TASTE

6.01.2025

Jérôme Legrand

Contents

Contents

1 Introduction

Ellidiss Technologies

- Ellidiss Technologies propose des méthodes et outils pour le développement des systèmes critiques à forte concentration de logiciel.

- Les trois activités de la société sont :
  - La maintenance au long cours et le support technique de ses outils en opération dans des programmes industriels majeurs tels que les A340, A380 et A350 et l'Eurofighter Typhoon.
  - Le développement de nouveaux outils de modélisation et d'analyse supportant des standards industriels tels que HOOD, AADL ou SysML.
  - La participation à des projets collaboratifs de recherche ou industriels.

Objectives of this course

- Present an industrial/Research software design methodology :
  - Interesting concepts
  - Implementation not up to these concepts
- Prepare the tutorial/practical work

Contents

2 TASTE Overview

What is TASTE

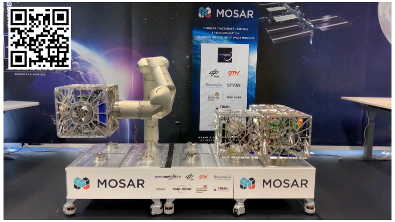
- A tool-chain targeting heterogeneous systems, using model-based development.
- A process supporting the creation of systems, using formal models and automatic code generation.
- A laboratory platform experimenting with safety-critical SW technologies, based on open-source, freely accessible solutions.
- https://taste.tools and https://gitrepos.estec.esa.int/taste/
- TASTE's lineage can be traced back to the EU/FP6 ASSERT project : an effort led by the European Space Agency back in 2005, whose purpose was to bring true, formal models-based Engineering into the way we develop space SW.

ERGO Project

- European Robotic Goal-Oriented Autonomous Controller.
- Objective : deliver an advanced yet flexible space autonomous software framework/ system suitable for single and/or collaborative space robotic means/missions (orbital and surface rovers).
- https://www.h2020-ergo.eu/

MOSAR Project

- Modular and Re-Configurable Spacecraft
- Existing commercial satellites and space platforms are traditionally the result of a highly customized monolithic design with very limited or no capability of servicing and maintenance.
- Number of ageing satellites is growing rapidly. There is currently no available technology, and as a result typically no plan for maintenance and the ageing satellites are simply dismissed and left in orbit as a space debris.
- Objective : developing a ground demonstrator for on-orbit modular and reconfigurable satellites.
- https://www.h2020-mosar.eu/

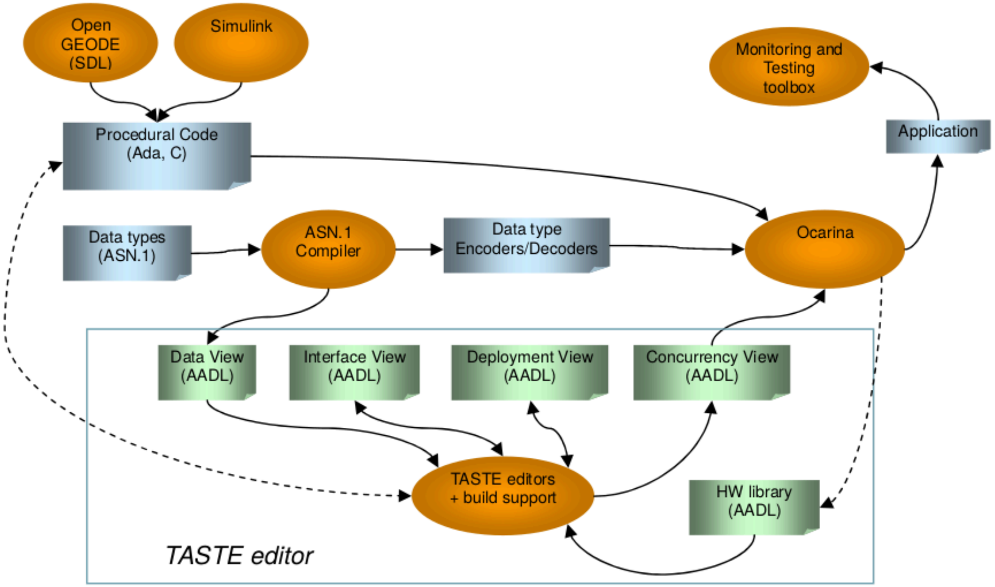


Taste Workflow

It is an iterative process :

