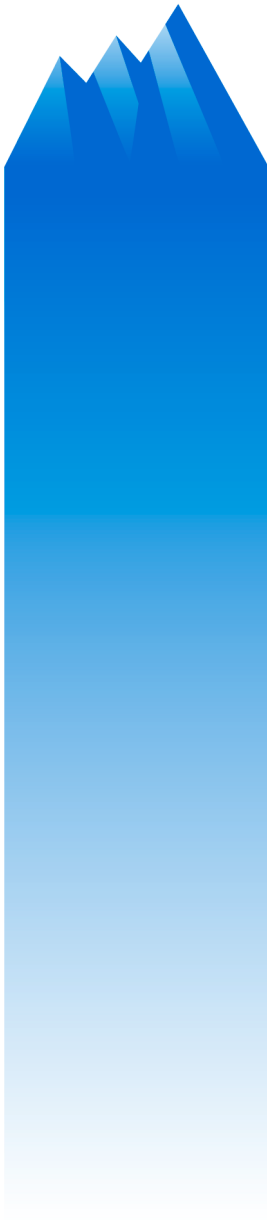
QUALITY ENGINEERING



Towards Application-Oriented Policy Configuration for SELinux

Berthold Agreiter

berthold.agreiter@uibk.ac.at

Research Group Quality Engineering
University of Innsbruck

SELinux Developer Summit
Ottawa, 07/22/08

Security Configuration

- **Currently mostly for experts**
- **Extensions needed to broaden target audience**
  - Encourage integration of SELinux for “internal” security requirements
  - Not only preconfigured policies/modules
- **Protect *specific objects* (e.g. patient records) instead of just confining applications**
  - application-centered -> data-centered
- **Admins know how applications work -> the difficulty is to know which kernel objects they need**

Idea

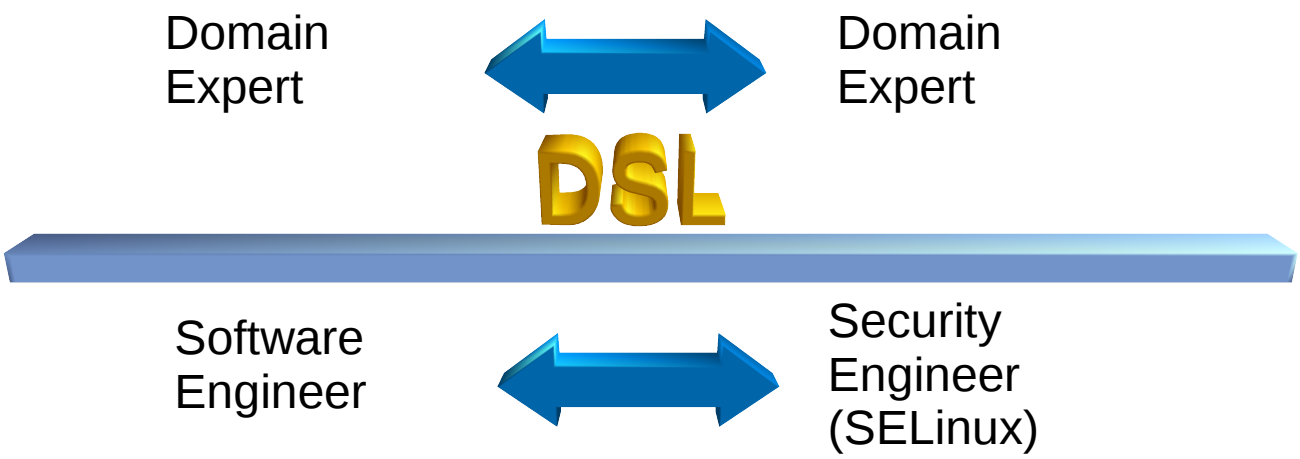
- **Abstraction helps**
- **Abstraction from policy language is already there**

- **But: the objects/permissions are hard to handle with**
- **Abstract from objects/permissions**
  - map them to real system objects

