

For a reliable
and scientific
approach
in system
and software
engineering



Leveraging AADL for High Level Architecture Modeling, Analysis and Generation The ASSERT Approach

RST Ada Europe 2009 - 12/06/09

assert-project - Dr Eric Conquet & Maxime Perrotin - European Space Agency



Starting with a statement of the problem.

- Targeted domain: Real-Time, mission critical and embedded system.
- SW dominant systems (But expandable to HW)
- Current development approach: mostly based on paper work.
 - Few models,
 - Tools: mainly MS office!
- Suppliers face difficulties to master design before testing
- Customers find reviews not efficient enough.
- Needs:
 - Capture system model with associated properties,
 - Verify early and continuously during design,
 - Smoothly handle system heterogeneity,
 - Use automatic coding.

Those needs were addressed in ASSERT.



The ASSERT Requirement baseline

- System families: from market segments to property oriented design,
- Proofs: from an empirical to a scientific approach,
- New development process: from nice concepts to actual steps,
- Tools: From paper to models,
- Heterogeneity handling: from multiple models to one single and integrated SW.
- Case studies: from toy examples to real cases.

ASSERT = an ambitious and pragmatic approach to develop critical SW dominant RT systems.



Bright and dark faces of our achievements.

- System families:
 - No usable implementation of reference architectures,
 - Technically very complex,
 - New business model to be setup,
 - Requirement capture phase too much driven towards existing solutions.
- The bright face: a process and a toolset.
 - Process is covering all phases from SW system definition and properties capture to SW generation and deployment,
 - The toolset fully supports all process steps and uses AADL as a system definition language.



Why has AADL been chosen?

- The ASSERT process is independent of any technology,
- Two different toolsets have been initiated in ASSERT:
 - One supports the AADL language (the one we are showing in this presentation).
 - A second one is using HRT-UML.
- The choice of AADL was motivated by:
 - The maturity and adequacy of the language,
 - A simple and readable textual formalism,
 - The facilities for capturing system properties,
 - The ability of making connections to external languages
 - The existence of the growing community around the AADL committee.



ASSERT and post-ASSERT activities

- The existing results we will show have been produced by both ASSERT and follow-up activities.
- The ASSERT IP delivered:
 - A first definition of the process,
 - Some early prototypes of tools,
 - First case studies,
- ASSERT follow-up activities:
 - Were funded by ESA (internal R&D Budget)
 - Transformed the first tool versions into a workable toolset,
 - Complete the list of case studies with additional examples.



Future of ASSERT: a strategy around three main axis.

- Extending the process and toolset
 - Link with system modelling: an ESA funded study to be kicked-off very soon.
 - Integration of HW components: Another ESA funded study to be started.
 - Integration of space specific components (PUS, Spacewire, ...): partly done but to be completed.
 - Connection to system simulator and schedulability analyser.
- Market the toolset
 - Create an ecosystem gathering innovative SMEs and research centres` into a network,
 - Each network node will lead a technical domain,
 - The network can be dynamically configured to address a specific need (virtual company),
 - To have ESA coordinating the network and protecting the community investment.
- Disseminate the technology.
 - Case study on formation flying experiment with space industry (Spacebel and Space System Finland).
 - Preparation of case studies with space and non space industry.

