Toolchain to connect EDS and TASTE

Executive Summary Report

ETB-N7S-RP-001 rev. 1.0

N7 SPACE SP. Z O.O.

Prepared by	Date and Signature
Michał Kurowski	
Verified by	
Filip Demski	
Michał Kurowski	

Table of Contents

Change Record

Issue	Date	Change
1.0	2022-12-09	Initial release

1 Applicable and reference documents

1.1 Applicable documents

ID	Title	Reference	Rev.
AD1	ECSS – Space engineering Software	ECSS-E-ST-40C	6 March 2009
AD2	Space Onboard Interface Services – XML Specification for Electronic Data Sheets	CCSDS 876.0-B-1	April 2019

1.2 Reference documents

ID	Title	Reference	Rev.
RD1	Toolchain to connect EDS and TASTE Software Requirements Specification	ETB-N7S-SRS-001	1.2
RD2	Toolchain to connect EDS and TASTE Architecture Design Document	ETB-N7S-ADD-001	1.2
RD3	Toolchain to connect EDS and TASTE SEDS Analysis Report	ETB-N7S-TN-001	1.0
RD4	EDS Device Driver Feasibility Report	ETB-N7S-TN-002	1.0
RD5	Toolchain to connect EDS and TASTE Statement of Work	ESA-TECSWT-SOW-MP-200330	1.0
RD6	MBSE2022 paper and presentation “Toolchain to connect EDS and TASTE”	https://indico.esa.int/event/407/	N/A
RD7	SpaceCreator Repository	https://gitrepos.estec.esa.int/taste/spacecreator	N/A
RD8	Kazoo Repository	https://gitrepos.estec.esa.int/taste/kazoo	N/A
RD9	TASTE Runtime Common Repository	https://github.com/n7space/TASTE-Runtime-Common	N/A
RD10	TASTE Linux Runtime Repository	https://github.com/n7space/TASTE-Linux-Runtime	N/A
RD11	TASTE Linux Drivers Repository	https://github.com/n7space/TASTE-Linux-Drivers	N/A
RD12	TASTE SAMV71 Runtime Repository	https://github.com/n7space/TASTE-SAMV71-Runtime	N/A
RD13	TASTE SAMV71 Drivers Repository	https://github.com/n7space/TASTE-Linux-Drivers	N/A
RD14	SAMV71 BSP Repository	https://github.com/n7space/SAMV71-BSP	N/A
RD15	DemoSat2 Repository	https://github.com/n7space/Demo-Sat-2	N/A
RD16	TASTE Wiki	https://taste.tuxfamily.org/wiki/	N/A
RD17	Custom arm-eabi-gcc Compiler Wrapper Repository	https://github.com/n7space/adac-hybrid-arm	N/A

2 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
EDS	Electronic Data Sheet
HW	Hardware
IDE	Integrated Development Environment
IO	Input/Output
LIDAR	LIght Detection And Ranging
MBSE	Model Based Software/System Engineering
MSC	Message Sequence Charts
N7S	N7 Space
RAM	Random Access Memory
SAVOIR	Space AVionics Open Interface aRchitecture
SDL	Specification and Description Language
SEDS	SOIS Electronic Data Sheet
SOIS	Space Onboard Interface Services
SW	Software
TLR	TASTE Linux Runtime
TSR	TASTE SAMV71 Runtime
VM	Virtual Machine
XML	eXtensible Markup Language

3 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.

CCSDS SOIS EDS (SEDS or simply EDS further with this document) is a new XML based standard for specifying data interfaces offered by flight hardware such as sensors and actuators. It can be used to formally describe device interfaces (including the required data types), logical structure (via components) and behaviour (via component implementations containing activities and state machines). Its scope overlaps with the combined feature set of AADL, ASN.1 and SDL used in TASTE.

As TASTE is a toolchain which can be used to design, specify, implement and deploy complete heterogeneous distributed systems, and EDS is a specification which focuses on individual devices (such as sensors and actuators), interfacing both can provide:

- a complete toolchain for implementing and deploying EDS based devices;
- means to achieve greater interoperability by TASTE with other systems.

As stated in the statement of work [RD5], "The objective of this study is to provide a working toolset for EDS by extending TASTE with import/export functionalities, allowing to make actual use of EDS in real-life applications. TASTE will be extended with an optimized runtime that will be compatible with EDS components. A demonstration of EDS with TASTE will be done on a microcontroller connected to sensors and actuators that represent a small space system.".

The above high-level objective was refined through analyses and discussions with the Client to:

- providing EDS to ASN.1 and ACN translator
- providing EDS to XML Interface View translator,
- providing EDS to SDL translator,
- providing ASN.1 and ACN to EDS translator,
- providing XML Interface View to EDS translator,
- integrating the abovementioned translators with the SpaceCreator Integrated Development Environment,
- providing a new TASTE runtime for Linux, which uses modern C++ constructs instead of PolyORB middleware,
- providing a new TASTE runtime for ARM SAMV71 microcontroller, which uses FreeRTOS primitives directly, without relying on PolyORB middleware,
- analysing the suitability of EDS for defining device drivers,
- creating a demonstration application in the form of a LIDAR instrument.