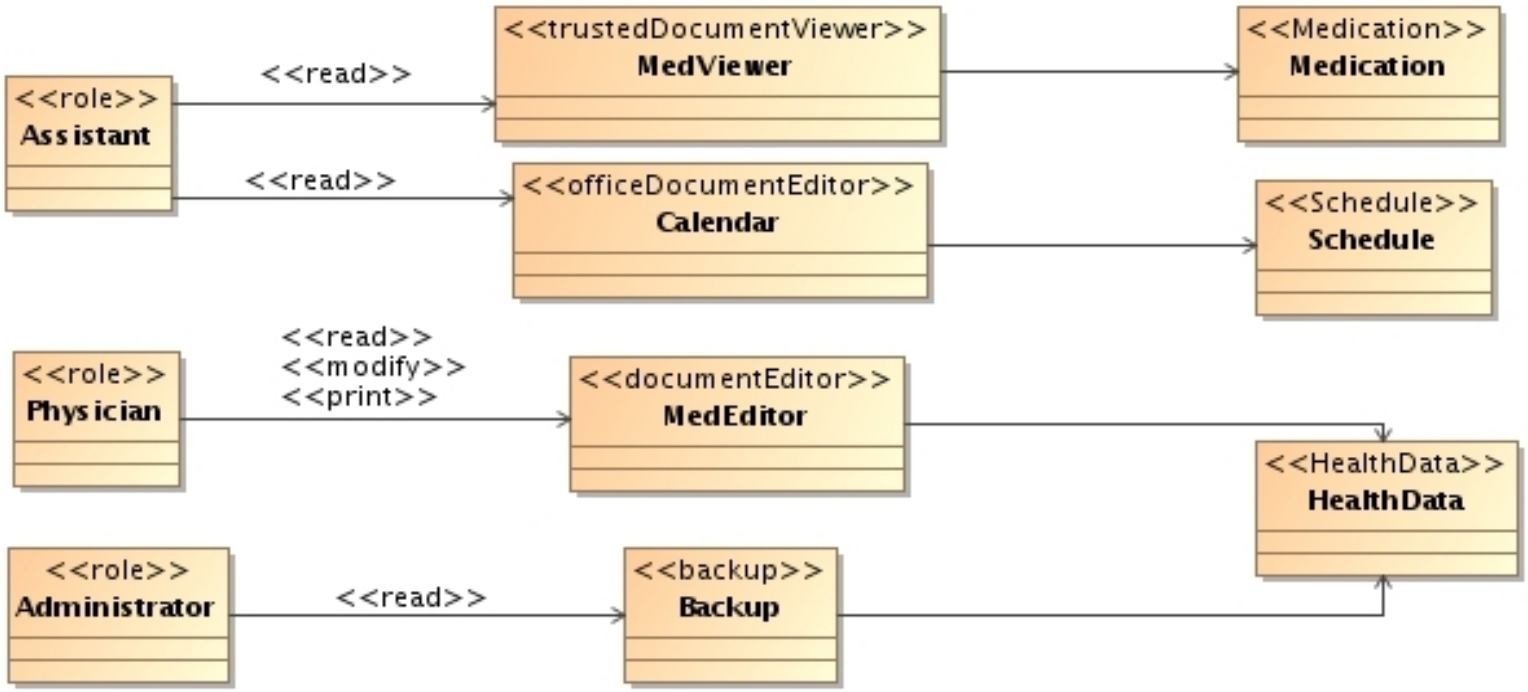
Abstraction

- **Well-explored topic in Software Engineering**
- **Security Engineering needs to catch up**

- **Models help**
  - Models for a specific purpose -> DSL
  - Models are versatile

