Extending Access Control Models with Break-glass

Achim D. Brucker Helmut Petritsch

{achim.brucker, helmut.petritsch}@sap.com

SAP RESEARCH

Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany

ACM Symposium on Access Control Models and Technologies
(SACMAT 2009)
Stresa, Italy, 5th June 2009

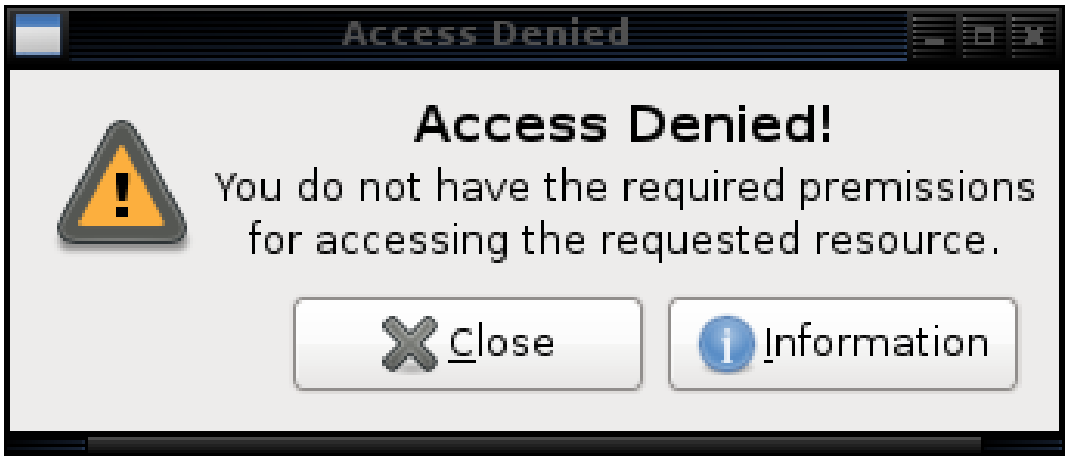
Outline

Our Vision

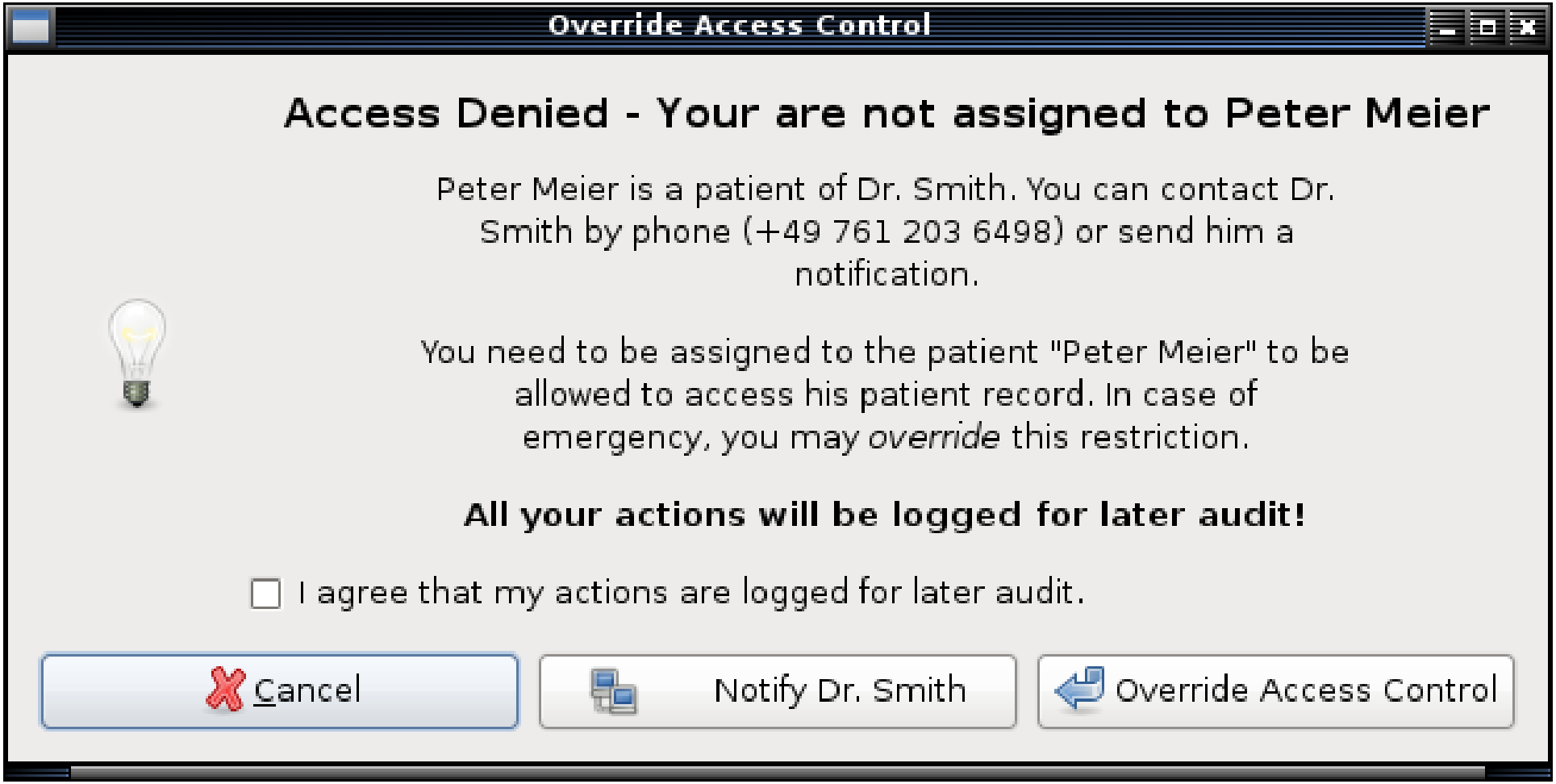
Assume,

- we are a nurse
- trying to access the patient record of Peter Meier …

Our Vision


Access Denied
Access Denied!
You do not have the required premissions
for accessing the requested resource.
Close
Information

Our Vision


Override Access Control
Access Denied - Your are not assigned to Peter Meier
Peter Meier is a patient of Dr. Smith. You can contact Dr. Smith by phone (+49 761 203 6498) or send him a notification.
You need to be assigned to the patient "Peter Meier" to be allowed to access his patient record. In case of emergency, you may override this restriction.
All your actions will be logged for later audit!
I agree that my actions are logged for later audit.
Cancel
Notify Dr. Smith
Override Access Control

Break-glass or Overriding Access Control

While often motivated with

- health care or
- public security

scenarios, also enterprises demand break-glass solutions:

- for preventing stagnation on the system administration level and
- for preventing stagnation on the business process level.

In fact, state of the art enterprise systems support break-glass, e.g.,

- Virsa Firefighter for SAP,
- Oracle's Role Manager.

The Situation Today

Mostly implemented using pre-staged accounts that are

- either stored in sealed covers or
- electronically issued on request.

