

For a reliable and scientific approach in system and software engineering



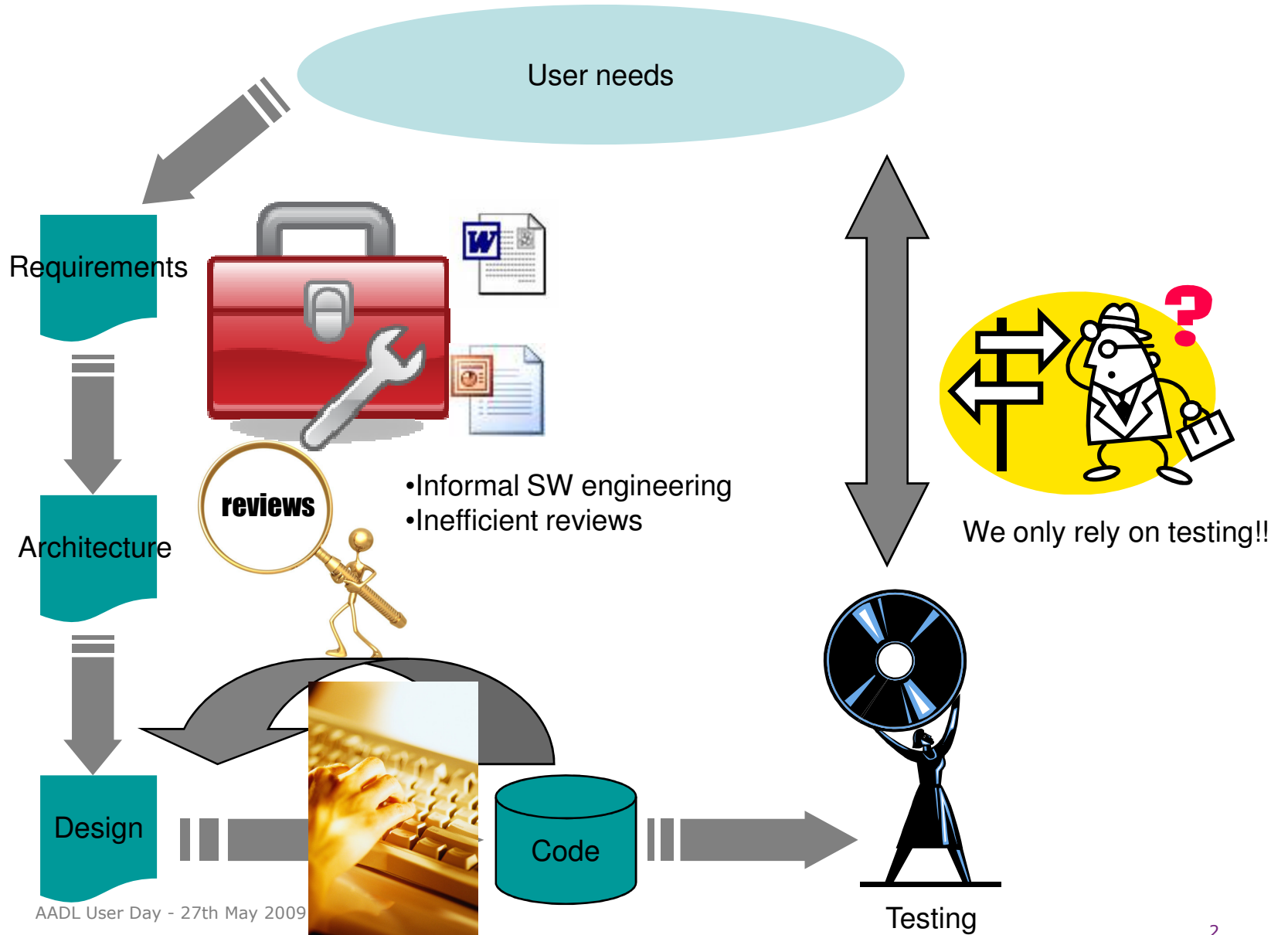
Results of ASSERT and follow-up studies: technical status and exploitation activities.