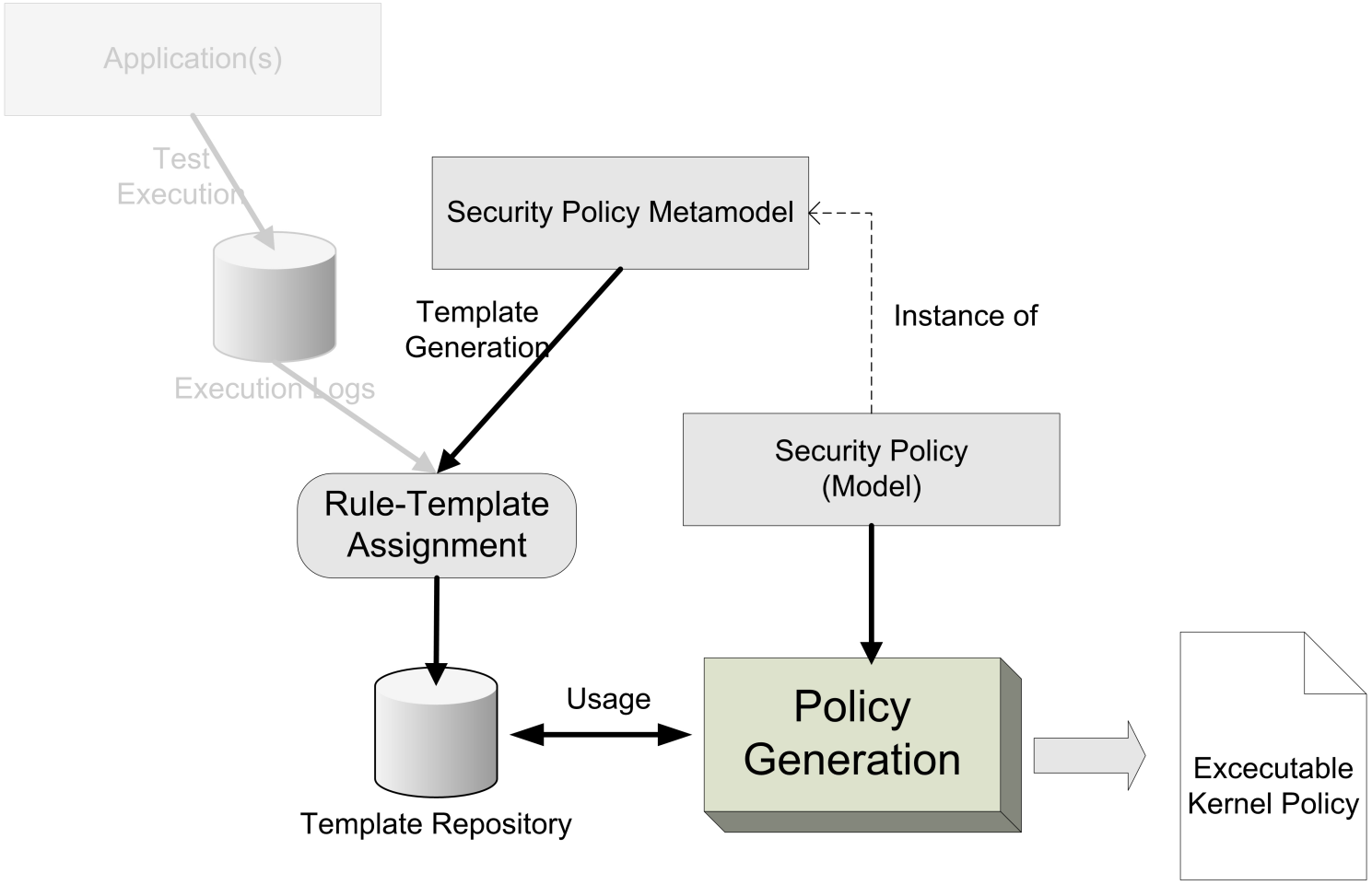


Domain Specific Language

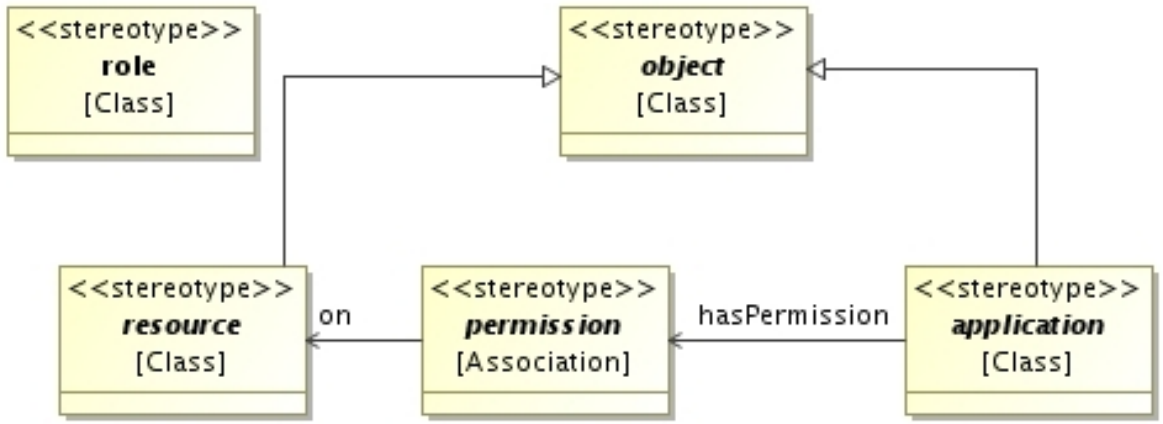


The overall process

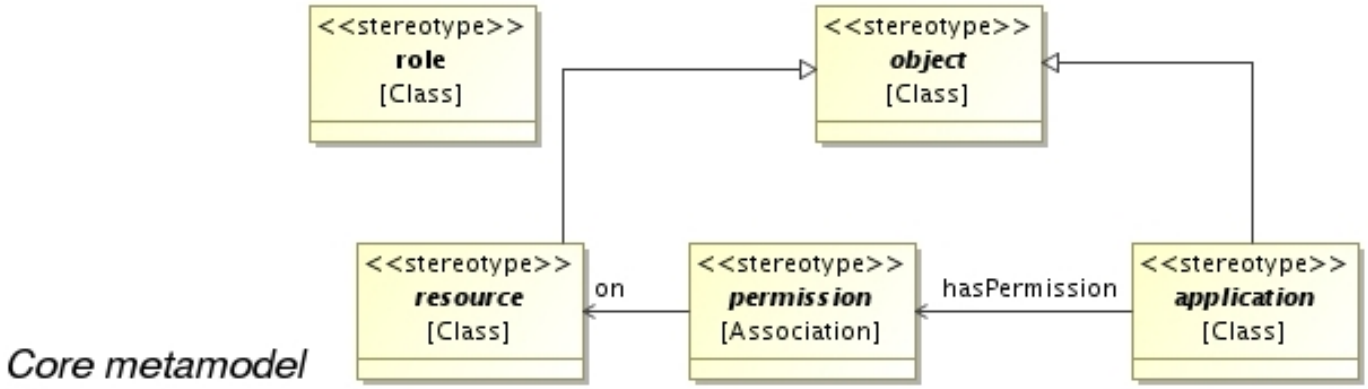