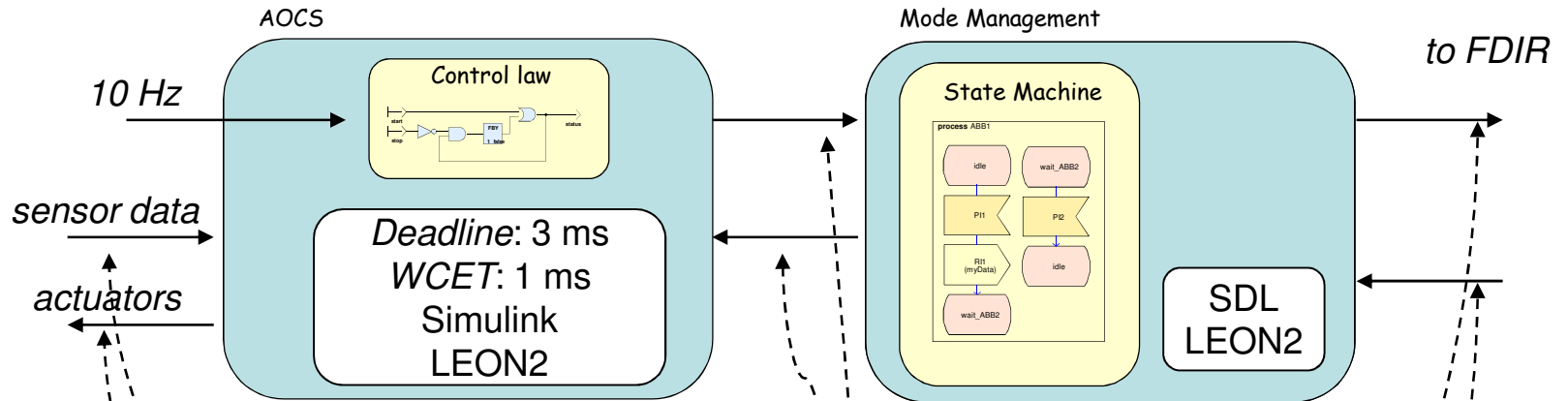


What is the Assert process?

- **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.



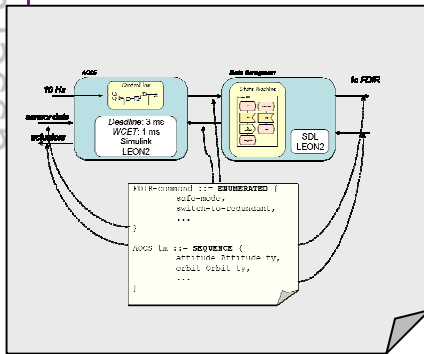
```

FDIR-command ::= ENUMERATED {
    safe-mode,
    switch-to-redundant,
    ...
}

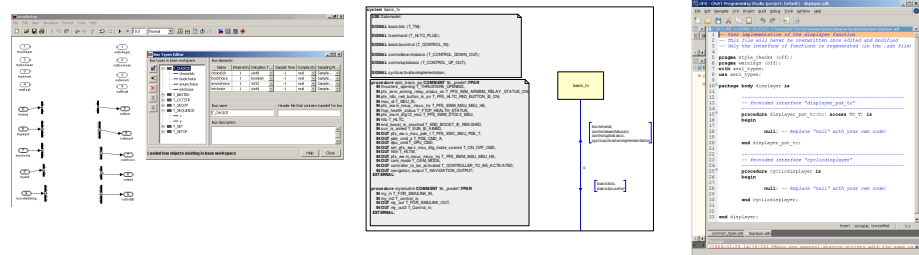
AOCs-tm ::= SEQUENCE {
    attitude Attitude-ty,
    orbit Orbit-ty,
    ...
}
    
```

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

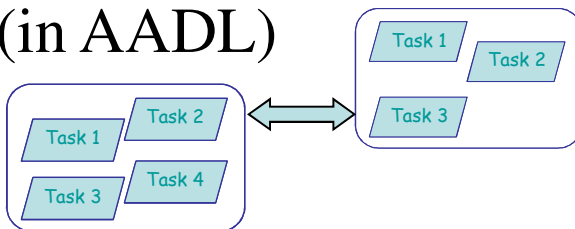
What Assert tools do with the models



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



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

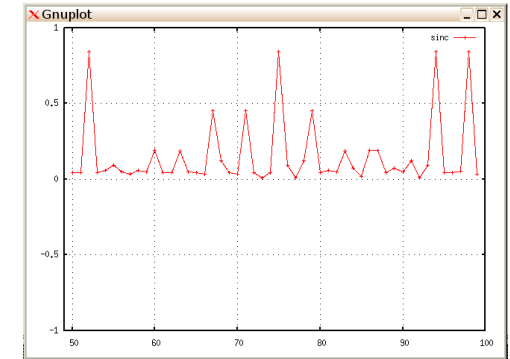
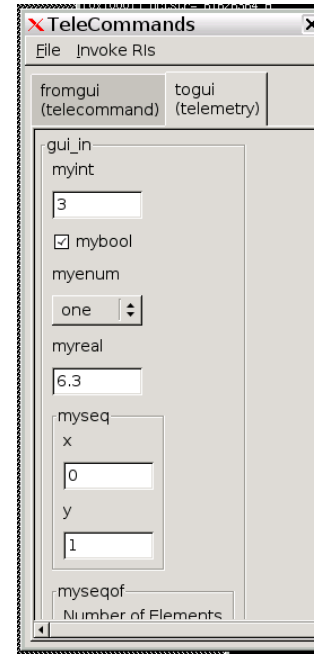


③ Generate glue code to put everything together on a real-time operating system



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)

