Rubacon: Automated Support for Model-based Compliance Engineering

Sebastian Hoehn[1] and Jan Jürjens[2]

[2]Computing Department
The Open University
GB

[1]Inst. of CS & Soc. Studies
Univ. Freiburg
Germany

J.Jurjens@open.ac.uk

http://www.jurjens.de/jan

Why Analysis of Security Permissions?

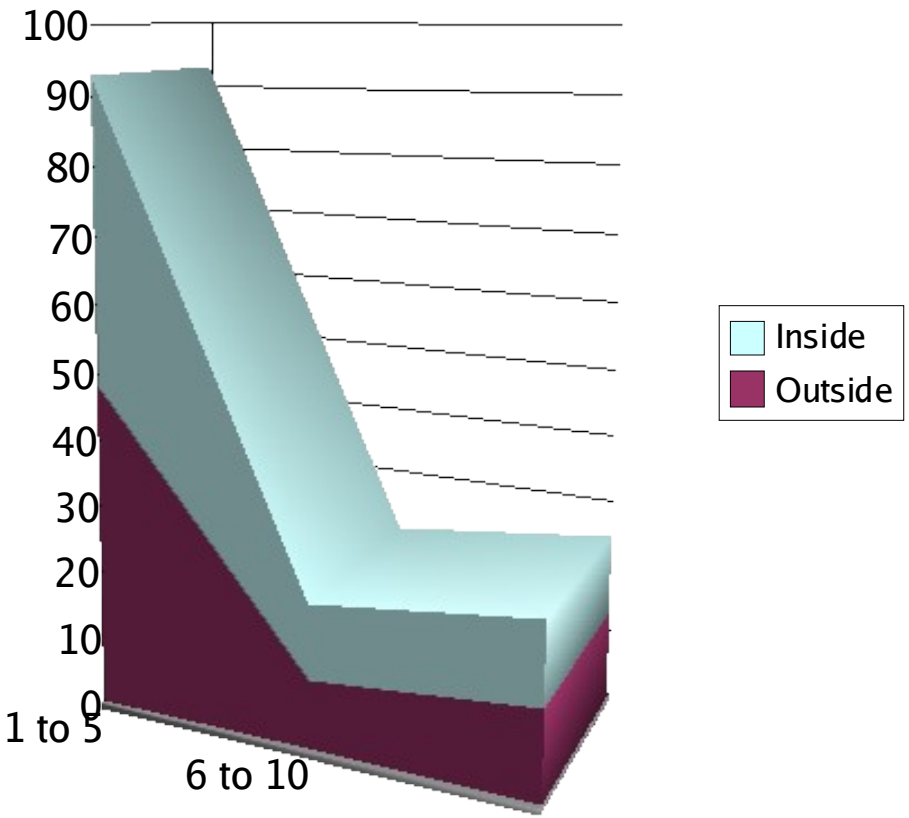
Computer Crime:

- 99% detected breaches
- 223 firms loose $ 455,848,000
- 50% inside attacks

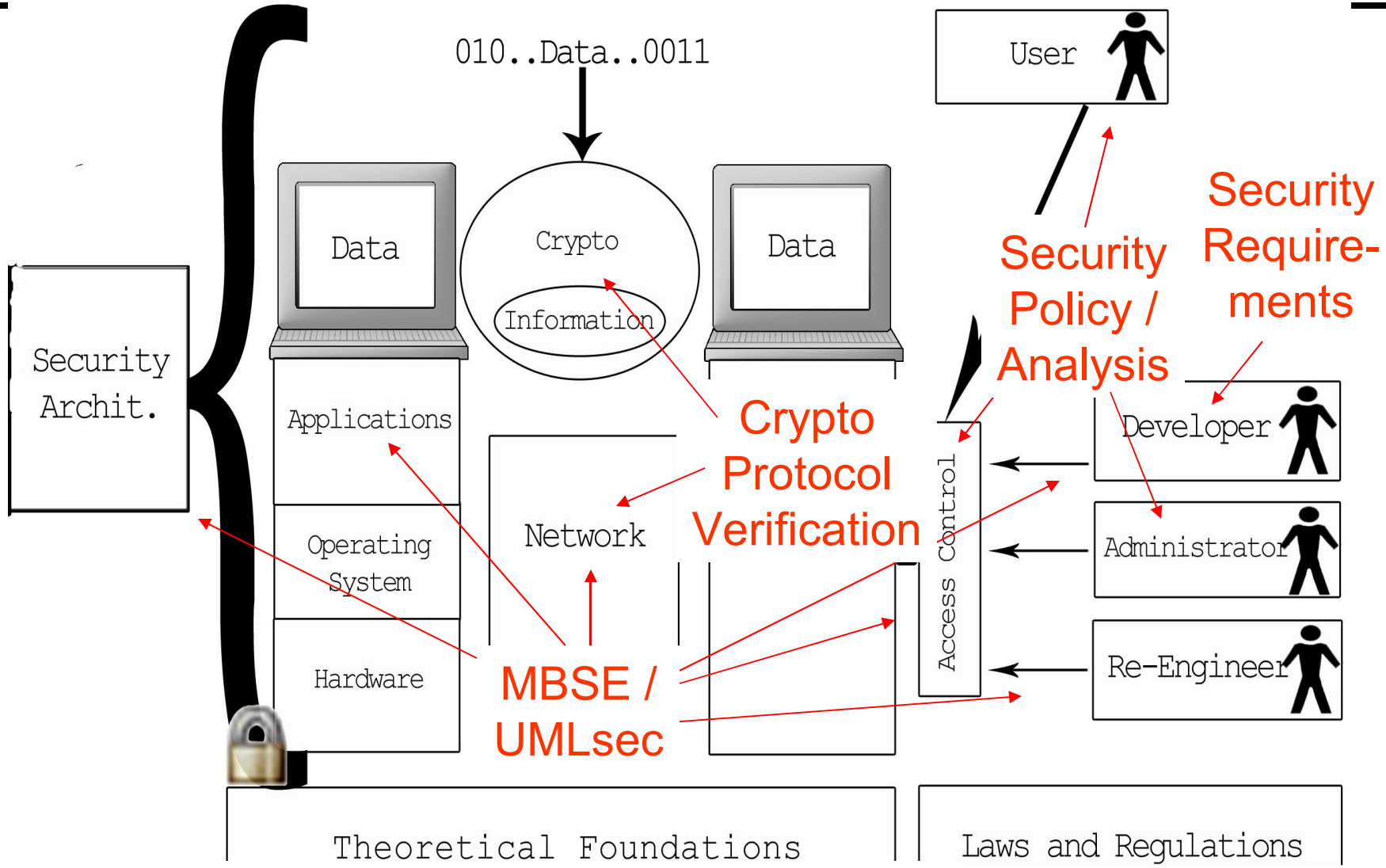
Automated Analysis

- Manual review of permissions impossible
- Huge amount of data
- Attacks from reviewer ?

