COMPASS WORKSHOP

22 OCTOBER 2015 – ESTEC – ESCAPE DANCE HALL

Alessandro Cimatti, Fondazione Bruno Kessler

Joost-Pieter Katoen, Rheinisch-Westfalische Technische Hochschule Aachen

Marcel Verhoef, European Space Agency (TEC-SWE)

Agenda

1. Workshop scope and purpose
2. Program for today
3. ESA vision on the future of COMPASS*

Workshop scope and purpose

COMPASS provides a wide range of techniques for the design and analysis of system safety and reliability, whose applicability has already been demonstrated in several case studies.

The objective of the workshop is to understand how to bring these promising results to higher technology readiness levels.

We would like to interact with the audience to identify hurdles of introducing COMPASS in industrial practice, and discuss and explore ways these hurdles can be taken or circumvented, with potential solutions both in technology as well as process.

Program for today (1)

1. Vision for COMPASS Future (08:30 – 09:30)
   - 08:30 ESA Vision - Marcel Verhoef (ESA/ESTEC)
   - 08:50 FBK Vision - Alessandro Cimatti (FBK)
   - 09:10 RWTH Vision - Joost-Pieter Katoen (RWTH)

2. Current state of COMPASS (09:30 – 10:30)
   - 09:30 Error modelling in COMPASS - Harold Bruinjes (RWTH)
   - 09:45 Safety assessment in COMPASS - Marco Bozzanno (FBK)
   - 10:00 Contract based verification of AADL – Stefano Tonetta (FBK)
   - 10:15 Fault propagation modelling and analysis through TFPG Benjamin Bittner (FBK)

3. Coffee break (10h30 – 11h00)

Program for today (2)

4. Open Challenges and Future Directions (11:00 – 12:00)
    - 11:00 The Marriage of COMPASS and MILS
      Rance De Long (The Open Group)
    - 11:10 COMPASS in the D-MILS project - an experience report
      Harald Ruess (Fortiss GmbH)
    - 11:20 Model Repair in Systems Design
      Panagiotis Katsaros (Aristotle University of Thessaloniki)
    - 11:30 Model based safety assessment of space operations – toward integration of failure analysis of system and operation - Jean-Paul Blanquart (Airbus DS)
    - 11:40 A Statistical Approach for Timed Reachability in AADL Models
      Harold Bruintjes (RWTH)
    - 11:50 The FoReVer MBSE solution for system composition correctness analysis
      Silvia Mazzini (Intecs)
5. Open discussion (12:00 – 12:45)
6. Round-up and closing remarks (12:45 – 13:00)

Context: system dependability (1)

- Spacecraft design must ensure that mission objectives are met
- Resilience to faults is therefore an important design driver
- System dependability is notoriously difficult to assess
    - Fault likelihood is hard to quantify or estimate
    - Number of possible (or combined) fault scenarios explodes
    - Acceptable residual risk levels (ALARP) are hard to define
    - It is hard to demonstrate that these ALARP levels are actually met
- Wider scope: failure and anomaly management: where and how to handle anomalies that do not (yet) lead to failure
- Human factor aspect: “Fear-based” analysis $\rightarrow$ *there is no limit to fear*

“Something that can happen with a chance of one-in-a-million, will happen nine-out-of-ten times.” [D. Adams]

Context: system dependability (2)

- Current challenges:
  - FDIR not considered as a subsystem hence no-one is responsible
  - FDIR is bottom-up activity driven by (domain specific) RAMS analysis
  - Hazard analysis is typically not performed [systematically]
  - System impact is considered (too) late in the design
  - Complexity usually converges in SW (and AOCS) design
  - Completeness difficult to assess (i.e. during reviews)
  - Fitness-for-purpose hard to demonstrate (during V&V)
- Significant risk to cost and impact to schedule
- Our proposed antidote:
  - Design automation to manage consistency and complexity: *model-based system and software engineering*
  - Move FDIR design and analysis forward in life cycle *handbook and reference architectures*

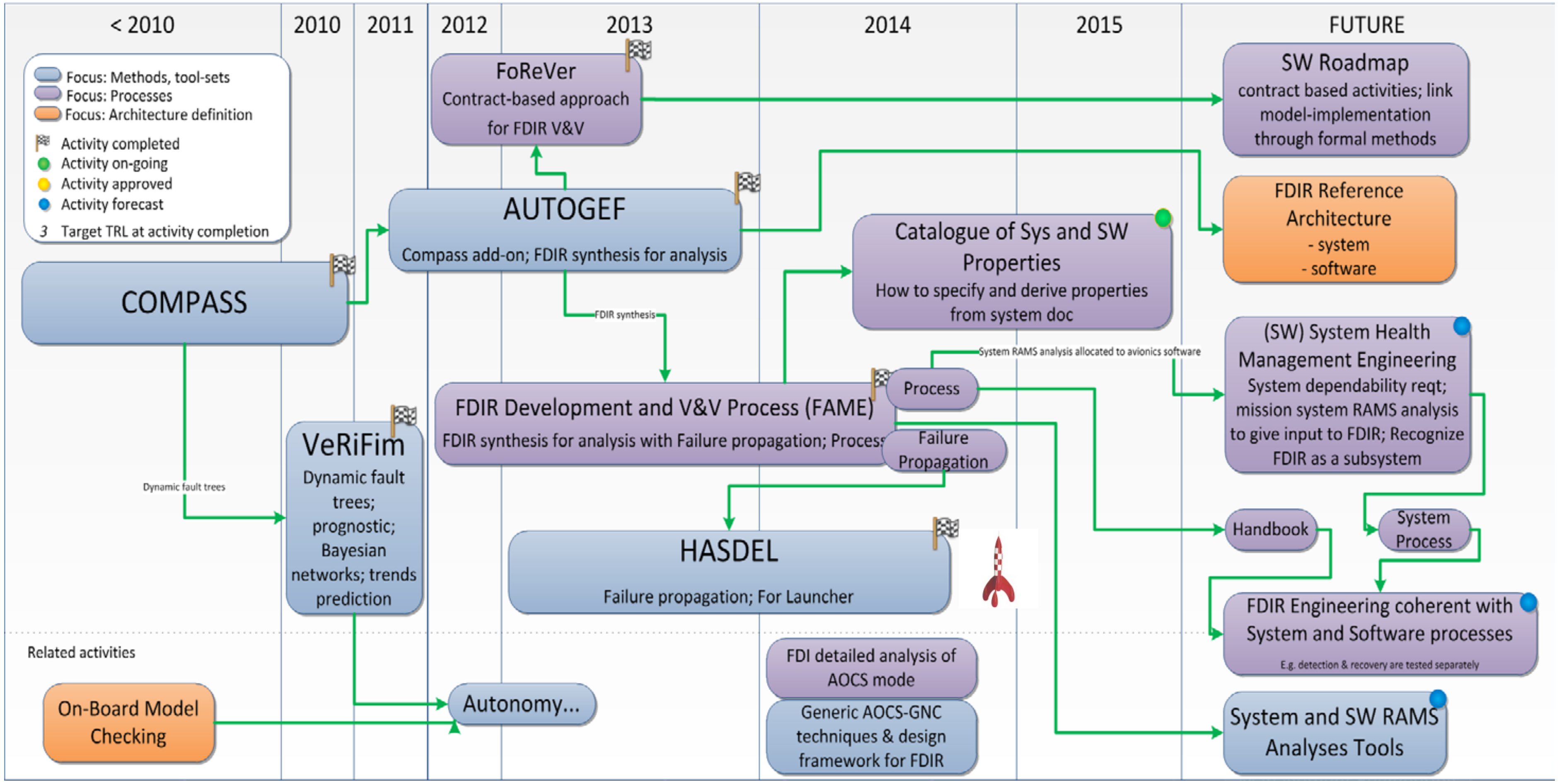
Towards model-based dependability engineering (FDIR as carrier)

