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

• 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
<table>
<thead>
<tr>
<th></th>
<th>PARANAL (VLT)</th>
<th>ALMA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Observatory Expected Lifetime</strong></td>
<td>&gt;20 years</td>
<td>&gt;30 years</td>
</tr>
<tr>
<td><strong>New Instruments / Receivers</strong></td>
<td>~ Every year</td>
<td>~ Every 2 years</td>
</tr>
<tr>
<td><strong>Initial SW Platform</strong></td>
<td>HP-UX, HP RTAP C, TCL/TK</td>
<td>Linux, VxWorks C++, Java, Python CORBA</td>
</tr>
<tr>
<td><strong>Current SW Platform</strong></td>
<td>Linux, VxWorks C++, C, TCL/TK CCS (TCP/IP)</td>
<td>Linux, Linux RTAI C++, Java, Python CORBA, DDS</td>
</tr>
<tr>
<td><strong>Technical Downtime</strong></td>
<td>&lt; 3% observation time (night time operation)</td>
<td>&lt; 5% observation time (24h operation)</td>
</tr>
</tbody>
</table>
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**.

<table>
<thead>
<tr>
<th></th>
<th>Modeling Cost</th>
<th>Should be <strong>less</strong> than the cost of developing by hand the part of application that is generated from the model.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>Number of Generated Applications</td>
<td>Should be <strong>big enough</strong> to pay off the investment in the modeling infrastructure.</td>
</tr>
</tbody>
</table>
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
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.

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


\[ N \times (TMI + TMD) \geq N \times (TMI + TM) + (TMMDEF + TMMNAV + TTPL) \]

\[ N \times TMD \geq N \times TM + (TMMDEF + TMMNAV + TTPL) \]

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

- Scripts & GUIs
- Supervisory SW
- Local Control SW
- SW Platform
Crane & Dingel paper on differences between Statecharts syntax & semantics: “Not all models are created equal”

State Chart XML
Supported by IBM, HP, Microsoft, Nokia

Interpreted Statecharts

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

L. Andolfato | 2014-10-02
<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

Platform Independent

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
StopWatch Example

```java
public class ResetDisplay extends Action {
    public void execute() {
        // ...
    }
}
```

```java
public class ActivityCount extends Activity {
    public void run() {
        // ...
    }
}
```
Cross Platform Model2Text Transformations

1) Input:
   - Model
   - TargetPlatform

2) One workflow per platform + SCXML

3) Verify Model is complete and unambiguous

4) Xpand Templates generates the artifacts using Xtend functions

5) Xtend functions to help navigating the model