AADL User Day - 27th May 2009

assert-project / ESA - E. Conquet, M. Perrotin, C. Colombo, M.-A. Esteve - TEC-SWE



SW engineering: current approach and tools



AADL User Day - 27th May 2009



Our real needs for SW engineering of the 21st century.

- Formal languages to capture:
 - SW architecture,
 - Required properties,
 - Data and Interfaces,
 - Behaviour.
- Multi-formalisms support to handle heterogeneity,
- Auto-coding capabilities to ensure consistency,
- Fast and safe design round-trips to improve reactivity to late changes,
- Real-time executive to guarantee the preservation of properties (What we prove is what we get).

ASSERT provides answers to real user needs

For a reliable
and scientific
approach
in system
and software
engineering



The ASSERT process and toolset in details

Maxime Perrotin (TEC-SWE)

AADL User Day - 27th May 2009

assert-project / ESA - E. Conquet, M. Perrotin, C. Colombo, M.-A.Esteve - TEC-SWE

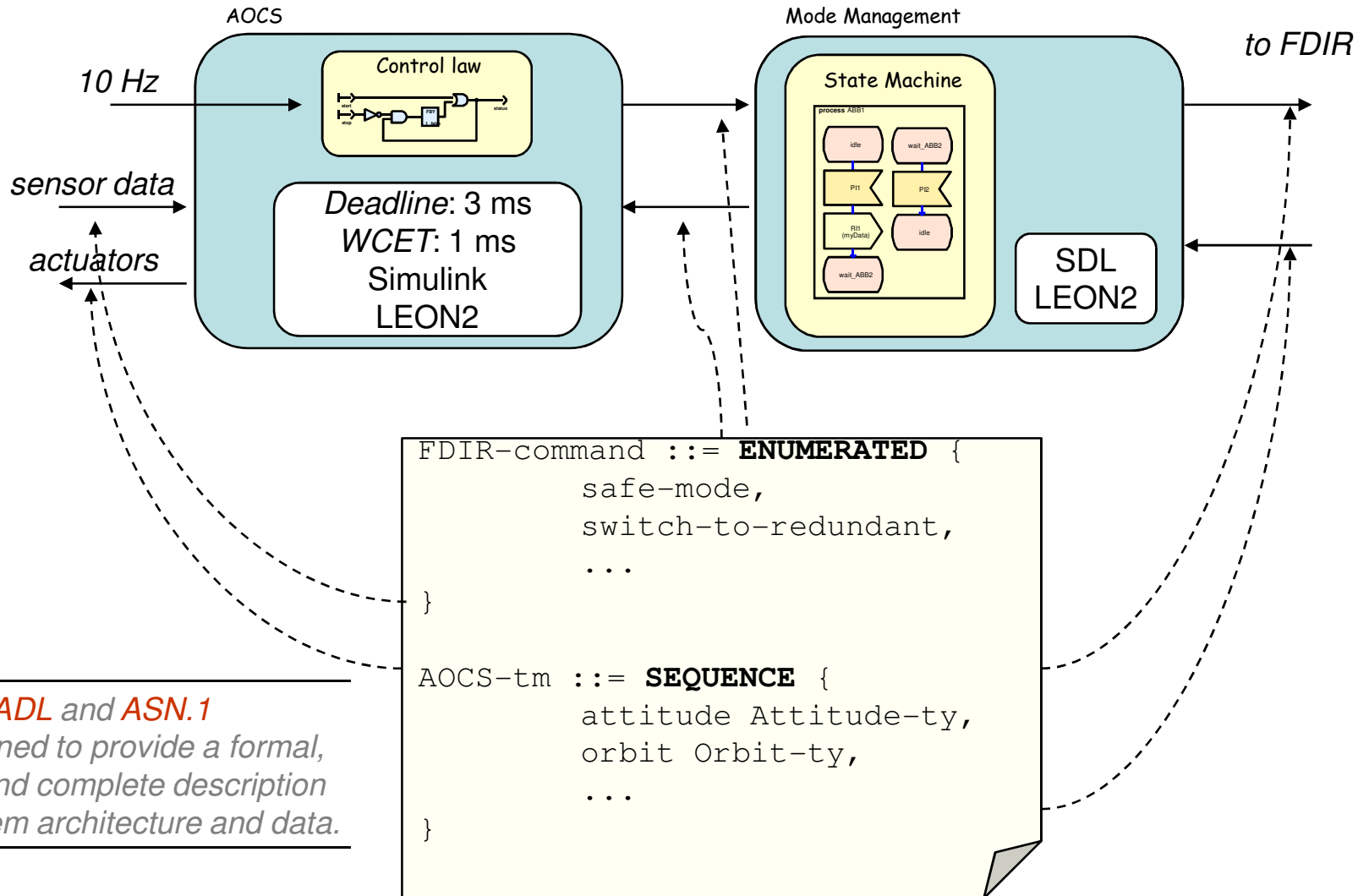


What is the Assert process?

- **The Assert process is based on simple observations**
 - a system – ANY system – is made of *heterogeneous components*, that have to live and communicate together