1. Describe the system logical architecture and interfaces with ASN.1 and AADL
2. Generate code skeletons and write the applicative code or models
3. Capture the system hardware and deployment
4. Verify models
5. Build the system and download it on target
6. Monitor and interact with the system at run-time

Taste Workflow (2)



Contents

3 Model

ASN.1

Abstract Syntax Notation One (ISO and ITU-T)

- Simple text notation to describe data types and constraints International standard (ISO and ITU-T).
- An ASN.1 definition can be readily mapped (by a pre-run-time processor) into a C or C++ or Java data structure that can be used by application code, and supported by run-time libraries providing encoding and decoding of representations.
- Separate the encoding rules from the types specification
- e.g. My-Integer::= INTEGER (0..255)
- https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx

ASN.1 : Vol 501 d’Ariane 5

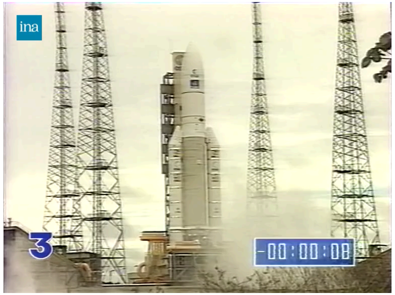


Figure 4: Video INA

- Aucune victime.
- Autour de 500 million de dollars de pertes : fusée, 4 satellites de la mission cluster.
- Rapport de l’incident : http://www.capcomespace.net/dossiers/espace_europeen/ariane/ariane5/AR501/AR501_rapport_denquete.htm

ASN.1 : Vol 501 d’Ariane 5 rapport

- La chaîne de pilotage d’Ariane 5 repose sur un concept standard :
  - l’attitude du lanceur et ses mouvements dans l’espace sont mesurés par un système de référence inertielle (SRI) : calcule les angles et les vitesses.
  - Les données du SRI sont transmises au calculateur embarqué (OBC) qui exécute le programme de vol et qui commande les tuyères des étages d’accélération à poudre et du moteur cryotechnique Vulcain.
- Redondance au niveau des équipements :
  - 2 SRI travaillant en parallèle : ces systèmes sont identiques tant sur le plan du matériel que sur celui du logiciel. L’un est actif et l’autre est en mode “veille active” ; si l’OBC détecte que le SRI actif est en panne, il passe immédiatement sur l’autre SRI à condition que ce dernier fonctionne correctement.
- La conception des SRI d’Ariane 5 est pratiquement la même que celle d’un SRI qui est actuellement utilisé à bord d’Ariane 4, notamment pour ce qui est du logiciel.
- L’exception logiciel interne du SRI s’est produite pendant une conversion de données de représentation flottante à 64 bits en valeurs entières à 16 bits (sur le biais horizontal). Le nombre en représentation flottante qui a été converti avait une valeur qui était supérieure à ce que pouvait exprimer un nombre entier à 16 bits. Il en est résulté une erreur d’opérande. Les instructions de conversion de données (en code Ada) n’étaient pas protégées contre le déclenchement d’une erreur d’opérande

ASN.1 : Basic Types

Type	Examples

INTEGER	`My-int ::= INTEGER (0..7)`
	`value My-int ::= 5`

REAL	`My-real ::= REAL (10.0 .. 42.0)`
BOOLEAN	
ENUMERATED	`My-enum ::= ENUMERATED { hello, world }`

