Open Archive Toulouse Archive Ouverte (OATAO)
OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: http://oatao.univ-toulouse.fr/
Eprints ID: 10292


Any correspondence concerning this service should be sent to the repository administrator: staff-oatao@inp-toulouse.fr
Code generation and models
Perspectives from the TASTE project

Jérôme Hugues, ISAE/DMIA
What is the TASTE process?

> TASTE derives from ASSERT (FP6, 2004 - 2008)
> The TASTE 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 “just” to develop application code and follow associated process: their skills are misused, better used to hack-n-play: add drivers, play with hardware, add verification/testing facilities, configure RTOS, etc.
> Some considerations on Model-Based S(oftware|ystem) Engineering
  » Desired functionality have reached levels (dangerously) close to being unmanageable by humans. “Empirical” methodologies have reached their limit...
  » Model Driven Engineering offers some notion of correctness, at the model level
  » Tools generate C/Ada code – but no established standard for interoperability
  » Messages/data exchanged face the same issue – no common data modeling
  » Some options for automation – remove human from the loop
> The TASTE process proposes solutions to
  » capture a system using user-friendly (yet formal) modeling 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 ISAE, Ellidiss, PragmaDev, Neuropublic
  » Open source toolchain: see http://taste.tuxfamily.org
Rationale for multiple modeling notations

AADL and ASN.1 are combined to provide a complete Description of the system, SDL SCADE, C, Ada for impl.

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

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

-- ASN.1 type definition
Overview of the TASTE process

Interface, functional and data views

Modeling phase

Hardware and deployment views

Vertical transformation

VM level containers

TASTE runtime

Automatic code generation

Complete system

Tool chain allowing the user to capture his system using a set of models and to generate the complete code of the application without manual intervention.
TASTE proposes an integrated set of tools for modeling

- Graphical DSL, provides basic blocks + semantics
  - Follow Ravenscar computational model
- Or use directly AADL, and connect it with your own process
Adding function behavior: pick your notation

> Generation of code “skeletons”

```plaintext
#include "user_code.h"

void cyclicactivationimplfortc()
{
    /* Write your code here! */
}
```

System basic_fv
USE Datamodel;
SIGNAL basictotc (T_TM);
SIGNAL tcommand (T_HLTC_PLUS);
SIGNAL basiccontrol (T_CONTROL_IN);
SIGNAL controldown tobasic (T_CONTROL_DOWN_OUT);
SIGNAL controlup tobasic (T_CONTROL_UP_OUT);
SIGNAL cyclicactivationimplementation;

procedure aplc_basic_op COMMENT '#c_predef'
FPAR IN thrusters_opening T_THRUSTERS_OPENING,
IN pfs_iwm_arming_relay_status_on T_PFS_IWM_ARMING_RELAY_STATUS_ON,
IN pfs_hltc_red_button_is_on T_PFS_HLTC_RED_BUTTON_ON,
IN mcu_is_T_MBU_D,
IN pfs_ewc_msux_msuy_hs T_PFS_EWM_MSU_HS,
IN health_status T_FTCP_HEALTH_STATUS,
IN pfs_ewm_dtg12_msu T_PFS_EWM_DTG12_MSU,
IN hltc T_HLTC,
IN end_boost_is_reached T_END_BOOST_IS_REACHED,
IN sun_is_aimed T_SUN_IS_AIMED,
IN pfs_ewc_msux_msuy_hs T_PFS_EWM_MSU_HS,
IN cam_mode T_CAM_MODE,
IN controller_to_be_activated T_CONTROLLER_TO_BE_ACTIVATED,
IN navigation_output T_NAVIGATION_OUTPUT;

EXTERNAL;

procedure mysimulink COMMENT '#c_predef'
FPAR IN my_in T_FOR_SIMULINK_IN,
IN my_in2 T_control_in,
OUT my_out T_FOR_SIMULINK_OUT,
OUT my_out2 T_Control_in;

EXTERNAL;
```
What TASTE 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
Implementation of the TASTE process

The source code for each subsystem, as generated by the tools using graphical DSL or plain AADL.

The messages exchanged between subsystems.

The encoders and decoders of the ASN.1 messages.

The semantically equivalent types of the messages per tool.
Provide a centralized place to add instrumentation, configure the build or the deployment

» Also allow for automatic execution from toolsets
» Main objective: convenient way to test models on the final target

Targets: Ada, C (RT-POSIX, Xtratum, RTEMS, …)

» Scripts to deploy on your target (board, simulator)
Gathering metrics from projects

> Scheduling, memory, coverage analysis
  » Integrate third-party tools, consolidate results at project level
> All available from main GUI, or command line
Testing and so forth

- **Rapid prototyping:** the toolchain generates GUIs to quickly test the system, replay capabilities using MSCs.

- **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).

```plaintext
MY-MODULE DEFINITIONS ::= BEGIN

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

<table>
<thead>
<tr>
<th>MySequence (SEQUENCE)</th>
<th>Bit mask</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence preamble</td>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>No</td>
<td>Field</td>
<td>Type</td>
<td>Optional</td>
</tr>
<tr>
<td>----</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>1</td>
<td>field1</td>
<td>INTEGER</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>field2</td>
<td>INTEGER</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>field3</td>
<td>BOOLEAN</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>field4</td>
<td>MyChoice</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>field5</td>
<td>OCTECT STRING</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>field6</td>
<td>MySequenceIf</td>
<td>Yes</td>
</tr>
</tbody>
</table>
```
GUI and graphical editors are optional
» This is not Eclipse ;)
The toolchain follows the Unix philosophy
» One concern := one tool
Rely only on standards: AADL, ASN.1 + your models
» Eases maintenance, extensibility, adaptation by user
» Hooks to extend each step of the process
» Easy to extend and add new features, tools, checks, etc.
» Allows for batch processing, regression testing, etc.
GUI (Ellidiss), orchestrator are TCL or Python scripts
» Configure other tools from TASTE, main entrypoints
» Add your own tools (!)
» E.g. COMPASS for safety analysis
A library of components for runtime deployment
» SpaceWire, Ethernet, Serial, MIL-STD1553, ...
» As both AADL models and source code, per OS
» Add your own drivers (!)
Tools under the hood (cont’d)

> **Buildsupport (ESA)**
>  » Mapping from high-level AADL to detailed levels
>  » Generation of wrappers for instrumentation

> **ASN.1 Space Certifiable Compiler (Neuropublic)**
>  » Generation of marshallers for ASN.1, Ada/SPARK & C
>  » No dynamic memory allocation, no system calls
>  » *Let you specify data encoding in ACN (!)*

> **“AADL runtime” – two options**
>  » Code generated from AADL models (Ocarina, ISAE)
>  » Ada: any Ravenscar-compliant Ada runtime
>  » C: RT-POSIX, RTEMS, Xtratum, POK, Native, ..
>  » Runtime is not part of the generator itself
>  » *Add your OS of choice (!)*
TASTE is now 8 years old

- Many demonstrators: FDIR, Galileo receiver, robotic arm, ...
- Pragmatic view on code generation
  - Remove manual work in critical paths
  - Enable/disable features automatically
- Everything is static, no dynamic model analysis at run-time
  - Code ready to be embedded on final target
  - Instrumentation present only on demand
- Focus on supporting most activities in V cycle
  - Prototyping, testing, early V&V activities, assessment of binary
- Integrate heterogeneous models
  - Depending on their relevance, *but* textual syntax is a must
- (coming) Support for formal methods

TASTE is more than a working prototype: it is now a full proof of concept of initial vision on code generation

- Clear separation between system and software engineers

TASTE is free software, enjoy!