

MODEL-BASED SOFTWARE DEVELOPMENT LIFECYCLE



TEC-ED & TEC-SW Final Presentation Days - December 9th 2015

Joan Clua, Patricia López Martínez, Dominique Torette, Jorge Garrido

PROJECT OVERVIEW

Model-Based Methodology Review

- Define a methodology covering the entire SW lifecycle
- Work on ECSS to accommodate model-based projects (deliverables, phases, reviews...)

TASTE Extension

- Explore areas where lifecycle is not deeply covered by the toolset (requirements, timing analysis) and select suitable tools (**jUCMNav**, **RDALTE**, **Rapitime**)
- Integrate these tools in the toolset

Proof of Concept in a Mission

- Define specific mission (Flight Formation based).
- Implement the mission using the toolset integrated in the project according to methodology.
- Exercise point of view of SW developers, system engineers and reviewers to validate the methodology and the tools in a relevant domain.

CONSORTIUM

Project Coordination

- Indra

Model-Based Methodology Review

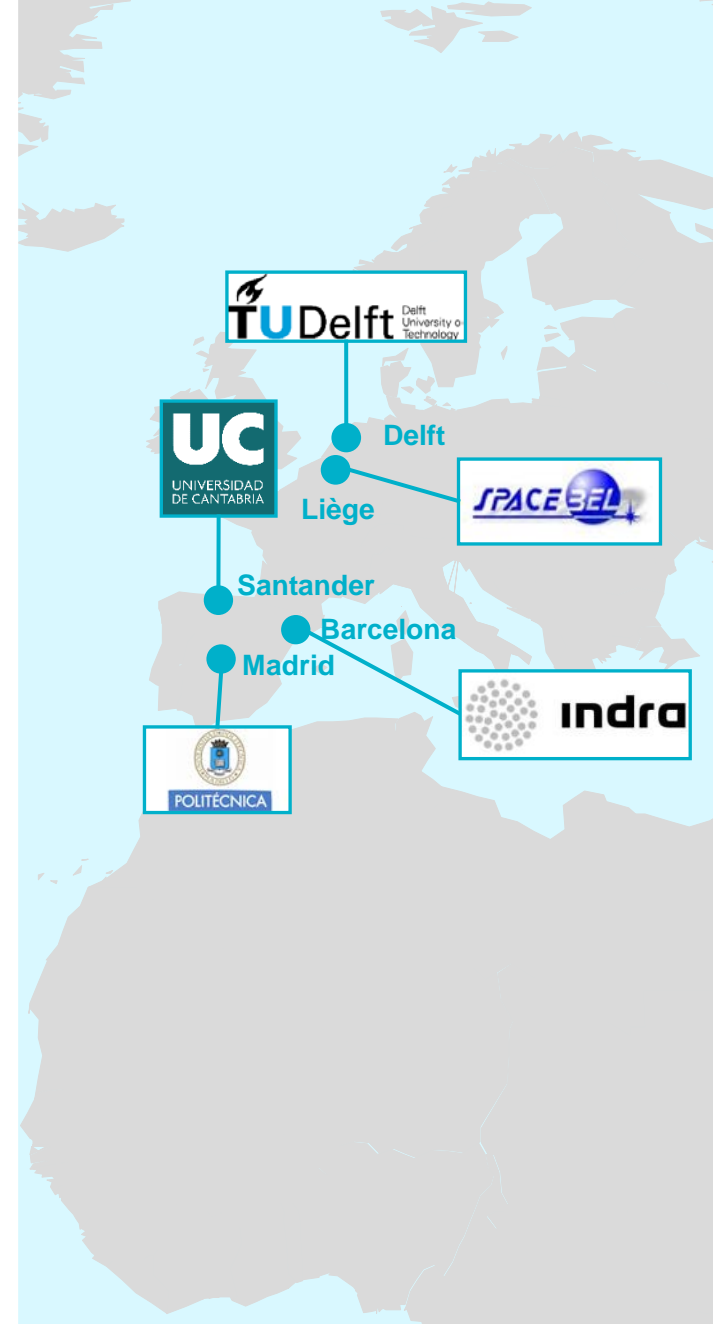
- Universidad de Cantabria (UC)
- Universidad Politécnica de Madrid (UPM)
- Spacebel

TASTE Extension

- Indra
- UC
- UPM

Proof of Concept in a Mission

- TU Delft (Mission Definition)
- Spacebel (Mission Implementation)



01 Model Based Methodology (P. López, UC)

02 Timing Analysis in TASTE (J. Garrido, UPM)

03 Mission Proof of concept (D. Torette, Spacebel)

04 Conclusions

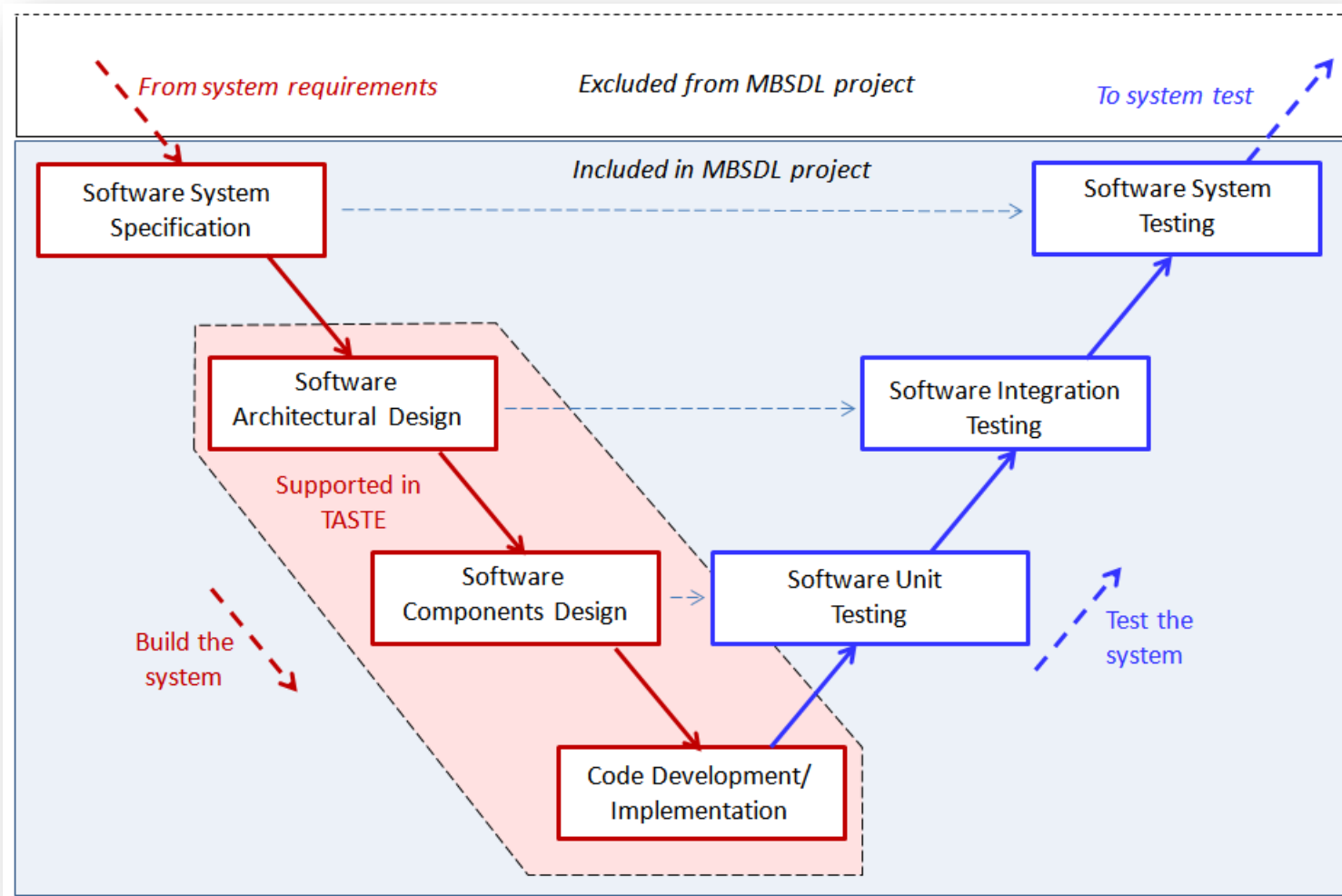
01 Model Based Methodology (P. López, UC)

02 Timing Analysis in TASTE (J. Garrido, UPM)

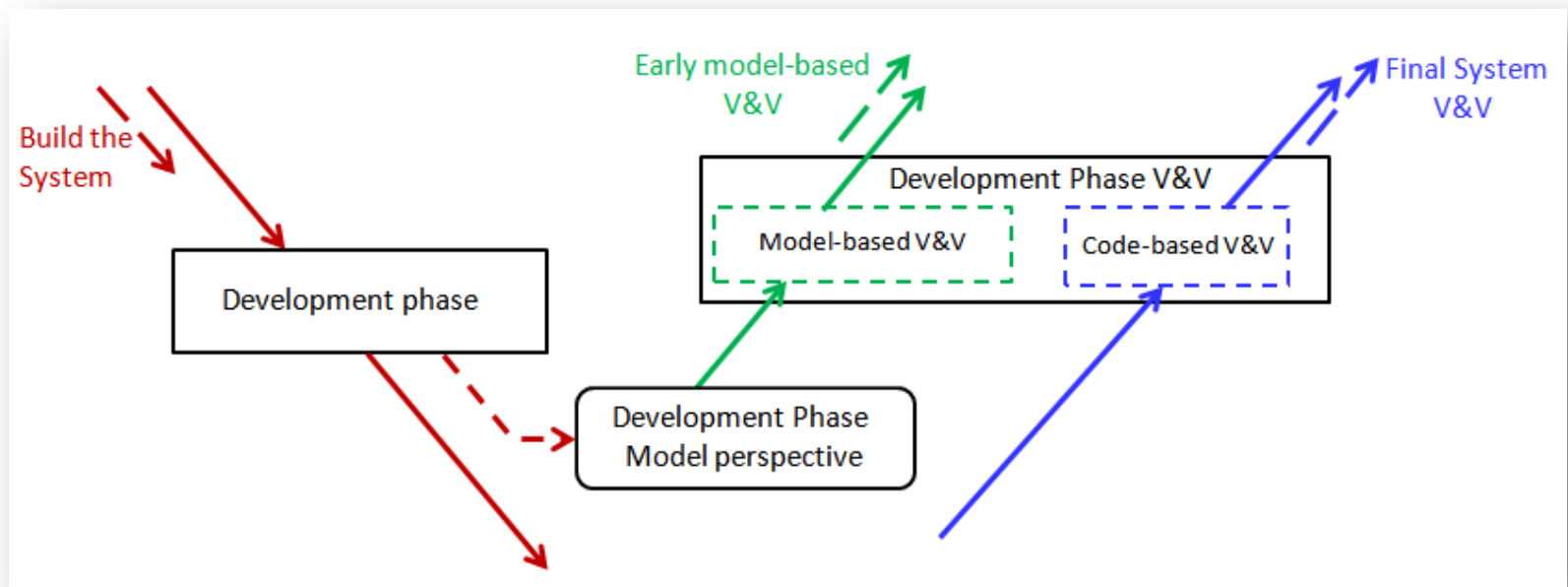
03 Mission Proof of concept (D. Torette, Spacebel)

