Lightweight Analysis of Access Control Models with Description Logic

Christiano Braga
Universidade Federal Fluminense

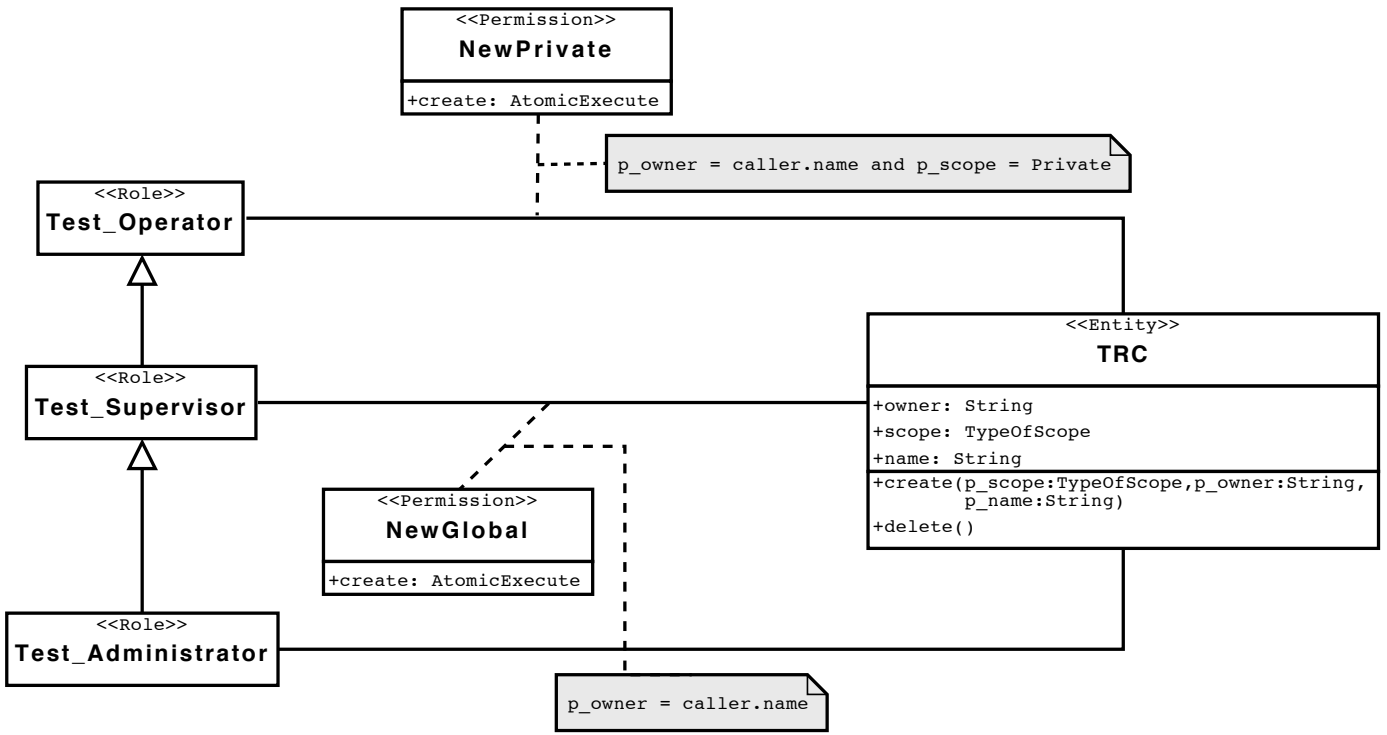
Joint work with E. Hermann Hæusler, PUC-Rio

UML&FM'09

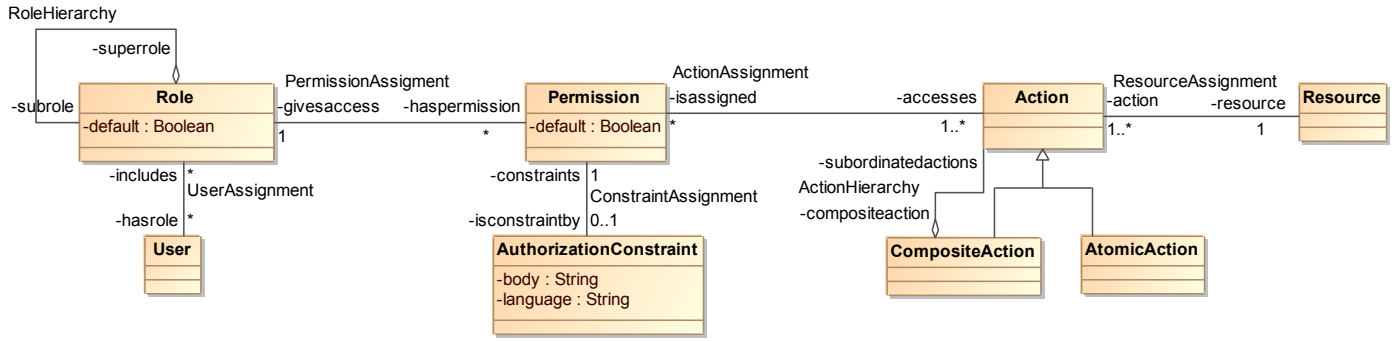
December 08th, 2009
Rio de Janeiro, Brazil