MDS approach to SELinux

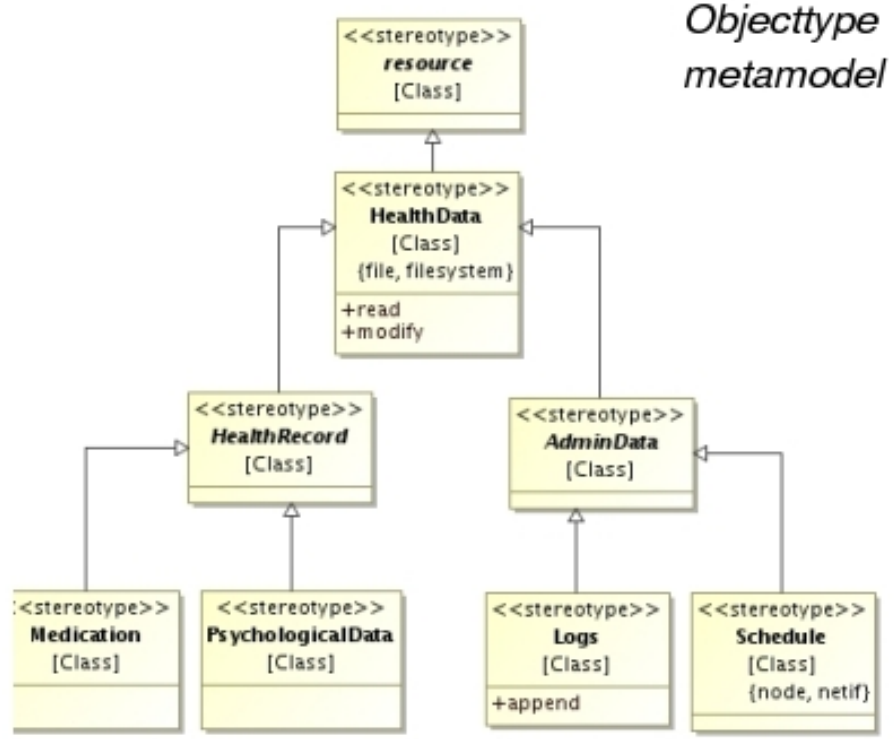


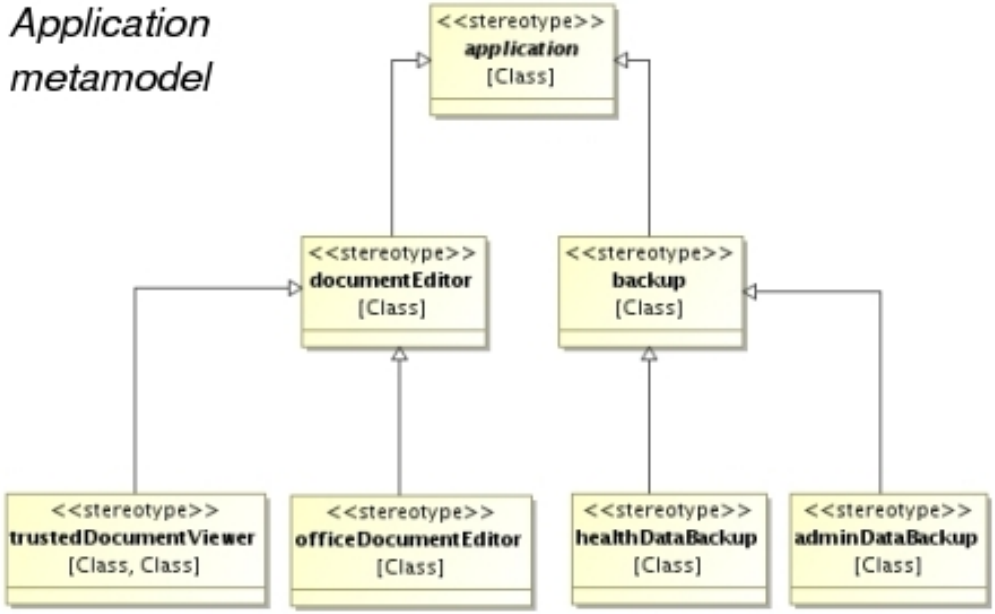
Security Metamodel (1/2)