04 Conclusions

MBSDL – STARTING POINT

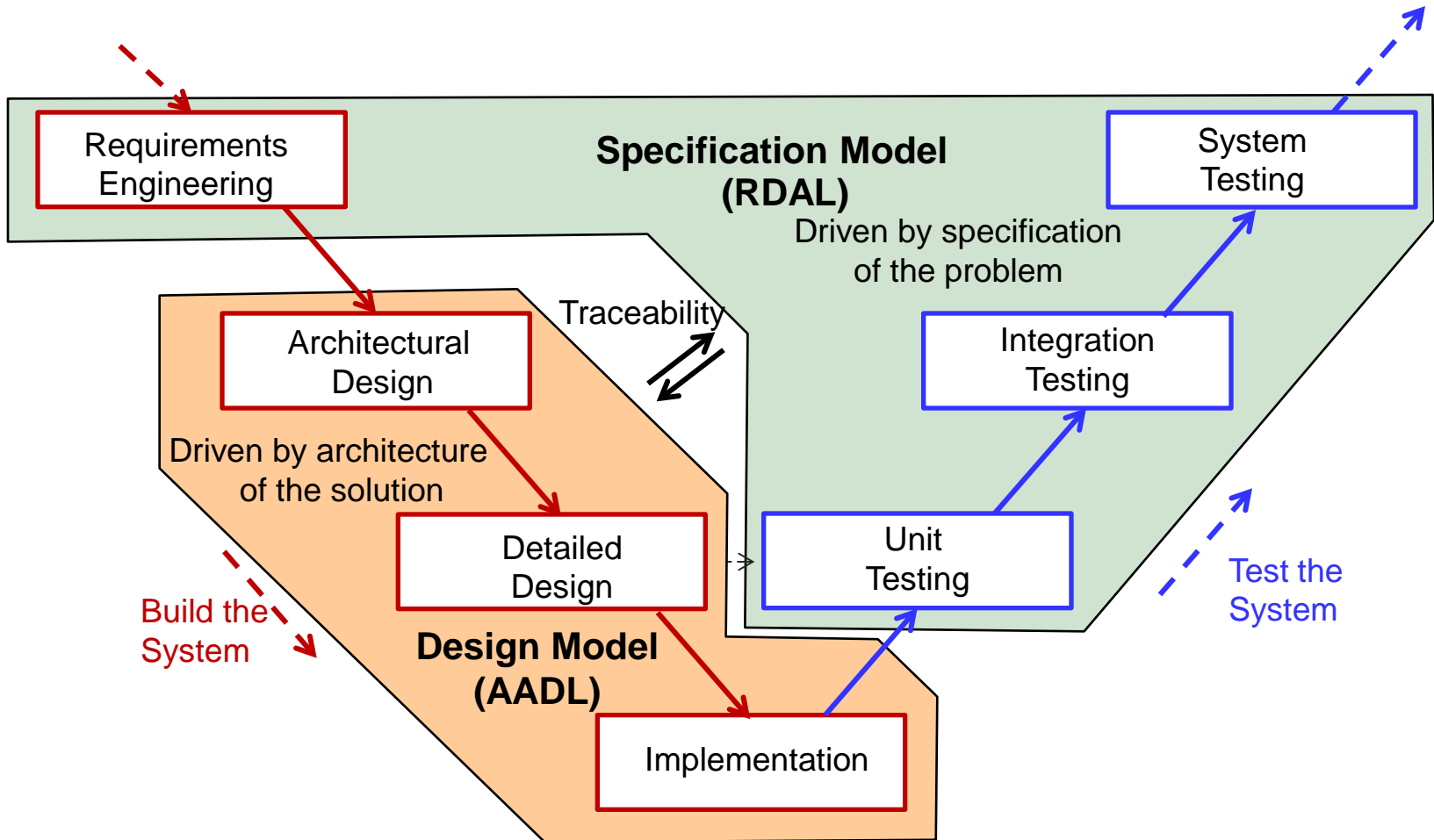


MBSDL – EARLY MODEL-BASED ANALYSIS



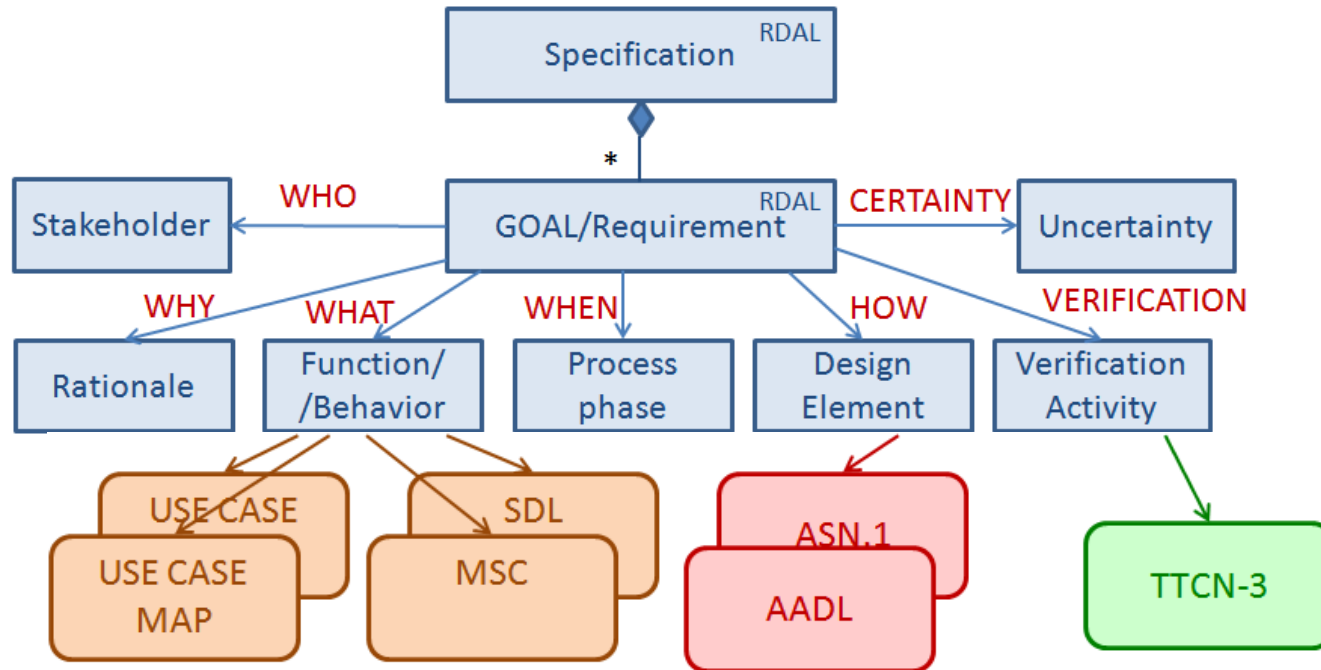
MBSDL DEFINITION

An architecture-centric and specification-centric lifecycle

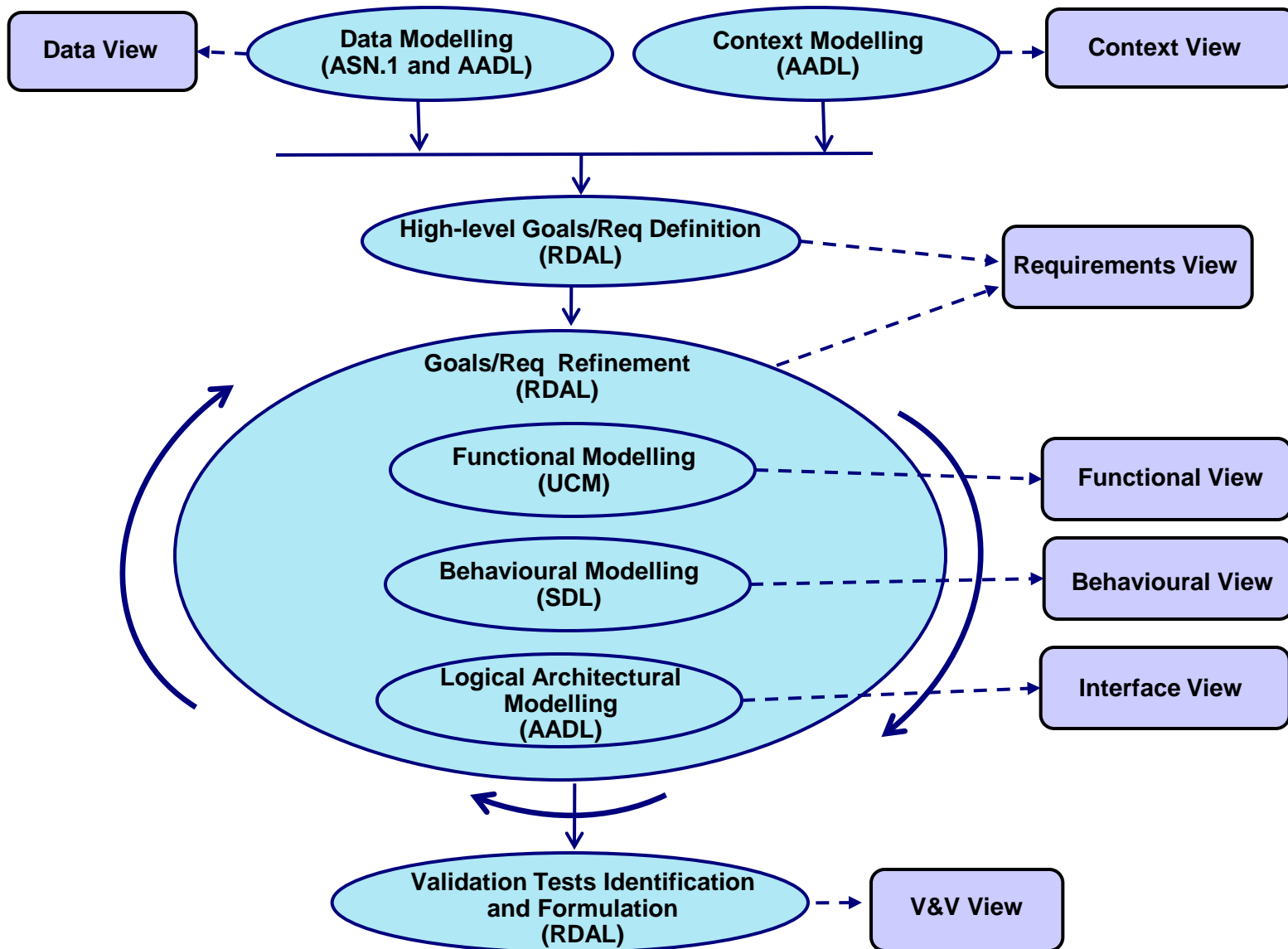


RDAL LANGUAGE

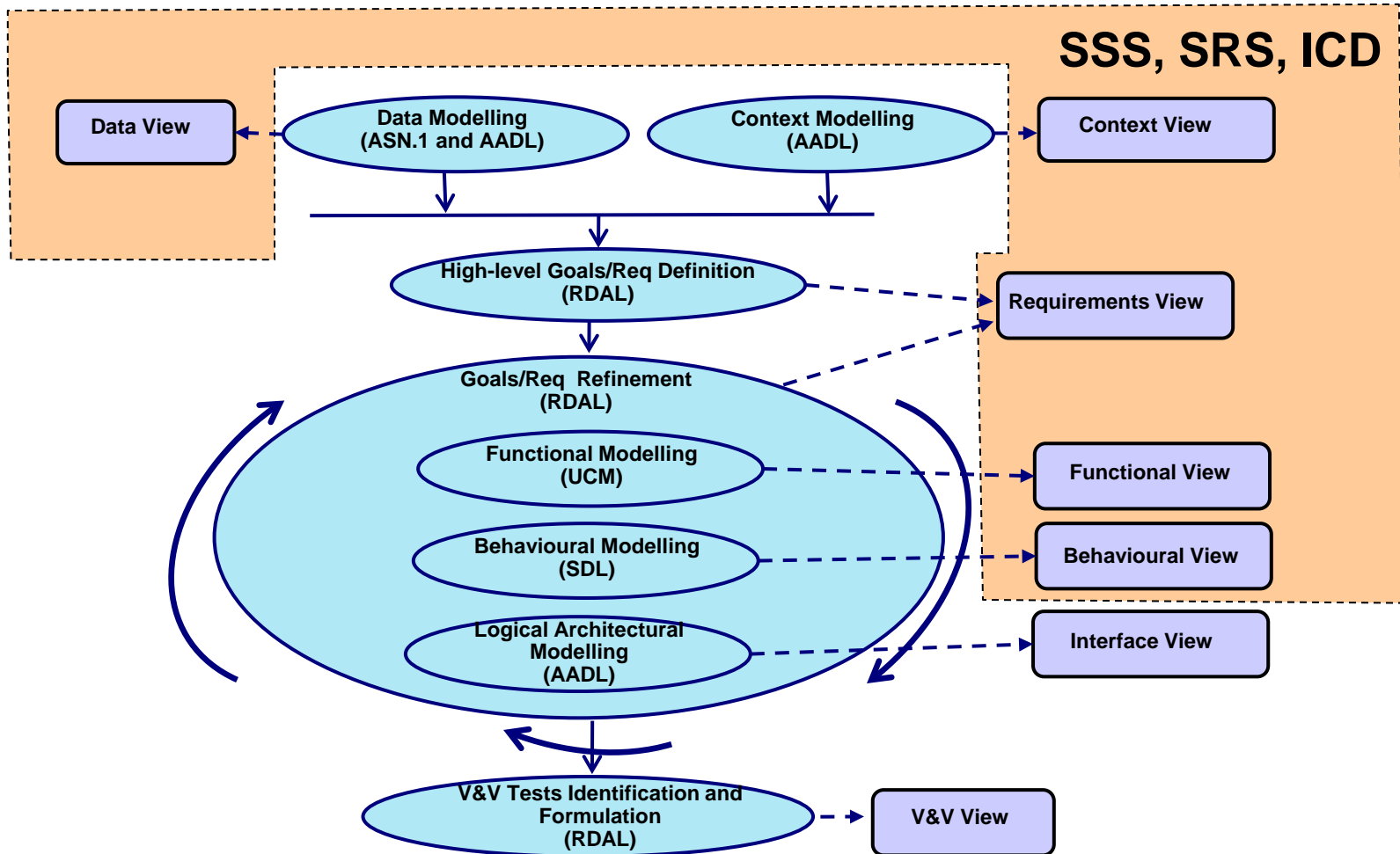
- RDAL: Requirements Definition and Analysis Annex
 - AADL Annex but independent from it
 - In process of standarization
 - Traceability to design elements, functional models, reactive models, V&V Actitivites, etc.



SOFTWARE REQUIREMENTS SPECIFICATION IN MBSDL



SOFTWARE REQUIREMENTS SPECIFICATION IN MBSDL



TOOL SELECTION

MBSDL Phase	Tools
Requirements Specification	RDALTE (RDAL Tool) jUCM (Use Case Maps Tool) + OpenGeode (SDL Tool) AADL + ASN.1
Software Architectural Design	AADL + ASN.1 OpenGeode (SDL) MAST/Cheedar (Schedulability Analysis)
Software Components Design	AADL + ASN.1 OpenGeode + Simulink/SCADE/RTDS MAST/Cheedar
Implementation	TASTE Code Generators
Software Testing	Rapita AADL + MAST/Cheedar MSC Test Tracer (TASTE)

01 Model Based Methodology (P. López, UC)

02 Timing Analysis in TASTE (J. Garrido, UPM)

03 Mission Proof of concept (D. Torette, Spacebel)

04 Conclusions

WORK OVERVIEW

- Objective: provide a working integration of Rapitime
 - Structural analysis
 - Execution time analysis

- UPM and Rapitime Systems collaboration
 - Rapita Systems:
 - Provision of tools
 - Initial integration version
 - UPM:
 - Providing previous integration as starting point
 - Support on TASTE internals
 - Versions iteration with Rapita
