Reaching into space TOGETHER

Deployment of the PUS-C Standard in Projects supported by an Automatic Generation Toolset

Final Presentation

Agenda

- Project Context and Overview
- Part I: PUS foundation modelling
- Part II: The PUS C Toolset
- Part III: Applying MSC & SDL modelling to PUS
- Part IV: Applications, Parallel activities and Future Work
- Conclusions

Project Context

Given the size of the PUS-C document, this project aimed at tackling a few potential problems with the new version of the standard:

- **Internal consistency** (production of contents in a systematic way according to the foundation model)
- **Usability** (production of a toolset easing the tailoring of the PUS standard)

Project Overview

The objectives of the PUSCGen project are the following (1/2):

- Verify and validate the **PUS foundation model** (clause 5);
- Extend the PUS foundation model to capture missing semantics to fully cover the functional spec (clause 6);
- Using an ORM tool, develop a PUS foundation database and populate it with the standardized PUS data;

Project Overview

The objectives of the PUSCGen project are the following (2/2):

- Develop a document generator able to reproduce the full standard or tailored versions of it;
- Develop an ASN.1 code generator able to cover the full scope of PUS services or tailored versions of them;
- Verify and illustrate the PUS system requirements through modelling in SDL and MSC formalisms.
- Verify the developed tools using a realistic use case.

PUS foundation model

The PUS foundation is a conceptual model

- Captures the generic PUS concepts with associated rules
- Provides rules that are applicable to any service type and their instances
- Is expressed in clause 5 of the PUS-C Standard

Conceptual modelling is supported by formal methods and tools

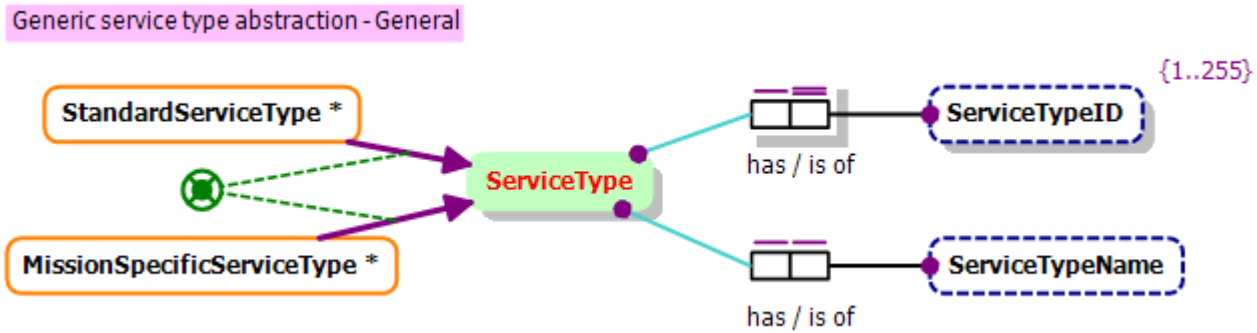
- Selected method: Object Role Modelling (ORM)
- Selected tool: NORMA

PUS foundation model

Contents of an ORM model:

- Object types: The concepts
- Fact types: The relations between these concepts
- Constraints: Restrictions on the valid populations

The PUS specification (Clauses 6 & 8) is a population of this model !



PUS foundation model

Methodology

1. Review of an ESA provided input ORM model
   - Adequacy of the existing diagrams w.r.t. PUS clause 5
   - Completeness of existing model (identification of requirements present in clause 5 but not modelled)
2. Review of the PUS standardised service types specification against foundation model
   - Identify requirements violating the foundation model
   - Identify requirements not covered by the foundation model
3. Assess feasibility of generating study outputs
   - Database software
   - Generated PUS standards (clauses 6 & 8)
   - ASN.1 & ACN representation of the PUS interfaces

PUS foundation model

Generation assessment

- Identify the skeleton of the PUS
- Verify that all information is captured in model

Section	Contents
TransactionCapability Level	
a. Capability declaration	The <subservice> capability to <capability> shall be declared when specifying that subservice
NOTE1: RequestLink	Link to the request which is directly defined by this capability
b. Request & Instructions	Each request to <request> shall contain <number> instruction(s) to <instruction>
b*. Request with toplevel arguments	If request-level arguments exist, this requirement is presents them together with the instruction list.
c. Request rejection conditions	If there exists rejection conditions at request level, their definition
d. Instruction contents (for each instruction)	The argument list of the instruction
e. Instruction rejection conditions (for each instruction)	If there exists rejection conditions at instruction level, their definition
f. Instruction rejection consequences	Standard requirement requiring the rejection of request with failed start of execution
g. Processing of valid instruction in case of error ?	Requirement stating whether valid instructions are executed when an invalid instruction is detected in the request.
h. Valid instruction processing definition (for each instruction)	The list of actions to take for each valid instruction
i. Valid request processing definition (if any)	The list of actions to take for each valid request (note that the order of (h) and (i) can be reversed if the request processing must be performed prior to the instruction processing)

PUS foundation model

Generation assessment

- Identify the skeleton of the PUS
- Verify that all information is captured in model

One main limitation

- Each subservice type defines and/or accesses “System Objects” (e.g. “Function”, “Memory”,…)
- Those are only modelled in a generic way
- Specific modelling has been performed in the scope of ATOP, to further refine the PUS model.

Modelling activity outcomes

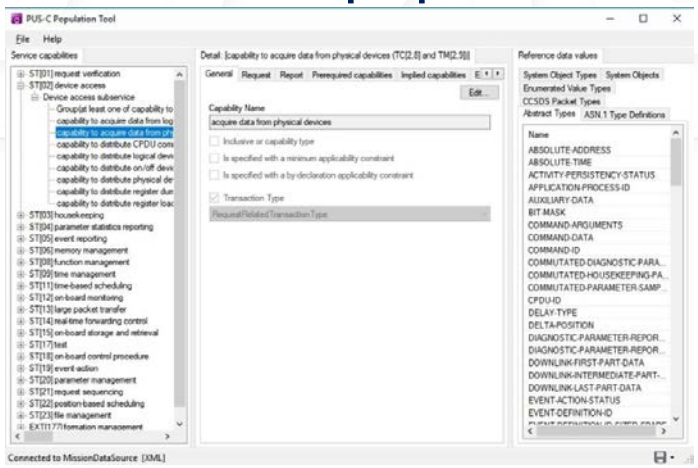
Lessons learned document produced gathering:

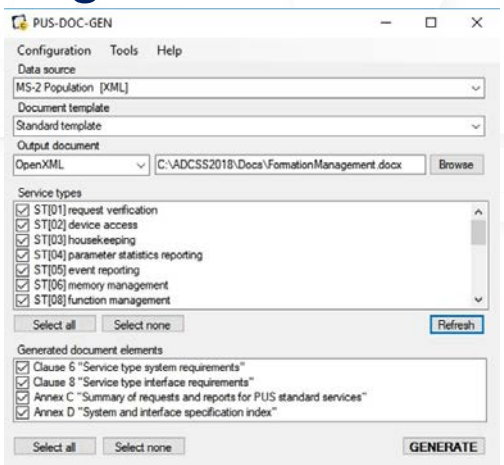
- **Valuable feedback on using the NORMA tool for performing conceptual modelling**
- **21 change requests towards the PUS-C standard, out of which**
  - **7 are considered major (The specification has a functional problem)**
  - **7 are considered minor (The specification lacks internal consistency, or is not covered by the foundation)**
  - **7 are considered editorial only.**

