Constructing Tool-support for Sophisticated Analysis of UML Models

Jan Jürjens

Software & Systems Engineering
TU Munich, Germany

juerjens@in.tum.de

http://www.jurjens.de/jan

Personal Introduction + History

Me: Leading the Competence Center for IT-Security at Software & Systems Engineering, TU Munich

- Extensive collaboration with industry (HypoVereins-bank, T-Systems, Munich Re, BMW, Deutsche Bank, Allianz, Siemens, Infineon, …)
- PhD in Computer Science from Oxford Univ., Masters in Mathematics from Bremen Univ.
- Numerous publications incl. 1 book on the subject

This tutorial: part of series of 30 tutorials at international conferences. Continuously improved (please fill in feedback forms).

Software Development

Quality vs. Cost:

In software development:
Correctness in conflict with cost.

Thorough methods of system design not used if too expensive.

In particular: critical systems.

Towards Solution

Increase quality with bounded investment in time, costs. Idea:

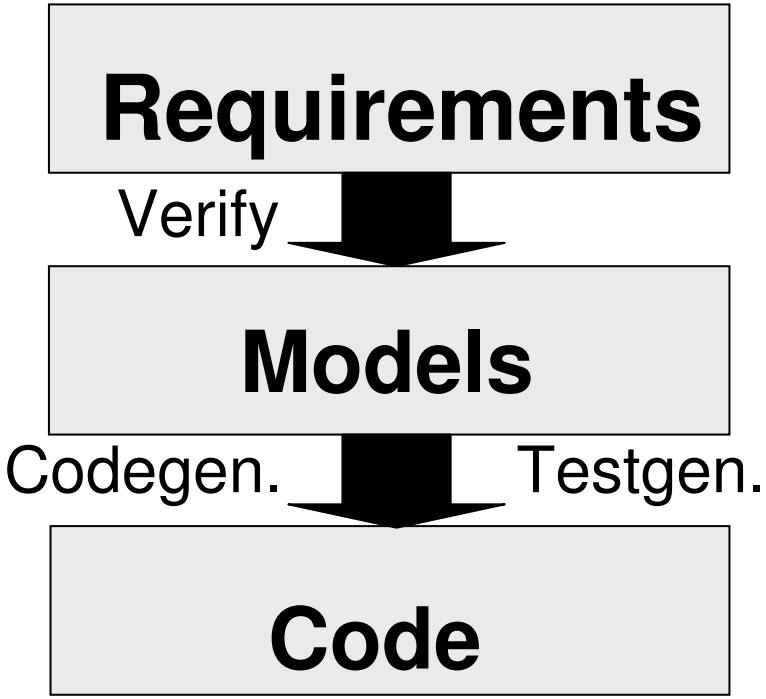
- Extract models from artefacts arising in industrial development and use of software systems (UML models, source code, configuration data).
- Tool-supported theoretically sound efficient automated critical analysis.

➔ *Model-based Software Development*

Model-based Development

Combined strategy:

- Verify models against requirements
- Generate code from models where reasonable
- Write code and generate test-sequences otherwise.



Using UML

UML: unprecedented opportunity for high-quality and cost- and time-efficient software development:

- De-facto standard in industrial modeling: large number of developers trained in UML.
- Relatively precisely defined (given the user community).
- Many drawing tools etc.

Challenge

Advanced tool support. For example:

- consistency checks
- mechanical analysis of complicated requirements on model level (bindings to model-checkers, constraint solvers, automated theorem provers, …)
- code generation
- test-sequence generation
- configuration data analysis against UML.

This Tutorial

Background knowledge on constructing tool-support for sophisticated analysis of UML models.

- Drawing tools
- Tool-bindings
- The CSDUML framework
- Example application (crypto checker)
- Other approaches, UML 2.0

Discussion

Requirements on advanced tool-support for UML ?

Roadmap

Prologue

UML drawing tools

Tool-bindings

The CSDUML framework

Example application (crypto checker)

Other approaches, UML 2.0

Tool-support: Pragmatics

Commercial modelling tools: so far mainly syntactic checks and code-generation.

Goal: sophisticated analysis. Solution:

- Draw UML models with editor.
- Save UML models as XMI (XML dialect).
- Connect to verification tools (automated theorem prover, model-checker …), e.g. using XMI Data Binding.

UML Drawing Tools

Wide range of existing tools.

Consider some, selected under following criteria (Shabalin 2002):

- Support for all relevant diagram types.
- Support for custom UML extensions.
- Availability (test version, etc).
- Prevalence on the market.

Some Examples for Tools

- Rational Rose. Developed by major participant in development of UML; market leader.
- Visio for Enterprise Architect. Part of Microsoft Developer Studio .NET.
- Together. Often referenced as one of the best UML tools.
- ArgoUML. Open Source Project, therefore interesting for academic community. Commercial variant Poseidon.

Comparison

Evaluated features:

Support for custom UML extensions.

- Model export; standards support; tool interoperability.
- Ability to enforce model rules, detect errors, etc.
- User interface quality.
- Possibility to use the tool for free for academic institutions.

Rational Rose (IBM Rational)

One of the oldest on the market.

+ Free academic license.
+ Widely used in the industry.
+ Export to different XMI versions.
- Insufficient support for UML extensions (custom stereotypes yes; tags and constraints no).
- Limited support for checking syntactic correctness.
- Lack of compatibility between versions and with other Rational products for UML modelling.

Together from TogetherSoft

Widely used in the development community. Very good round-trip engineering between the UML model and the code.

+ Free academic license.
+ Written in Java, therefore platform-independent.
+ Nice, intuitive user interface.
+ Export to different XMI versions; recommendations which for which tool.
- Insufficient support for UML extensions (custom stereotypes yes; tags and constraints no).

Visio from Microsoft Corporation

Has recently been extended with UML editing support

+ Good user interface
+ Full support for UML extensions
+ Very good correspondence to UML standard.
  Checks dynamically for syntactic correctness;
  suggestions for fixing errors
- No free academic license
- Proprietary, undocumented file format;
  very limited XMI export
- No round-trip engineering support.
  No way back after code generation

ArgoUML / Poseidon

ArgoUML: Open Source Project. Commercial extension Poseidon (Gentleware), same internal data format

+ Open Source
+ Written in Java, therefore platform-independent
+ XMI default model format
+ Poseidon: solid mature product with good UML specification support
- Performance

Model Exchange

	import to			
export from	**Poseidon**	**Rational Rose**	**Together**	**Visio**
Poseidon	+	+	-	-
Rational Rose	+	+	+	-
Together XMI 1.1 unisys	(+)	+	+	-
Microsoft Visio	(+)	(+)	(+)	+

Roadmap

Prologue

UML drawing tools

Tool-bindings

The CSDUML framework

Example application (crypto checker)

Other approaches, UML 2.0

Tool-support: Tool Binding

Several possibilities:

- General purpose language with integrated XML parser (Perl, …)
- Special purpose XML parsing language (XSLT, …)
- Data Binding (Castor; XMI: e.g. MDR)

Data-binding with MDR

MDR: MetaData Repository,
Netbeans library (www.netbeans.org)

Extracts data from XMI file into Java Objects, following UML 1.5 meta-model.

Access data via methods on UML level.

Advantage: No need to worry about XML.

Relevant Standards

- MOF (Meta Object Facility)

  Abstract format for describing metamodels.

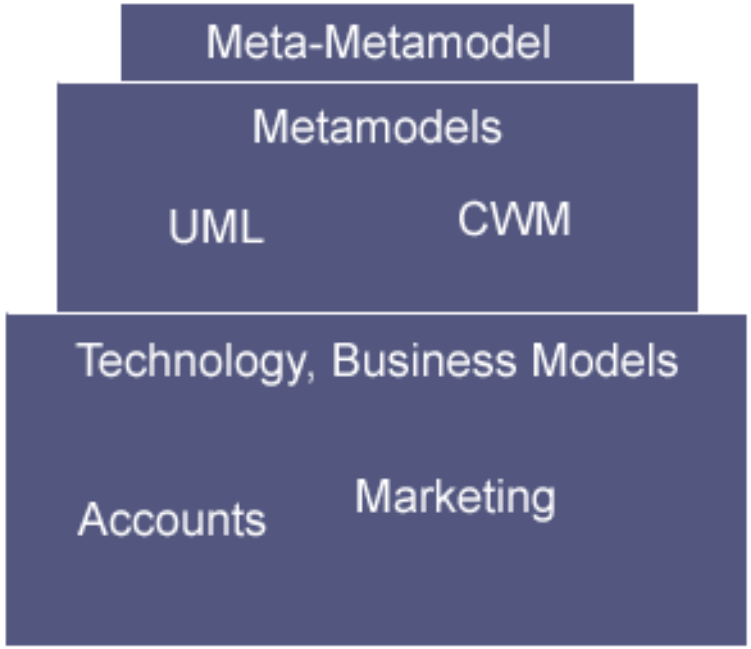
- XMI (XML Metadata Interchange)

  Defines XML format for a MOF metamodel.
  Defined by DTDs.

- JMI (Java Metadata Interface)

  Defines mapping from MOF to Java.

MOF Architecture



- Meta-Metamodel (M3)
  – defined by OMG
- Metamodels (M2)
  – user-defined
  – e.g. UML 1.5, MOF, CWM
- Business Model (M1)
  – instances of Metamodels
  – e.g. UML class diagram
- Data (M0)
  – instance of model
  – e.g. implementation of UML classes in Java

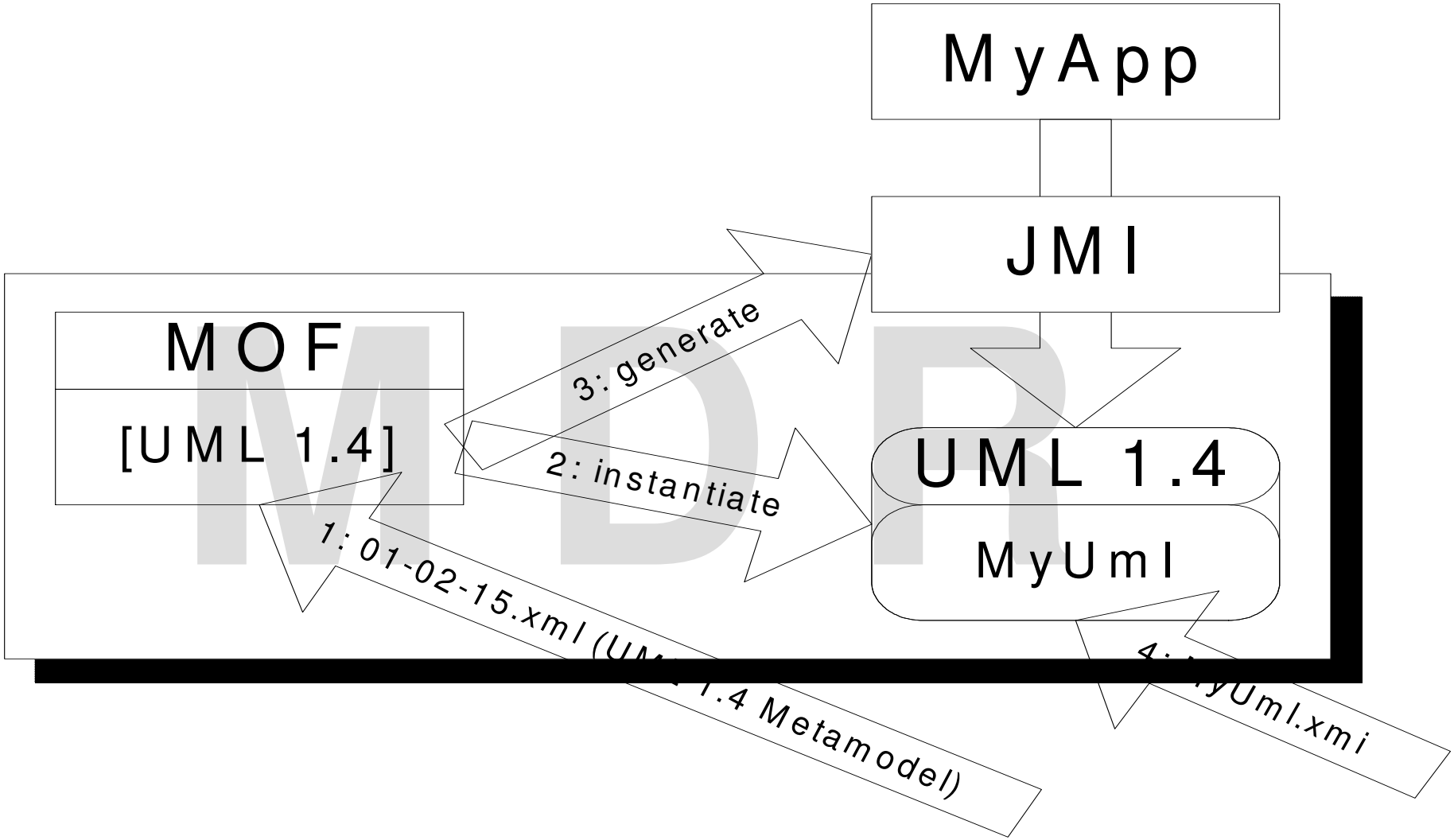
MOF Example

Meta-Metamodel	MetaClass, MetaAssociation - MOF Model
Metamodel	Class, Attribute, Dependency - UML (as language), CWM
Model	Person, House, City - UML model
Data	(Bob Marley, 1975) (Bonn) - Running Program

MDR Services

- Load and Store a MOF Metamodel (XMI format).
- Instantiate and Populate a Metamodel (XMI format).
- Generate a JMI (Java Metadata Interface) Definition for a Metamodel.
- Access a Metamodel Instance.

UML Processing



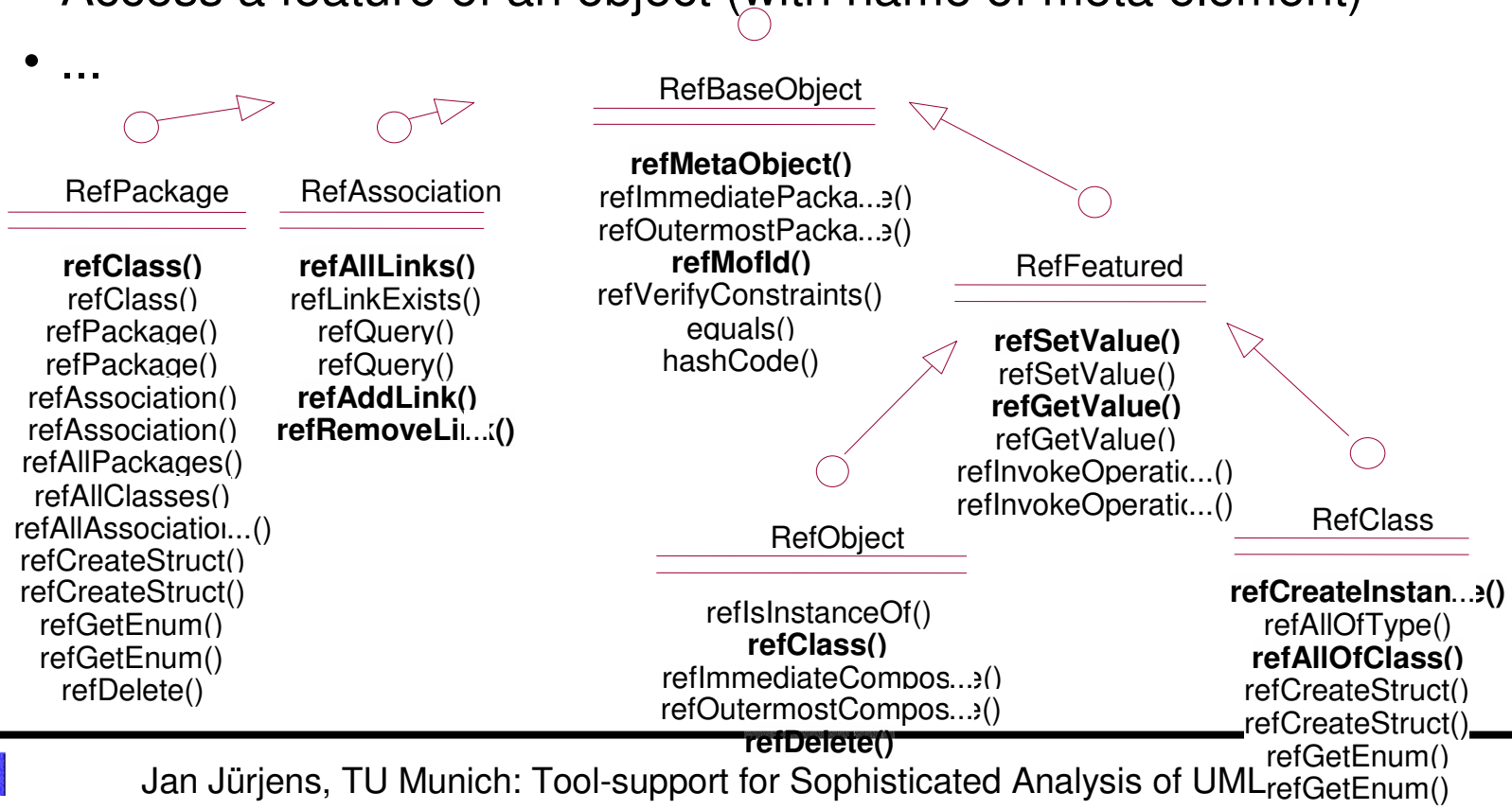
JMI: MOF Interfaces

- IDL mapping for manipulating Metadata
  - API for manipulating information contained in an instance of a Metamodel
  - MOF is MOF compliant !
  - Metamodels can be manipulated by this IDL mapping
  - JMI is MOF to Java mapping
- Reflective APIs
  - manipulation of complex information
  - can be used without generating the IDL mapping
  - MDR has implemented these interfaces

JMI: Reflective Facilities

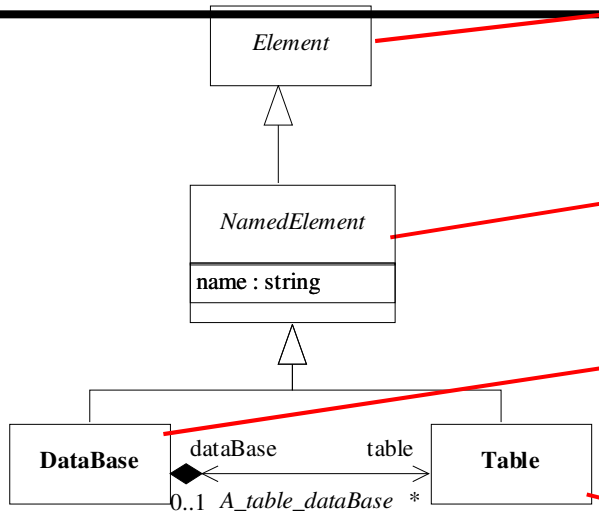
[Fondement 04]

– As defined in the MOF, it is possible to
  - Access the metatype of an object
  - Asks a metatype for each one of its instances
  - Access a feature of an object (with name of meta element)
  - ...