➔ Support FDIR process with a <u>model based approach</u>
- Tools are required to manage this complexity
- Support early analysis and to enable reuse
- Fully embedded in the engineering process

➔ <u>Create an FDIR community:</u>
- Exchange experiences, identify best practices, identify potential tool and process needs and improvements
- Consolidate knowledge: towards FDIR handbook

➔ <u>Establish FDIR as an engineering discipline</u>
- Unification across industry to further support interoperability (through SAVOIR and ECSS)
- Provide means to better support ISVV early in the life cycle

Roadmap of TEC-SW activities



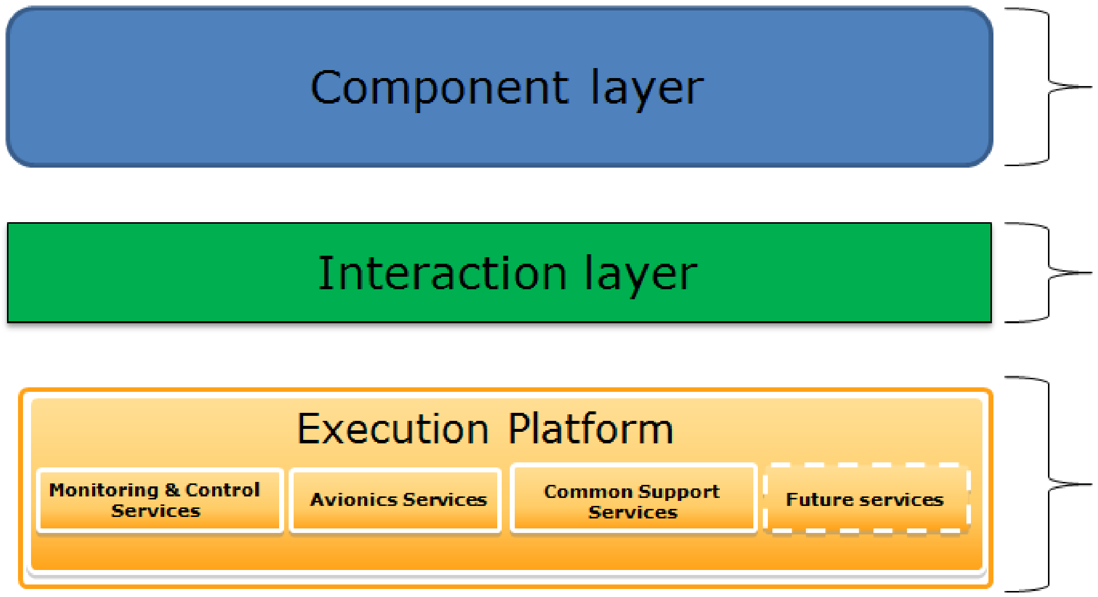
< 2010
2010
2011
2012
2013
2014
2015
FUTURE
Focus: Methods, tool-sets
Focus: Processes
Focus: Architecture definition
Activity completed
Activity on-going
Activity approved
Activity forecast
3 Target TRL at activity completion
FoReVer
Contract-based approach for FDIR V&V
AUTOGEF
Compass add-on; FDIR synthesis for analysis
COMPASS
FDIR synthesis
Dynamic fault trees
VeRiFim
Dynamic fault trees; prognostic; Bayesian networks; trends prediction
FDIR Development and V&V Process (FAME)
FDIR synthesis for analysis with Failure propagation; Process
Process
Failure Propagation
Catalogue of Sys and SW Properties
How to specify and derive properties from system doc
System RAMS analysis allocated to avionics software
HASDEL
Failure propagation; For Launcher
SW Roadmap
contract based activities; link model-implementation through formal methods
FDIR Reference Architecture
- system
- software
(SW) System Health Management Engineering
System dependability reqt; mission system RAMS analysis to give input to FDIR; Recognize FDIR as a subsystem
Handbook
System Process
FDIR Engineering coherent with System and Software processes
E.g. detection & recovery are tested separately
Related activities
On-Board Model Checking
Autonomy…
FDI detailed analysis of AOCS mode
Generic AOCS-GNC techniques & design framework for FDIR
System and SW RAMS Analyses Tools

ESA UNCLASSIFIED – For Internal Use

MBSSE at ESA

1. the use of ontologies and formal verification techniques to create a *catalogue of system and software properties*, which forms the basis for correct-by-construction software synthesis and re-use of requirements
2. The definition of *reference architectures* and the use of architecture description languages to *explore system resilience by analyzing explicit fault models* using model checking
3. improve the production of flight software by *integrating well-founded formal technologies* in those parts of the engineering chain where their benefit is clear and the gain is significant
4. the use of time and space partitioning kernels to implement mixed criticality applications on multi-core processors, supported by formal analysis techniques to study *deterministic schedulability in the presence of (WCET) uncertainty*

SAVOIR OSRA (1)