Where Attackers come from



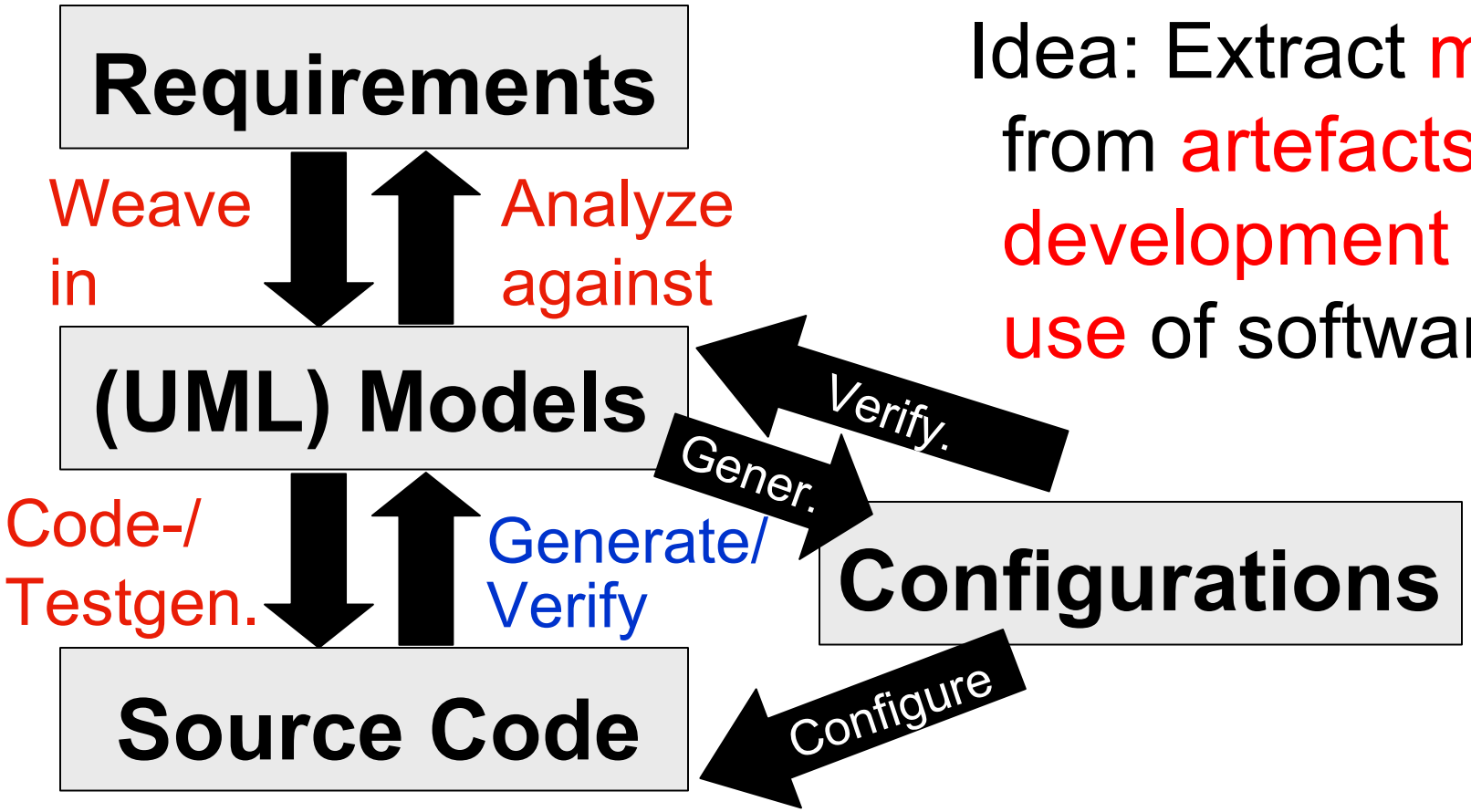
Security: Systems View



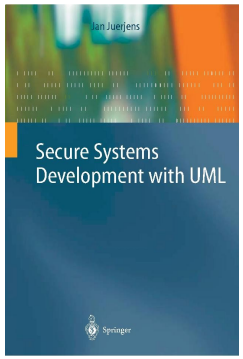
Formal Methods, vs Comput. Complex.

Privacy Regulations

Background: Model-based Security Engineering



Idea: Extract models from artefacts in development and use of software.



➔ Long-term goal: Tool-supported, theoretically sound, efficient automated security design & analysis.

Check configuration data

Example: check SAP permissions for security rules. Cannot do this manually:

- large data set (60.000 entries)
- complex interrelations between permissions on different levels
- dynamic changes, delegation
- manual analysiss trustworthy ?

Automated check increases security at central point.