 - Validation of final integration

RAPITIME OVERVIEW

On-target timing analysis tool

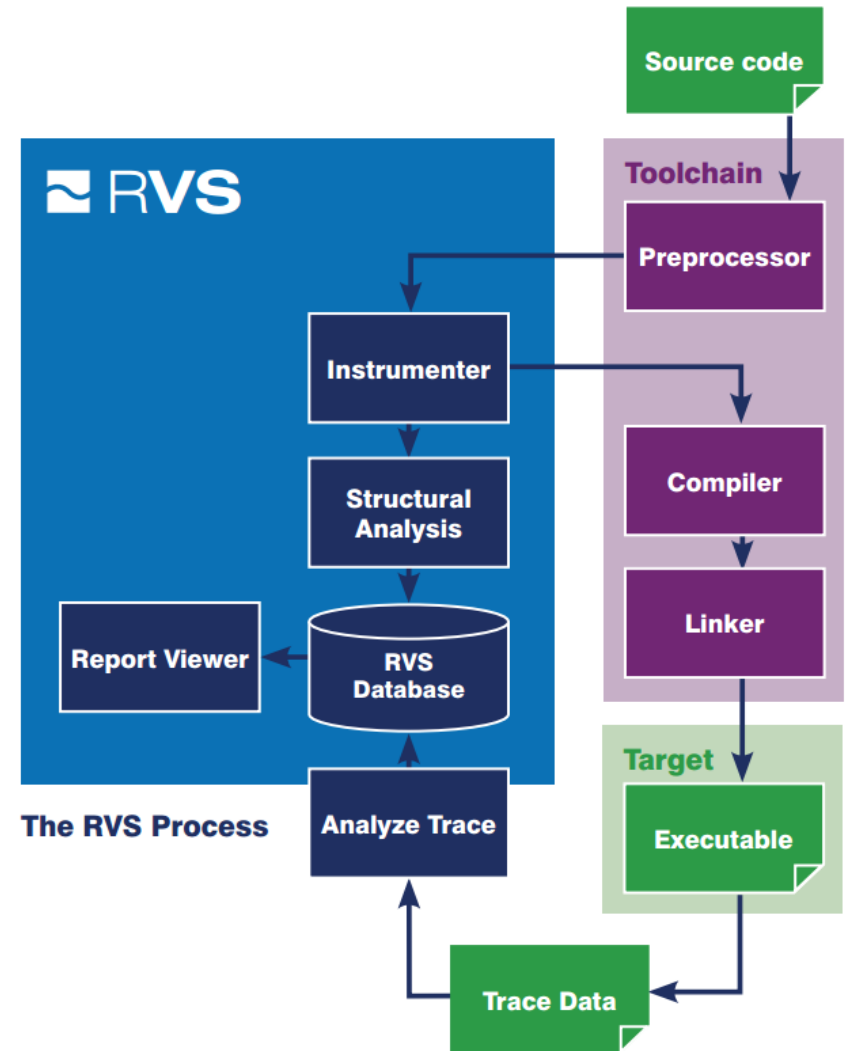
Based on:

- Structural analysis of the code (compiler and user aided)
- Automated instrumentation points
- Execution traces

Provides:

- Automated timing analysis framework
- Execution paths analysis
- Timing behavior of the system (including WCET estimation)
- Graphical and textual results
- Other analysis offered in Rapita Verification Suite (RVS)

Rapitime process



INTEGRATION OVERVIEW

- **Integration based on previous version by UPM**
- **Aimed to follow TASTE methodology:**
 - Command-line based process
 - Python and bash scripts
 - Text-files results
 - Automated system analysis based on AADL and Ocarina generated files
 - Integration within assert-builder-ocarina.py with new parameter –instrument
- **Two-steps integration:**
 - First build generates system as usual
 - Second build generates instrumented system at an auxiliary directory

INTEGRATION RESULTS

- Automated timing analysis for TASTE projects
- Textual and graphical results
 - Stored in a SQL database
 - Textual output customizable

```
== project.vt_ground_result_wrappers.artificial_result.rvd  
vt_ground_result_wrappers.artificial_result: 4077 cycles
```

```
== project.vt_myfunction_pulse_wrappers.artificial_pulse.rvd  
vt_myfunction_pulse_wrappers.artificial_pulse: 73745 cycles
```

```
== project.vt_myfunction_start_processing_wrappers.artificial_start_processing  
vt_myfunction_start_processing_wrappers.artificial_start_processing: 2328 cycles
```