JMI: Generated Interfaces

[Fondement 04]




```
public interface Element
extends javax.jmi.reflect.RefObject {
}
```

```
public interface NamedElement
extends database.Element {
  public String getName();
  public void setName(String newValue);
}
```

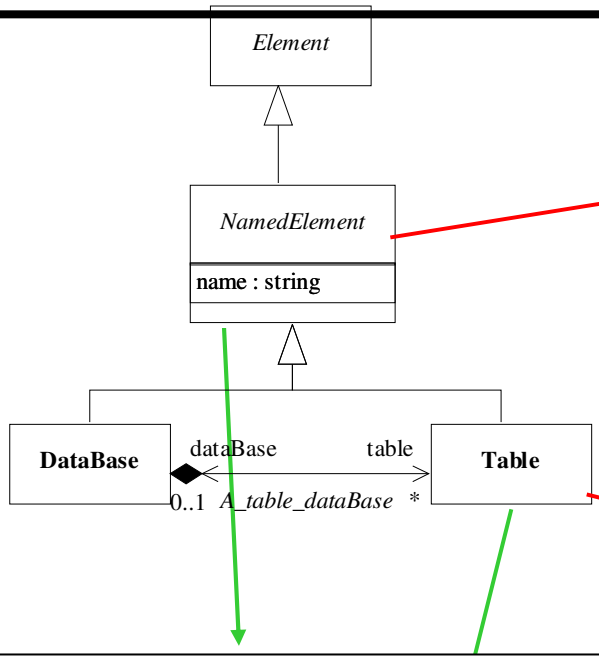
```
public interface DataBase
extends database.NamedElement {
  public Collection getTable();
}
```

```
public interface Table
extends database.NamedElement {
  public DataBase getDataBase();
  public void setDataBase(String newValue);
}
```

- It is possible to access “meta objects” here
  - “Element” interface extends RefObject, so have a RefClass
  - Can access the meta properties of an element
    - Name (any direct “DataBase” instance returns the “DataBase” string)
    - Contents (applied on any NamedElement returns the “name” meta-attribute)
    -

JMI: Generated Interfaces

[Fondement 04]




```
public interface Element
extends javax.jmi.reflect.RefObject {
}
```

```
public interface NamedElement
extends database.Element {
  public String getName();
  public void setName(String newValue);
}
```

```
public interface DataBase
extends database.NamedElement {
  public Collection getTable();
}
```

```
public interface Table
extends database.NamedElement {
  public DataBase getDataBase();
  public void setDataBase(String newValue);
}
```

```
public interface NamedElementClass
extends javax.jmi.reflect.RefClass {
}
```

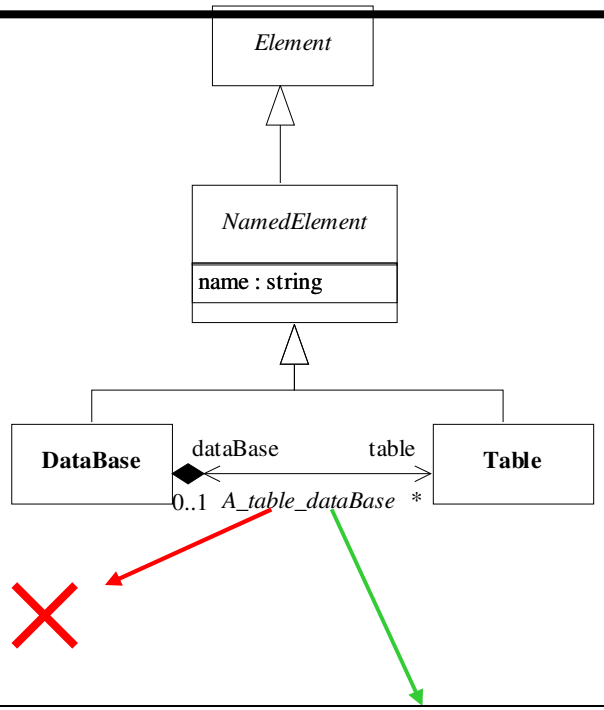
```
public interface TableClass
extends javax.jmi.reflect.RefClass {
  public Table createTable();
  public Table createTable (String name);
}
```

In order to create an object, you must contact its metaclass

A metaclass is a *singleton*

JMI: Generated Interfaces

[Fondement 04]




```
public interface Element
extends javax.jmi.reflect.RefObject {
}
```

```
public interface NamedElement
extends database.Element {
  public String getName();
  public void setName(String newValue);
}
```

```
public interface DataBase
extends database.NamedElement {
  public Collection getTable();
}
```

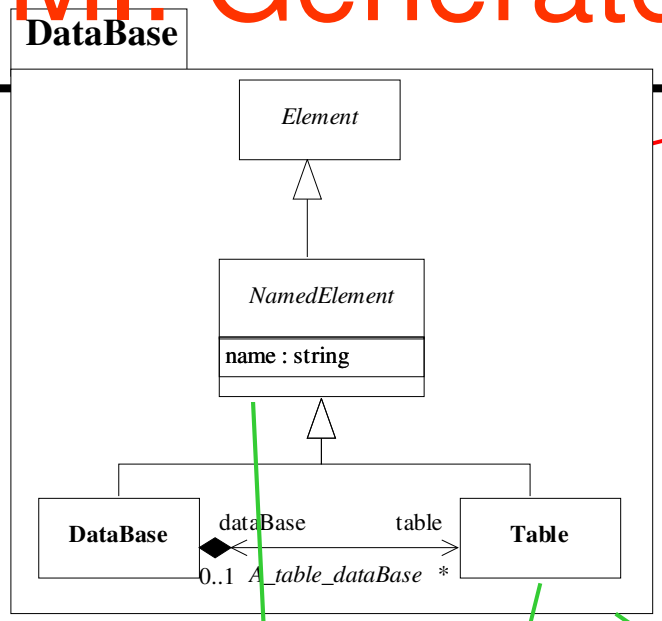
```
public interface Table
extends database.NamedElement {
  public DataBase getDataBase();
  public void setDataBase(String newValue);
}
```

```
public interface ATableDataBase
extends javax.jmi.reflect.RefAssociation {
  public boolean exists(Table table, DataBase dataBase);
  public Collection getTable(DataBase dataBase);
  public DataBase getDataBase(Table table);
  public boolean add(Table table, DataBase dataBase);
  public boolean remove(Table table, DataBase dataBase);
}
```

– A meta association is a *singleton*

JMI: Generated Interfaces

[Fondement 04]



- In order to create an object, you must contact its metaclass
- A meta element is a *singleton* and provide access to its nested meta elements
- The root meta package is the entry point to access these singletons
- Need to be provided a mechanism to retrieve the root package singleton

```
public interface NamedElementClass
extends javax.jmi.reflect.RefClass {
}
```

```
public interface TableClass
extends javax.jmi.reflect.RefClass {
  public Table createTable();
  public Table createTable (String name);
}
```

```
public interface XMLModelPackage
extends javax.jmi.reflect.RefPackage {
  public NodeClass getNode();
  public AttributeClass getAttribute();
  public ElementClass getElement();
  public RootNodeClass getRootNode();
  public TextNodeClass getTextNode();
  public Contains getContains();
}
```

MDR Repository: Loading Models

- Metamodel is instance of another Metamodel
- Loading Model = Loading Metamodel
- Needed Objects:
  - MDRepository
  - MofPackage
  - XMISaxReaderImpl

- Java Code-Snippet:

```
MDRepository rep;
UmlPackage uml;
// Objekte erzeugen:
rep =
   MDRManager.getDefault().getDefaul
   tRepository();
reader =
   (XMISaxReaderImpl)Lookup.getDefau
   lt().lookup(XmiReader.class);
// loading extent:
uml =
   (UmlPackage)rep.getExtent(„name“)
   ;
// creating Extent:
uml =
   (UmlPackage)rep.createExtent(„nam
   e“);
// loading XMI:
reader.read(„url“, MofPackage);,
```

MDR Repository: Reading Data

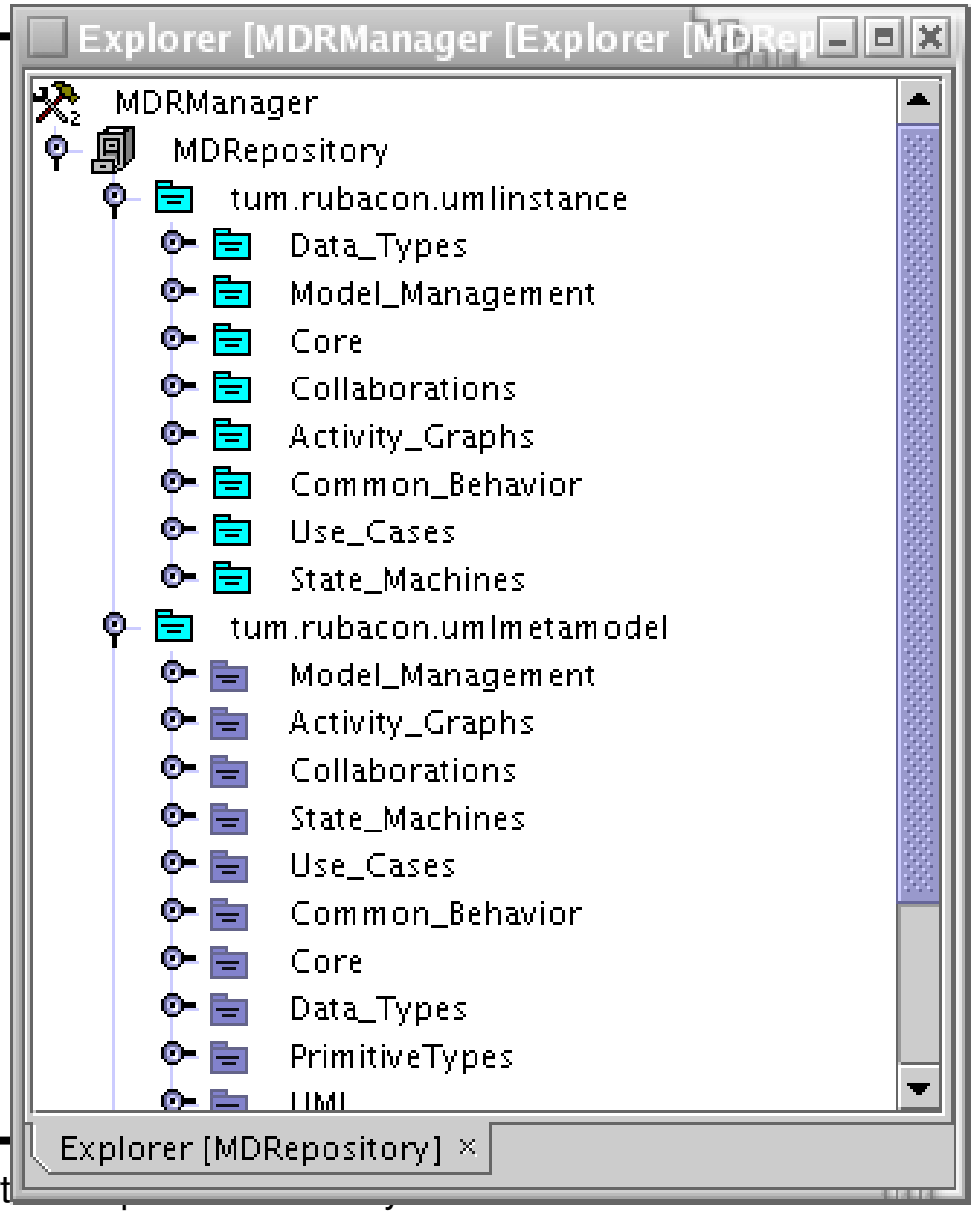
- Requires open Repository and Package
- Requires JMI Interfaces

- Example: Loading UML Class:

```
Iterator it =
  uml.getCore().getUmlClass
  ().refAllOfClass().iterat
  or();
while (it.hasNext()) {
   UmlClass uc =
  (umlClass)it.next();
   // .. do anything with
  UmlClass ..
}
```

Netbeans MDR Explorer

- Part of Netbeans IDE
- Browse Repositories
- Create Instances
- Load XMI Data
- Generate JMI Interfaces
- Shows
  - Extents
  - Metamodels
  - Instances



Roadmap

Prologue

UML drawing tools

Tool-bindings

The CSDUML framework

Example application (crypto checker)

Other approaches, UML 2.0

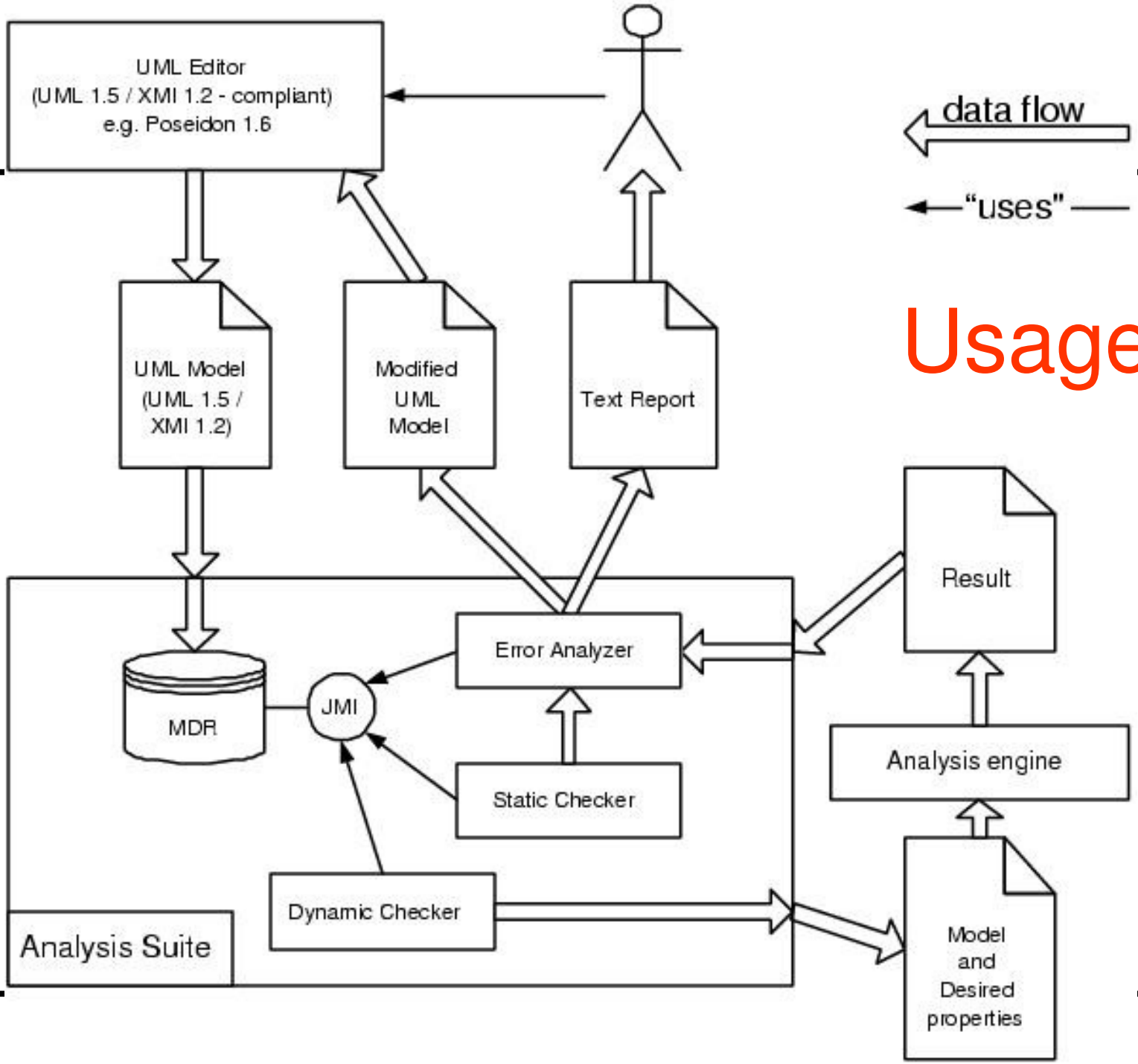
CSDUML Framework: Features

Framework for analysis plug-ins to access UML models on conceptual level over various UI's.

Exposes a set of commands. Has internal state (preserved between command calls).

Framework and analysis tools accessible and available at http://www4.in.tum.de/~umlsec .

Upload UML model (as .xmi file) on website. Analyse model for included critical requirements. Download report and UML model with highlighted weaknesses.


