Experiences in Applying MDE to Telescope and Instrument Control System Domain

L. Andolfato, R. Karban, M. Schilling,

H. Sommer, M. Zamparelli, G. Chiozzi

European Southern Observatory (www.eso.org)

Outline

ESO
European Organisation for Astronomical Research in the Southern Hemisphere

- Introduction to the Telescope and Instrument Domain
- Lessons Learned
- Projects

Paranal Observatory

Cerro Paranal, 2635m, Atacama desert, Chile.

ALMA Observatory

Atacama Large Millimeter Array, 5000m, Atacama desert, Chile.

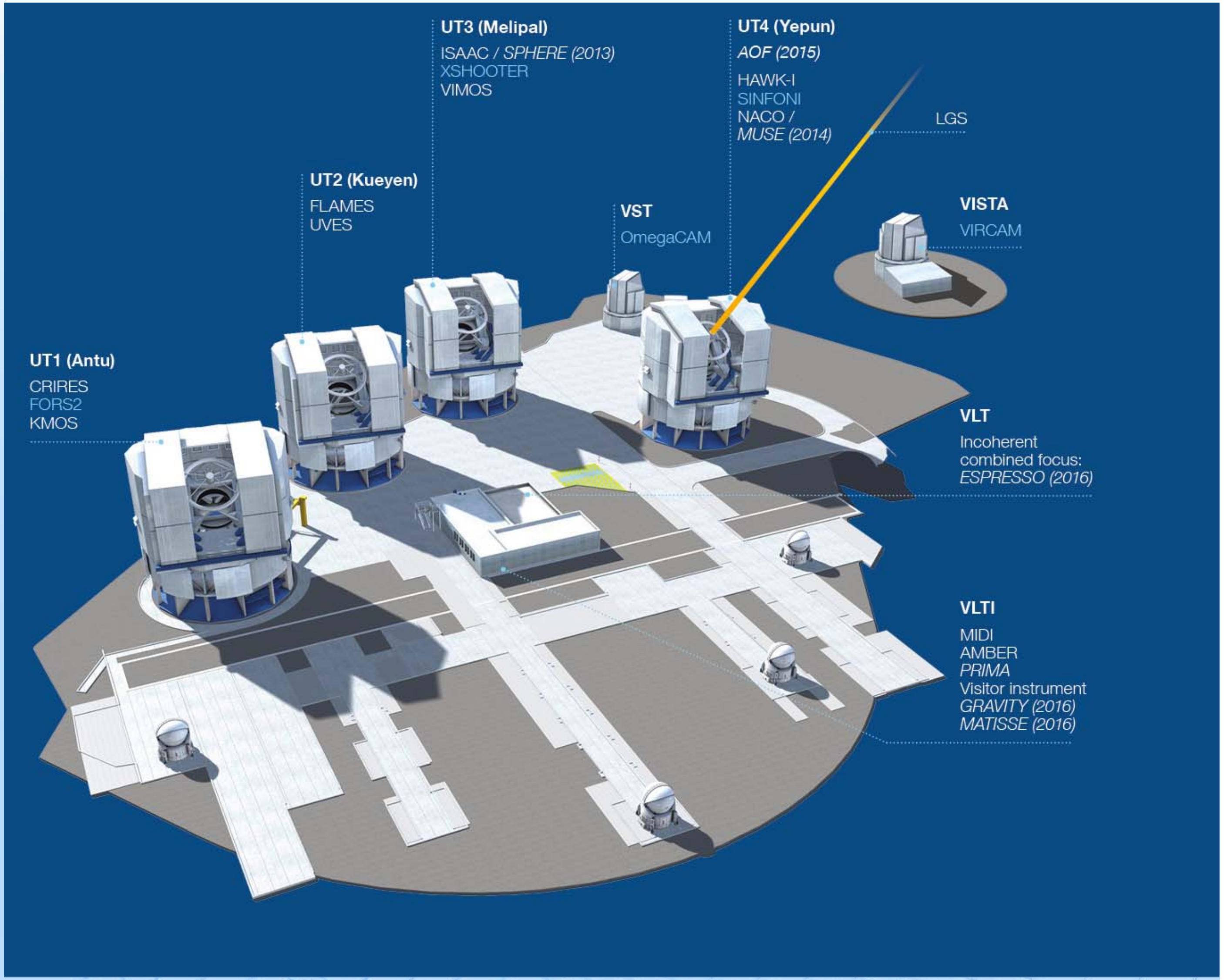
Telescope and Instrument Domain

Telescope

- Pointing
- Tracking
- Auto Guiding
- Field Stabilization
- Adaptive Optics
- Laser Guide Star
- Active Optics
- Temperature control
- Dome Tracking

Interferometer

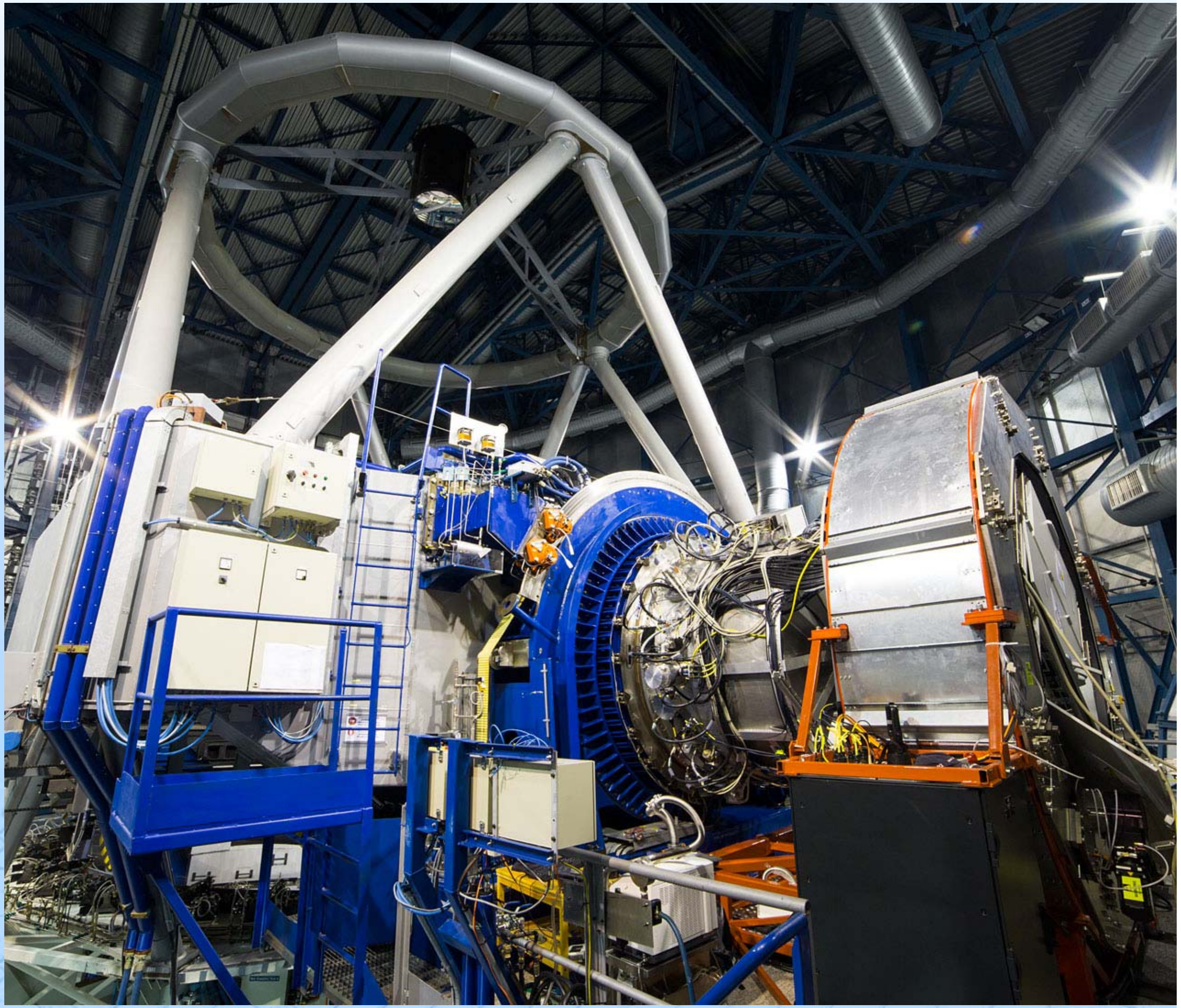
- Image & Pupil Stabilization
- Pupil Relay
- Fringe Search & Tracking