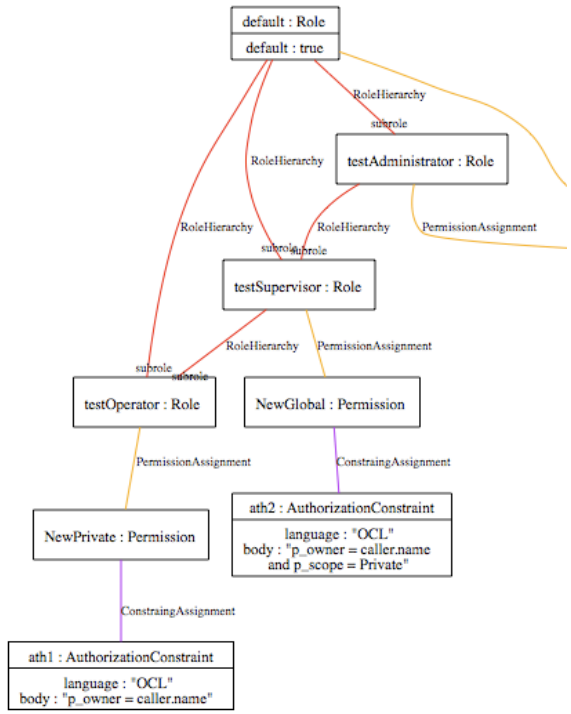
Motivation

- Previous work: validated code generation of access control models to aspects.
- We have defined a model-to-model transformation from access control models specified in the SecureUML modelling language to aspects modeled in the AAC (aspects for access control) modelling language.

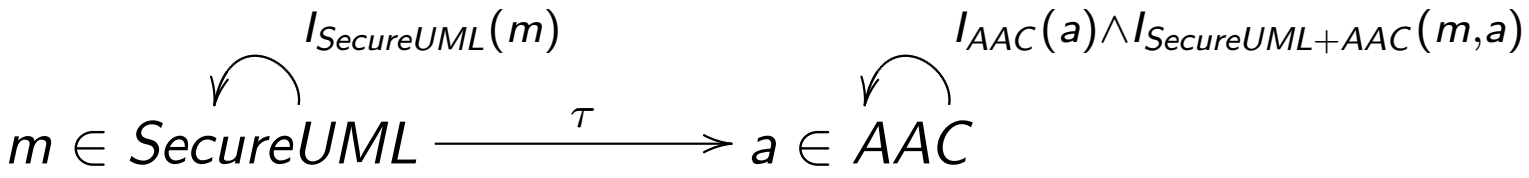


Excerpt of the TRC policy as an instance of the SecureUML metamodel





The transformation process



- where
  - m and a are models instances of the metamodels of SecureUML and AAC, respectively,
  - τ is the transformation from SecureUML abstract syntax to AAC abstract syntax, and
  - $I_{\mathcal{L}}$ represent the OCL invariants of the metamodel $\mathcal{L}$.

Summary of our transformation approach

- Code generation is a model-to-model transformation.
- Validation: OCL invariants are applied to source and target models and also to the *pair* formed by the source and target models.
- OCL invariants from the source metamodel represent well-formedness rules of a given model wrt. to the source metamodel. The same for the target metamodel.
- The OCL invariants applied to the pair formed by the source and target metamodels represent the specification of the transformation. (We call such invariants the *transformation contract* of τ.)

The model consistency problem

- Our validation approach only makes sense if the metamodels, the source and target models involved in the transformation are *consistent*.
- A consistent model is one that has a scenario in conformance with it.
- Consistency, in the presence of OCL invariants, is, in general, undecidable, since the problem reduces to decidability of first-order logic formulæ.

$\mathcal{DL}$ formalization

- This is where $\mathcal{DL}$ comes into place.
- $\mathcal{DL}$ is a family of logics, fragments of FOL, designed to be decidable and as efficient as possible.
- Each fragment allows for the specification and reasoning of increasingly expressive theories.
- For instance, $\mathcal{ALCN}$ extends the $\mathcal{ALN}$ with general concept negation.

