Experiences in Modeling [Telescope] Control Systems

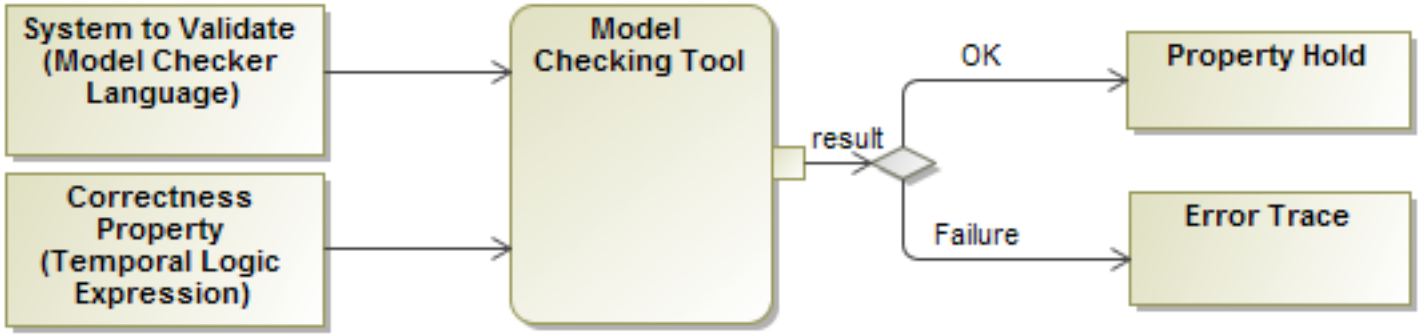
Luigi Andolfato, Robert Karban, Gianluca Chiozzi, Michele Zamparelli

Table of Content

Model Checking

Model Transformations

Model Checking

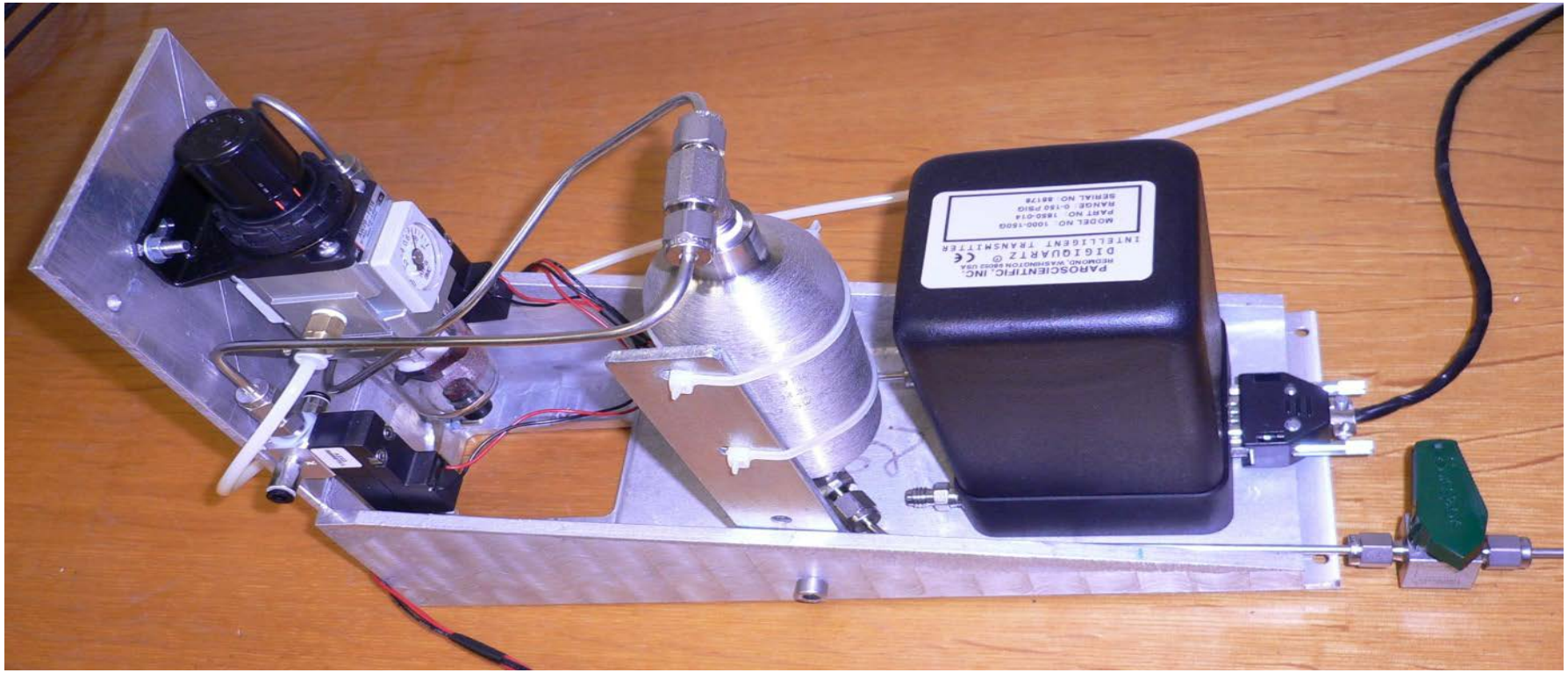


- Discovered by Clarke, Emerson, and Sifakis in 1980 (2008 Turing Award)
- System to be validated + correctness property are transformed into nondeterministic finite automata and executed simultaneously
- A model checker checks for an input string that is accepted by both automata: the concurrent program to check and the negation of the correctness property to verify
- If such input string is found, it represents a counter-example otherwise it means that the property holds.