INTEGRATION RESULTS

RVS - /home/assert/rapita/mbsdl_test/demo/rvs_clone/binary/rvs/project.vt_myfunction_pulse_wrappers.artificial_pulse.rvd - RVS Report Viewer

File Edit Navigate Search Window Help

Report Navigator

- project.vt_myfunction_pulse_wrappers.artificial_pulse.rvd
- project.vt_myfunction_pulse_wrappers.artificial_pulse.rvd
- Bookmarks
- Source Files
- Functions
- Call Tree
 - vt_myfunction_pulse_wrappers.artificial_pulse-U
 - myfunction_wrappers.protected_myfunction_pulse-U
 - myfunction_wrappers.callinglist.push-U
 - myfunction_pulse-U
 - myfunction_PI_pulse-U
 - myfunction.runtransition-U
 - system.io.put_line-U
 - myfunction_RI_compute_gnc-U
 - system.io.put-U
 - system.io.put-U
 - system.io.put-U
 - system.io.put_line-U
 - myfunction_RI_result-U
 - myfunction_wrappers.callinglist.pop-U

project.vt_myfunction_pulse_wrappers.artificial_pulse.rvd

Contribution Comparison for Function: gnc_wrappers.compute_gnc

Up to Report level Up to File level Source code Show Context cycles (c) Find element

Contribution Summary

Self Contribution Time

Function	Avg-CT	H-CT	W-CT	#Tests
Name	A-SelfCT*	H-SelfCT*	W-SelfCT*	
.compute_gnc	(x0.527) 3,543.6	(x1) 7,218	(x1) 15,219	10

Calls (3)

Overall Contribution Time

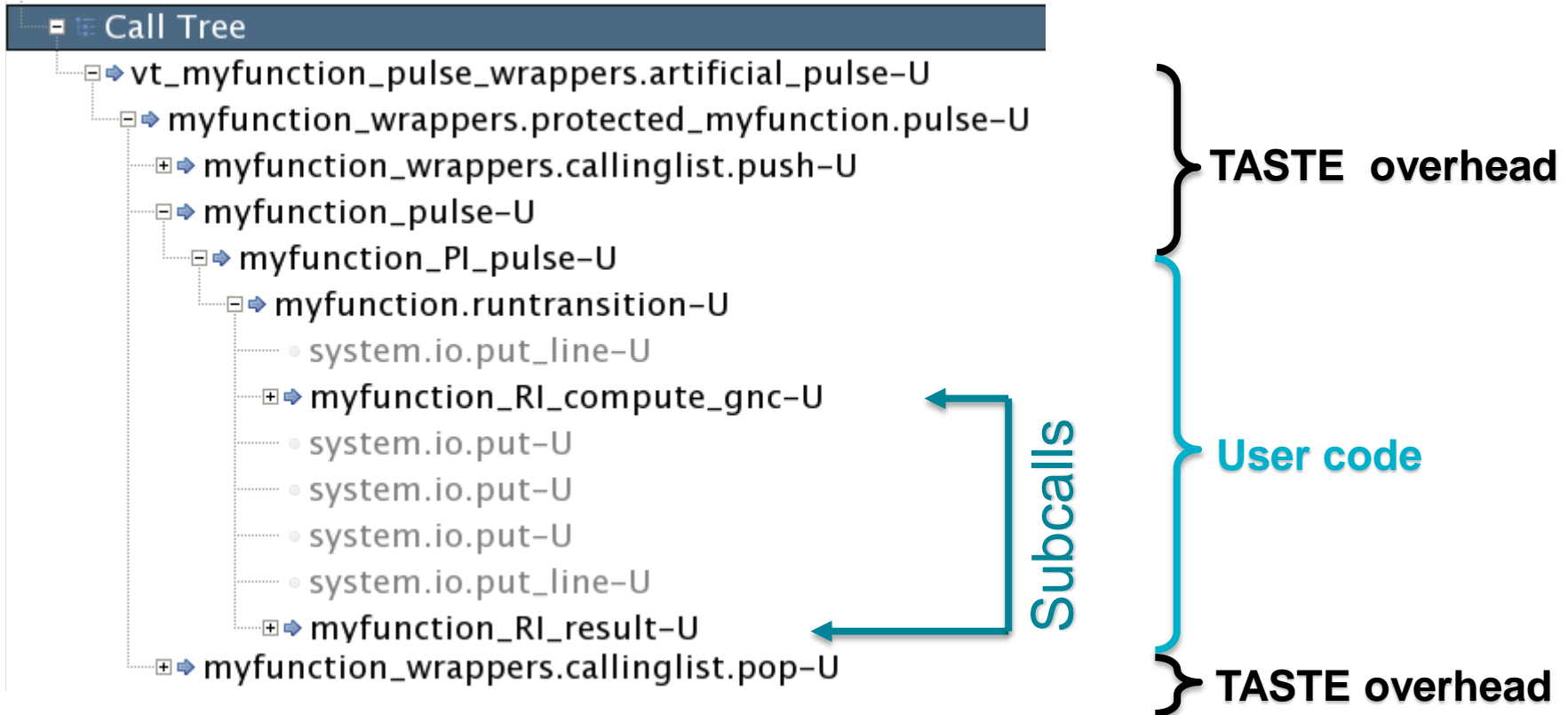
Call	Avg-CT	H-CT	W-CT	#Tests
Target	A-OverCT*	H-OverCT*	W-OverCT*	
callinglist.push	(x0.527) 386.1	(x1) 828	(x1) 133	10

Static Tests Min Average High WM Max WCET CT Comparison ET Comparison

Properties

assert@assertv... [rapita] RVS - /home/ass... 06:41

INTEGRATION RESULTS



INTEGRATION RESULTS

project.vt_myfunction_pulse_wrappers.artificial_pulse.rvd

Static Analysis for Report: project.vt_myfunction_pulse_wrappers.artificial_pulse.rvd

Show: Folders Files Functions

Find element

Static Summary

- 38 functions
- 183 lpoints
- 209 Blocks
- 197 lines of code

Code Size

Lines of Code (Self)

Function	#LOC-Self	#LOC-Self%	#LOC-Over	#LOC-Over%	#Blocks	#Loops	#Conditionals	#Calls	#Callers	Instr. Y/N
PrintASN1MyReal	6	3.0%	6	3.0%	6	0	0	4	2	✓ Yes
asn1ScMyReal_Initialize	5	2.5%	5	2.5%	2	0	0	0	1	✓ Yes
getTimeInMilliseconds	5	2.5%	5	2.5%	3	0	0	1	2	✓ Yes
gnc_PL_compute_gnc	3	1.5%	3	1.5%	1	0	0	0	1	✓ Yes
gnc_compute_gnc	10	5.0%	18	9.1%	15	0	1	6	1	✓ Yes
.callinglist.pop	9	4.5%	9	4.5%	13	1	4	1	1	✓ Yes
.callinglist.push	8	4.0%	8	4.0%	11	1	3	1	1	✓ Yes

Overview Static Tests Min Average High WM Max WCET CT Comparison ET Comparison

INTEGRATION RESULTS

project.vt_myfunction_pulse_wrappers.artificial_pulse.rvd

Block Tests for Context: vm_myfunction_compute_gnc-U

Up to Report level | Up to Function level | Source code

Find element

Context Summary

Context	Tested Blocks	Tested Calls	#Tests
Name	T-Block*	T-Call*	
vm_myfunction_compute_gnc-U	5 / 5	2 / 2	14

Calls (2)

Call	T-Call	#Tests
Target		
.get_top_value	✓ T	14
.compute_gnc	✓ T	14

Blocks (5)