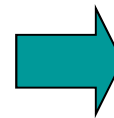


```

MY-MODULE DEFINITIONS ::= BEGIN

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

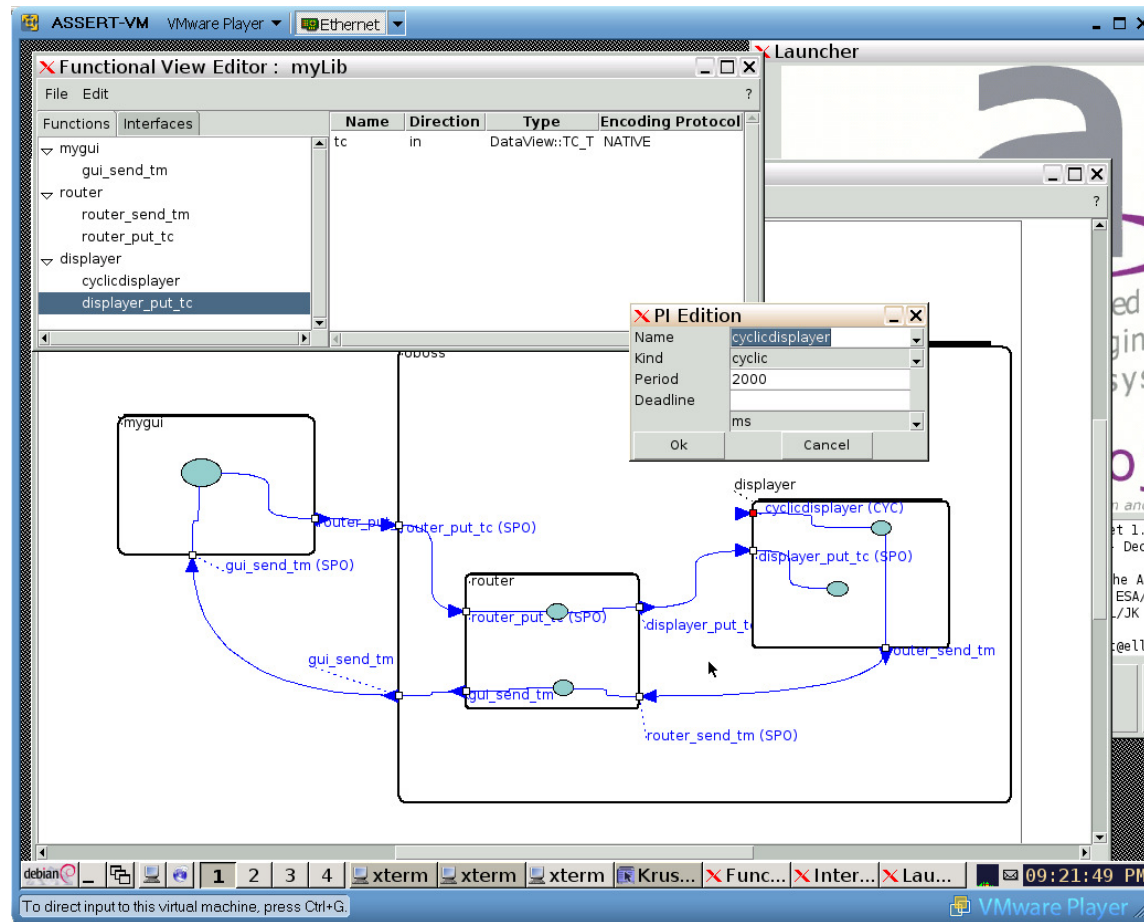
```