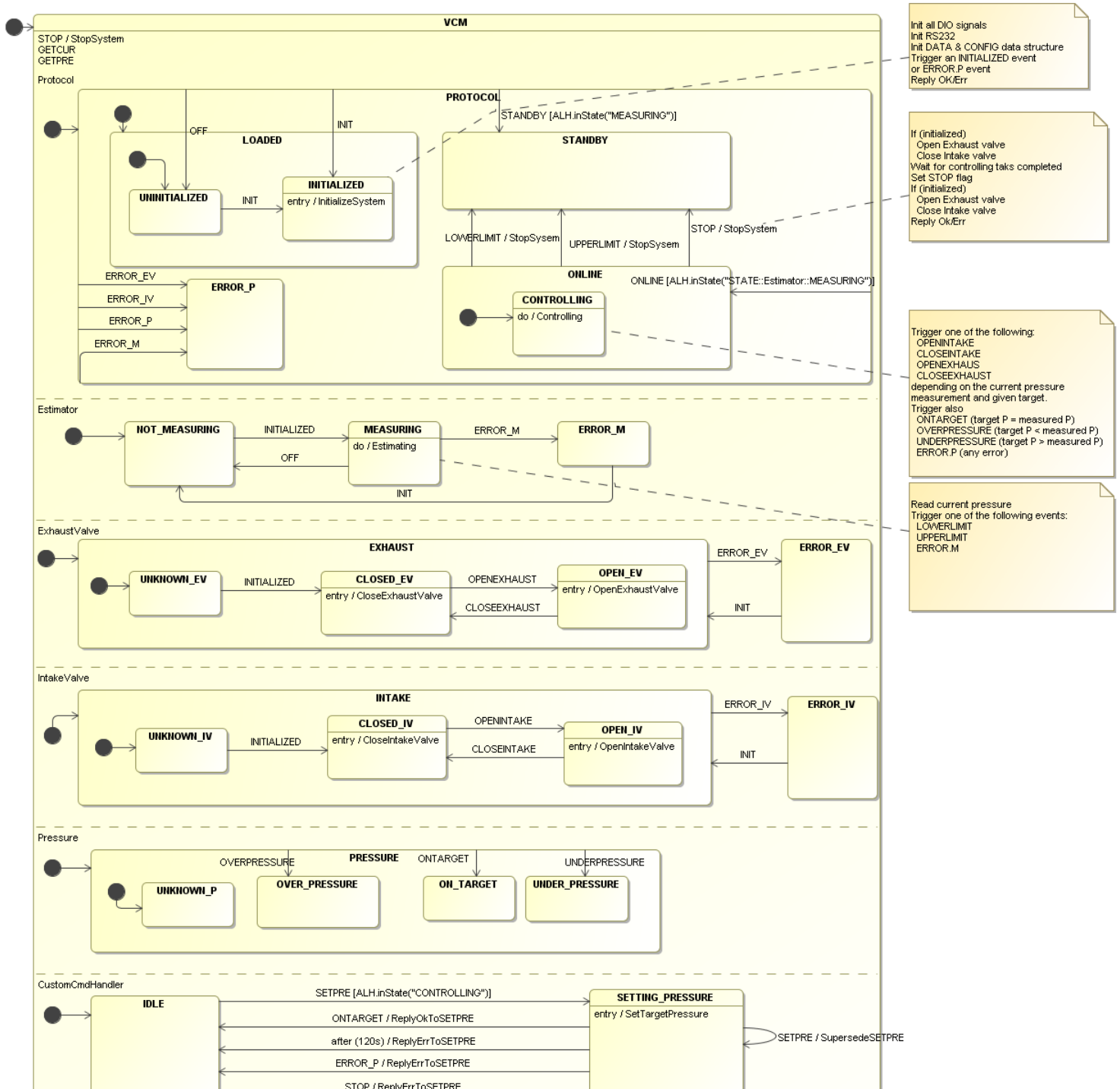
Some Model Checking Tools

Tool	Temporal Logic Expression	Developed by	Domain	Input Language
SPIN	Linear Temporal Logic (Pnueli and Manna)	G. J. Holzmann (NASA/JPL)	Distributed systems	PROMELA (textual)
SMV / NuSMV	Computation Tree Logic (Clark and Emerson)	CMU, University of Genova & Trento, …	Synchronous systems (HW)	SMV (textual)
UPPAAL	(Invariant and reachability properties)	Aalborg (Denmark) & Uppsala (Sweden) Universities	Real-time systems	Custom state machine
Java PathFinder	(Assertions)	Peter Mehlitz, et all (NASA Ames Research Center)	Java multi-threaded applications	Java

Variable Curvature Mirror



Variable Curvature Mirror Statecharts Model



COMODO & Java PathFinder

- Use COMODO to transform the model into Java code compliant with Java Pathfinder Statecharts plug-in
- Add assertions (CONTROLLING && MEASURING)
- Run the generated Java code + JPF jars
- Wait …
- Read the error trace and identify the sequence of events
- (optional) Re-play the sequence of events with a simulator
- (optional) Add test case