OCTET STRING	`My-string ::= OCTET STRING (SIZE (0..255))`
	`value My-string::= 'DEADBEEF'H`

BIT STRING	`My-bitstring ::= BIT STRING (SIZE (10..12))*`
	`value My-bitstring ::= '00111000110'B`

ASN.1 : complex types

Type	Examples

SEQUENCE	
	```
	My-seq ::= SEQUENCE {
	  x My-int,
	  y My-enum OPTIONAL
	}
	value My-seq::= { x 5 }
	```

CHOICE	
	```
	My-choice ::= CHOICE {
	  choiceA My-real,
	  choiceB My-bitstring
	}
	value My-choice ::= choiceA : 42.0
	```

SEQUENCE OF	
	```
	My-seq ::= SEQUENCE (SIZE (0..5)) OF BOOLEAN
	value My-seq:= { 1, 2 ,3 }
	```

SET / SET OF	

ASN.1 : Encoding Rules

- Several standardized encoding rules :
  - BER (Basic Encoding Rules)
  - PER (Packed Encoding Rules) : more recent, most compact encoding rules
  - …
- Describe how the values defined in ASN.1 should be encoded for transmission (i.e., how they can be translated into the bytes ‘over the wire’ and reverse).

ASN.1 : ACN

ACN was created as a simple **ASN.1** companion language allowing to describe custom binary encodings for complex data structures. It is useful in the following case:

- you need to implement a binary packet encoder/decoder for a legacy protocol
- no standard **ASN.1** encoding rules (*PER*, *DER*, *XER*, *OER*, *JER*…) fits
- **ECN**, the Encoding Control Notation is too complicated or you have no tool available
- the targeted encoding is not too complex
- https://taste.tuxfamily.org/wiki/index.php?title=Technical_topic:_ASN.1_-_An_introduction_to_ACN

ACN works in pair with **ASN.1** and provides various ways to customize the memory layout of data structures.

ASN.1 : ACN Examples

Here is a simple example, starting from a basic integer type:

```
MyInteger ::= INTEGER (0..7)
```

If you choose to represent a data of this type using the **ASN.1 PER**, the encoding will be optimal: 3 bits. However, your protocol may say that this has to be encoded using 32 bits with a little endian representation. **ACN** offers the syntax for this:

```
MyInteger [size 32, endianness little, encoding pos-int]
```

AADL

- Architecture Analysis and Design Language, a standard from **SAE** (https://www.sae.org/)
- Used to model the **logical** and the **physical** architecture of the system
- Textual and graphical representations
- Used in **TASTE** to capture the system structure, interfaces, hardware and deployment.

Interface View

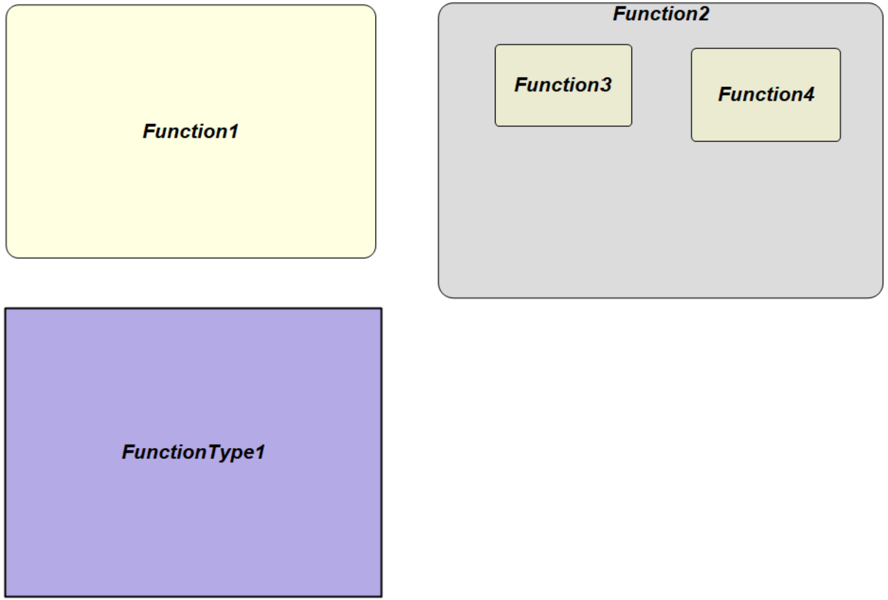
- The interface view in **TASTE** captures the model of the system logical architecture, independently from a software implementation.
- In the most abstract way, a system modeled with **TASTE** contains functions that exchange messages with other functions through interfaces.
- The semantics of the interface view is largely derived from the Specification and Description Language (SDL) communication view.
- The AADL language (AADL v2.2) is used as an intermediate textual format for the Interface View.

