InSTA 2016, Chicago, Apr 10, 2016

TESEM A Tool for Verifying Security Design Pattern Applications

Hironori Washizaki

Waseda University, Tokyo, Japan

Collaborators: Takanori Kobashi, Masatoshi Yoshizawa, Yoshiaki Fukazawa (Waseda University)

Takao Okubo (Institute of Information Security)

Haruhiko Kaiya (Kanagawa University)

Nobukazu Yoshioka (National Institute of informatics)

“Hiro”nori Washizaki

- Prof., Head, Global Software Engineering Laboratory, Waseda University
- Visiting Assoc. Prof., National Institute of Informatics
- Chair, IEEE CS Japan Chapter
- Chair, SEMAT Japan Chapter
- Convenor, ISO/IEC/JTC1/SC7/WG20
- Co-Chair, IEEE ICST’17 Toyo

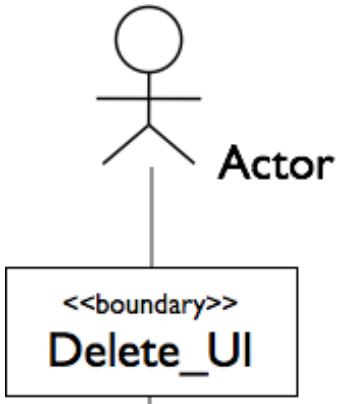
Waseda University

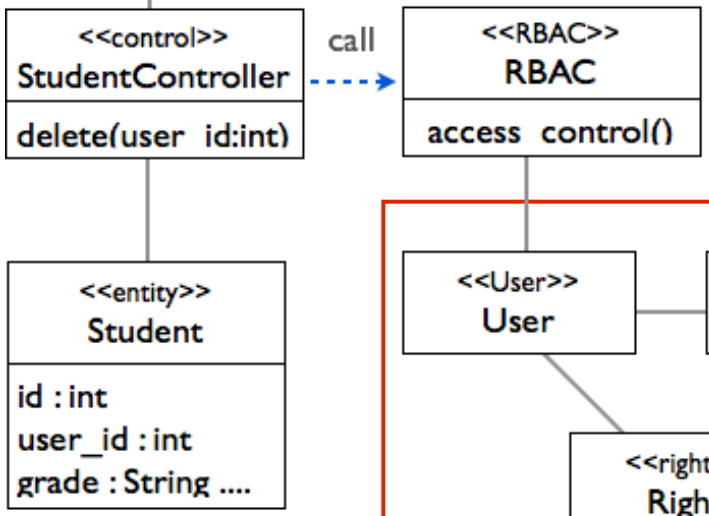
NII
National Institute of Informatics

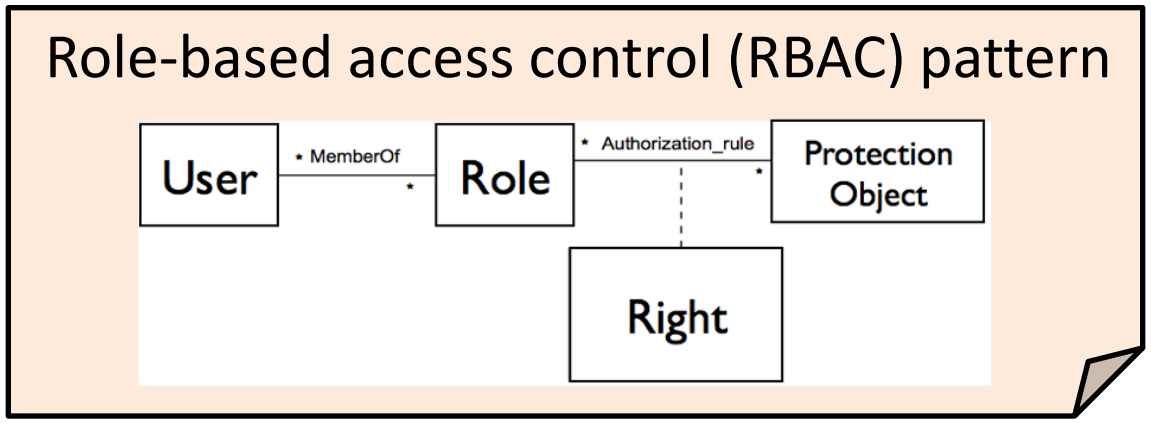
Agenda

- Introduction
- Security patterns
- TESEM: testing models
- TESEM: testing code
- Conclusion and discussion

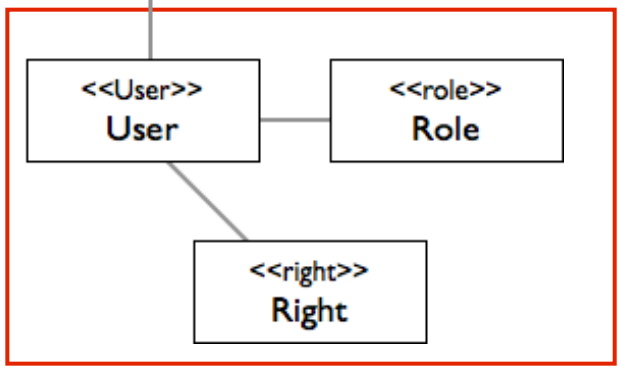
What's the problem?



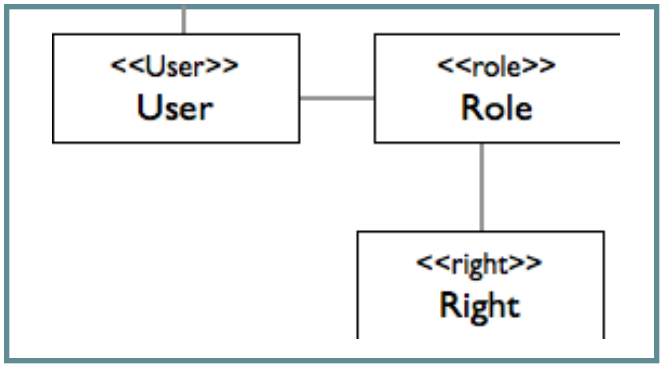




Inappropriate design

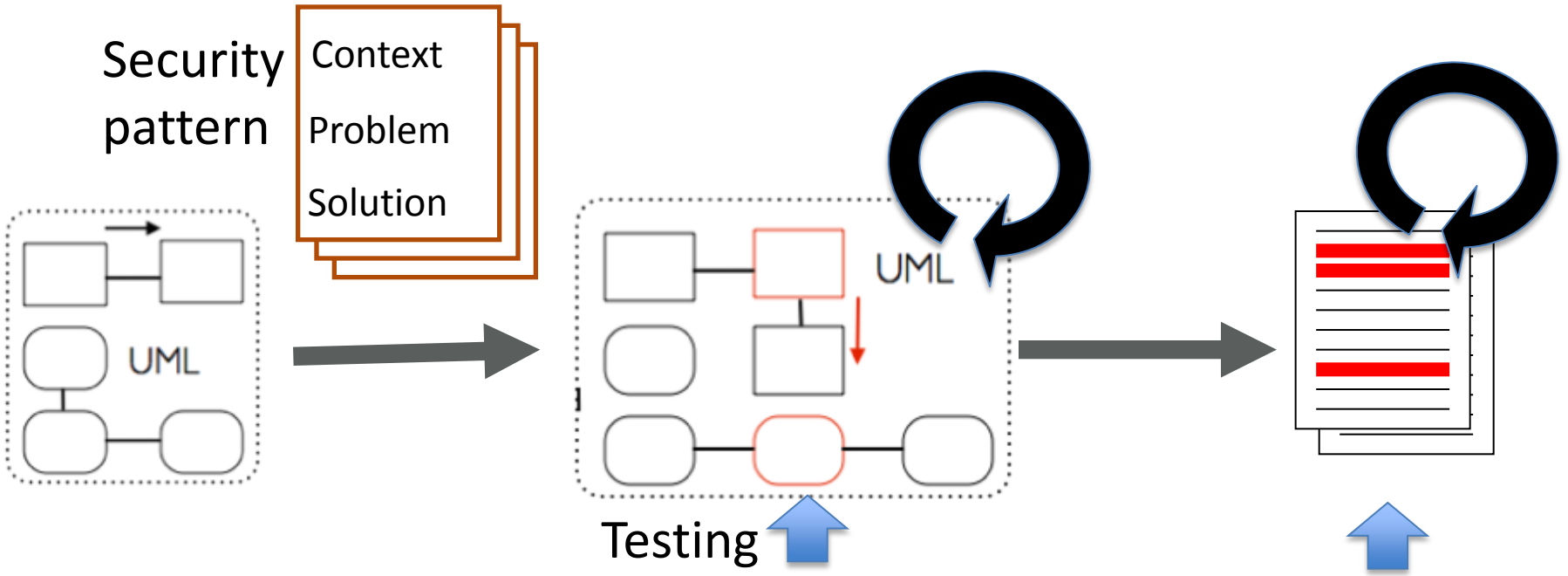


Appropriate design



TESEM: Test Driven Secure Modeling Tool

- Security design/implementation guided by testing pattern applications [ARES'13][ARES'14][IJSSE'14][ICST'15]