VCM Results

Java Pathfinder Error Trace

```
========================================================== error #1
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
java.lang.AssertionError: Controlling without measuring!
        at vcm$S_STATE$S_PROTOCOL$S_ONLINE$S_CONTROLLING.doAction(vcm.java:71)
        at gov.nasa.jpf.sc.StateMachine.executeDoAction(gov.nasa.jpf.jvm.JPF_gov_nasa_jpf_sc_StateMachine)
        at gov.nasa.jpf.sc.StateMachine.step(gov/nasa/jpf/sc/StateMachine.java:242)
        at gov.nasa.jpf.sc.StateMachine.run(gov/nasa/jpf/sc/StateMachine.java:346)
        at gov.nasa.jpf.sc.StateMachine.main(gov/nasa/jpf/sc/StateMachine.java:167)

========================================================== choice trace #1
  0: sc.SCEventFromSet[id="getEnablingEvent",>INIT(),STANDBY(),OFF(),ERROR_IV(),ERROR_EV(),ERROR_M(),ERROR_P(),ONLINE(),INITIALIZED(),
OVERPRESSURE(),UNDERPRESSURE(),ONTARGET(),SETPRE(),SETCUR()]
  1: sc.SCEventFromSet[id="getEnablingEvent",>STANDBY(),OFF(),ERROR_IV(),ERROR_EV(),ERROR_M(),ERROR_P(),INIT(),ONLINE(),INITIALIZED(),
OVERPRESSURE(),UNDERPRESSURE(),ONTARGET(),SETPRE(),SETCUR()]
  2: sc.SCEventFromSet[id="getEnablingEvent",STANDBY(),>OFF(),ERROR_IV(),ERROR_EV(),ERROR_M(),ERROR_P(),INIT(),ONLINE(),INITIALIZED(),
OVERPRESSURE(),UNDERPRESSURE(),ONTARGET(),SETPRE(),SETCUR()]
 ...
```

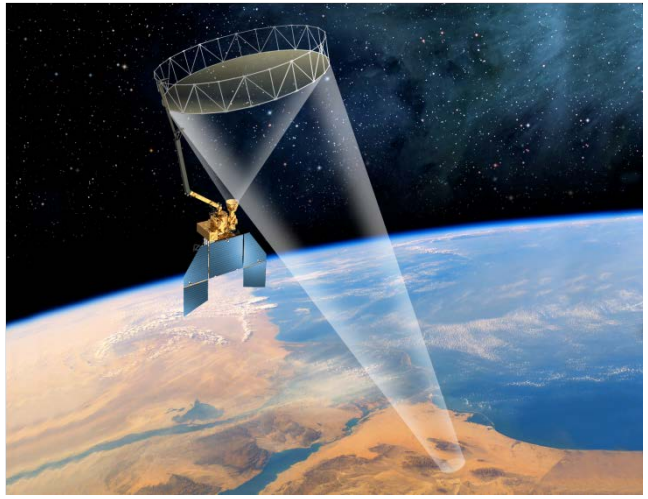
Sequence of events:

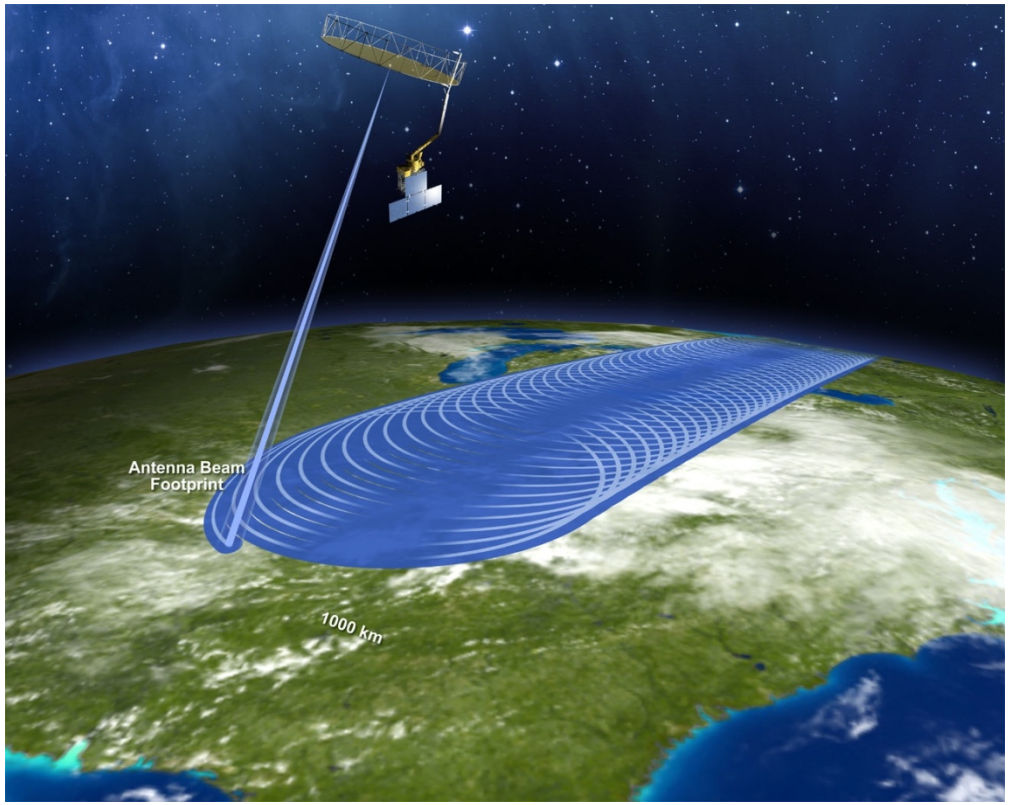
INIT, STANDBY, OFF, ERROR_IV, ERROR_EV, INIT, ERROR_EV, ONLINE

Active state configuration:

Protocol region: STATE/PROTOCOL/ONLINE/**CONTROLLING**
Estimator region: STATE/**NOT_MEASURING**
ExhaustValve region: STATE/ERROR_EV
IntakeValve region: STATE/INTAKE/UNKNOWN_IV
Pressure region: STATE/PRESSURE/UNKNOWN_P
CustomCmdHandler region: STATE/IDLE

NASA - Soil Moisture Active Passive





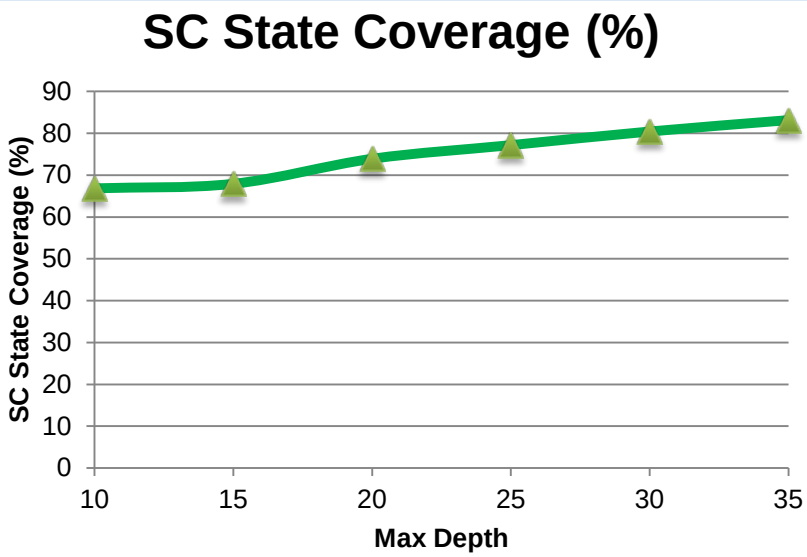
Global mapping of soil moisture and freeze/thaw state

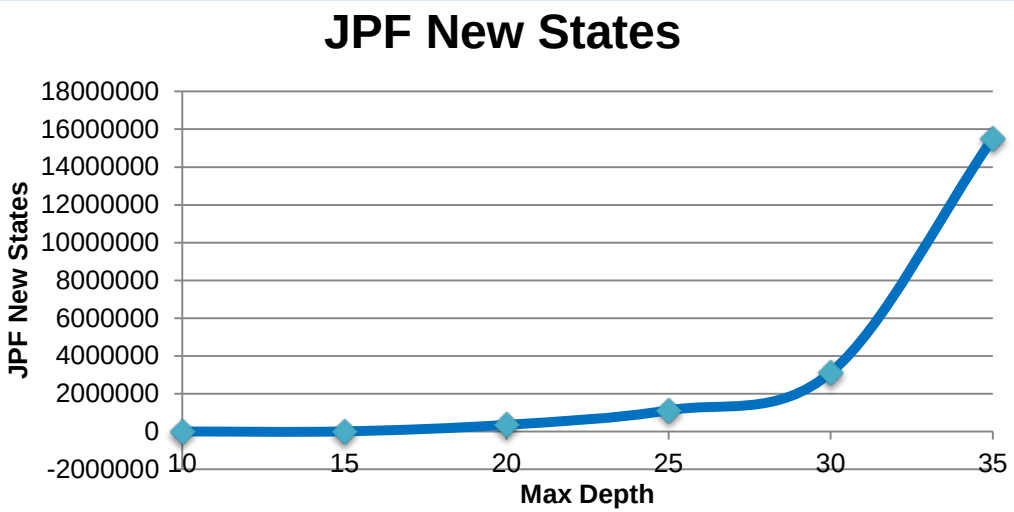
NASA – SMAP SysML Model

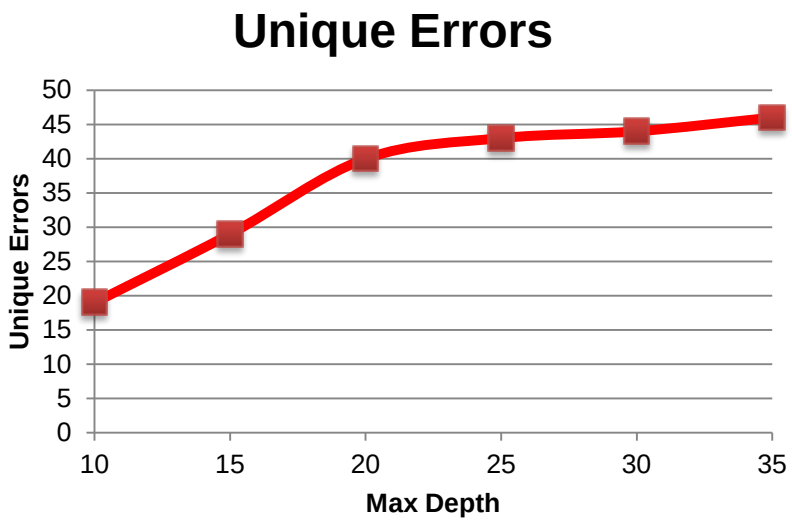
- Model of SMAP Fault Protection logical design in terms of structure and behavior.
- Behavior is modeled as network of collaborating Statecharts.