Telescope and Instrument Domain

Instruments

- Drive the Observation
- Create Images
- Analyse Images for intensity, size, morphology, or spectral content
- Verify Scientific Data against Calibration
- Archive Scientific Data



Telescope and Instrument Domain

	PARANAL (VLT)	ALMA
Observatory Construction Time	10 years (1988-1998)	14 years (1998-2012)
Observatory Expected Lifetime	>20 years	>30 years
New Instruments / Receivers	~ Every year	~ Every 2 years
Initial SW Platform	HP-UX, HP RTAP C, TCL/TK	Linux, VxWorks C++, Java, Python CORBA
Current SW Platform	Linux, VxWorks C++, C, TCL/TK CCS (TCP/IP)	Linux, Linux RTAI C++, Java, Python CORBA, DDS
Technical Downtime	< 3% observation time (night time operation)	< 5% observation time (24h operation)

Semantic Consistency

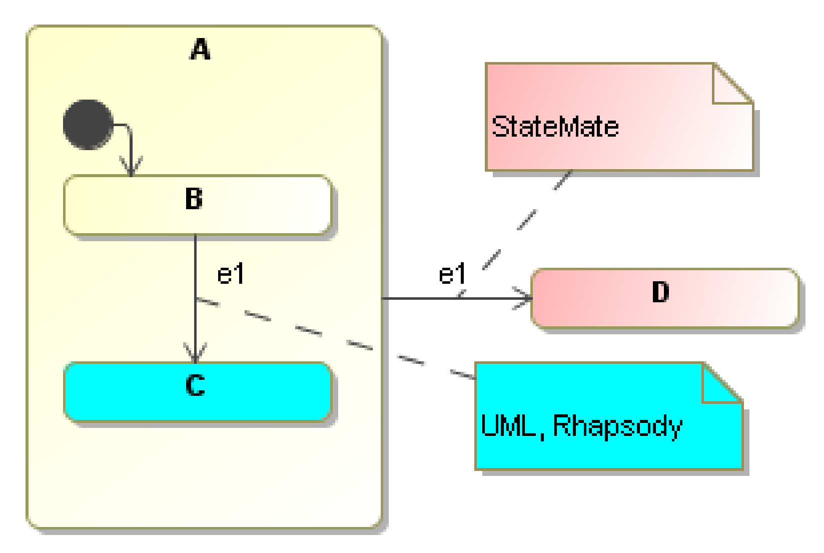
Problem: Different tools/libraries interpret models differently.

Context: Model reuse (Simulation, Model Checking, Code Generation).

Lessons Learned: Select a (standard) execution semantic and stick to it in the whole tool-chain.

Examples: Statecharts semantic.

SCXML (StateChart XML)
Defines syntax and semantic for Statecharts execution.



Modifying Behaviour@Runtime

Problem: Efficiently apply last minute changes.

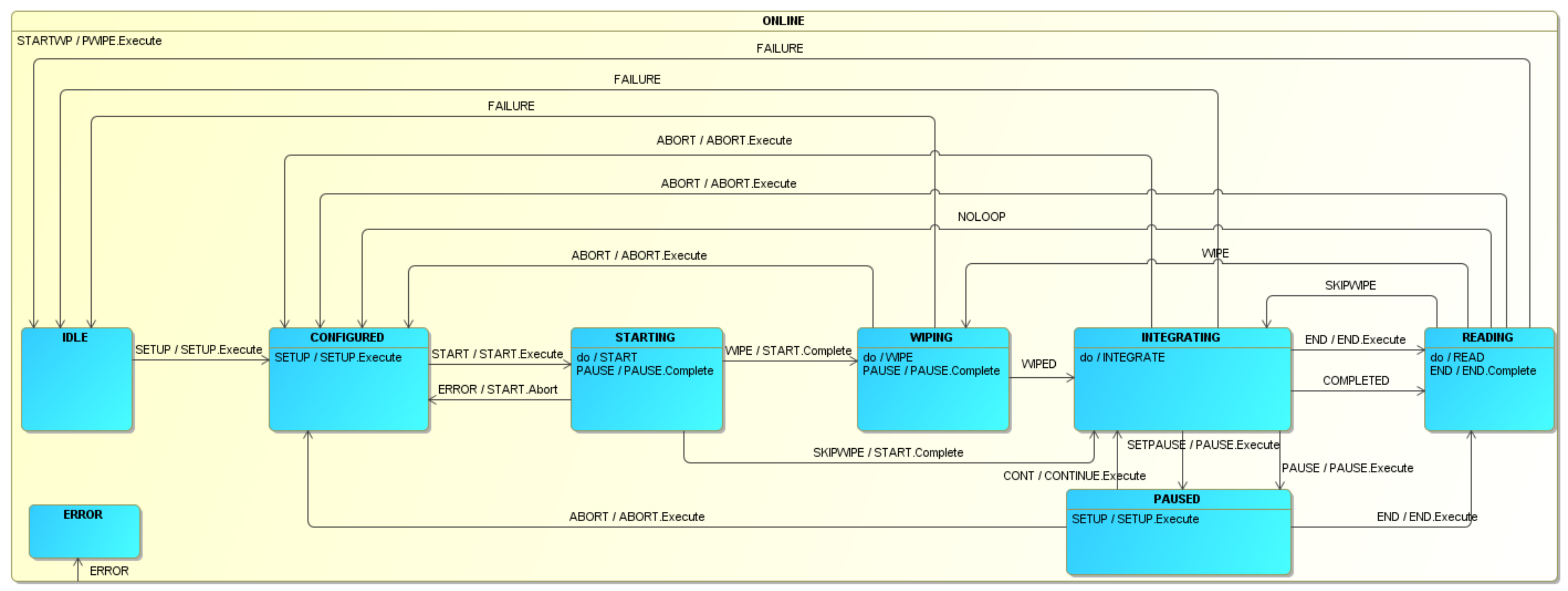
Context: Large systems that can be fully integrated only once at the final remote location.

Lessons Learned:

- Use a mix of compiled and script languages.
- Introduce the ability to change behaviour at runtime.

Modifying Behaviour@Runtime

Examples: Acquisition sequences.



Performance Indicator

Problem: How much **time** should we spend in **modeling**?

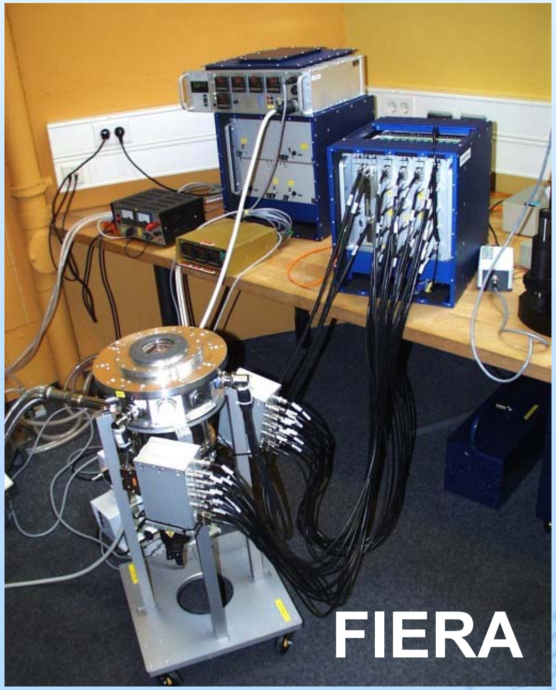
Context: Some dev would model forever others never. Some project managers consider modeling expensive.

Lessons Learned: Constantly measure the **ROI**.