Break-glass solutions should cover

- the creation of break-glass accounts,
- the distribution pre-staged accounts,
- the monitoring of the use of break-glass accounts, and
- the cleanup after an break-glass situation.

This solution is

- quite coarse-grained and
- not integrated into regular access control.

Outline

Observations and Goals

- During discussions with end users, we observed:
  - depending on the situation, different overrides can be justified
  - some restrictions can never be overridden
- The two main design goals are:
  - access-control decisions should be overrideable on a per permission basis and
  - fine-grained configuration of the restrictions that can be overridden.

Emergeny Levels

Definition

A policy p *refines* a policy p' (written $p \sqsubseteq p'$) if and only if the set of system traces that are allowed under p is a subset of the system traces that are allowed under p'.

- A policy p refines a policy p' iff p is at least as restrictive as p'.
- $p^\top$ is the policy that allows all actions and
- $p^\bot$ is the policy that denies all actions.
- $p^\bot$ refines all policies and every policy is a refinement of $p^\top$.
- $P_{\mathcal{A}}$ be the set of all policies of the access control model $\mathcal{A}$.
- $(P_{\mathcal{A}}, \sqsubseteq, p^\bot, p^\top)$ is a lattice.

Regular Policies and Emergeny Policies

Definition

We refer to the *regular policy*, i. e., the policy that should be obeyed in normal operations, as p^{reg} and we refer to the set of policies that are refined by the regular policy, i. e.,

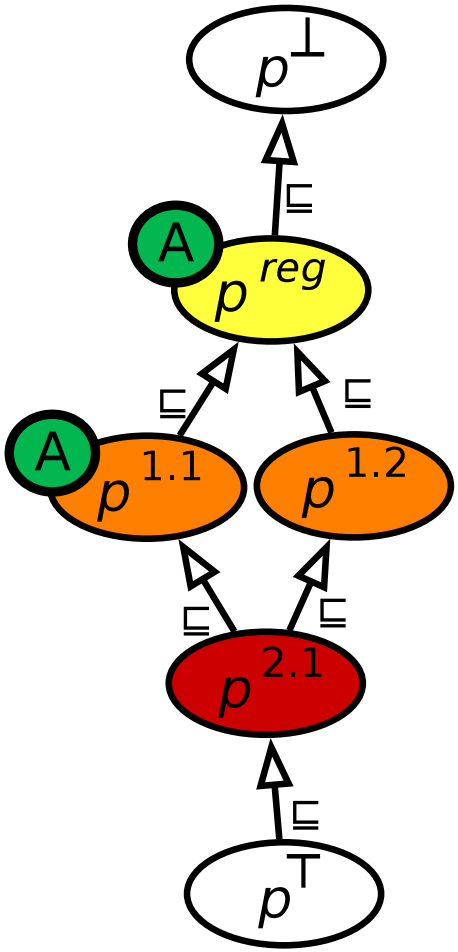
$$L_{\mathcal{A}} = \{p \mid p \in P_{\mathcal{A}} \wedge p^{\text{reg}} \sqsubseteq p \wedge p \neq p^{\text{reg}}\}$$

as *emergency levels* or *emergency policies* of the policy p^{reg}. We require that $(P_{\mathcal{A}} \setminus p^{\perp}, \sqsubseteq, p^{\text{reg}}, p^{\top})$ is a lattice, i. e., $\inf(P_{\mathcal{A}} \setminus p^{\perp}) = p^{\text{reg}}$.

- An emergency level can be *active* or *inactive*.
- Only active emergency levels contribute to the access control decision.
- The regular policy is always active.

Hierarchical Break-glass Access Control

- An access that is only granted by an emergency policy $\ell \in L_{\mathcal{A}}$ is called *override access*.
- Override accesses are only granted if there is an active policy granting access.
- *Obligations* can be attached to an (emergency) policy, i.e., requiring user confirmations or for activating monitoring.
- By evaluating the policies in topological order, the refinement relation holds **by construction**!



Outline

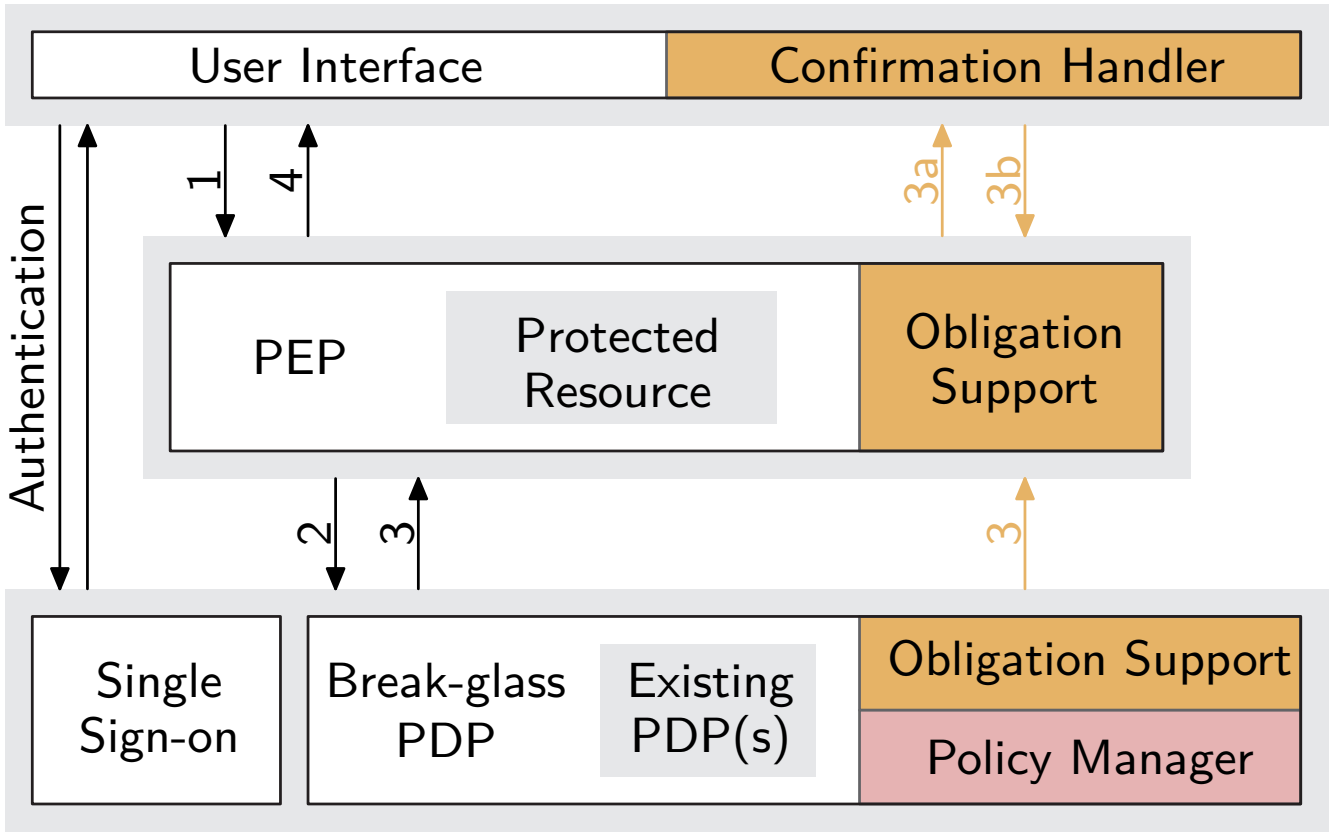
Break-glass Architecture: Main Idea

The break-glass policy combination strategy can be implemented by a meta PDP.