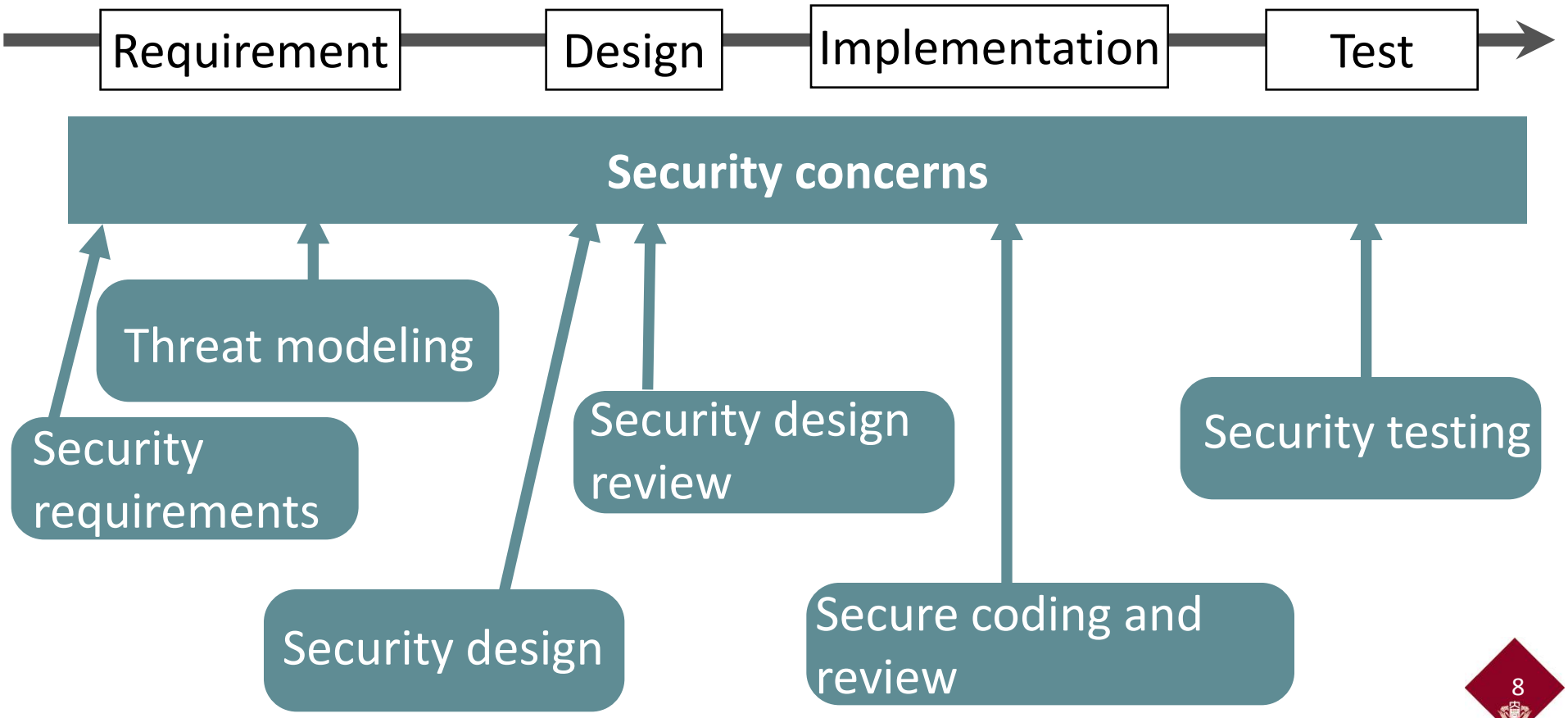
[ARES'13] Validating Security Design Pattern Applications Using Model Testing, Int'l Conf. Availability, Reliability and Security
[ARES'14] Verification of Implementing Security Design Patterns Using a Test Template, Int'l Conf. Availability, Reliability and Security
[IJSSE'14] Validating Security Design Pattern Applications by Testing Design Models, Int'l J. Secure Software Engineering 5(4)
[ICST'15] TESEM: A Tool for Verifying Security Design Pattern Applications by Model Testing, IEEE ICST'15 Tools Track

Agenda

- Introduction
- Security patterns
- TESEM: testing models
- TESEM: testing code
- Conclusion and discussion

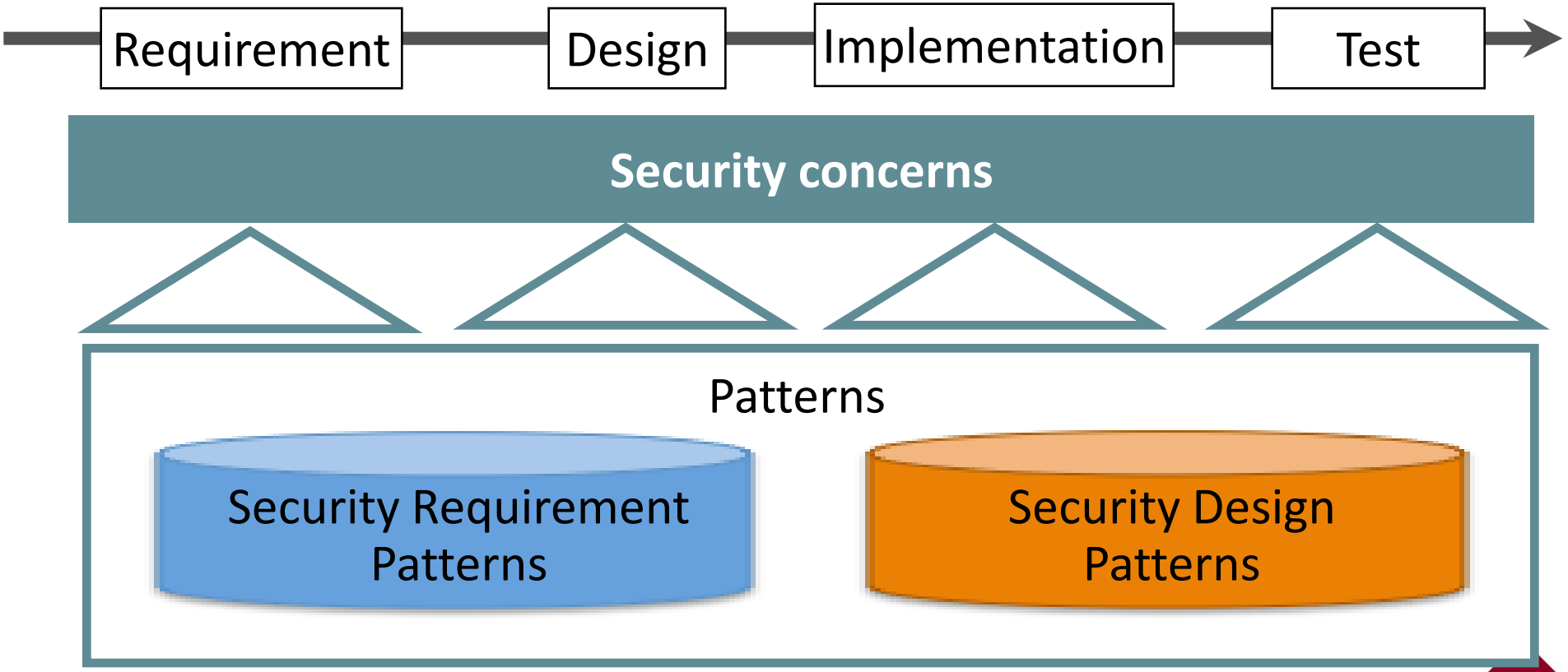
Security for every phase

- Security concerns must be addressed at every phase



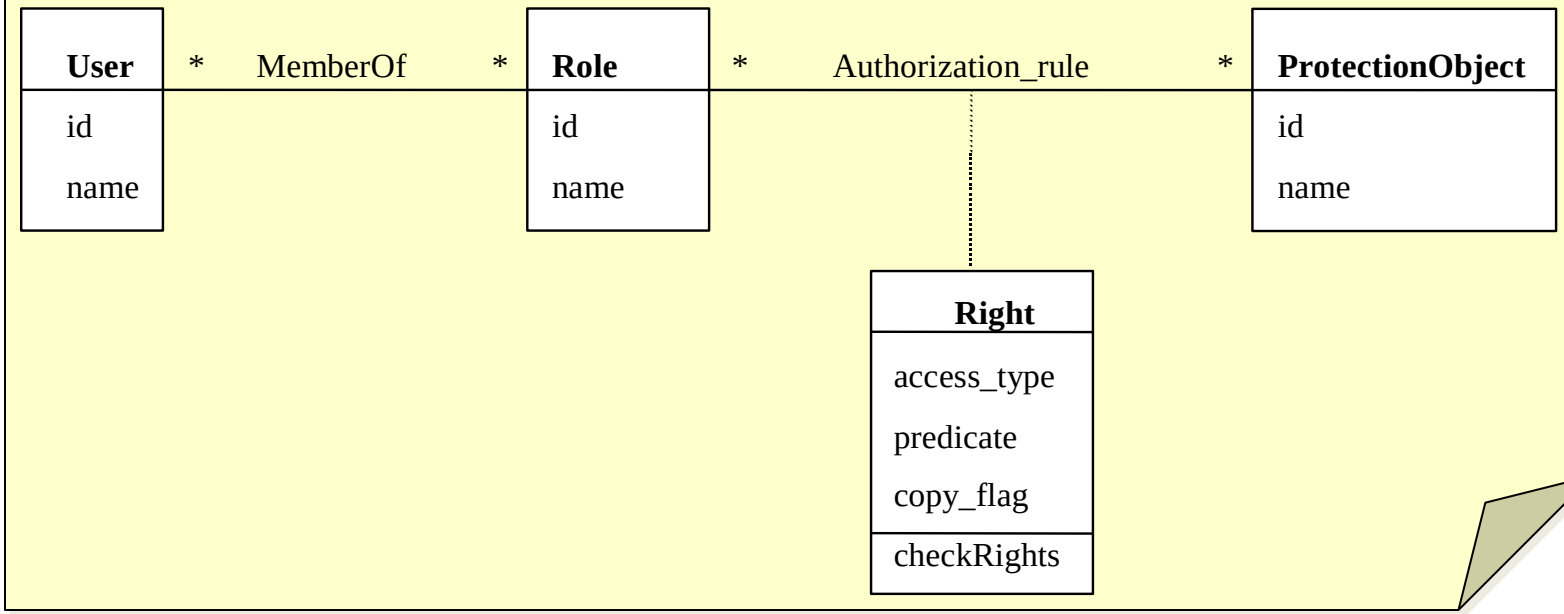
Patterns are promising at any phase

- Recurrent problems and solutions under specific contexts
- For requirements definition, design, implementation and testing!