UML Editor
(UML 1.5 / XMI 1.2 - compliant)
e.g. Poseidon 1.6
data flow
"uses"
UML Model
(UML 1.5 /
XMI 1.2)
Modified
UML
Model
Text Report
Result
Error Analyzer
MDR
JMI
Static Checker
Analysis engine
Dynamic Checker
Analysis Suite
Model
and
Desired
properties

Usage

Vision

- Simple independent tools
- Media-independent
- Easy to use
  - Simple developer interface
- Easy to maintain
  - Simple architecture

[joint work with TUM UMLsec group, in part. Pasha Shabalin]

Concept

- Set of plug-in tools
  - Tool exposes predefined interfaces
  - Tool can use framework interfaces
- Tool implements a set of commands
  - Each command has parameters
- Framework = common code
  - UML model management
  - Other services

viki Tool

- Works in GUI and/or Text mode
- Implements interfaces
  - IVikiToolCommandLine
    - Text output only
  - IVikiToolGui
    - Output to JPanel + menu, buttons, etc
- Exposes set of commands
  - Automatically imported by the framework

Framework Interfaces

IMdrContainer

use and control the MDR repository

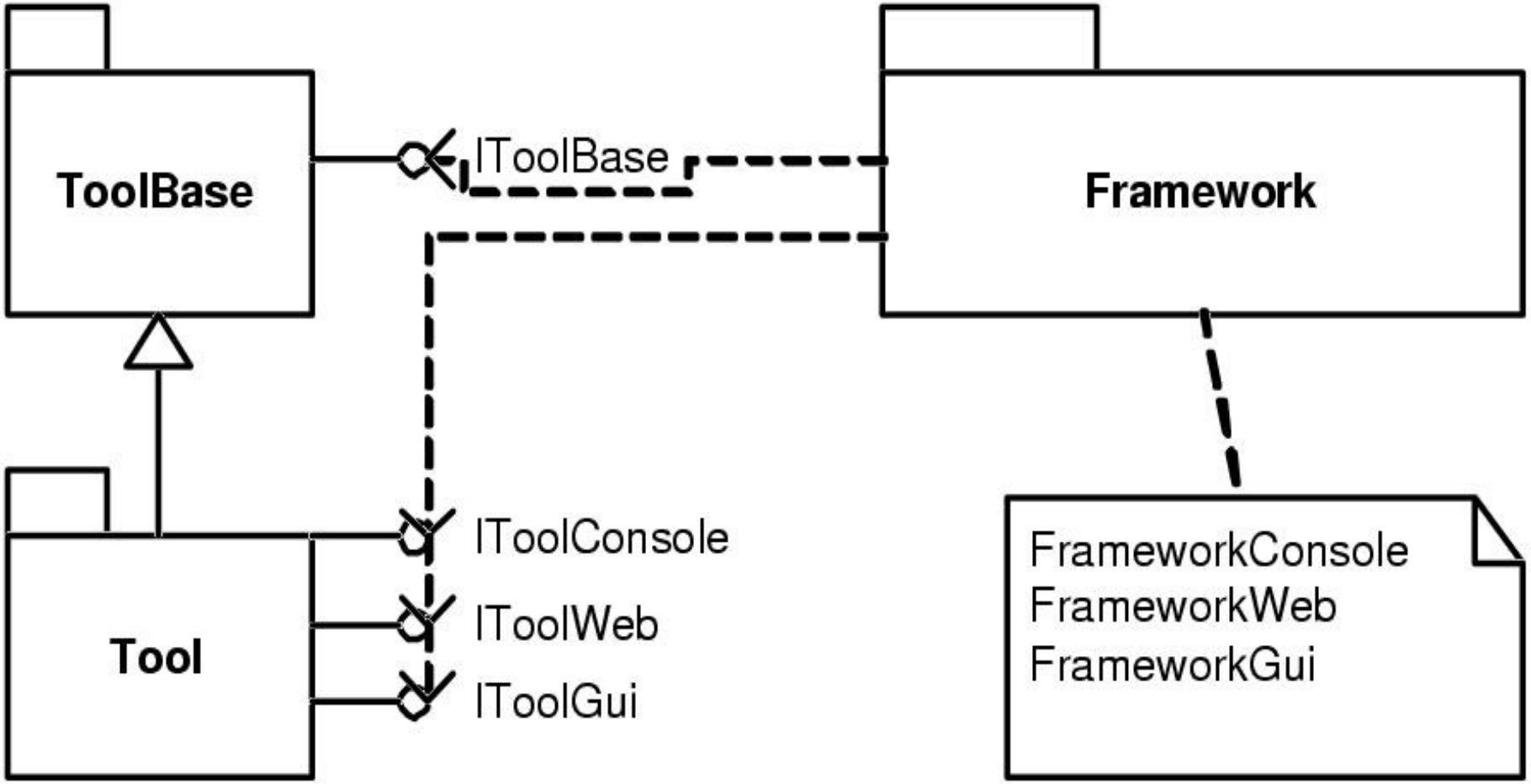
ITextOutput, ILogOutput

render textual information

IAppSettings

store / retrieve tool settings

Tool Interfaces



Tool Interfaces

«interface»
IVikiToolBase

+getConsole() : IVikiToolConsole
+getGui() : IVikiToolGui
+getWeb() : IVikiToolWeb
+initialiseBase(in mdr : IMdrContainer)
+getToolName() : string
+getToolDescription() : string

«interface»
IVikiToolConsole

+getBase() : IVikiToolBase
+initialiseConsole()
+getConsoleCommands() : Iterator
+executeConsoleCommand(in cmd : CommandDescriptor, in p : Iterator, in to : ITextOutput, in lo : ILogOutput)

Tool Interfaces

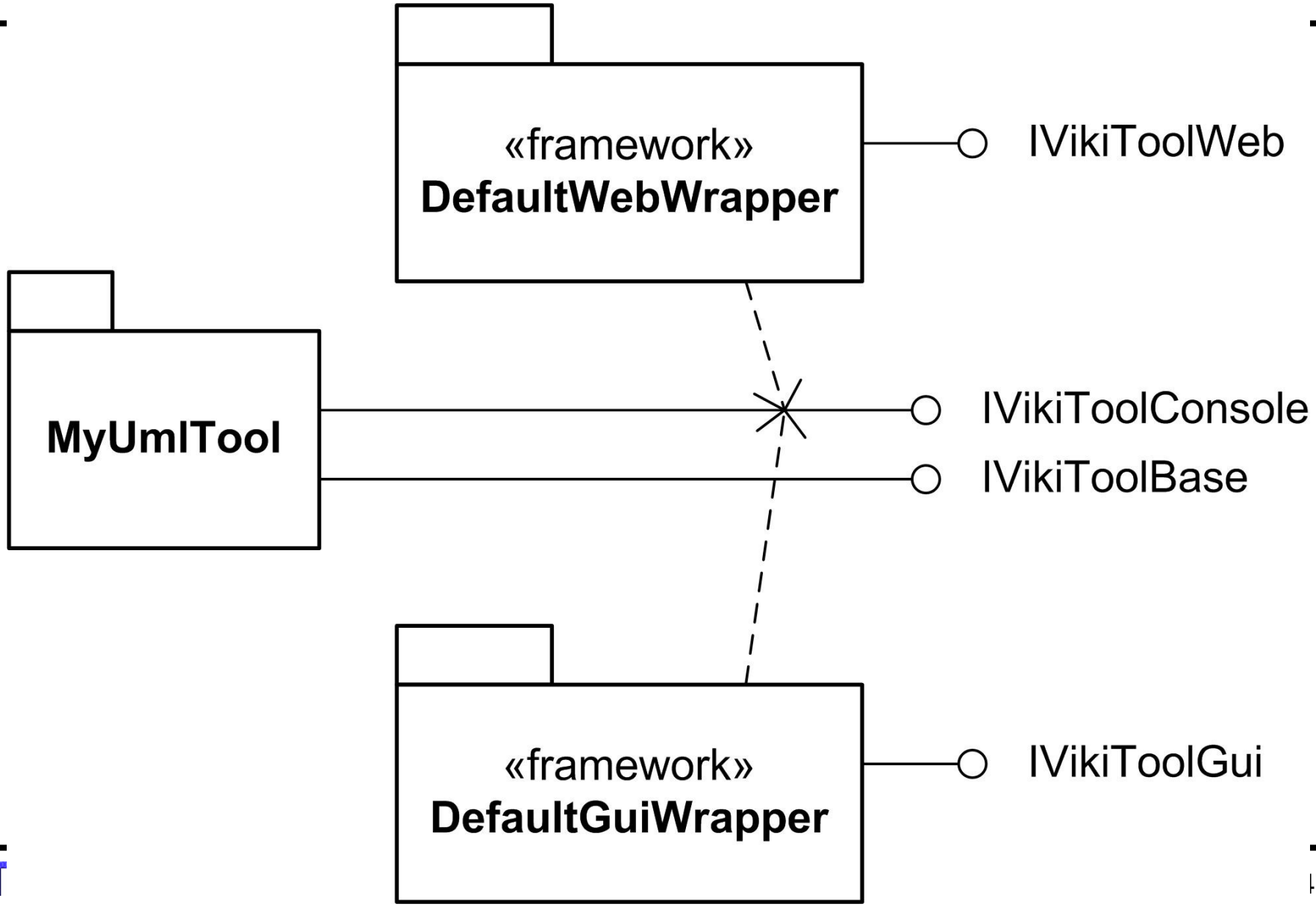
«interface»
IVikiToolGui

+getBase() : IVikiToolBase
+initialiseGui(in log : ILogOutput)
+getUiPanel() : JPanel
+isEnabledGui() : bool
+getGuiCommands() : Iterator
+executeGuiCommand(in cmd : CommandDescriptor, in p : Iterator)

«interface»
IVikiToolWeb

+getBase() : IVikiToolBase
+initialiseWeb()
+getWebCommands() : Iterator
+executeWebCommand(in cmd : CommandDescriptor, in p : Iterator, in to : ITextOutput, in lo : ILogOutput)

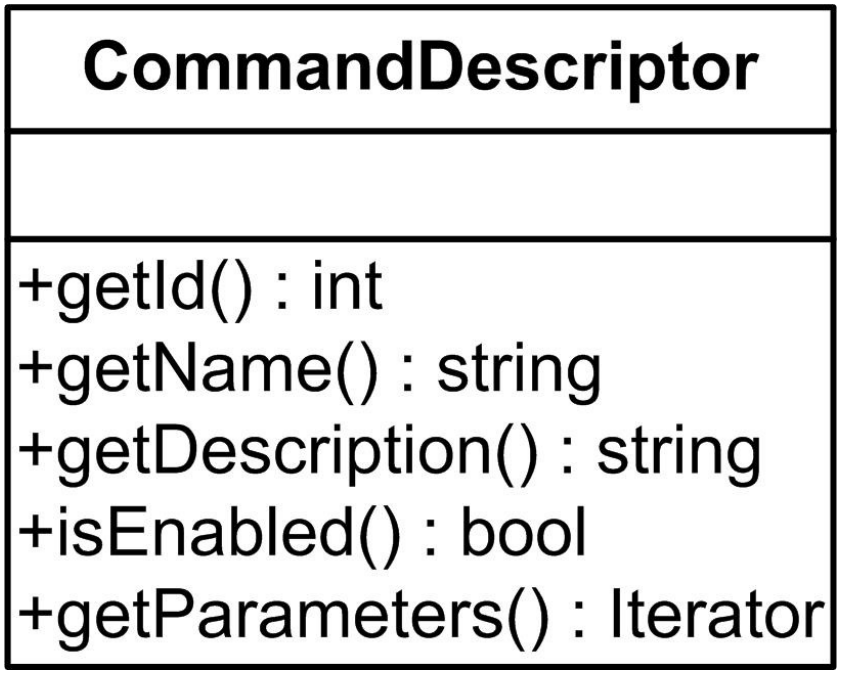
Default Wrappers

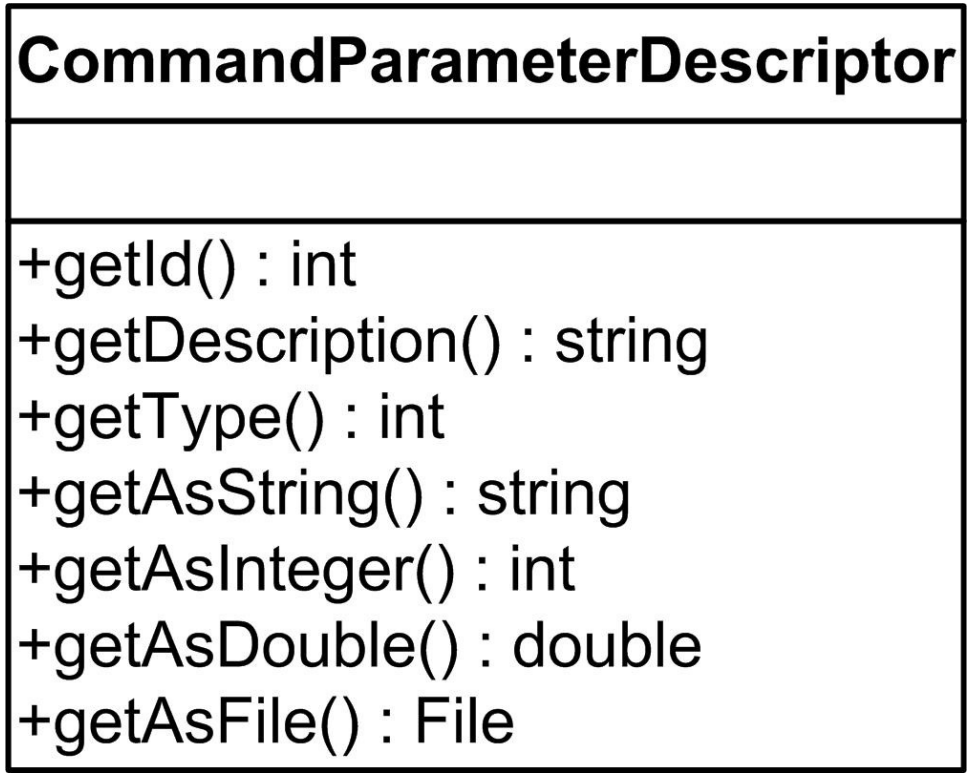


Command Parameters

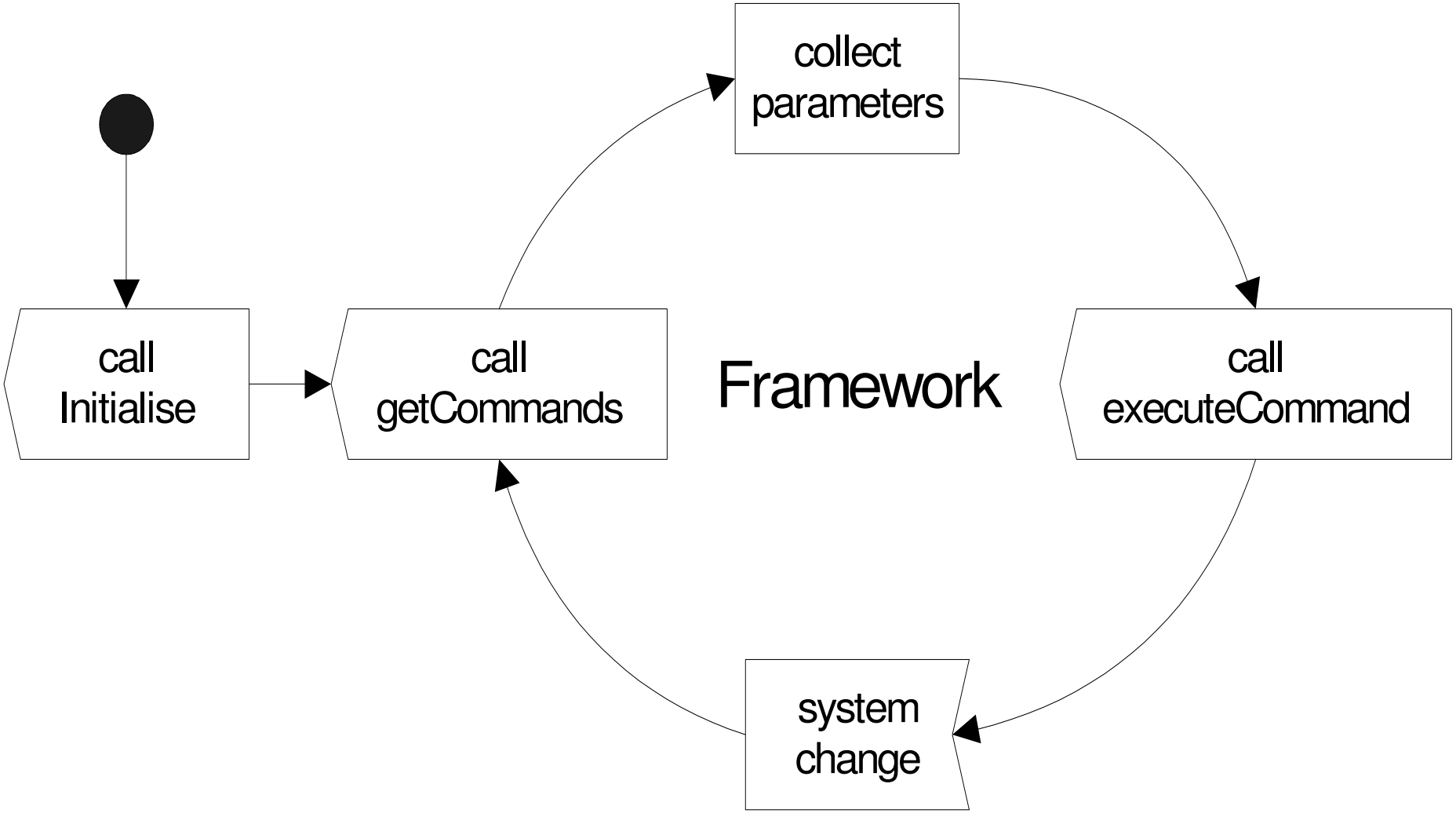
Media-independent functionality

But each mode can have own list of commands

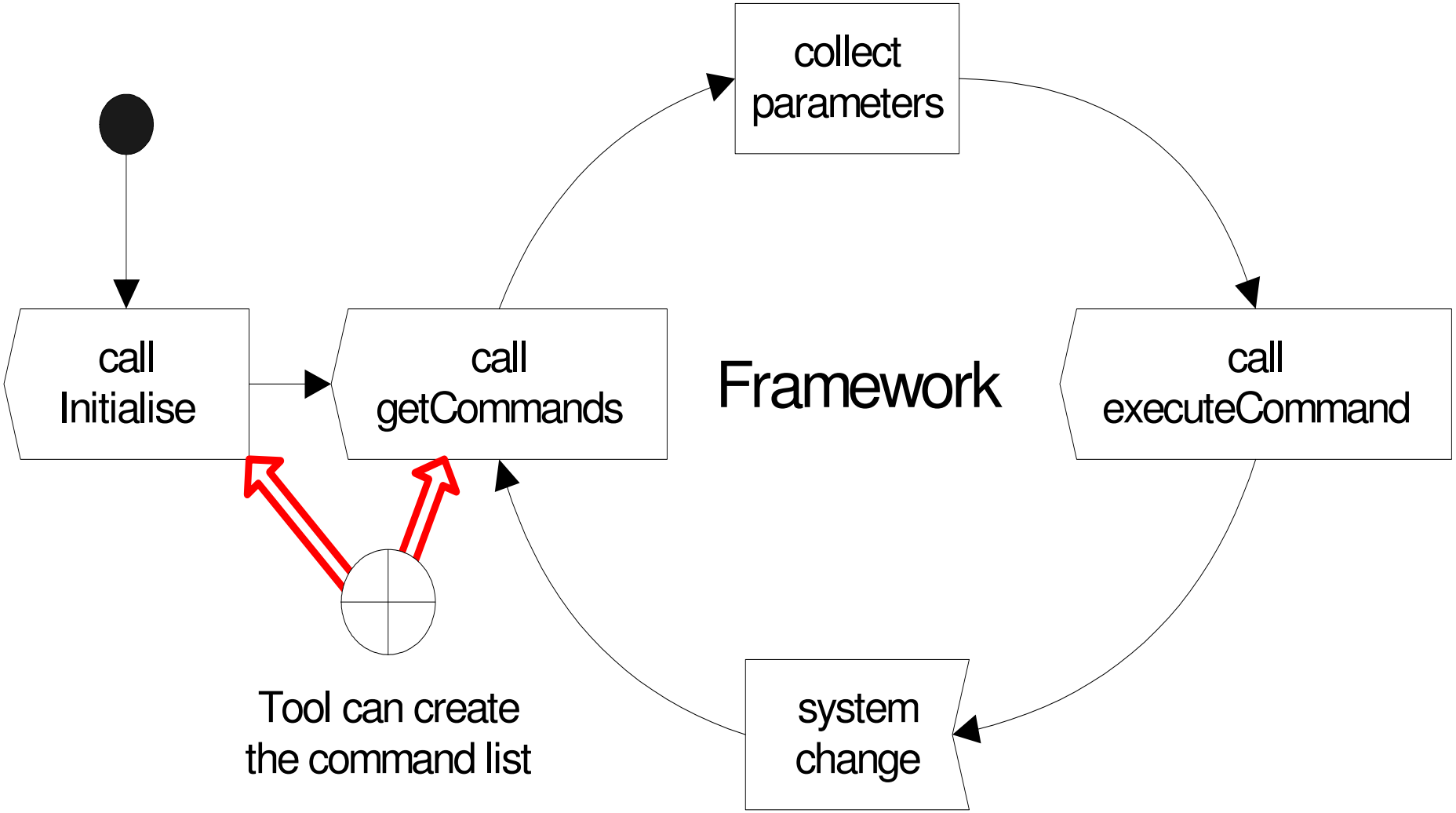




Command Execution Cycle



Exposing Commands



Implementing Tools

- Define the set of commands
  - have parameters
- Tool State preserved between commands
- Commands are not interactive
  - receive parameters
  - execute
  - deliver output