Goals

- Read configuration from business application
- Generate report of detected weaknesses
- Flexible configuration of report's data
- Easily configurable for different business applications / use-cases
- Check large-scale databases
- Checks based on freely configurable rules

Architecture

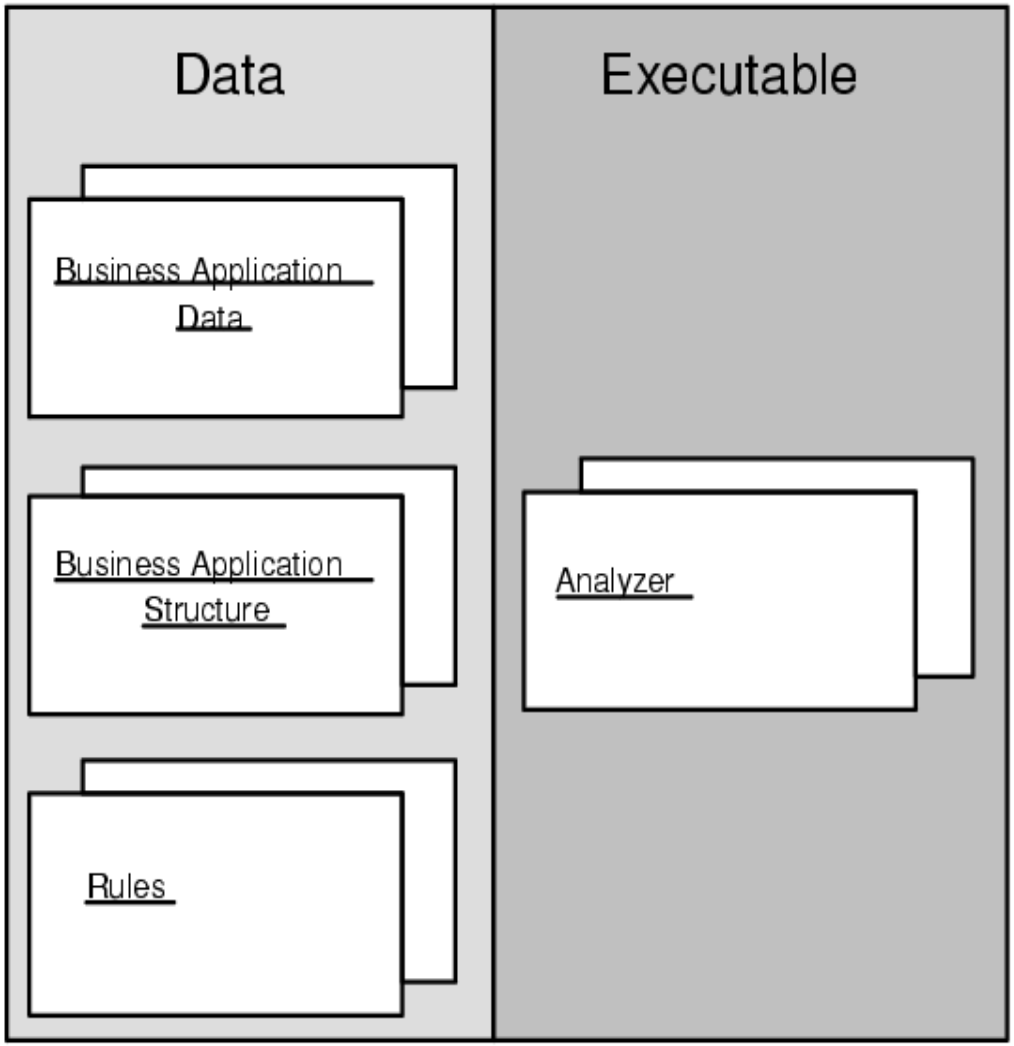
Data:

- Application structure
- Application data
- Rules, parameters

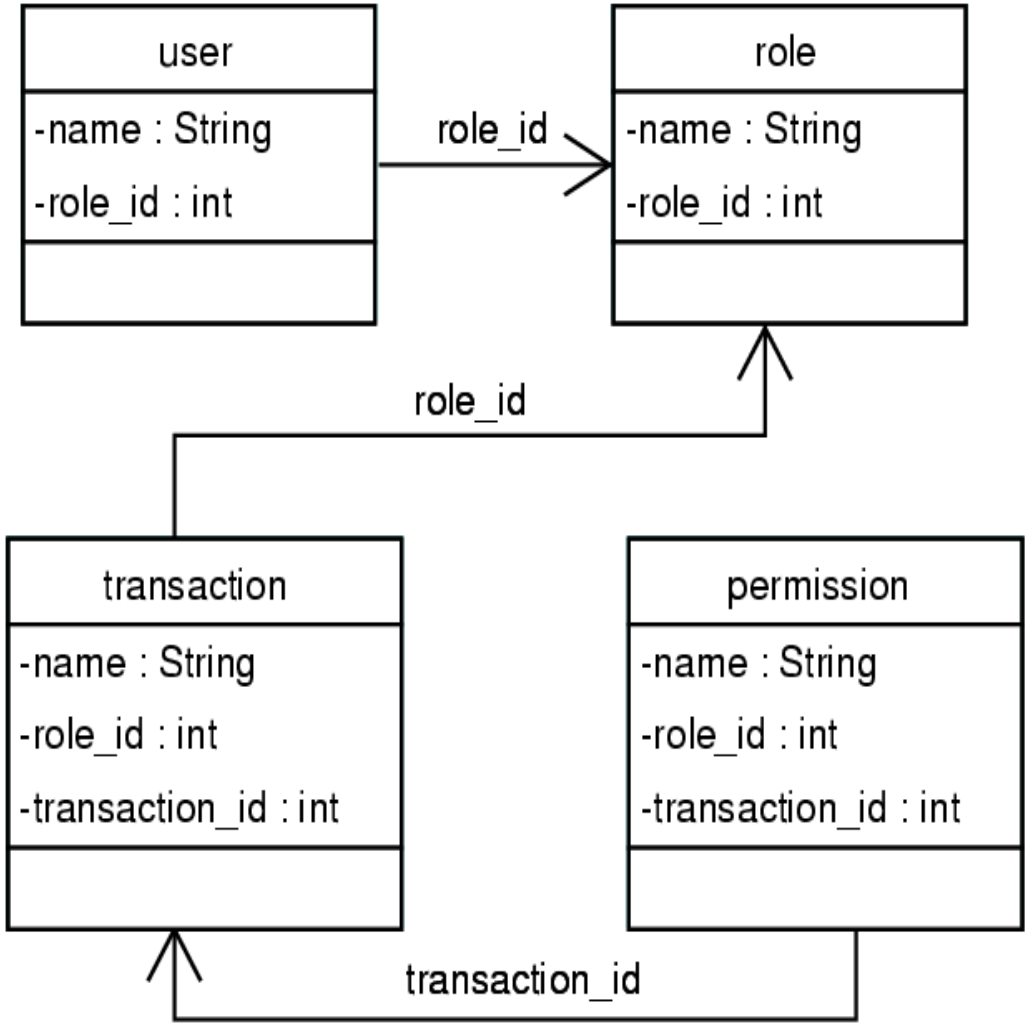
Executable

- Analyser application

Analyser reads data and generates report