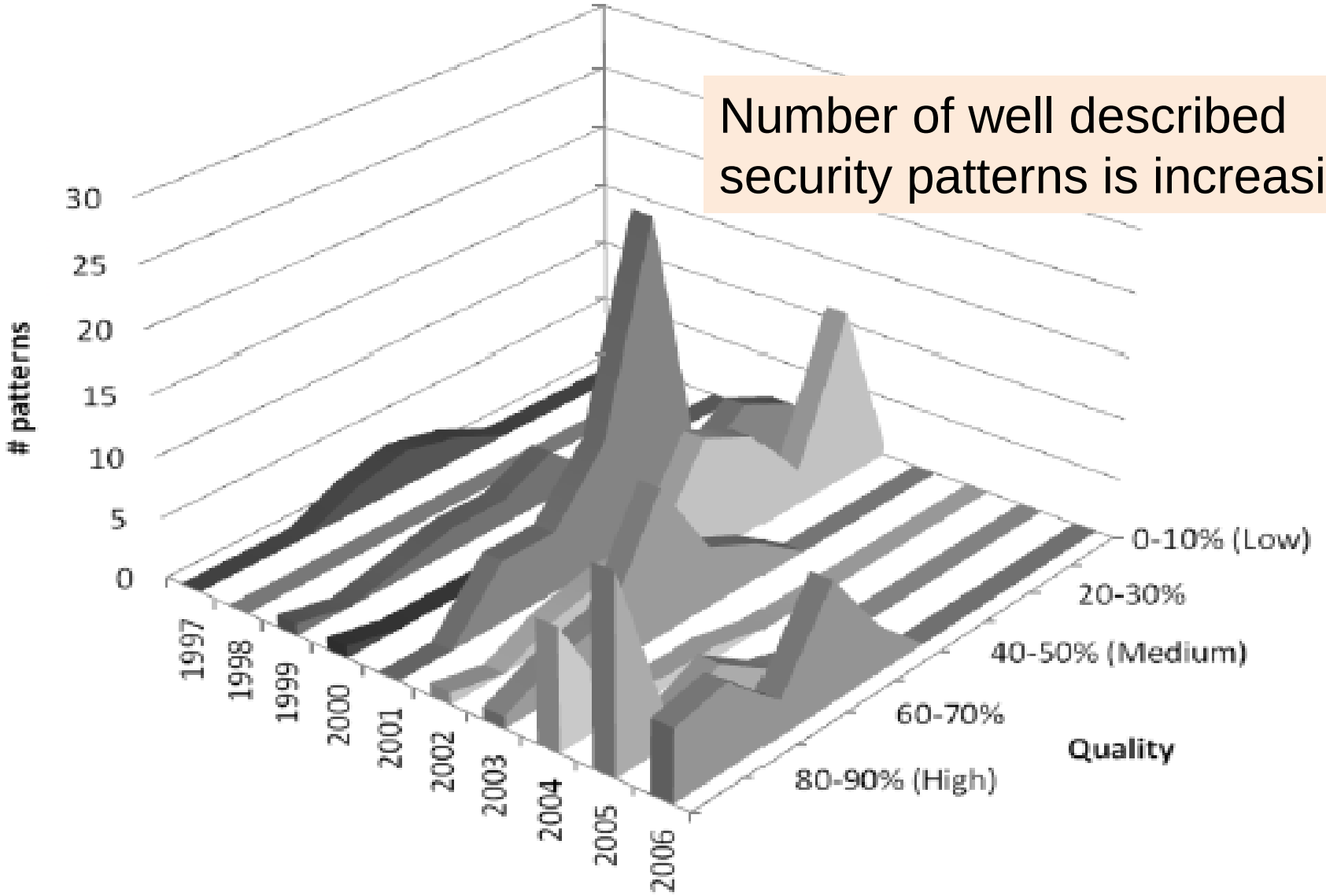


Example of security pattern

- Name: ***Role-based access control (RBAC)***
- Problem: How do we assign rights to people based on their functions or tasks?
- Solution: Assign users to roles and give rights to these roles so they can perform their tasks.
- Related patterns: ***Authorization,*** . . .



Security patterns landscape [Heyman'07]

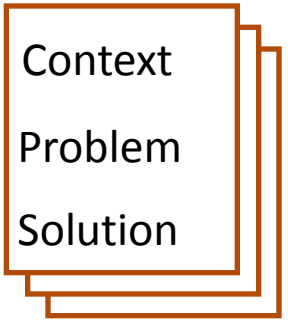


T. Heyman, "An Analysis of the Security Patterns Landscape," 2007

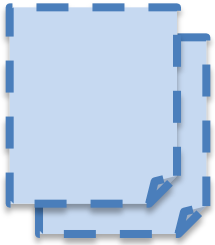
Pattern-oriented test architecture

- Security by proven patterns
  - Security requirements, secure design and implementation
- Patterns as abstract test cases
  - Possible to prepare abstract “constraints” and “templates” for testing model and code
  - Necessary to concretize patterns against concrete requirement/design/code

Security patterns

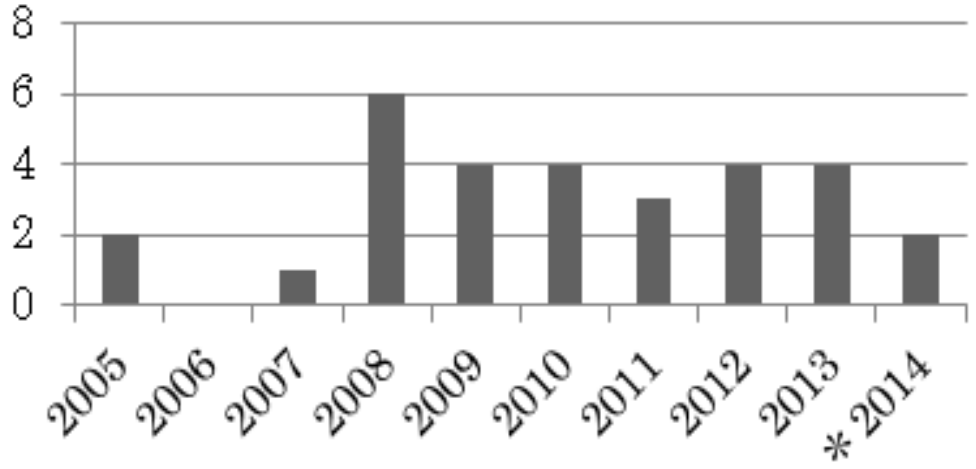


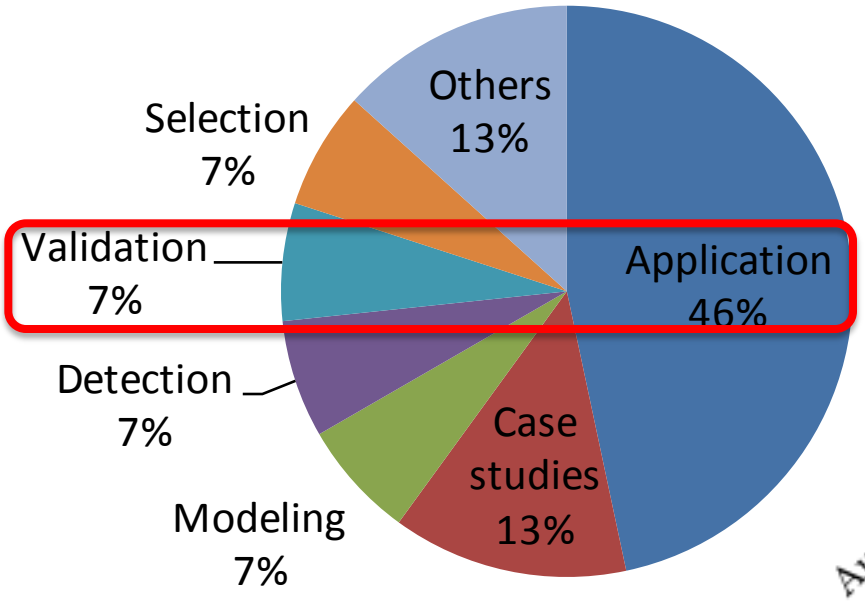
Abstract constraints, templates

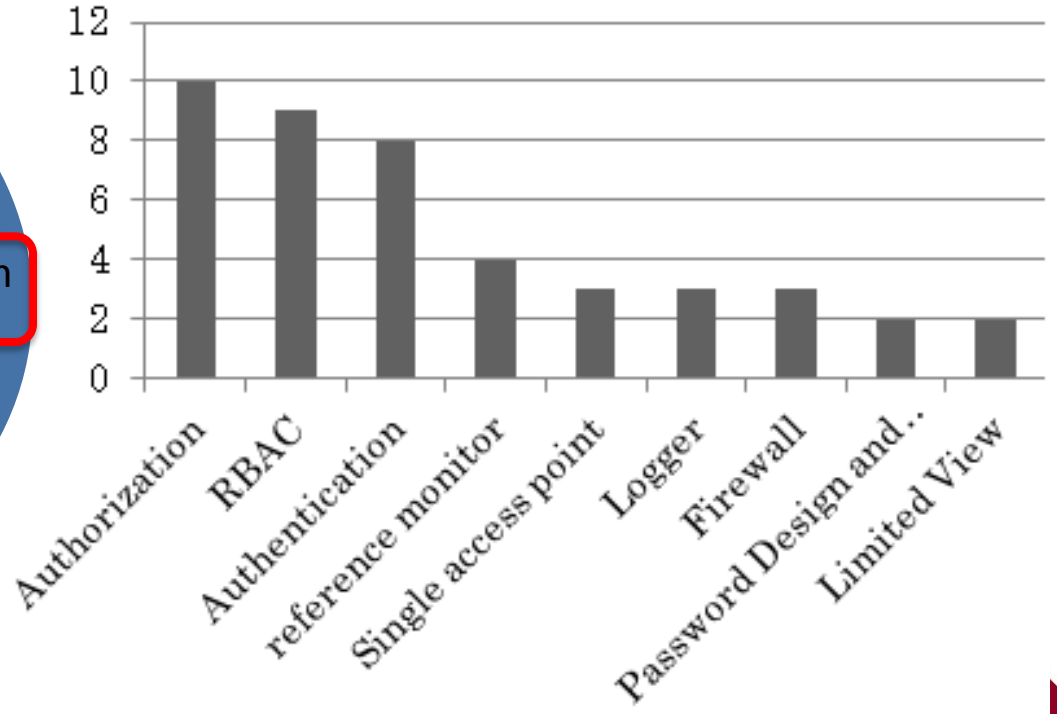


Concrete constraints, test cases

Security pattern researches [PLoP'15]







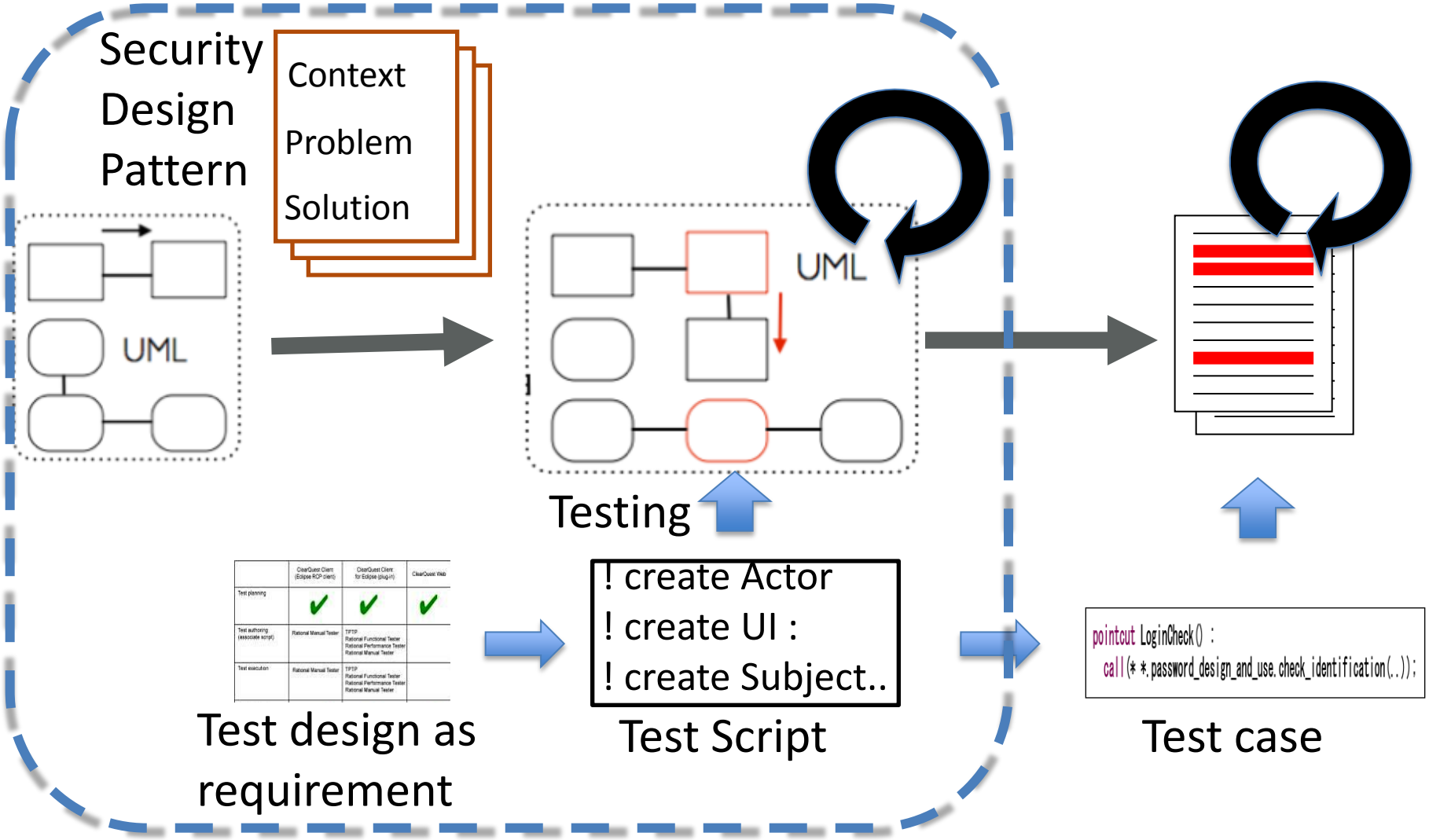
[PLoP'15] Systematic Mapping of Security Patterns Research, Conf. Pattern Languages of Programs Conference