Conventions

Do not use classes from other tools

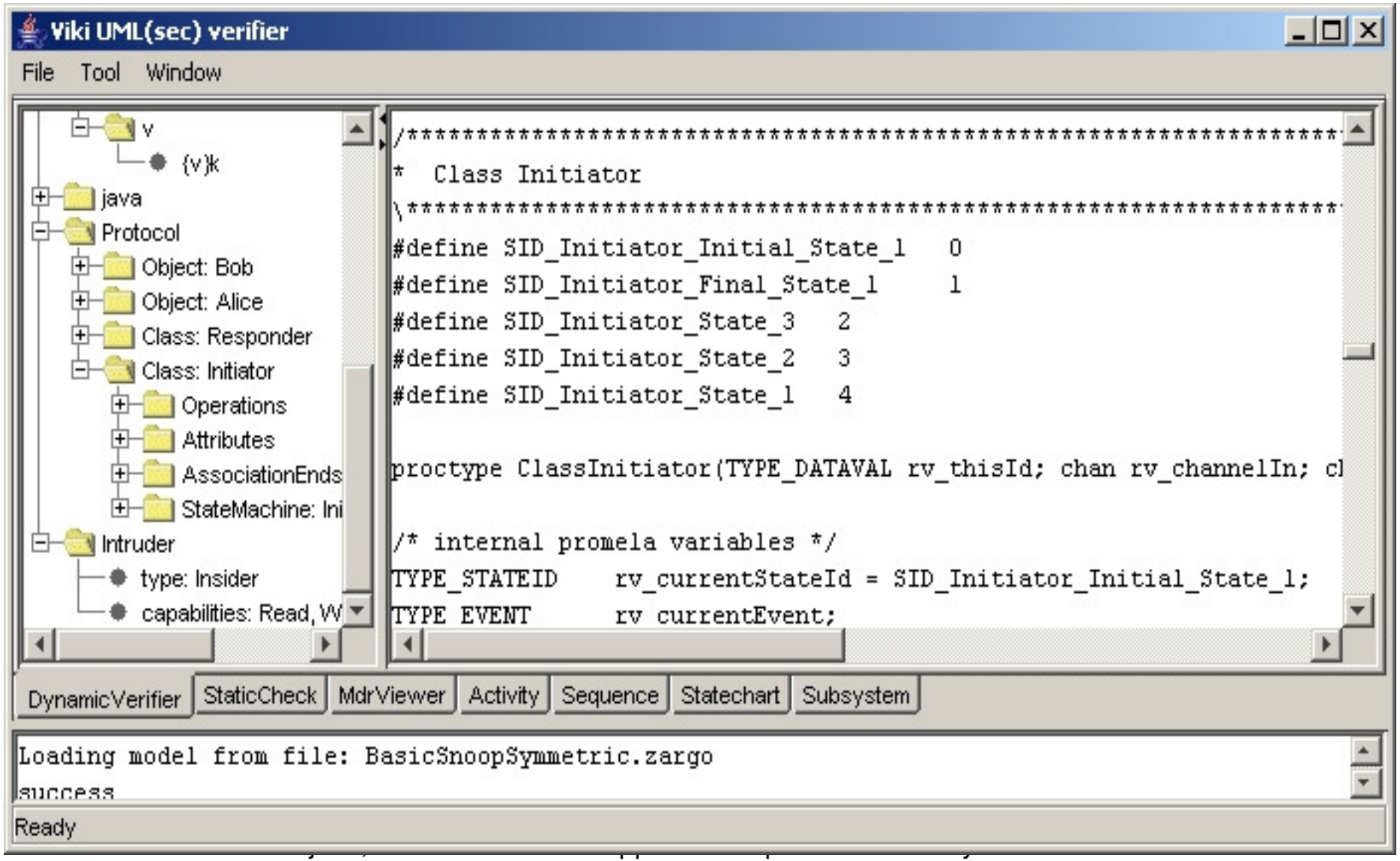
- may be changed by another developer at any time
- make a copy if you need.

Do not use <br> or \n

- You don't know the output media
- Use writeLn .

Always comment your check-ins to viki.framework .

GUI


Viki UML(sec) verifier
File
Tool
Window
v
{v}k
java
Protocol
Object: Bob
Object: Alice
Class: Responder
Class: Initiator
Operations
Attributes
AssociationEnds
StateMachine: Ini
Intruder
type: Insider
capabilities: Read, W
/***
* Class Initiator
****
#define SID_Initiator_Initial_State_1 0
#define SID_Initiator_Final_State_1 1
#define SID_Initiator_State_3 2
#define SID_Initiator_State_2 3
#define SID_Initiator_State_1 4
proctype ClassInitiator(TYPE_DATAVAL rv_thisId; chan rv_channelIn; cl
/* internal promela variables */
TYPE_STATEID rv_currentStateId = SID_Initiator_Initial_State_1;
TYPE_EVENT rv_currentEvent;
DynamicVerifier
StaticCheck
MdrViewer
Activity
Sequence
Statechart
Subsystem
Loading model from file: BasicSnoopSymmetric.zargo
success
Ready

Roadmap

Prologue

UML drawing tools

Tool-bindings

The CSDUML framework

Example application (crypto checker)

Other approaches, UML 2.0

Tool-support: Concepts

Meaning of diagrams stated informally in (OMG 2003).

Ambiguities problem for

- tool support
- establishing behavioral properties (e.g. safety, security)

Need precise semantics for used part of UML, especially to ensure critical requirements.

Formal semantics for UML: How

Diagrams in context (using subsystems).

Model actions and internal activities explicitly.

Message exchange between objects or components (incl. event dispatching).

For UMLsec/dep: include adversary/failure model arising from threat scenario in deployment diagram.

Use Abstract State Machines (pseudo-code).

Execution Semantics

Behavioral interpretation of a UML subsystem:

(1) Takes input events.

(2) Events distributed from input and link queues between subcomponents to intended recipients where they are processed.

(3) Output distributed to link or output queues.

(4) Apply adversary / failure model.

UML Machines

Based on Abstract State Machines (Gurevich: Evolving Algebras, 1991)

- Transition System
- States: multi-sorted first order structure

  set of function names with their interpretations
- Update rule

  modifies function interpretation

UML Machines: communication

Component *S* of *A* sends message

$$msg = op(\exp_1, \ldots, \exp_n) \in \mathbf{Events}$$

to component *R:*

- *S* places *R.msg* into outQu_S.
- Sched_A distributes messages from in-queues to out-queues:
  R.msg removed from outQu_S, *msg* added to inQu_R.
- *R* removes *msg* from in-queue and processes content.

For operation calls, remember sender for return signal.

Sequence diagram

```
Rule Exec(D.O)
if cncts = [] then finished_{D.O} := true
else
    if source(head(cncts)) = O ∧ guard(head(cncts))
        then
            ActionRuleSD(msg(head(cncts)));
            if target(head(cncts)) ≠ O then
                cncts := tail(cncts);
    if target(head(cncts)) = O then
        choose e with e ∈ inQu_O ∧
        msgnm(msg(head(cncts))) = msgnm(e) do
            inQu_O := inQu_O \ {{e}};
            args_{D,lnum} := Args(e);
            lnum := lnum + 1;
            if msgnm(e) ∈ Op then
                sender(msgnm(e)) :=
                    sndr(e).sender(msgnm(e));
            cncts := tail(cncts)
```

UML Machine Systems

Supports modularity concept (models subsystems).

Groups several UML Machines to form a new complex UML Machine.

Executable specification for the UML Machine System is derived.

UML Machine System

$Rule\ \langle \mathcal{A} \rangle$

seq

forall S **with** $S \in \mathsf{Comp}_{\mathcal{A}}$ **do**

$\mathsf{inQu}_{\langle S \rangle} := \mathsf{inQu}_{\langle S \rangle} \uplus$

$\{\!\{ \mathbf{tail}(e) : e \in (\mathsf{inQu}_{\langle \mathcal{A} \rangle} \setminus \mathsf{Msgs}_{\mathcal{A}}) \uplus$

$\biguplus_{l \in links_S} \mathsf{linkQu}_{\langle \mathcal{A} \rangle}(l) \wedge \mathbf{head}(e) = S \}\!\}$

$\mathsf{inQu}_{\langle \mathcal{A} \rangle} := \emptyset$

$\langle \mathsf{Sched}_{\mathcal{A}} \rangle$

forall l **with** $l \in \mathsf{Links}_{\mathcal{A}}$ **do**

$\mathsf{linkQu}_{\langle \mathcal{A} \rangle}(l) := \{\!\{ e \in \mathsf{outQu}_{\langle S \rangle} :$

$S \in \mathsf{Comp}_{\mathcal{A}} \wedge l = \{\mathbf{head}(e), A_i\} \}\!\}$

$\mathsf{outQu}_{\langle \mathcal{A} \rangle} := \mathsf{outQu}_{\langle \mathcal{A} \rangle} \uplus \biguplus_{S \in \mathsf{Comp}_{\mathcal{A}}} \{\!\{ \mathbf{tail}(e) :$

$e \in \mathsf{outQu}_S \wedge \mathbf{head}(e) = \langle \mathcal{A} \rangle \}\!\}$

forall S **with** $S \in \mathsf{Comp}_{\mathcal{A}}$ **do**

$\mathsf{outQu}_{\langle S \rangle} := \emptyset$

endseq

Example Application: Security

Following Dolev, Yao (1982): To analyze system, verify against attacker model from threat scenarios in deployment diagrams who

- may participate in some protocol runs,
- knows some data in advance,
- may intercept messages on some links,
- injects messages that it can produce in some links
- may access certain nodes.

Adversaries

Model classes of adversaries.

May attack different parts of the system according to threat scenarios.

Example: insider attacker may intercept communication links in LAN.

To evaluate security of specification, simulate jointly with adversary model.

Cryptography

Keys are symbols, crypto-algorithms are abstract operations.

- Can only decrypt with right keys.
- Can only compose with available messages.
- Cannot perform statistical attacks.

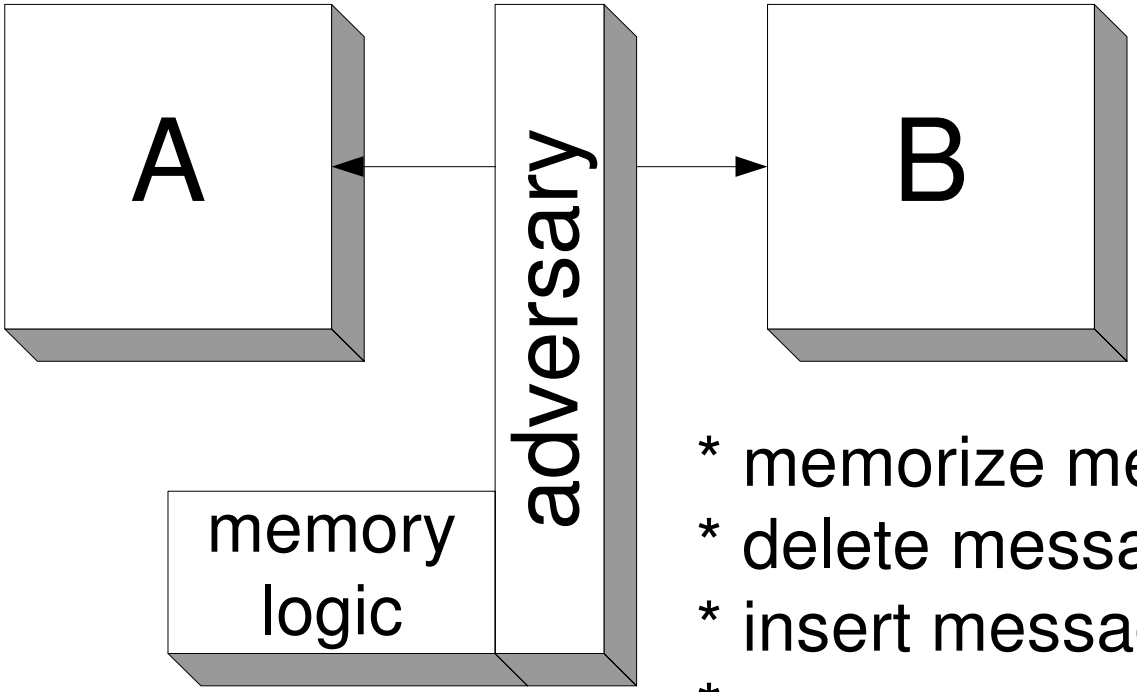
Expressions

Exp: term algebra generated by Var ∪ Keys ∪ Data and

- _ :: _ (concatenation) and empty expression $\mathcal{E}$,
- { _ } _ (encryption)
- *Dec ( )* (decryption)
- *Sign ( )* (signing)
- *Ext_( )* (extracting from signature)
- *Hash(_)* (hashing)

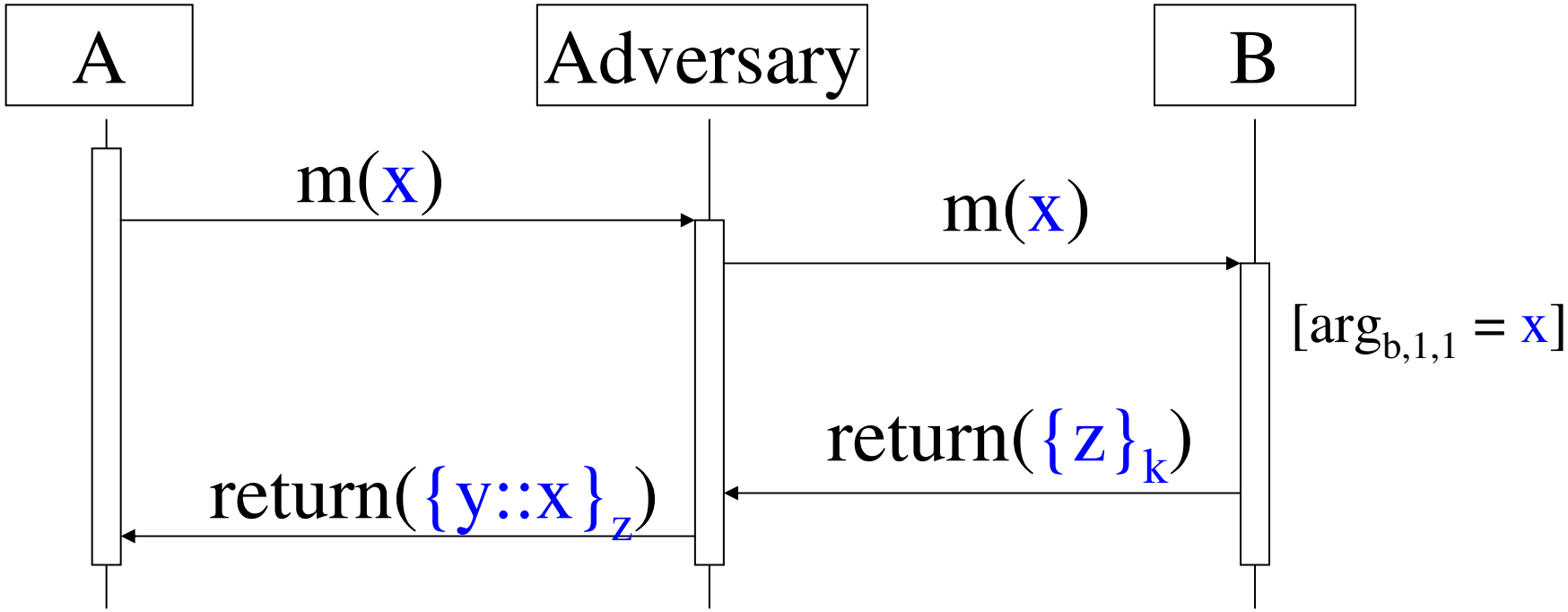
by factoring out the equations $Dec_{K^{-1}}(\{E\}_k) = E$ and $Ext_K(Sign_{K^{-1}}(E)) = E$ (for $K \in$ Keys).

Adversary Model



* memorize message
* delete message
* insert message
* compose own message
* use cryptographic primitives

Adversary: Simulation



Adversary knowledge:

k^{-1}, y, x

$\{z\}_k$, z

- $\forall e, k. \mathcal{D}ec_{k^{-1}}(\{e\}_k) = e$

Security Analysis in First-order Logic

Idea: approximate set of possible data values flowing through system from above.

Predicate *knows(E)* meaning that the adversary may get to know *E* during the execution of the protocol.

For any secret *s*, check whether can derive *knows(s)* using automated theorem prover.

First-order Logic: Basic Rules

For initial adversary knowledge (K^0): Define *knows(E)* for any E initially known to the adversary (protocol-specific, e.g. K_A, K_A^{-1}).