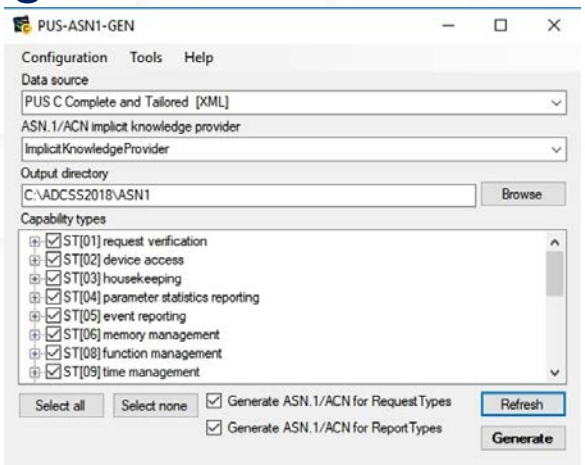
Relational database directly generated from the ORM model to support the toolset development.

PUS-C Toolset

- The toolset consists of 3 applications that work together:
  - Population Tool
    - population definition and tailoring
  - DOC-GEN
    - ECSS-E-ST-70-41C compatible document generation
  - ASN1-GEN
    - population tailoring and ASN.1/ACN generation





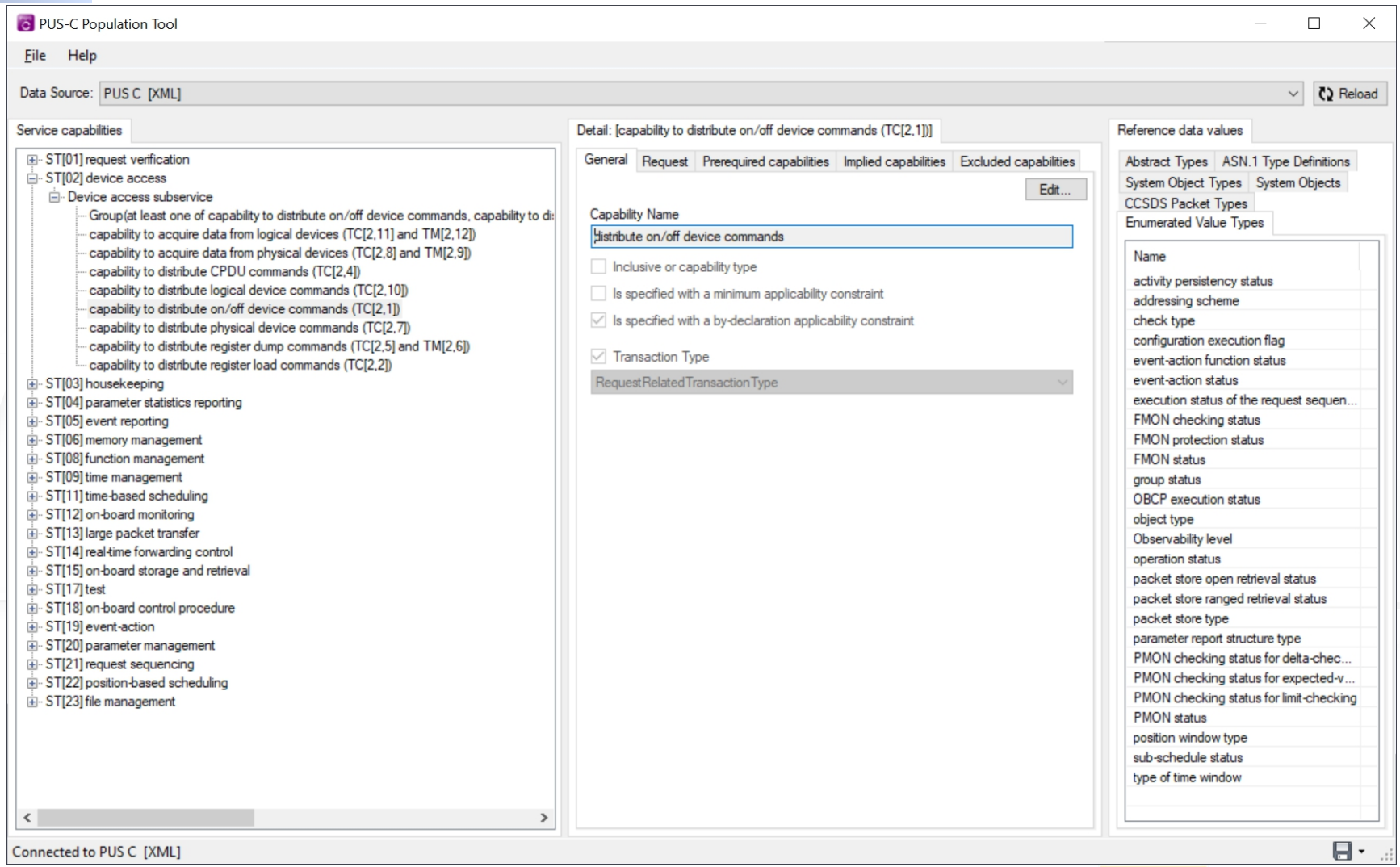


PUS-C Toolset workflow (part 1)