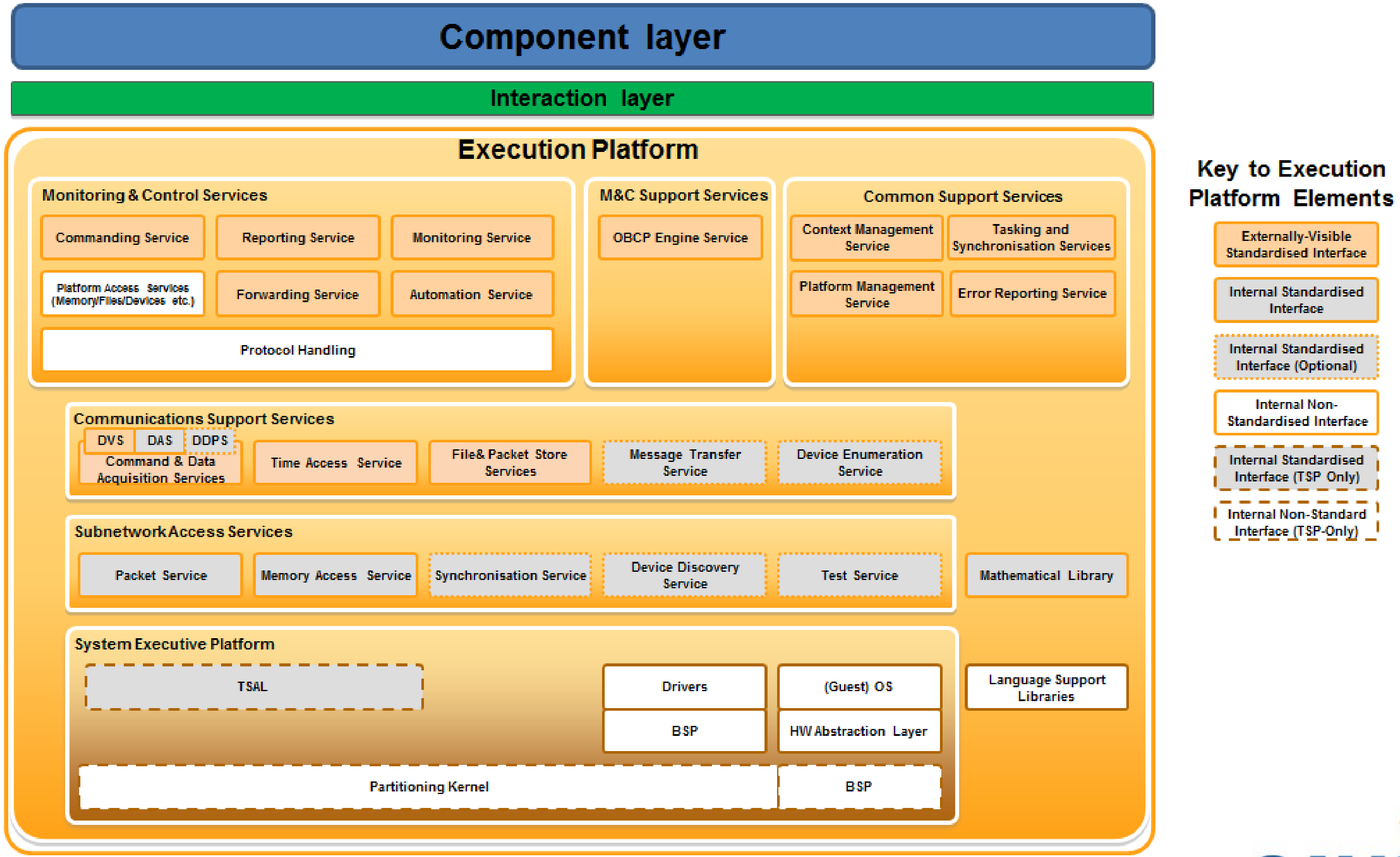


Application software is designed by specifying "software components". Non-functional aspects are only "declaratively specified". Their implementation is left to the automation capabilities of the design environment

Automatically generated layer that implements the non-functional concerns declared by components (e.g., tasking, synchronization)

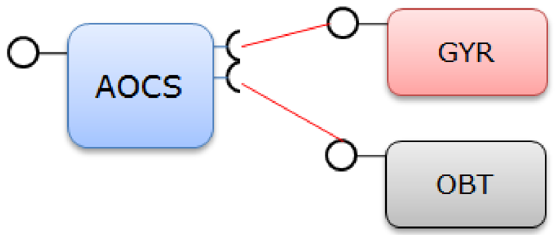
Provides services to the component layer (e.g., context saving) and interaction layer (e.g., tasking primitives, access to devices, message transfer)

SAVOIR OSRA (2)

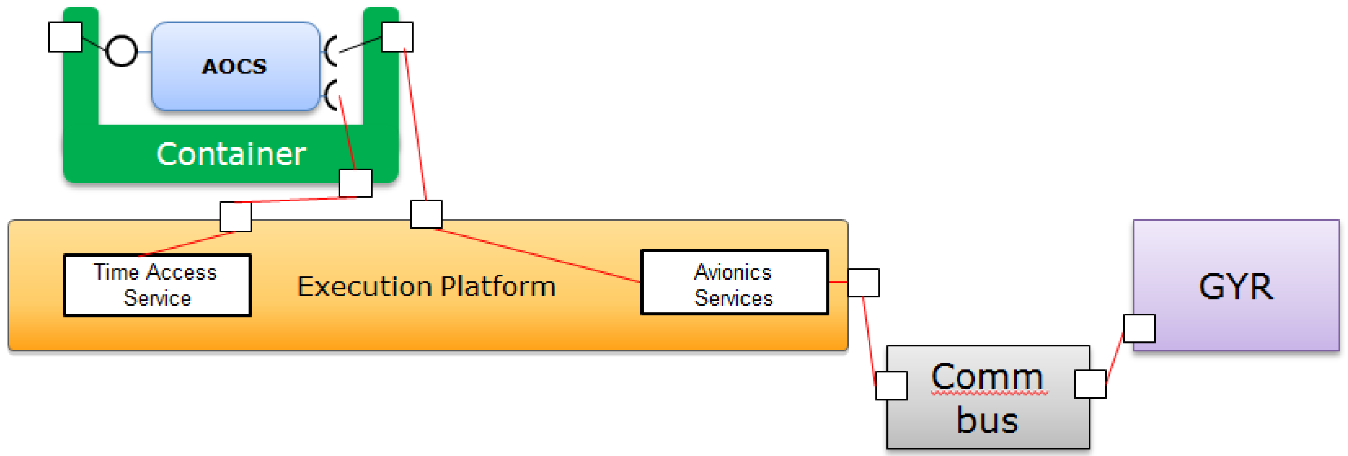


Component layer
Interaction layer
Execution Platform
Monitoring & Control Services
Commanding Service
Reporting Service
Monitoring Service
Platform Access Services (Memory/Files/Devices etc.)
Forwarding Service
Automation Service
Protocol Handling
M&C Support Services
OBCP Engine Service
Common Support Services
Context Management Service
Tasking and Synchronisation Services
Platform Management Service
Error Reporting Service
Communications Support Services
DVS
DAS
DDPS
Command & Data Acquisition Services
Time Access Service
File& Packet Store Services
Message Transfer Service
Device Enumeration Service
Subnetwork Access Services
Packet Service
Memory Access Service
Synchronisation Service
Device Discovery Service
Test Service
Mathematical Library
System Executive Platform
TSAL
Drivers
(Guest) OS
BSP
HW Abstraction Layer
Language Support Libraries
Partitioning Kernel
BSP
Key to Execution Platform Elements
Externally-Visible Standardised Interface
Internal Standardised Interface
Internal Standardised Interface (Optional)
Internal Non-Standardised Interface
Internal Standardised Interface (TSP Only)
Internal Non-Standard Interface (TSP-Only)