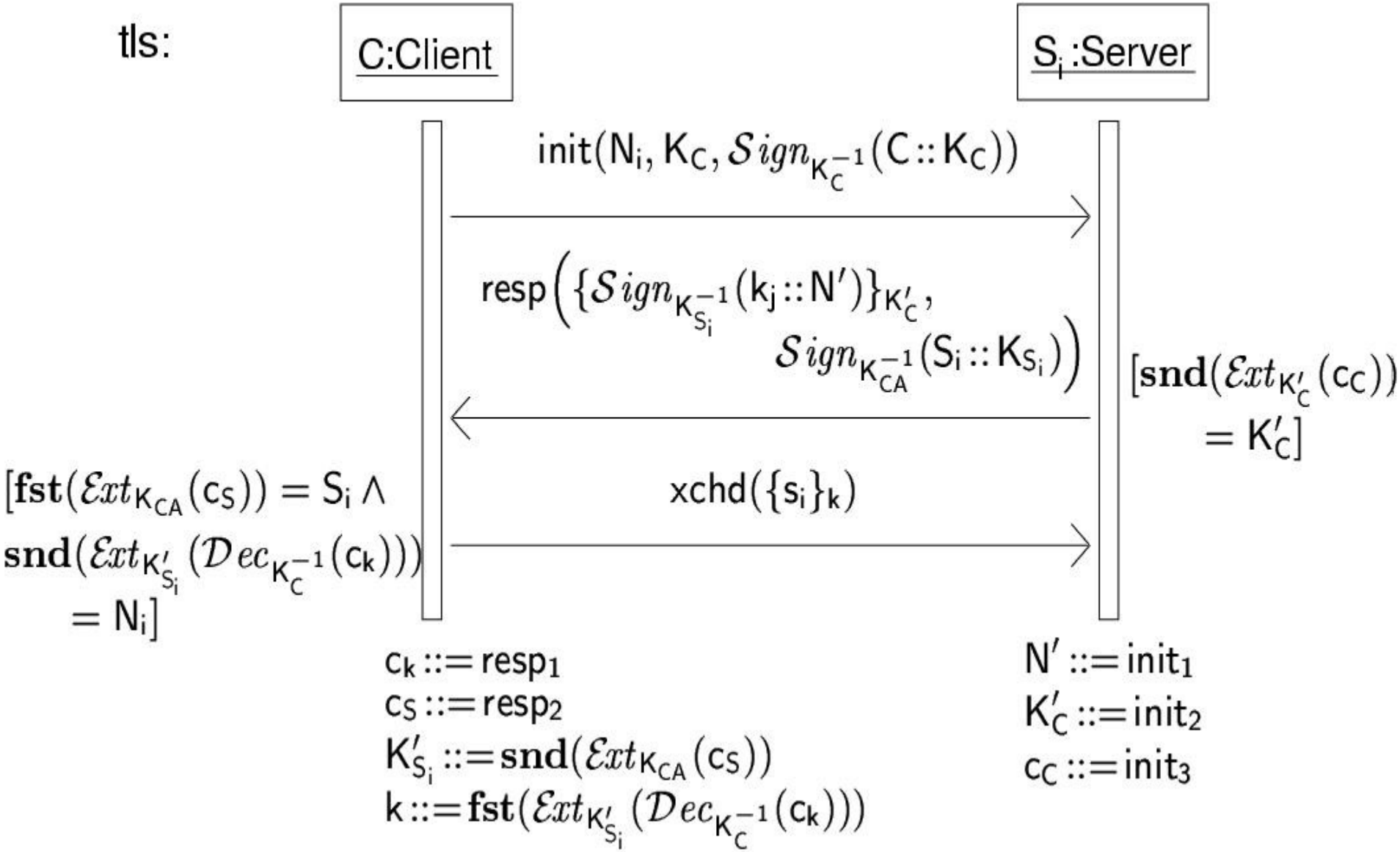
Define above equations.

For evolving knowledge (K^n) define

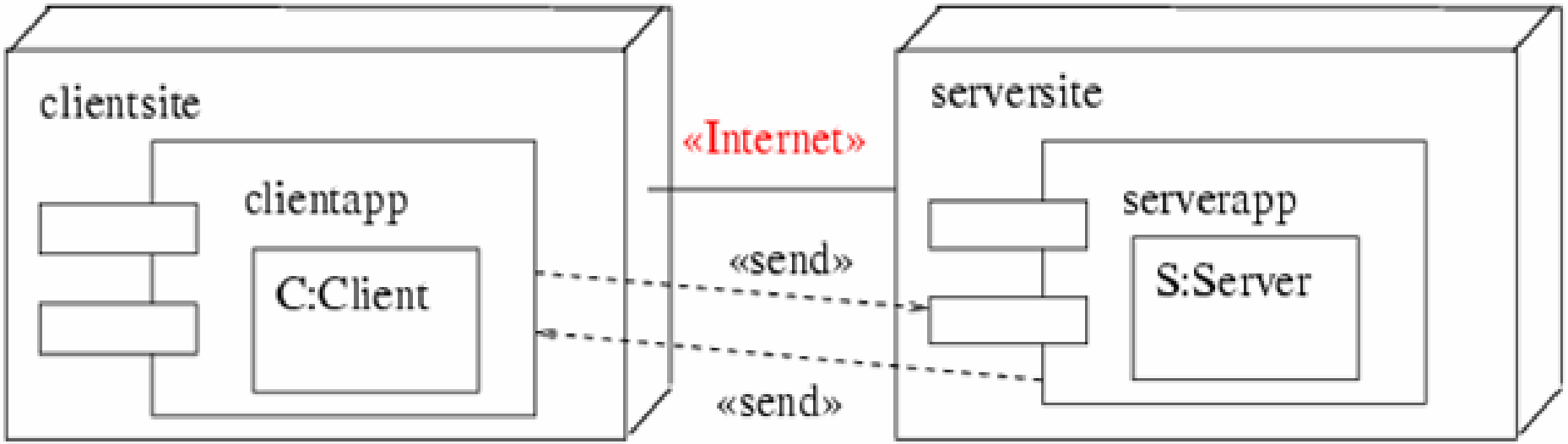
$$\forall E_1, E_2.(knows(E_1) \wedge knows(E_2) \Rightarrow knows(E_1::E_2) \wedge knows(\{E_1\}_{E2}) \wedge knows(Dec_{E2}(E_1)) \wedge knows(Sign_{E2}(E_1)) \wedge knows(Ext_{E2}(E_1)))$$

$$\forall E.(knows(E) \Rightarrow knows(head(E)) \wedge knows(tail(E)))$$

Given Sequence Diagram …



… and Physical Layer Model …



Deployment diagram.

Derived adversary model: read, delete, insert data.

… Translate to 1st Order Logic

Connection (or statechart transition)

$TR1=(in(msg\_in),cond(msg\_in),out(msg\_out))$

followed by $TR2$ gives predicate $PRED(TR1)=$

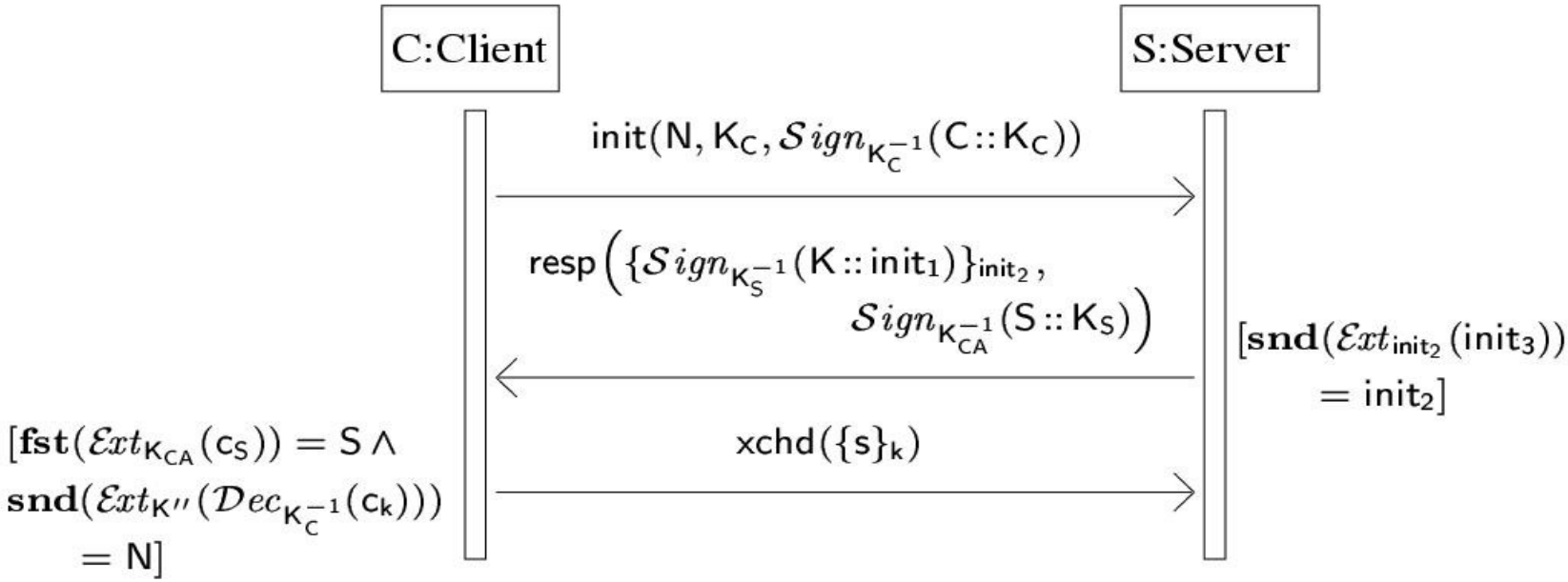
$$\forall msg\_in.\ [knows(msg\_in) \wedge cond(msg\_in) \Rightarrow knows(msg\_out) \wedge PRED(TR2)]$$

(Assume: order enforced (!).)

Can include senders, receivers in messages.

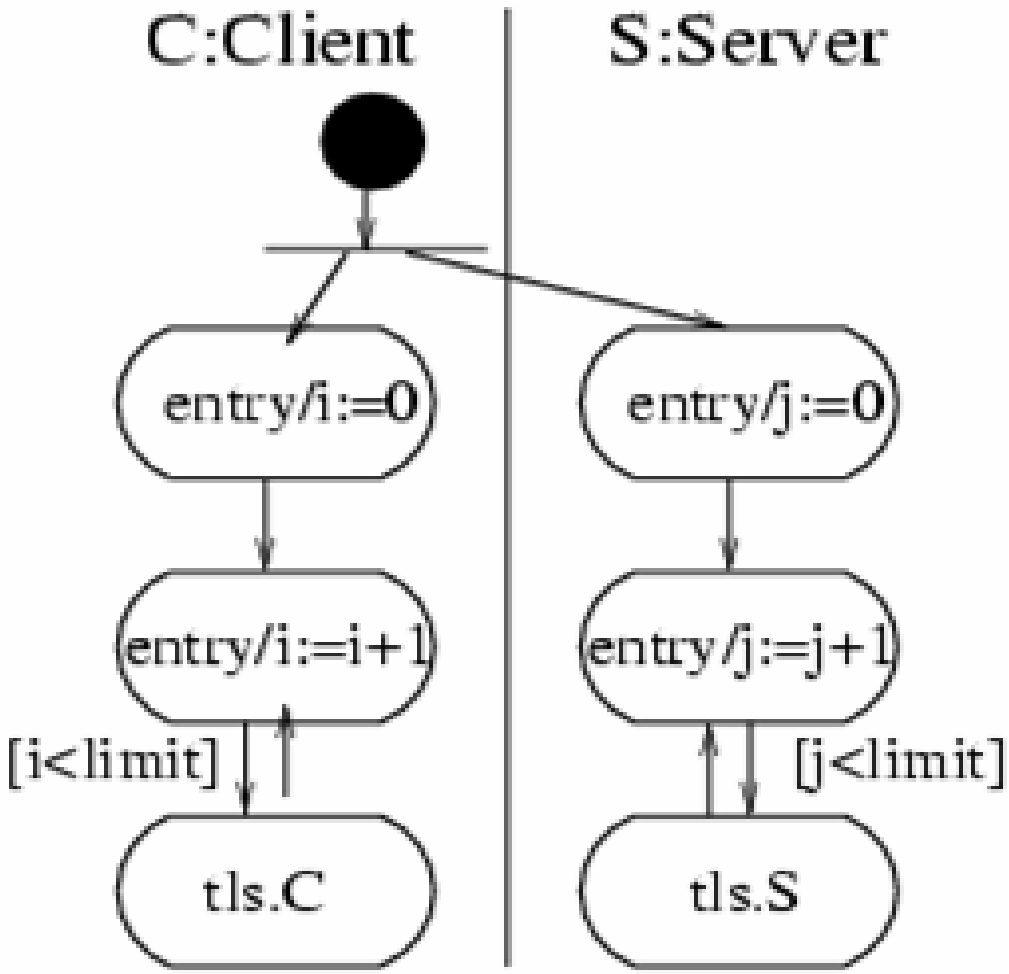
Abstraction: find all attacks, may have false positives.

Example: Translation to Logic



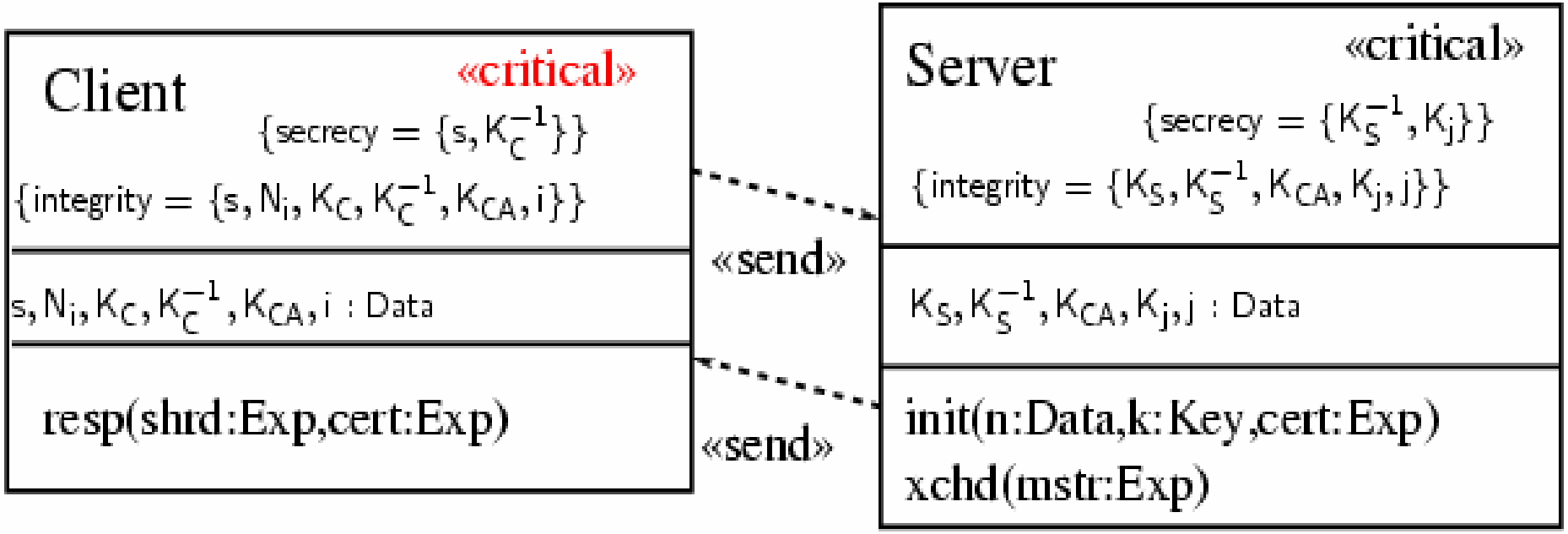
$$knows(N) \wedge knows(K_C) \wedge knows(Sign_{KC\text{-}1}(C::K_C))$$
$$\wedge\ \forall init_1, init_2, init_3.[knows(init_1) \wedge knows(init_2) \wedge$$
$$knows(init_3) \wedge snd(Ext_{init2}(init_3)) = init_2$$
$$\Rightarrow knows(\{Sign_{KS\text{-}1}(\ldots)\}_{\ldots}) \wedge [\ldots] \wedge [\ldots \Rightarrow \ldots]\ldots]$$

Execute in System Context



Activity diagram

Formulate Data Security Requirements



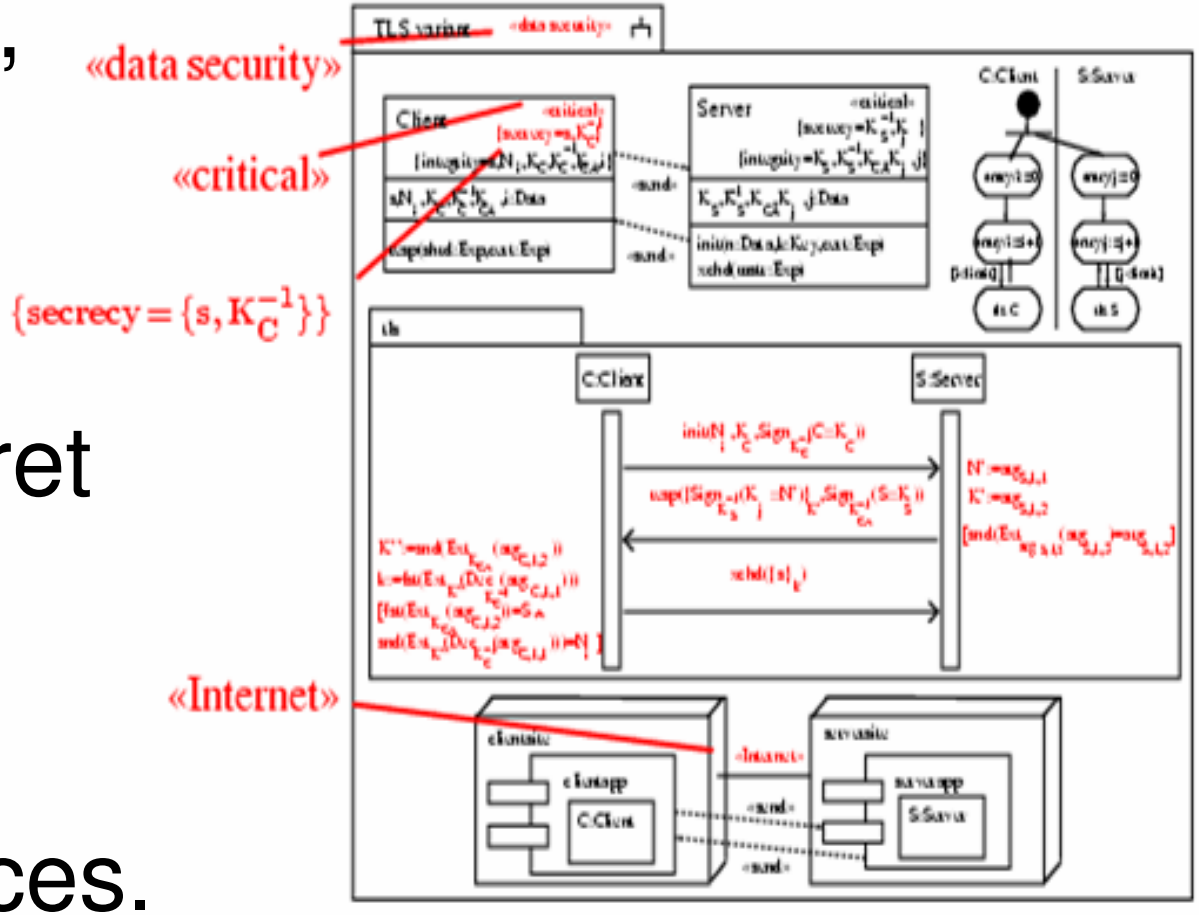
Class diagram.

Gives conjecture: *knows(s)* derivable ?

Example: Proposed Variant of TLS (SSL)

Apostolopoulos, Peris, Saha; IEEE Infocom 1999.

Goal: send secret protected by session key using fewer server resources.