Agenda

- Introduction
- Security patterns
- TESEM: testing models
- TESEM: testing code
- Conclusion and discussion

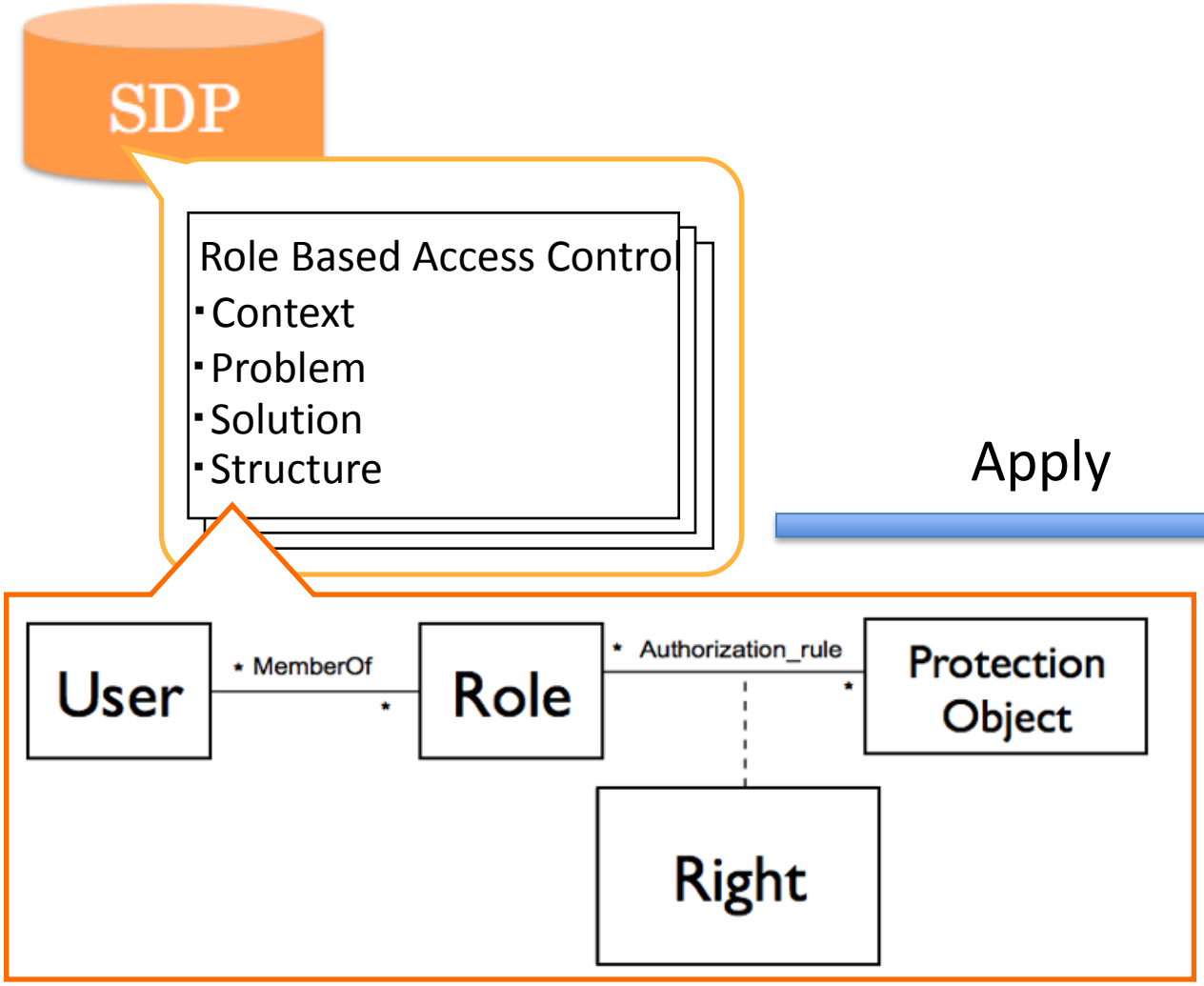
TESEM: Test Driven Secure Modeling Tool

[ARES'13][IJSSE'14][ICST'15]

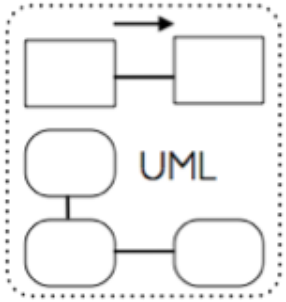


[ARES'13] Validating Security Design Pattern Applications Using Model Testing, Int'l Conf. Availability, Reliability and Security
[IJSSE'14] Validating Security Design Pattern Applications by Testing Design Models, Int'l J. Secure Software Engineering 5(4)
[ICST'15] TESEM: A Tool for Verifying Security Design Pattern Applications by Model Testing, IEEE ICST'15 Tools Track

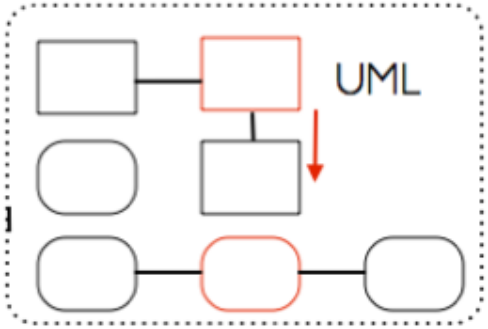
Application of Security Design Patterns (SDP)