SAVOIR OSRA (3)



Code is automatically generated to interface the component with the required execution platform services or to use avionics services to access the equipment *(even if the execution platform itself is not necessarily based on components)*

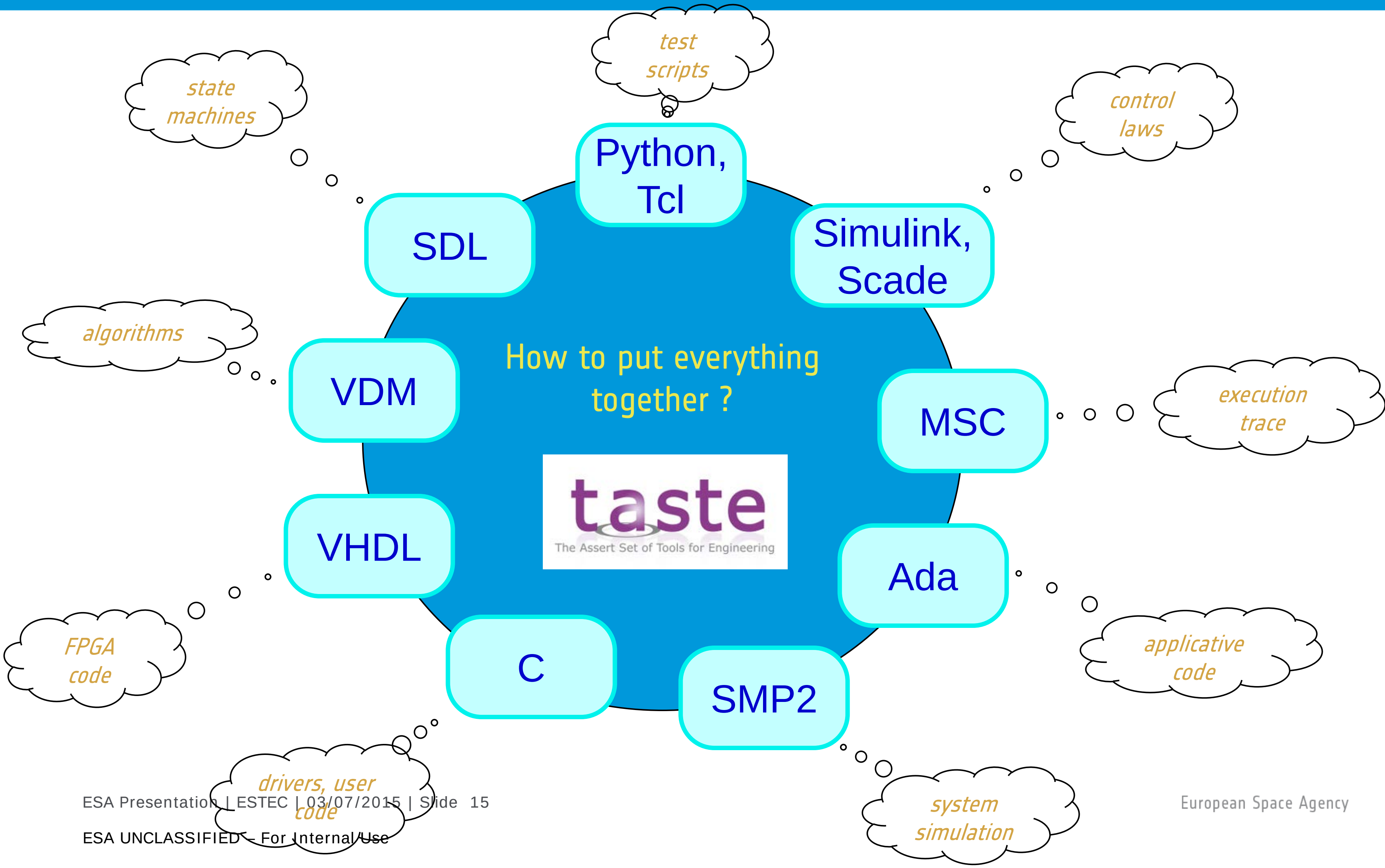


The TASTE toolset (1)

- consolidated result from (and continued development of) the ESA-led EU-FP6 ASSERT project: **T**he **A**SSERT **S**et of **T**ools for **E**ngineering
    - open-source tool suite for rigorous software engineering
    - aimed at development of heterogeneous embedded systems
    - focus on (but not limited to) space on-board software
    - based on mature (formal) notations with long term support
    - model-centric development with high levels of automation
    - seamless interoperability offers DSL-like approach
    - model synthesis towards wide range of target platforms
    - robust tools maintained by active (but small) community

For more information see http://taste.tuxfamily.org/

Dealing with heterogeneity



The TASTE toolset (2)

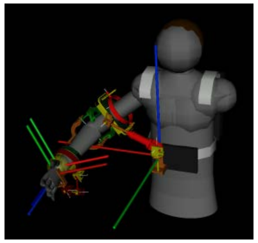
The main elements of TASTE are:

1. *Abstract Syntax Notation One* (ASN.1, ITU X.680-X.693)
   - used to describe (abstract) data types (i.e. TC and TM)
   - orthogonal encoding rules for physical representation
   - code generation and run-time support for C and Ada
   - generation of interface documentation and test sets

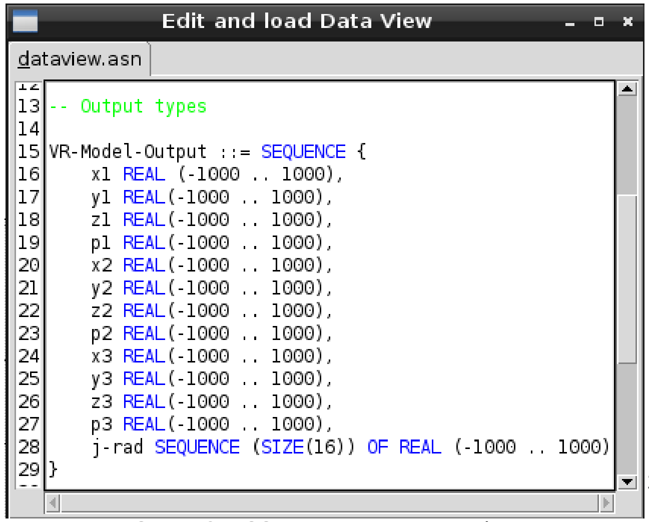
2. *Architecture Analysis and Design Language* (AADL, SAE AS 5506B)
   - extensible formal textual and graphical notation, used to describe the system logical and physical architecture
   - used to capture avionics hardware components, their communication interfaces and deployment of software artifacts
   - generation of high-integrity (SPARK) Ada code