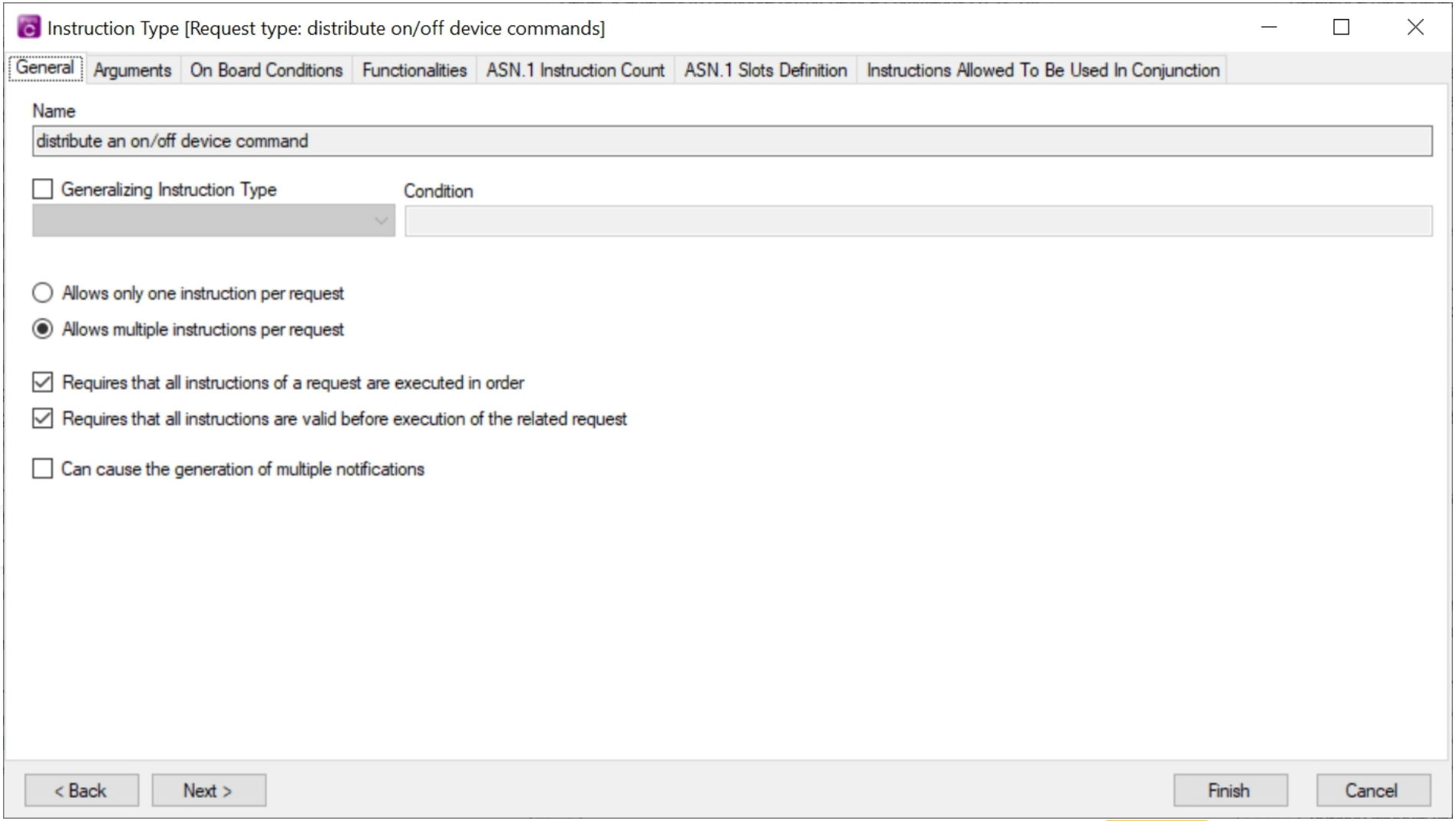
- Define common concrete and abstract (to be tailored) types
- Define service types
- Define subservice types
- Define capability types
- Define message types (request and report types)
- Define instruction and notification types
- Define instruction arguments and notification reporting data
- Define conditions, functionalities, objects...

- No need to write the requirements manually – just capture the essence of a given service type within the constraints of the PUS Foundation Model
- Describe X type, not X instance (to be tailored per-mission)

PUS-C population



PUS-C population



Document generation

- The database population can be used to generate requirements and descriptions structured as chapters 6 and 8, as well as annexes C and D of ECSS-E-ST-70-41C
- Population representing the entire PUS C standard was created during the project, allowing to recreate the ECSS standard with good fidelity (given a suitable template, containing other chapters, (mostly) not dependent on particular service types)
- In many cases, a generated page is hard to distinguish from the actual standard
- Some discrepancies were detected and reported to ESA

PUS-C standard document

m. For each logical device, the device identifier shall be declared when specifying that subservice.

n. For each logical device, the set of parameter identifiers used for data acquisition shall be declared when specifying that subservice.

o. For each logical device, the set of supported commands and associated arguments that are used to address that device shall be declared when specifying that subservice.

6.2.4.2 Distribute on/off device commands

a. The device access subservice capability to distribute on/off device commands shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[2,1] distribute on/off device commands".

NOTE For that declaration, refer to requirement 6.2.3a.

b. Each request to distribute on/off device commands shall contain an ordered list of one or more instructions to distribute an on/off device command.

c. Each instruction to distribute an on/off device command shall contain:

1. the device address.

d. The device access subservice shall reject any request to distribute on/off device commands if:

1. that request contains an instruction that refers to an unknown device address.

e. For each request to distribute on/off device commands that is rejected, the device access subservice shall generate a failed start of execution notification.

f. For each request to distribute on/off device commands that contains only valid instructions, the device access subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to distribute an on/off device command, the device access subservice shall:

1. distribute the related on/off command to the related device address.

6.2.4.3 Distribute register load commands

a. The device access subservice capability to distribute register load commands shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[2,2] distribute register load commands".

NOTE For that declaration, refer to requirement 6.2.3a.

b. Each request to distribute register load commands shall contain an ordered list of one or more instructions to distribute a register load command.

c. Each instruction to distribute a register load command shall contain:

8.2 ST[02] device access

8.2.1 General

a. Each packet transporting a device access message shall be of service type 2.

8.2.2 Requests and reports

8.2.2.1 TC[2,1] distribute on/off device commands

a. Each telecommand packet transporting a request to distribute on/off device commands shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.2.4.2.

b. For each telecommand packet transporting a request to distribute on/off device commands, the application data field shall have the structure specified in Figure 8-10.

repeated N times

N	on/off device address
unsigned integer	enumerated

Figure 8-10 Distribute on/off device commands

8.2.2.2 TC[2,2] distribute register load commands

a. Each telecommand packet transporting a request to distribute register load commands shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.2.4.3.

b. For each telecommand packet transporting a request to distribute register load commands, the application data field shall have the structure specified in Figure 8-11.

repeated N times

N	register address	register data
unsigned integer	enumerated	deduced

Figure 8-11 Distribute register load commands

8.2.2.3 TC[2,4] distribute CPDU commands

a. Each telecommand packet transporting a request to distribute CPDU commands shall be of message subtype 4.

PUS-C Toolset workflow (part 2)

- Provide mission-specific tailoring:
  - Select service types
  - Select subservice types
  - Select capability types
  - Provide definitions of the abstract types
  - [if required] Adjust arguments and reporting data

- Selection is subject to constraints defined in the population (capabilities depend on each other, or exclude each other; some can be declared, some are required)
- Abstract type tailoring and argument/reporting data adjustment are performed by providing/editing ASN.1/ACN

ASN.1/ACN generation+

- Tailored population can be used to generate ASN.1/ACN definitions of TC and TM
- The process requires a set of ASN.1/ACN definitions reflecting (e.g.) CCSDS packet encapsulation (see ECSS-E-ST-70-41C chapter 7) - the so-called implicit knowledge
- The ASN.1/ACN definition can be used by ESA's ASN1SCC compiler to generate:
  - data structures
  - encoders/decoders (C, with optional Python bindings)
  - ICDs
  - and more...

