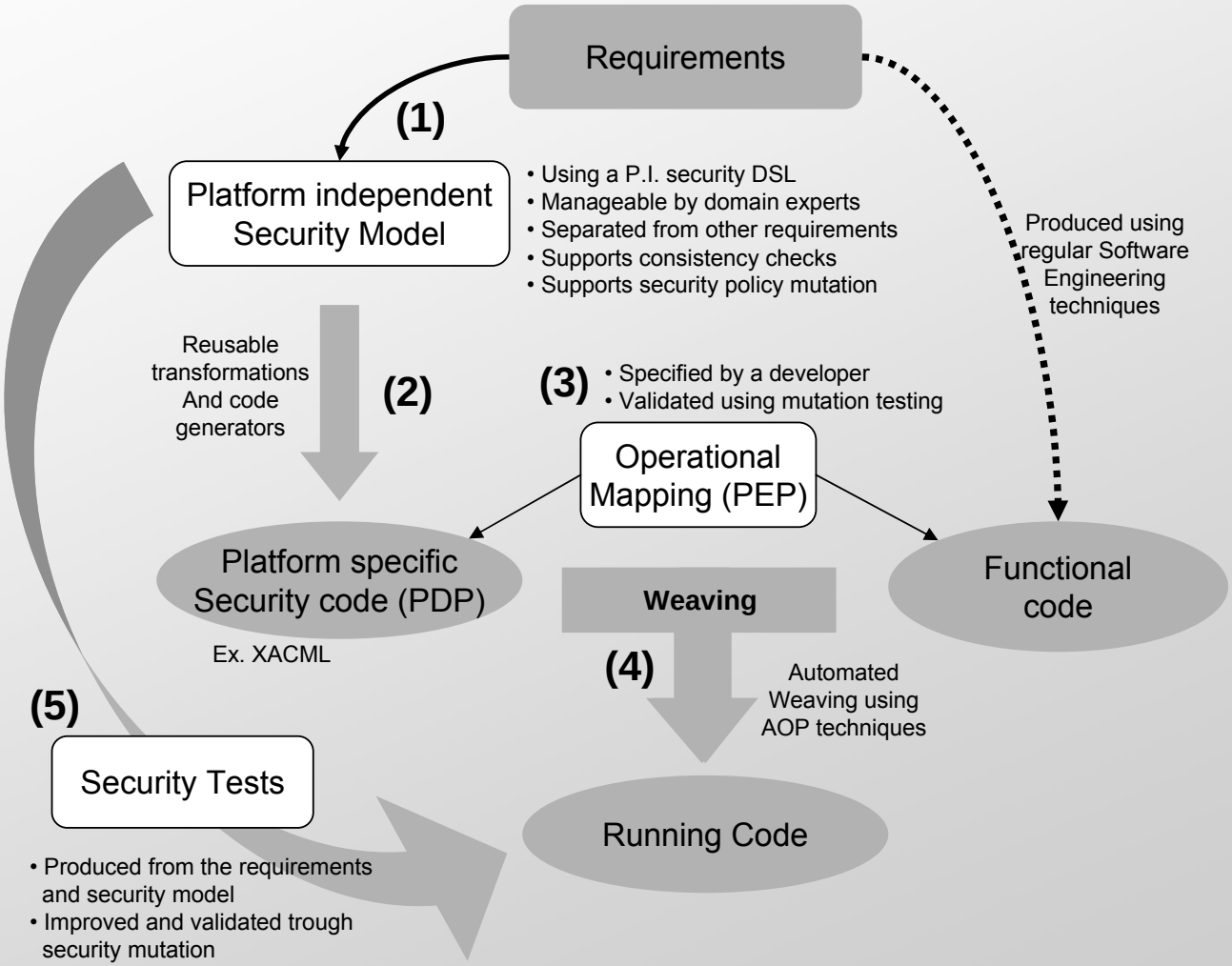
- Generic meta-model for Access Control Policies
  - Platform independent (RBAC,HRBAC, OrBAC, MAC, DAC)
- Certification process defined at MM
  - verification
  - mutation testing
- Open questions
  - A specific “language-independent” metamodel
  - Metamodels Composition



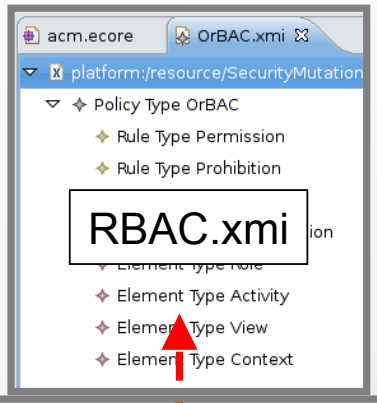
Thank you !