Apply

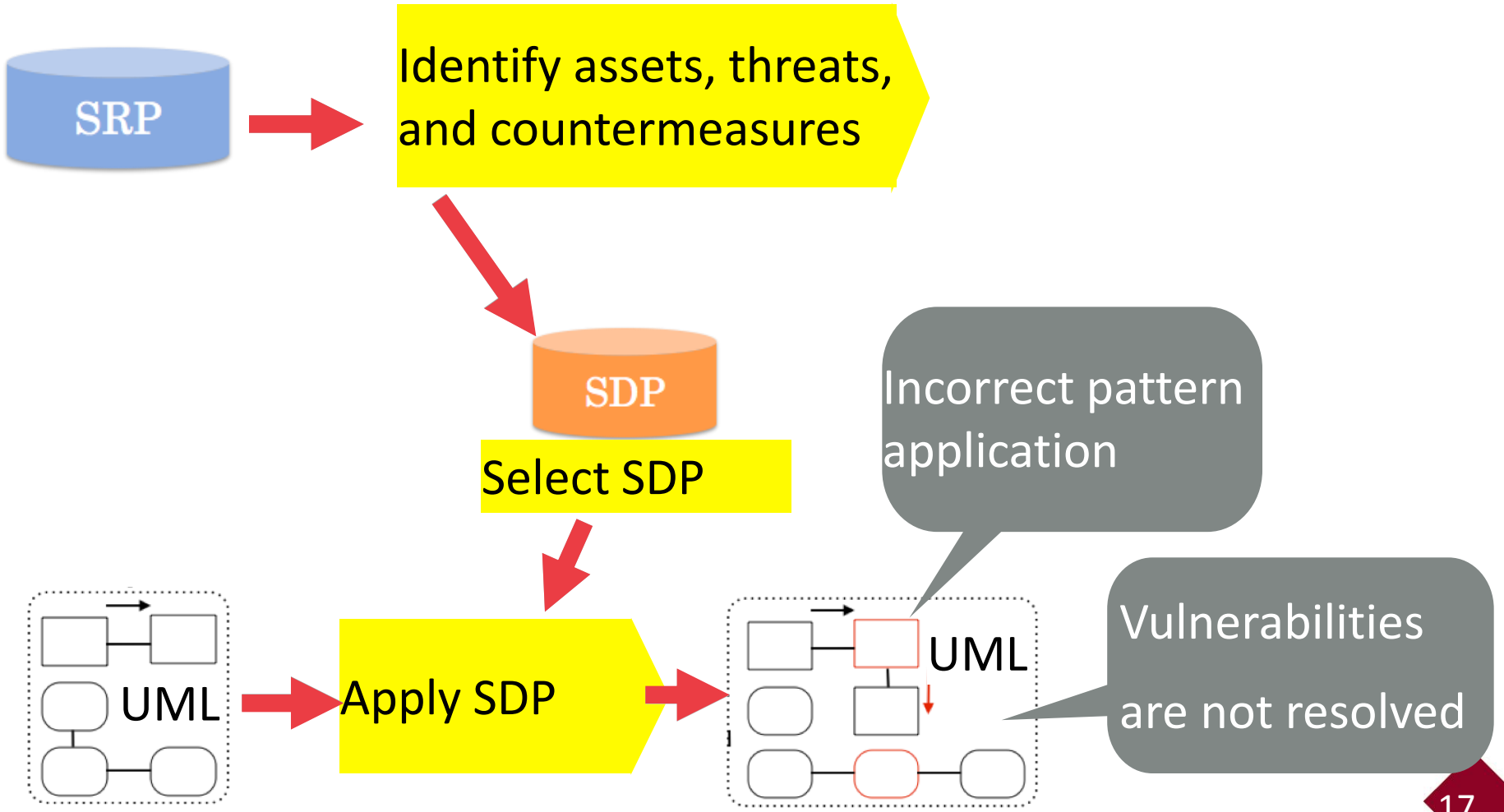


Model not considering security

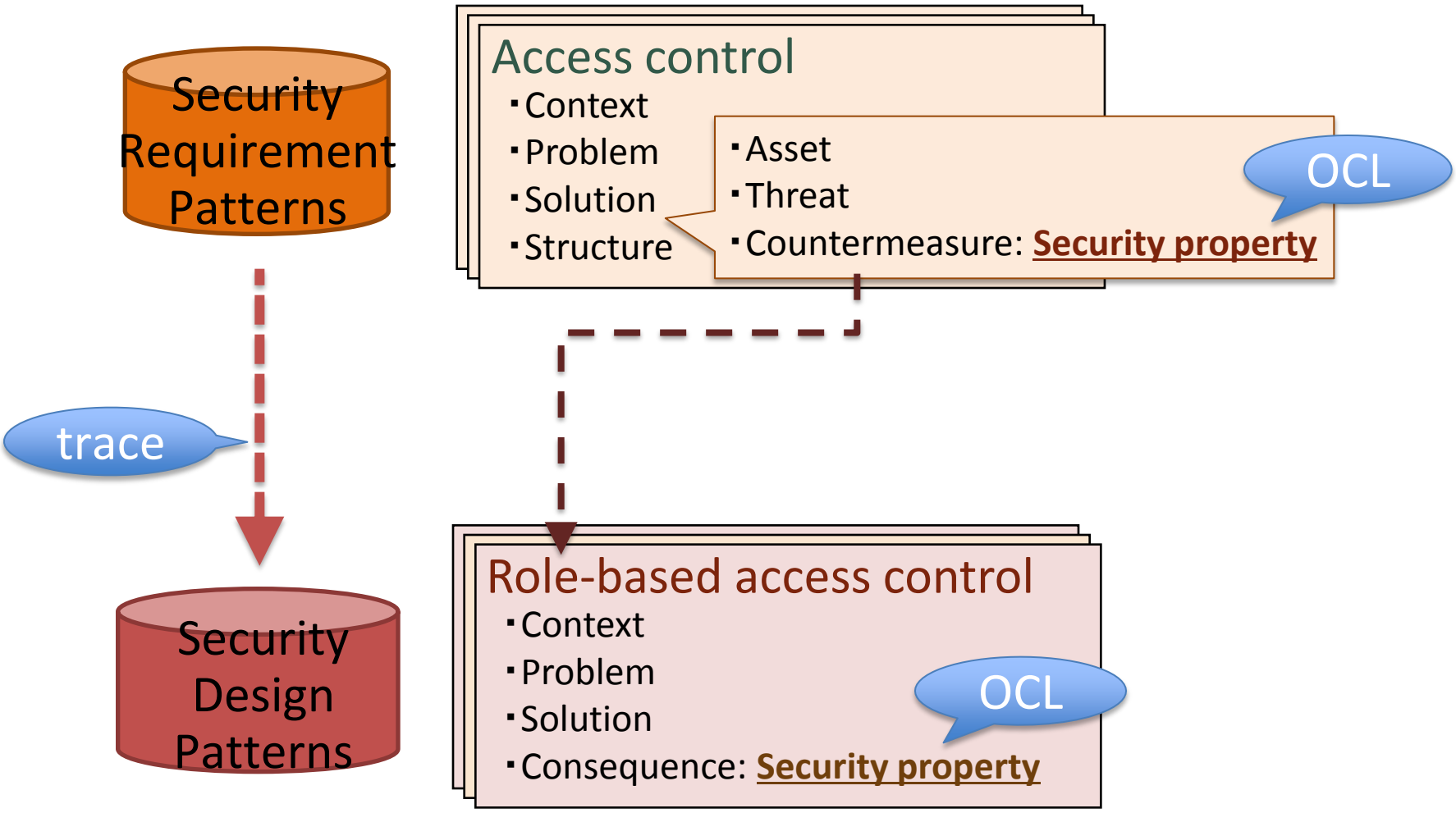


Model that realizes access control based on Role

Conventional problematic process

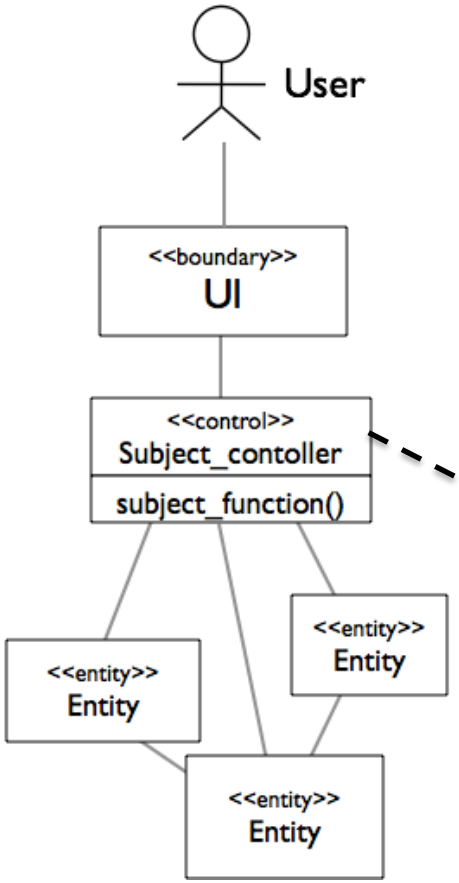


Security patterns with OCL constraints



Security property at requirements level

- Nine types of security properties
- E.g. “Access Control”



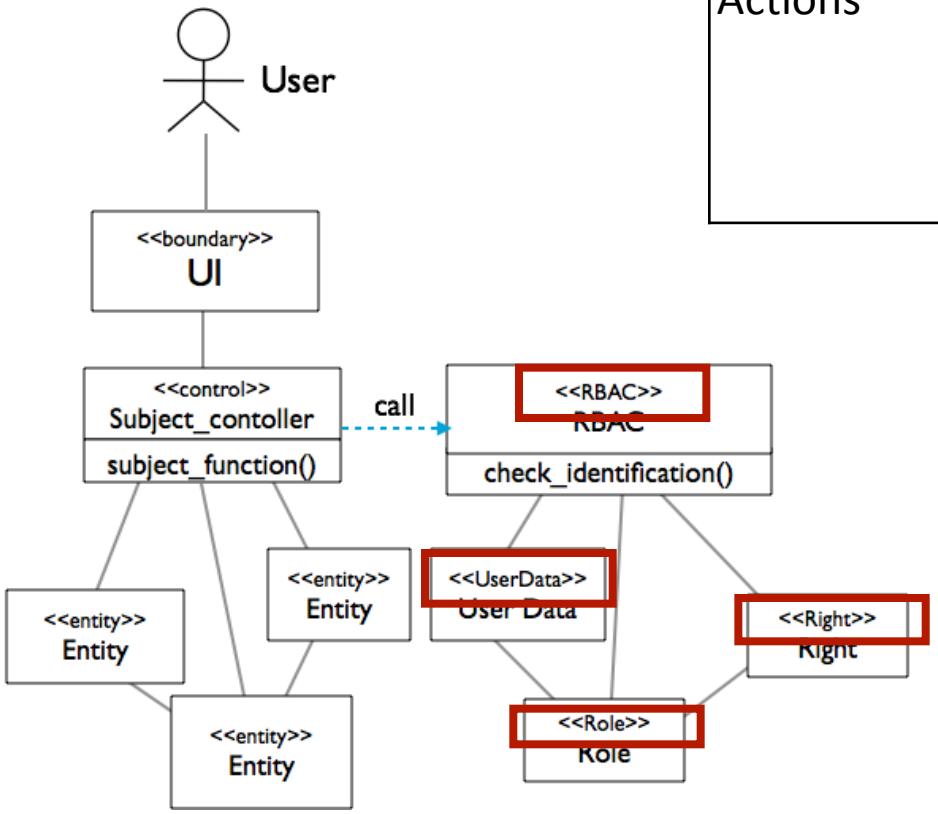
		1	2
Conditions	has access permission	Yes	No
Actions	execute subject function	×	
	not execute subject function		×

```
context controller
 inv Security Requirement :
  if self.UI.Actor.right = true then
   self.subject_function = true
 else
   self.subject_function = false
 endif
```

Security property at design level

- E.g. “Role-based access control (RBAC)”

		1	2
Conditions	access permission is given to <Role> which an <UserData> belongs	Yes	No
Actions	considers that actor has access permission	×	
	consider that actor does not have access permission		×
	execute subject function	×	
	not execute subject function		×

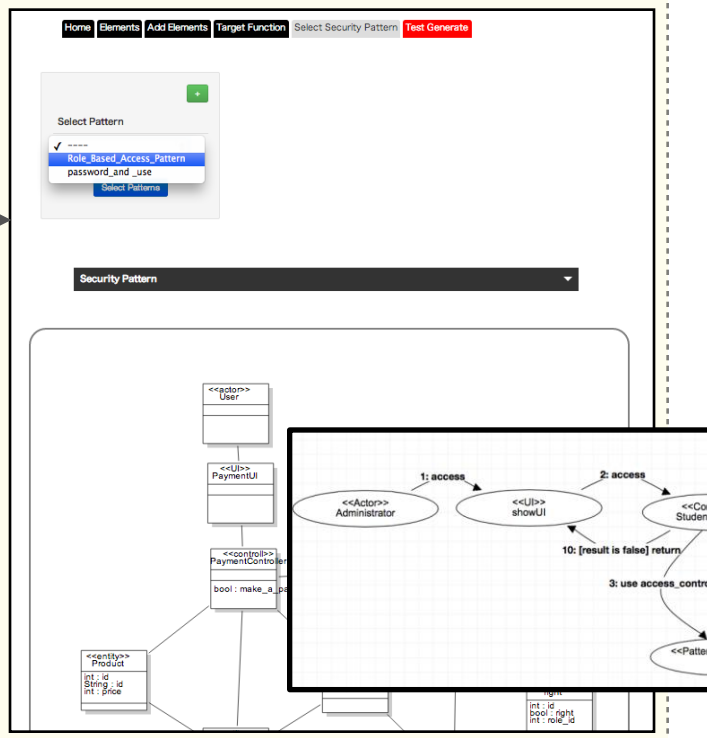



```
context subject_controller
 inv access_control:
  if self.RBAC.Right->exists(p |
        p.right = true and
        p.role_id = p.Role.id and
        p.role_id = p.Role.User_Data.role_id )
 then
   self.Subject_UI.User.Right = true
 else
   self.Subject_UI.User.Right = false
```

Test cases for “RBAC”

		1	2
Conditions	access permission is given in <Role> to which an <UserData> belongs	Yes	No
Actions	considers that an actor have access permission	×	
	consider that an actor does not have access permission		×
	execute subject function	×	
	not execute subject function		×

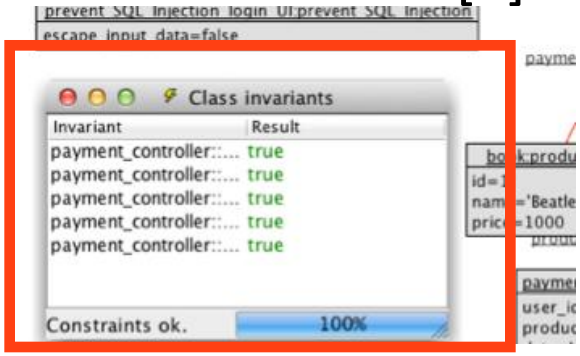
Our tool “TESEM”



- Class structure modeling
- Behavior modeling
- Application of security design patterns
- Generation of test cases