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

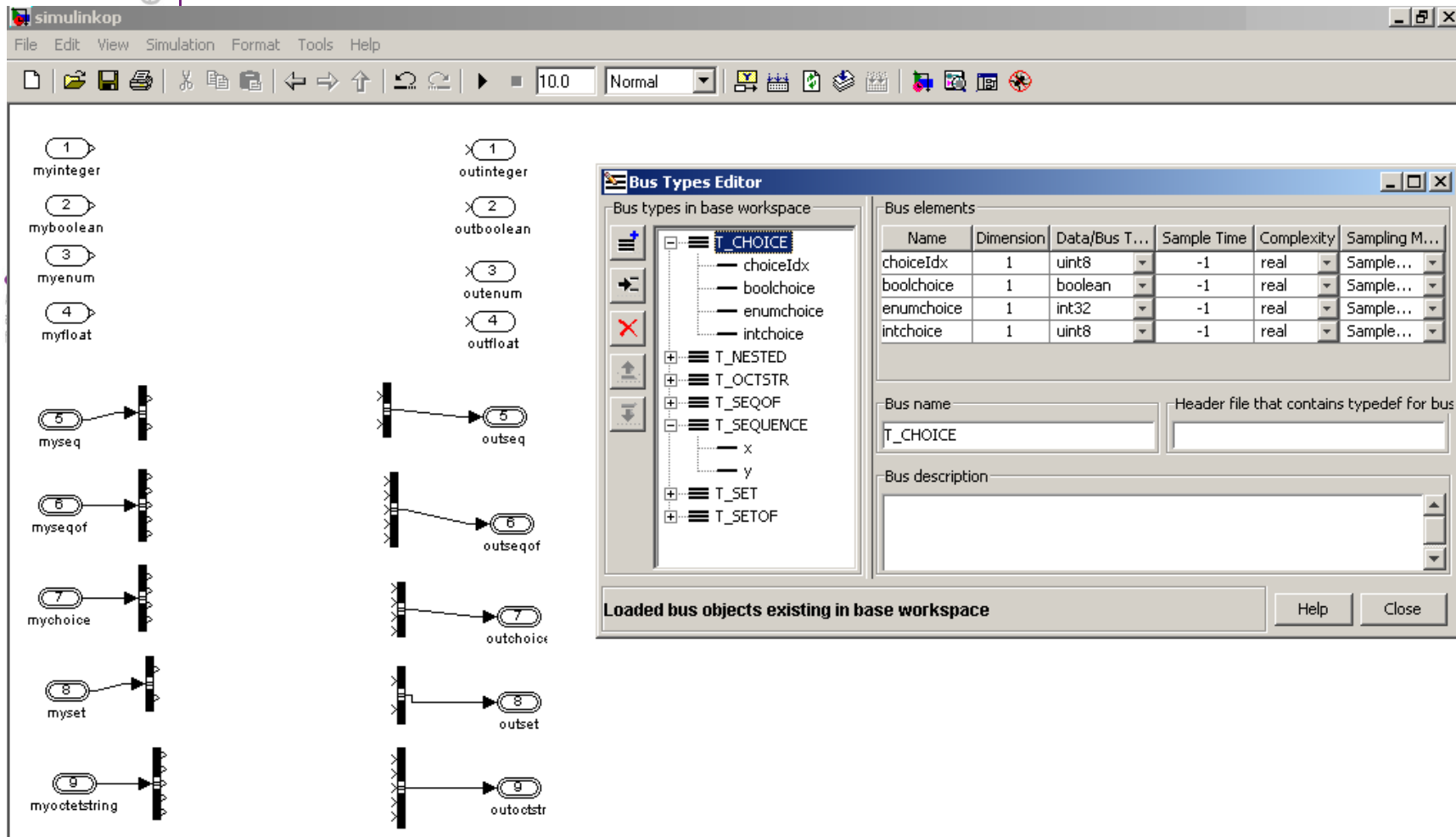
A few screenshots of the toolset (1)