TLS Variant in TPTP Notation I

```
input_formula(tls_abstract_protocol,axiom,(
 ![ArgS_11, ArgS_12, ArgS_13, ArgC_11, ArgC_12] : (
   ![DataC_KK, DataC_k, DataC_n] : (
      % Client -> Attacker (1. message)
      (       knows(n)
           & knows(k_c)
           & knows(sign(conc(c, k_c), inv(k_c) ) )  )
    & % Server -> Attacker (2. message)
      (  (    knows(ArgS_11)
           & knows(ArgS_12)
           & knows(ArgS_13)
           & ( ? [X] : equal( sign(conc(X, ArgS_12), inv(ArgS_12) ),
                              ArgS_13 ) )  )
      => (    knows(enc(sign(conc(kgen(ArgS_12), ArgS_11), inv(k_s) ),
                        ArgS_12 ) )
           & knows(sign(conc(s, k_s), inv(k_ca) ) ) )  )
```

TLS Variant in TPTP Notation II

```
  & % Client -> Attacker (3. message)
    (  (   knows(ArgC_11)
         & knows(ArgC_12)
         & equal(sign(conc(s, DataC_KK), inv(k_ca)), ArgC_12 )
         & equal(enc(sign(conc(DataC_k, DataC_n), inv(DataC_KK) ),
                     k_c), ArgC_11 )
         & ( ? [DataC_ks] : equal(sign(conc(s, DataC_ks), inv(k_ca) ),
                                  ArgC_12 ) )
         & equal(enc(sign(conc(DataC_k, n), inv(DataC_KK) ), k_c),
                 ArgC_11 )
         & equal(enc(sign(conc(DataC_k, DataC_n), inv(DataC_KK) ), k_c),
                 ArgC_11 )
       )
    => ( knows(symenc(secret, DataC_k)) )  )
) )  )).
```

Surprise ...

```
E-SETHEO csp03 single processor running on host ...
(c) 2003 Max-Planck-Institut fuer Informatik and
    Technische Universitaet Muenchen

tlsvariant-freshkey-check.tptp
...
time limit information: 300 total (entering statistics module).
problem analysis ...
testing if first-order ...
first-order problem
...
statistics: 19 0 7 46 3 6 2 0 1 2 14 8 0 2 28 6
...
schedule selection: problem is horn with equality (class he).
schedule:605 3 300 597
...
entering next strategy 605 with resource 3 seconds.
...
analyzing results ...
proof found
time limit information: 298 total / 297 strategy (leaving wrapper).
...
e-SETHEO done. exiting
```

Attack

… Which Means:

Can derive *knows(s)* (!).

That is: Protocol does not preserve secrecy of *s* against adversaries.

➔ Completely insecure wrt stated goals.

But why ?

Could look at proof tree.

Or: use prolog-based attack generator.

Man-in-the-Middle Attack

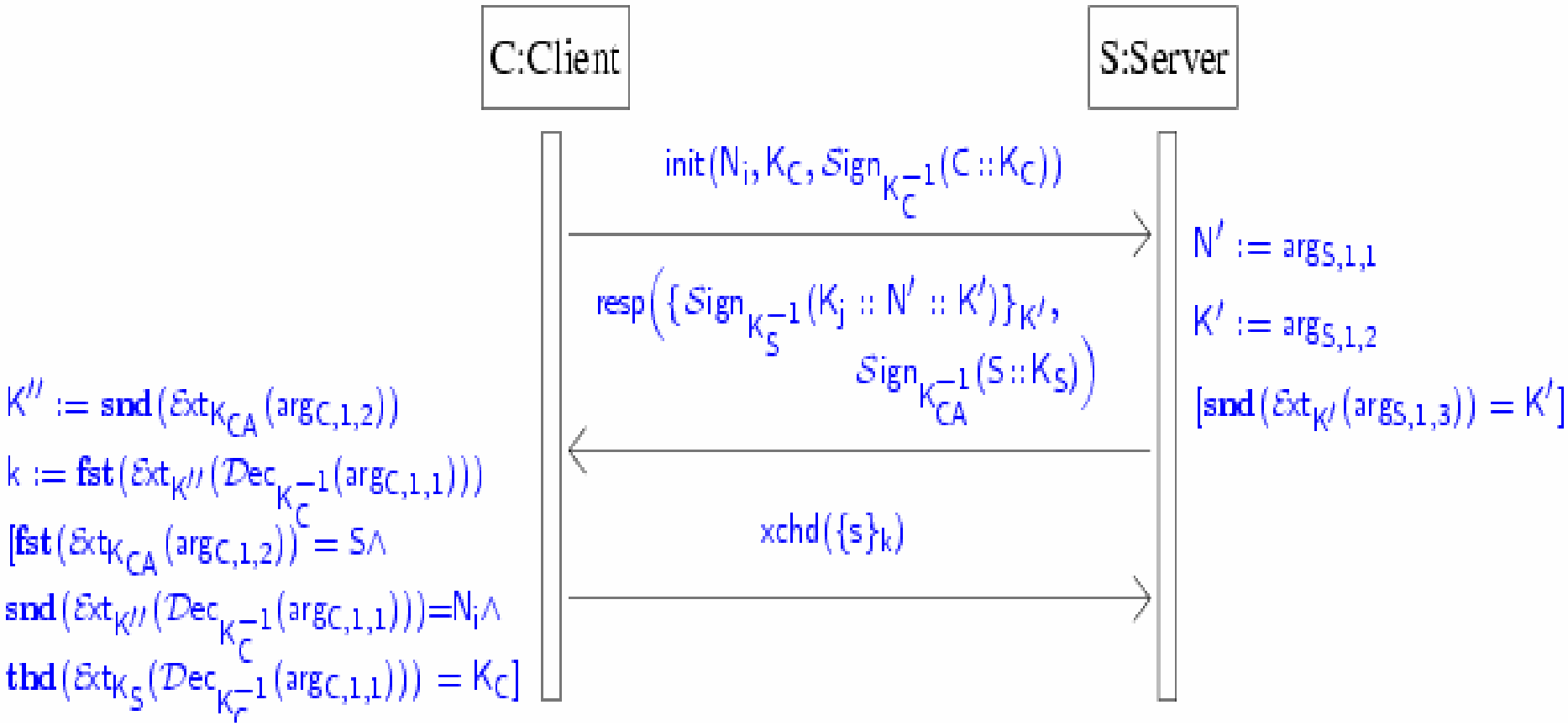
$$C \xrightarrow{N_i::K_C::Sign_{K_C^{-1}}(C::K_C)} A \xrightarrow{N_i::K_A::Sign_{K_A^{-1}}(C::K_A)} S$$

$$A \xleftarrow{\{Sign_{K_S^{-1}}(K_j::N_i)\}_{K_A}::Sign_{K_{CA}^{-1}}(S::K_S)} S$$

$$C \xleftarrow{\{Sign_{K_S^{-1}}(K_j::N_i)\}_{K_C}::Sign_{K_{CA}^{-1}}(S::K_S)} A$$

$$C \xrightarrow{\{s\}_{K_j}} A \xrightarrow{\{s\}_{K_j}} S$$

The Fix



e-Setheo: *knows(s)* not derivable. Thus secure.

Roadmap

Prologue

UML drawing tools

Tool-bindings

The CSDUML framework

Example application (crypto checker)

Other approaches, UML 2.0

Look Around: AutoFocus

Industrial CASE tool with
UML-like notation: AUTOFOCUS
(http://autofocus.informatik.tu-muenchen.de)

- Simulation
- Validation (Consistency, Testing, Model Checking)
- Code Generation (e.g. Java, C, Ada)
- Connection to Matlab

Some History

[Slotosch 03]

1996: AutoFOCUS editor

1997: Simcenter: (Java-) Simulation

1998-1999 Quest

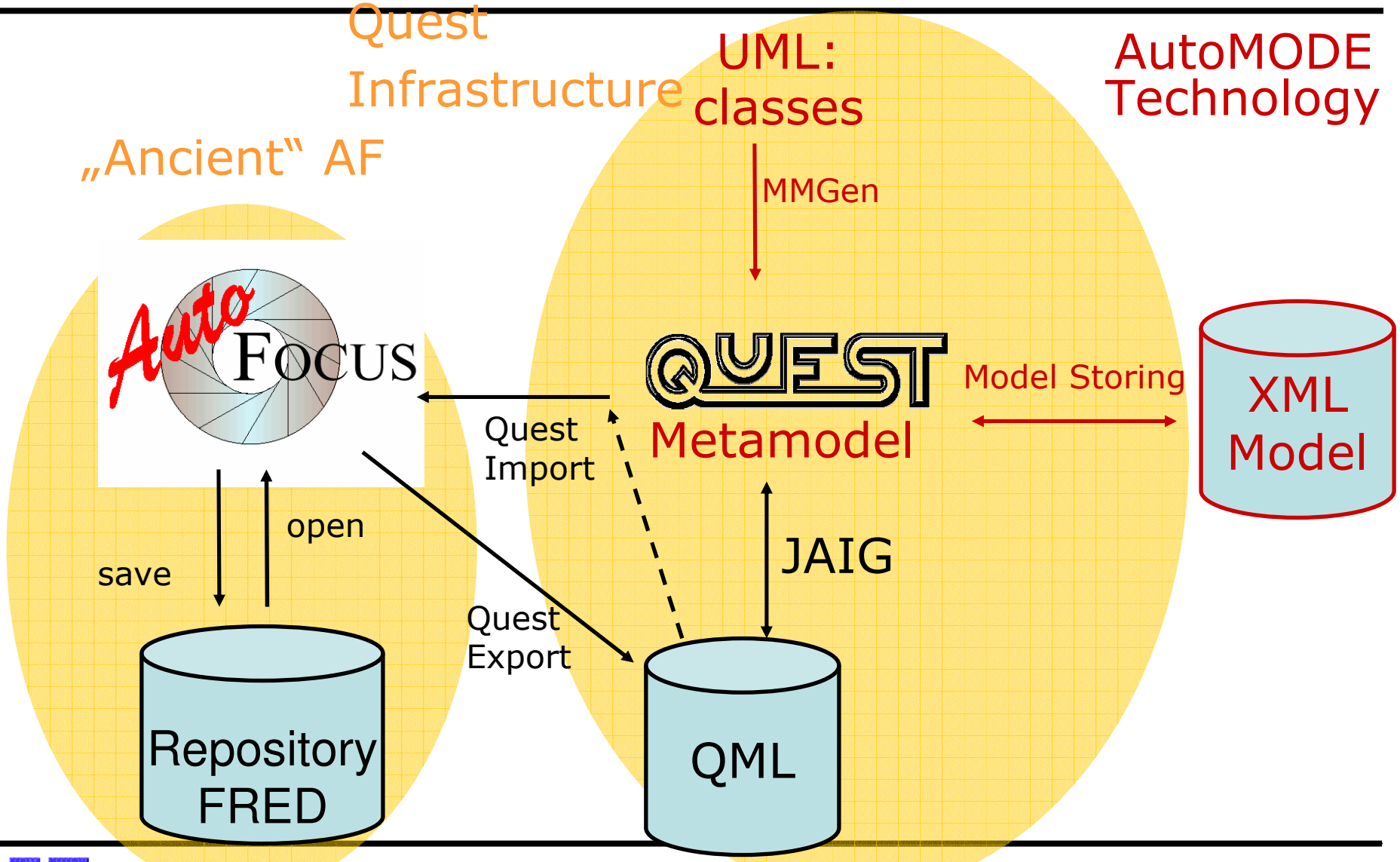
- infra-structure (data structure: Quest-Model-Files „.qml“)
- binding to model checker, theorem prover

From 2000: Validas company (23 releases)

- many features: (♦-Ports, Replay, Restart, Check,..)
- Simulation, Prototype code-generation (Java, Ada, C)
- Model API (via QML-Export)
- Testgeneratoren (TIG, Prolog, C, Ada)
- Integration / bindings ATTOL, Matlab, ASCET..)
- XML-Export/Import via API

AutoFOCUS Architecture

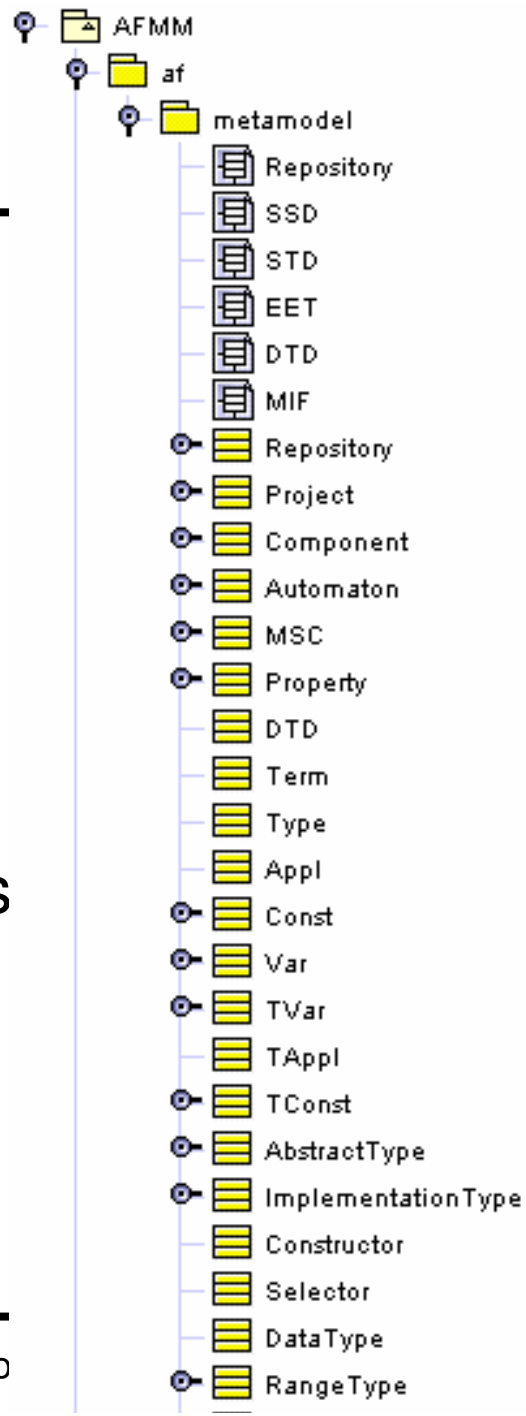
[Slotosch 03]



AutoFocus Meta Model

57 classes in 6 diagrams

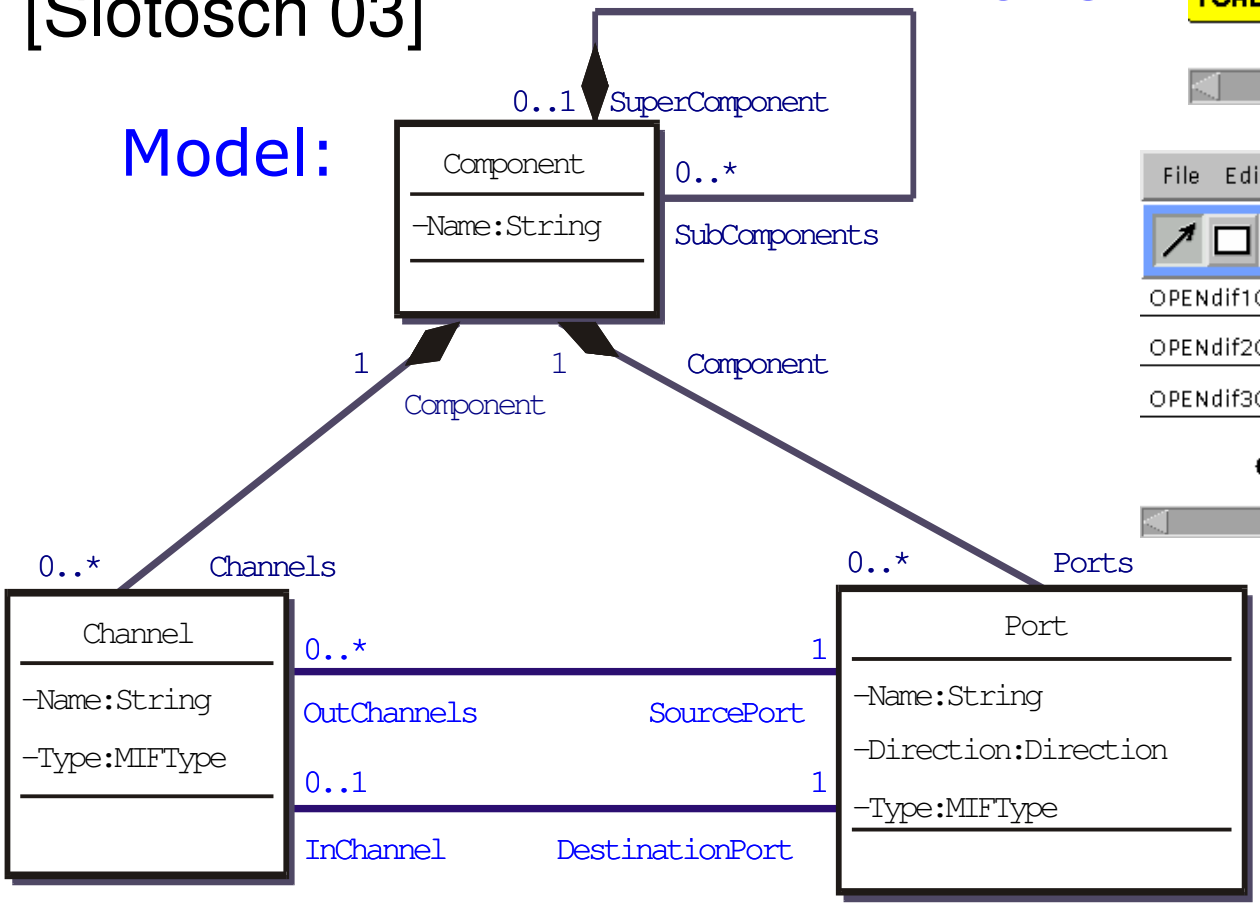
- Repository:
  - main structure: Project, Component
  - gathers: SSD,EET,STD,DTD
- SSD (and implementation types)
- STD (hierarchical automata)
- EET (complex MSCs)
- DTD (terms, types, type classes, definitions
- MIF: (ModelInterFace)
  - MIF-Term, MIFType: Modell-> Term
  - ModelConst: Term -> Model
  - Diverses: Comment, Direction



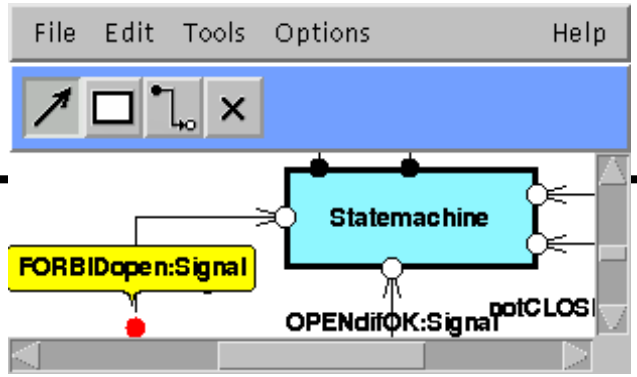
Example Metamodel

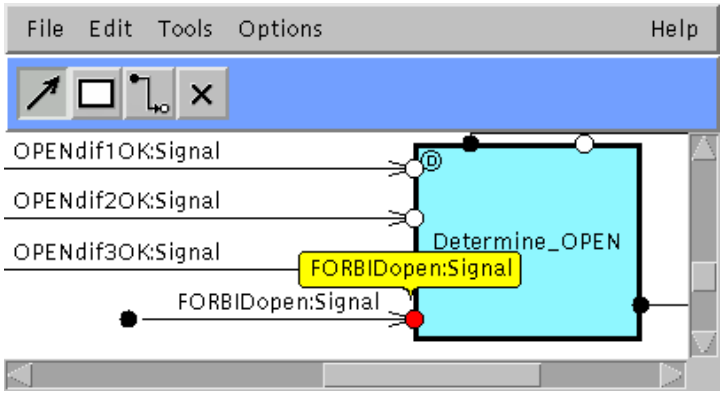
[Slotosch 03]

