Validating SEDS as a bridge between hardware and software models

Executive Summary Report

ETD-N7S-ESR-001 rev. 1.0

N7 SPACE SP. Z O.O.

Prepared by	Date and Signature
Konrad Grochowski	
Verified and approved by	
Michał Mosdorf	

Table of Contents

Change Record

Issue	Date	Change
1.0	2023-06-23	Initial release

1 Introduction

1.1 Purpose, scope and content

This document defines the Executive Summary Report for the “Validating SEDS as a bridge between hardware and software models” project. The following introduction provides a short description of the project objectives.

1.2 Project motivation and objectives

TASTE is ESA’s established toolchain targeting heterogeneous systems implementing a model-based system/software engineering approach. At its core, it utilizes AADL and ASN.1 models for architecture and data description respectively. System behaviour can be described via various methods, including, but not limited to, SDL, Simulink models, C or Ada. TASTE can be used both for system design, analysis, simulation and actual implementation targeting embedded devices.

Recently TASTE has been enriched with the support for CCSDS SOIS EDS (SEDS, or simply EDS further within this document), an XML based standard for specifying data interfaces offered by flight hardware such as sensors and actuators. The support is provided via conversion of SEDS to Interface View, ASN.1/ACN and SDL.

Currently TASTE users can design the logical system architecture and its logical behaviour using Interface View and applicable Function source language editors. When the logical architecture is deployed onto physical architecture using Deployment View, TASTE generates all the necessary glue and communication code. Such an approach is quite convenient, but is has two major drawbacks:

- users lack control over the protocol details, which may hinder interoperability and make it impossible to meet certain interface requirements
- the actual communication drivers have to be provided for the runtime in a traditional, non-MBSE fashion, making the final, complete software not fully modelled

The objective of this project is to resolve this issues, reinforcing software re-use and interoperability by providing the capabilities to:

- choose between different packetization schemes for communication interfaces
- design communication device drivers using MBSE
- use SEDS in modelling of the communication device drivers
- use the designed communication device drivers with traditional code, outside of TASTE

2 Applicable and reference documents

2.1 Applicable documents

ID	Title	Reference	Rev.
AD1	ECSS – Space engineering Software	ECSS-E-ST-40C	6 March 2009

2.2 Reference documents

ID	Title	Reference	Rev.
RD1	ETD Software Requirements Specification	ETD-N7S-SRS-001	1.0
RD2	Architecture Technical Note	ETD-N7S-TN-001	1.1
RD3	SpaceCreator Repository	https://gitrepos.estec.esa.int/taste/spacecreator	
RD4	Kazoo Repository	https://gitrepos.estec.esa.int/taste/kazoo	
RD5	TASTE Runtime Common Repository	https://github.com/n7space/TASTE-Runtime-Common	
RD6	TASTE Linux Runtime Repository	https://github.com/n7space/TASTE-Linux-Runtime	
RD7	TASTE SAMV71 Runtime Repository	https://github.com/n7space/TASTE-SAMV71-Runtime	
RD8	TASTE Wiki	https://taste.tuxfamily.org/wiki/	
RD9	TASTE-CD-DEMO	https://github.com/n7space/TASTE-CD-Demo	
RD10	TASTE-CD-TEST	https://github.com/n7space/TASTE-CD-Tests	
RD11	ETD Final Report	ETD-N7S-FR-001	1.0
RD12	ETD Technology Achievement Summary	ETS-N7S-TAS-001	1.0

3 Terms, definitions and abbreviated terms

This document acronyms and abbreviations are listed here under.

AADL	Architecture Analysis & Design Language
ACN	ASN.1 Control Notation
ASN.1	Abstract Syntax Notation 1
CD	Communication device
DV	Deployment view
EDS	Electronic Data Sheet
IV	Interface view
N7S	N7 Space
SDL	Specification and Description Language
SEDS	SOIS Electronic Data Sheet
SOIS	Space Onboard Interface Services
SW	Software
VM	Virtual Machine
XML	eXtensible Markup Language

4 Project Objectives

The high-level project objectives described in 1.2 and Architecture Technical Note [RD2] were refined through analyses and discussions with the Client to:

- Provide SEDS TASTE Driver Editor Software as a SpaceCreator plugin for graphical editing of the communication device drivers.
- Provide TASTE Linux C++ Communication Device Templates for TASTE devices compatible with TASTE Linux Runtime.
- Provide TASTE SAMV71 Communication Device Templates for TASTE devices compatible with TASTE SAMV71 Runtime.
- Provide changes in TASTE SAMV71 Runtime, TASTE Linux C++ Runtimes and TASTE Common Runtime for accommodating changes to communication device Packetizer.
- Provide fixes to OpenGEODE C backend.