1)	Modeling Cost		Should be **less** than the cost of developing by hand the part of application that is generated from the model.
2)	Number of Generated Applications		Should be **big enough** to pay off the investment in the modeling infrastructure.

Performance Indicator (cont)

Example: Comparing control SW for two detector controllers.



- Similar projects, NGC slightly more complex, ~same team.
- Number of NGC applications based on MDE: 5
- Theoretical pay-off thresholds: 3.3 applications.

Model Validation

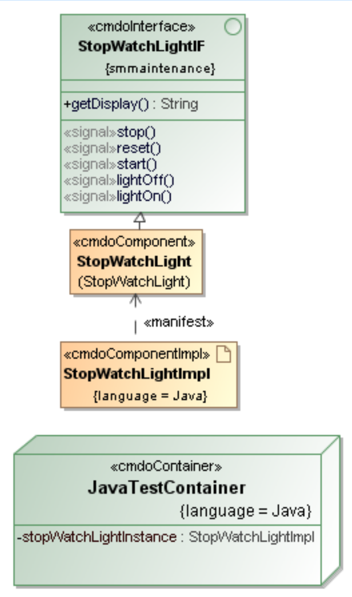
Problem: Is my model a valid instance of my meta-model?

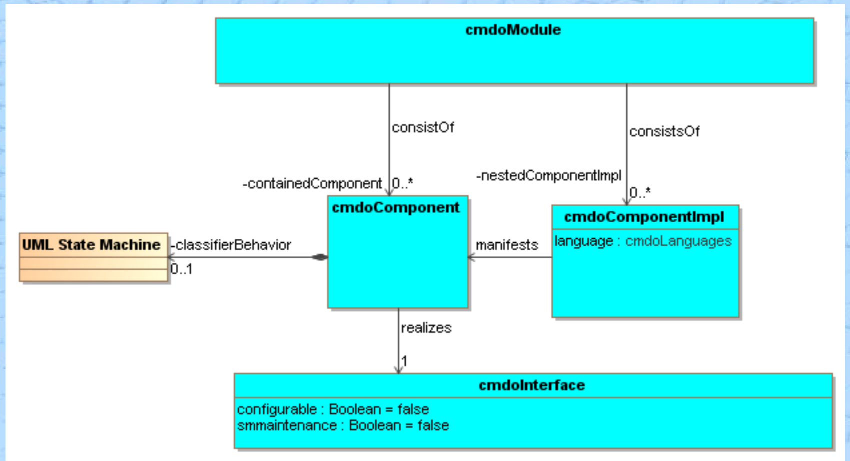
Context: Many modeling mistakes are discovered only during transformation, execution or compilation.

Lessons Learned:

- Assign an expert to each project.
- Automatic model validation while building the model.

Examples:





Tool-chain Obsolescence

Problem: Tool-chain evolves with time.

Context: Development last >10 yrs, maintenance >30 yrs.

Lessons Learned:

- Vendor independent representation of the model.
- M2M transformation know-how.
- Transformation languages with large user base, open source, and compliant with standards.
- **Archive modeling tools and runtime environments!**

SW Evolution

Problem: SW platforms, standards, guidelines, document templates evolve with time.

Context: Development last > 10 yrs, maintenance > 30 yrs, need to support multiple (versions of) SW platforms.

Lessons Learned: M2T approach simplifies a lot maintenance as long as the transformation can be modified (and the meta-model is stable enough).

Generated Code vs. Libraries

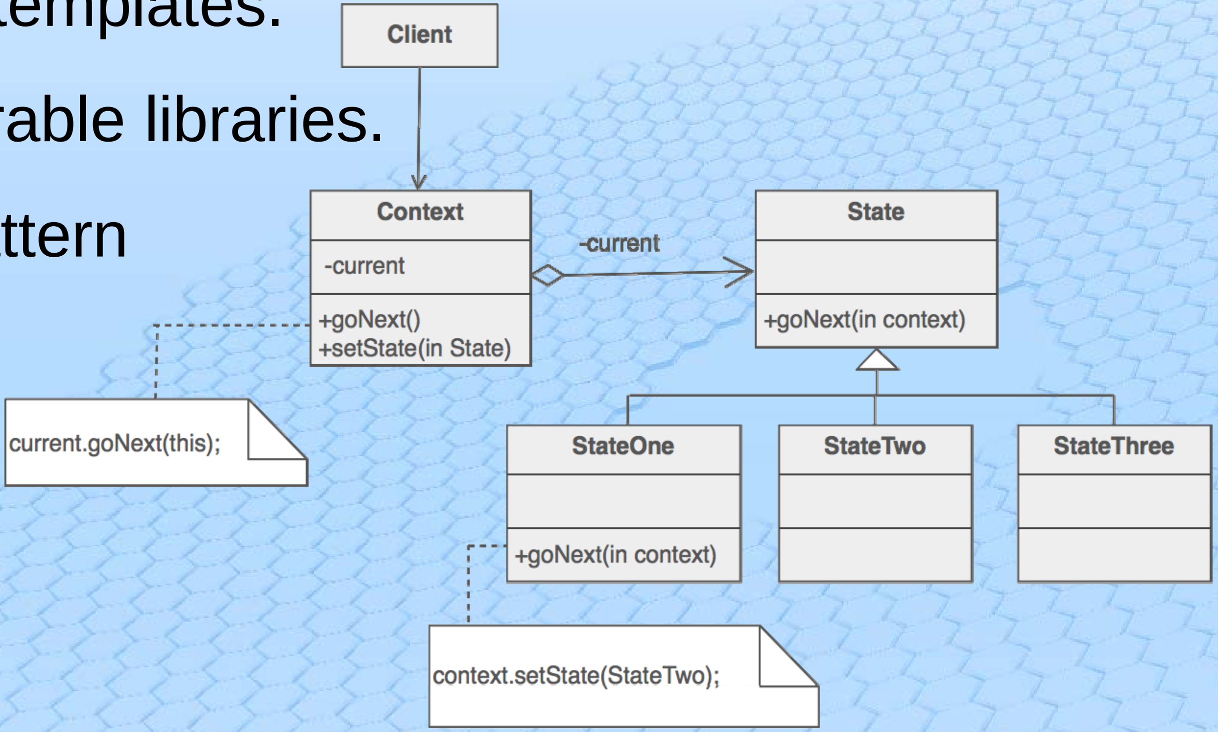
Problem: Should libraries be replaced by M2T transformation?

Context: SW rebuild takes time.

Lessons Learned: Use libraries.

- Refactor M2T templates.
- Prefer configurable libraries.

Examples: State Pattern



Projects

Auxiliary Telescope

Period: 1999-2005

FTE: 14

New Components: 29

(11 DSL based,

0 UML based)

UML State/Tran: 0/0

PRIMA

Period: 2002-2008

FTE: 14.4

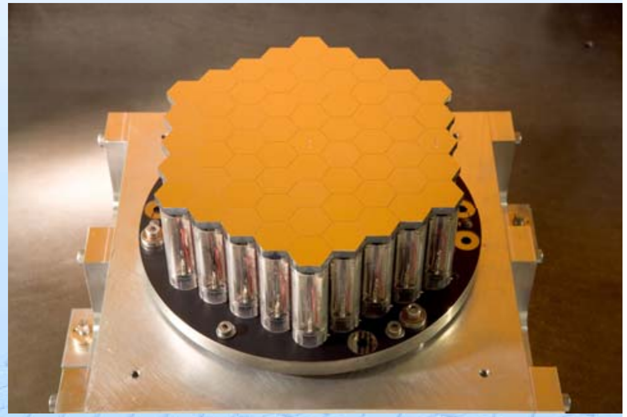
New Components: 53

(15 DSL based,

10 UML based)

UML State/Tran: 252/864

APE



Period: 2005-2009

FTE: 17.35

New Components: 37

(13 DSL based,

11 UML based)

UML State/Tran: 432/1260

Questions?

Acknowledgments

Nicolas Beneš, Nicola Migliorini, Alexis Tajeda, Arturo Hoffstadt,

Cristian Morales, Joao Lopez

ESO

European Organisation
for Astronomical
Research in the
Southern Hemisphere

Backup Slides

Future Plans

- Achieve **Semantic Consistency**

  (Model Checker for SCXML).
- Improve **Model Validation**

  (Conceptual Modeling Ontology and Framework).
- Support for **new SW Platforms**.