- The **Break-glass PDP** implements the break-glass policy combination strategy on top of existing PDPs
- User confirmations can be implemented using obligations:
  - the various PDPs need to support obligations
  - the various PEPs need to support obligations
  - the user interface needs to support confirmation requests

Break-glass does not impose restrictions on the underlying access control model!

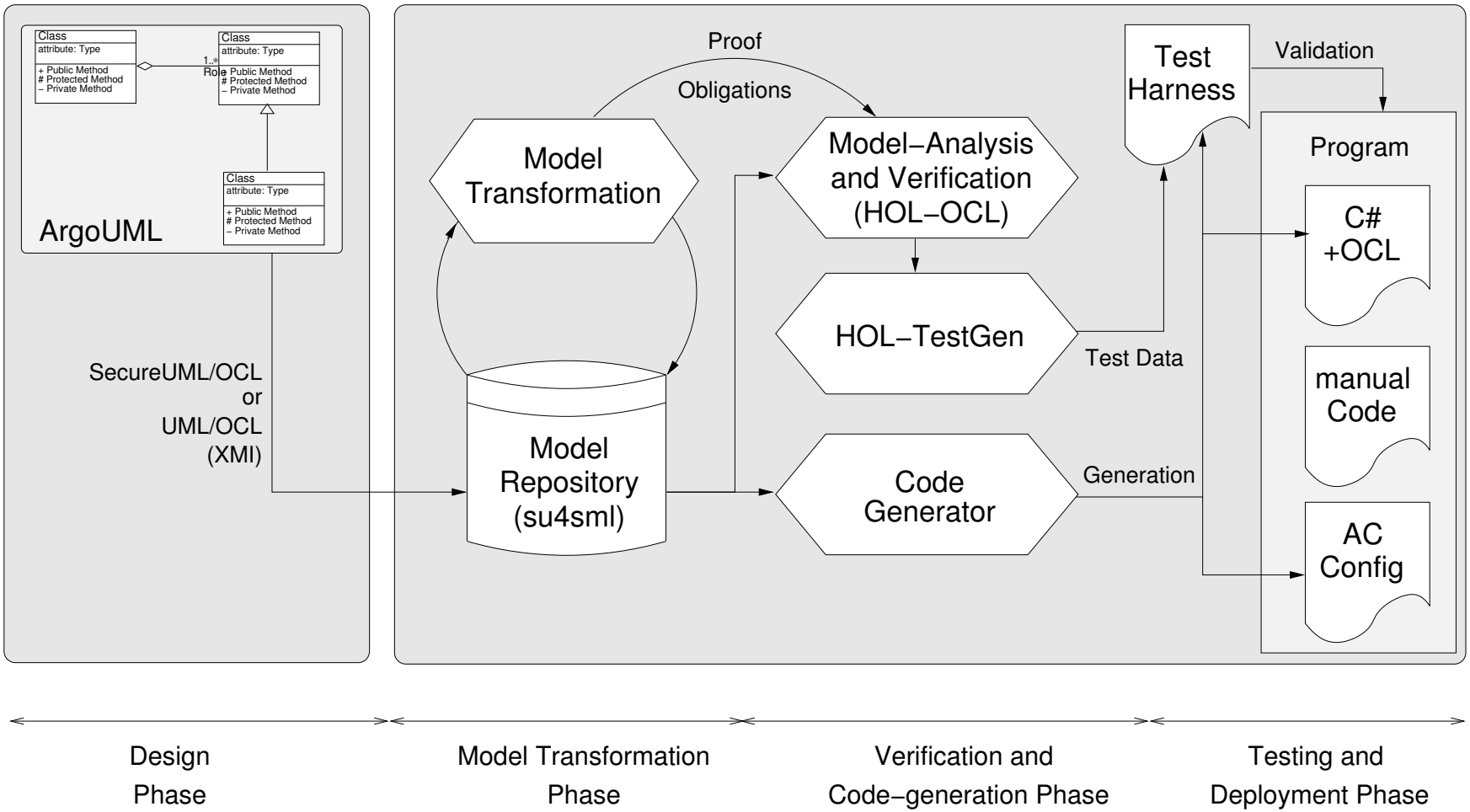