The above objectives were captured in the Software Requirements Specification [RD1].

5 Work logic

Work Breakdown Structure is illustrated in Figure 1.

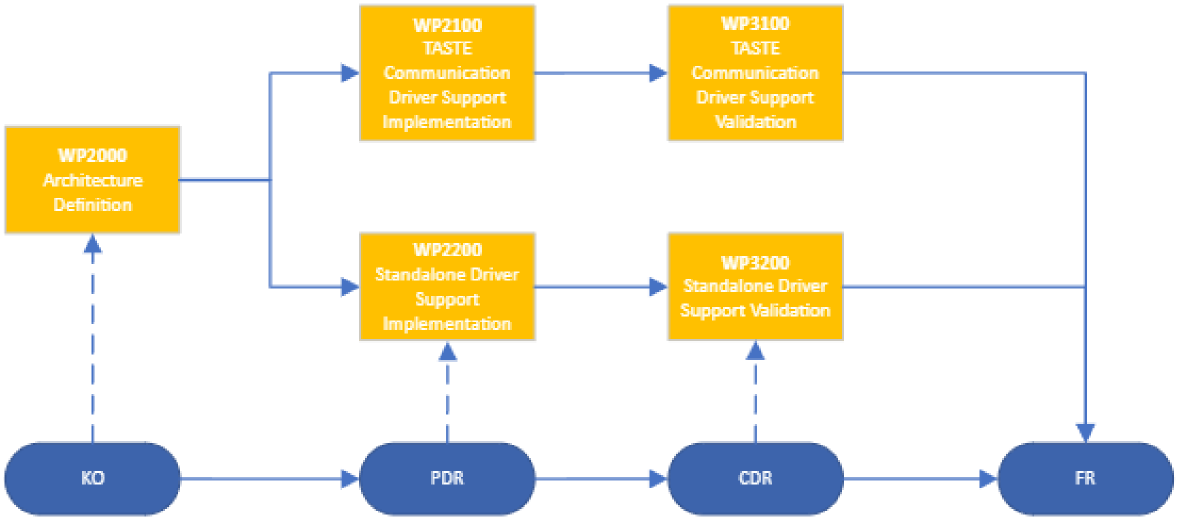

Figure 1 – ETD Work Breakdown Structure (WBS)

The project was formally divided into the following work-packages:

- WP1000 – Management and Reporting
- WP2000 – Architecture Definition
  - TASTE Communication Driver EDS specification
  - Example drivers selection (UART, MCAN, RS485)
- WP2100 – TASTE Communication Driver Support Implementation
  - Implement device to TASTE mapping through SEDS
  - Implement device to TASTE mapping through custom Interface View
- WP2200 – Standalone Driver Support Implementation
  - Stand-alone driver generation
  - Functional mocks generation
- WP3100 – TASTE Communication Driver Support Validation
- WP3200 – TASTE Standalone Driver Support Validation

6 Project achievements

6.1 ETD integration into TASTE toolchain

The developed creators were integrated into the SpaceCreator IDE source code [RD3] as described in the Architecture Technical Note [RD2]. A description of each component with its status is provided in Table 1.