PUS-C demonstration ICD

Telecommands (CHOICE) ASN.1 ACN — Min: 0 bytes Max: 2284 bytes

No	ACN Parameters [?]	Type
1	service-type-id	TYPE-ID
2	message-subtype-id	TYPE-ID
3	remainder-size	INTEGER

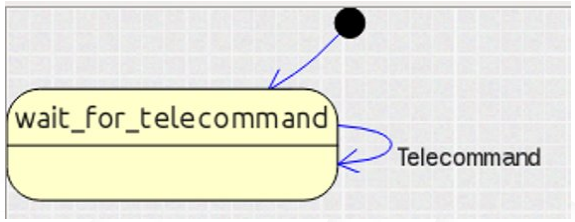
No	Field	Comment	Present	Type	Constraint	Min Bits	Max Bits
1	tc2-2-distributeRegisterLoadCommands		service-type-id=2 AND message-subtype-id=2	TC2-2-DistributeRegisterLoadCommands-Type		8	72
2	tc2-5-distributeRegisterDumpCommands		service-type-id=2 AND message-subtype-id=5	TC2-5-DistributeRegisterDumpCommands-Type		8	72
3	tc2-4-distributeCpduCommands		service-type-id=2 AND message-subtype-id=4	TC2-4-DistributeCpduCommands-Type		8	2160
4	tc2-7-distributePhysicalDeviceCommands		service-type-id=2 AND message-subtype-id=7	TC2-7-DistributePhysicalDeviceCommands-Type		8	136
5	tc2-8-acquireDataFromPhysicalDevices		service-type-id=2 AND message-subtype-id=8	TC2-8-AcquireDataFromPhysicalDevices-Type		8	112
6	tc2-10-distributeLogicalDeviceCommands		service-type-id=2 AND message-subtype-id=10	TC2-10-DistributeLogicalDeviceCommands-Type		8	136
7	tc2-11-acquireDataFromLogicalDevices		service-type-id=2 AND message-subtype-id=11	TC2-11-AcquireDataFromLogicalDevices-Type		8	176
8	tc2-1-distributeOnoffDeviceCommands		service-type-id=2 AND message-subtype-id=1	TC2-1-DistributeOnoffDeviceCommands-Type		8	72
9	tc3-5-enableThePeriodicGenerationOfHousekeepingParameterReports		service-type-id=3 AND message-subtype-id=5	TC3-5-EnableThePeriodicGenerationOfHousekeepingParameterReports-Type		11	67
10	tc3-6-disableThePeriodicGenerationOfHousekeepingParameterReports		service-type-id=3 AND message-subtype-id=6	TC3-6-DisableThePeriodicGenerationOfHousekeepingParameterReports-Type		11	67
11	tc3-1-createAHousekeepingParameterReportStructure		service-type-id=3 AND message-subtype-id=1	TC3-1-CreateAHousekeepingParameterReportStructure-Type		24	1024
12	tc3-3-deleteHousekeepingParameterReportStructures		service-type-id=3 AND message-subtype-id=3	TC3-3-DeleteHousekeepingParameterReportStructures-Type		11	67
13	tc3-9-reportHousekeepingParameterReportStructures		service-type-id=3 AND message-subtype-id=9	TC3-9-ReportHousekeepingParameterReportStructures-Type		11	67
14	tc3-27-generateAOneShotReportForHousekeepingParameterReportStructures		service-type-id=3 AND message-subtype-id=27	TC3-27-GenerateAOneShotReportForHousekeepingParameterReportStructures-Type		11	67
15	tc3-29-appendParametersToAHousekeepingParameterReportStructure		service-type-id=3 AND message-subtype-id=29	TC3-29-AppendParametersToAHousekeepingParameterReportStructure-Type		16	1016
16	tc3-31-modifyTheCollectionIntervalOfHousekeepingParameterReportStructures		service-type-id=3 AND message-subtype-id=31	TC3-31-ModifyTheCollectionIntervalOfHousekeepingParameterReportStructures-Type		4	132
17	tc3-33-reportThePeriodicGenerationPropertiesOfHousekeepingParameterReportStructures		service-type-id=3 AND message-subtype-id=33	TC3-33-ReportThePeriodicGenerationPropertiesOfHousekeepingParameterReportStructures-Type		11	67
18	tc3-7-enableThePeriodicGenerationOfDiagnosticParameterReports		service-type-id=3 AND message-subtype-id=7	TC3-7-EnableThePeriodicGenerationOfDiagnosticParameterReports-Type		11	67
19	tc3-8-disableThePeriodicGenerationOfDiagnosticParameterReports		service-type-id=3 AND message-subtype-id=8	TC3-8-DisableThePeriodicGenerationOfDiagnosticParameterReports-Type		11	67
20	tc3-2-createADiagnosticParameterReportStructure		service-type-id=3 AND message-subtype-id=2	TC3-2-CreateADiagnosticParameterReportStructure-Type		34	2418
21	tc3-4-deleteDiagnosticParameterReportStructures		service-type-id=3 AND message-subtype-id=4	TC3-4-DeleteDiagnosticParameterReportStructures-Type		11	67
22	tc3-11-reportDiagnosticParameterReportStructures		service-type-id=3 AND message-subtype-id=11	TC3-11-ReportDiagnosticParameterReportStructures-Type		11	67
23	tc3-28-generateAOneShotReportForDiagnosticParameterReportStructures		service-type-id=3 AND message-subtype-id=28	TC3-28-GenerateAOneShotReportForDiagnosticParameterReportStructures-Type		11	67
24	tc3-30-appendParametersToADiagnosticParameterReportStructure		service-type-id=3 AND message-subtype-id=30	TC3-30-AppendParametersToADiagnosticParameterReportStructure-Type		18	2402
25	tc3-32-modifyTheCollectionIntervalOfDiagnosticParameterReportStructures		service-type-id=3 AND message-subtype-id=32	TC3-32-ModifyTheCollectionIntervalOfDiagnosticParameterReportStructures-Type		27	195
26	tc3-34-reportThePeriodicGenerationPropertiesOfDiagnosticParameterReportStructures		service-type-id=3 AND message-subtype-id=34	TC3-34-ReportThePeriodicGenerationPropertiesOfDiagnosticParameterReportStructures-Type		11	67
27	tc3-37-applyParameterFunctionalReportingConfigurations		service-type-id=3 AND message-subtype-id=37	TC3-37-ApplyParameterFunctionalReportingConfigurations-Type		19	75
28	tc3-38-createAParameterFunctionalReportingDefinition		service-type-id=3 AND message-subtype-id=38	TC3-38-CreateAParameterFunctionalReportingDefinition-Type		13	4829
29	tc3-39-deleteParameterFunctionalReportingDefinitions		service-type-id=3 AND message-subtype-id=39	TC3-39-DeleteParameterFunctionalReportingDefinitions-Type		11	67
30	tc3-40-reportParameterFunctionalReportingDefinitions		service-type-id=3 AND message-subtype-id=40	TC3-40-ReportParameterFunctionalReportingDefinitions-Type		4	68
31	tc3-42-addParameterReportDefinitionsToAParameterFunctionalReportingDefinition		service-type-id=3 AND message-subtype-id=42	TC3-42-AddParameterReportDefinitionsToAParameterFunctionalReportingDefinition-Type		24	2419
32	tc3-43-removeParameterReportDefinitionsFromAParameterFunctionalReportingDefinition		service-type-id=3 AND message-subtype-id=43	TC3-43-RemoveParameterReportDefinitionsFromAParameterFunctionalReportingDefinition-Type		33	1267
33	tc3-44-modifyThePeriodicGenerationPropertiesOfParameterReportDefinitionsOfAParameterFunctionalReportingDefinition		service-type-id=3 AND message-subtype-id=44	TC3-44-ModifyThePeriodicGenerationPropertiesOfParameterReportDefinitionsOfAParameterFunctionalReportingDefinition-Type		24	2419