The above objectives were captured in the Software Requirements Specification [RD1]. Preparation of the requirements was supported by EDS analysis, which was documented in the SEDS Analysis Report [RD3]. Its purpose was to evaluate EDS background, scope, features, characteristics and use-cases, as well as establish potential high-level mapping onto TASTE (ASN.1, ACN, AADL and SDL) constructs.

4 Work logic

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

- **WP 1.1 - Management and Reporting**
- **WP 1.2 - Project documentation**
- **WP 2.1 - Requirements consolidation and preliminary design**
- **WP 3.1 - EDS Importer development**
- **WP 3.2 - EDS Exporter development**
- **WP 3.3 - TASTE support**
- **WP 4.1 - Case study development and testing**

During the project execution a design decision was made to integrate all the translators in the SpaceCreator Integrated Development Environment (as described in [RD2]), re-using some of the existing code, and creating mechanisms which were re-used in two different activities running in parallel – "Model Checking for Formal Verification of Space Systems" and "AURORA" (project which received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004291). This design decision effectively merged WP 3.1 and WP 3.2 tasks, as well as the relevant deliverables.

The project contains the following milestones:

- **KO** – Kick-off - held on 2021 01 11 via teleconference,
- **MTR** – Mid-Term Review - held on 2021 12 13 via teleconference,
- **AR** – Acceptance Review – to be held on 2022 12 13 via teleconference (this document is a deliverable due for AR).

5 Project achievements

5.1 EDS integration into TASTE toolchain

The developed translators were integrated with the SpaceCreator IDE source code [RD7], as described in the Architecture Design Document [RD2].

The created software exposes the required translation capabilities via both a command line application, SedsConverter (packaged within spacecreator.AppImage portable binary), and import/export commands available directly in the SpaceCreator IDE, provided by SedsPlugin. Software User Manual has been provided as a contribution to the TASTE wiki [RD16].

Figure 1 summarizes the possible end-products of the implemented EDS translation:

- HTML interface control documents, generated by ESA's ASN1SCC compiler from the ASN.1/ACN data specifications,
- C or Ada data type definitions and transcoders, generated by ESA's ASN1SCC compiler from the ASN.1/ACN,
- Component interface descriptions in XML and AADL Interface View,
- C or Ada code implementing component behaviour, generated by ESA's OpenGEODE SDL editor and compiler.

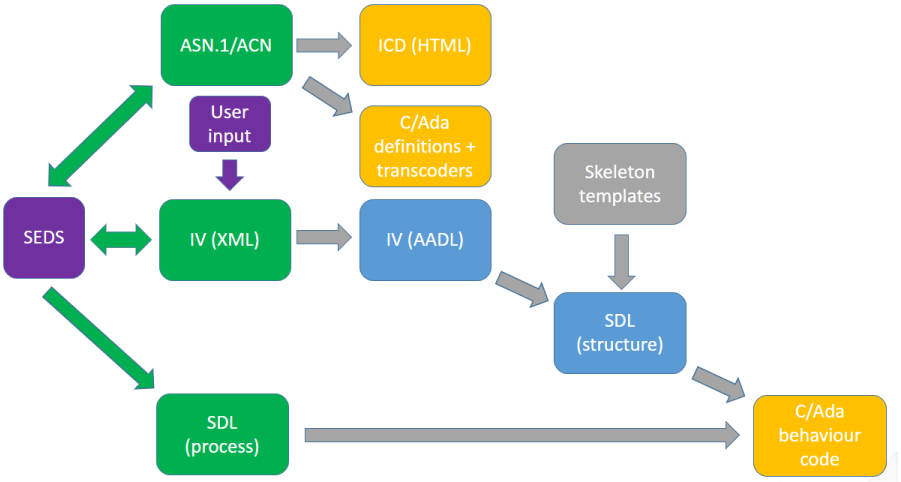

Figure 1 - EDS translation products

As illustrated in Figure 2, the above products can be complemented by user inputs in the form of deployment specification and inter-component connections in order to produce a complete embedded software application. The code generation process relies on Kazoo templating engine, which uses both generic and runtime specific templates to produce the necessary glue-code and build scripts. The build scripts implement the necessary actions and declarations to integrate the generated code with all necessary software libraries, such as e.g., runtime-specific communication drivers and board support packages.