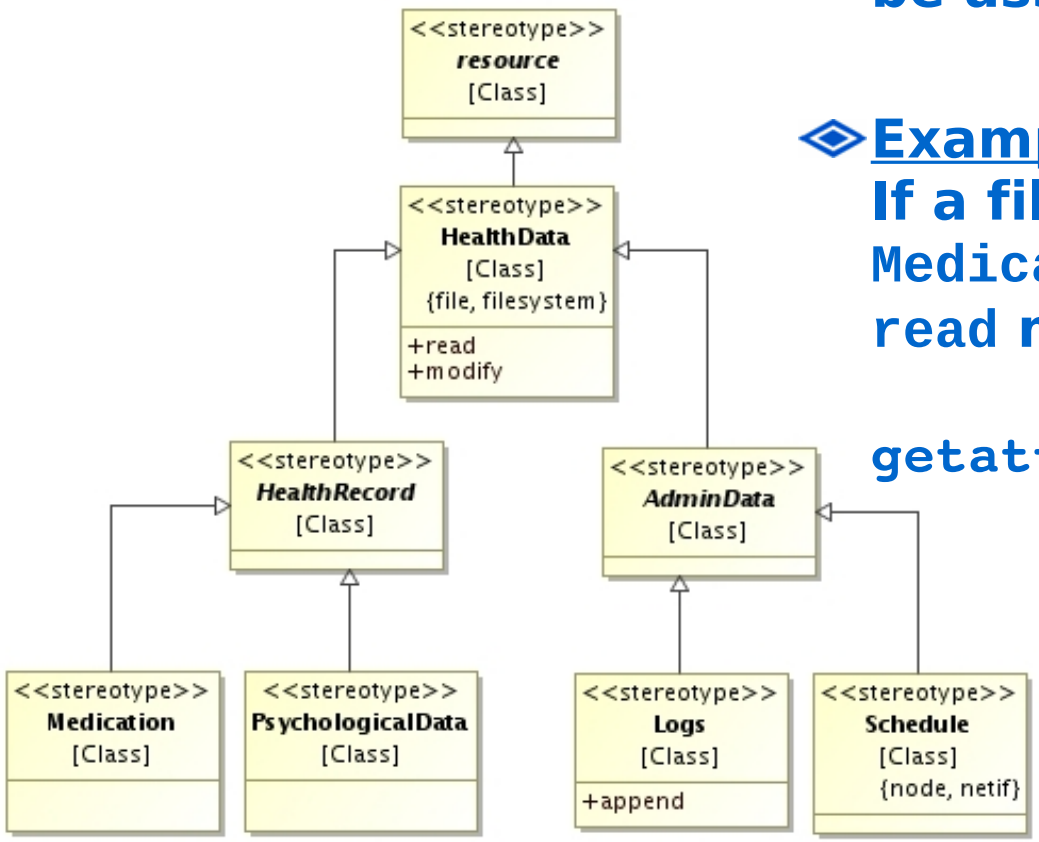
- **The core metamodel**
  - ...is combined with two exchangeable metamodels
  - ...does not change
  - ...defines the basic abstract syntax