PUS-C demonstration ICD

ModifyAParameterMonitoringDefinition-Type (SEQUENCE) ASN.1 ACN						Min: 9 bytes	Max: 11 bytes
No	Field	Comment	Present	Type	Constraint	Min Bits	Max Bits
1	pmonId	Possible values: ▪ *tailoringDemoPmonid* (0)	always	PMON-ID	(tailoringDemoPmonid)	8	8
2	monitoredParameterId	Possible values: ▪ *tailoringDemoMonitoredparameterid* (0)	always	MONITORED-PARAMETER-ID	(tailoringDemoMonitoredparameterid)	8	8
3	repetitionNumber		always	MONITORING-REPETITION-NUMBER	(0 .. 16383)	16	16
4	checkType		always	CHECK-TYPE		2	2
5	checkTypeDependentCriteria		always	CHECK-TYPE-DEPENDENT-CRITERIA		32	48

SDL modelling

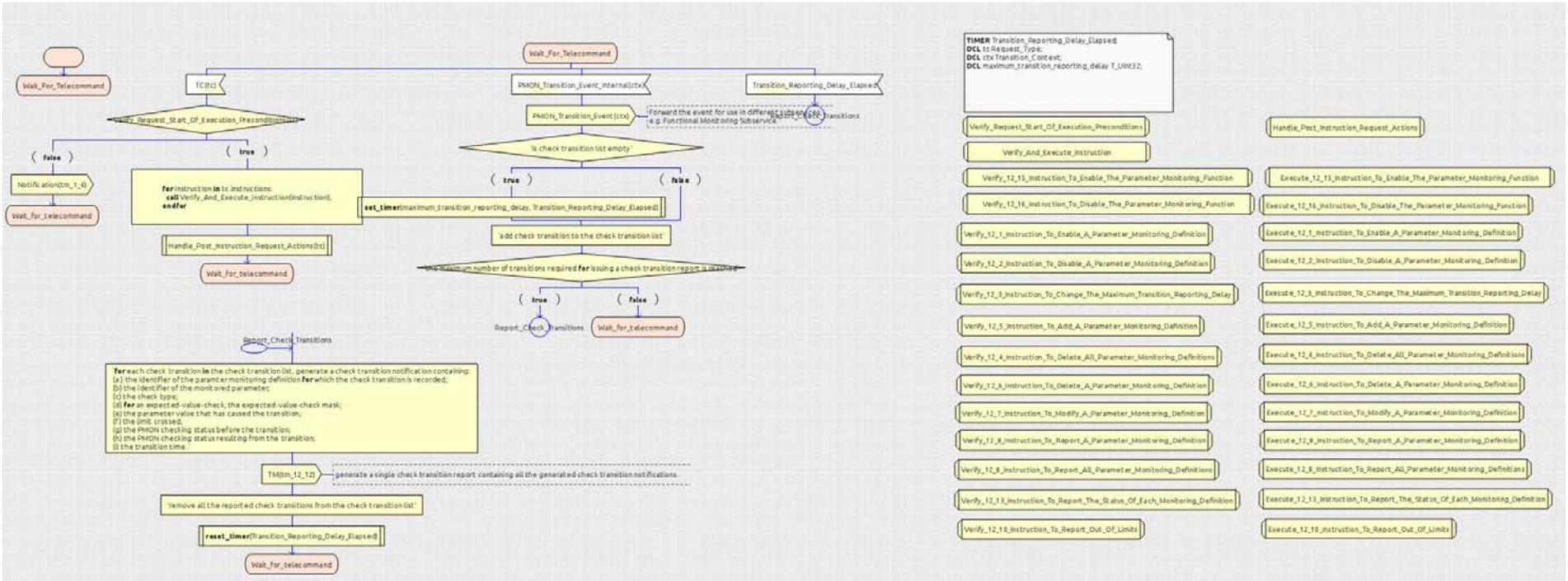
- Almost all PUS C services follow a very simple, practically stateless template



- Only a few services are more complex or interesting for SDL modelling
- The following services were modelled:
  - service 1 (very different from other services)
  - "generic PUS C service" (a template)
  - service 6
  - service 9
  - service 12

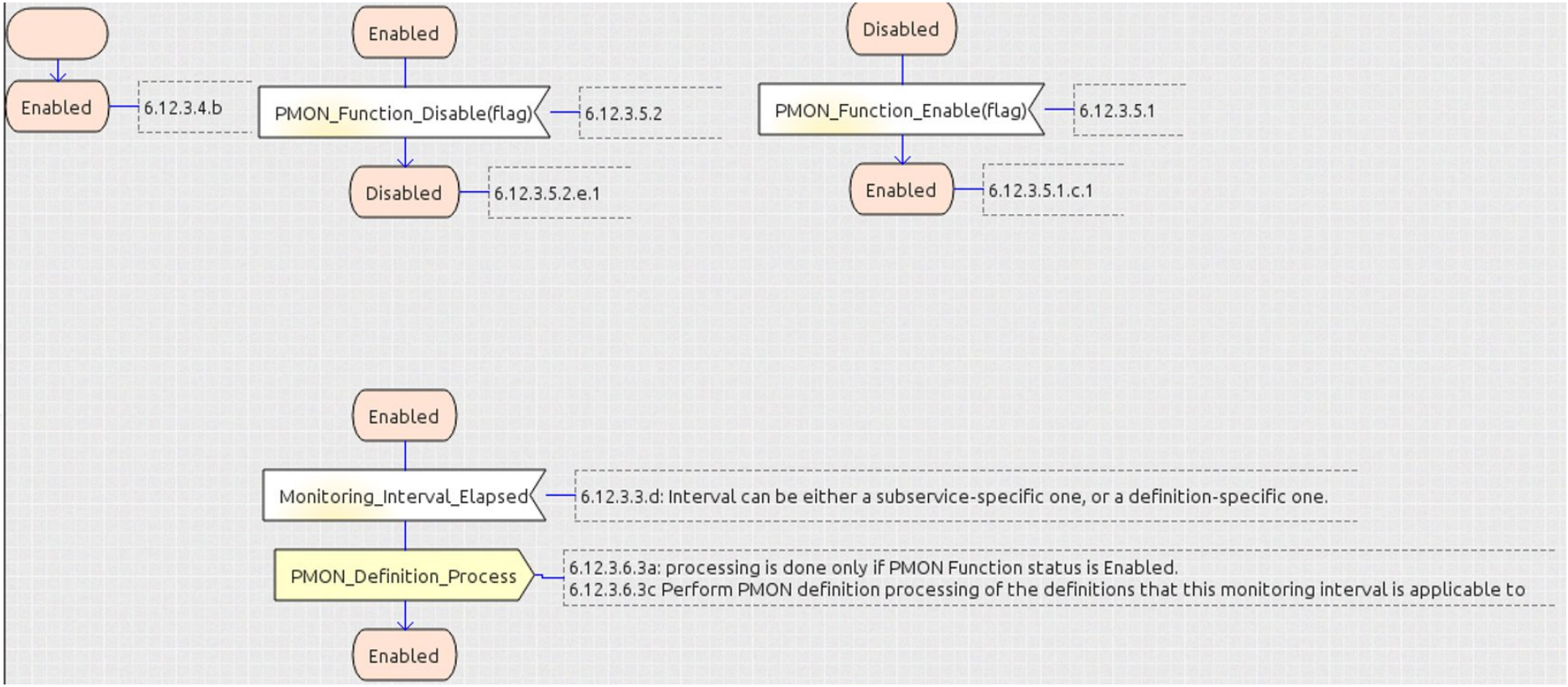
Parameter monitoring (in PUS 12)