ASN.1 to describe interfaces

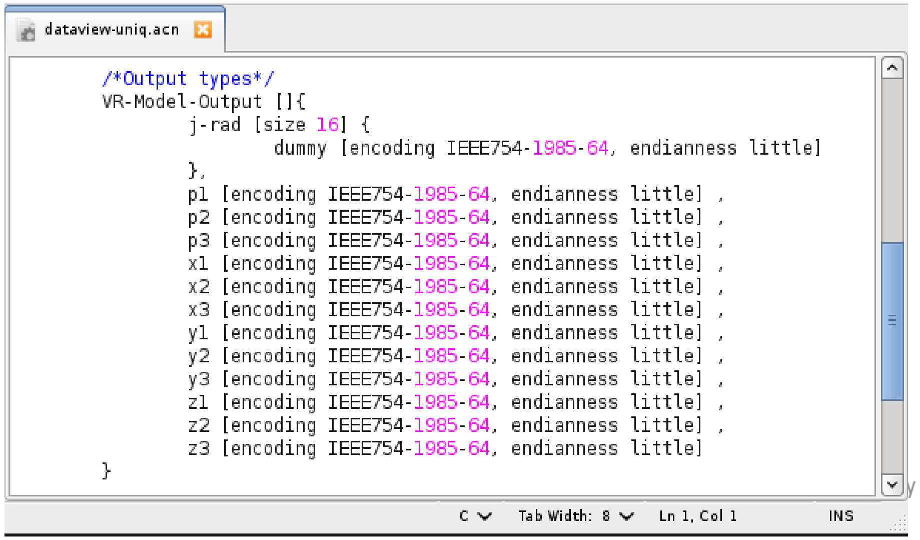
1. A simple notation to describe software and hardware interfaces
2. Our tools generate code for embedded systems (no malloc, no system call, support for C and [Spark] Ada)



+



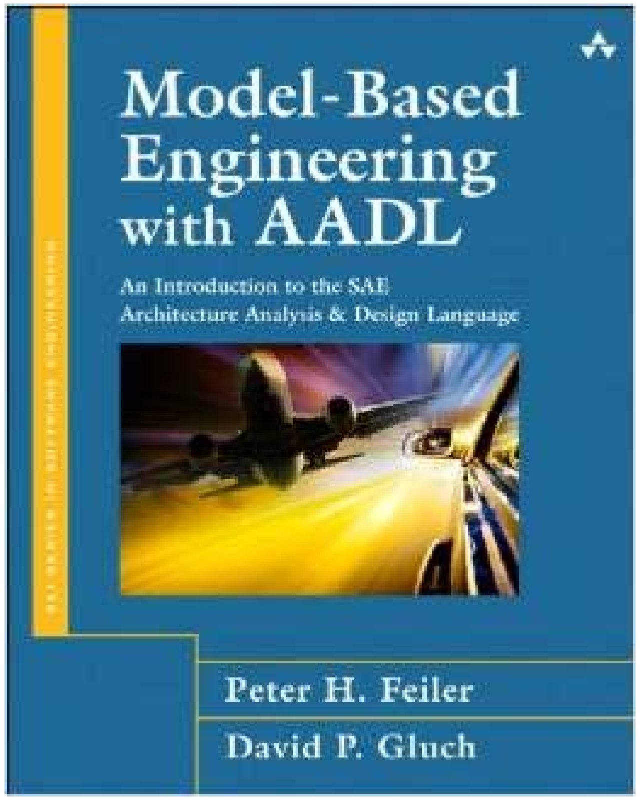
```
-- Output types

VR-Model-Output ::= SEQUENCE {
    x1 REAL (-1000 .. 1000),
    y1 REAL(-1000 .. 1000),
    z1 REAL(-1000 .. 1000),
    p1 REAL(-1000 .. 1000),
    x2 REAL(-1000 .. 1000),
    y2 REAL(-1000 .. 1000),
    z2 REAL(-1000 .. 1000),
    p2 REAL(-1000 .. 1000),
    x3 REAL(-1000 .. 1000),
    y3 REAL(-1000 .. 1000),
    z3 REAL(-1000 .. 1000),
    p3 REAL(-1000 .. 1000),
    j-rad SEQUENCE (SIZE(16)) OF REAL (-1000 .. 1000)
}
```



```
        /*Output types*/
        VR-Model-Output []{
                j-rad [size 16] {
                        dummy [encoding IEEE754-1985-64, endianness little]
                },
                p1 [encoding IEEE754-1985-64, endianness little] ,
                p2 [encoding IEEE754-1985-64, endianness little] ,
                p3 [encoding IEEE754-1985-64, endianness little] ,
                x1 [encoding IEEE754-1985-64, endianness little] ,
                x2 [encoding IEEE754-1985-64, endianness little] ,
                x3 [encoding IEEE754-1985-64, endianness little] ,
                y1 [encoding IEEE754-1985-64, endianness little] ,
                y2 [encoding IEEE754-1985-64, endianness little] ,
                y3 [encoding IEEE754-1985-64, endianness little] ,
                z1 [encoding IEEE754-1985-64, endianness little] ,
                z2 [encoding IEEE754-1985-64, endianness little] ,
                z3 [encoding IEEE754-1985-64, endianness little]
        }
```