- Interface and deployment view editors



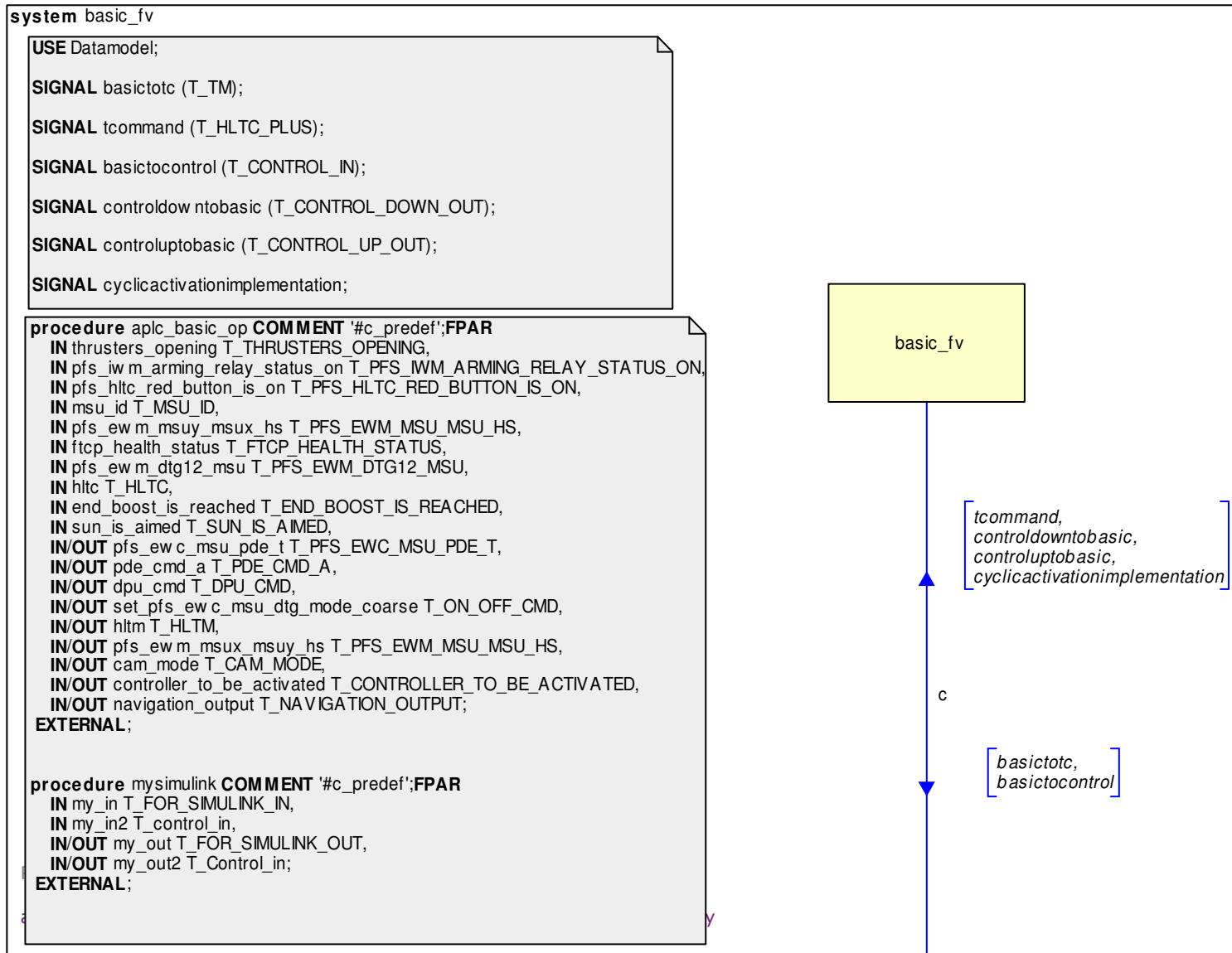
A few screenshots of the toolset (2)

- Generation of Simulink "skeletons"



A few screenshots of the toolset (3)

- Generation of SDL “skeletons”



A few screenshots of the toolset (4)

- C “skeletons”

 -- AP-Level Container: TC

SYSTEM TC
 FEATURES

CyclicActivationImplForTc : IN EVENT PORT

```
{
  Compute_Entrypoint => "CyclicActivationImplForTc";
  Assert_Properties::RCMoperation => SUBPROGRAM PFS::CyclicActivationImplForTc;
  Assert_Properties::RCMoperationKind => sporadic;
  Assert_Properties::RCMperiod => 1 ms;
};
```

TCommand : OUT EVENT PORT

```
{
  Assert_Properties::RCMoperation => SUBPROGRAM PFS::TCommand;
  Assert_Properties::RCMoperationKind => sporadic;
};
```

END TC;

SYSTEM IMPLEMENTATION TC.others

PROPERTIES

Source_Language => C;

END TC.others;

```
Listier (hpg_ed) - [C:\data\projects\ASSERT\ocarina\my_parser\output\tc\user_code.c]
Fichier Edition Options Aide
Reload Save Cut Copy Paste Undo Edit mode ? Options C++/C 1/9:1
1 /* Functions to be filled by the user (never overwritten by buildsupport tool) */
2
3 #include "user_code.h"
4
5 void cyclicactivationimplfortc()
6 {
7     /* Write your code here! */
8 }
9
10
11
```

```
Listier (hpg_ed) - [C:\data\projects\ASSERT\ocarina\my_parser\output\tc\user_code.h]
Fichier Edition Options Aide
Reload Save Cut Copy Paste Undo Edit mode ? Options C++/C 1/15:1
1 /* This file was generated automatically: DO NOT MODIFY IT ! */
2
3 /* Declaration of the functions that have to be provided by the user */
4
5 #ifndef __USER_CODE_H_tc__
6 #define __USER_CODE_H_tc__
7
8 #include "C_ASN1_Types.h"
9
10 extern void tcommand(const T_HLTC_PLUS *);
11
12 void cyclicactivationimplfortc();
13
14
15 #endif
16
```

A few screenshots of the toolset (5)