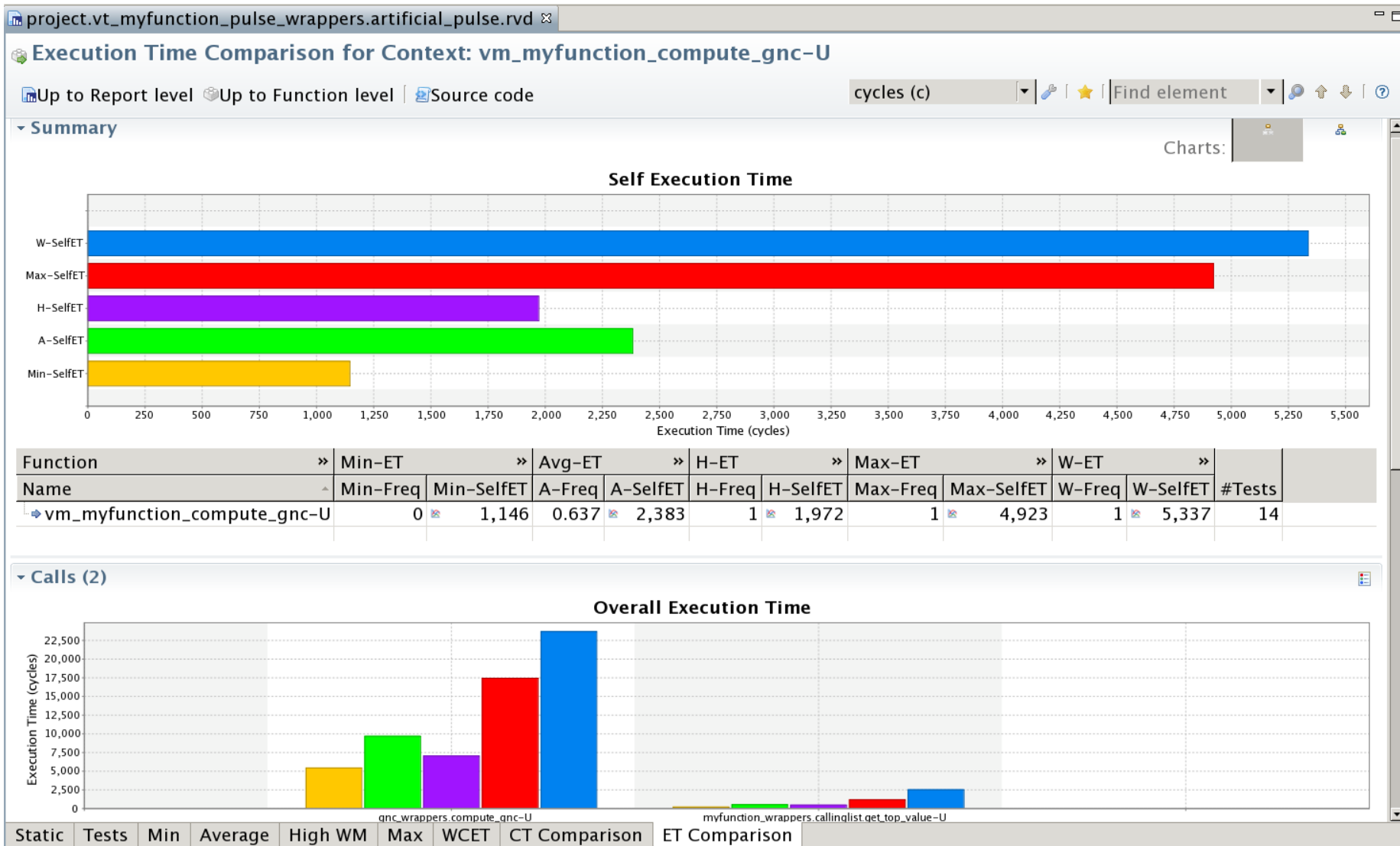
Call Tree

The hierarchy of calls to this element

Call	Tested Blocks	#Tests
Name	T-Block*	
.artificial_pulse-U	3 / 3	22
.protected_myfunction_pulse-U	7 / 7	22
.myfunction_pulse-U	4 / 4	22
.myfunction_PI_pulse-U	6 / 7	22
.myfunction_runtransition-U	12 / 20	22
.myfunction_RI_compute_gnc-U	15 / 28	14
.vm_myfunction_compute_gnc-U	5 / 5	14
.myfunction_runtransition-U	12 / 20	22
.myfunction_RI_compute_gnc-U	15 / 28	14
.vm_myfunction_compute_gnc-U	5 / 5	14

Static | Tests | Min | Average | High WM | Max | WCET | CT Comparison | ET Comparison

INTEGRATION RESULTS



01 Model Based Methodology (P. López, UC)

02 Timing Analysis in TASTE (J. Garrido, UPM)

03 Mission Proof of concept (D. Torette, Spacebel)

04 Conclusions

USE CASE

- Software Requirement Specification for a Formation Flying Mission
 - GNC FF Algorithms
 - FF Operational modes
 - Distribution of the responsibilities across the formation
 - Routing of PUS TM/TC
 - Ground and Inter Satellites Link communication interfaces
- Top inputs : High Level Specifications from TUDelft
 - Introduction to AOCS/GNC functionalities of FF Satellites
 - Mission Requirement Document
 - System Requirement Document
 - Operational Requirement Document
- Additional inputs : In-house ongoing PROBA-3 FF project feedbacks

OUTPUTS

- Models
 - Specification and Documentation Generation
 - Use Case Maps : URN/UCM
 - RDAL : RDAL
 - Design and Code Generation
 - TASTE (Interface, Deployment) : AADL
 - TASTE (Data model) : ASN.1
 - TASTE/OpenGeode (Behavior, modes) : SDL
 - Traceability links
- ECSS-like documentation (Fully generated from models - template based)
 - SSS
 - SRS

USER REQUIREMENTS NOTATION (URN)

- User Requirements Notation (URN) is intended for
 - Elicitation and specification of requirements.
 - Discover and specify requirements for a proposed system
 - Analysis and validation of requirements.
 - Analyze such requirements for correctness and completeness
 - Execution of Scenarios and branches coverage
 - Simple formal language for Behaviors and Scenario expected results definition
 - Impact analysis of Implementation Trade-Off.
 - Explore impacts of alternative responsibilities assignments.
 - Graphical Representation
 - Visually communicate design choices

USE CASE MAPS (UCM)

- Use Case Maps (UCM)
 - graphically describes the functional requirements of a software system,
 - precisely describe processes (alt. view to Message Sequence Chart),
 - represents control flows and distributes responsibilities to actors, by superimposition of scenario paths on a structure of components.
- Use Case Map fairly addresses
 - operational requirements,
 - functional requirements,
 - architectural *trade-offs*.
- At later stages, UCM may be further refined into:
 - MSCs or UML sequence diagrams,
 - SDL state machines or UML state chart diagrams.

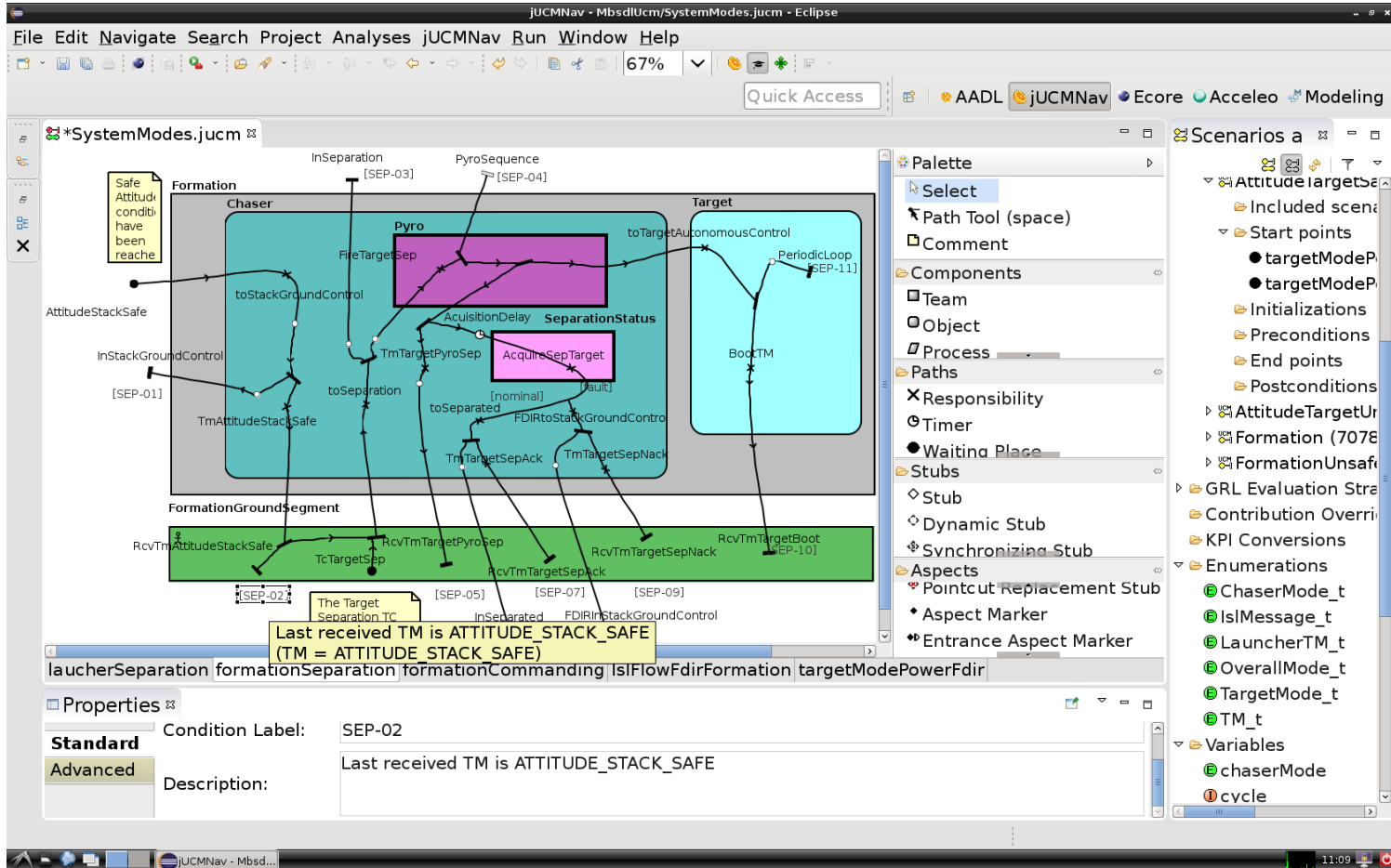
No new concepts, but same concepts introduced earlier in the development cycle.

JUCMNAV

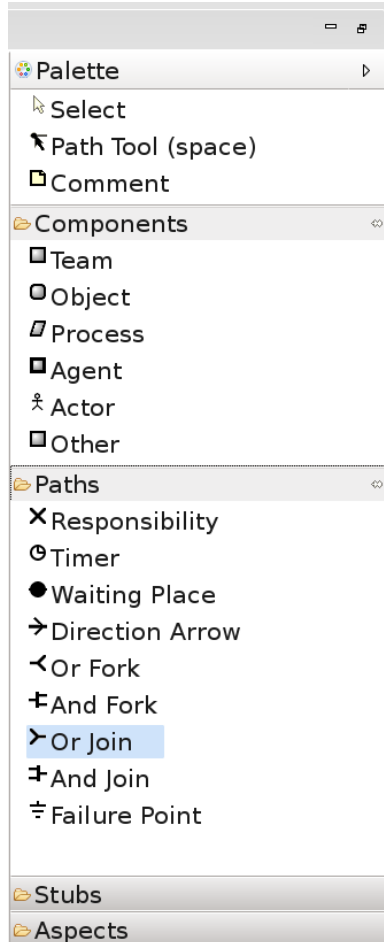


- UCM Tool
- Free Eclipse-based plug-in/editor
 - Now in version 6.0, August 2014
 - Since May 2006
- Dedicated Perspective, Graphical editor with palette
- Eclipse Navigator, Outline view and Preferences
- EMF metamodel
 - Allows model to model and model to text transformations
- Export to graphical representations (.jpg, .dot, .png)
- Export to Message Sequence Charts (.jucmscenarios)
 - Included Message Sequence Charts viewer
- Allows the early definition, execution and verification of Scenarios
- Allows graphical animation of Scenarios
- Cross platform (Linux, Windows 7, Mac OS X)

JUCMNAV: PERSPECTIVE



JUCMNAV: PALETTE



- *Structure of components*
 - Team, Object, Process, Agent, Actor.
- *Scenario paths*
 - Start Point : Scenarios initial conditions
 - **End Point** : Scenarios completion conditions
 - **This is the place to allocate/verify requirements**
 - **Textual description**, but also **formal expression to be verified** on scenario completion. Boolean expression based on variables updated by Responsibilities.
 - Or Fork : Execution path Or Fork (logical conditions)
 - And Fork : Execution path And Fork (logical conditions)
 - Or Join : Execution path Or Join
 - And Join : Execution path And Join
 - Responsibility: Pseudo-code fragment
 - Stubs : Hierarchical decomposition, refer another UCM