Table 1 – ETD and TASTE integration software components

Component	Functionality	Type
DeviceGeneratorPlugin	Plugin providing STDESW GUI frontend	new
QtCreator	IDE hosting the plugins	N/A

make	Build system	N/A
Kazoo	Template based code generator	reused
OpenGEODE	SDL to Ada and SDL to C translator	reused
Interface View Prefabs	Prefabricated Interface Views containing the components and interfaces required for Communication Device Integration Views	new
Skeleton Templates	Templates for generating the code skeletons and build Makefiles	reused
SedsXmlImporter	Module for importing SEDS data from XML.	reused
libiveditor	Interface View editor	reused
ivcore	Interface View data model	reused
libdveditor	Deployment View editor	reused
dvcore	Deployment View data model	reused
SedsToAsn1Translator	Translator of SEDS to ASN.1	reused
SedsToSdlTranslator	Translator of SEDS to SDL	reused
SedsToIvTranslator	Translator of SEDS to Interface View	reused
DriverHelper	Helper for platform runtime	reused
SAMV71 Runtime	Runtime for SAMV71	reused
Linux Runtime	Runtime for Linux	reused
Broker	Broker for driver communication	reused
Packetizer	Packetizer for driver communication	reused
ThinPacketizer	Simple version of packetizer	new
Hwas	Low-level driver library	reused
Linux CD templates	Communication device templates for Linux x86	new
SAMV71 CD templates	Communication device templates for SAMV71	new
Threading	Threading library for drivers	N/A
FreeRTOS	FreeRTOS library	N/A

The created software presents its capabilities via creation wizard available in the SpaceCreator IDE. Software description has been provided in TASTE wiki [RD8].

Communication device wizard, after successful device creation presents *InterfaceView* file in classical SpaceCreator Editor with functions specific for communication device. These functions have to be complemented with proper implementations to handle their interfaces. During configuration, user can choose between only Broker device and Broker with user provided *Packetizer*.

ETD driver after being provided with proper implementation (SEDS or any other supported implementations) has to be compiled and installed with support of Make tool, which makes it available for users to select on *DeploymentView*.

The compilation of communication device project generates necessary files generated from TASTE Communication Device templates and provides a sanity check for all the provided and generated files. Project compilation does not include linking or executable generation as communication device is not a stand-alone application, it needs to be injected into target TASTE project. To inject communication device, proper scripts invoked by Make tool have been provided. The scripts modify SpaceCrator configuration files for *DeploymentView* editor what allows to incorporate communication device driver in the same way as other non-generated drivers. They also copy selected ETD driver files to proper installation directories.

6.2 OpenGEODE C Generator

Until recently, the OpenGEODE C Generator was largely abandoned in favour of the Ada generator. In the scope of "Tiny Runtime to Run Model-Based Software on CubeSats" project it was noticed that C generator might lead to a smaller footprint (which is especially important for low resource platforms), which might be critical in the case of communication drivers. This led to work focused on re-enabling the C Generator in OpenGEODE and making it a fully-fledged alternative to the Ada generator. The ongoing process is focused on adapting the existing tests for Ada Generator to work with C Generator. An important aspect of this process is to provide high-quality generated source code. Achieving this is essential to use the C generator for communication drivers.

6.3 TASTE support

In order to create proper ETD project files, TASTE skeleton and glue code generator Kazoo [RD4] has been extended with a new set of templates provided for communication devices. These new templates target Linux x86 or SAMV71 platforms. These new templates are based on existing templates from concurrency view set. New templates generate files needed to access necessary *Broker* and *Packetizer* interfaces and allow to incorporate user provided implementation files in a form of TASTE supported implementation. Moreover TASTE supports implementation generation from SEDS files, so it can also be used as function implementation for communication devices. The design is provided in [RD2]. The source code has been provided in [RD3].

6.4 TASTE Runtime Common

TASTE Runtime Common has be extended to accommodate usage of user provided *Packetizer* from communication device project. Usage of user provided *Packetizer* in communication device project is optional and needs to be specified in *DeploymentView* during communication port configuration specification. The source code has been provided in [RD5].

6.5 Demonstration application

The demonstration application has been created to demonstrate capabilities to create communication devices using MBSE methodology targeting both Linux x86 and SAMV71 platforms. The demonstration will also show the capabilities to incorporate these communication device drivers into target TASTE project consisting of several partition implementing communication between Linux x86 and SAMV71. The drivers been designed in an the MBSE fashion, using SpaceCreator DeviceGenerator wizard.

6.6 Demo application

The demonstration model has been created as a simple system which implements a manageable PID controller of a simulated physical entity (see Figure 2). It allows to facilitate non-trivial communication between partitions for both UART and RS485 drivers. MCGUI which is a user interface that handles control communication uses RS485 driver provided by user-created communication device. It has been implemented and installed for both Linux x86 and SAMV71 platform. ScienceGUI which is user interface to interact with simulated data from model on the other hand uses UART driver also implemented as a communication device.