- Follows the generic template, customized for readability
- Reacts to requests, transitions and reporting delay
- Implements high-level handling of all instructions



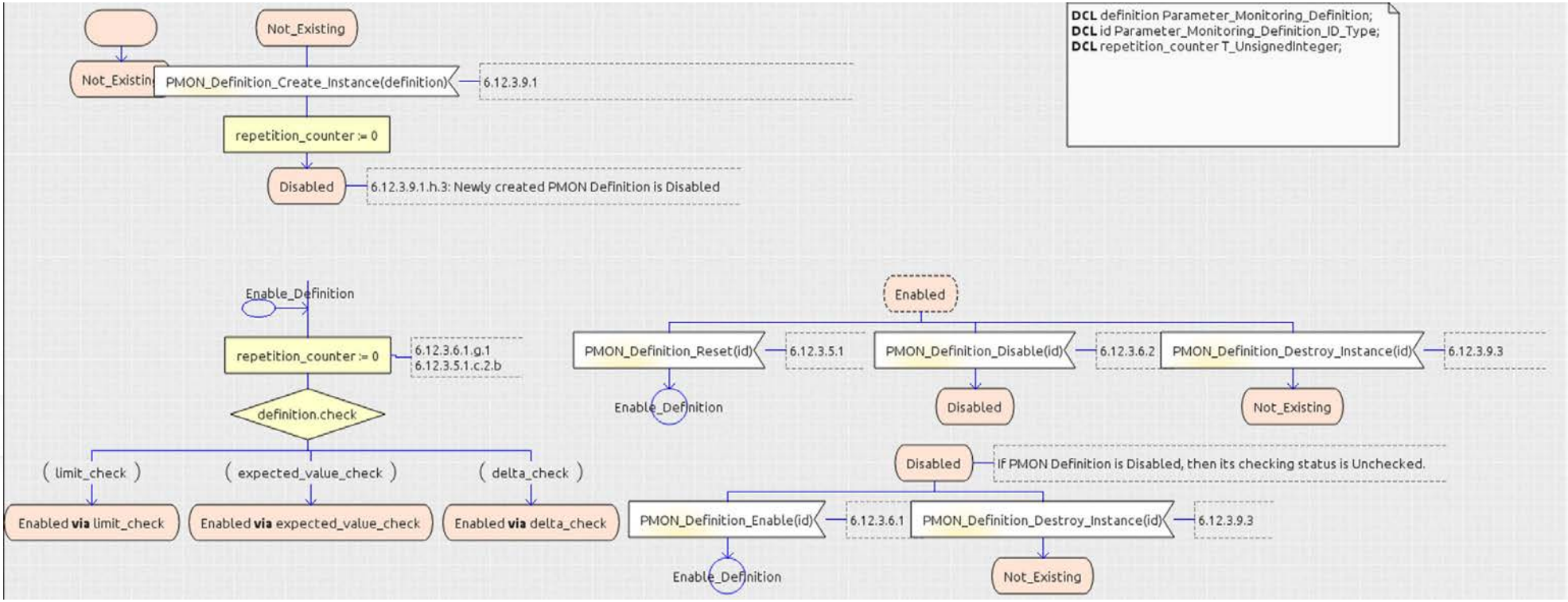
Parameter monitoring process

- Very simple
- Comments trace to ECSS requirements

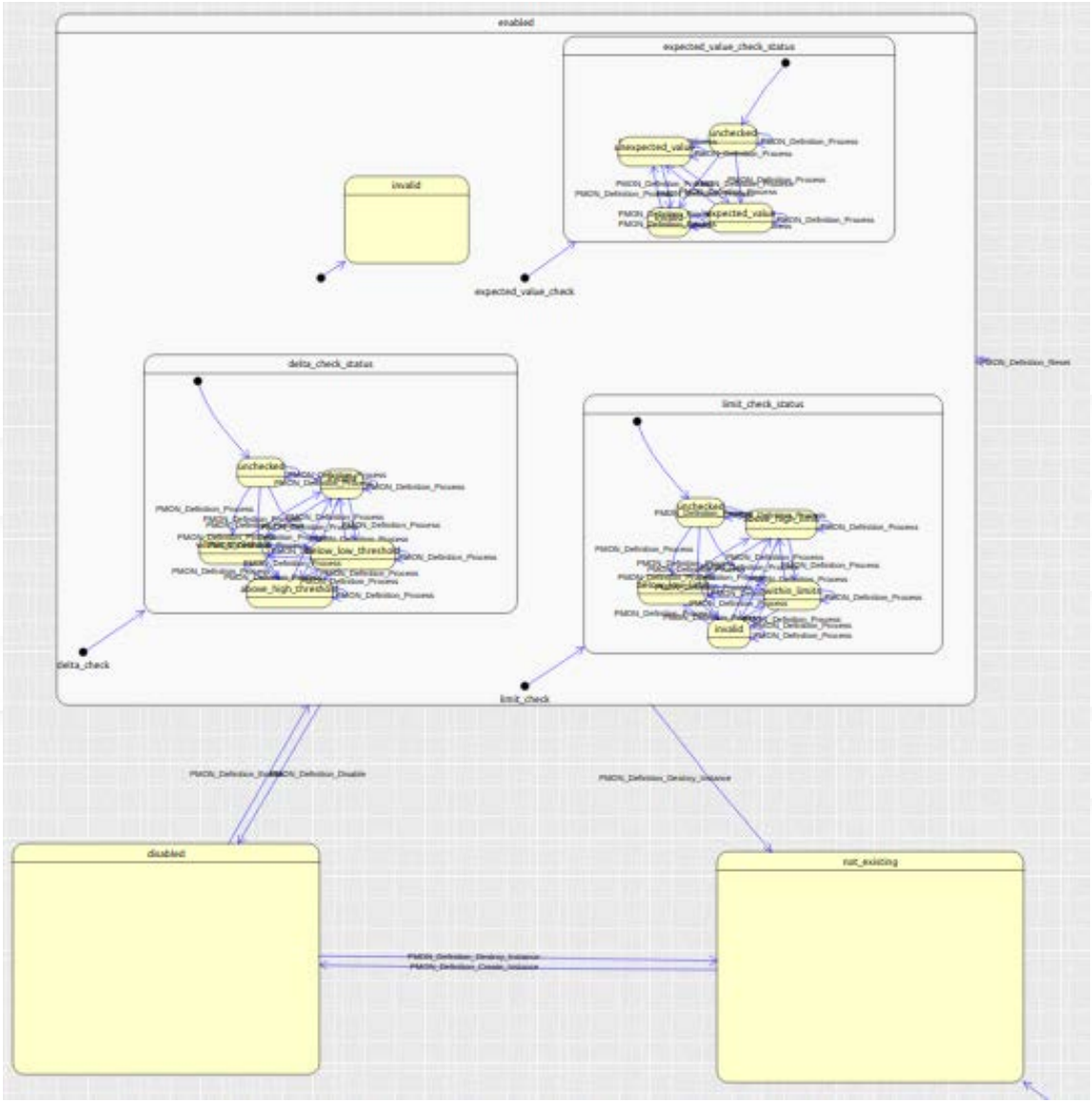


Parameter monitoring definition

- Quite complex, with nested states

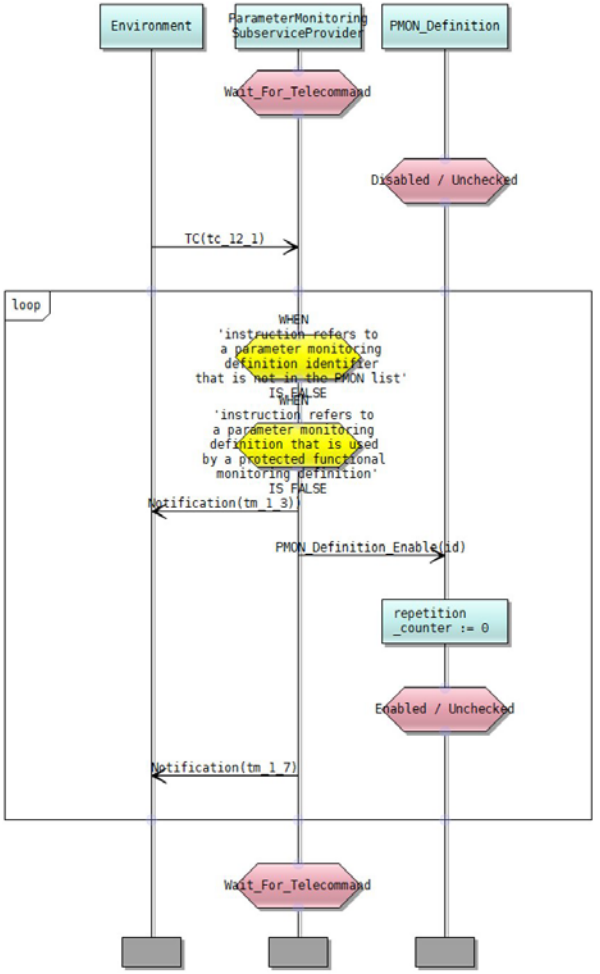


Parameter monitoring definition



MSC modelling

- MSC diagrams are based on SDL models: entities, states, signals, and critical (depending on the scenario) conditions, loops, actions correspond to their SDL equivalents
- A choice of the most relevant scenarios, each representing a particular state of conditions – one SDL diagram usually corresponds to multiple MSC diagrams



SDL and MSC modelling

- SDL and MSC modelling is complementary
- MSC modelling discovered issues in SDL models
- SDL models were improved based on the feedback
- Generation of certain Service 1 notifications is not properly defined in the standard
- Early adoption of SDL and MSC modelling during the standard preparation could improve the standard readability, quality and formalism

Toolset deployment – Proba 3

Use Case Validation using Proba 3 as a use case

- Install all required tools on an empty environment;
- Create a non-standard PUS service and add a non-standard subservice to a standard service;
- Tailor these services with mission parameters;
- Generate the corresponding ICD, PUS specification, and ASN.1 data.

Toolset deployment – Proba 3

Add a mission-specific subservice:

- Extend service 3 with HK compression subservice (Pocket)
- Provided capabilities:
  - Generate compressed housekeeping telemetry packet
  - Define Housekeeping Compression Configuration
  - Define Housekeeping Compression Reference Packet
  - Define Housekeeping Compression Packet Mask
  - Delete Housekeeping Compression Configuration
  - Enable Housekeeping Compression
  - Disable Housekeeping Compression
  - Generate new housekeeping compression mask
  - Send On-board generated housekeeping compression mask