 - system builders have other concerns than *software implementation details*,
 - and good software engineers are unhappy when they have to develop application code: their skills are misused
- **The Assert process proposes solutions to**
 - capture a system using user-friendly (yet formal) modelling techniques
 - automate repetitive and error-prone software activities
 - build an homogeneous system having heterogeneous components
- **The toolset has been specified, designed, and implemented by ESA together with some Assert partners.**
- **It is unique on the market.**

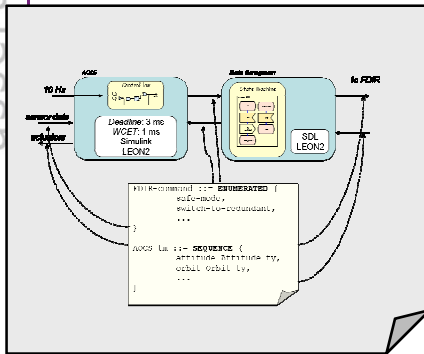


Capture of the system : architecture, behaviour, data, real-time attributes, and hardware platform.

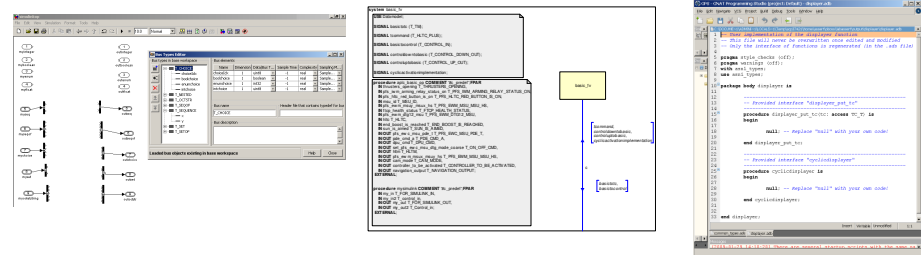


AADL and ASN.1
are combined to provide a formal, precise, and complete description of the system architecture and data.

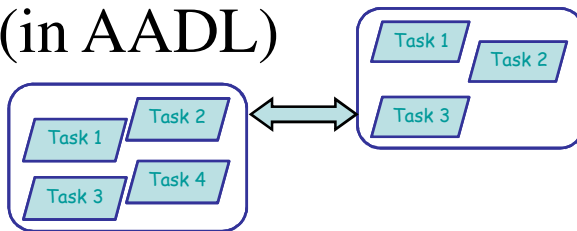
What Assert tools do with the models



① Generate “application skeletons” in Simulink, SDL, C, and Ada



② Generate a software real-time architecture (in AADL)