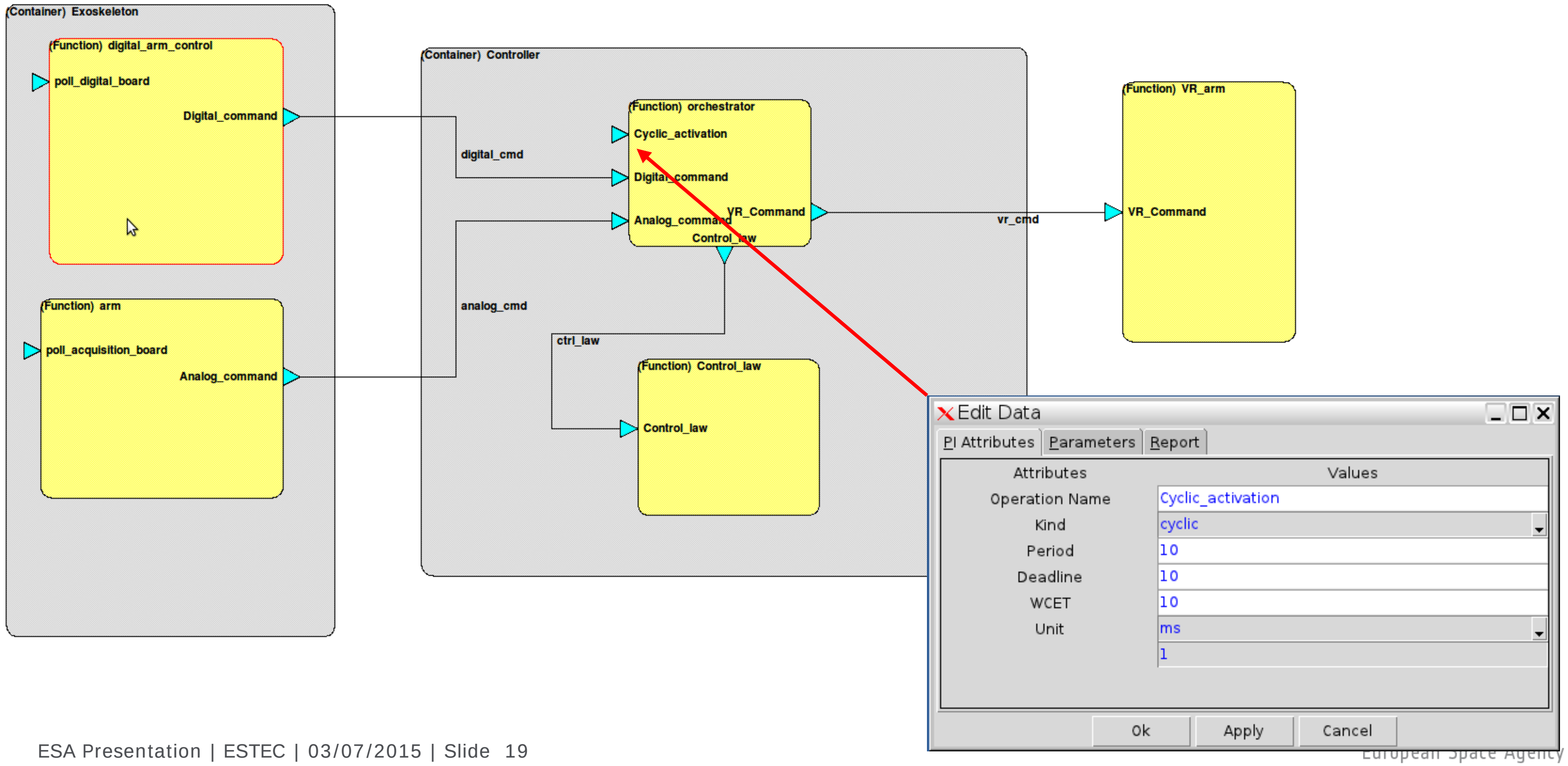
Some words on AADL

- Roots in avionics: META-H (Honeywell [Vestal])
- Now coordinated by CMU-SEI
- Standard sanctioned by SAE
- Used by ARINC
- Well-defined formal language (stronger than SysML, UML-MARTE)
- Extensible
- Active (research) community
- Tool support (Eclipse: OSATE2)
- http://www.aadl.info


Model-Based Engineering with AADL
An Introduction to the SAE Architecture Analysis & Design Language
Peter H. Feiler
David P. Gluch

System architecture modelling

Graphical approach to unambiguously capture the system architecture and its real-time properties



Attributes	Values
Operation Name	Cyclic_activation
Kind	cyclic
Period	10
Deadline	10
WCET	10
Unit	ms
	1

The TASTE toolset (3)

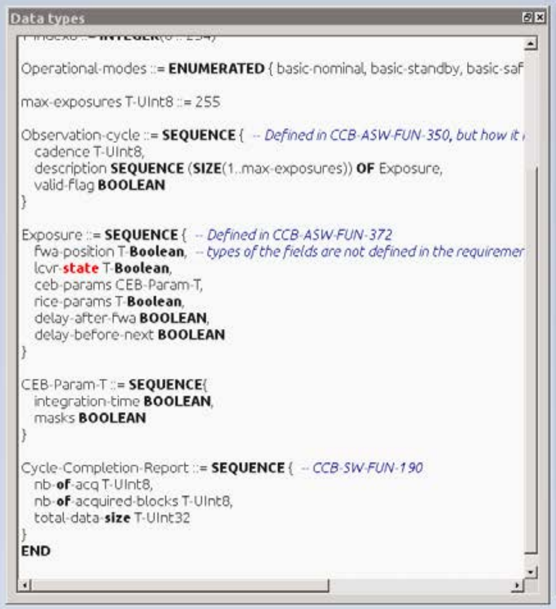
The main elements of TASTE are (continued):

3. *Specification and Description Language* (SDL, ITU-T Rec. Z.100)
    - formal language to describe *state machines*
    - graphical and textual notation, native support for ASN.1 types
    - model evolution visualized as *message sequence chart* (Z.150)
    - record and playback useful for analysis and testing
    - code generation to (SPARK) Ada

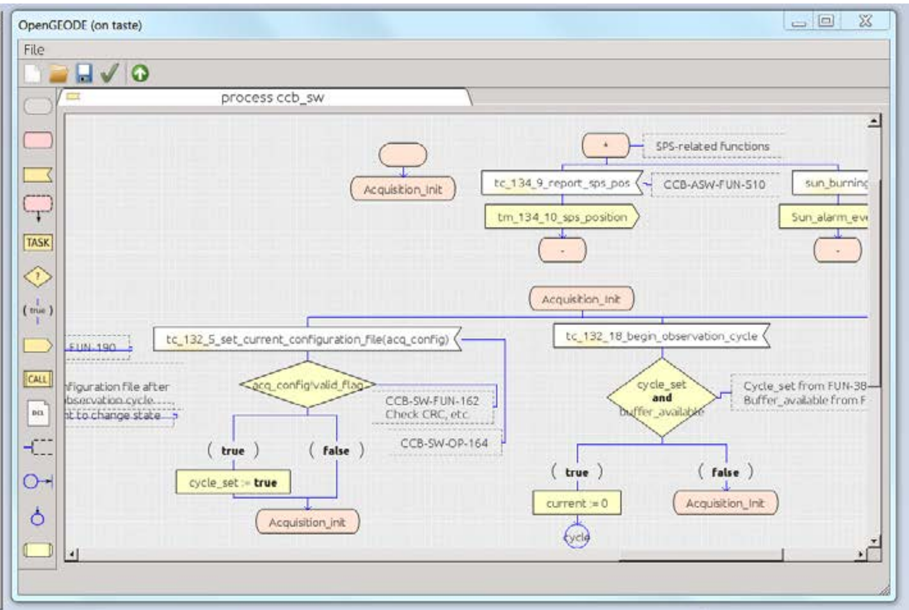
4. *build automation and automatic target deployment*
    - Linux and SMP2 simulation environments
    - RTEMS and Xenomai on (virtualized) QEMU or TSIM
    - RTEMS or Ada-Ravenscar run-time on target hardware