Modeling Language

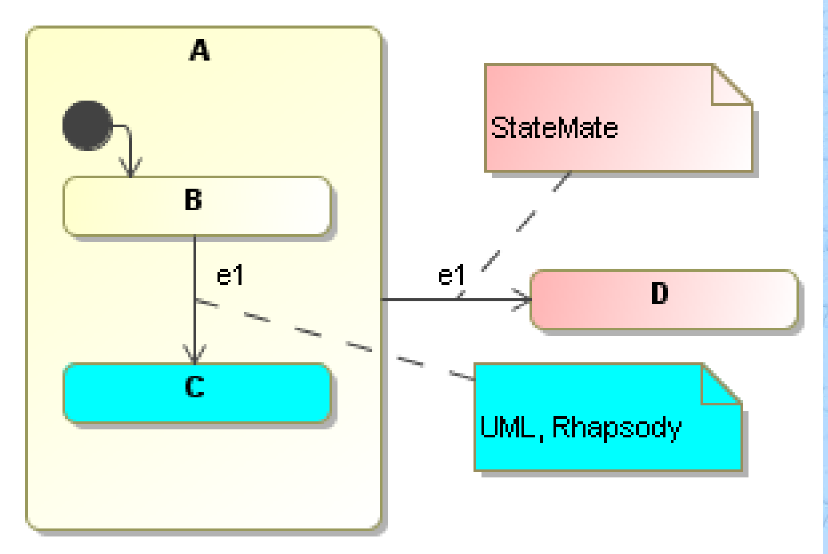
Problem: Graphical or textual language?

Context:

- SW development: many (large) changes.
- SW maintenance: few (small) changes per year.
- Not everybody likes graphical languages.

Lessons Learned: Use both.

```
<state id=“A" initial=“B">
   <state id=“B">
      <transition event="e1" target=“C“/>
   </state>
   <state id=“C">
   </state>
   <transition event="e1" target=“D“/>
</state>
 <state id=“D">
 </state>
```



Archive Generated Artefacts

Problem: Do we archive what is generated?

Context: Pressure to avoid observatory downtime.

Lessons Learned: Yes we do because

- Speed-up the build process.
- Makes faster the comparison (diff).

Performance Indicator

(Avg. Cost for N Traditional Appl.) ≥ (Avg. Cost for N MDE Appl.)

N*(**TMI**+TMD) ≥ N*(**TMI**+TM) + (TMMDEF + TMMNAV + TTPL)

N*TMD ≥ N*TM + (TMMDEF + TMMNAV + TTPL)

(TM ≤ TMD) and (N big enough)

N = number of applications

TMI = average cost for the model independent part of the application

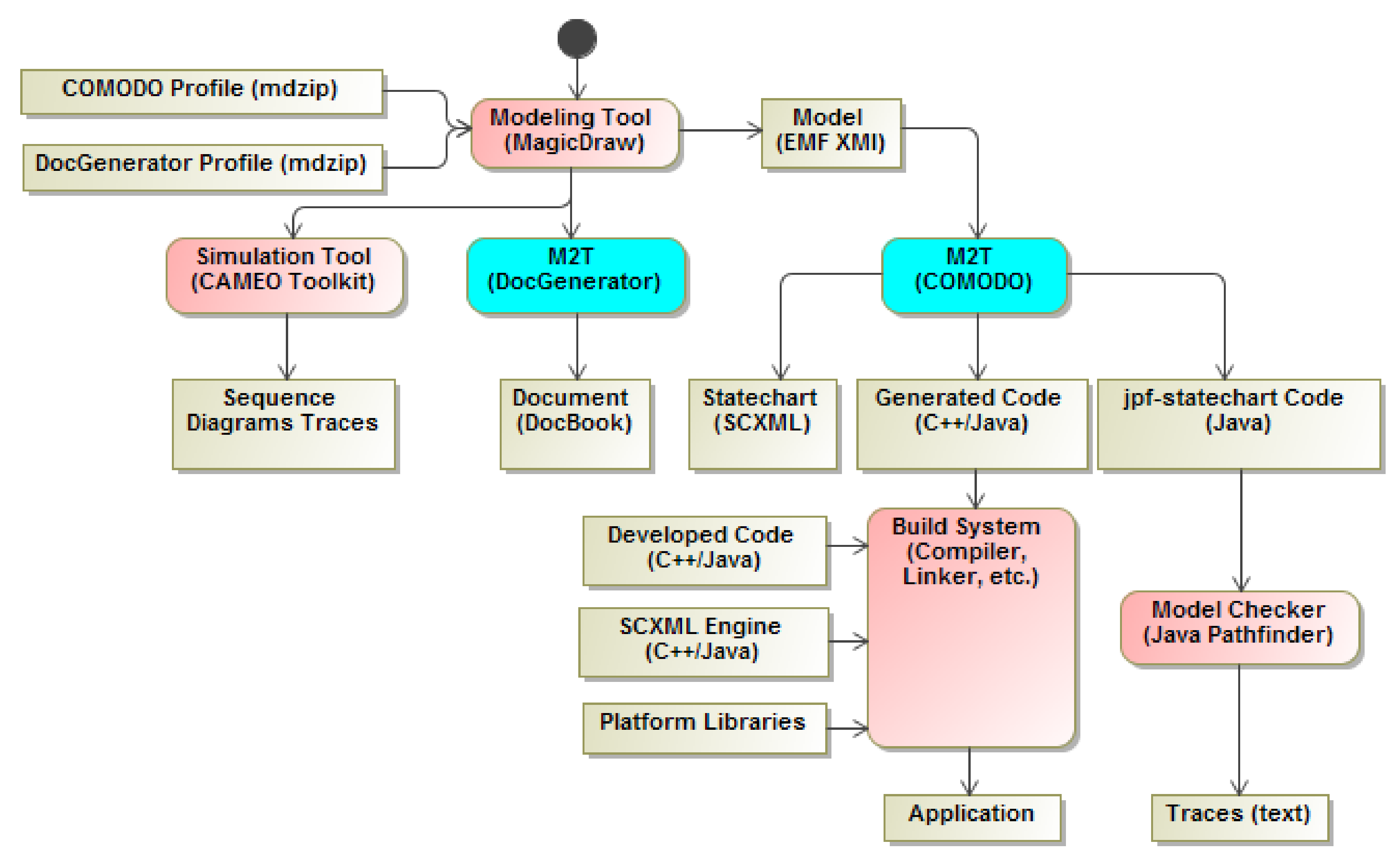
TMD = average cost for the model dependent part of the application

TM = average cost for modeling one application

TMMDEF = cost for mm definition

TMMNAV = cost for mm navigation, TTPL cost for the templates

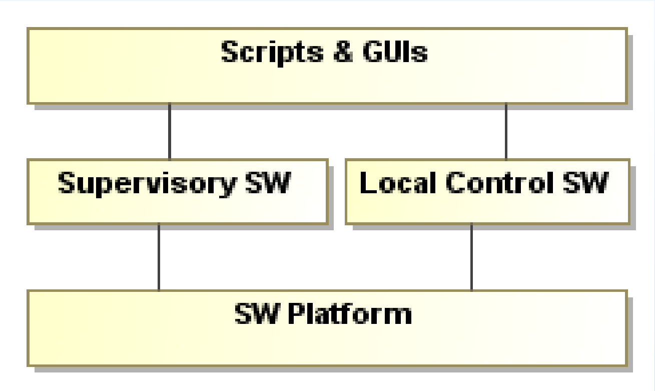
Tools


COMODO Profile (mdzip)
DocGenerator Profile (mdzip)
Modeling Tool (MagicDraw)
Model (EMF XMI)
Simulation Tool (CAMEO Toolkit)
M2T (DocGenerator)
M2T (COMODO)
Sequence Diagrams Traces
Document (DocBook)
Statechart (SCXML)
Generated Code (C++/Java)
jpf-statechart Code (Java)
Developed Code (C++/Java)
SCXML Engine (C++/Java)
Platform Libraries
Build System (Compiler, Linker, etc.)
Model Checker (Java Pathfinder)
Application
Traces (text)