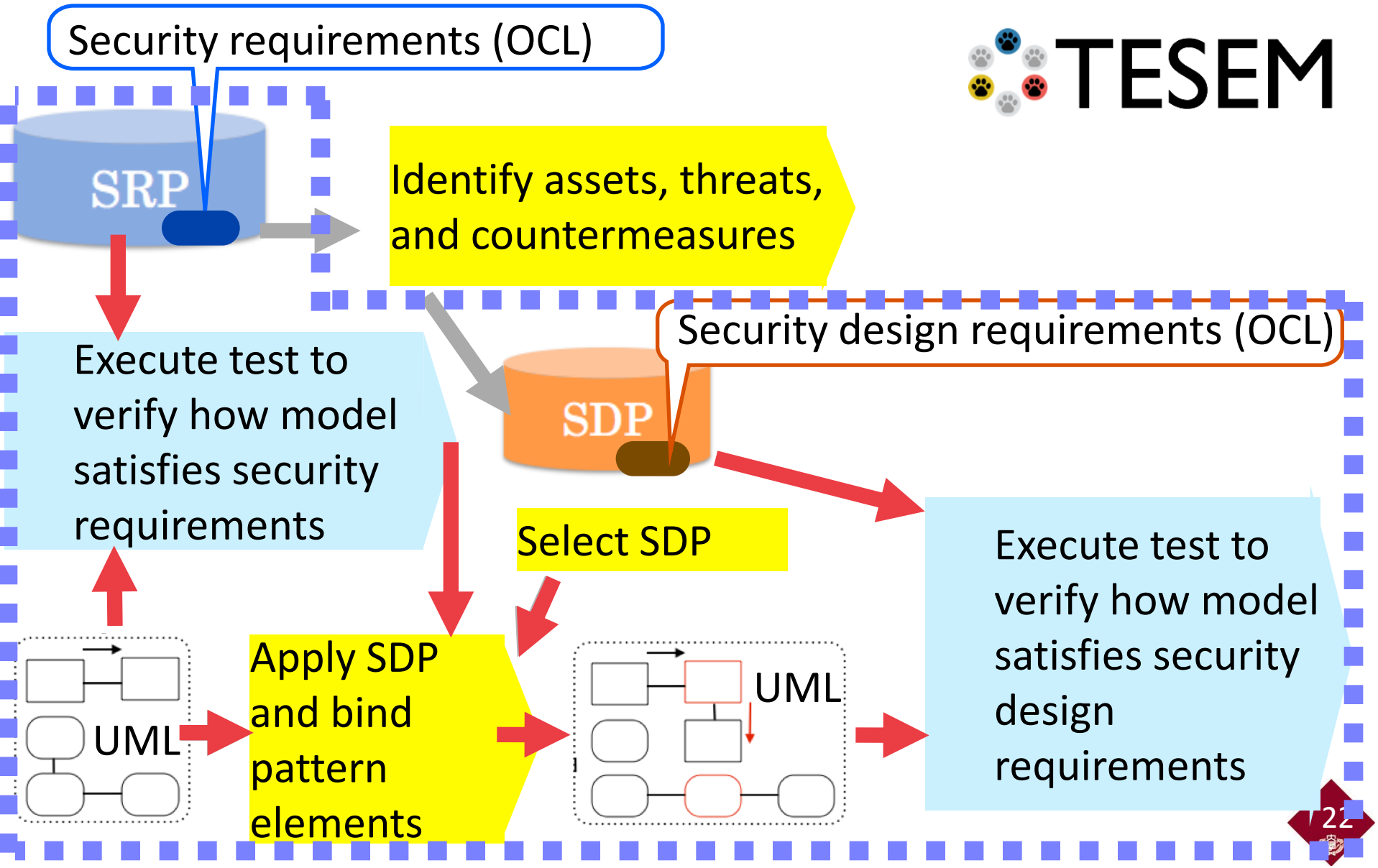
Test Script

Create instances and check OCL constrains on USE [*]

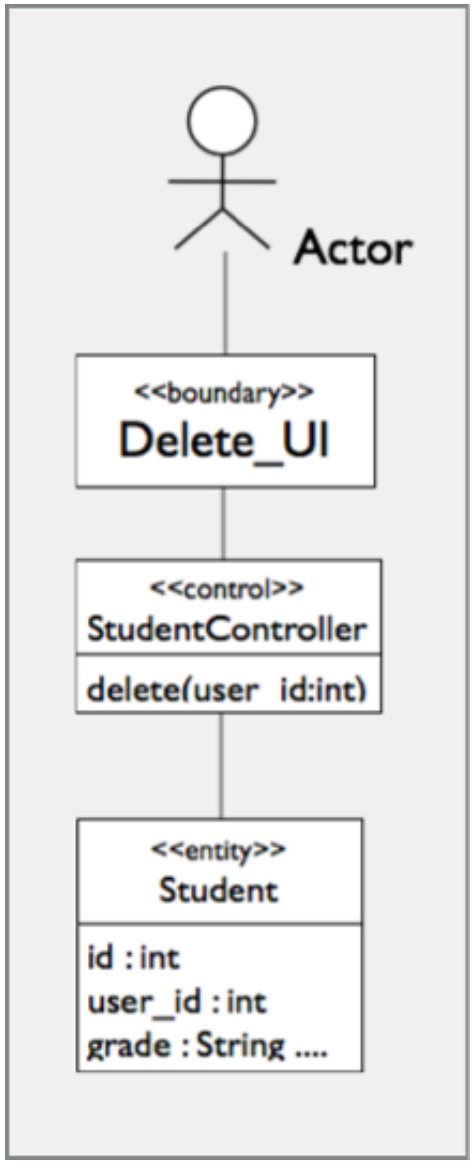


[*] F. Büttnera, et al., “USE: a UML-based specification environment for validating UML and OCL,” SCP, vol.69. 2007.

Design process using extended patterns



Case study: Setting



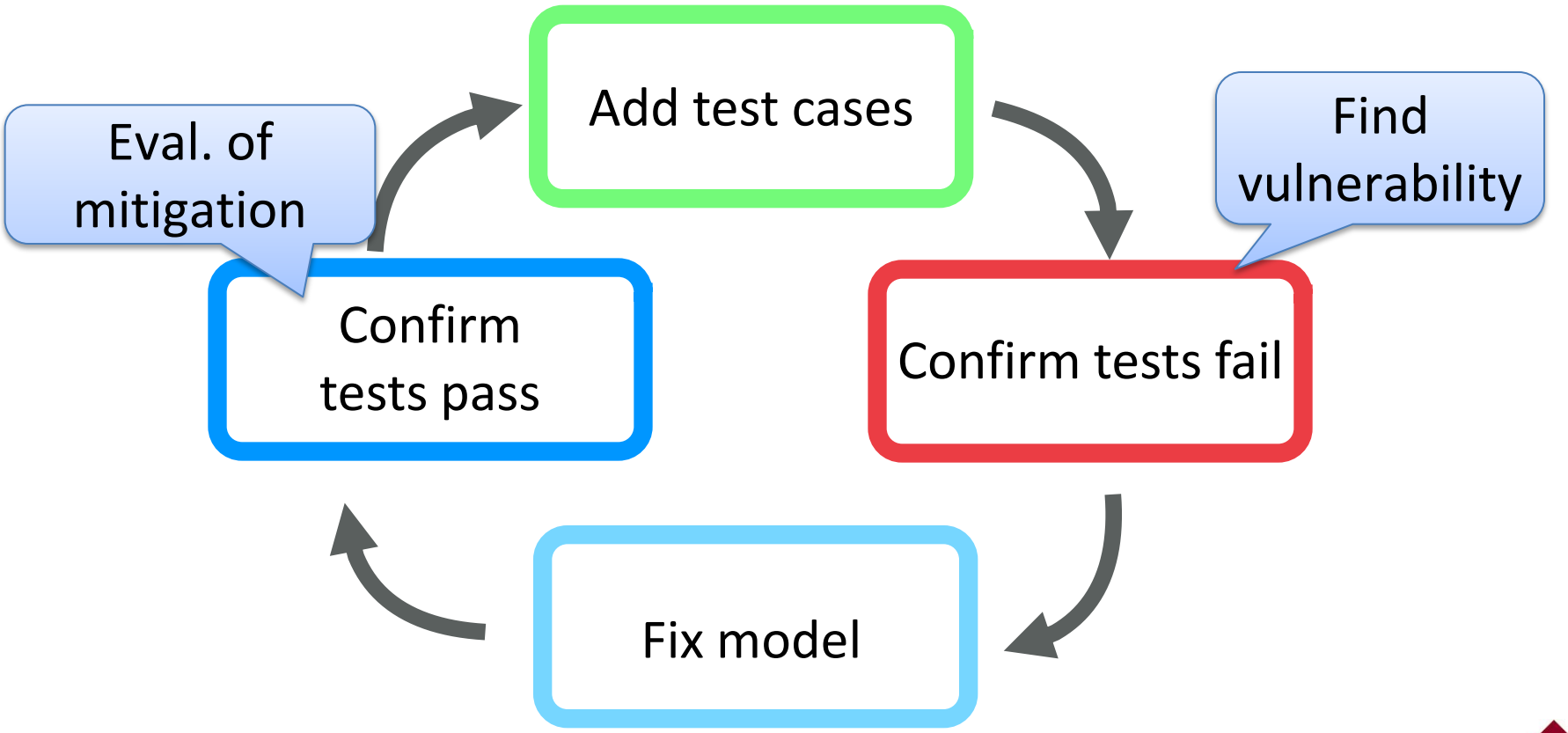
- Target: Delete function of StudentController
- Threat: Privilege Escalation

  → Any user can delete student's data
- Countermeasure: Access Control
- Selected Pattern: Role-based access control

  →Realize access control based on role's right

Test-driven secure design

- Security Properties are in the Test cases



Case study: Initial test for security requirement

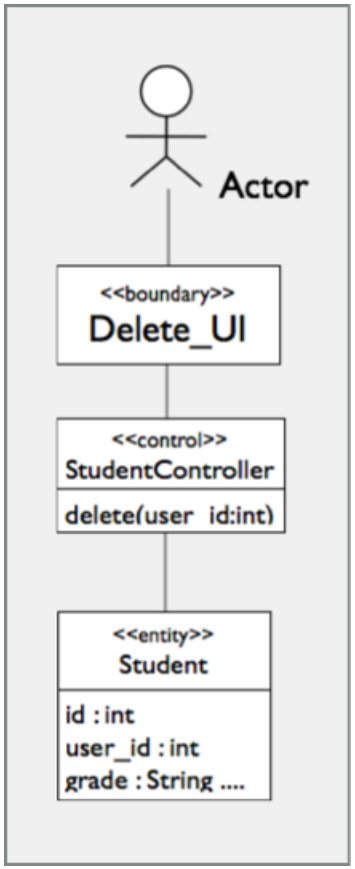
Security requirement as decision table

		1	2
Conditions	“Actor” has access right	Yes	No
Actions	execute “delete” function	×	
	cannot execute “delete” function		×