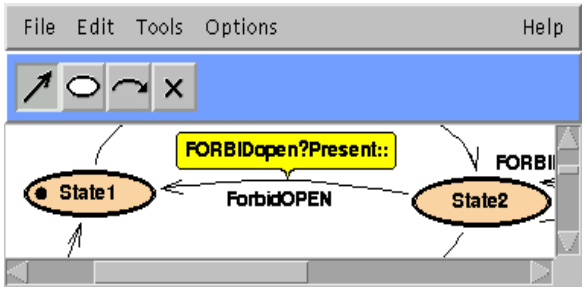
Model:



Views:

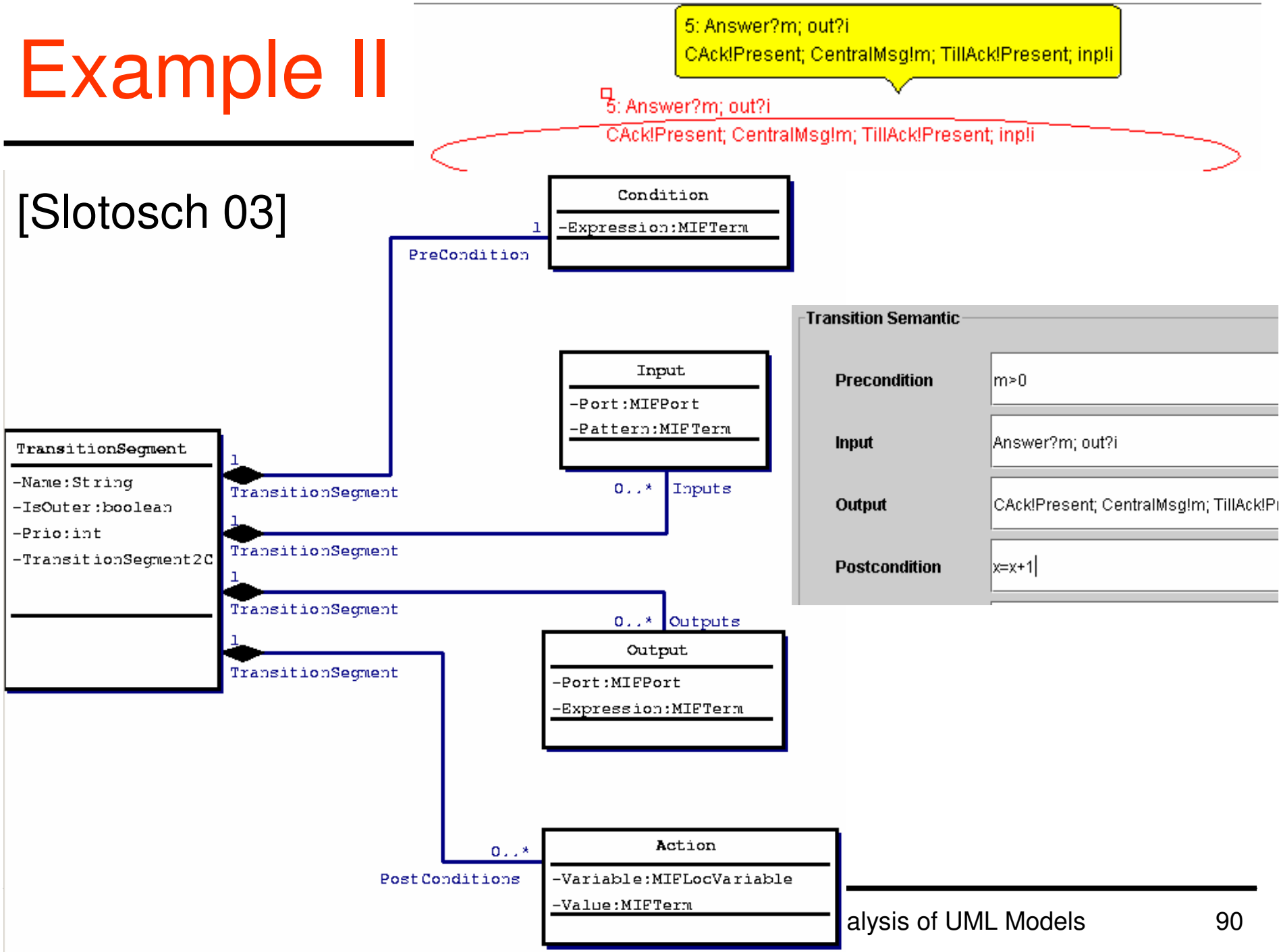






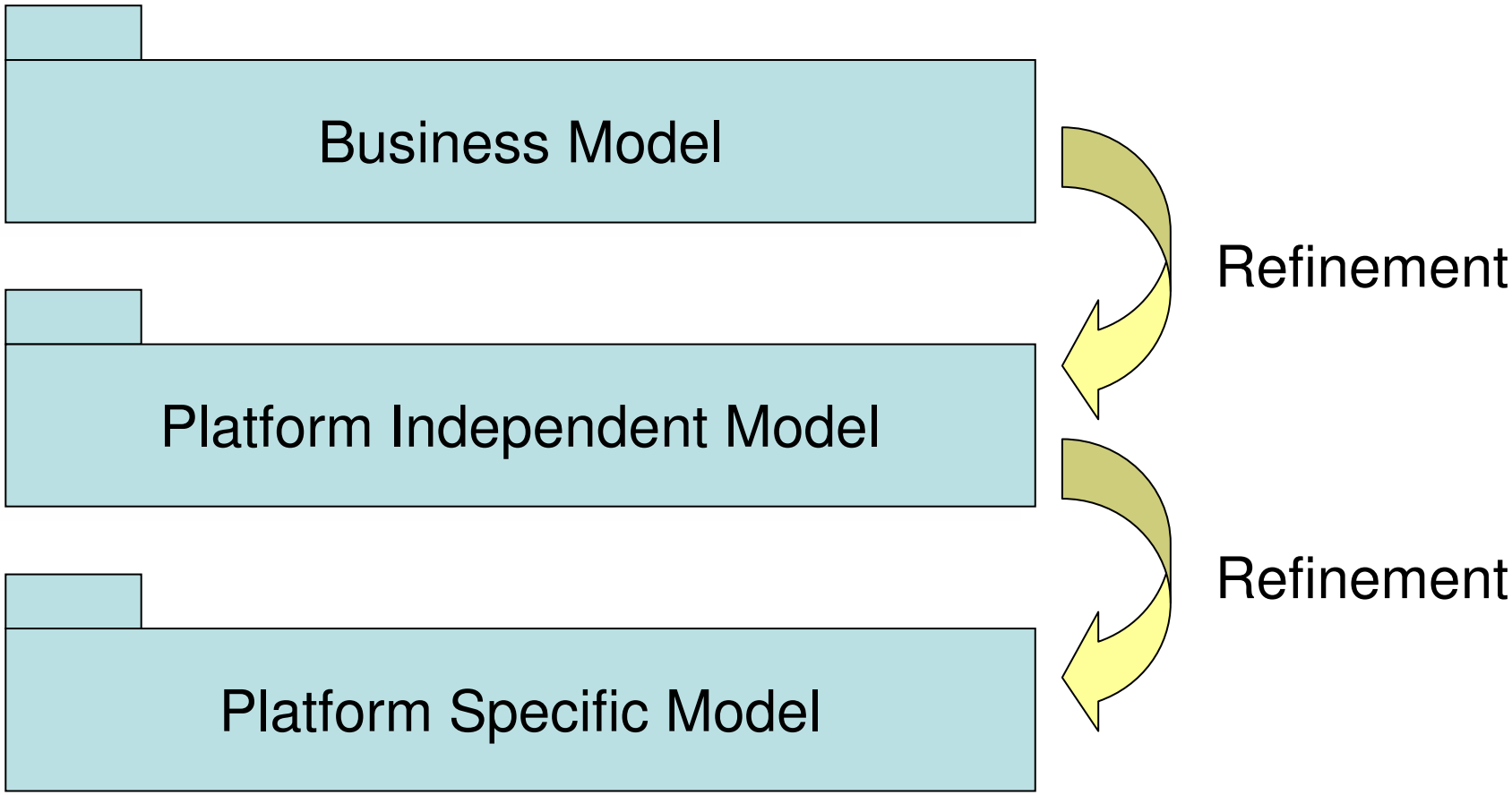
Example II

[Slotosch 03]



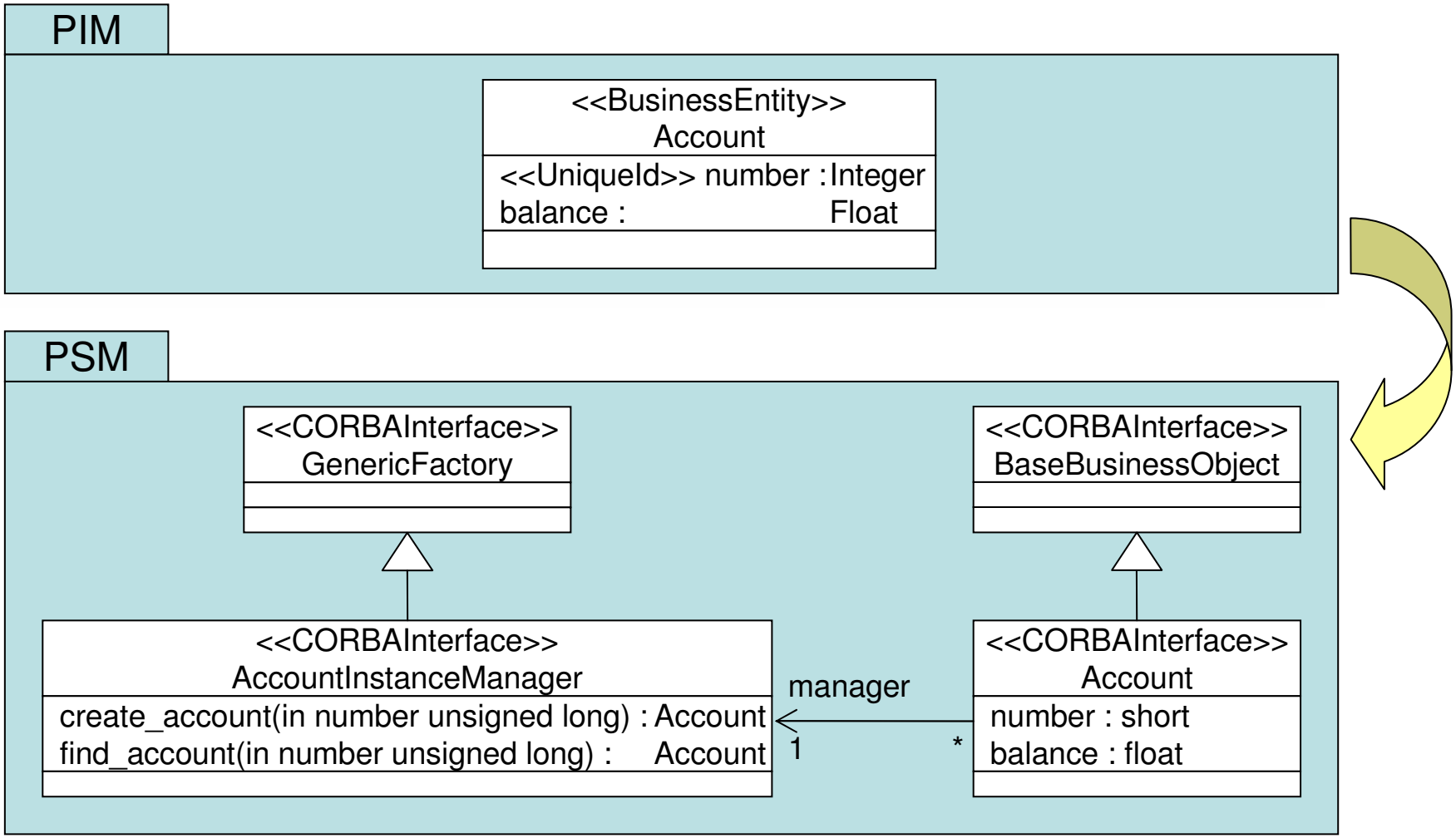
Look around: MDA Transformations

[Braun, Marschall 03]



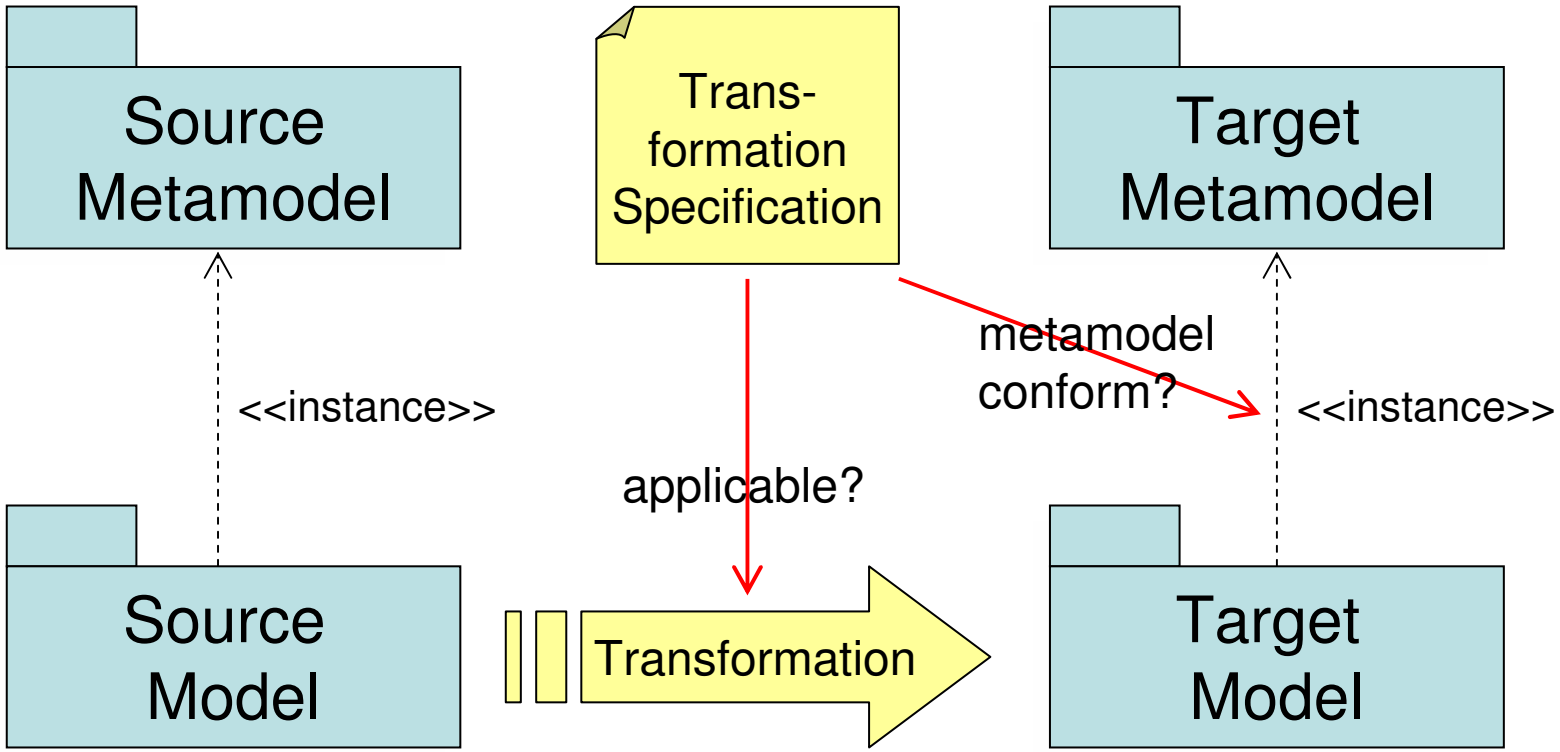
Sample Transformation

[Braun, Marschall 03]



A Language for Model Transformations

[Braun, Marschall 03]



BOTL

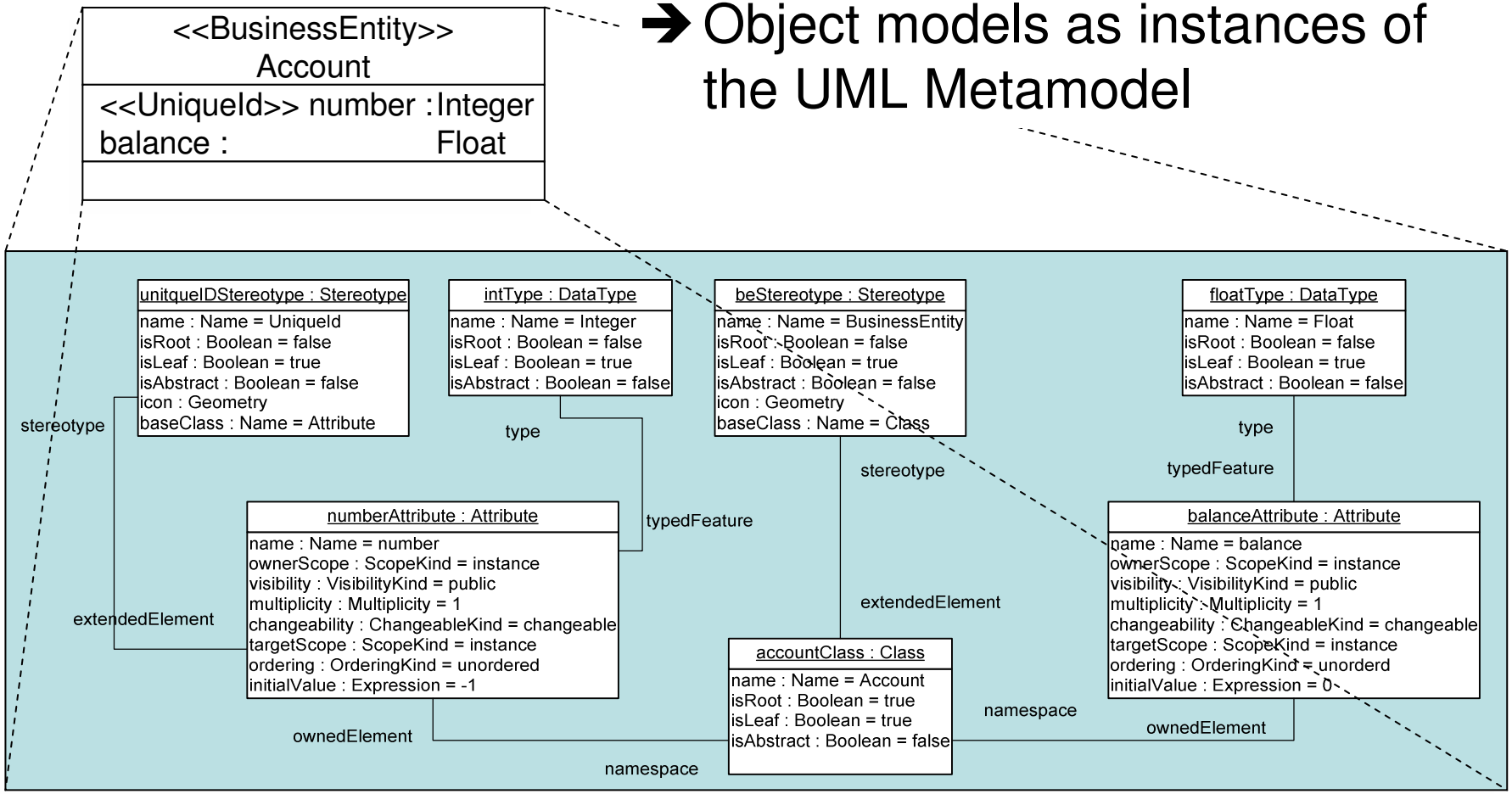
[Braun, Marschall 03]

- Formally founded model of
  - metamodels
  - models
  - model transformations
- Rule-based: Rules translate clippings of a source model into clippings of a target model and merge them
- Ordering of rules and pattern matching strategy is irrelevant
- Allows an automated verification of the
  - applicability and
  - metamodel conformance

  of rule sets based on the rules, the source, and target metamodels

What will be transformed?