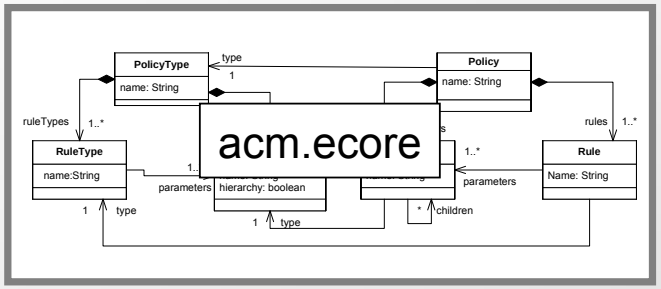
Questions ?

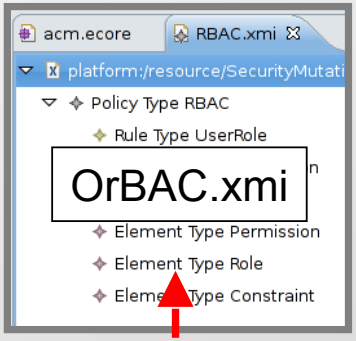
Overview of the approach

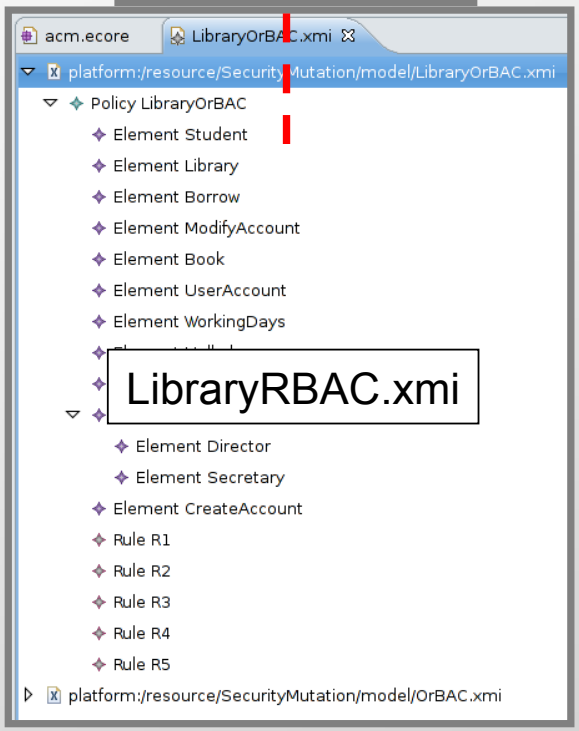


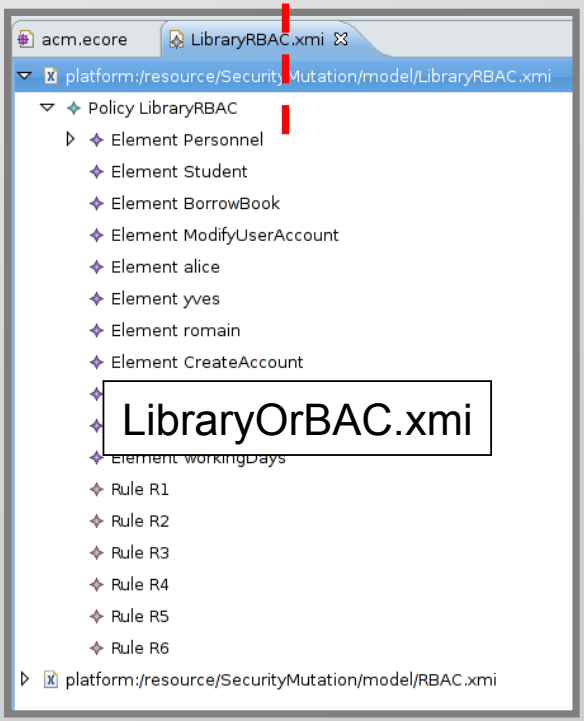
Implementation in EMF











Outline

Using MDE for security

Access control meta-model

Generation of security components

Mutation testing for security

Case studies and results

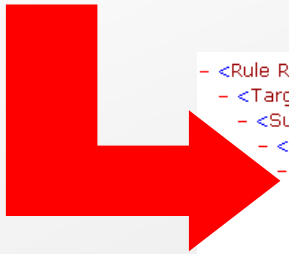
Summary and conclusions

Generation of security components

- Generate the Platform specific security code (PDP)
  - Process the policy model
  - Target existing technologies such as XACML
  - Typical MDE model to code
- Link the security code with the application (PEP)
  - No assumptions on the way the application is developed
  - Needs to be systematic
  - Needs to be automated