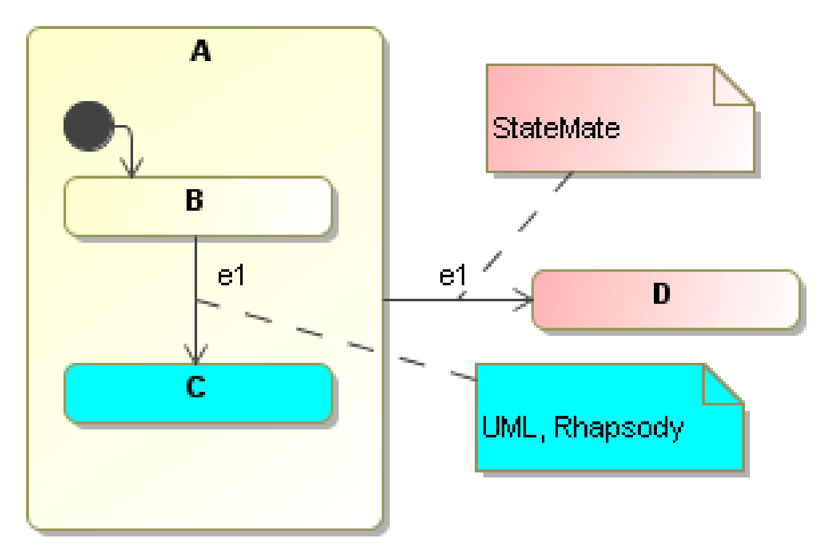
Tools

Purpose	Tool	Description
Modeling	MagicDraw	NoMagic
Simulation	CAMEO Simulation Toolkit	NoMagic, SCXML based
Documentation	Model Based Document Generator	MD plug-in developed in house to transform SysML models into DocBook XML files.
Code Generation	COMODO	Java application developed in house based on EMF and Xpand/Xtend to transform UML models into SCXML based applications.
Model Verification	Java Pathfinder (jpf-statechart)	NASA Ames Model Checker for Java.
Statecharts Engine	Apache Commons SCXML, scxml4cpp	For Java by Apache. For C++ developed in house.
Model Validation	Conceptual Modeling Framework	Transforms ontology into SysML profile and MD plug-in.

SW Architecture



W3C SCXML Standard

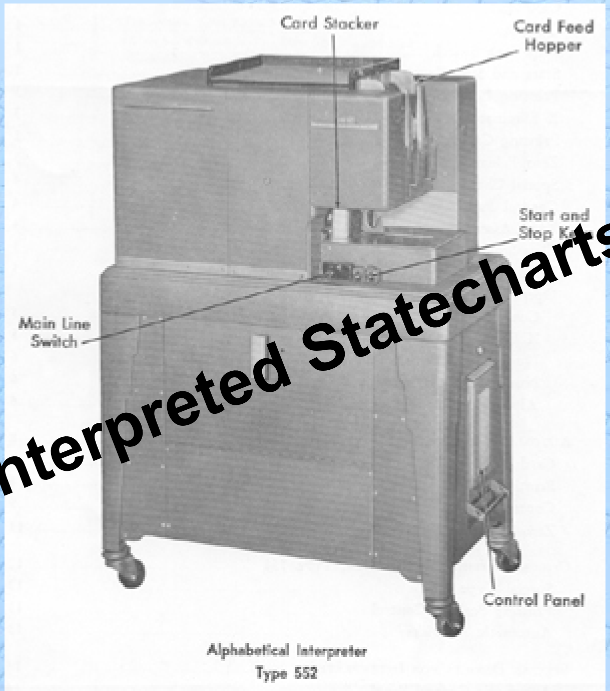


Crane & Dingel paper on differences between Statecharts syntax & semantics: “Not all models are created equal”

State Chart XML

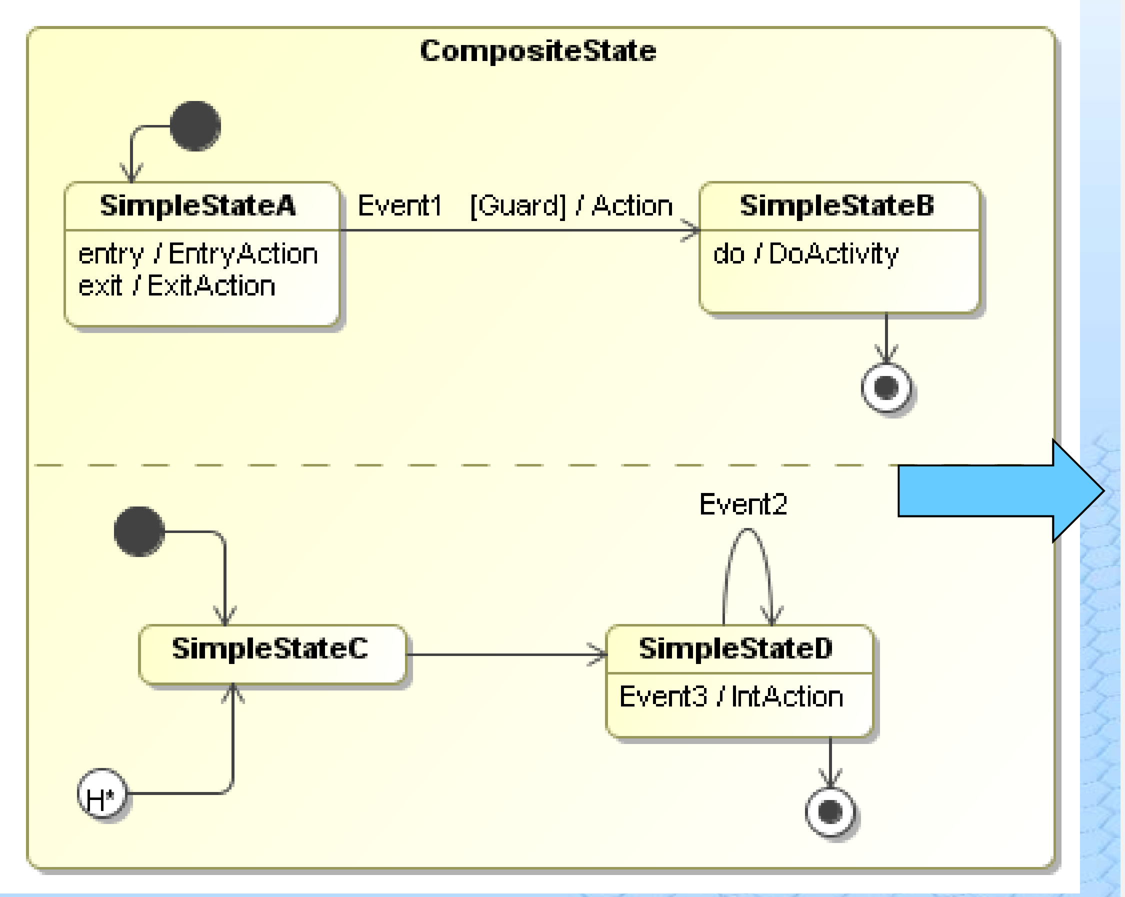
Supported by IBM, HP, Microsoft, Nokia

```
<state id=“A" initial=“B">
   <state id=“B">
      <transition event="e1" target=“C“/>
   </state>
   <state id=“C">
   </state>
   <transition event="e1" target=“D“/>
</state>

 <state id=“D">
 </state>
```