[Braun, Marschall 03]



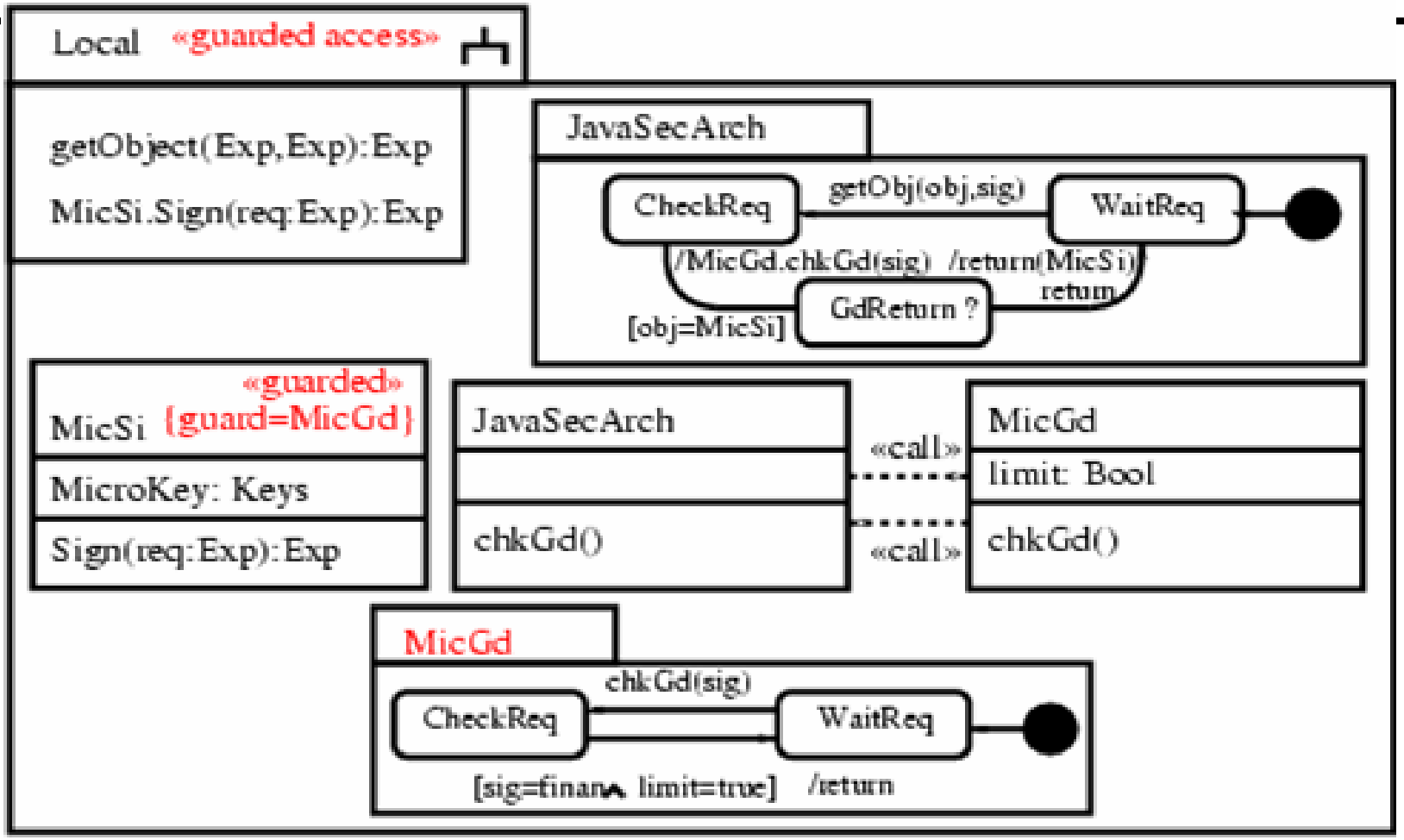
➔ Object models as instances of the UML Metamodel

Outlook: UML 2.0

Changes for advanced UML tool developers:

- some new diagrams
- changes in execution semantics (activity diagrams)
- changes in diagram interchange: UML 1.x metamodels for diagram interchange do not support layout information. UML 2.0 diagram interchange supposed to solve this (in finalization, see http://www.gentleware.com)

Exercise: Secure Use of Java Security



Enforces overall security policy ?

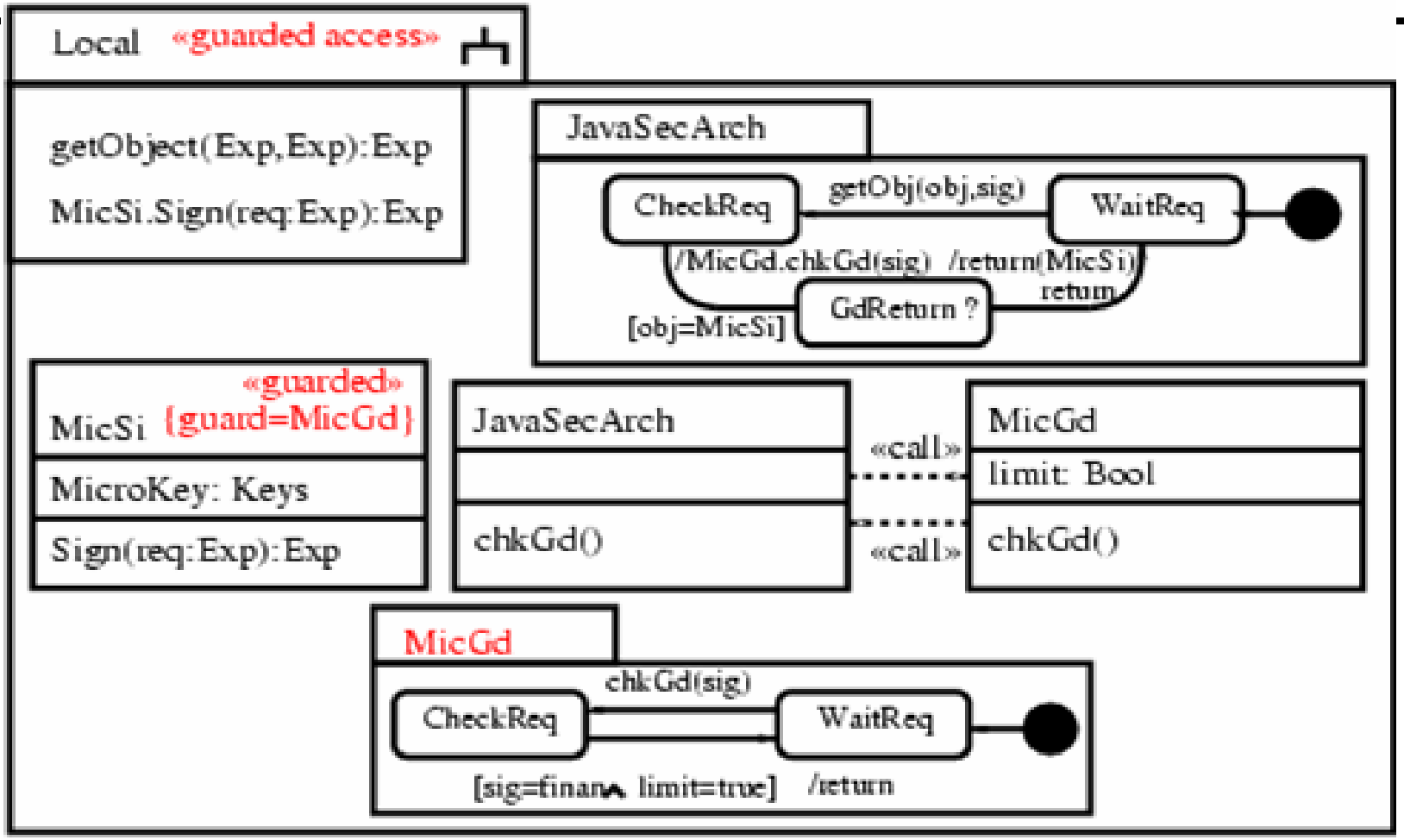
≪guarded access≫

Ensures that in Java, ≪guarded≫ classes only accessed through {guard} classes.

Constraints:

- References of ≪guarded≫ objects remain secret.
- Each ≪guarded≫ class has {guard} class enforcing security policy.

Example ≪guarded access≫



≪guarded access≫ fulfilled.

Demo

Viki UML(sec) verifier

File Tool Window

- v
  - {v}k
- java
- Protocol
  - Object: Bob
  - Object: Alice
  - Class: Responder
  - Class: Initiator
    - Operations
    - Attributes
    - AssociationEnds
    - StateMachine: Ini
- Intruder
  - type: Insider
  - capabilities: Read, W

```
/*****************************************************************
*  Class Initiator
\*****************************************************************
#define SID_Initiator_Initial_State_1    0
#define SID_Initiator_Final_State_1      1
#define SID_Initiator_State_3    2
#define SID_Initiator_State_2    3
#define SID_Initiator_State_1    4

proctype ClassInitiator(TYPE_DATAVAL rv_thisId; chan rv_channelIn; cl

/* internal promela variables */
TYPE_STATEID     rv_currentStateId = SID_Initiator_Initial_State_1;
TYPE_EVENT       rv_currentEvent;
```

DynamicVerifier | StaticCheck | MdrViewer | Activity | Sequence | Statechart | Subsystem

```
Loading model from file: BasicSnoopSymmetric.zargo
success
```

Ready

Exercise

Consider example implementation (handout).

Modify so that it checks that all classes carry signatures required by guards (using {signed} tag).

(Just sketch changes.)

Discussion

Role of advanced tool-support for model-based development with UML ?

Conclusions

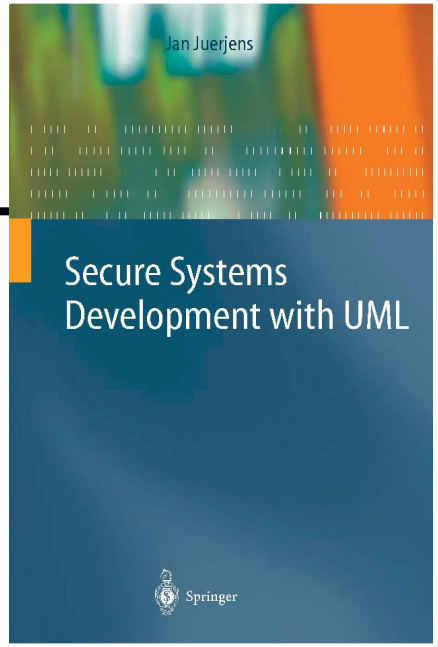
Tool-supported Model-based Software Engineering using UML:

- formally based approach to (critical) software engineering
- automated tool support
- integrated approach (source-code, configuration data)
- increase quality with bounded costs, time-to-market.

Literature

- Fondement: MDR introduction (slides), 2004
  http://lgl.epfl.ch/members/fondement
- Slotosch et al.: AutoFocus, 1996-2004
  http://www.validas.de
- Braun, Marschall: BOTL, 2003
  http://www4.in.tum.de/~marschal
- Knapp, Merz et al.: Hugo, 2001
  http://www.pst.ifi.lmu.de/projekte/hugo

Resources



Jan Jürjens, Secure Systems Development with UML, Springer 04

Tutorials: Nov.: SISBD (Malaga), ISSRE (Rennes).

Spring School: May 2005, Carlos IV Univ. Madrid

Workshops: WITS05@POPL05, CSDUML05

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

http://www.umlsec.org

(user Participant, password Iwasthere)

Finally

We are always interested in industrial challenges for our tools, methods, and ideas to solve practical problems.

More info: http://www4.in.tum.de/~secse

Contact me here or via Internet.

Thanks for your attention !

BREAK !

Note:

We are always interested in industrial challenges for our tools, methods, and ideas to solve practical problems.

More info: http://www4.in.tum.de/~secse

Contact me here or via Internet.

Source code for checking the UMLsec Stereotype <<guarded access>>

```
/**
 * Created by JCreator.
 * User: shasha meng
 * Date: Jan 22, 2003
 * Time: 8:59:27 PM
 * To change this template use Options | File Templates.
 */
package tum.umlsec.mdrsupport;

import org.omg.uml.UmlPackage;
import org.omg.uml.foundation.datatypes.*;
import org.omg.uml.foundation.core.*;
import org.omg.uml.behavioralelements.commonbehavior.*;
import org.omg.uml.behavioralelements.activitygraphs.*;
import org.omg.uml.behavioralelements.statemachines.*;
import javax.jmi.model.GeneralizableElement;
import javax.jmi.model.Attribute;
import javax.jmi.reflect.RefPackage;
import javax.jmi.reflect.RefClass;
import java.util.Iterator;
import java.util.List;
import java.util.ArrayList;
import java.util.ListIterator;
import java.util.Vector;
import java.util.HashMap;
```

```
public class MdrUmlParser_guardedaccess {
boolean condition1 = true;
boolean condition2 = true;
HashMap obj_Val = new HashMap();
//defines a list for the names of all the classes
ArrayList list_objname = new ArrayList();
ArrayList list_all = new ArrayList();
UmlPackage root ;
CorePackage corePackage ;
ActivityGraphsPackage activityPackage;
StateMachinesPackage stateMachines;
TransitionClass transitionClasses;
//initial
public void init (MdrUmlManager _manager) {
  manager = _manager;
  root = (UmlPackage) manager.getPackageModelContainer();
  corePackage = root.getCore();
  activityPackage = (ActivityGraphsPackage) root.getActivityGraphs();
  stateMachines = (StateMachinesPackage)
     activityPackage.getStateMachines();
  transitionClasses = (TransitionClass) stateMachines.getTransition();
}
```

```
public void dump() {
  //list all the tagged values in the diagram
  System.out.println ("======= All TaggedValue");
  TaggedValueClass tagvalueClasses = (TaggedValueClass)
     corePackage.getTaggedValue();
  for (Iterator it_Tag_V = tagvalueClasses.refAllOfClass().iterator(); it_Tag_V.hasNext();) {
    TaggedValue tagValue = (TaggedValue) it_Tag_V.next();
    //if the tagged type equal to "guard"
    //reads all the names of the classes with the tagged type equal to "guard" //in a list
      list_objname.
    if ((tagValue.getType()).getTagType().equals("guard")) {
        String objname = null;
        //defines a list for the tagged values of the "guard" of every class
        ArrayList list_valname = new ArrayList();
      try {
              //list all the classes in the diagram
              UmlClass uml_C = (UmlClass) (tagValue.getModelElement());
              //list the name of the classes
              objname = uml_C.getName();
              //print the name of the classes
        System.out.println ("modelElementName is "+objname);
        if (objname!=null)
          //the name of the class is added to the list_objname
          list_objname.add(objname);
```

```
for (Iterator it_tagVa_A = (tagValue.getDataValue()).iterator();
    it_tagVa_A.hasNext();) {
        String tagValue_Da_A = (String) it_tagVa_A.next();
        //list the tagged values of the "guard" for every class
         System.out.println ("TaggedValue (Data) von guard is
    "+tagValue_Da_A);
        //Then it reads all the tagged values of the tagged type "guard" of
    //every class in the list list_objname in a list with the name //list_valname.
        if (tagValue_Da_A!=null)
          //all the tagged values of the "guard" are added to the //list_valname
          list_valname.add(tagValue_Da_A);
      }
     } catch (Exception ep) {
            System.out.println ("exception at find class : "+ep.getMessage());
     }
     //It defines a hash map with the name obj_Val, the key of it is the name
    //of the classes objname in the list list_objname, and the value of the
    //key is the list list_valname.
      if (list_valname != null && objname != null)
        obj_Val.put(objname,list_valname);
   }
 }
```

```
System.out.println("========= All Transitions");
  //here begins the Test.
  //reads the value of the obj of the guard of the transition in the activity
// diagram.
  //reads the action of this transition and checks whether it is equal to the //value of the key
      according to the hash map obj_Val.
  for (Iterator it_Tran= transitionClasses.refAllOfClass().iterator(); it_Tran.hasNext();) {
    Transition transition = (Transition) it_Tran.next();
    Guard guard = (Guard) transition.getGuard();
    Action action = (Action) transition.getEffect();
    if (guard!=null&&action!=null) {
      BooleanExpression b_Expression = (BooleanExpression) guard.getExpression();
      ActionExpression a_Expression = (ActionExpression) action.getScript();
      String bodyname_a = (String) a_Expression.getBody();
      String bodyname = (String) b_Expression.getBody();
      int i = bodyname.indexOf("=");
        int j = bodyname_a.indexOf(".");
        String before_a;
        if (j!=-1) {
          before_a=bodyname_a.substring(0,j);
        } else {
          before_a = new String (bodyname_a);
        }
        System.out.println ("bodyname_a before is "+ before_a);
      String before = bodyname.substring(0,i-1);
      String after = bodyname.substring(i+2);
```

```
if (before.equals("obj")) {
        try {
                if (list_objname.contains(after)) {
                  ArrayList list_val = (ArrayList) obj_Val.get(after);
                          condition1 = list_val.contains(before_a);
                          System.out.println ("condition1 is "+condition1);
                          if(!condition1)
                            condition2 = false;
                }
          System.out.println ("Body after is "+after);
              } catch (Exception ep) {
                System.out.println ("exception at test : "+ep.getMessage());
              }
      }
    }
  }
  if (condition1&&condition2)
    System.out.println ("The system satisfies <<guarded access>>.");
  else
    System.out.println ("The system does not satisfy <<guarded access>>.");
}
private MdrUmlManager manager;
}
```