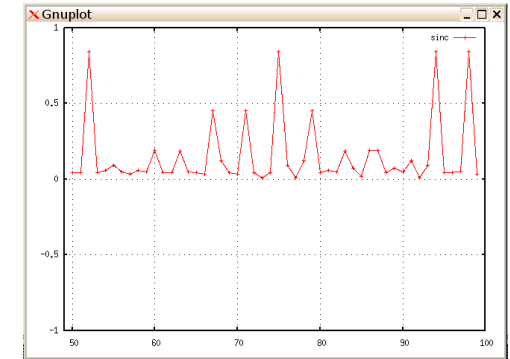
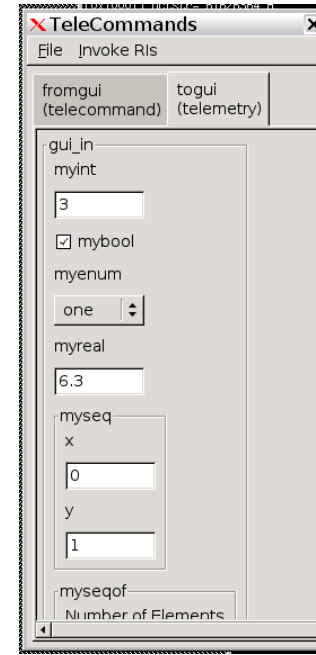


③ Generate glue code to put everything together on a real-time operating system



In addition - bonuses

- Rapid prototyping: the toolchain generates GUIs to quickly test the system under development
- Simulation and Analysis: Data can be monitored using real-time plotting.
- Documentation: ICDs are generated automatically with a description of the data binary encodings (ASN.1 uPER Encodings)

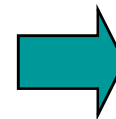


```

MY-MODULE DEFINITIONS ::= BEGIN

MySequence ::= SEQUENCE {
  field1    INTEGER (5..4294967295),
  field2    INTEGER (5..4096) OPTIONAL,
  field3    BOOLEAN ,
  field4    MyChoice,

```



MySequence (SEQUENCE)				min	max
Sequence preamble				46	∞
Bit mask				2	2
No	Field	Type	Optional	Min length	Max length
1	field1	INTEGER	No	32	32
2	field2	INTEGER	Yes	12	12
3	field3	BOOLEAN	No	1	1
4	field4	MyChoice	No	3	162
5	field5	OCTET STRING	No	8	∞
6	field6	MySequenceOf	Yes	16	1207