Interpreted Statecharts

UML2SCXML Mapping



<state> id="" initial="" </state>

<parallel> </parallel>

<transition> event="" guard="" target="" </transition>

<initial> </initial>

<final> </final>

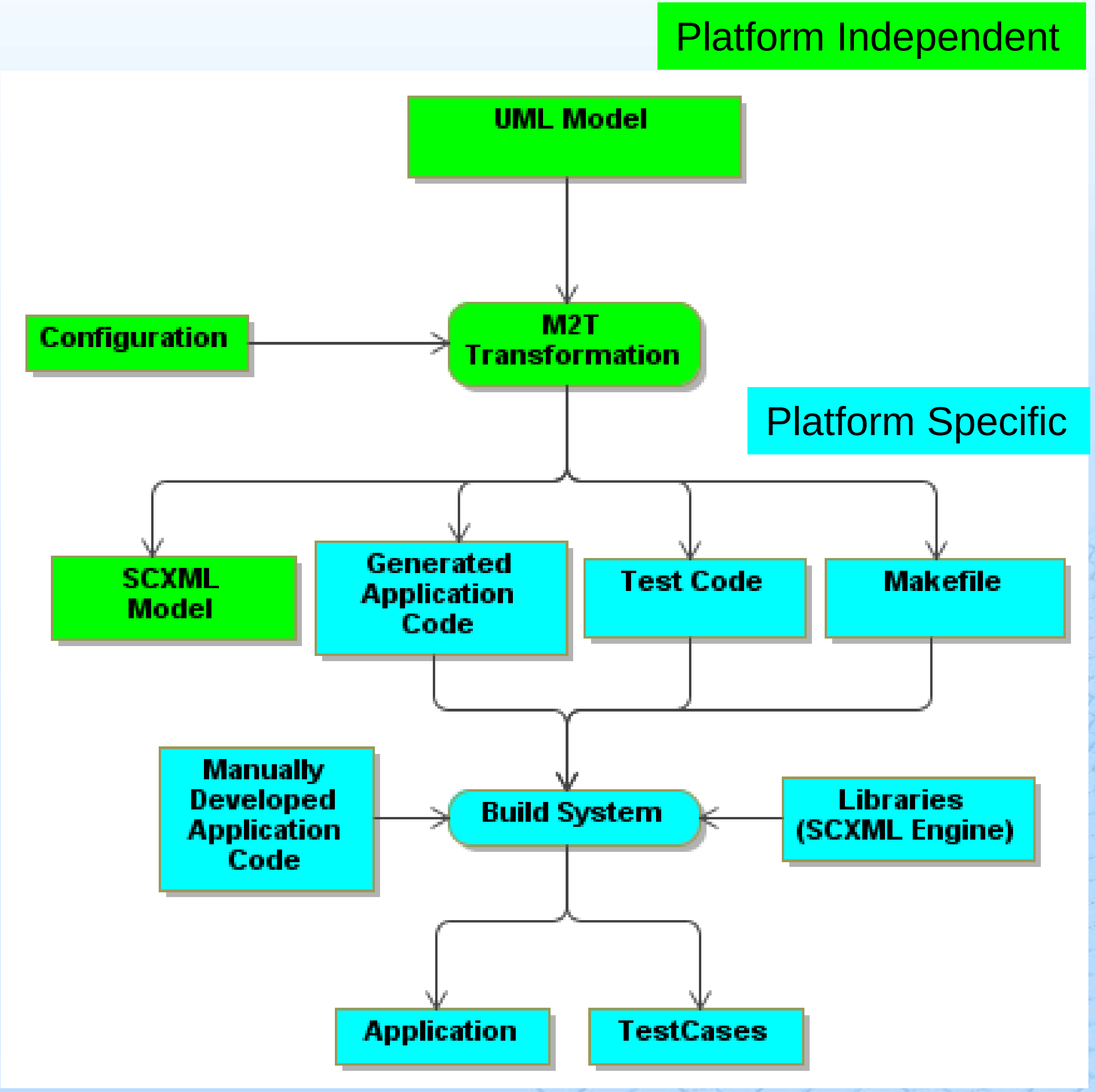
<history> type="deep|shallow" </history>

<onentry> </onentry>

<onexit> </onexit>

<invoke> </invoke>

The Data Flow



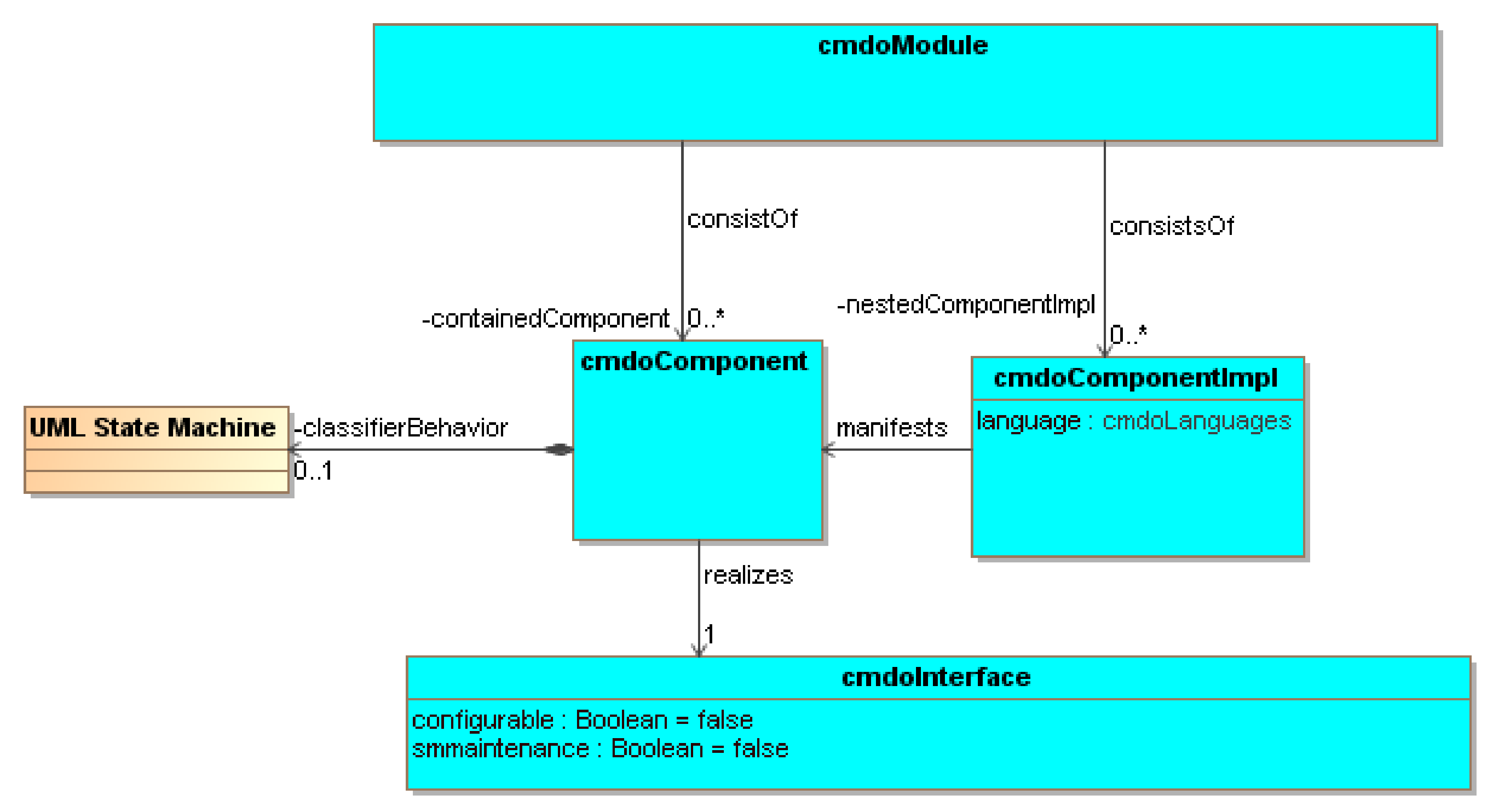
COMODO Profile and MetaModel

Cross Platform M2T Transformation (EMF + Xpand)