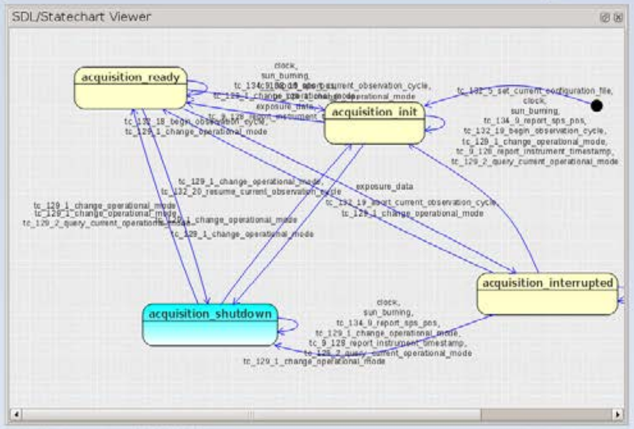
(caveat: also support for Simulink, SCADE, VHDL, Ada, C)

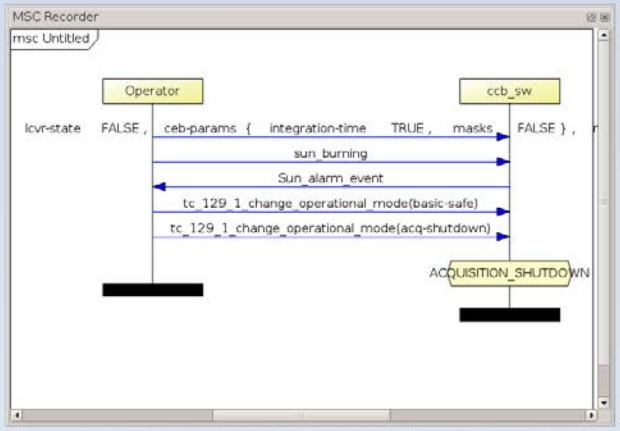
Modelling state machines

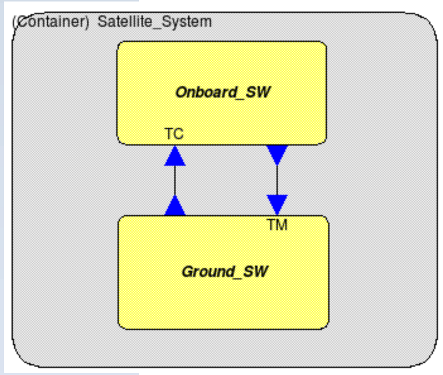


```
Operational-modes ::= ENUMERATED { basic-nominal, basic-standby, basic-saf

max-exposures T-UInt8 ::= 255

Observation-cycle ::= SEQUENCE {  -- Defined in CCB-ASW-FUN-350, but how it
   cadence T-UInt8,
   description SEQUENCE (SIZE(1..max-exposures)) OF Exposure,
   valid-flag BOOLEAN
}

Exposure ::= SEQUENCE {  -- Defined in CCB-ASW-FUN-372
   fwa-position T-Boolean,  -- types of the fields are not defined in the requiremer
   lcvr-state T-Boolean,
   ceb-params CEB-Param-T,
   rice-params T-Boolean,
   delay-after-fwa BOOLEAN,
   delay-before-next BOOLEAN
}

CEB-Param-T ::= SEQUENCE{
   integration-time BOOLEAN,
   masks BOOLEAN
}

Cycle-Completion-Report ::= SEQUENCE {  -- CCB-SW-FUN-190
   nb-of-acq T-UInt8,
   nb-of-acquired-blocks T-UInt8,
   total-data-size T-UInt32
}
END
```






MSC Recorder
msc Untitled
Operator
ccb_sw
lcvr-state FALSE , ceb-params { integration-time TRUE , masks FALSE } ,
sun_burning
Sun_alarm_event
tc_129_1_change_operational_mode(basic-safe)
tc_129_1_change_operational_mode(acq-shutdown)
ACQUISITION_SHUTDOWN



The Assert Set of Tools for Engineering

The TASTE toolset (4)

The TASTE development process consist of the following steps:

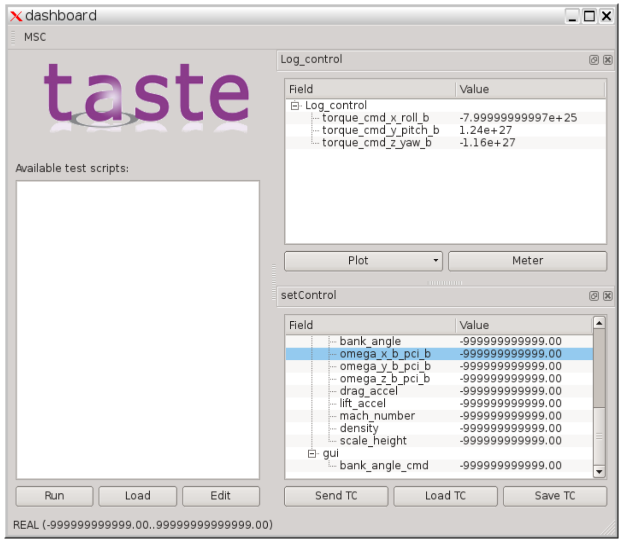
1. describe the system logical architecture (AADL) and interfaces (ASN.1)
2. describe the system behavior (SDL)
3. describe the deployment of functionality on the avionics (AADL)
4. generate code, build the system and download on simulator or target
5. monitor and interact with the system at run-time (test execution)

TASTE allows complementary analysis (re-)using the AADL model

- Schedulability analysis using MAST and CHEDDAR tools
- Use AADL extension capability to specify explicit fault behavior using *System-Level Integrated Modelling* language (SLIM) which can be verified using the (TASTE compatible) FAME tools (nuSMV)

Support for interactive V&V

Generate additional code to help users test their system (real-time monitoring and interaction with the binary)



