asn.1

Language and tools ensuring consistency in data for space systems

Maxime Perrotin
ESA/ESTEC
TEC-SWE

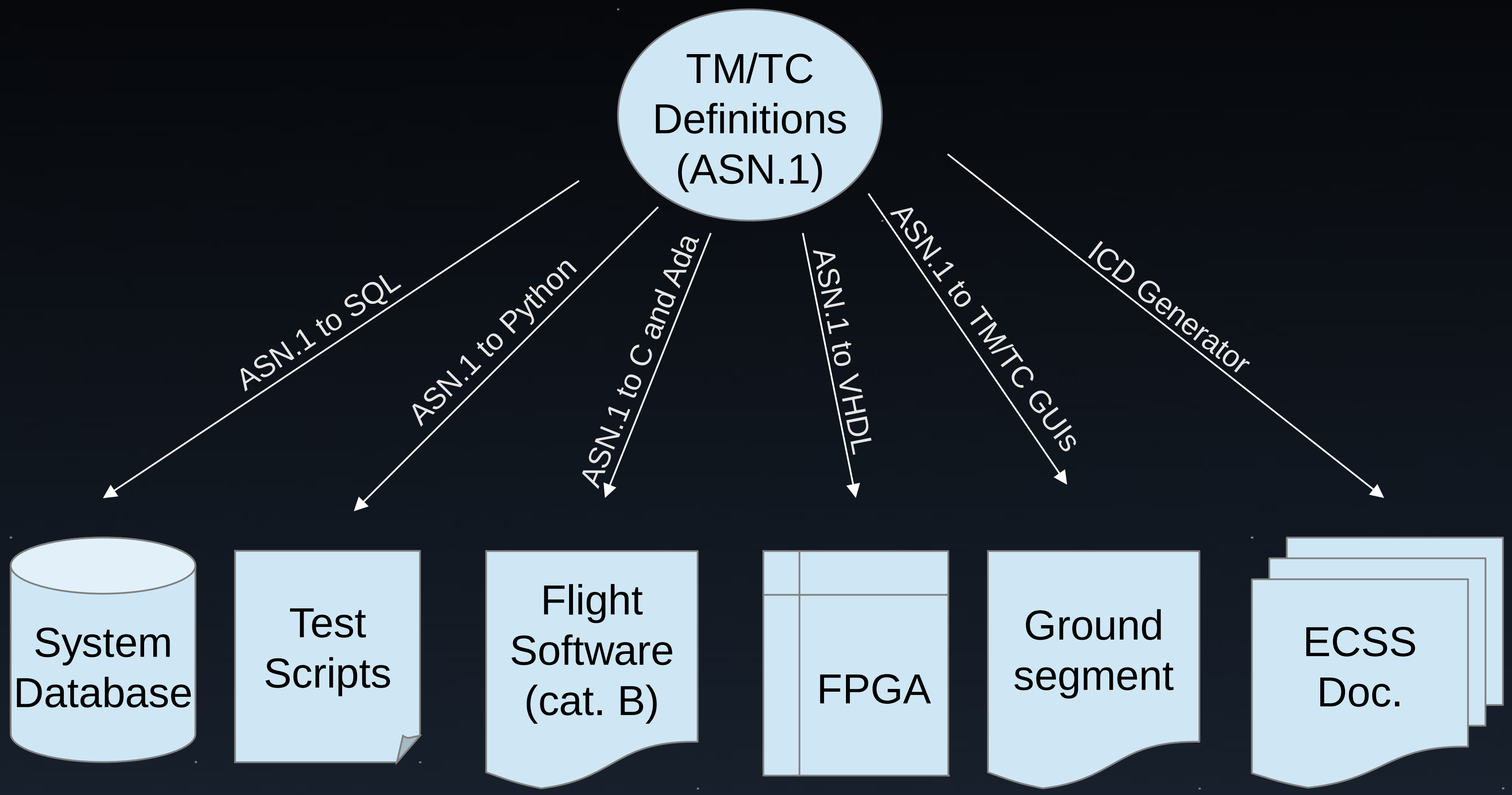
ASN.1

- International, widely used standard (ISO and ITU-T)
- Simple text notation for precise and complete **data type description**
- But with an added value : **the physical encoding rules** (compact binary encoding, endianness-neutral, but also XML encoding, legacy encoding specifications).
- Separate the encoding rules from the types specification