A Generic Break-glass Architecture



Outline

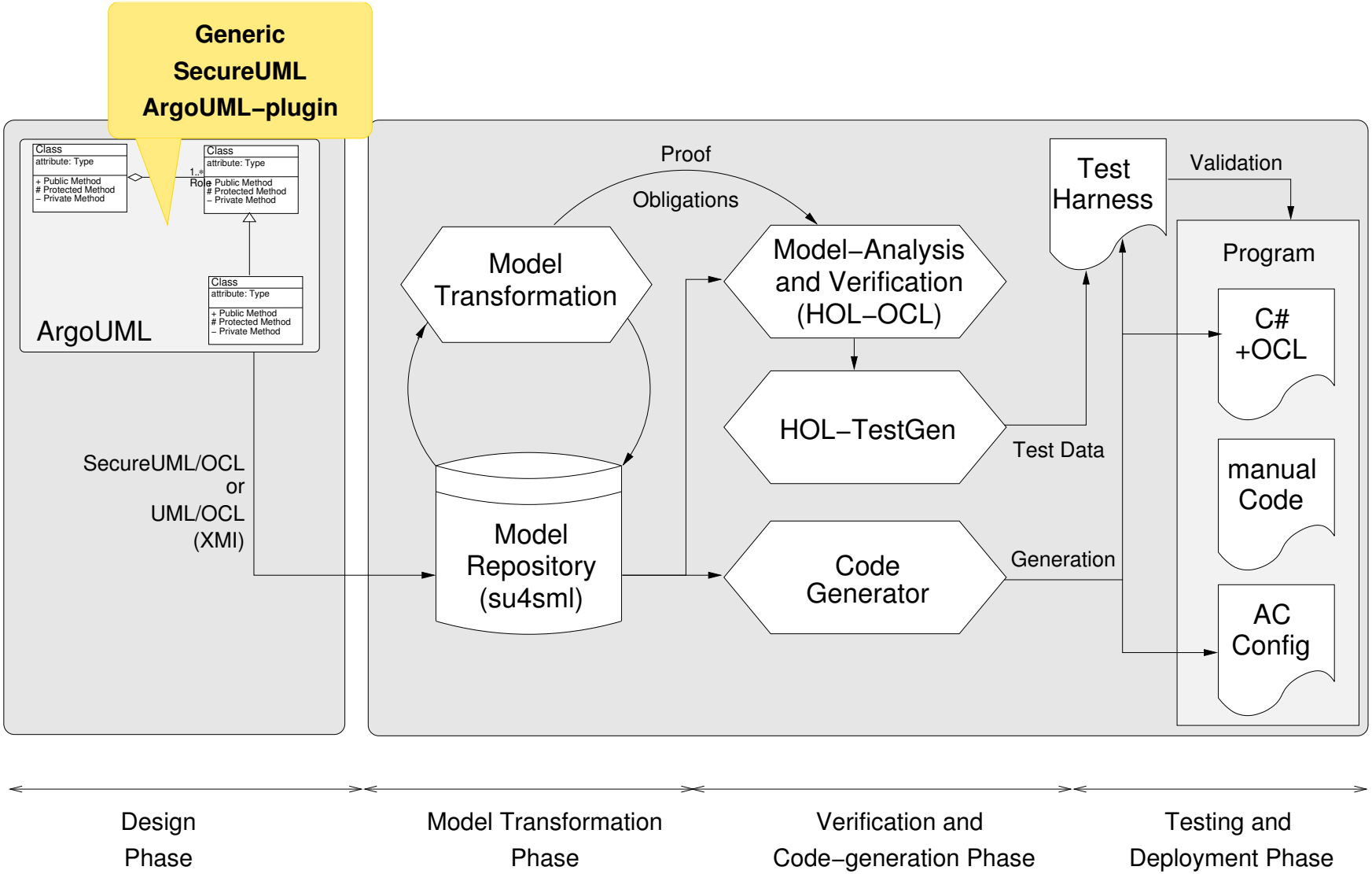
The Model-driven Security Vision

A Tool-supported and Security-aware Formal Model-driven Engineering Process



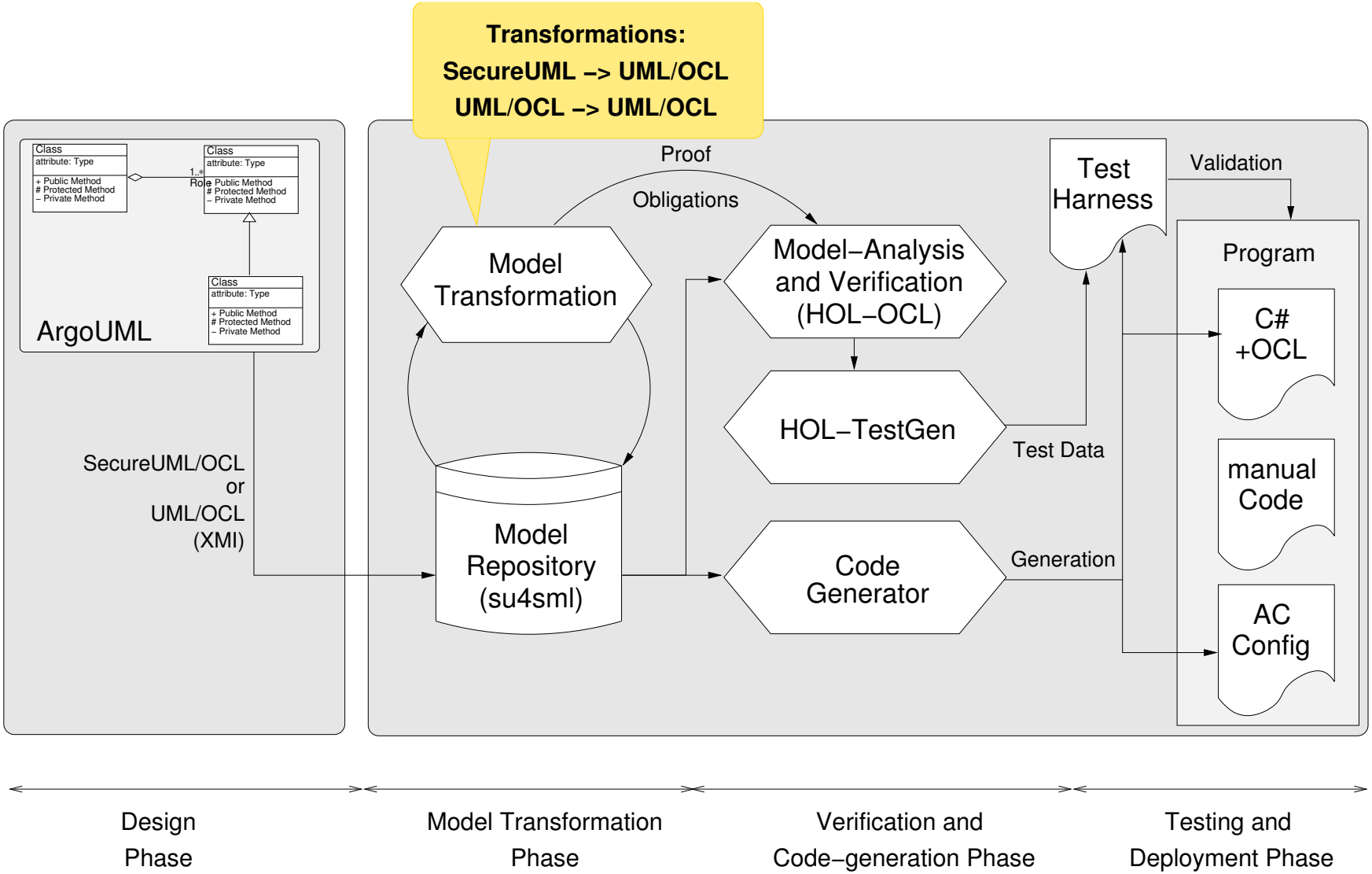
The Model-driven Security Vision

A Tool-supported and Security-aware Formal Model-driven Engineering Process



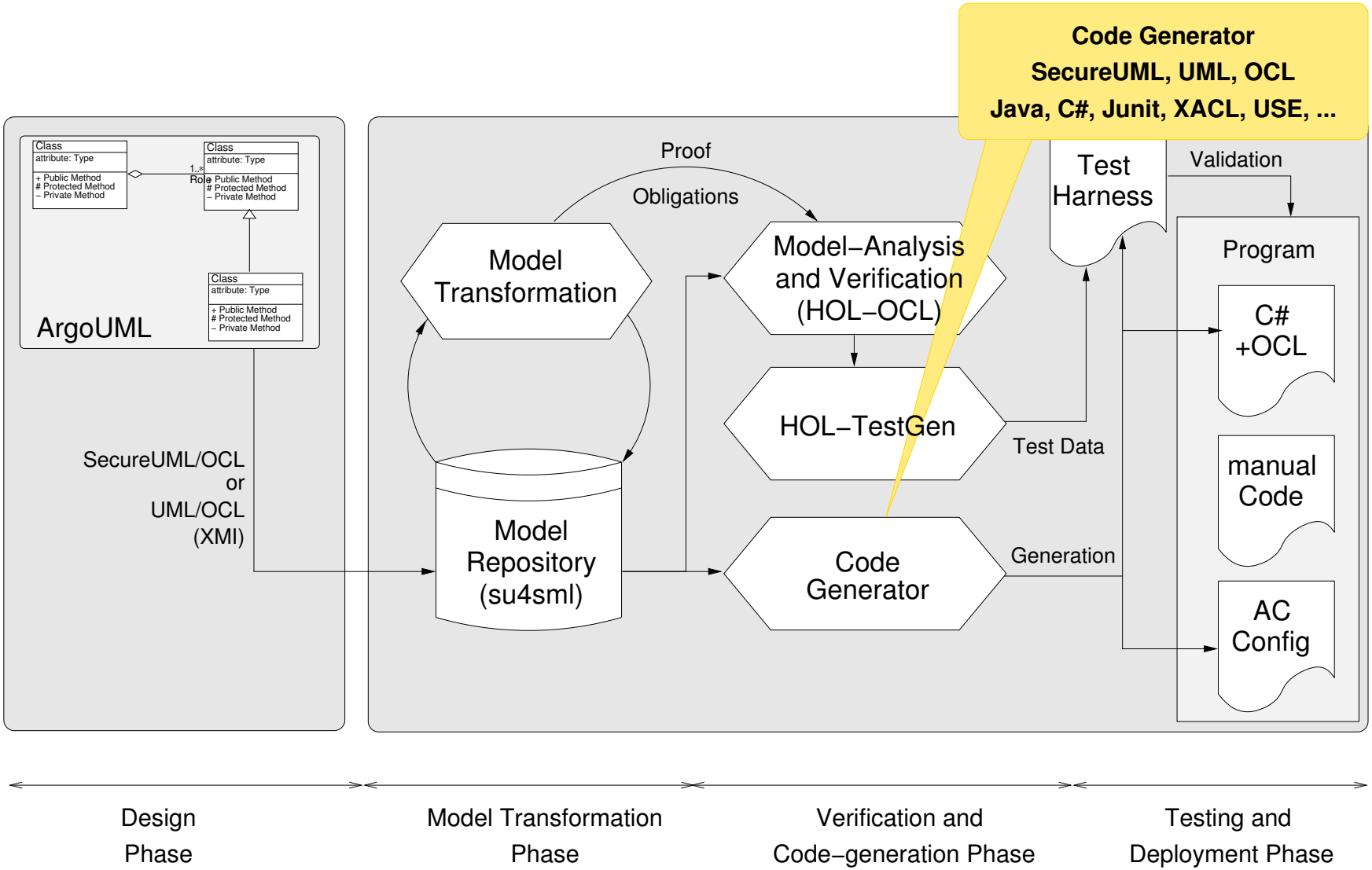
The Model-driven Security Vision