Core metamodel
<<stereotype>>
role
[Class]
<<stereotype>>
object
[Class]
<<stereotype>>
resource
[Class]
on
<<stereotype>>
permission
[Association]
hasPermission
<<stereotype>>
application
[Class]


Objecttype metamodel
<<stereotype>>
resource
[Class]
<<stereotype>>
HealthData
[Class]
{file, filesystem}
+read
+modify
<<stereotype>>
HealthRecord
[Class]
<<stereotype>>
AdminData
[Class]
<<stereotype>>
Medication
[Class]
<<stereotype>>
PsychologicalData
[Class]
<<stereotype>>
Logs
[Class]
+append
<<stereotype>>
Schedule
[Class]
{node, netif}


Application metamodel
<<stereotype>>
application
[Class]
<<stereotype>>
documentEditor
[Class]
<<stereotype>>
backup
[Class]
<<stereotype>>
trustedDocumentViewer
[Class, Class]
<<stereotype>>
officeDocumentEditor
[Class]
<<stereotype>>
healthDataBackup
[Class]
<<stereotype>>
adminDataBackup
[Class]

Template Creation



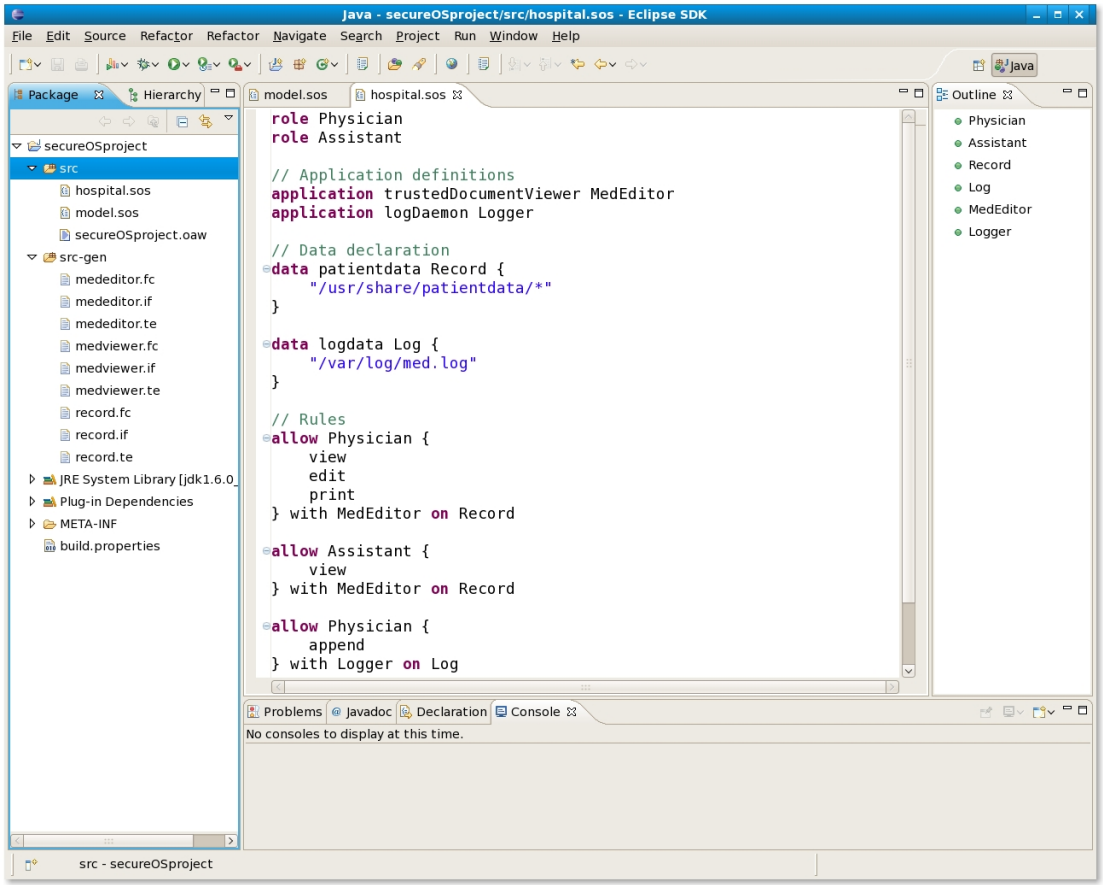
- **Abstract objects have to be assigned semantics**
- **Example:**
  If a file is of type `Medication`, what does `read` mean?