Describing the Application



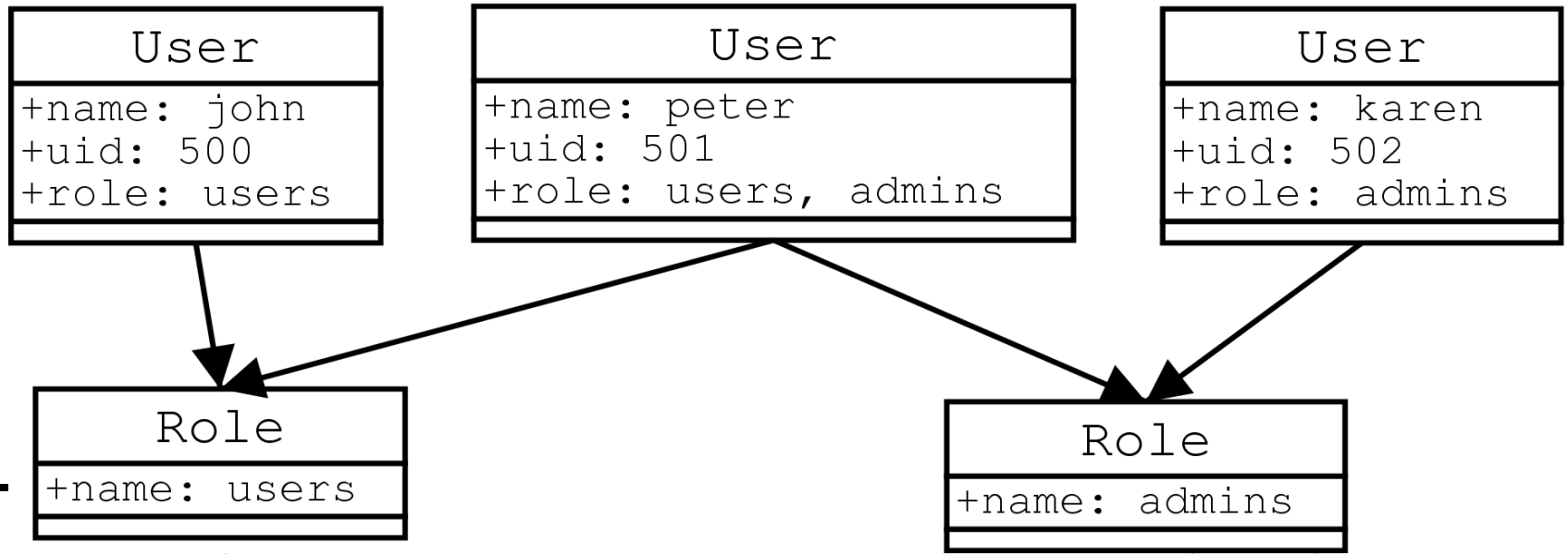
Configurable model of configuration data

- UML diagram
- Classes represent „objects“
- Associations represent „flow of information“