For a reliable and scientific approach in system and software engineering



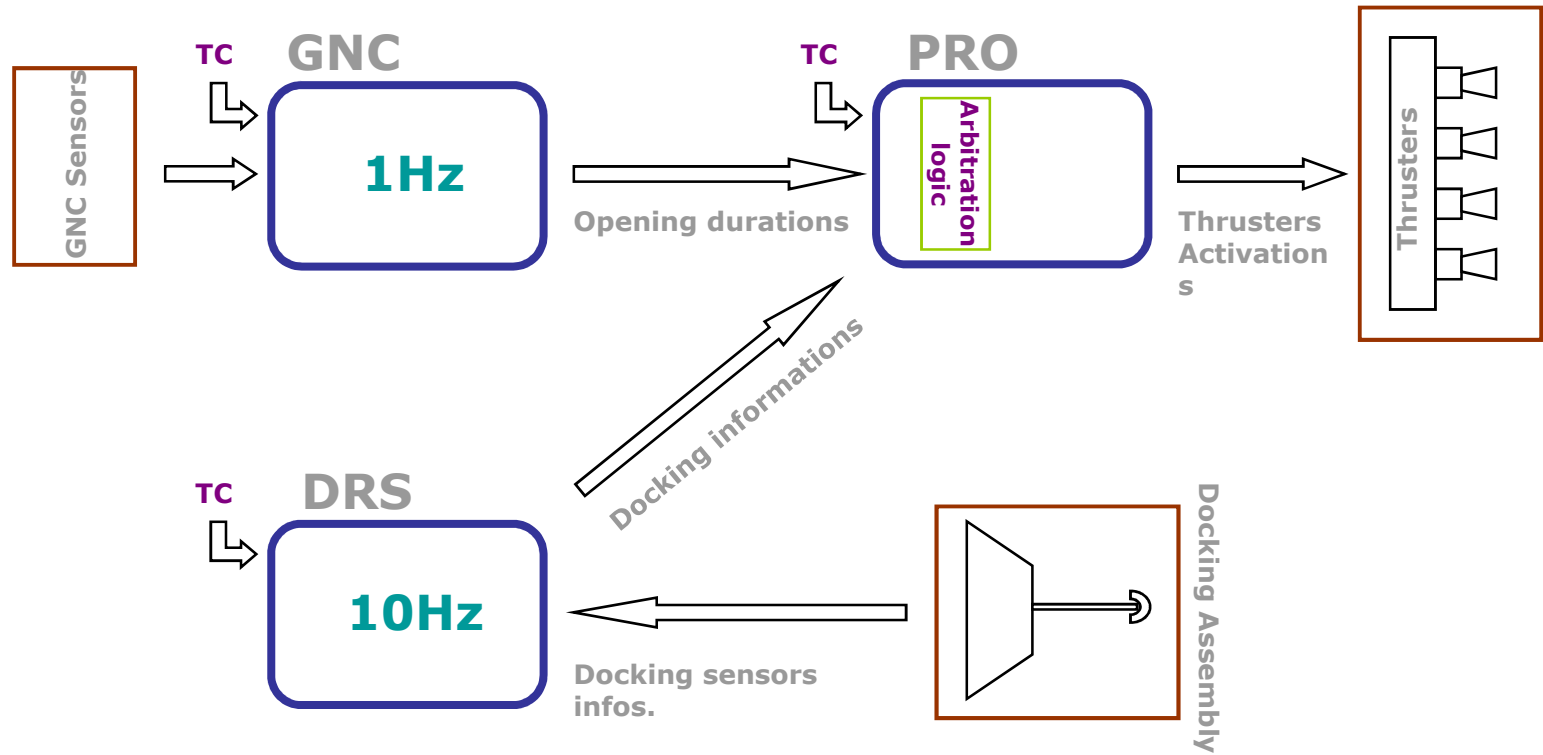
The ASSERT Process in action

Cyril Colombo, Marie-Aude Esteve (TEC-SWE)

AADL User Day - 27th May 2009

assert-project / ESA - E. Conquet, M. Perrotin, C. Colombo, M.-A.Esteve - TEC-SWE