A Tool-supported and Security-aware Formal Model-driven Engineering Process



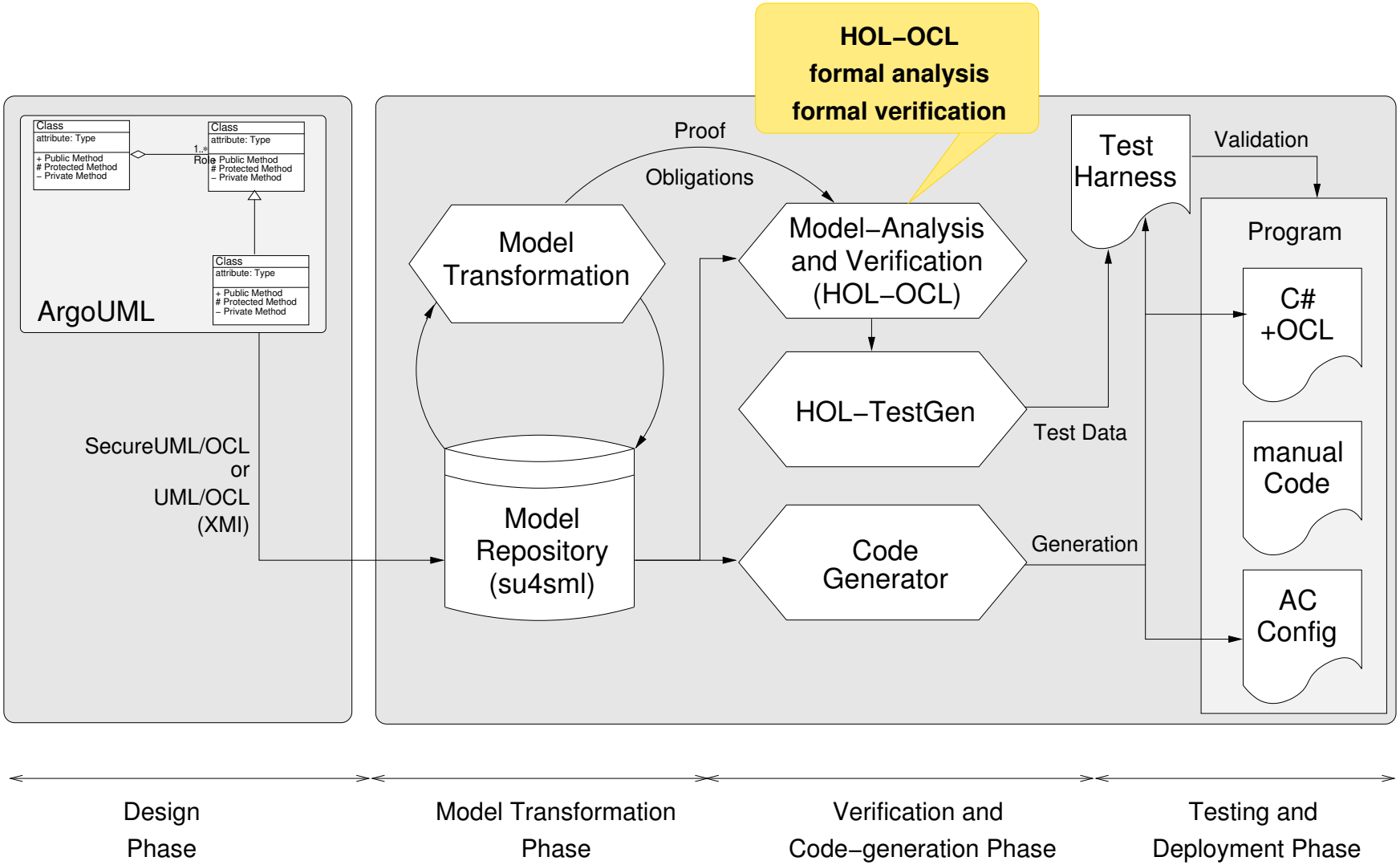
The Model-driven Security Vision

A Tool-supported and Security-aware Formal Model-driven Engineering Process



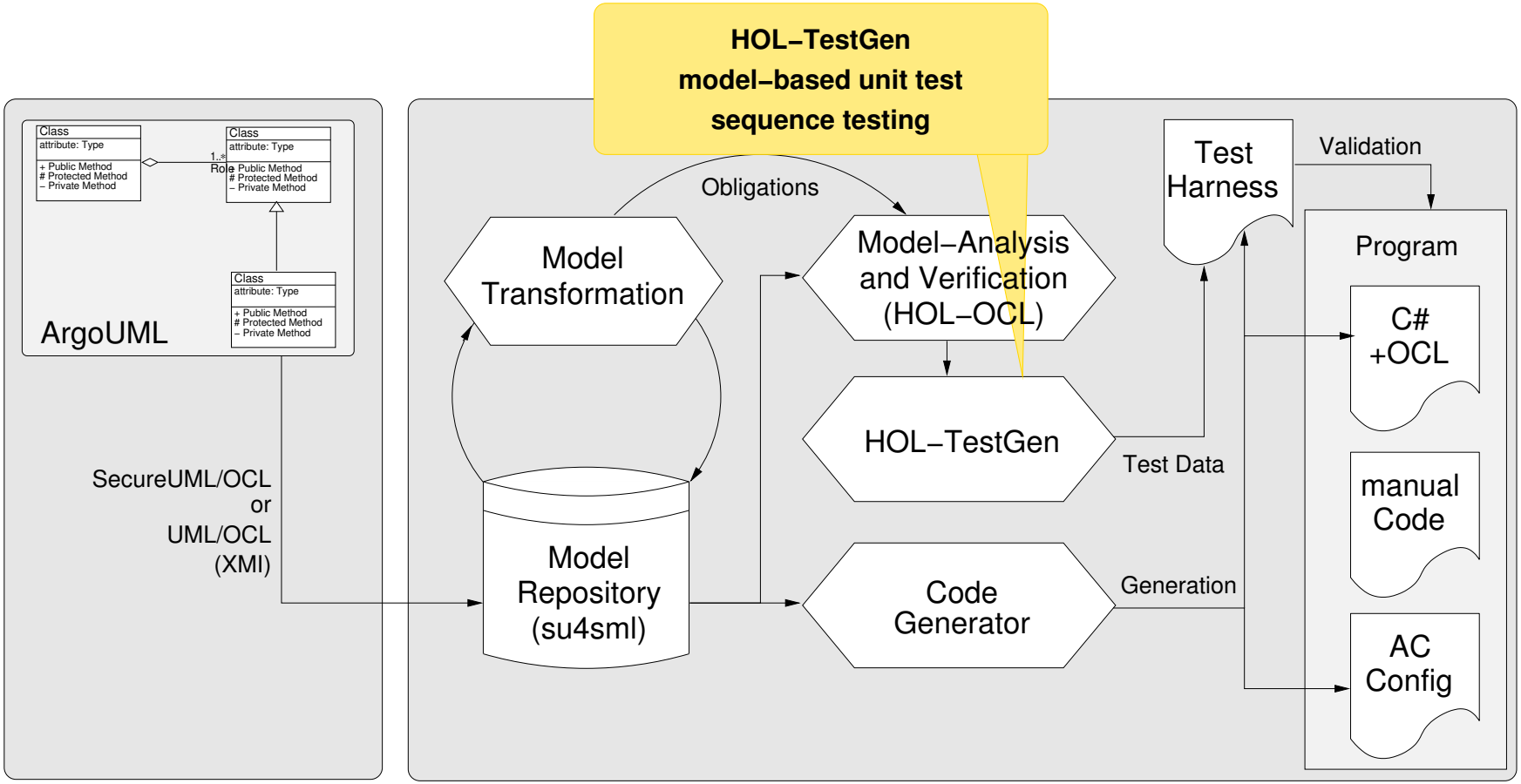
The Model-driven Security Vision

A Tool-supported and Security-aware Formal Model-driven Engineering Process



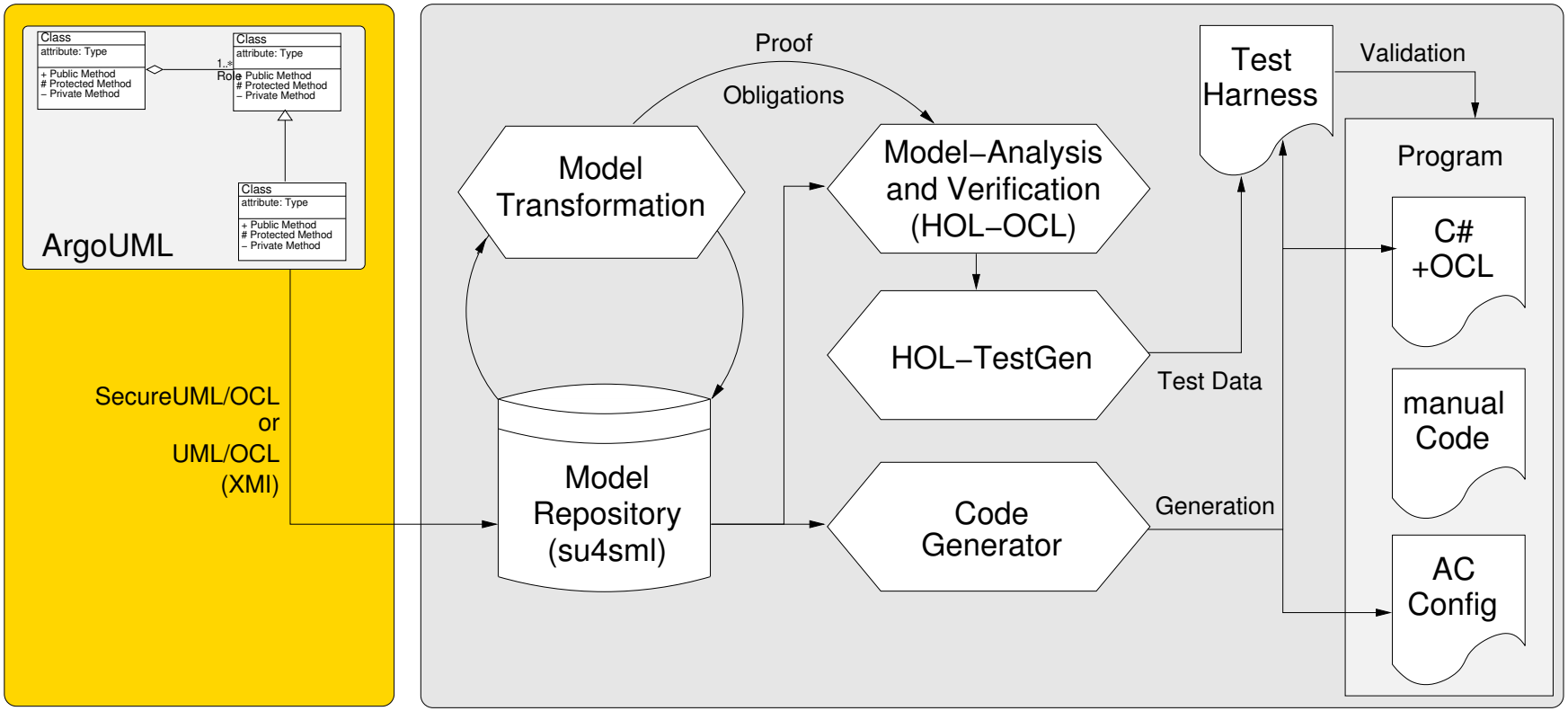
The Model-driven Security Vision