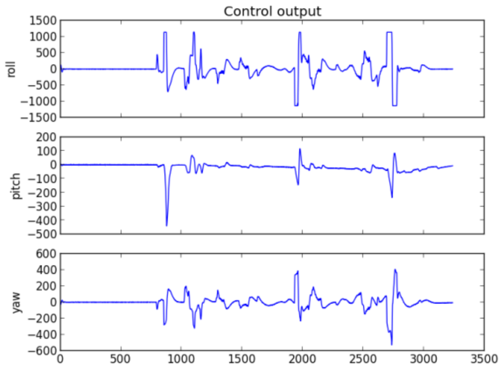
Log_control

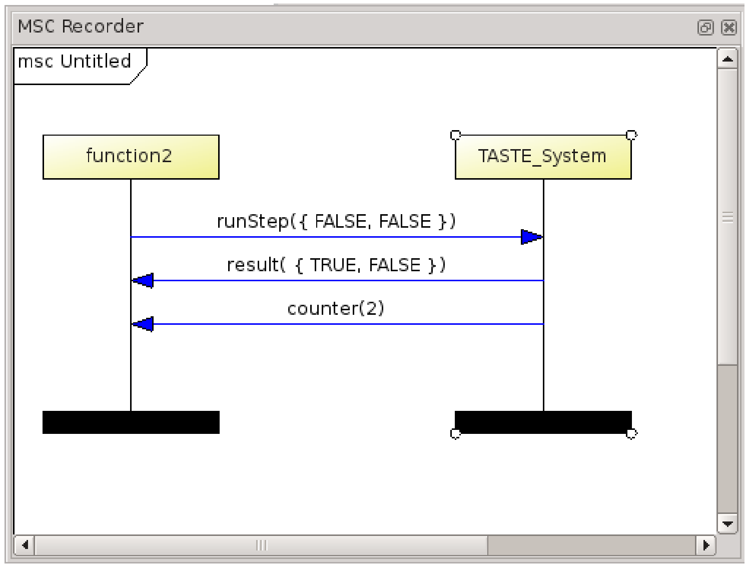
Field	Value
Log_control	
torque_cmd_x_roll_b	-7.99999999997e+25
torque_cmd_y_pitch_b	1.24e+27
torque_cmd_z_yaw_b	-1.16e+27

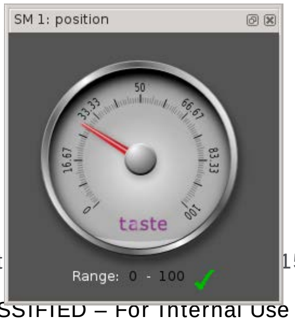
setControl

Field	Value
bank_angle	-999999999999.00
omega_x_b_pci_b	-999999999999.00
omega_y_b_pci_b	-999999999999.00
omega_z_b_pci_b	-999999999999.00
drag_accel	-999999999999.00
lift_accel	-999999999999.00
mach_number	-999999999999.00
density	-999999999999.00
scale_height	-999999999999.00
gui	
bank_angle_cmd	-999999999999.00

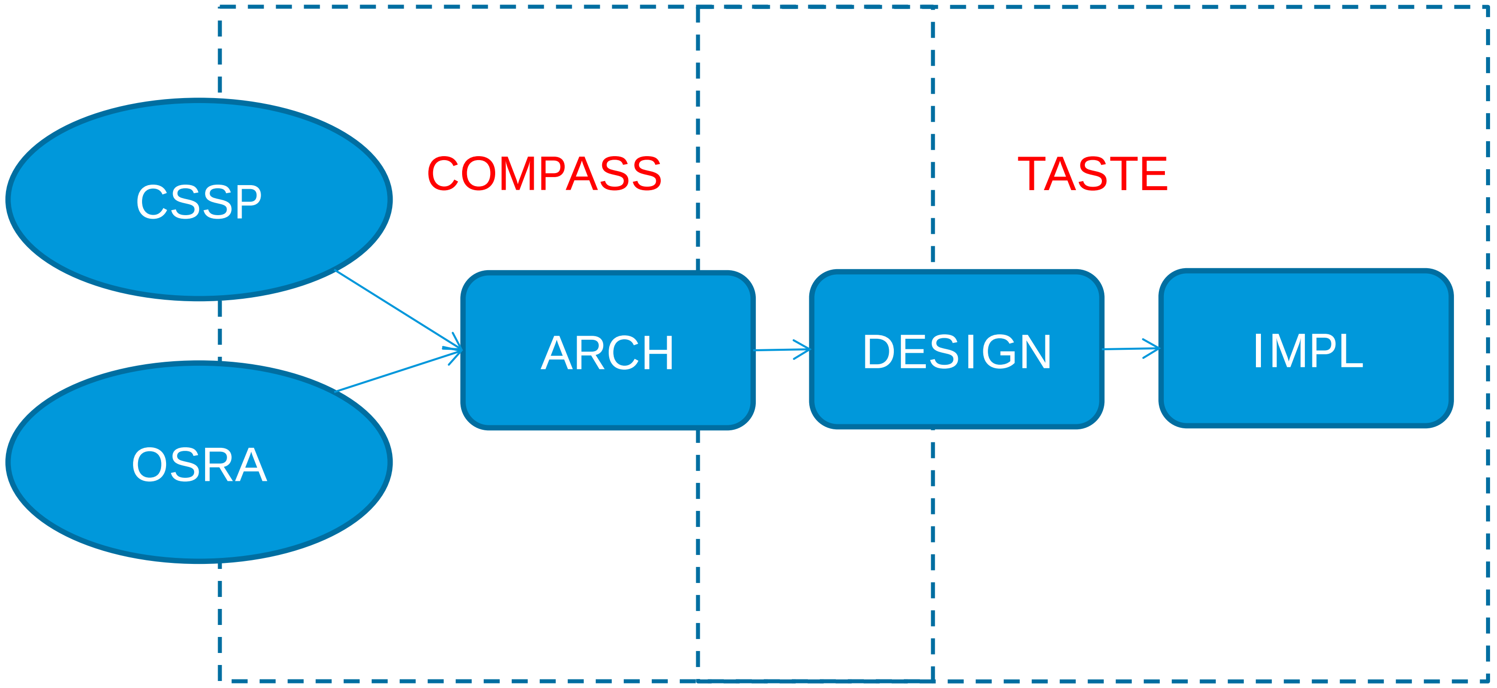
REAL (-999999999999.00..99999999999999.00)







Long-term vision: software factory



Short term future: COMPASS 3.0

Objective of this new activity is to:

- integrate, harmonize and update selected features from previous projects (COMPASS, AUTOGEF, FAME and HASDEL)
- Upgrading the verification engines to latest SothA
- Improvements to the user interface
- Create stable basis for use and further future developments
- Improve dissemination (tutorials, examples)
- Define a roadmap for COMPASS*

Aims to further establish and support the COMPASS community at large