Case study for this demonstration the ATV docking sequence

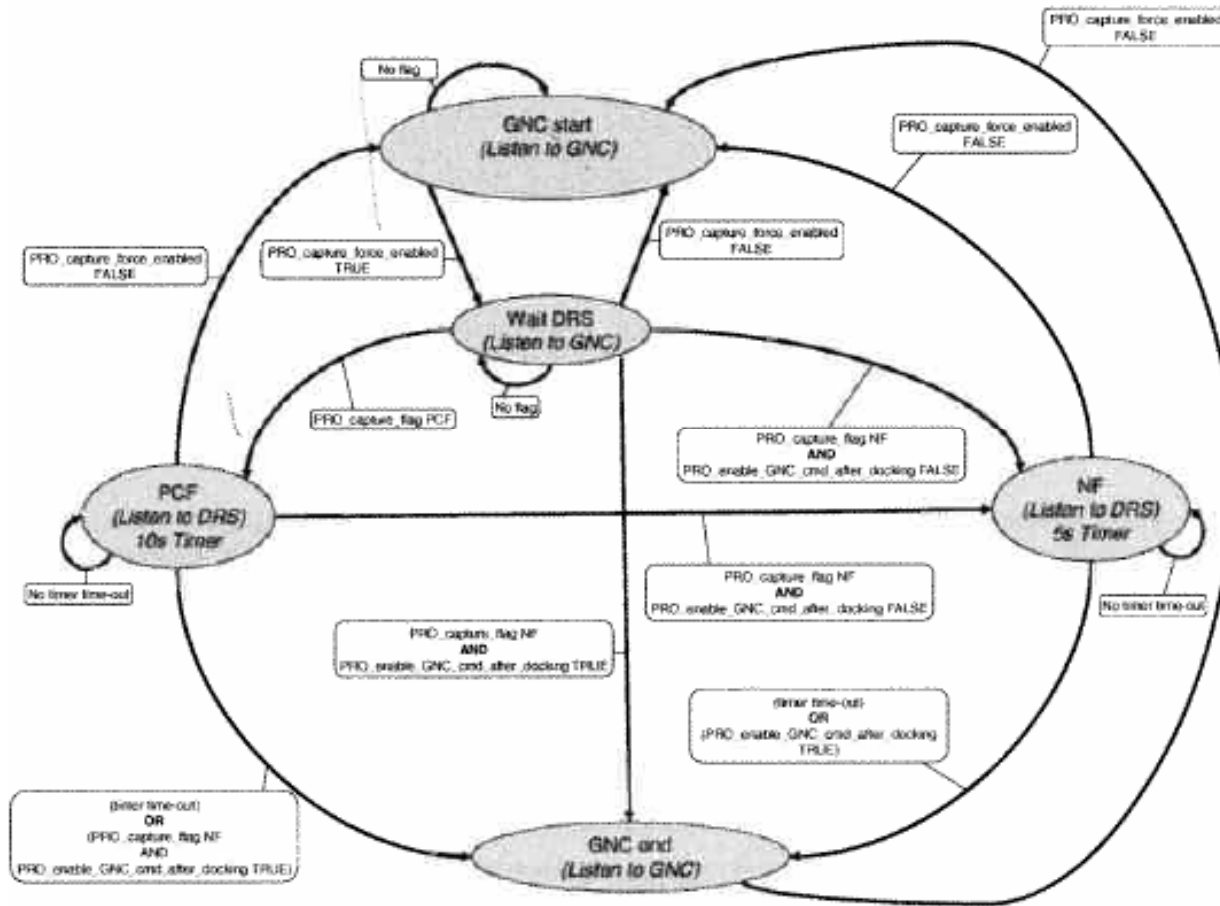


Arbitration logic must :

- respect reactivity constraints
- be robust to any nominal or non nominal situation
- be robust to asynchronous request sequence such as TC

The arbitration logic behaviour

PCF algorithm :





How to integrate this behavioural model in our example using the ASSERT process ?

Select an appropriate modeling language to model the expected behavior

SDL is a very good candidate in this situation

- Allows to clearly specify the expected behavior as a Finite State Machine
- Allows to further check some properties at **model level** i.e. prior to integration at code level
- Allows to be transparently integrated in the application thanks to the ASSERT tool chain

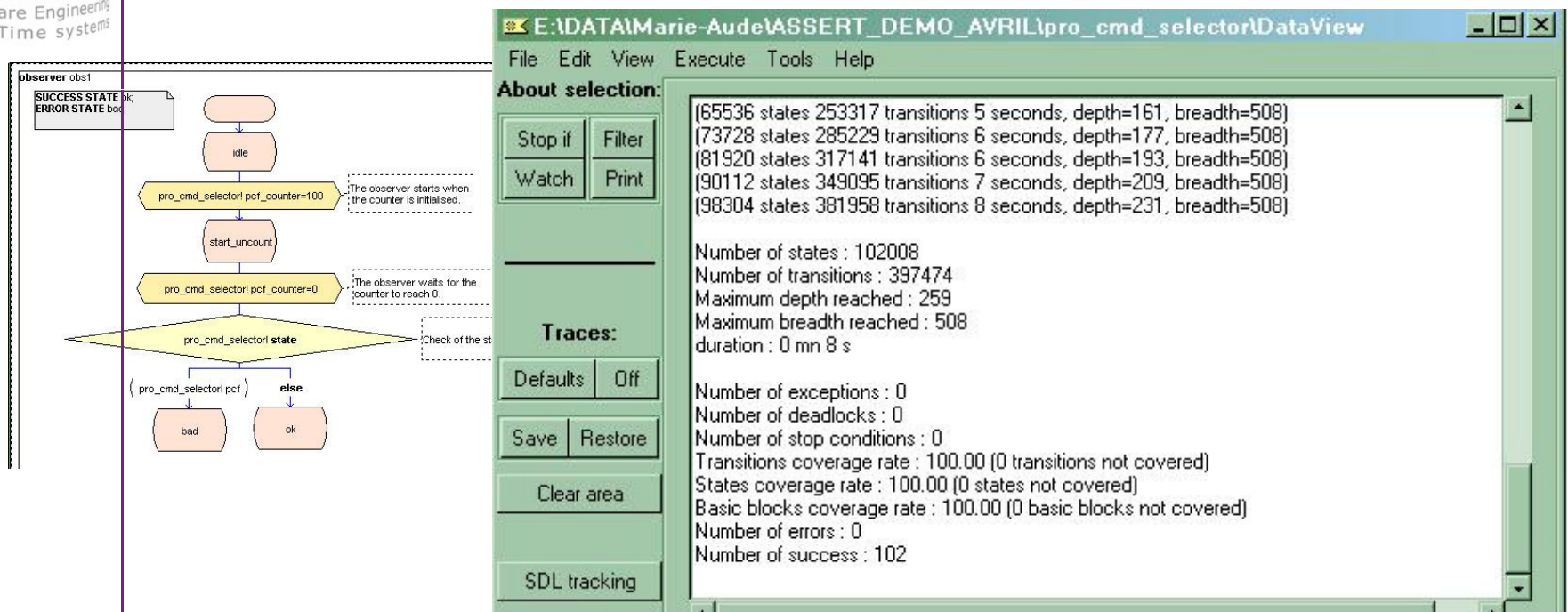
Simply defining the modeling language at Interface View level allows:

- 1. To automatically generate a correct by construction model interfaces & skeleton from where to start the modeling work**
- 1. To regenerate automatically the full system without having to write manually a single line of code !**

Examples of proofs made on the SDL model

- “The reception of the TC disabling the PCF ensures in any case the selection of the GNC as the command source ”
- “The PCF state of the automaton will be left at least at the expiration of the PCF timer”
- “There is no combination of input leading to a deadlock of the automaton”

Results with **exhaustive** simulation:



observer obs1

SUCCESS STATE ok;
ERROR STATE bad;

idle

pro_cmd_selector! pcf_counter=100

start_uncount

pro_cmd_selector! pcf_counter=0

pro_cmd_selector! state

(pro_cmd_selector! pcf) else

bad ok

The observer starts when the counter is initialised.

The observer waits for the counter to reach 0.

Check of the st

E:\DATA\Marie-Aude\ASSERT_DEMO_AVRIL\pro_cmd_selector\DataView

File Edit View Execute Tools Help

About selection:

Stop if Filter
Watch Print

Traces:

Defaults Off
Save Restore
Clear area
SDL tracking

(65536 states 253317 transitions 5 seconds, depth=161, breadth=508)
(73728 states 285229 transitions 6 seconds, depth=177, breadth=508)
(81920 states 317141 transitions 6 seconds, depth=193, breadth=508)
(90112 states 349095 transitions 7 seconds, depth=209, breadth=508)
(98304 states 381958 transitions 8 seconds, depth=231, breadth=508)

Number of states : 102008
Number of transitions : 397474
Maximum depth reached : 259
Maximum breadth reached : 508
duration : 0 mn 8 s

Number of exceptions : 0
Number of deadlocks : 0
Number of stop conditions : 0
Transitions coverage rate : 100.00 (0 transitions not covered)
States coverage rate : 100.00 (0 states not covered)
Basic blocks coverage rate : 100.00 (0 basic blocks not covered)
Number of errors : 0
Number of success : 102



On this specific example, the ASSERT process and toolset would have been more efficient