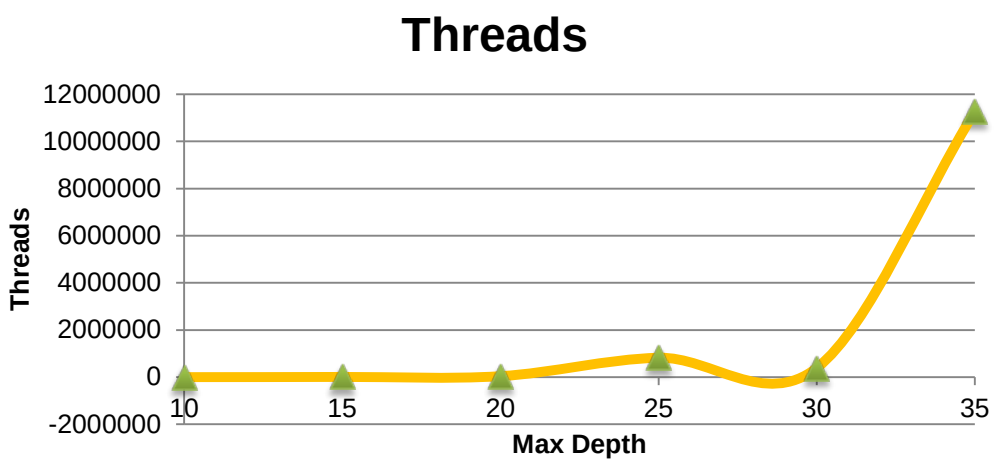
SysML Element	Total
State Machines	8
States	184
Orthogonal Regions	68
Transitions	307
Guards	181
Signals	134
Actions/Activities	173

NASA – SMAP Design Validation







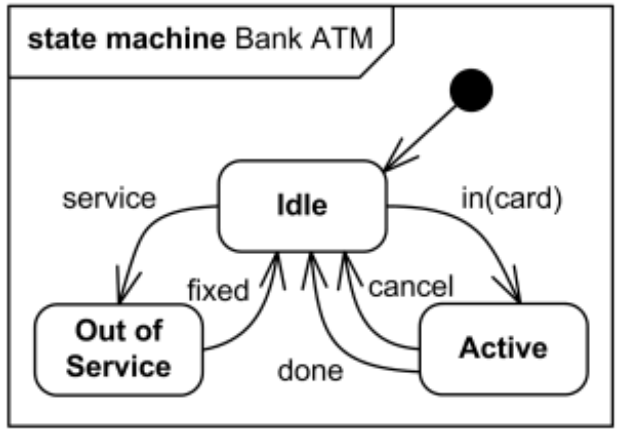


Model Transformations

Platform Independent Models

Model-to-Text Unidirectional Transformation
(hard-coded mapping & assumptions)