IV Functions

There are several forms of functions in **TASTE** :

- functions that represent an active component of the system (e.g. a state machine or a control law), something that can communicate with surrounding functions through interfaces.
- nesting functions, that allow to structure the system in a hierarchical way (grouping by context, or for better readability). Nesting is recursive.
- function types, which contain a generic behavior and that can be instantiated
- function instances

IV Functions View



IV Functions Properties

Attribute	Description
Label	Graphical label
Language	Source code langage (c, ...)
Source Text	Path to the source code
Fill Color	Choose the color of the graphical element
is_Instance_of	For function instance only: reference to the function type
...	

IV Functions Context Parameters

The «Context Parameters» tab offers a space for flexibility :

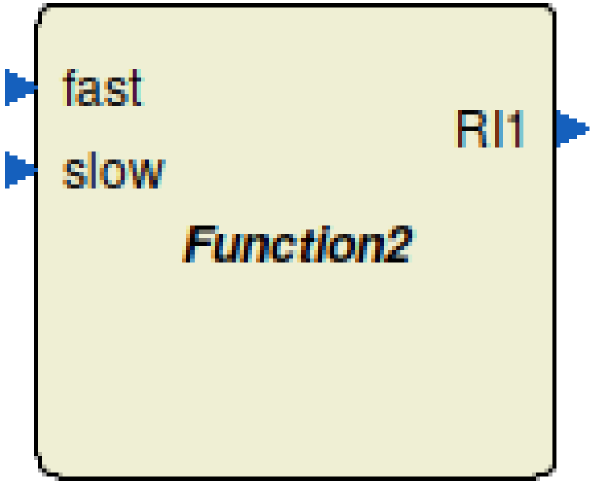
- Context parameters allow defining constants at model level and make them accessible from user code
  - Support for C, Ada and Simulink (instructs code generator to generate « tuneable parameters », which are global variables)
  - Value can be generated from an external source
- TASTE directives are used to fine-tune the build process with additional properties (e.g. compilation or link flags that are specific to a piece of code)
  - Used to integrate Simulink code when it requires special defines (-DRT, -DUSE_RTMODEL)
  - When a property proves usefulness, it gains a dedicated entry in the GUI

IV Provided and Required interfaces

A provided/required interface is a service offered/required by a function. It can be

- Cyclic: only PI, execution of the cyclic activities of the system, it does not take any parameter.
- Sporadic: asynchronous execution and optionally carry one parameter at most.
- Protected: synchronous execution, mutual exclusion execution (any number of in and out parameters).
- Unprotected: synchronous execution, no mutual exclusion execution (any number of in and out parameters)
- Any: only RI, can be connected to any interfaces.