- Ada "skeletons"

```

GPS - GNAT Programming Studio (project: Default) - displayer.ads
File Edit Navigate VCS Project Build Debug Tools Window Help
C:\DOCUMENTS~1\ADMINI~1\LOCALS~1\Temp\scp39062\home\assert\oboss\labassert\output\displayer\displayer.ads
1  -- This file was generated automatically: DO NOT MODIFY IT !
2
3  -- Declaration of the provided and required interfaces
4
5  pragma style_checks (off);
6  pragma warnings (off);
7  with asnl_types;
8  use asnl_types;
9
10 package displayer is
11
12     -----
13     -- Required interface "router_send_tm"
14     -----
15     procedure router_send_tm(tm: access TM_T);
16     pragma import(C, router_send_tm, "router_send_tm");
17
18     -----
19     -- Provided interface "displayer_put_tc"
20     -----
21     procedure displayer_put_tc(tc: access TC_T);
22     pragma export(C, displayer_put_tc);
23
24     -----
25     -- Provided interface "cyclicdisplayer"
26     -----
27     procedure cyclicdisplayer;
28     pragma export(C, cyclicdisplayer);
29
30 end displayer;
31
-----
Insert Writable Unmodified 32:1
common_types.ads displayer.adb displayer.ads
Messages
[2009-01-29 14:10:20] There are several startup scripts with the same na
    
```

```

GPS - GNAT Programming Studio (project: Default) - displayer.adb
File Edit Navigate VCS Project Build Debug Tools Window Help
C:\DOCUMENTS~1\ADMINI~1\LOCALS~1\Temp\scp37421\home\assert\oboss\labassert\output\displayer\displayer.adb
1  -- User implementation of the displayer function
2  -- This file will never be overwritten once edited and modified
3  -- Only the interface of functions is regenerated (in the .ads file)
4
5  pragma style_checks (off);
6  pragma warnings (off);
7  with asnl_types;
8  use asnl_types;
9
10 package body displayer is
11
12     -----
13     -- Provided interface "displayer_put_tc"
14     -----
15     procedure displayer_put_tc(tc: access TC_T) is
16     begin
17
18         null; -- Replace "null" with your own code!
19
20     end displayer_put_tc;
21
22     -----
23     -- Provided interface "cyclicdisplayer"
24     -----
25     procedure cyclicdisplayer is
26     begin
27
28         null; -- Replace "null" with your own code!
29
30     end cyclicdisplayer;
31
32 end displayer;
33
-----
Insert Writable Unmodified 1:1
common_types.ads displayer.adb
Messages
[2009-01-29 14:10:20] There are several startup scripts with the same na
    
```