SecureUML $\mathcal{DL}$ formalization

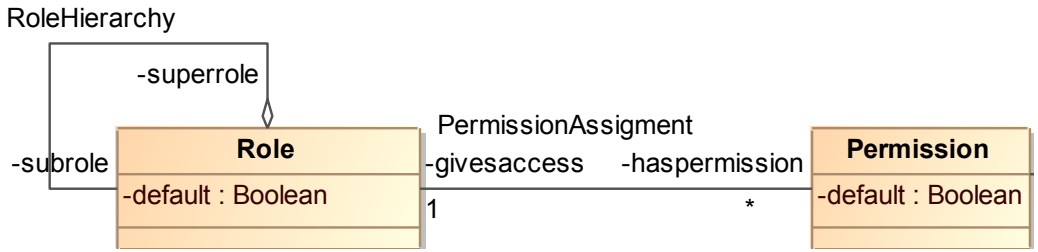
- In this work, we have formalized the SecureUML metamodel in $\mathcal{DL}$, together with its OCL invariants.
- This formalization allows for:
  - consistency checking of SecureUML metamodel, proving it allows for the specification of access control policies and
  - consistency checking of a given access control policy m expressed in SecureUML, that is, a model instance of the SecureUML metamodel, proving that there are possible scenarios of m.
- A combination of consistency checking, provided by $\mathcal{DL}$, together with the OCL validation, gives us more confidence that the transformation is correct.

Mapping UML models to $\mathcal{DL}$ knowledge bases

- $\mathcal{DL}$ has two basic elements: concepts and roles.
- Concepts formalize classes and roles formalize association ends.
- A $\mathcal{DL}$ knowledge base has two parts: a terminology box (TBOX) and an assertion box (ABOX).
- The TBOX is described using concept (and role) subsumptions, which is essentially concept (and role) inclusion.
- The ABOX is described using constants that instantiate the concepts and roles in the TBOX.
- A TBOX represents a UML class model and an ABOX represents an instance model.

$\mathcal{DL}$ specification of SecureUML roles

- We started with a standard mapping (Berardi, Calvanesi and de Giacomo) from UML class model to a TBOX.



$$
\begin{aligned}
\texttt{giveaccess} &\equiv \texttt{haspermission}^- \\
\texttt{subrole} &\equiv \texttt{superrole}^- \\
\top &\sqsubseteq \forall \texttt{haspermission.Permission} \sqcap \\
&\quad\ \forall \texttt{giveaccess.Role} \\
\top &\sqsubseteq \forall \texttt{superrole.Role} \sqcap \forall \texttt{subrole.Role} \\
\texttt{Permission} &\sqsubseteq \exists \texttt{giveaccess.Role} \sqcap (\leq 1\ \texttt{givesaccess})
\end{aligned}
$$

$\mathcal{DL}$ specification of the defaultRole invariant

- For each OCL invariant we have extended the TBOX resulting from the application of the default mapping to the SecureUML metamodel with new assertions.
- For the defaultRole invariant:
  - The first four assertions specify that there exists only one default Role.
  - The last assertion specify that there exists a defualt Role in the transitive closure of superrole.

context Role **inv** defaultRole :
self.allinstances() $\rightarrow$ select(r | r.default) $\rightarrow$ size() $= 1$
self.superrolePlus() $\rightarrow$ exists(r | r.default)

$$
\begin{aligned}
\top &\sqsubseteq \forall \mathtt{isdefault}.\mathtt{A} \sqcap \forall \mathtt{isdefault}^{-}.\mathtt{Role} \\
\forall \mathbf{id}(\top).\mathtt{A} &\sqsubseteq \exists \mathbf{id}(\top).\mathtt{A} \\
\mathtt{A} &\equiv (= 1\ \mathtt{isdefault}^{-}).\mathtt{Role} \\
(\leq 1\ \mathtt{isdefault}^{-}).\mathtt{Role} &\equiv \neg\top \\
\mathtt{Role} &\equiv \exists \mathtt{superrole}.\mathtt{Role} \sqcup \exists \mathtt{isdefault}.\mathtt{A}.
\end{aligned}
$$

The ABOX of the role hierarchy for the TRC policy

$Role(ta)$ $Role(ts)$ $Role(to)$ $Role(dr)$
$hasdefaultRole(dr)$
$superrole(ta, ts)$ $superrole(ts, to)$
$superrole(ts, dr)$ $superrole(to, dr)$ $superrole(ta, dr)$

- Note that if a security policy p does not say anything about a default role, for instance, our $\mathcal{DL}$ analysis would not complain about it, that is, if the assertion *hasdefaultRole(dr)* is not present in the knowledge base, the consistency check would not complain.

Lessons learned

- OCL limitations: we can not prove general properties using OCL lightweight validation, that is, executing OCL invariants over instance models. With $\mathcal{DL}$ we can.
- $\mathcal{DL}$ limitations: due to $\mathcal{DL}$ open world semantics, in a given ABOX, the absence of an individual does not prove a theory wrong. With OCL we can.
- Lightweight OCL+$\mathcal{DL}$ validation is an interesting analysis tool.

Final remarks

- We have presented a $\mathcal{DL}$-based approach for the lightweight analysis of access control policies.
- From this experiment we have learned that consistency check and lightweight OCL validation are complementary analysis techniques.
- We have applied a domain-specific approach for the $\mathcal{DL}$ specification of OCL constraints. In the future, we would like to relate the different $\mathcal{DL}$ logics with subsets of OCL constructs.