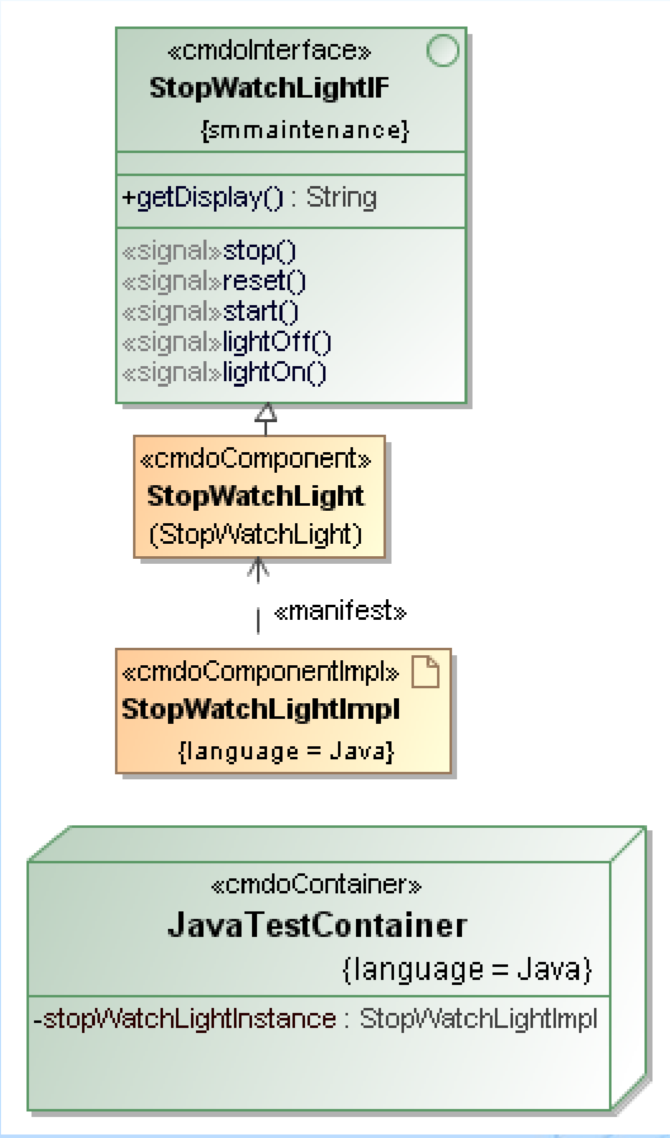
Clear separation between generated and manually developed code

Generated Application uses SCXML Model and SCXML Engine library

COMODO Profile for UML



StopWatch Example



«cmdoInterface»
StopWatchLightIF
{smmaintenance}

+getDisplay() : String

«signal»stop()
«signal»reset()
«signal»start()
«signal»lightOff()
«signal»lightOn()

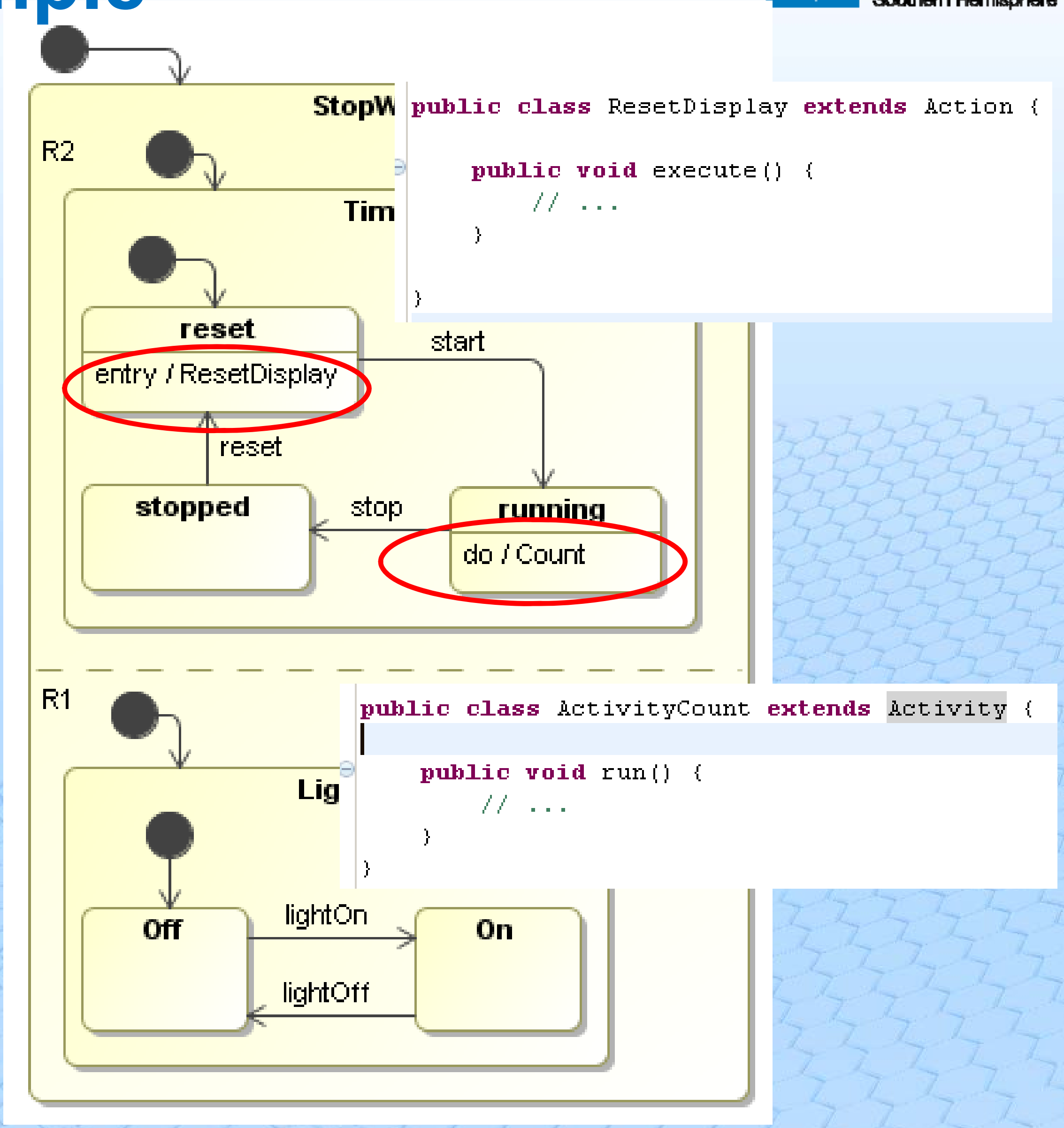
«cmdoComponent»
StopWatchLight
(StopWatchLight)

«manifest»