A Tool-supported and Security-aware Formal Model-driven Engineering Process

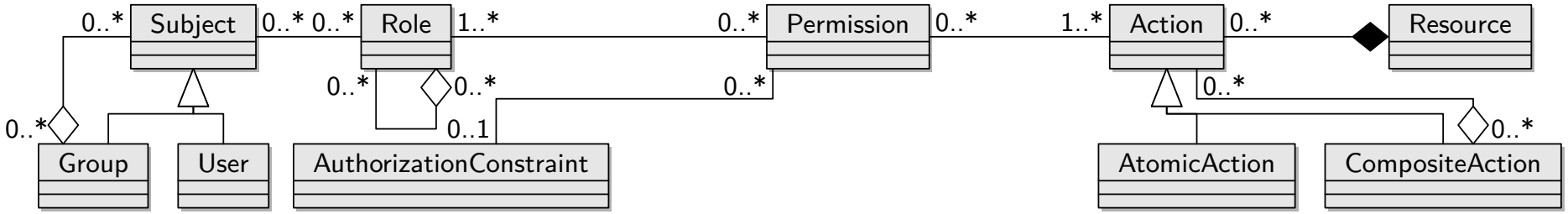


The Model-driven Security Vision

A Tool-supported and Security-aware Formal Model-driven Engineering Process



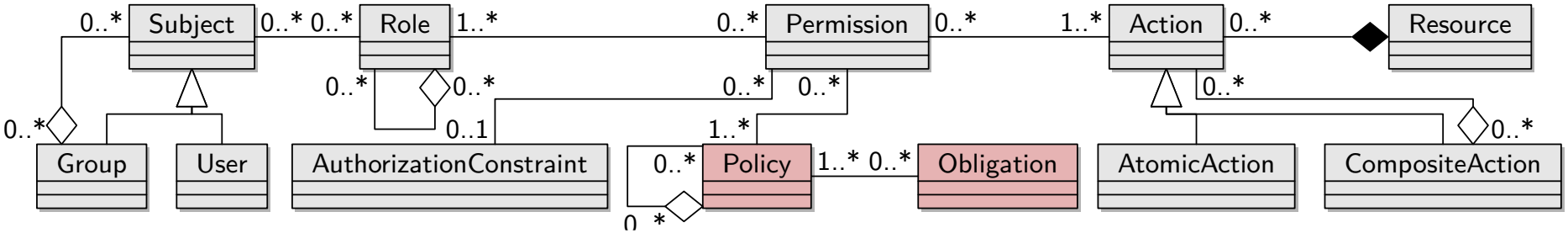
SecureUML



SecureUML

- is a UML-based notation,
- provides abstract Syntax given by MOF compliant metamodel,
- is pluggable into arbitrary design modeling languages,
- is supported by an ArgoUML plugin.