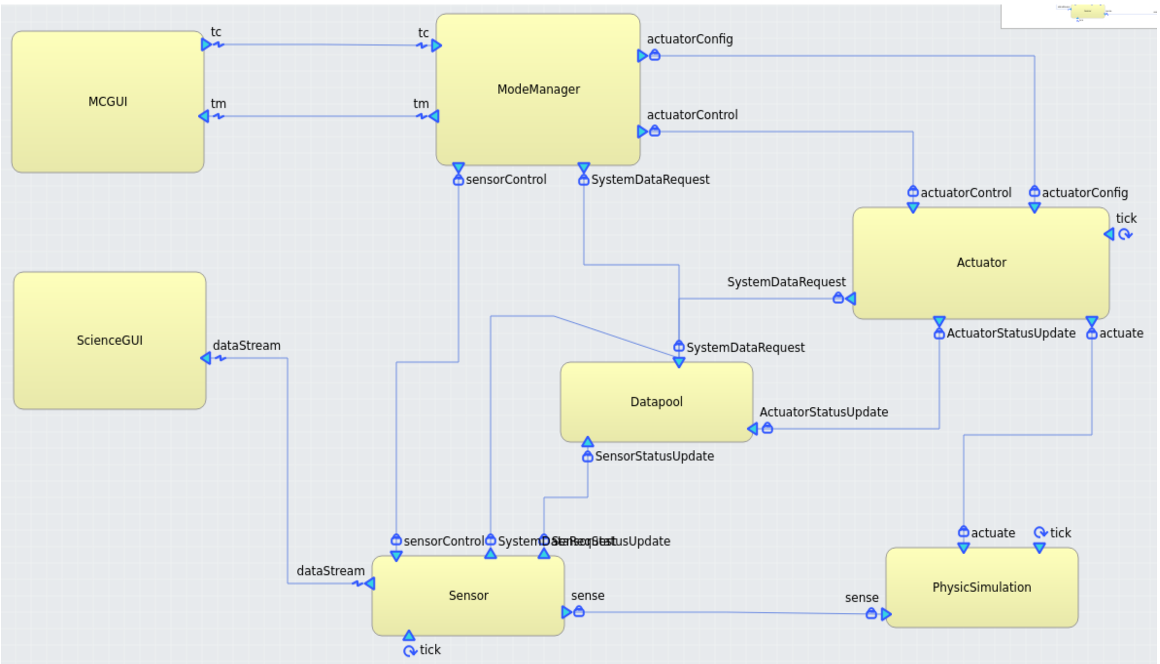

Figure 2 – InterfaceView of demo project.

In demonstration application we have two GUI partitions: MCGUI for control communication and ScienceGUI for interacting with the data generated by the model. In the MCGUI we can send telecommands that change the current mode of the system from OFF to Idle and ON. Every mode changes the way the science data are returned from SAMV71 partition. Moreover, we can change the type of telecommand and send messages with configuration data that changes parameter of the simulation. The changed parameter is *target_height*. In the ScienceGUI interface we can see data generated by the model returned to the Linux partition with their plot.

All the demonstration application and communication devices source codes are available on [RD9].

7 Conclusions

As a result of ETD project, user gains control over the communication protocol details including packet encapsulation and low-level transmission capabilities. It increases interoperability and makes it easier to meet certain interface requirements. The drivers may be designed in a MBSE fashion making the final software fully modelled.

Additionally TASTE users can now easily incorporate SEDS models into their logical models as Functions of the Interface View. This increases the role and convenience of the SEDS models in TASTE.

The implementation proved to be much more complex than initially planned, mostly due to the level of coupling in internal components of TASTE, which triggered some refactoring, but the achieved features are a huge milestone in the TASTE development and design.

Delivered changes significantly improve the entire TASTE environment and increase the functionality of the modelling process. Designing drivers and the ability to choose the method of packetizing makes the whole process more consistent and scalable and the addition of full support for the new SEDS language greatly increases the capabilities of TASTE.