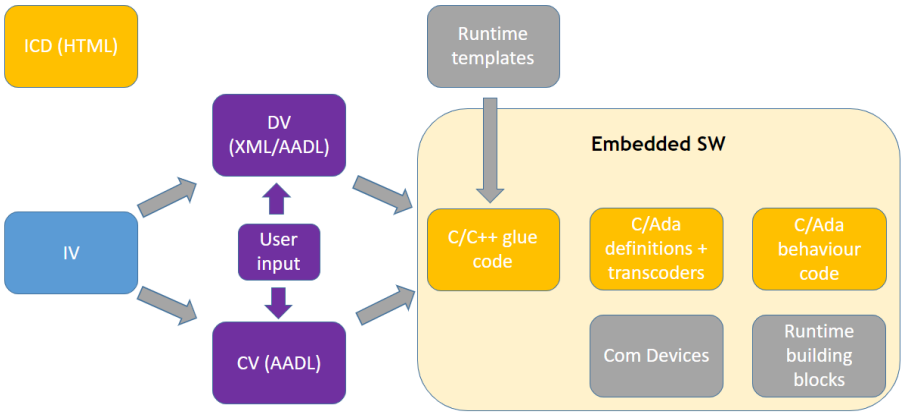

Figure 2 - TASTE code generation products

5.2 TASTE support

In order to evaluate the EDS, TASTE was extended with two new runtimes, for Linux and SAMV71 respectively. Both were designed with simplicity and low-resource usage in mind. As a consequence of abandoning the PolyORB middleware, they cannot communicate with legacy TASTE runtimes via built-in communication mechanisms. However, the used CCSDS Space Packet inspired message encapsulation is significantly more lightweight, reducing resource usage. However, full CCSDS Space Packet compatibility was not achieved. The main reason for this is extending the packet length field to accommodate large (> 65kB) packet sizes. Design of both is provided in [RD2].

5.2.1 TASTE Linux Runtime

TASTE Linux Runtime targets Linux compatible systems, however, it was tested only on Debian Linux running on x86-64 architecture. It uses modern C++ constructs for threading and synchronization. The runtime provides two communication device drivers – IP socket (for communication between distributed nodes) and UART (for communication with embedded systems). TASTE Linux Runtime is available by default in up-to-date TASTE installations. The source code is distributed between several repositories: [RD8], [RD9], [RD10] and [RD11].

5.2.2 TASTE SAMV71 Runtime

TASTE SAMV71 Runtime target ARM Cortex M-7 SAMV71 microcontroller included in the SAMV71 Xplained Ultra evaluation kit, which includes 2 MB of SDRAM. It uses FreeRTOS for threading and synchronization. The runtime provides one communication device driver – UART (for communication with the Linux runtime or other embedded systems). The runtime code also includes a reference Hwas (Hardware Access EDS) implementation, which provides high-level interfaces (described via EDS and convertible to XML Interface View) for direct memory access and interrupt handling, as neither of these capabilities is supported natively by EDS nor SDL. The source code is distributed between several repositories: [RD8], [RD9], [RD12], [RD13] and [RD14].

In order to compile Ada code (either provided directly by users, or generated from SDL by OpenGEODE), AdaCore's Ada compiler was integrated in the custom toolchain available in [RD17].

5.3 Demonstration application

The demonstration application was created to demonstrate the EDS to ASN.1, Interface View and SDL translation, TASTE Linux and TASTE SAMV71 runtimes, and the capability to operate low-level hardware via components with interfaces and behaviour fully described via EDS. It contains:

- TF Luna LIDAR rangefinder device connected via UART,
- Stepper motor for rotating the LIDAR,
- Limit switches for guarding the LIDAR rotation range,
- Sun/Light sensor,
- LED for simulating actuation based on the data retrieved from the LIDAR.

As the demonstration application was designed in an MBSE fashion, the detailed design can be reviewed directly in the project repository [RD15].

5.4 EDS Device Driver Feasibility

Lessons learned derived from the implementation of EDS based components for use in the demonstration application were used to prepare a paper for MBSE2022 workshop [RD6] and EDS Device Driver Feasibility Report [RD4]. The report summarizes the demonstration application design, relevant aspects of the TASTE SAMV71 runtime architecture and Hwas implementation, as well as the encountered technical limitations.

No functional limitations were found that would make EDS specifications unsuitable for describing device drivers. The lack of dedicated constructs for direct memory access or interrupt handling was demonstrated to be mitigable by providing a proxy component, such as Hwas. However, it was discovered that the provided tooling along with the provided runtime introduces a performance

overhead, which may limit the EDS applicability to device drivers which do not require fast commanding or short interrupt response times. The impact is application specific and might be mitigated in the future iterations of the tooling. Lack of publicly available dedicated authoring tools supporting user-friendly editing of EDS state machines was also mentioned.

6 Conclusions

All stated project objectives have been achieved:

- TASTE toolchain has been extended with EDS import and export capabilities,
- runtimes capable of hosting EDS based components were implemented,
- demonstration application has been developed and successfully tested.

The EDS specification has been demonstrated to be suitable for describing device drivers, and the discovered limitations of the current implementation have been reported.