Generation of the PDP

Rule R1 - Permission (Library Student Borrow Book WorkingDays)



```
- <Rule RuleId="R1" Effect="Permit">
  - <Target>
    - <Subjects>
      - <Subject>
        - <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">BORROWER</AttributeValue>
            <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
              DataType="http://www.w3.org/2001/XMLSchema#string" />
          </SubjectMatch>
        </Subject>
      </Subjects>
    - <Resources>
      - <Resource>
        - <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">BOOK</AttributeValue>
            <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
              DataType="http://www.w3.org/2001/XMLSchema#string" />
          </ResourceMatch>
        </Resource>
      </Resources>
    - <Actions>
      - <Action>
        - <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">BORROW</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
              DataType="http://www.w3.org/2001/XMLSchema#string" />
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  - <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <EnvironmentAttributeDesignator AttributeId="current-context" DataType="http://www.w3.org/2001/XMLSchema#string"
          Issuer="admin" />
      </Apply>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">WORKINGDAYS</AttributeValue>
    </Condition>
</Rule>
```

Weaving in the application

- Using AspectJ
- Example :

```
pointcut borrowBookCall(User user,Book book) :
        target(BookService) && call(void borrowBook(User,Book)) &&  args(user,book);


before(User user,Book book) throws
    SecuritPolicyViolationException : borrowBookPC(user,book)  {

        // call security policy service to check for security rule
        Utils.checkSecurity(user, LibrarySecurityModel.BORROWBOOK_METHOD ,
            LibrarySecurityModel.BOOK_VIEW,
                ContextManager.getTemporalContext());

    }
```

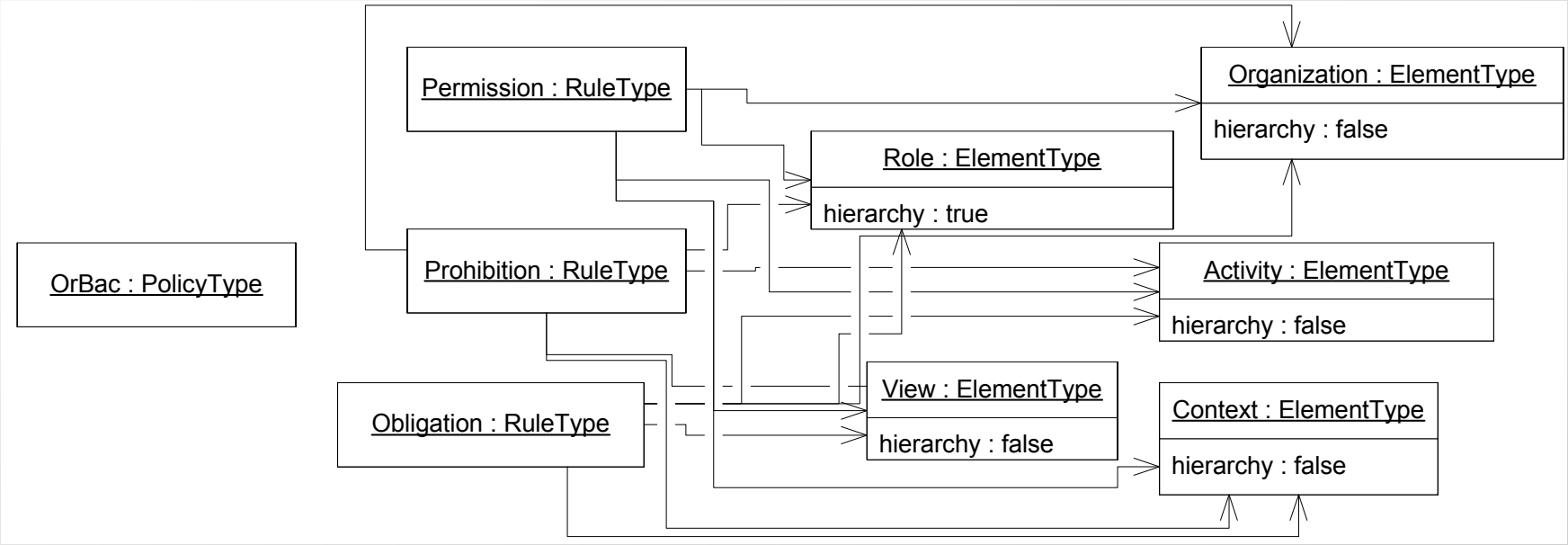
Verification checks

- Some checks are not relevant for all access control languages

	Policy_ is_conform	No_ conflicts	No_ redundancies
RBAC	y	n	y
HRBAC	y	n	n
OrBAC	y	y	y
DAC	y	n	n
MAC	y	n	n

Modeling rule types (2)

- Example: OrBAC Model



Modeling rules (2)

- Example

R1 - Permission (Library Student Borrow Book WorkingDays)