Toolset deployment – Proba 3

Add a mission specific service: Reaction wheels management (S8 in Proba3)

- **Reaction wheels commanding subservice:**
  - Activate one reaction wheel
  - Activate three reaction wheels
  - Activate four reaction wheels
  - Deactivate one reaction wheel
  - Deactivate all reaction wheels
- **Reaction wheels FDIR subservice:**
  - Invalidate reaction wheel
  - Power-cycle reaction wheel
- **Reaction wheels Rate Sensors management subservice:**
  - Enable/disable rate sensor
  - Invalidate rate sensor
- **Reaction Wheels direct commanding subservice:**
  - Send direct command to reaction wheel
  - Generate reaction wheel report

Toolset deployment – Proba 3

All added capabilities could be inserted, and produces:

- An XML file containing the resulting population after tailoring, which can be imported into an empty PUS-C Gen database;
- The PUS-C Specification word document;
- The ICD (html document) after tailoring, generated from the ASN.1 data.

Toolset deployment - CoreSight

- R&D project for ESA (not flight software)
- Goal (summary): assess the use of CoreSight technology available in ARM SoCs for software execution tracing and analysis in the context of space applications
- Uses PUS-C for C&C: services 1, 6 and custom 192
- Much of the population was created by 2 engineers with no previous experience with the toolset, after a 2-hour training session (iteratively, with reviews)
- ESA's ASN1SCC v4 and DMT was used to create ICD, C TC/TM transcoders (for use in SW) and Python bindings (for use in integration tests)

Toolset deployment - CoreSight

- Some modifications were performed:
  - name format change from tcx-y-z to tc-x-y-z
  - addition of a CLI interface for PUS-ASN1-GEN
  - addition of the ability to generate ASN.1/ACN from multiple XML-based populations
- The last change is due to a usage optimization:
  - one population for each untailored service
  - one population containing project specific tailoring
- More manageable solution enabling parallel work and improving potential reuse
- Project is nearing completion; deployment is a success!