IV PI and RI View



IV Cyclic PI Attributes

Attribute	Description
Operation Name	Graphical label
Kind	Type of the PI (cyclic here)
Period (ms)	Period value of the cycle
Deadline (ms)	Deadline of the execution
WCET (ms)	Duration of the execution

IV Sporadic PI Attributes

Attribute	Description
Operation Name	Graphical label
Kind	Type of the PI (sporadic here)
Min inter-arrival Time (ms)	Minimum delay between 2 executions
Deadline (ms)	Deadline of the execution
WCET (ms)	Duration of the execution
Queue Size	

IV (un)Protected PI Attributes

Attribute	Description
Operation Name	Graphical label
Kind	Type of the PI (protected here)
Deadline (ms)	Deadline of the execution
WCET (ms)	Duration of the execution

IV RI Attributes

Attribute	Description
Label	Graphical label
Inherits from PI	Inherits the label, parameters, … from the connected PI (true or false)
Kind	Type of the PI (protected here)

IV PI and RI Parameters

Each parameter has a type (from the **ASN.1** model), a direction (in or out), and an encoding protocol :

- **Native** : means memory dump, no special treatment
- **UPER** : compact binary encoding
- **ACN** : user-defined encoding

IV AADL Mapping

IV	AADL
Interface View	Package + System composition
Function, Function Type, Function Instance	Package + System comp. + System subcomp.
Provided Interface	Subprogram + Provides Spg Access feature
Required Interface	Subprogram + Requires Spg Access feature
Context Parameter	Data subcomponent
Connection	Subprogram Access Connection
Textual description	Comment at the feature or subcomp. level
Graphical note	Comment at the Package level

Deployment View

- Map functions on hardware
- Centralized and distributed systems
- Can add buses, drivers.. Extensible (every component is described in an AADL file)

DV elements

- Node: hardware support of a one processor (only monoprocessor) and devices.
- Processor: Processor.
- Partition: adress space of a processor.
- Device: communication element between nodes through bus.
- Bus: communication element between devices.
- Interfaces (Devices and Bus) : used to make the connection between devices and buses.
- Connections between interfaces.

DV AADL Mapping

DV	AADL
Deployment View	Package + System
Node	Package + System
Processor	Processor
Partition	Partition
Device	Device
Bus	Bus
Bus Connection	Bus Access Connection
Textual description	Comment at the feature or subcomp. level
Graphical note	Comment at the Package level

Contents

4 Tools

Tools list

Here are the tools used in the taste toolchain:

- **ASN1SCC** (taste toolchain): ASN.1 Compiler
- **Ocarina**: AADL model processor, written in Ada.
- **Kazoo** (taste toolchain): it generates code skeletons, concurrency view, build scripts, …
- **Taste Editors** (taste toolchain): UI editors/analysis for the different models (IV, DV, …)
- **Cheddar**: real time scheduling analysis tool.
- **Marzhin**: AADL simulator.
- …

ASN1SCC

- Overview:
  - The **ASN1SCC ASN.1** compiler parses an **ASN.1** grammar.
  - The **ASN.1** glue generators parse an **ASN.1** grammar types and create run-time data translation "bridges".
- Features:
  - The **ASN.1** compiler:
    - supports *NATIVE*, *Unaligned PER* (uPER) and *ACN* (user-controlled) encodings
    - creates both *C* and *Ada* type declarations and encoders/decoders
  - The **ASN.1** glue generators:
    - Perform type mapping of **ASN.1** grammars to declarations in *SCADE*, *Simulink*, *Pragmadev Studio*, *C* and *Ada*.
    - Create run-time translation bridges between the C types generated by SCADE, Simulink, OpenGEODE, Pragmadev Studio and the C types generated by the ASN1SCC ASN.1 compiler.
    - The combination of the two, allows ASN.1 to be used as a "universal translator" between modelling tools - and forms the "heart" of TASTE's data modelling.