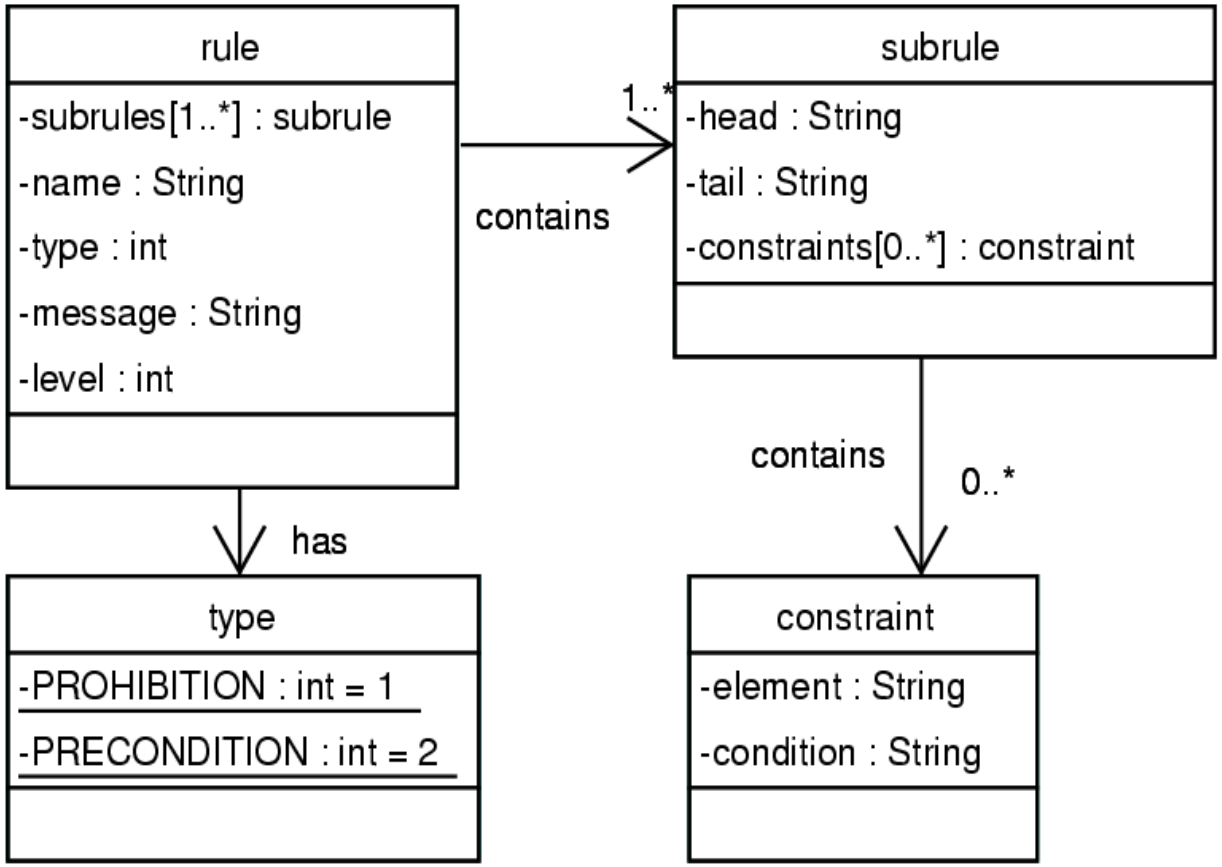
Configuration Data

- „Instantiates“ objects
- Given as XML

```
<rubacon>
   <user>
      <name>john</name>
      <uid>500</uid>
      <group>users</group>
   </user>
   <group>
      <group>users</group>
   </group>
...
</rubacon>
```



Security Permission Rules



Rule elements

- Name
- Type
- Message
- Priority
- Sub-rules
  - Head
  - Target
  - Constraints

Example: a simple rule

Check whether user with permission read_data_xyz does not have role_writer.

Rule:

- Name: check_user_role_perm
- Type: prohibited
- Message: User ... has role ...
- Priority: warning
- Sub-rule 1: Head: permission, Target: user
- Sub-rule 2: Head: user, Target: role, Constraint: role != role_writer

Evaluating Rules

Rules evaluated using Prolog

- Every graph node atomic term
- Every rule gets associating term
- Prolog finds unification for set of parameters (A, B, C)

Example:

Graph node:

- user(john, 501).

Associating Term:

- user_role(A, B, C) :- user(A, B), role(B, C).

Example: Separation of Duty

Karen: create purchase

Susan: release purchase

John: place orders

Goal:

- Ensure roles „create“, „release“ separated

Rule:

- Type: prohibition
- Message: SOD violated

Subrule 1: Head: create role, Target: user

Sub-rule 2: Head: release role, Target: user, Constraint: subrule1.user == subrule2.user

Example: SAP Transactions

Transactions in SAP implicitly grant access to sub-transactions

„Transitive“ permissions error-prone

Goal: Find users with access to „dangerous“ transaction_xyz

Transactions part of UML model

Rule:

- Type: prohibited
- Message: user ... can access ...

Subrule:

- Head: transaction_xyz
- Target: user

Implementation

Features:

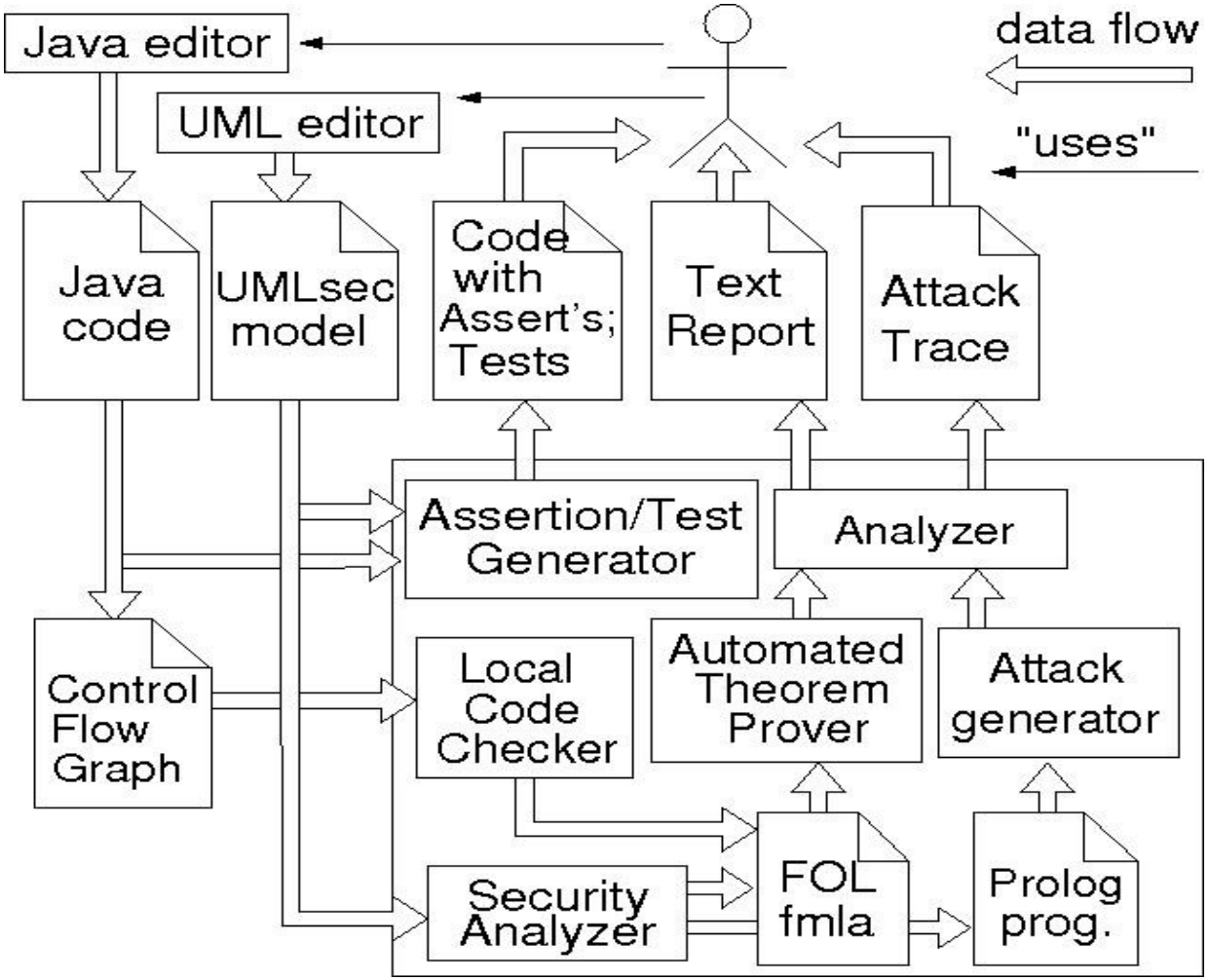
- Java-application
- GUI for tool configuration
- Configurable for different business-applications

Techniques:

- Swing
- Component-based
- XMI for UML parsing
- XML for information parsing
- Prolog for checking rules

Tool Support

[FASE05,ICSE06,ASE07, STTT07,ICSE08]



Configuration analysis:

- user permissions,
- firewall rules/policies

Open-source

Some Applications

Analyzed designs / implementations / configurations e.g. for

- Biometry- or smart-card-based identification
- authentication (crypto protocols)
- authorization (user permissions, e.g. SAP systems)

Analyzed security policies, e.g. for privacy regulations.

Related Projects

- PhD project on Verifying Implementations of Crypto-Protocols in C (MSR Cambridge / A. Gordon)
- RoySoc JIP with TU Munich on Formal Model-based Analysis of Cryptoprotocol Implementations
- RoySoc JIP with NII (Tokyo) on Security Requirements vs Design
- PhD project on IT security risk assessment with Munich Re
- PhD project on Adaptive Security for Ambient Technology
- PhD project on fuzzy reasoning for IT security risks
- PhD project on model-based development for avionics

Questions ?

More information (papers, slides, tool etc.):

http://www.jurjens.de/jan

J.Jurjens@open.ac.uk