For a reliable
and scientific
approach
in system
and software
engineering



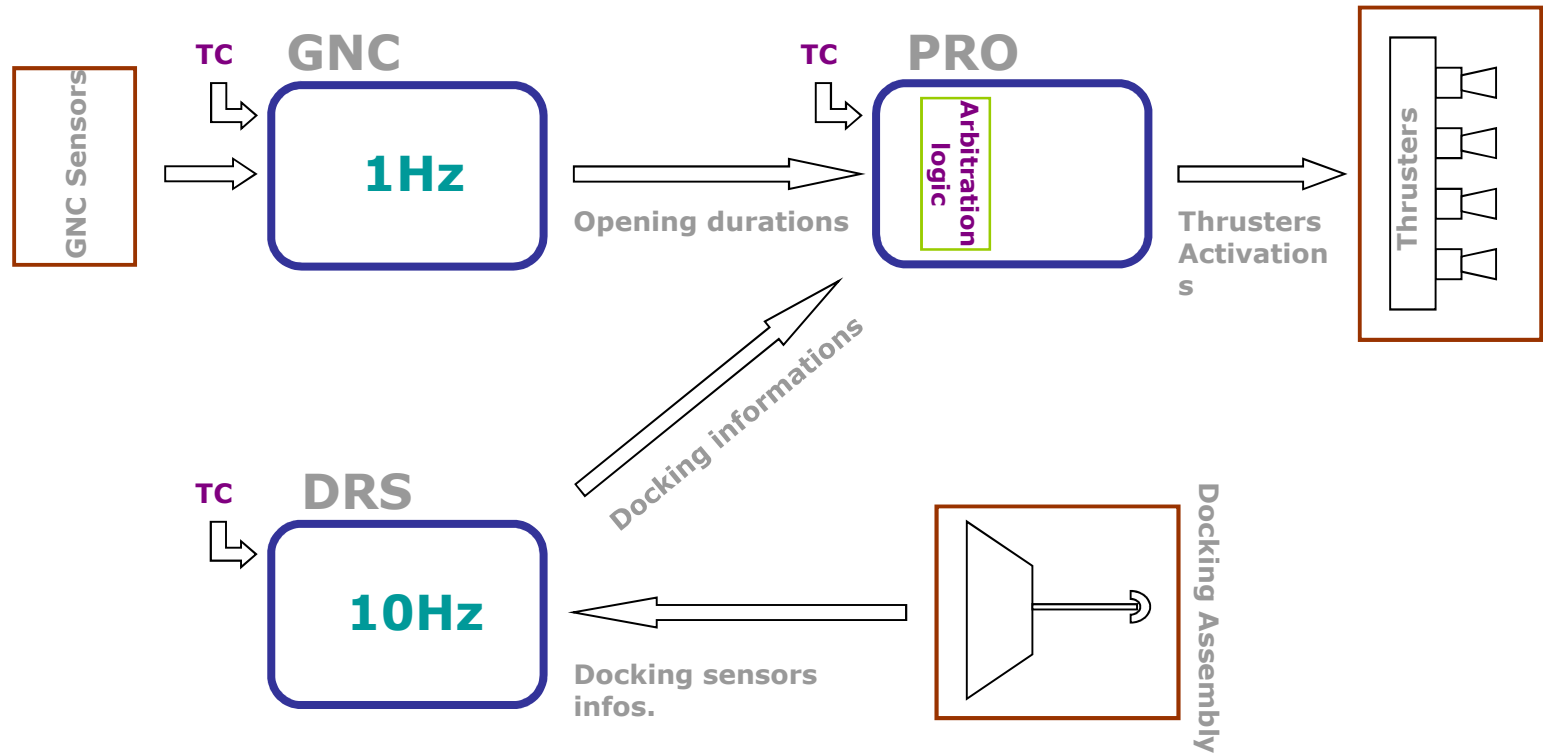
The ASSERT Process in action

Prepared by Cyril Colombo and Marie-Aude Esteve (ESA)

RST Ada Europe 2009 - 12/06/09

assert-project - Dr Eric Conquet & Maxime Perrotin - European Space Agency

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



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

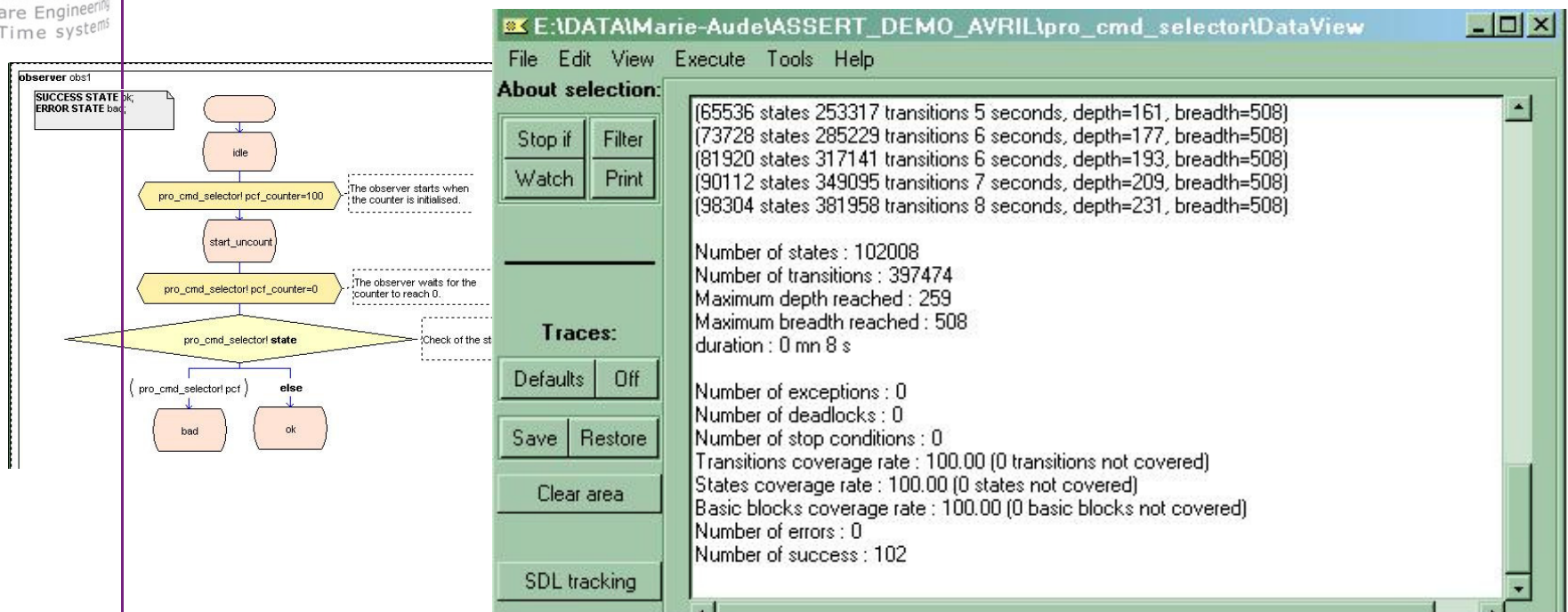
Simply defining the modeling language at Interface View level allows:

- 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

File Edit View Execute Tools Help

About selection:

Stop if Filter
Watch Print

Traces:

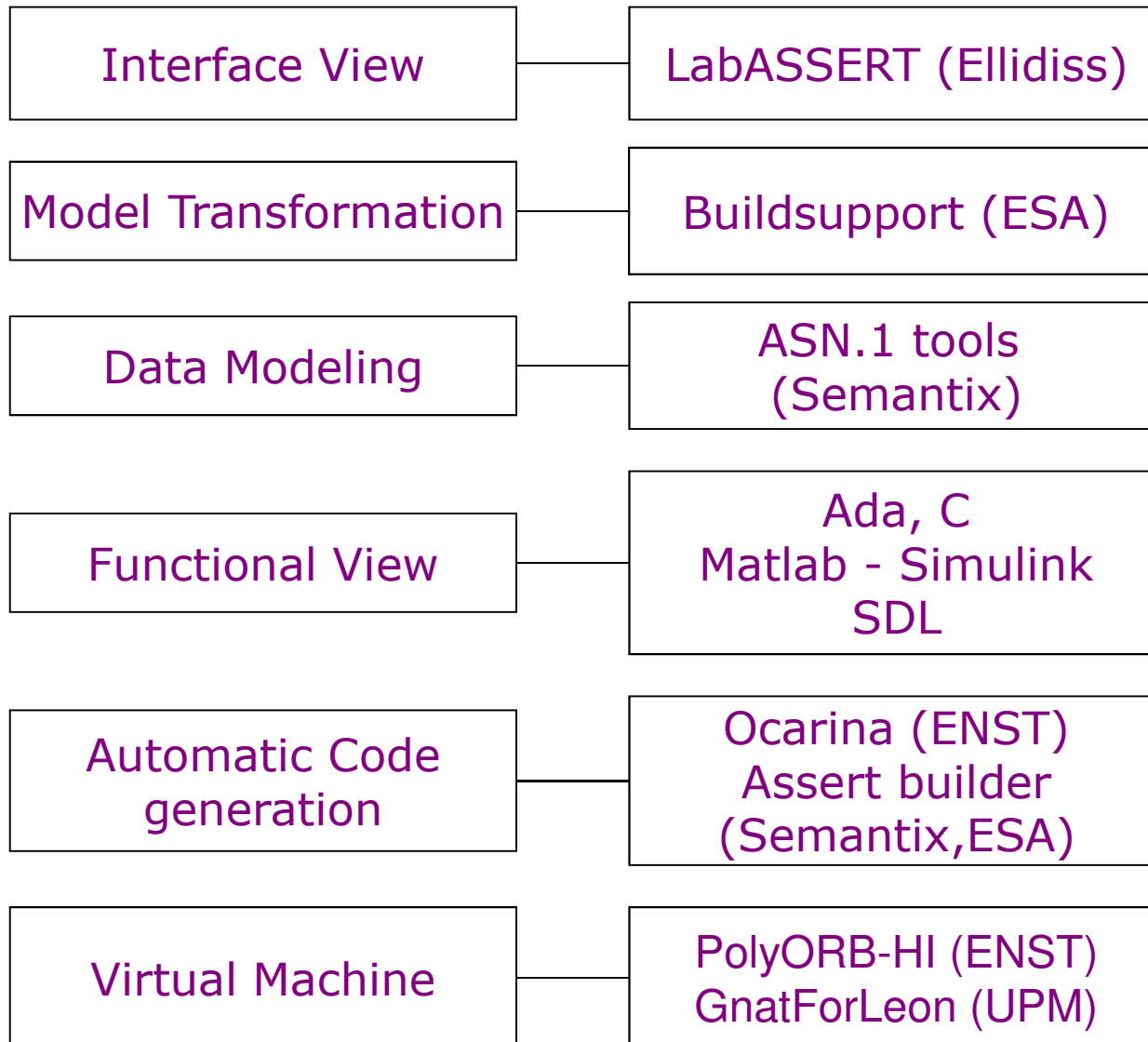
Defaults Off
Save Restore
Clear area
SDL tracking

(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

Assert toolset - Credits





assertproject

For a reliable and scientific approach in system and software engineering

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