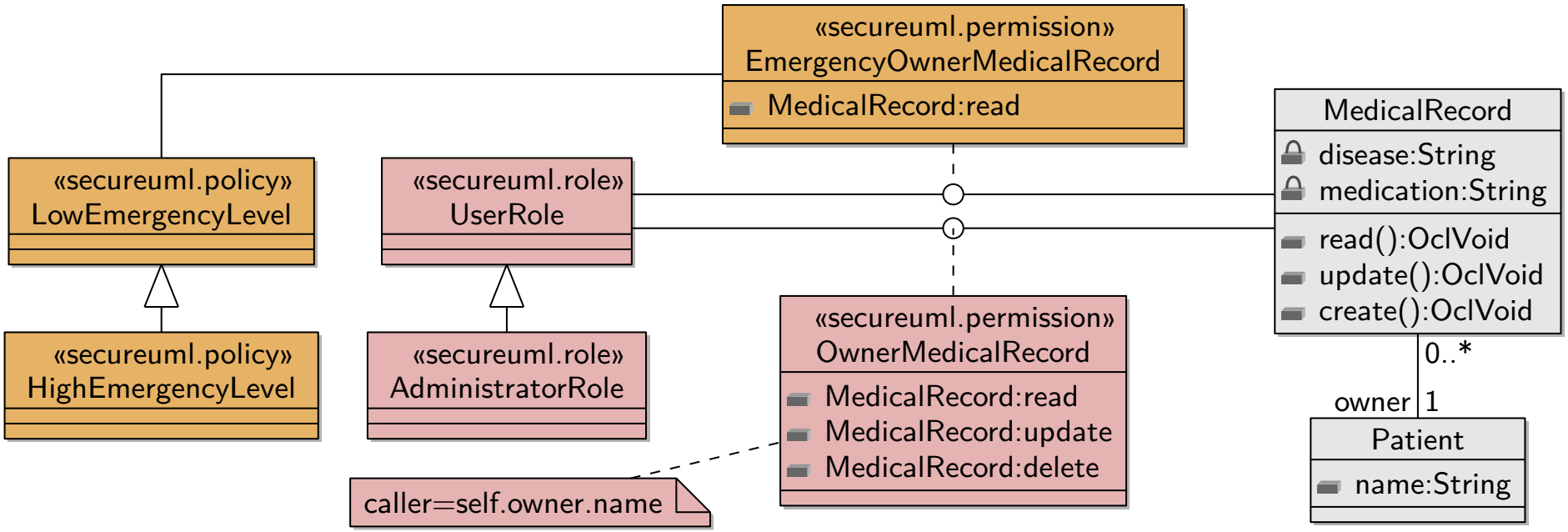
SecureUML



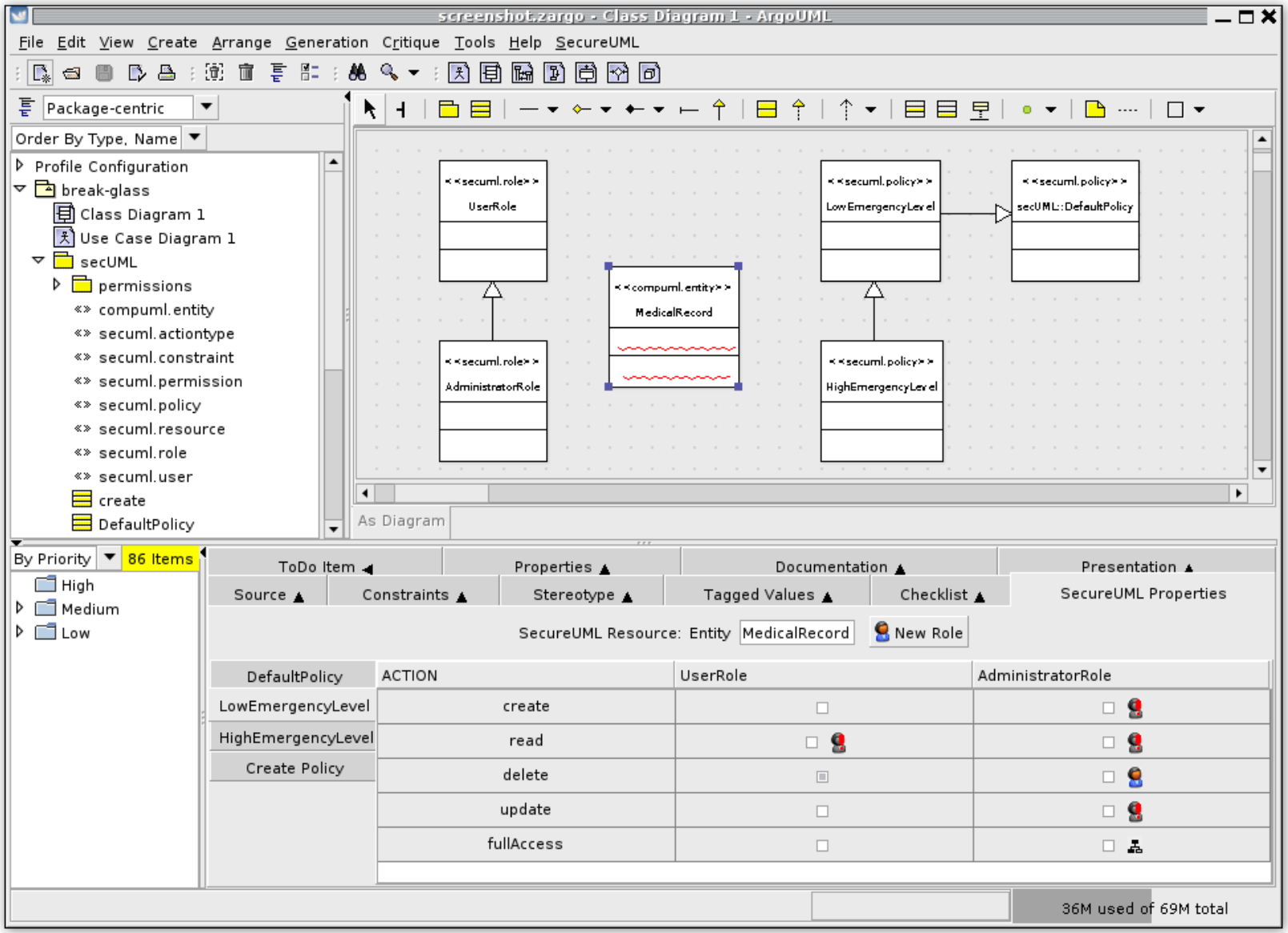
SecureUML

- is a UML-based notation,
- provides abstract Syntax given by MOF compliant metamodel,
- is pluggable into arbitrary design modeling languages,
- is supported by an ArgoUML plugin.
- can easily be extended with support for **break-glass**.

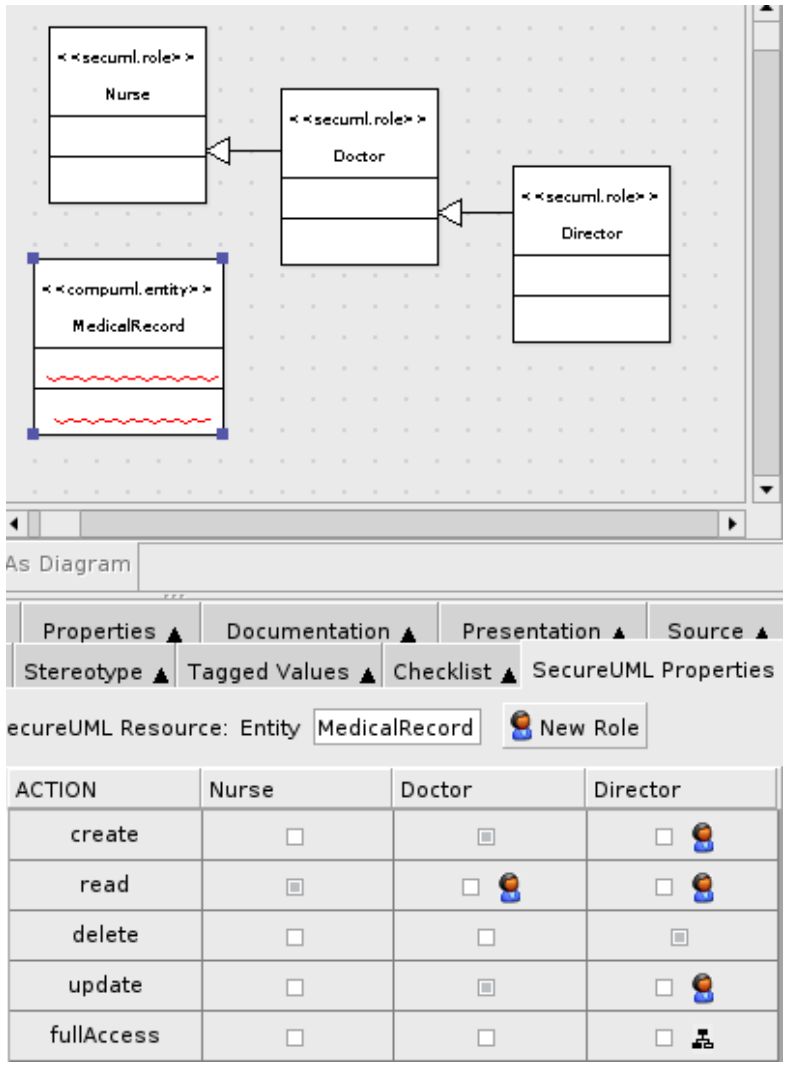
Modeling Access Control with SecureUML



ArgoUML Support



ArgoUML Support



Code Generation (Java and XACML)

- In case of XACML, we can generate
  - the policies and
  - the PDP configuration.
- In particular, we
  - sort the policies topological,
  - use the "first-applicable" combining algorithm of XACML, and
  - exploit the obligations support of XACML.
- With respect to the application, we generate
  - (stubs of) the business logic,
  - the calls to PDP, and
  - the PEP.

Outline

Conclusion and Future Work

We presented a

- a generic break-glass model that allows the fine-grained, overriding of access control decisions,
- an generic architecture for implementing break-glass,
- an extension of SecureUML supporting break-glass, and
- the mapping of break-glass to XACML

Future work includes the integration and development of

- analysis techniques for user providing feedback to the user,
- break-glass concepts for IT compliance, and
- techniques for a posteriori analysis of incidents.

Thank you
for your attention!

Any questions or remarks?

Bibliography

Achim D. Brucker and Helmut Petritsch.

Extending access control models with break-glass.

In *ACM symposium on access control models and technologies (SACMAT)*, pages 197–206. ACM Press, 2009.