JUCMNAV: CODE/CONDITION EDITOR

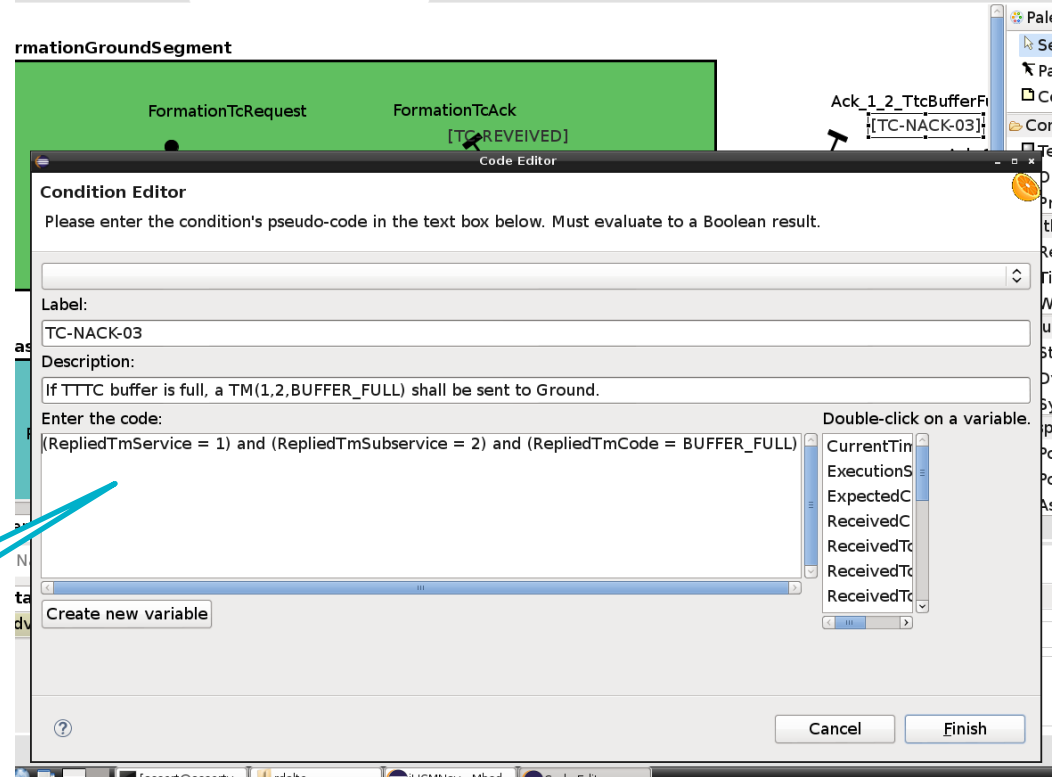
Editor for

- Responsibility code
- Requirement condition

Variables list

Variables creation

Syntax
Verification



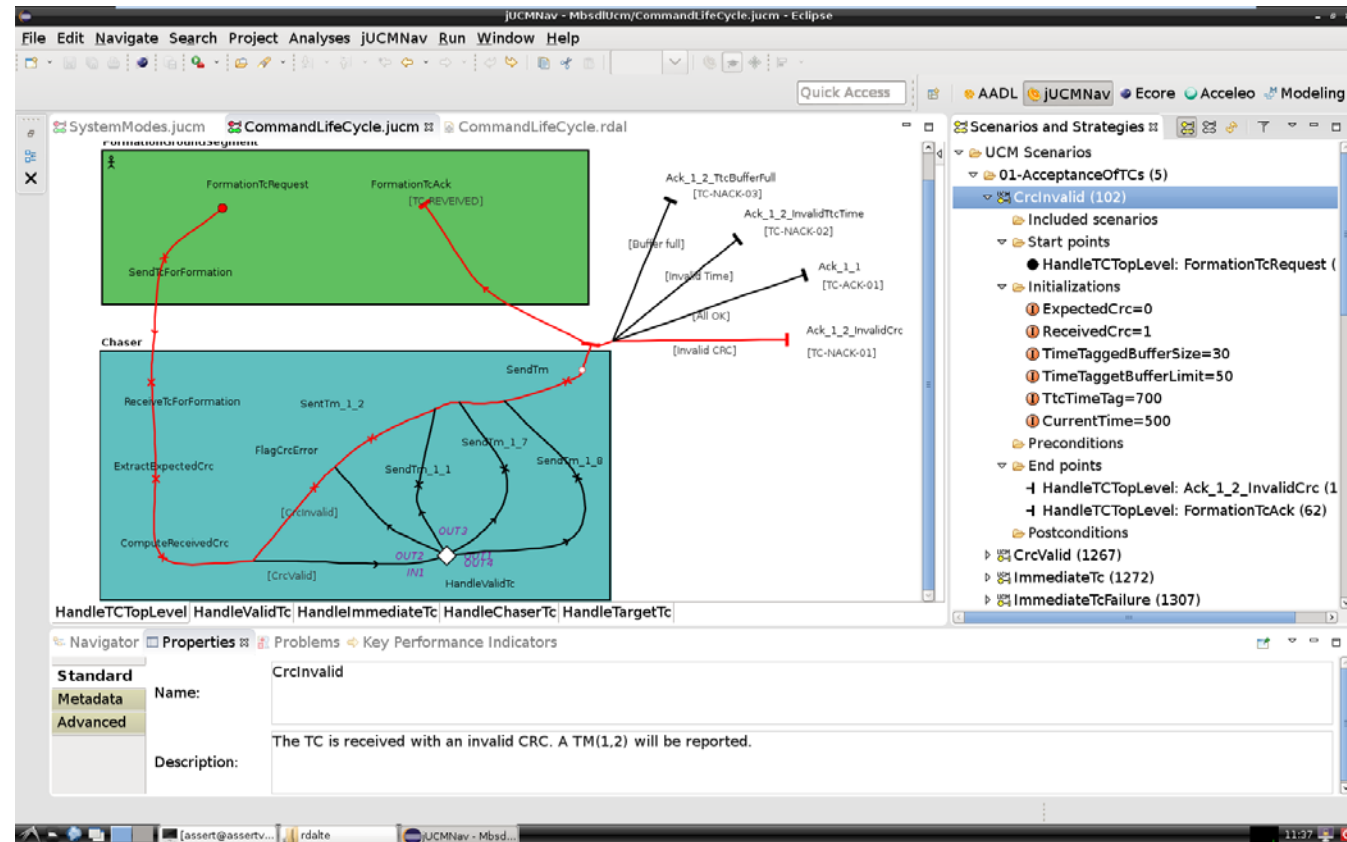
JUCMNAV: SCENARIO EXECUTION

Scenario:

- Start points
- Initializations
- End points
- Inclusions

Scenario Group:

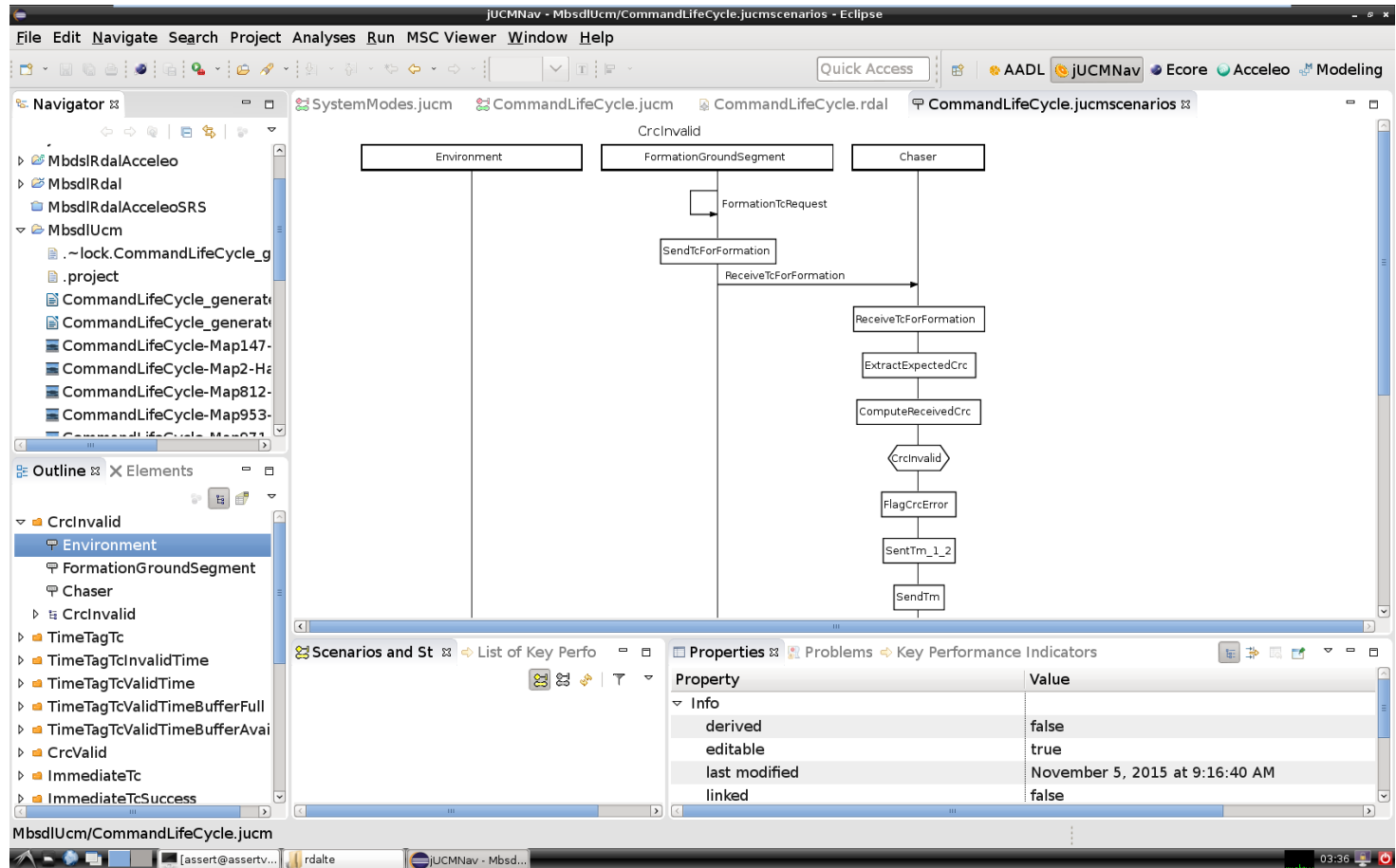
- Run all
- Coverage



Unsatisfied Conditions reported as errors in problems panel

JUCMNAV: MESSAGE SEQUENCE CHART EXPORT

Once
Executed
scenarios
can be
exported
and viewed
as MSC



GENDOC: MODEL BASED SSS GENERATION

Template based
Model2Txt
transformation



Embedded Aceleo
in .odt or .docx
templates

Support for tables
and figures

Test coverage and
requirements
traceability's

The left screenshot shows a template for a requirement table. The table has the following structure:

Requirement Id	[endPoint.name]/-[endPoint.postcondition.label/]
Description	[endPoint.description/]
Formal expression	[endPoint.postcondition.expression.replaceAll("\\n", "/")]
Formal description	[endPoint.postcondition.description/]
Verified by scenario	[endPoint.scenarioEndPoints.endPoint.scenarioEndPoints.scenarioDef.name/]
Test methods	[for (Method: Metadata endPoint.AllContents(Metadata) ->select(m:Metadata m.name = 'testMethod')) separator (' ') Method.value/ /for]
Parent requirements	[for (pReq: Metadata endPoint.AllContents(Metadata) ->select(m:Metadata m.name = 'parentReq')) separator (' ') pReq.value/ /for]
Child requirements	[for (pReq: Metadata endPoint.AllContents(Metadata) ->select(m:Metadata m.name = 'childReq')) separator (' ') pReq.value/ /for]

The right screenshot shows a generated diagram with a table for Requirement Id: Ack_1_1-TC-ACK-01. The table has the following structure:

Requirement Id	Ack_1_1-TC-ACK-01
Description	If the received TC is valid, a TM(1,1,NO_ERROR) shall be sent to Ground.
Formal expression	(RepliedInService = 1) and (RepliedInSubservice = 1) and (RepliedInCode = NO_ERROR)
Formal description	If the received TC is valid, a TM(1,1,NO_ERROR) shall be sent to Ground.

RDAL METAMODEL: SYSTEM OVERVIEW

Identification of the System and its Environment.

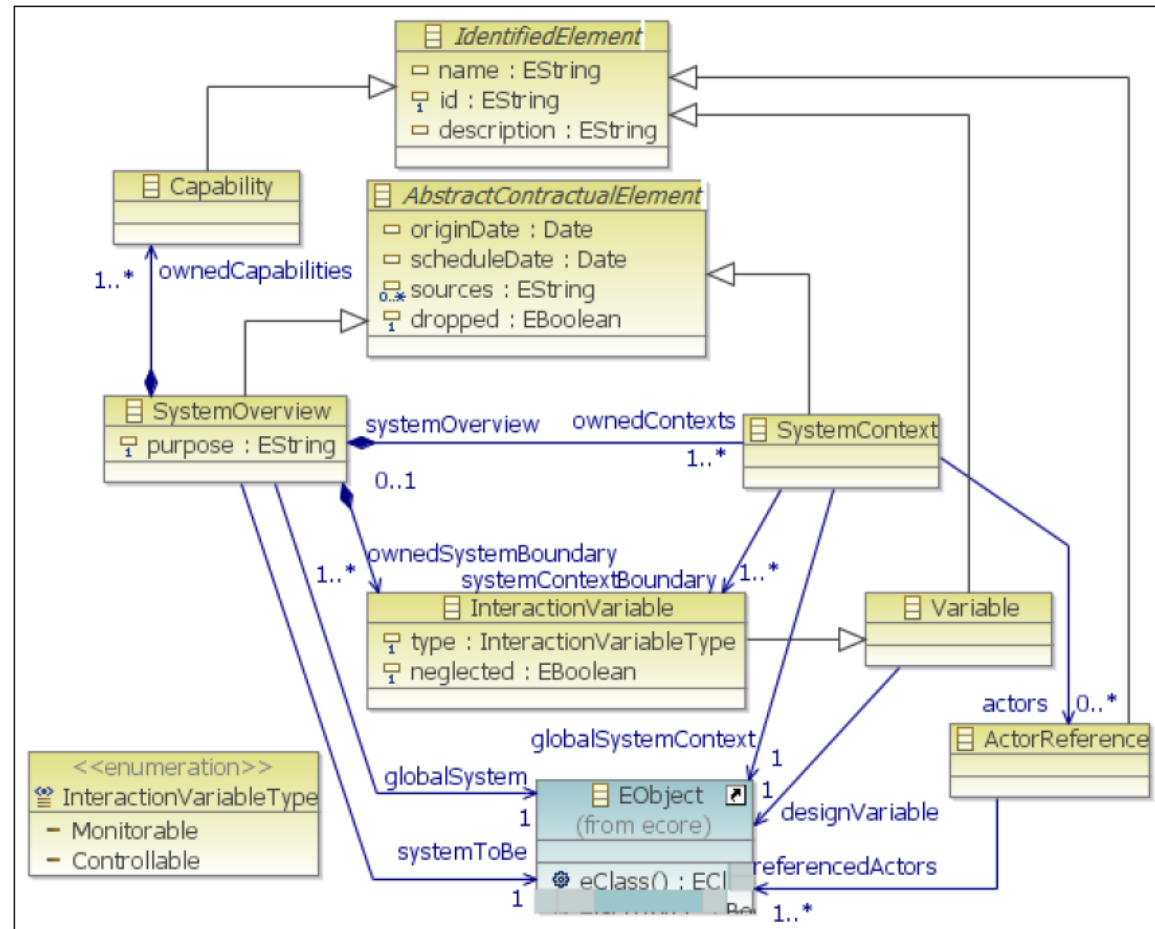
Identification of Data Interface.

Concept of

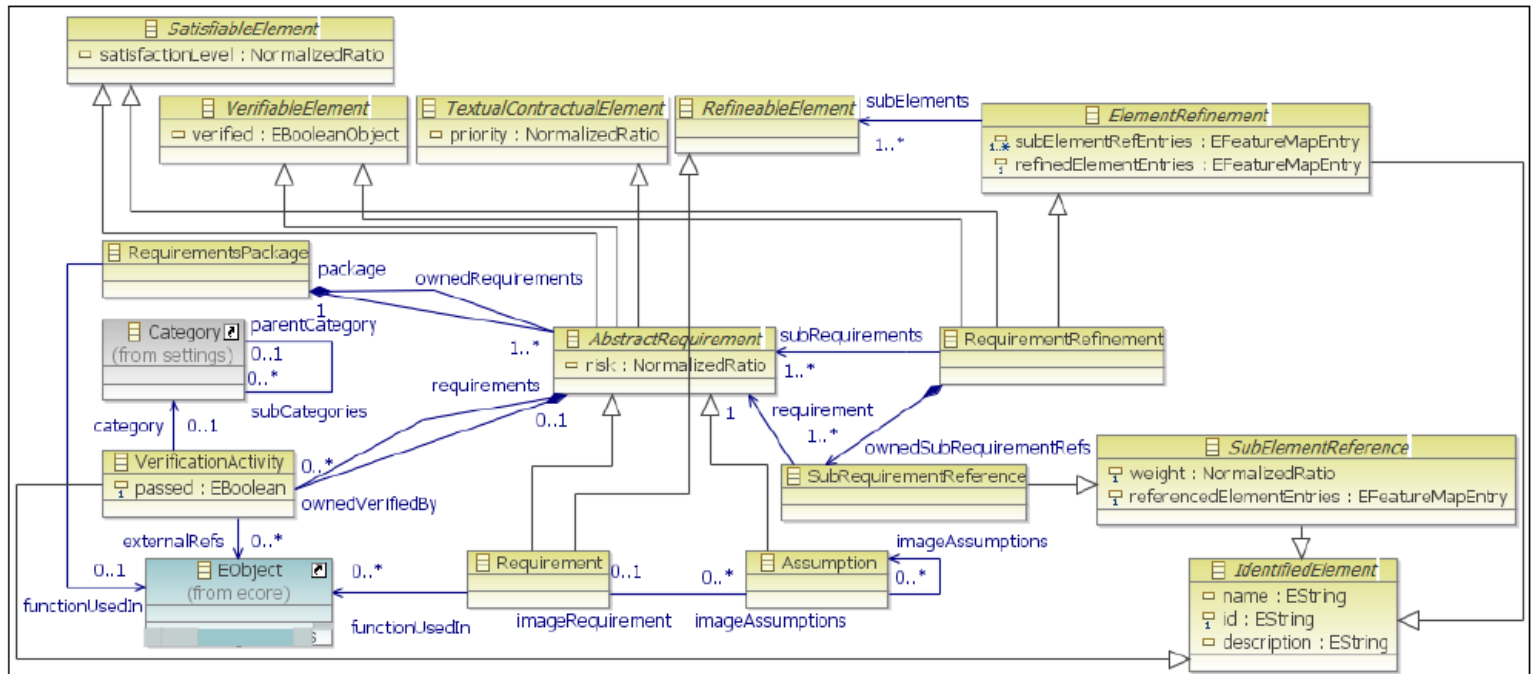
- Monitored Variables
- Controlled Variables

but only at top level.

Similar to Simulink Block inputs/outputs.