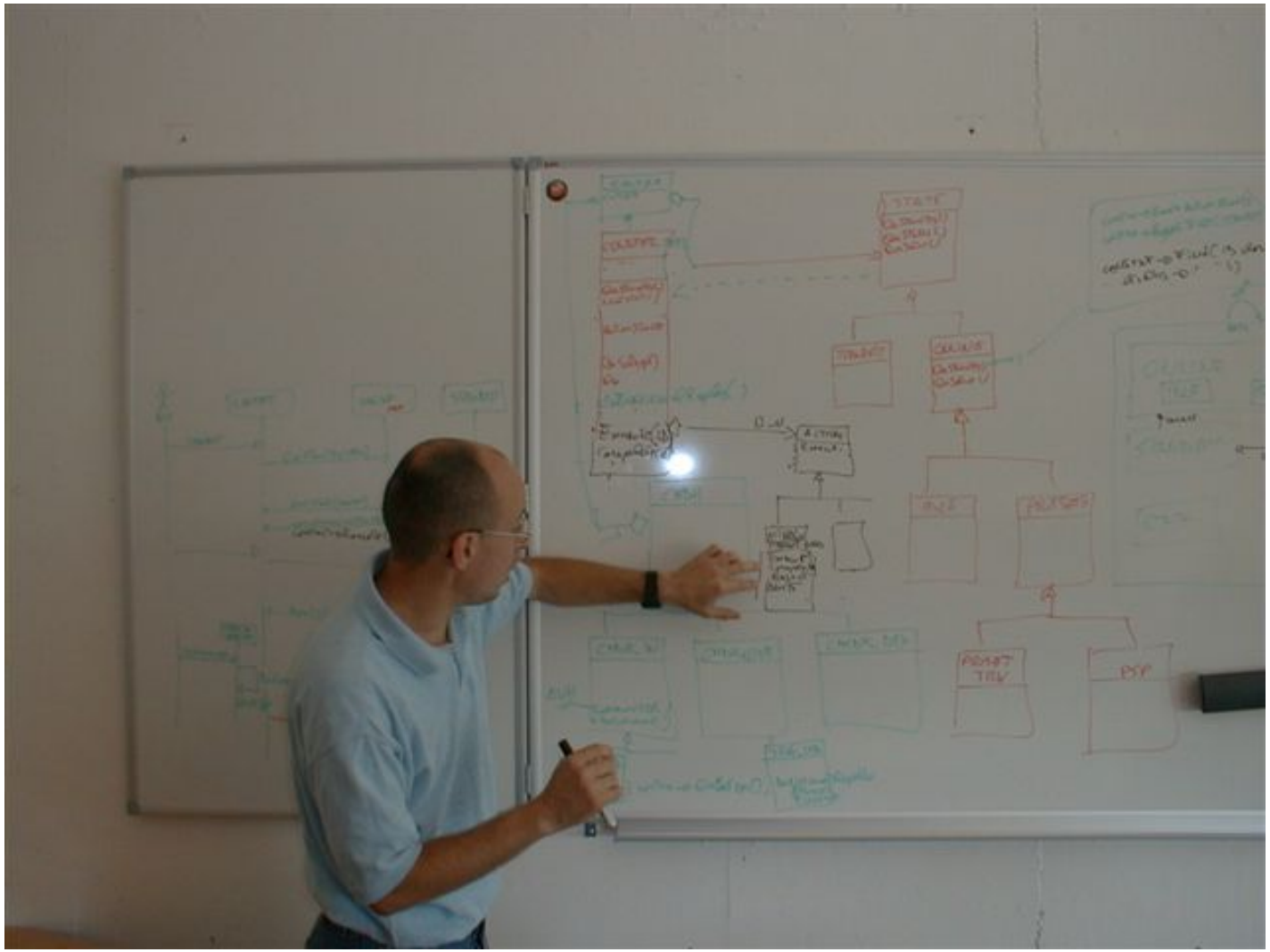
Platform Specific Artefacts



Design,
ICDs,
Code,
Prototypes,
Analysis

Target Platform Configuration

Workstation Software Framework



- Started in 2004
- Purpose: create 15 control SW applications
- Transforms Statecharts models into C++ applications
- Specific for the VLT platform
- Custom textual syntax and semantic
- Compiled State Machine

WSF based 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

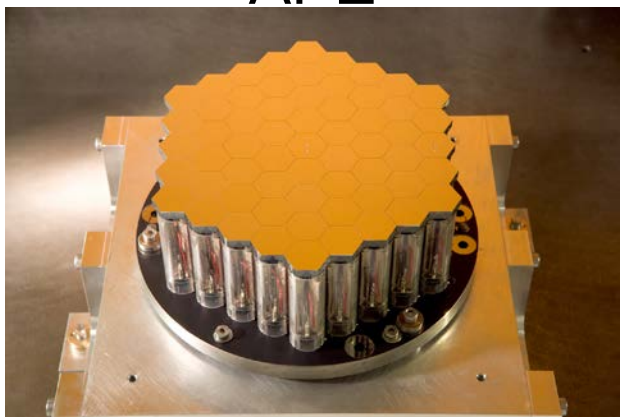
New Components: 53

(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

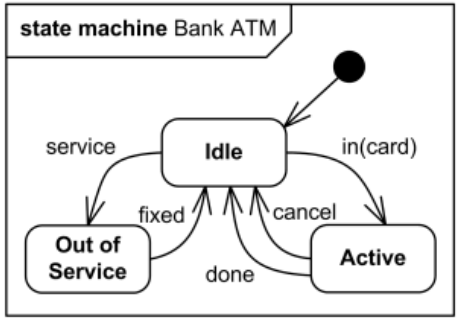
Workstation Software Framework pros & cons

- Efficient
- Support textual and graphical notation
- Consistency between design and implementation

- Custom Semantic
- Platform Specific
- Compiled Statecharts
- Vendor specific plugin

COMODO

Model (cmdoProfile)



Any UML Tool supporting EMF XMI (MagicDraw, Papyrus, …)

Model (EMF XMI)

ESO COMODO (based on Xpand/Xtend)

Target Platform Configuration

Platform Specific Artefacts (build, code, tests)

SCXML

COMODO based Projects

- EELT TCS Local Supervisor Prototype
- VLTI Adaptive Optics subsystem for Auxiliary Telescope
- VLTI new Variable Curvature Mirrors
- VLT Instruments for the VLT Adaptive Optics Facility
- VLT Instrument Technical Detector Control SW
- NASA/JPL SMAP

Transformation Ownership

Write your own / have control over the model transformations to:

- Modify metamodel/generated artifacts to fit the project needs.
- Easily port artifacts to different version of the target platform / different projects.
- Avoid vendor lock-in.

Transformation Efficiency

Always measure the Return On Investment

Key factors:

- Abstraction to reduce complexity and enable formal analysis
- Multiplicity of generated artefacts
- Reuse of the transformation
- Maintenance of Tool Chain vs Artefacts

ROI:

- Move from hand-crafted approach to industrial production -> homogenous artefacts
- Reusability of models
- Consistency between artefacts (design, ICDs, SW, …)
- Porting of generated artefacts

Questions?

BACKUP SLIDES

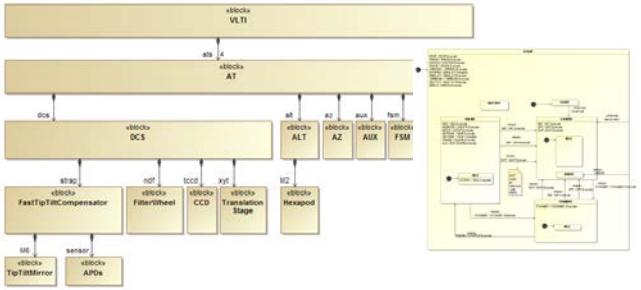
Lessons Learned - 1

Standard Modeling Languages

and

Tool Interoperability

Systems, Models, and Meta-models

M3	Meta Object Facility (MOF)	**Meta-Meta-Model** **represents** the syntax of a basic language. It **conforms to** itself.
M2	OMG SYSTEMS MODELING LANGUAGE™	**Meta-Model** **represents** the syntax of a modelling language and **conforms** to a meta-meta-model.
M1		**Model** **represents** certain aspects of a System and **conforms** to a meta-model.
M0		**System** to build.