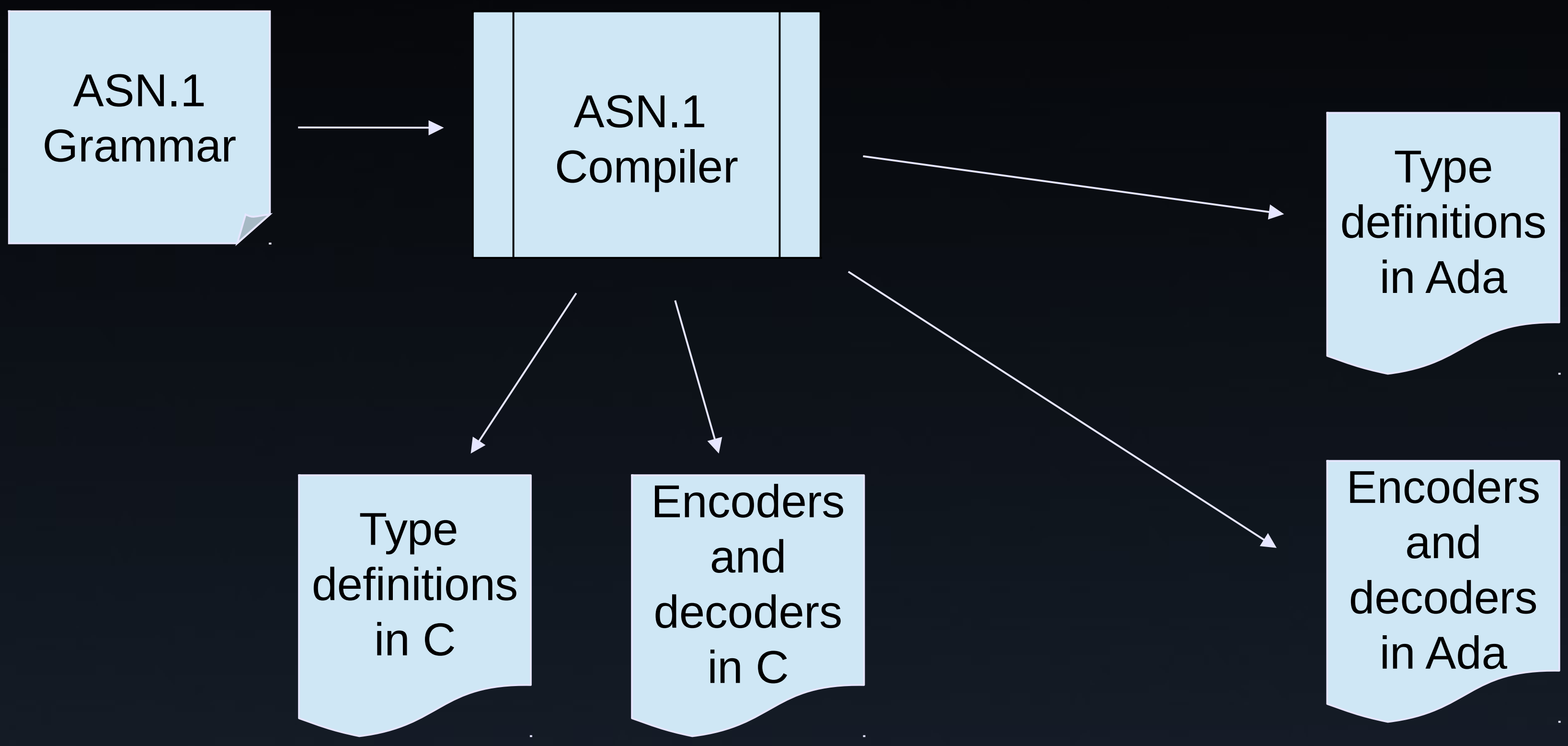
A very simple yet powerful syntax

```
Dataview DEFINITIONS ::= BEGIN

-- From simple types
Thruster-index ::= INTEGER (1 .. 10) -- Allowed: 1 to 10
Identifier ::= ENUMERATED { cpdu1, cpdu2 }

-- To complex data structures
TC ::= SEQUENCE {
    header PUS-header,
    payload Userdata,
    crc OPTIONAL
}

Userdata ::= CHOICE {
   tc-6-1 MemoryLoad,
   tc-6-2 ...
...
}

END
```

ASN.1 to ensure consistency



How does it work ?