RDAL METAMODEL: REQUIREMENT



Organized in Packages

Complex structure, oversized for mid-size project

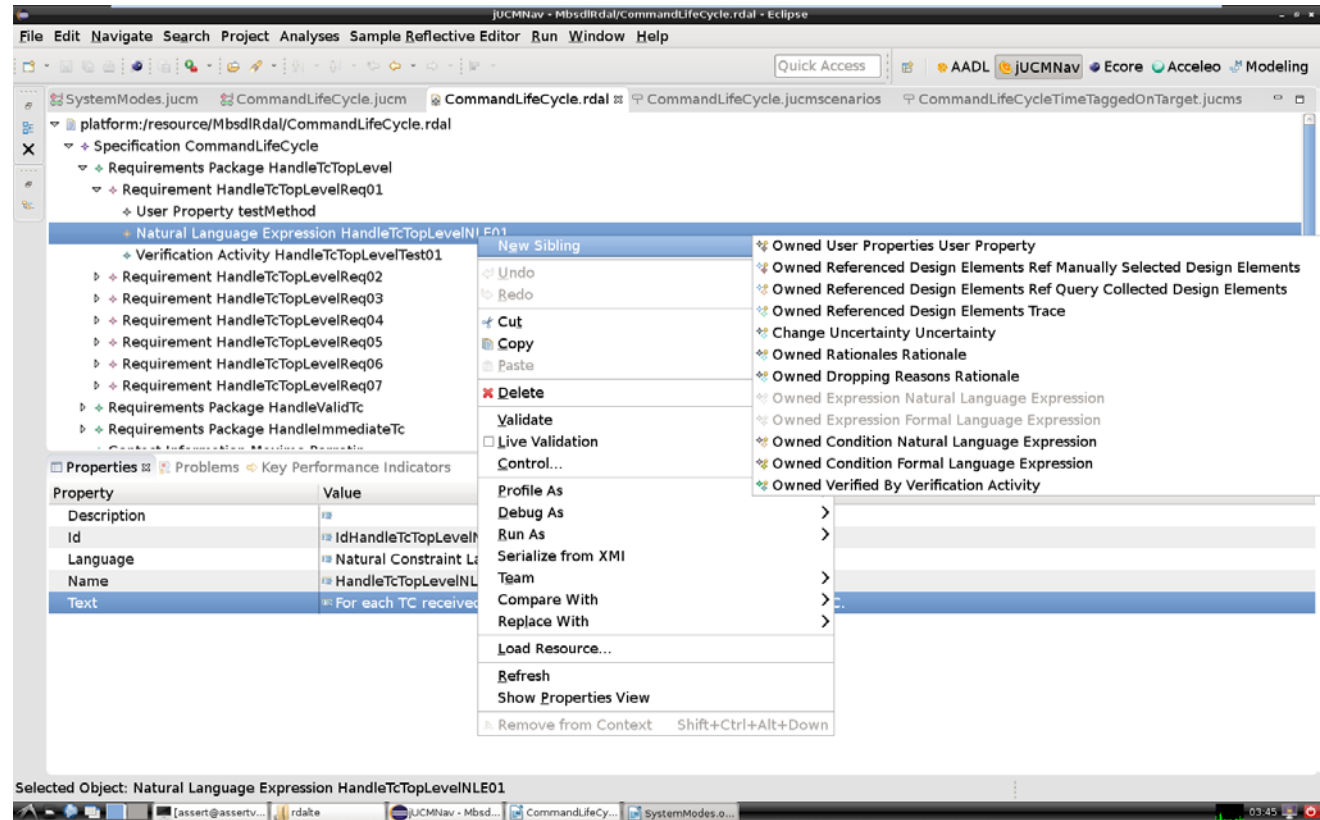
Reference to external models (traceability) through EObject

Natural language

Formal languages: BLESS, Lute, OCL but no provers support

RDAL TOOLING: RDALTE

- EMF meta-model
- EMF Sample Reflective Model Editor
- Live validation



- RDAL Graphical Representation V2 not yet implemented by RDALTE
- **Export to textual representation implemented in the project (UC)**

GENDOC: MODEL BASED SRS GENERATION

The image displays two windows of LibreOffice Writer. The left window, titled 'CommandLifeCycle.odt', shows a template for a Software Requirement Specification (SRS) document. It contains various placeholders for metadata and content, such as `<self.name/>`, `[self.contactInformation.name/]`, and `[self.description/]`. A small table is also visible at the bottom of the template.

The right window, titled 'CommandLifeCycle_generated.odt', shows the rendered output of the SRS document. It includes the following content:

Modified : Wed Nov 11 23:00:00 EST 2015
 Client : Maxime Perrotin maxime.perrotin@esa.int
 0031715654923

Description
 This section briefly describe the system under specification.
 This document is SRS of TM/TC service of Flying Formation. These scenarios show how Ground TCs are managed by the Formation in terms of CRC checks, acceptance and completion reporting, Time Tagged TC management and routing between Chaser and Target.

Specification part
 This chapter is the specification part of system.

Specification package HandleTcTopLevel
 A TC is sent from the Formation Ground Segment to the Formation. It is received by the Chaser. The Chaser will check the received CRC again the compute one. If the CRC is invalid, a TM(1,2) with CrcError will be sent to the Formation Ground Segment. If the CRC is valid, it will be handled according to next 'HandleValidTc' UCM.

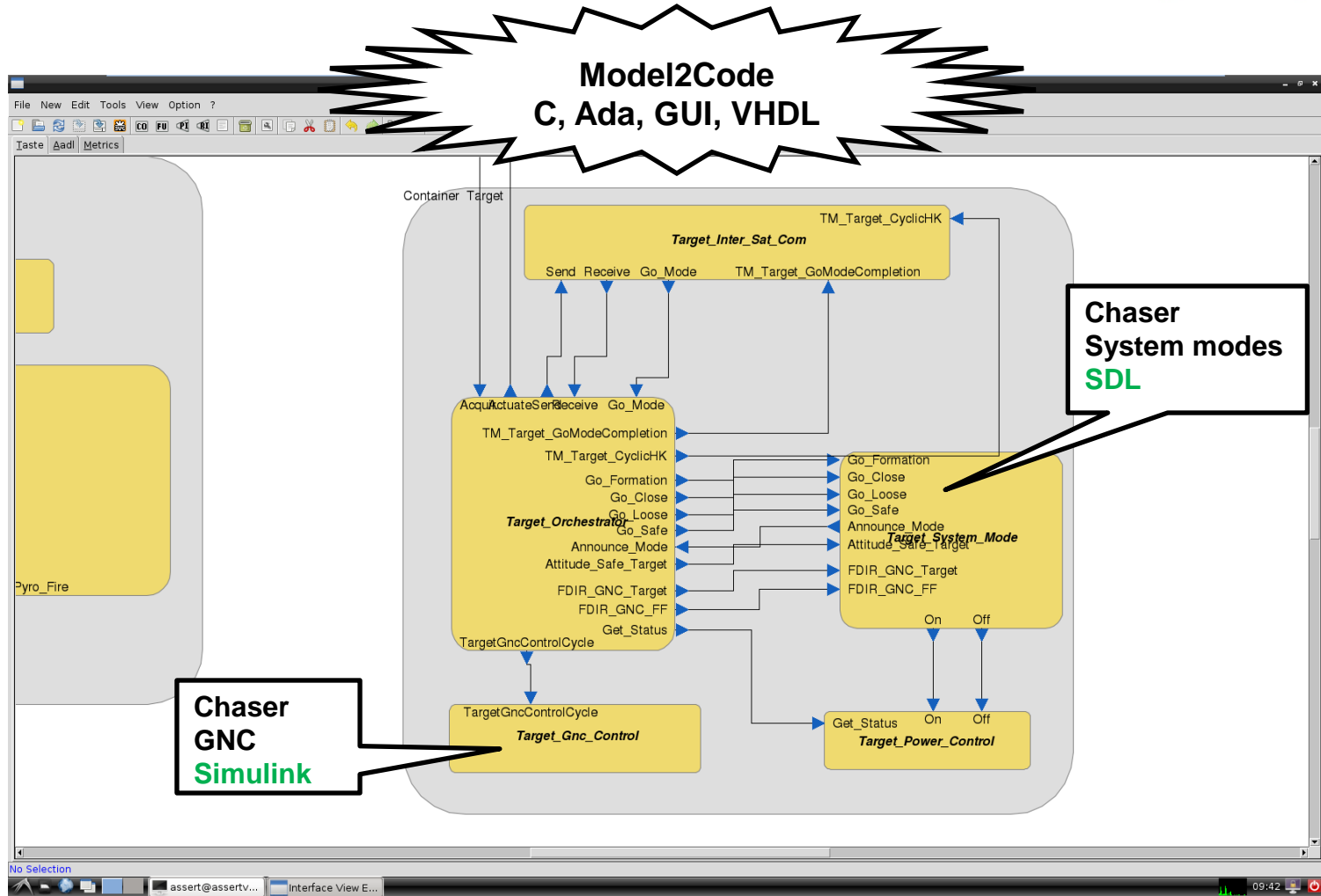
Requirement Id : HandleTcTopLevelReq01-HandleTcTopLevelINLE01

Requirement Id	<u>HandleTcTopLevelReq01-HandleTcTopLevelINLE01</u>
Description	
Formal expression (Natural)	For each TC received for the Formation, the Chaser shall extract the CRC.
Formal description	
Verified by scenario	

The status bars at the bottom of both windows indicate the page number, word count, and character count. The left window is on page 1 of 2, and the right window is on page 1 of 10.

Same M2T tooling as for jUCMnav
 No graphical representation

TASTE: AADL AND ASN.1 OPEN-SOURCE EDITOR



OPENGEODE: SDL OPEN-SOURCE EDITOR

The screenshot displays the OpenGEODE editor interface. On the left, a code editor shows SDL code for 'process target_system_mode'. A central diagram shows a state transition graph with nodes like 'Go_Formation', 'On(radar)', 'On(gps)', 'TargetLooseFF', 'Go_Close', 'TargetCloseFF', 'Go_Loose', 'Go_Safe', 'FDIR_GNC_FF', 'Off(gps)', 'Off(radar)', 'Off(thruster)', and 'TargetAutonomousControl'. On the right, a state transition diagram shows nodes for 'targetgroundcontrol', 'targetlooseff', 'targetcloseff', and 'targetautonomouscontrol' with transitions labeled 'Go_Formation', 'Go_Close', 'Attitude_Safe_Target', 'Go_Loose', 'Go_Safe', 'FDIR_GNC_FF', and 'Go_Safe'. Below the diagrams, a 'Data types' section lists variables like 'T-Orbv', 'T-Attitude', 'phi-t', 'theta-t', 'psi-t', 'wx-t', 'wy-t', 'wz-t' with their respective data types. A console window at the bottom shows warning messages about expression evaluation ranges.

**Model2Code
Ada and LLVM**

**SDL
variables**

**SDL Events:
TC, FDIR,...**

**SDL Actions:
On/Off, counters**

**SDL States:
System modes**

**Equivalent
FSM**

**ASN.1
Data Model**

Static Analysis

01 Model Based Methodology (P. López, UC)

02 Timing Analysis in TASTE (J. Garrido, UPM)

03 Mission Proof of concept (D. Torette, Spacebel)

04 Conclusions

CONCLUSIONS

- Software Requirements Specification has been integrated into a Model Based Software Development Lifecycle
- Use Case Maps and rdal language tools have been integrated in TASTE
- The methodology and the tools have been evaluated on a realistic mission
- Rapitime has been integrated in TASTE

- **Software Requirements can be adequately modelled using rdal and UCM, providing a bridge with system architects**
- **TASTE is an excellent platform to add new functionality**



Joan Clua

jclua@indra.es

Joan Ametller

Marc García

Laura Perea

Marcos Venteo



Dominique Torette

Dominique.Torette@spacebel.be

Paul Parisis

Philippe Créten



Patricia López Martínez

lopezpa@uncan.es

Michael González Harbour



Jorge Garrido

jgarrido@dit.upm.es

Juan Antonio de la Puente

POLITÉCNICA

Jian Guo

J.Guo@tudelft.nl



MODEL-BASED SOFTWARE DEVELOPMENT LIFECYCLE



**TEC-ED & TEC-SW Final Presentation Days
December 9th 2015**

THANK YOU VERY MUCH FOR YOUR ATTENTION