«cmdoComponentImpl»
StopWatchLightImpl
{language = Java}

«cmdoContainer»
JavaTestContainer
{language = Java}

-stopWatchLightInstance : StopWatchLightImpl



StopW

R2

Tim

reset

entry / ResetDisplay

start

reset

stopped

stop

running

do / Count

R1

Lig

Off

lightOn

On

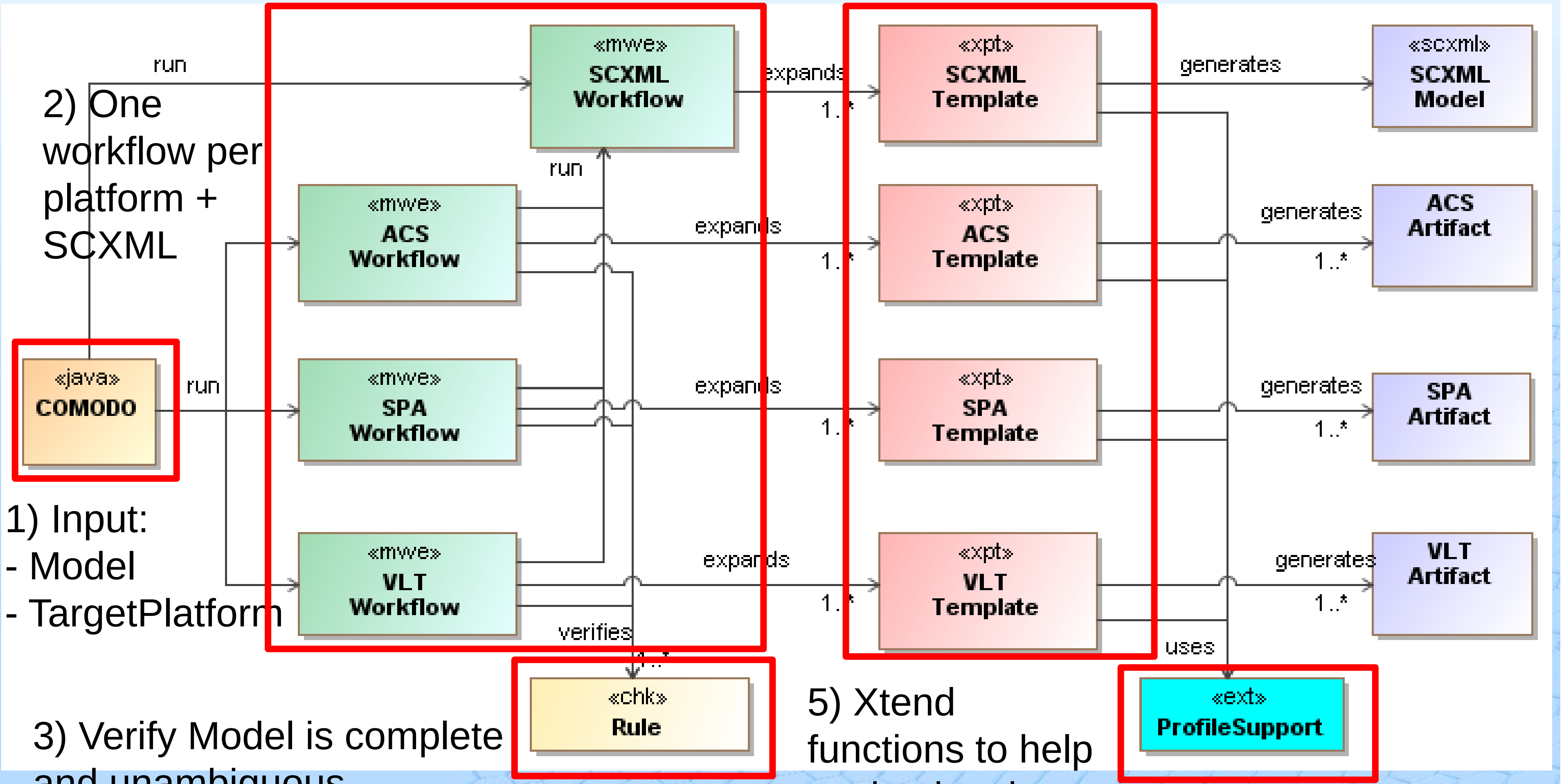
lightOff

```
public class ResetDisplay extends Action {

    public void execute() {
        // ...
    }

}
```

```
public class ActivityCount extends Activity {

    public void run() {
        // ...
    }
}
```

Cross Platform Model2Text Transformations

4) Xpand Templates generates the artifacts using Xtend functions



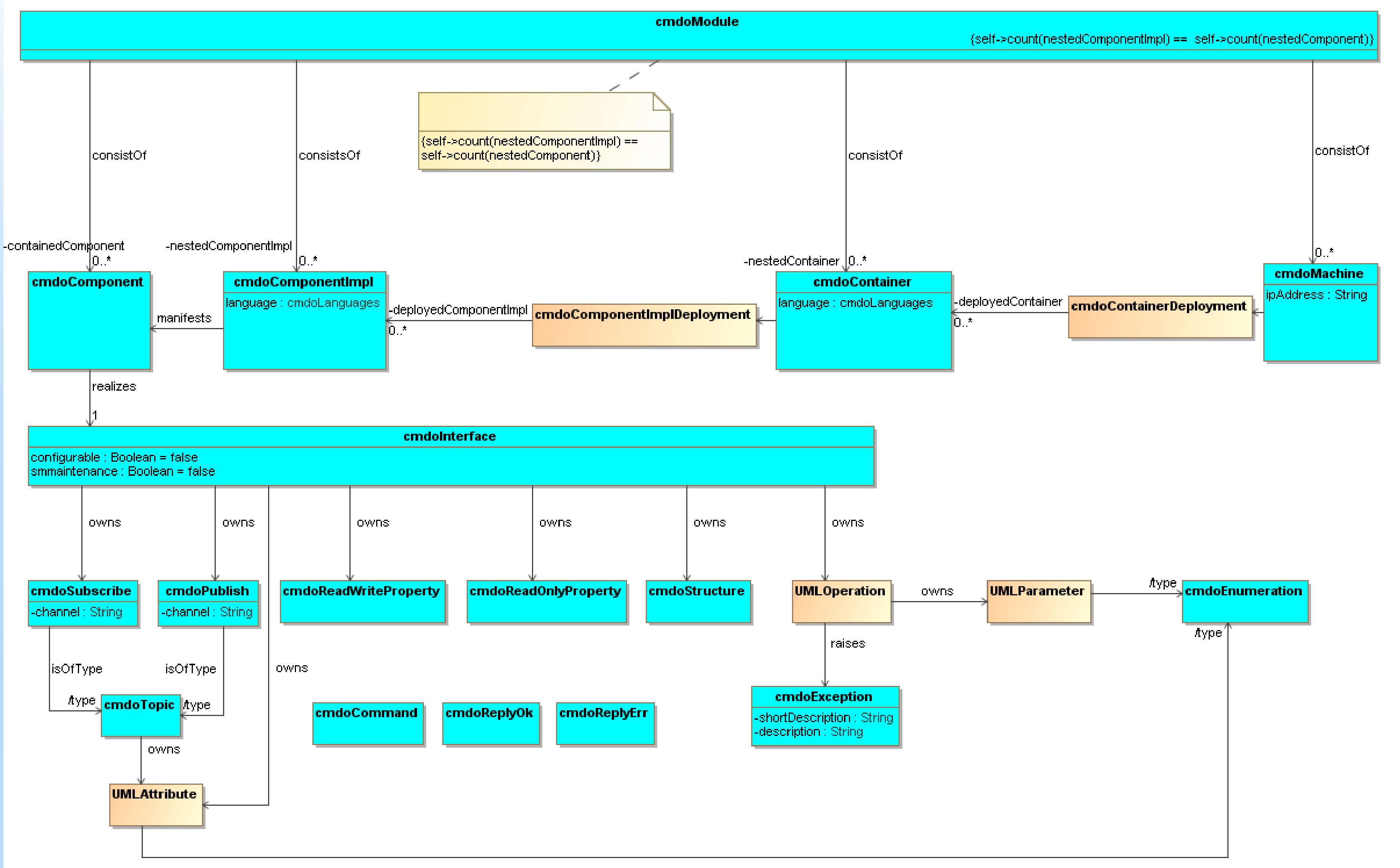
2) One workflow per platform + SCXML

1) Input:
- Model
- TargetPlatform

3) Verify Model is complete and unambiguous

5) Xtend functions to help navigating the model

COMODO MetaModel



cmdoModule
{self->count(nestedComponentImpl) == self->count(nestedComponent)}

{self->count(nestedComponentImpl) ==
self->count(nestedComponent)}

consistOf
consistsOf
consistOf
consistOf
-containedComponent 0..*
-nestedComponentImpl 0..*
-nestedContainer 0..*
0..*

cmdoComponent
manifests
cmdoComponentImpl
language : cmdoLanguages
-deployedComponentImpl 0..*
cmdoComponentImplDeployment
cmdoContainer
language : cmdoLanguages
-deployedContainer 0..*
cmdoContainerDeployment
cmdoMachine
ipAddress : String

realizes
1

cmdoInterface
configurable : Boolean = false
smmaintenance : Boolean = false

owns
owns
owns
owns
owns
owns

cmdoSubscribe
-channel : String
cmdoPublish
-channel : String
cmdoReadWriteProperty
cmdoReadOnlyProperty
cmdoStructure
UMLOperation
owns
UMLParameter
/type
cmdoEnumeration
/type

isOfType
isOfType
owns
raises
/type
cmdoTopic
/type
owns
UMLAttribute

cmdoCommand
cmdoReplyOk
cmdoReplyErr
cmdoException
-shortDescription : String
-description : String