Ocarina

- Overview:
  - **Ocarina** is used to analyze and build applications from **AADL** descriptions.
  - Thanks to its modular architecture, **Ocarina** can also be used to add **AADL** functions to existing applications.
  - http://www.openaadl.org/ocarina.html
- Features:
  - **Parser** : support both **AADL1.0** and **AADLv2** syntaxes;
  - **Code generation** : targeting *C* real-time operating systems: *RT-POSIX*, *Xenomai*, *RTEMS*; and *Ada* using *GNAT* for native and *Ravenscar* targets;
  - **Model checking** : mapping of **AADL** models onto *Petri Nets*, *timed* (TINA) or *colored* (CPN-AMI);
  - **Schedulability analysis** : mapping of **AADL** models onto **Cheddar** or *MAST* models
  - **Model Analysis** : using the *REAL* language, one can analyze an **AADL** model for particular patterns or compute metrics.

Kazoo

- **Kazoo** is a command line tool which processes input **AADL** models and produces derived models, code and scripts used to build complete system.
- It uses **Ocarina** for **AADL** parsing and templates-parser for templates processing and files generation.
- Standard library of components used by templates is defined by **TASTE** common models from ocarina_components.aadl file, which includes supported processors, devices, drivers etc.
- Generated build scripts apart from calling code compilers also run other tools like ocarina and aadl2glueC to create rest of the source code.
- https://taste.tuxfamily.org/wiki/index.php?title=Kazoo

Taste Editors

The TASTE editor is a graphical editor that provides the following functionalities for the TASTE modelling activities:

- Full integration of the **Dataview-IV-DV** activities within a single tool.
- Enhanced support for reuse of IV models and components.
- Updated real-time analysis tools for the **CV**, including the most recent version of the **Cheddar** scheduling analysis framework and of the **Marzhin AADL** simulator.
- https://taste.tuxfamily.org/wiki/index.php?title=TASTE_DataView/IV/DV/CV_Graphical_Editor

Taste Editors Technologies

The **TASTE editors** uses various background technologies:

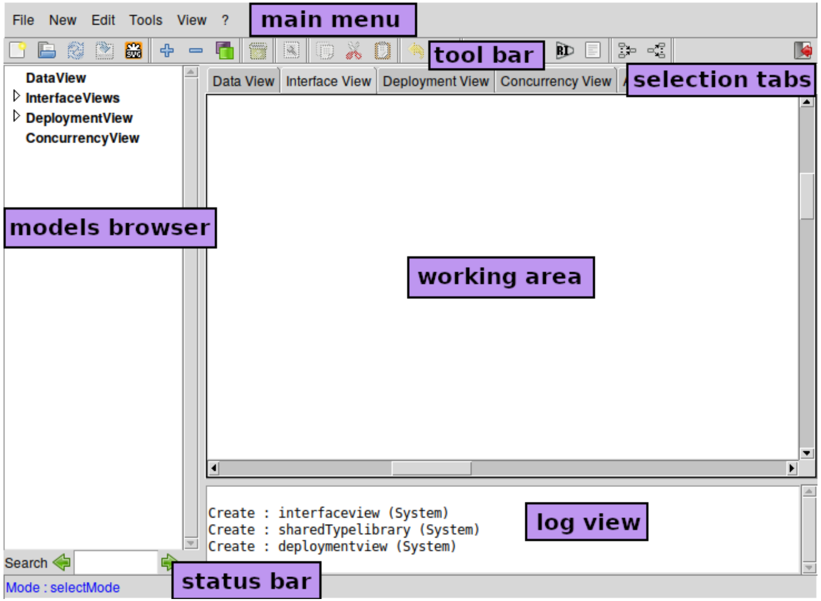
- **GMP** (Graphic Model Processing): generic graphical framework developed by *Ellidiss Technologies*.
- **LMP** (Logic Model Processing): generic model transformation and **AADL** toolbox developed by *Ellidiss Technologies*.
- **Cheddar**: real-time scheduling analysis tool developed by the *University of Brest* and *Ellidiss Technologies*.
- **Marzhin**: **AADL** run-time simulator developed by *Virtualys* and *Ellidiss Technologies*.
- **Ocarina**: **AADL** compiler and code generator developed by *ISAE*.