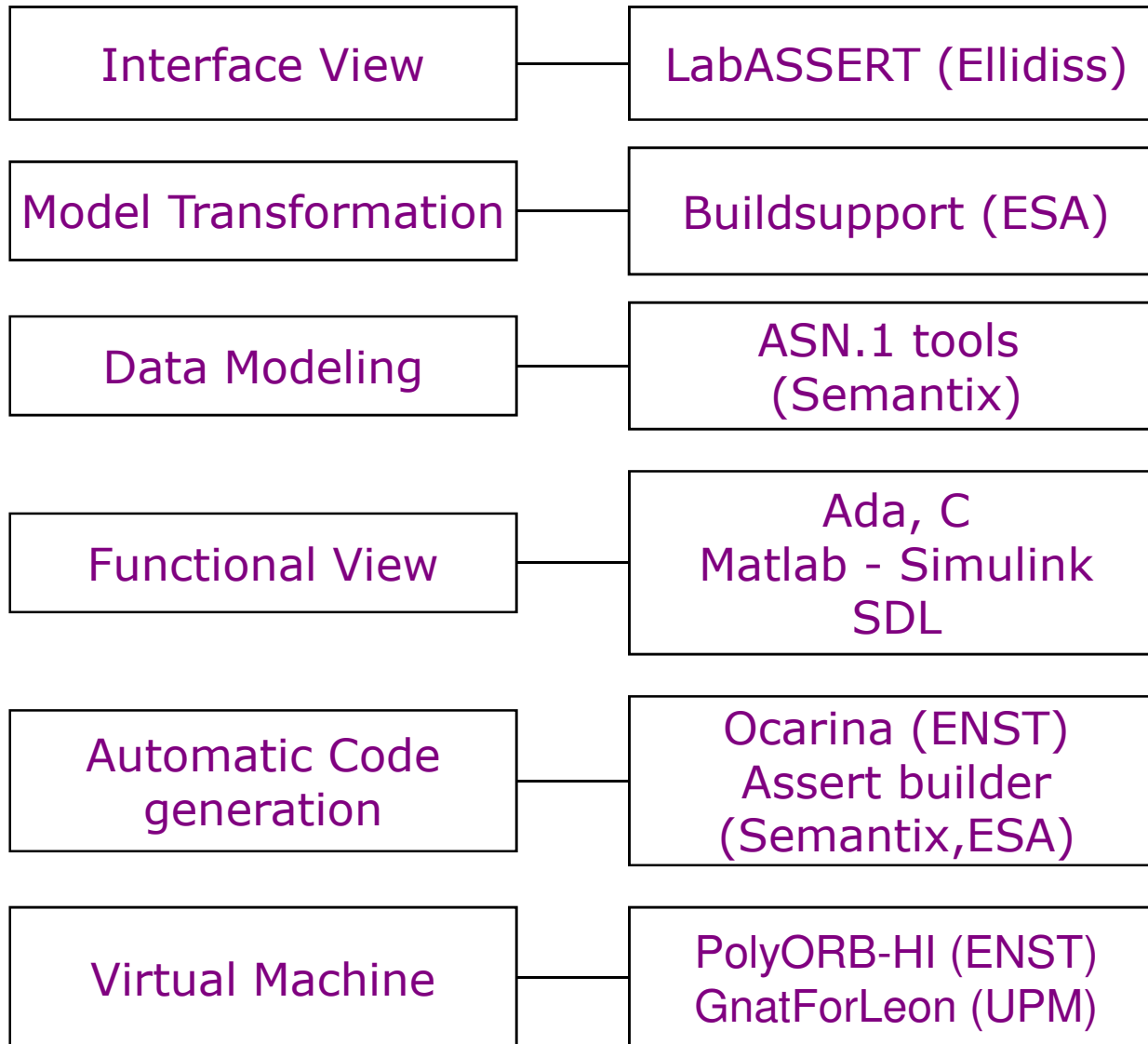
- During the ATV development, the arbitration logic has been specified and implemented manually:
 - No formal verification had been done before the actual SW implementation
 - The behaviour has been “only” verified by testing
- Using a model-based approach and the ASSERT toolset would have allowed to:
 - **better specify** the needs,
 - **catch specification issues** before the RB was issued,
 - **make** more complete & **systematic verifications**,
 - **automatically generate** the final OBSW from the model, minimizing the risk of introducing faults during the coding process



What has been experienced in this demonstration

- Transparent use of complementary languages (Matlab, SDL, Ada and C). Data types are converted automatically from one language to the other (both at model and code level)
- Automatic generation of binary encoders and decoders for data exchanged in distributed systems
- Use of a high level model for a precise description of the system architecture and properties: no ambiguities, easy verification and code generation with existing tools
- Use of SDL for the description and verification of the system functional behavior
- Use of Matlab for the description of the system algorithms: dedicated, appropriate language (but not appropriate for state machines)
- Use of C and Ada for legacy code

Assert toolset - Credits





assertproject

For a reliable and scientific approach in system and software engineering

<http://www.assert-project.net>