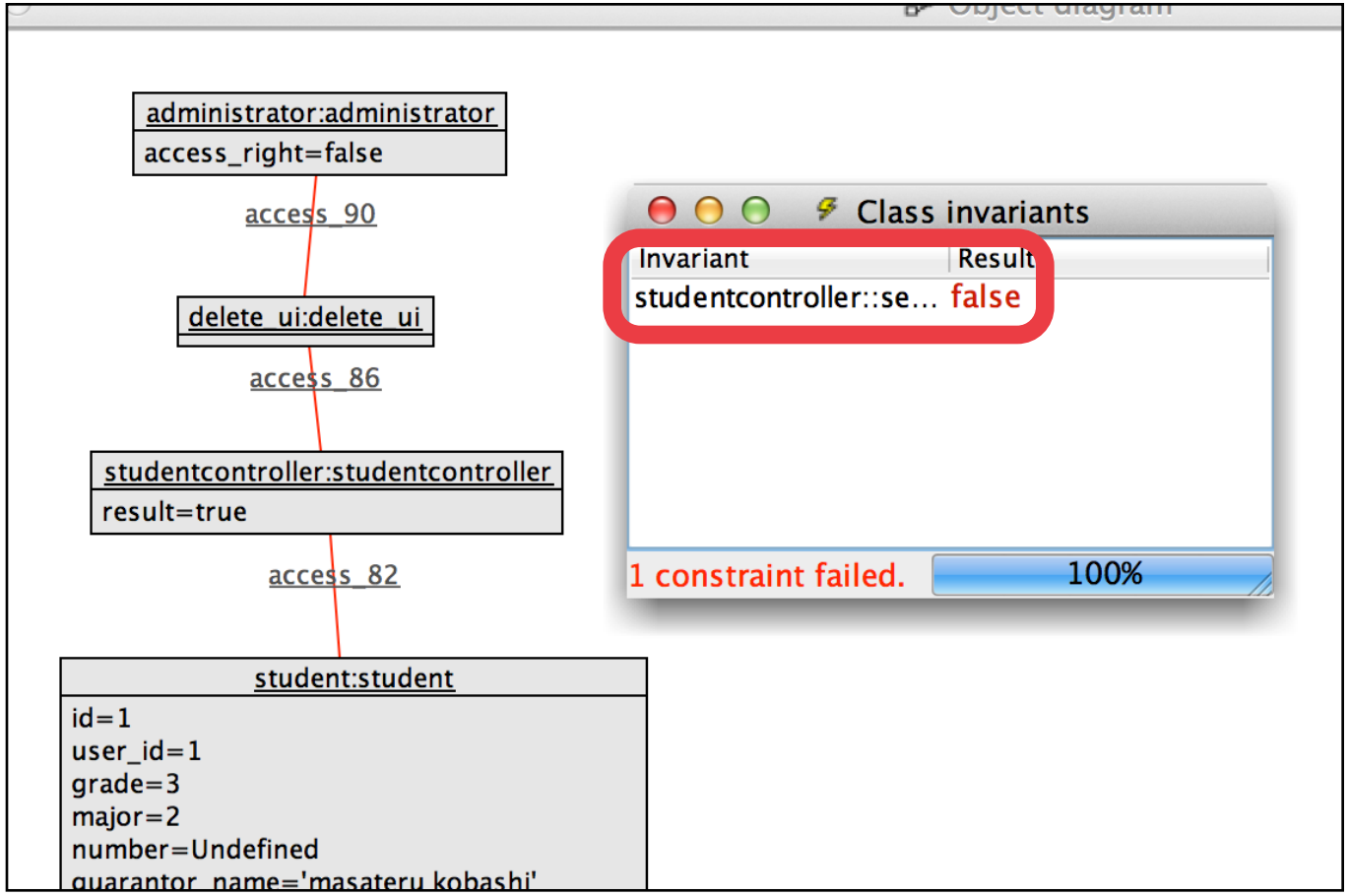
Security requirements as OCL expression

```
context StudentController
 inv SecurityRequirement :
  if self.DeleteUI.Actor.right = true and
   self.delete = true
 else
   self.delete = false
 endif
```

Verify whether model satisfies security requirement



Case Study: Test failed



Actor can execute “delete” function without access right !

Model may contain vulnerability causing Privilege Escalation.

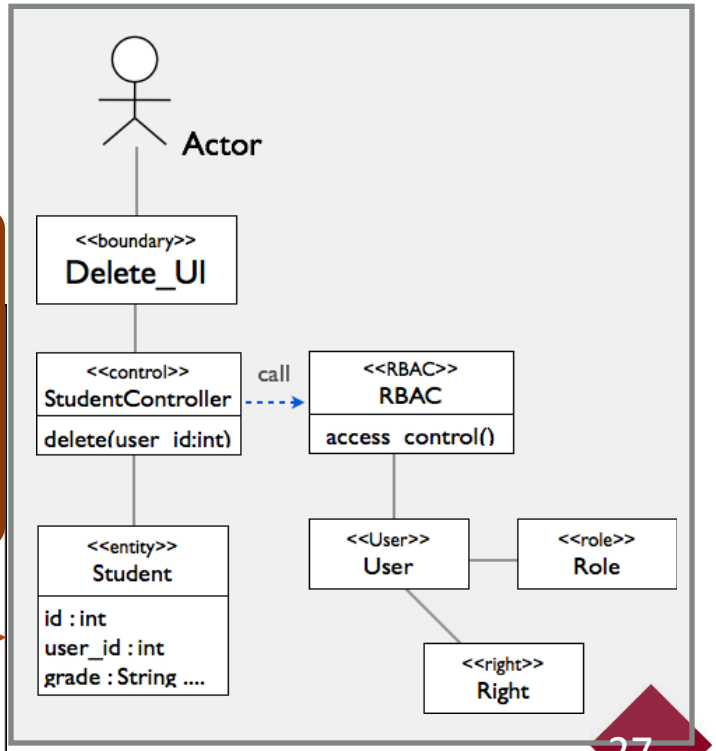
Case Study: Test for security design

Verify whether model with RBAC satisfies security design requirements

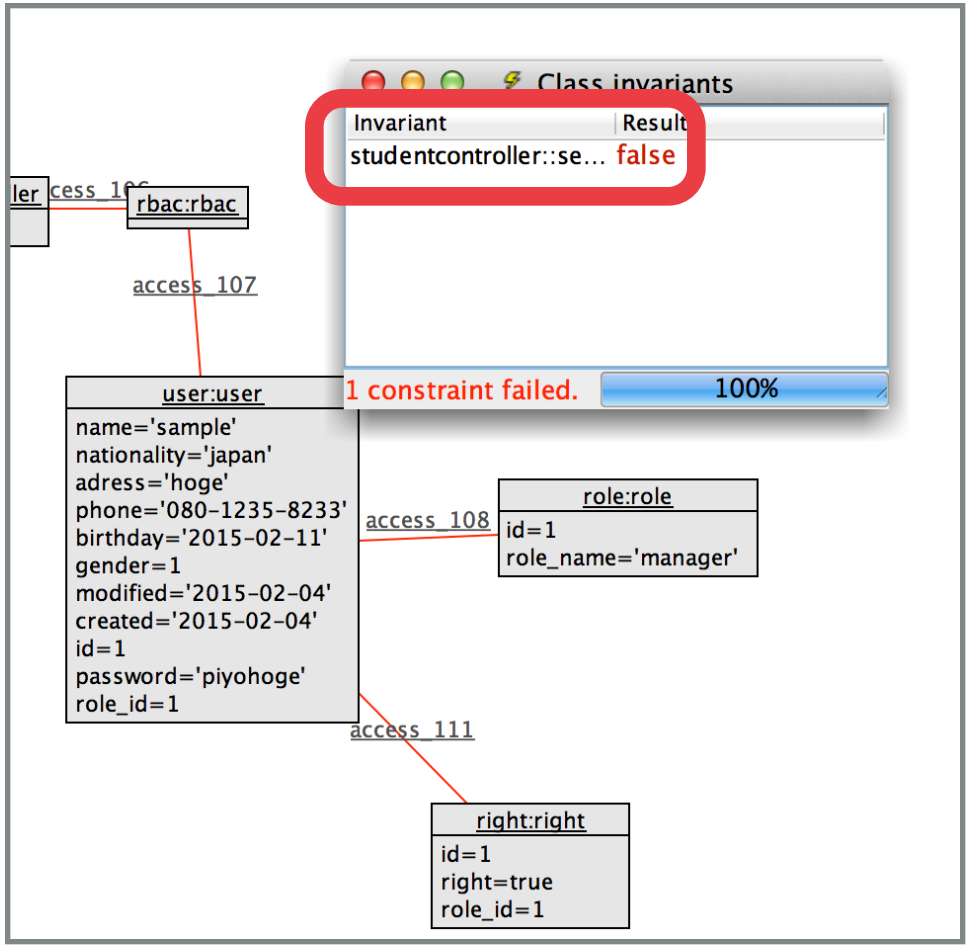
		1	2
Conditions	Rights are given in “Role” which an “User” belongs	Yes	No
Actions	consider that “Actor” have access permission.	×	
	consider that “Actor” does not have access permission.		×
	execute “delete” function	×	
	can not execute “delete” function		×

```
context subject_controller
 inv access_control:
  if self.RBAC.Right->exists(p |
        p.right = true and
        p.role_id = p.Role.id and
        p.role_id = p.Role.User.role_id )
 then
    self.DeleteUI.Actor.right = true and self.subject_function = true
 else
    self.DeleteUI.Actor.right = false and self.subject_function = false
 endif
```

Verify whether model satisfies security design requirement



Case Study: Test failed, again



Model does not satisfy security design requirements.