Toolset deployment - CoreSight

- Telecommands (part of ICD generated from ASN.1/ACN)

Telecommands (CHOICE) ASN.1 ACN — Min: 0 bytes Max: 8483 bytes

No	ACN Parameters [7]	Type
1	service-type-id	TYPE-ID
2	message-subtype-id	TYPE-ID

No	Field	Comment	Present	Type	Constraint	Min Bits	Max Bits
1	tc-6-1-loadObjectMemoryData		service-type-id=6 AND message-subtype-id=1	TC-6-1-LoadObjectMemoryData-Type		84	16511
2	tc-6-3-dumpObjectMemoryData		service-type-id=6 AND message-subtype-id=3	TC-6-3-DumpObjectMemoryData-Type		82	130
3	tc-6-17-checkAnObjectMemoryObject		service-type-id=6 AND message-subtype-id=17	TC-6-17-CheckAnObjectMemoryObject-Type		33	33
4	tc-192-1-startTrace		service-type-id=192 AND message-subtype-id=1	TC-192-1-StartTrace-Type		98	25554
5	tc-192-2-stopTrace		service-type-id=192 AND message-subtype-id=2	TC-192-2-StopTrace-Type		32	32
6	tc-192-3-resetTraceAcquisitions		service-type-id=192 AND message-subtype-id=3	TC-192-3-ResetTraceAcquisitions-Type	(0)	0	0
7	tc-192-4-takeATraceSnapshot		service-type-id=192 AND message-subtype-id=4	TC-192-4-TakeATraceSnapshot-Type		32	32
8	tc-192-16-startGatheringPerformanceCountersForProcess		service-type-id=192 AND message-subtype-id=16	TC-192-16-StartGatheringPerformanceCountersForProcess-Type		52	116
9	tc-192-20-configurePerformanceCountersStorageSize		service-type-id=192 AND message-subtype-id=20	TC-192-20-ConfigurePerformanceCountersStorageSize-Type		32	32
10	tc-192-17-stopGatheringPerformanceCountersForProcess		service-type-id=192 AND message-subtype-id=17	TC-192-17-StopGatheringPerformanceCountersForProcess-Type		32	32
11	tc-192-18-resetGatheringPerformanceCounters		service-type-id=192 AND message-subtype-id=18	TC-192-18-ResetGatheringPerformanceCounters-Type	(0)	0	0
12	tc-192-32-reportExecutionTimeStatisticsOfCodeRange		service-type-id=192 AND message-subtype-id=32	TC-192-32-ReportExecutionTimeStatisticsOfCodeRange-Type		160	160
13	tc-192-34-reportExecutionTimeStatisticsOfSymbol		service-type-id=192 AND message-subtype-id=34	TC-192-34-ReportExecutionTimeStatisticsOfSymbol-Type		40	2080
14	tc-192-36-reportTimeProfileOfCodeRange		service-type-id=192 AND message-subtype-id=36	TC-192-36-ReportTimeProfileOfCodeRange-Type		352	352
15	tc-192-38-reportTimeProfileOfSymbol		service-type-id=192 AND message-subtype-id=38	TC-192-38-ReportTimeProfileOfSymbol-Type		232	2272
16	tc-192-40-verifyThatTheExecutionTimeOfCodeRangeDidNotExceedAConfiguredValue		service-type-id=192 AND message-subtype-id=40	TC-192-40-VerifyThatTheExecutionTimeOfCodeRangeDidNotExceedAConfiguredValue-Type		224	224
17	tc-192-42-verifyThatTheExecutionTimeOfSymbolDidNotExceedAConfiguredValue		service-type-id=192 AND message-subtype-id=42	TC-192-42-VerifyThatTheExecutionTimeOfSymbolDidNotExceedAConfiguredValue-Type		104	2144
18	tc-192-44-verifyThatTheExecutionTimeProfileOfCodeRangeIsWithinAConfiguredRangeOfSimilarityToTheExpectedValueDistribution		service-type-id=192 AND message-subtype-id=44	TC-192-44-VerifyThatTheExecutionTimeProfileOfCodeRangeIsWithinAConfiguredRangeOfSimilarityToTheExpectedValueDistribution-Type		307	65939
19	tc-192-46-verifyThatTheExecutionTimeProfileOfSymbolIsWithinAConfiguredRangeOfSimilarityToTheExpectedValueDistribution		service-type-id=192 AND message-subtype-id=46	TC-192-46-VerifyThatTheExecutionTimeProfileOfSymbolIsWithinAConfiguredRangeOfSimilarityToTheExpectedValueDistribution-Type		187	67859

Toolset deployment - MSP

- TASTE enhancement project for ESA
- Goal (summary): implement support for MSP430 16-bit MCU target in TASTE
- Validation by a demo CubeSat-class software
- Uses PUS-C for C&C: services 1, 2, 3 and custom 222
- Demo SW is developed entirely in TASTE (SDL + C), with the generated ASN.1/ACN as input
- Runs on a flatsat built around MSP430FR5969 Launchpad

Toolset deployment - MSP

- Uses the same toolset version as the CoreSight project
- Very simple C&C – a single population
- Small amount of the available memory:
  - services are cut down to bare minimum
  - bare bone encapsulation instead of CCSDS
- First iteration took ~5h – from concept to ASN.1/ACN
- "DemoSat" is in progress:
  - communication has been established
  - service 3 works, 1 and 222 is in progress
  - MSP430 memory is a challenge

Toolset deployment - MSP

- Sample from the generated requirements (Pandoc):

6.4.3.2 Switch mode

1. The mode management subservice shall provide the capability to switch mode.

   NOTE The corresponding requests are of message type "TC[222,1] switch mode".

2. Each request to switch mode shall contain exactly one instruction to switch mode.
3. Each instruction to switch mode shall contain:
   1. satellite mode to switch to.
4. The mode management subservice shall reject any request to switch mode if any of the following conditions occurs:
   1. the requested transition is invalid;
   2. power supply voltage is invalid.
5. For each request to switch mode that is rejected, the mode management subservice shall generate a failed start of execution notification.
6. For each valid instruction to switch mode, the mode management subservice shall:
   1. switch to the desired mode.

Latest activities and way forward

Spacebel initiative to bridge the gap between the PUS-C gen toolset and their SRDB tooling:

- Output of PUS-Cgen: ASN.1 & ACN description of the PUS I/F;
- Output of SPB's SRDB tooling: SRDB component for on-board software (for integration in SCOS2000, for example).

Methodology:

- Use the ASN.1 abstract syntax tree as the model for TMTC I/F;
- Replace manual entries for TMTC syntax by references to this model (imported/referred using the XML AST representation).

Benefits:

- By construction consistency between software & SRDB;
- Methodology also applicable to older PUS versions

Conclusions

During the course of the PUS-C gen study, we have successfully:

- Reviewed and enriched an ORM model of the PUS foundation;
- Identified issues with the PUS-C standard and created the corresponding change requests;
- Generated a relational database corresponding to the PUS conceptual model;
- Developed a toolset relying on this relational database.

Conclusions

The developed toolset allows its users to:

- Create new services or subservices that are compliant by construction to the foundation;
- Create specific, PUS-related artifacts to tailor the PUS-C to any specific mission;
- Generate the related documentation (functional specification & ICDs);
- Generate ASN.1 definitions which can then be used to produce software code to encode & decode TM/TCs.

Conclusions

- The tooling developed during the activity proves the feasability of the automatic generation of such standards;
- The tools produced by this study have been successfully applied to a representative use case, and adopted on several other studies;
- Ongoing and future work may allow closing the gap with the SRDB representation of the TM/TC interface data.