  `getattr, read, link`

Hierarchical Resources

What has already been achieved?

- Abstraction from underlying policy language
- Sample policy template for example application
- Policy generator and textual editor (OAW)



Ideas and Concluding Thoughts

QUALITY ENGINEERING

- ***Close semantic gap between application level security requirements and its low level enforcement***
- **Use concepts from model-driven development**
- **Type creation, polymorphism on object types?**
  - SELinux types are not “typed”
- **Domain specific language for modelling security requirements**
  - No SELinux expert needed for policy development
  - E.g. corporate IT-landscape
- **Can be extended for object manager “generation”**
- **How to evaluate?**

QUALITY ENGINEERING

Thanks!

Berthold Agreiter

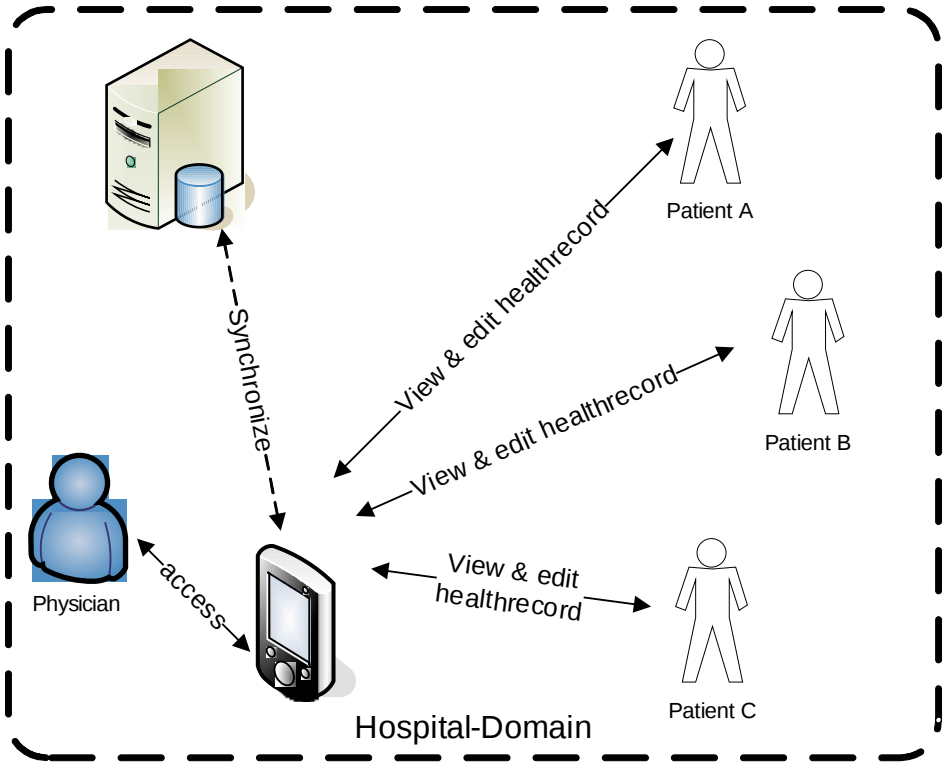
berthold.agreiter@uibk.ac.at

Research Group Quality Engineering
University of Innsbruck

QUALITY ENGINEERING

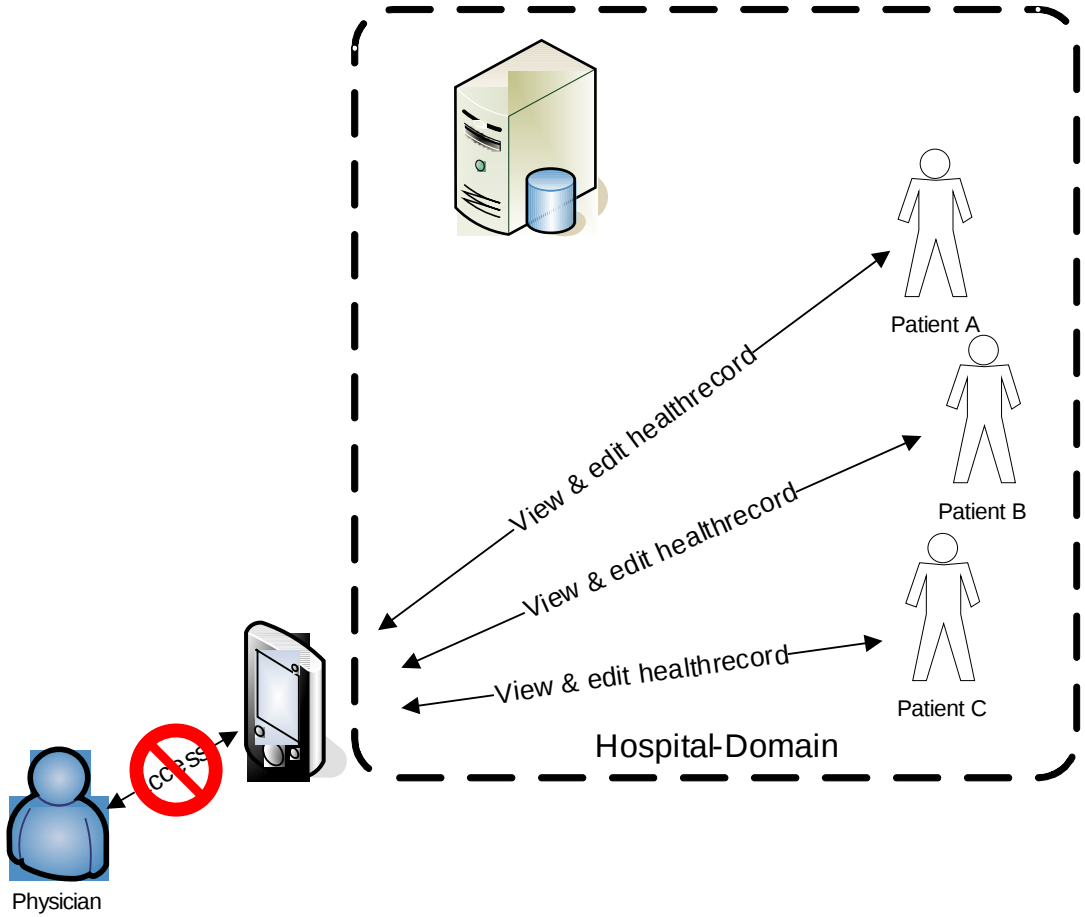
Backup Slides

Example: Mobile Healthcare



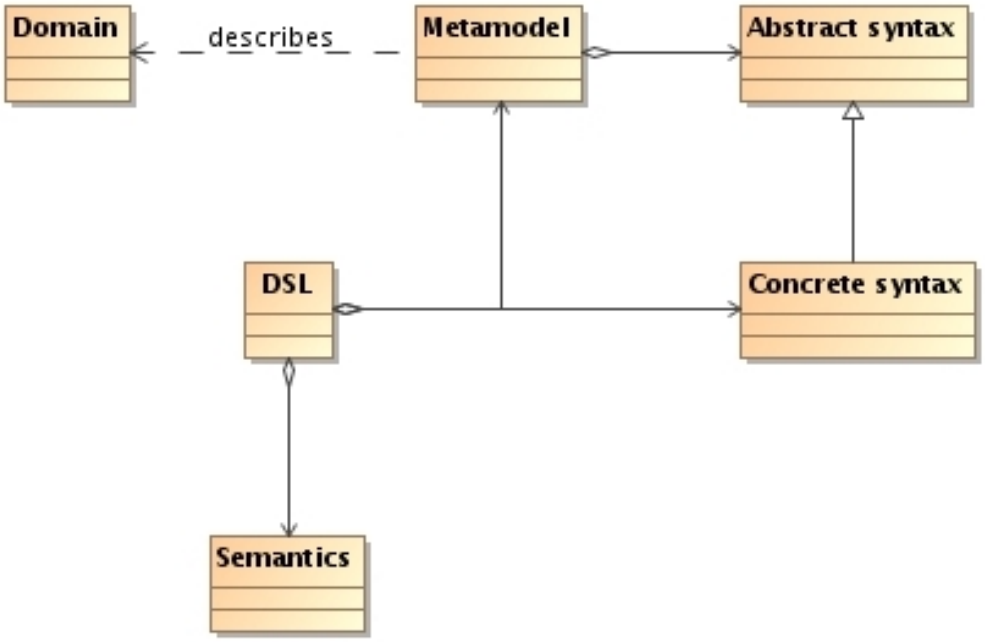
- Access to health-record is restricted by:
  - Role of accessor
  - Contextual conditions (e.g., time, location)
  - ...

Example continued



Physician unable to access healthrecords although stored on his mobile device

Metamodelling



- **Formal model can be instanciated from metamodel**
  - In terms of concrete syntax
  - With semantics assigned