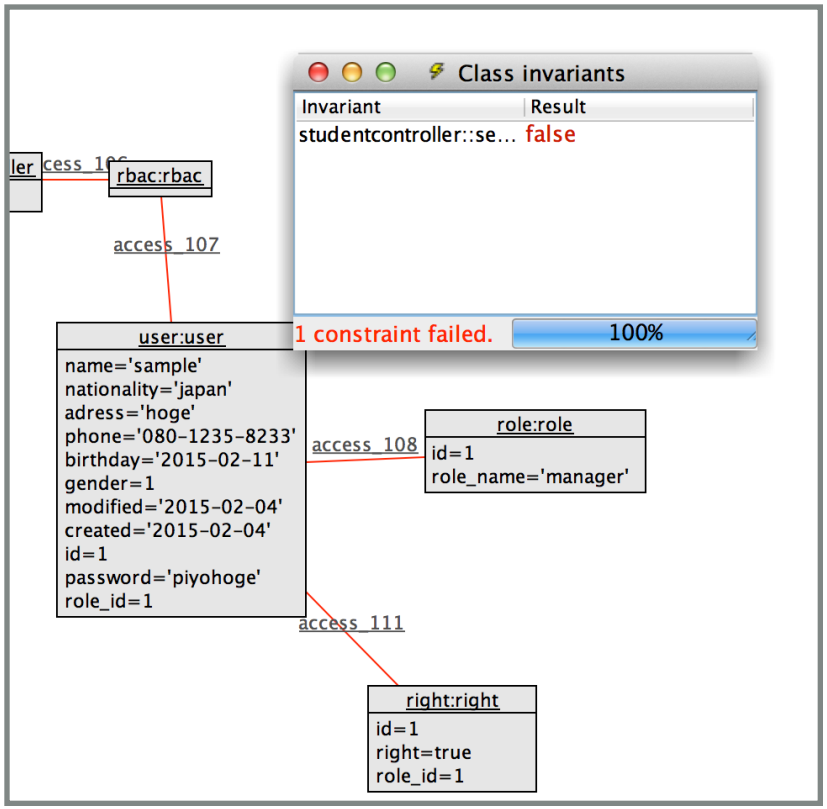
TESEM detected incorrect applications of design patterns

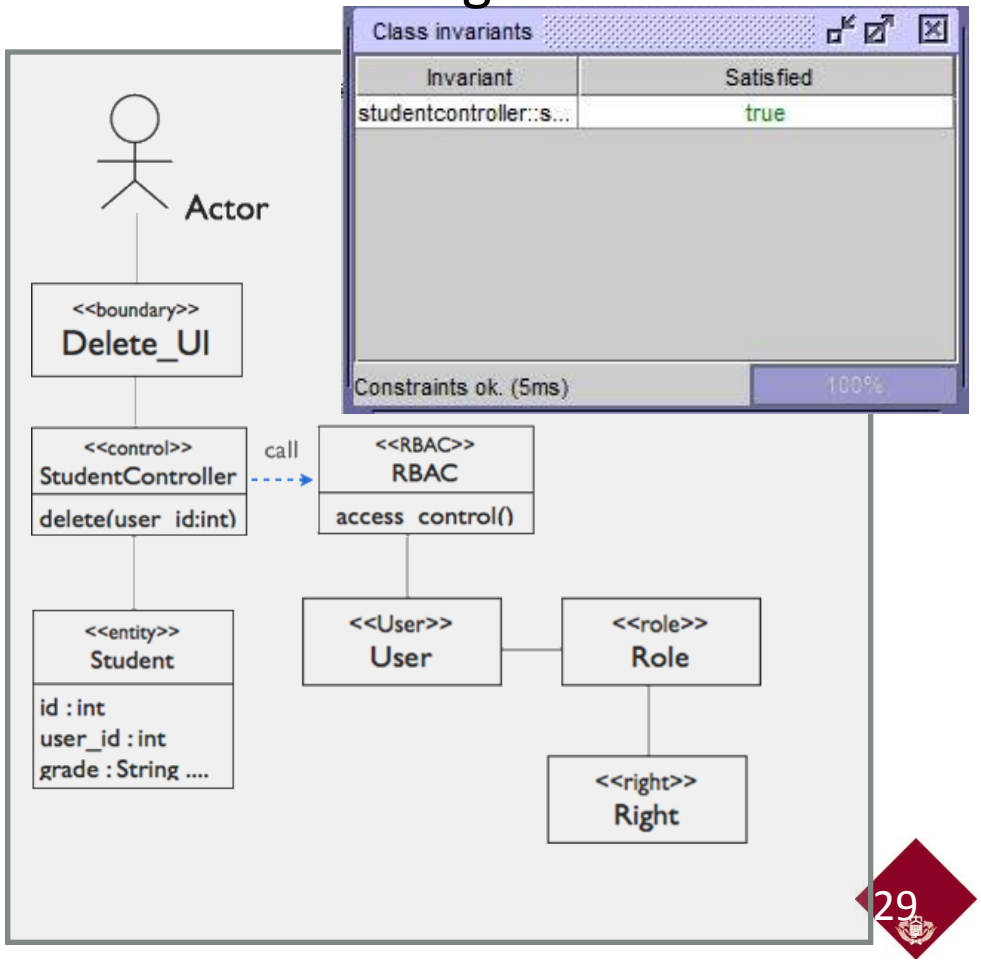
Case Study: Model fixing

Fix design model until the tests successfully pass.

Refactoring

Incorrect design → Correct design



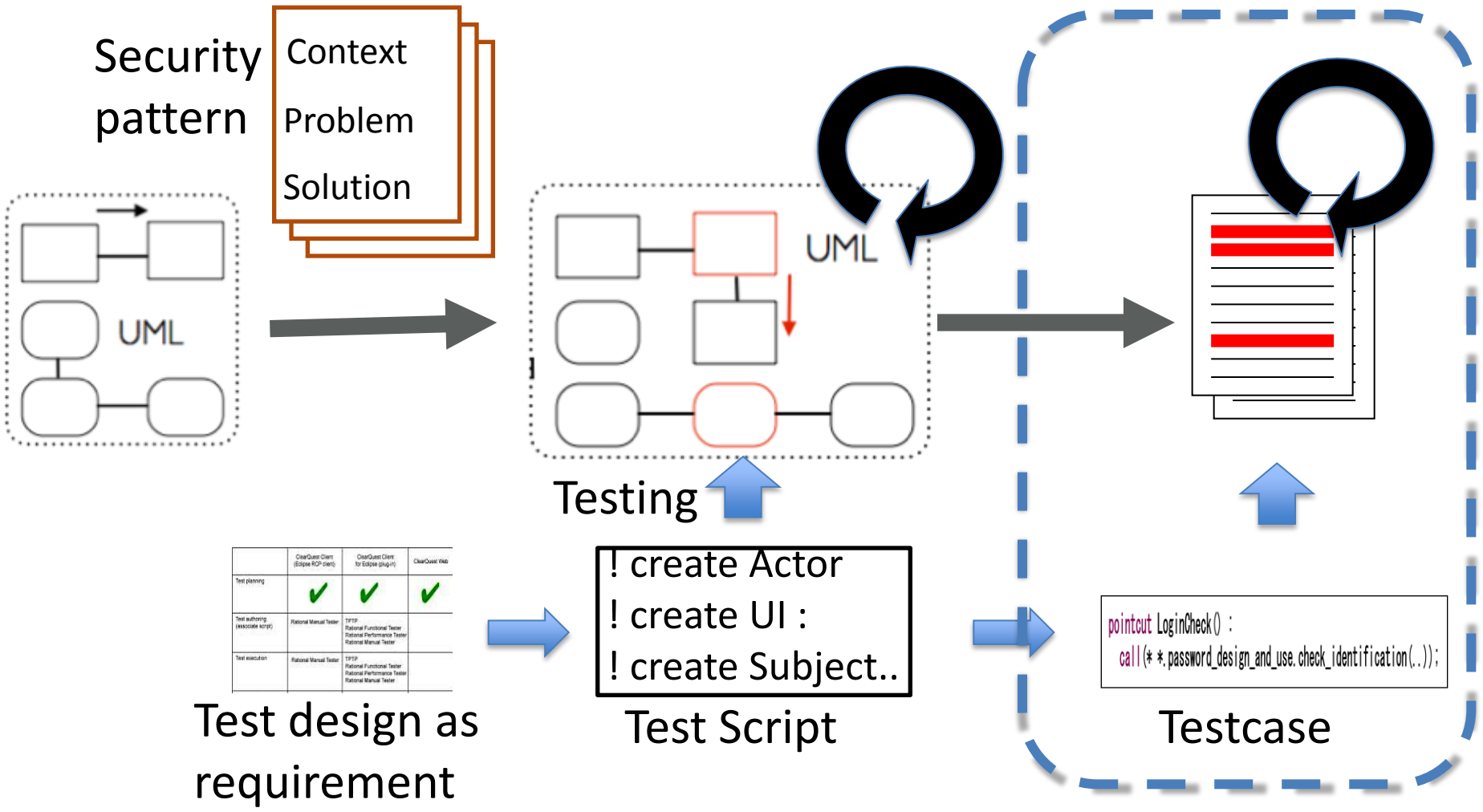


Agenda

- Introduction
- Security patterns
- TESEM: testing models
- TESEM: testing code
- Conclusion and discussion

TESEM: Test Driven Secure Modeling Tool

[ARES'14]



[ARES'14] Verification of Implementing Security Design Patterns Using a Test Template, Int'l Conf. Availability, Reliability and Security

Template for creating testcases

Security design pattern

Context
Problem
Solution

Preparation

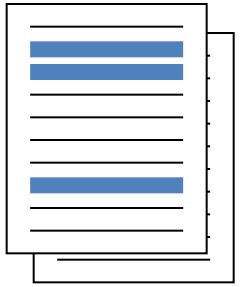
Instrumentation template in Aspect J

```
// PointCut
pointcut PasswordDesignAndUse() :
call(* *..PasswordDesignAndUse.check_identification(..));
// Advice
after() returning(Boolean right) :
PasswordDesignAndUse() {
setTemporary("PasswordDesignAndUse",right,h);
}
```

Concrete aspect in AspectJ

Instrumentation

SUT



Testcase template in Java

```
public class Password_Design_and_Use_Test {
String getTemporary(String name){
   . . .
}
@Test
public void test1() {
  login();
  assertEquals("PasswordDesignAndUse.check_identification() ",
PasswordDesignAndUse , "true");
}
```

Specify parameters

Concrete testcase in Java

Testing

Agenda

- Introduction
- Security patterns
- TESEM: testing models
- TESEM: testing code
- Conclusion and discussion

Controlled experiments

- Target: EMSec [*], 24 use cases, 31 classes
- Ex 1: Pattern application to design
  - 8 of 10 students applied patterns incorrectly without TESEM.
  - All students confirmed incorrect applications by TESEM.
  - Few students successfully fixed design.
- Ex 2: Fixing code with incorrect pattern application
  - All 4 students found more defects in shorter time per defect by using TESEM.
  - All 4 students successfully fixed most of defects by using TESEM, but required little longer time.

TESEM is useful for identifying incorrect applications.
Further fixing support is expected.

[*] EMSsec http://lab.iisec.ac.jp/~okubo_lab/Members/okubo/wiki/index.php?EMSSec

Conclusion

Pattern-oriented test architecture and extended security patterns using OCL-based constraints and templates, which include requirement- and design-level patterns

A new model/code-testing process based on TDD to verify appropriate pattern applications and the existence of vulnerabilities using these extended patterns

A tool called TESEM that supports pattern registration, application and verification

[ARES'13] Validating Security Design Pattern Applications Using Model Testing, Int'l Conf. Availability, Reliability and Security
[ARES'14] Verification of Implementing Security Design Patterns Using a Test Template, Int'l Conf. Availability, Reliability and Security
[IJSSE'14] Validating Security Design Pattern Applications by Testing Design Models, Int'l J. Secure Software Engineering 5(4)
[ICST'15] TESEM: A Tool for Verifying Security Design Pattern Applications by Model Testing, IEEE ICST'15 Tools Track

Discussion

- Pattern-oriented test architecture
  - Efforts for preparing constraints/templates paid off?
  - Correctness of patterns and concretization process?
  - Need more appropriate or different architecture?
- Security pattern ecosystem
  - Zero-day attack?
  - Common Vulnerabilities and Exposures (CVE) -> patterns -> concrete tests -> …
- Fixing / refactoring support
  - Automated fixing/refactoring ?

10th IEEE International Conference on Software Testing, Verification and Validation

Mar 13-18 (due Sep 2016)

aster.or.jp/conference/icst2017/