Eclipse Modeling Framework (EMF)

Ecore – “Eclipse” Meta Object Facility

- Library to create/manipulate meta-models/models
- **EMF XMI** serialization for persistence
- Reflective **API** for manipulating EMF objects

MDT/UML2 – Ecore implementation of the UML/SysML meta-model.

Some of the tools based on Ecore:

- **Papyrus** to create UML/SysML models
- **Xpand/Xtend** for model transformation
- **Xtext** to create DSL languages

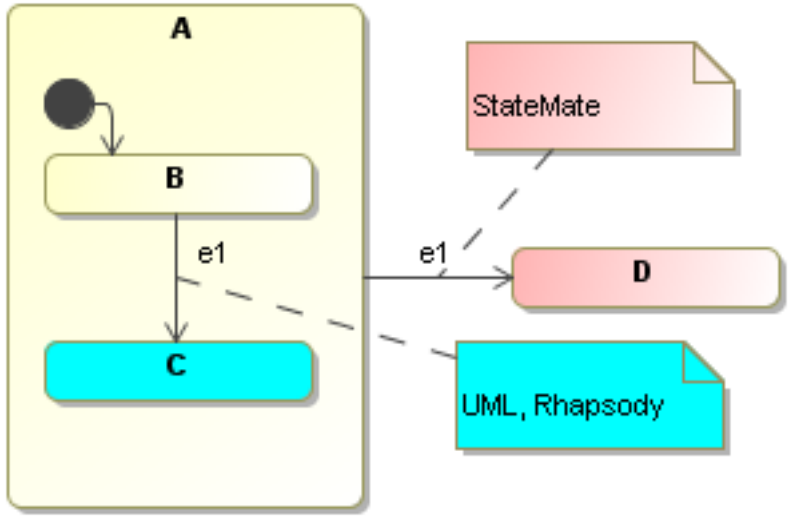
Modeling Languages

Language = Syntax + Semantic

Without precise semantic definition everybody can assign their own meaning to the model.

Standard Semantic

Crane & Dingel
“Not all models are created equal.”



- StateCharts XML (SCXML) - W3C - 2015
- Precise Semantic for State Machines (PSSM) – OMG – started WG in ~2016

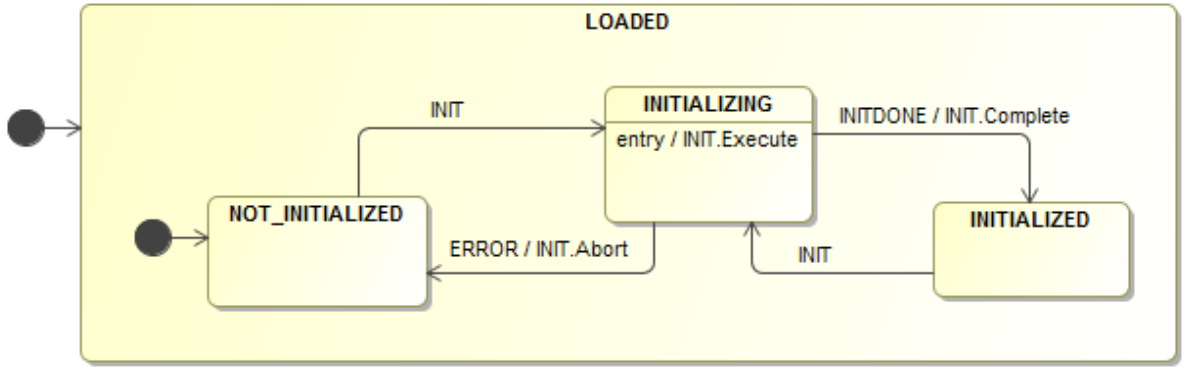
Lessons Learned - 2

(Graphical) General Purpose Languages

vs

(Textual) Domain Specific Languages

Graphical or Textual Notation




```
<state id="LOADED">
  <initial>
    <transition target="LOADED::NOT_INITIALIZED"/>
  </initial>

  <state id="LOADED::INITIALIZING">
    <onentry>
      <customActionDomain:INIT.Execute name="INIT.Execute"/>
    </onentry>
    <transition event="INITDONE" target="LOADED::INITIALIZED">
      <customActionDomain:INIT.Complete name="INIT.Complete"/>
    </transition>
    <transition event="ERROR" target="LOADED::NOT_INITIALIZED">
      <customActionDomain:INIT.Abort name="INIT.Abort"/>
    </transition>
  </state>
  ...
```

General Purpose Language (GPL)
Domain Specific Language (DSL)

- SysML/UML: customization using profiles (stereotypes)
- SysML/UML: imprecise semantic and difficult to validate
- Reuse infrastructure

- Usually based on GPL
- Lean small languages
- Require infrastructure (editors, debugger, etc.), and maintenance