Taste Editors Overview (1)

The **TASTE graphical editor** consists of a single window that encompasses several elements:

- **main menu** and **button bar**. The button bar is updated upon the current modelling or verification activity, as defined by the selection tab.
- **models browser** where the overall project hierarchy and organization is displayed in a deployable tree structure.
- set of **selection tabs** to enable one of the proposed **TASTE** modelling or verification activity, and update the content of the working area.
- **working area**, showing either:
    - A **textual editor** where the **ASN.1** and **ACN** representations of a **TASTE DataView model** can be edited, or
    - A **box-arrow editor** where a graphical representation of a **TASTE Interface View** or **Deployment View model** can be edited, or
    - The **timing analysis** results of a **TASTE Concurrency View model**, or
    - A **textual viewer** that displays the generated textual **AADL** code
- **log view** where main editing operations are logged.
- **status bar** showing system information, warning and error messages.

Taste Editors Overview (2)



```
Create : interfaceview (System)
Create : sharedTypelibrary (System)
Create : deploymentview (System)
```

Mode : selectMode

Taste Editors: Main Menu

- **File → New**: reset the editors.
- **File → Load**: load a data view, an interface view, a deployment view, a concurrency view or a hardware library.
- **File → Reload**: reload a view (data, interface, deployment or concurrency).
- **File → Save**: save all views (interface, deployment or concurrency).
- **File → Load HW Library Directory**: load a hardware library located in a directory (it can be split in multiple files).
- **New→«elements»**: create elements for the differents editors.
- **Tools → «external tools»**: this section contains configurable external tools thanks to a plugin mechanism.

Taste Editors: Main Toolbar

- Common buttons
  - New: same as main menu.
  - Load: same as main menu.
  - Reload: same as main menu.
  - Save: same as main menu.
- Specific to IV/DV:
  - “New section”: same as menu. All the add buttons (plus group/ungroup connection for iv).
- Specific to CV:
  - Simulation Control Panel: launch the Cheddar and Marzhin simulation control panel.
  - Real Time Properties: edit the real time properties of the concurrency view.

Taste Editors: Data View

The **data view** tab contains a text editor which allows to display/edit the content of the **acn/asn** files selected in the **DataView** part of the browser.

Taste Editors: IV/DV

- The **interface view** and **deployment view** tabs contain a diagram editor which allows to construct graphically an IV/DV.
- The **graphical objects** can be create through the main menu (New), the main toolbar or the diagram contextual menu. Furthermore, copy of existing object can be done with a drag & drop mechanism with the IV or DV browser.
- **Function Type/instance**: the user can create local or use shared function types. These types can be instantiated. A modification of a local type will be automatically applied to its instances.
- **IV import**: interface view aadl files can be imported and theirs functions copied in the local interface view through the drag & drop mechanism of the browser.
- **Hardware Library**: several deployment view objects have to be created with a hardware library. When the user load one, its objects (processors, devices and buses) appear in the deployment view browser. Theses objects can be copied to the local deployment view through the browser drag & drop mechanism.

Cheddar

- GNU GPL real-time scheduling simulator and schedulability tool.
- Only the framework part is used (Cheddar Kernel).
- **Cheddar** allows you to model software architectures of real-time systems and to check their schedulability or other performance criteria.
- Analysis Tools:
  - Scheduling simulations
  - Apply feasibility tests on tasks
  - Algorithms to assign priorities to tasks
  - Partitionning tools for periodic task set
  - ...
- http://beru.univ-brest.fr/cheddar/

Marzhin

- Marzhin is an AADL **multi-agent simulator**.
- Uses the standard **AADL Behavior Annex** to express threads and subprograms pseudo-code focused on timing analysis purpose.
- Develop by *Virtualys* (https://www.virtualys.fr/) and *Ellidiss Technologies*.