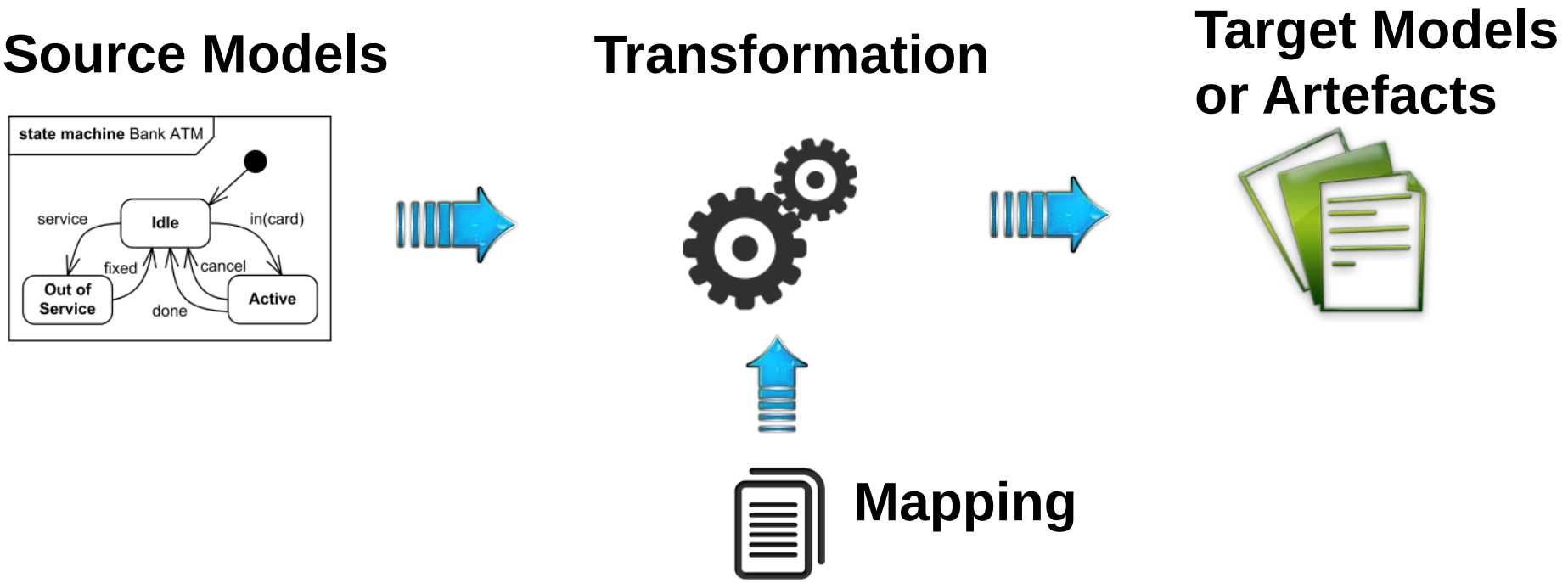
Both approaches requires execution/transformation

Lessons Learned - 3

Model Transformations

Ownership and Efficiency

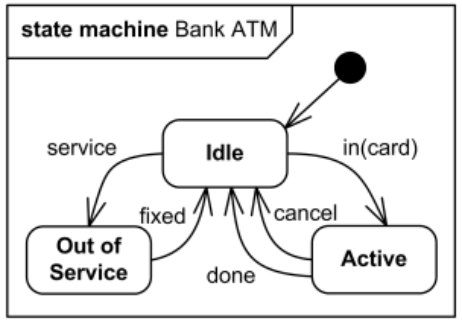
Model Transformations



- Model-To-Model (M2M) or Model-To-Text (M2T)
- Endogenous (source == target language) or Exogenous
- Unidirectional or Bidirectional

Model Driven Architecture (MDA)

Platform Independent Models (PIM)



Model-to-Model (exogenous) Transformation

Platform Specific Models (PSM)

Model-to-Text Transformation

Platform Specific Artefacts

M2M Mapping

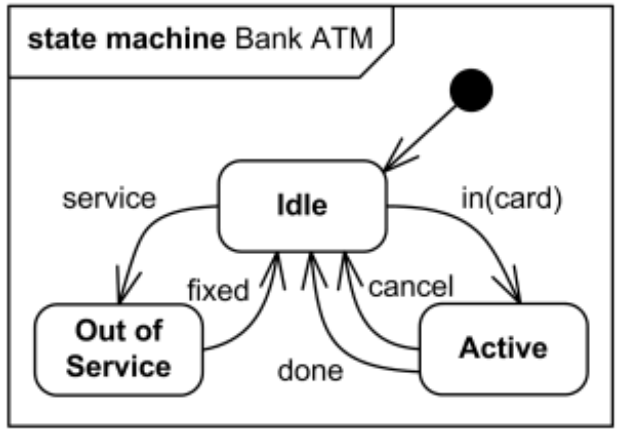
M2T Mapping

Simplified Approach

Platform Independent Models (PIM)

Model-to-Text Unidirectional Transformation
(hard-coded mapping & assumptions)

Platform Specific Artefacts



Target Platform Configuration

ESO Model Transformations Tools

Year	2000	2004		2009	
Tool	Local Software Framework (LSF)	**Workstation Software Framework (WSF)**	APDMGen	**COMODO**	**Model Based Document Generator**
Domain	Local Control	Supervisory Control	Data Model	Supervisory Control, Model Checking	Documentation
Projects	ATCS	VLTI, PRIMA, APE, NGC	ALMA	EELT LSV prototype, NAOMI, AOF, TDCS	Some EELT documents