Taste Editors: CV

- The **concurrency view** tab contains a horizontal paned window. Its upper part displays the **Cheddar** analysis tools and the lower part displays the **Marzhin** analysis tools.
- The **Cheddar** analysis tool widget is composed of a toolbar and each button activate a different analysis tool widget :
  - a synthesis table which displays the theoretical and simulation result of cheddar and the results of Marzhin for comparison.
  - a simulation chronogram produced by Cheddar.
  - a more complete theoretical results from Cheddar.
  - a more complete simulation results from Cheddar.
- The **Marzhin** analysis tool widget is composed of a tree representing the real time components hierarchy (processor, partition, threads, input/output data, …) and a **simulation chronogram**. The chronogram horizontal scrollbar is attached to Marzhin and Cheddar in order to make comparison between them.
- Furthermore, two other widgets can be used to modify the tool's configuration:
  - Simulation Control Panel.
  - Real Time Properties Editor.

CV: Simulation Control Panel

- The general tab:
  - **Zoom Factor**: the value can be chosen among given values (0.1, 0.5, 1, 2 or 4) and is used to zoom the simulation timeline.
  - **Filters** : the value is used to filters the components in the hierarchy tree
- The Marzhin tab:
  - **Simulation Control**: the tool buttons can be used to launch, pause, stop, refresh, go to the last tick and set the optimization of the Marzhin simulation. The current simulation result can be saved in a vcd file.
  - **Computation range**: this value represents the time interval of the simulation.
  - **Selected scenario**: set the selected scenario (among the ones found in the given scenario asc file in the command line).
  - **Speed factor**: the user can choose the simulation speed between several values (x1, x2, x5 or x10).
- The Cheddar tab:
  - **Computation range**: this value represents the time interval of the simulation.
- The **Help tab** displays all the color code used in the simulation chronogram.

CV: Real Time Properties Editor

- This editor is used to modify the **real time properties** of the **concurrency view threads** (dispatch protocol, period, priority, ...).
- The new values are taken into account by the analysis tools and can be saved in a file.

Taste Editors: AADL Viewer

- In the upper part: some aadl analysis tools
  - Parsing the model
  - Instantiate the model
  - Metrics on the model
- In the lower part:
  - textual viewer to display the content of the aadl file of the IV, DV or CV selected in the browser.

Taste Editors: AADL Models

- The **TASTE models** that can be loaded into and saved from the TASTE editors are serialized in **AADL format**.
- For the purpose of TASTE models serialisation, **only a subset of the AADL language is used**, and the TASTE entities are associated with specific AADL constructs that have been defined to ensure a correct semantic mapping. A TASTE project makes use of several kinds of models. Each of them is associated with a dedicated AADL subset. These models are:
  - **DataView**: automatically generated from **ASN.1** source code.
  - **Interface View**: automatically generated from the Interface View diagram.
  - **Hardware Library**: provided by the **Ocarina** environment.
  - **Deployment View**: automatically generated from the Deployment View diagram.
  - **Concurrency View**: automatically generated by the **TASTE Build Support** utility.
  - **Customised Properties**: specified by the **TASTE_IV_Properties** and **TASTE_DV_Properties** files located in the config directory of the TASTE editor distribution.
- WARNING: all the AADL files that are involved in a TASTE project are either provided or **automatically generated**. It is thus not necessary - and even not recommended - for a TASTE end-user to edit the AADL models. Directly editing the provided or generated AADL files may lead to load errors and model corruption.

Contents

5 Demonstration

Calculator Exemple

Make a calculator:

- Create a function named «calculator»: provides the calculator services.
- Create Pis:
  - «Add» with 3 parameters: in a, in b, out result
  - «sub» with 3 parameters: in a, in b, out result
  - ...
- Create a function named «UI»: interaction with a user, requires the calculator services.
- Create Ris
- Connect the Pis and the RIs

Contents

6 Conclusion