ASN.1 philosophy

Field	Length (bits)	Values	Comment
Version	3	'000'	For the first version.
PDU type	1	'0' — File Directive '1' — File Data	
Direction	1	'0' — toward file receiver '1' — toward file sender	Used to perform PDU forwarding.
Transmission Mode	1	'0' — acknowledged '1' — unacknowledged	
CRC Flag	1	'0' — CRC not present '1' — CRC present	
Reserved for future use	1	set to '0'	
PDU Data field length	16		In octets.
Reserved for future use	1	set to '0'	
Length of entity IDs	3		Number of octets in entity ID less one; i.e., '0' means that entity ID is one octet. Applies to all entity IDs in the PDU header.
Reserved for future use	1	set to '0'	
Length of Transaction sequence number	3		Number of octets in sequence number less one; i.e., '0' means that sequence number is one octet.
Source entity ID	variable		Uniquely identifies the entity that originated the transaction.
Transaction sequence number	variable		Uniquely identifies the transaction, among all transactions originated by this entity.
Destination entity ID	variable		Uniquely identifies the entity that is the final destination of the transaction's metadata and file data.

These fields are not application semantics! They concern the binary encoding rules of the PDUs and should not be mixed with the protocol useful information.

ASN.1 philosophy

- **Keep only application-semantic data**
- Tools will generate encoders and decoders to add the other fields

```
Packet-ty ::=  SEQUENCE {
          version                       Version-ty,
          direction                     Direction-ty,
          transmission-mode             Transmission-mode-ty,
          crc-flag                      CRC-flag-ty,
          source-entity-id              Entity-id-ty,
          transaction-sequence-number   Transaction-sequence-number-ty,
          destination-entity-id         Entity-id-ty,
          data                          Datafield-ty
}

Version-ty ::= INTEGER (0..7)

Direction-ty ::= ENUMERATED { toward-file-receiver, toward-file-sender
}
```

Our ASN.1 compiler

- Developped and maintained by Neuropublic for ESA
- Free software
- Features:
  - Generates safe and optimized C and Spark/Ada code (fast, low memory footprint)
  - Automatically generates test cases for a given grammar
  - Generates ICDs documents in HTML format
  - Supports customized (legacy) encodings (e.g. PUS format)
  - API and tools to interface ASN.1 with SDL, Simulink, SCADE, VHDL, SQL, and Python

Legacy encodings

- ACN allows to specify legacy encodings
- It can be used to describe the binary format of PUS packets, leaving the interesting part only (payload data) in the ASN.1 side.

```
MySeq ::= SEQUENCE {
    alpha  INTEGER,
    gamma  REAL OPTIONAL
}
```

```
MySeq[] {
    alpha [],
    beta  BOOLEAN  [],
    gamma [present-when beta, encoding IEEE754-1985-64]
}
```

Apply it to the PUS (1)

```
-------------------------------
-- General Telecommand structure
-------------------------------

T-telecommand ::= SEQUENCE
{
    packet-header       TC-packetHeader,
    data-field-header   T-tc-dataFieldHeader,
    application-data    T-tc-applicationData,
    crc                 T-uint16
}
```

```
-------------------------------
-- Telecommand application data
-------------------------------

-- List of all available TCs categorized by their respective pus(-sub)types
-- Definition of actual payload data is done in respective Types below
-- In the ACN-file this type is used to automatically assign the pustype and subtype fields
-- in encoding and determine the packet type from pustype and subtype in decoding
-- Types defined as T-NULL have no actual payload data besides the fields
-- for pustype and subtype.
T-tc-applicationData ::= CHOICE
{
  tc-3-27-update-hk-period   TC-UPDATE-HK-PERIOD,
  tc-6-2-load-memory         TC-LOAD-MEMORY,
  tc-6-5-dump-memory         TC-DUMP-MEMORY,
  tc-6-9-check-memory        TC-CHECK-MEMORY,
  tc-6-129-transfer-image    TC-TRANSFER-IMAGE,
  tc-210-3-reset-dpu         T-NULL,             -- T-NULL is for TCs which don't have any applicationData
  tc-210-4-enable-watchdog   T-NULL,                     -- but only service type and subtype. Still they have to
  tc-210-5-disable-watchdog  T-NULL,             -- Appear in the list of valid commands. T-NULL ensures that 0 bits will be encoded
  tc-210-6-boot-iasw         TC-BOOT-IASW,
  tc-197-2-report-boot       T-NULL
}
```

Apply it to the PUS (2)

```
-- Table which maps the pusType and subtype to the corresponding
-- packet payload data
T-tc-applicationData<T-uint8:pusType, T-uint8:pusSubType> []
{
    tc-3-27-update-hk-period  [present-when pusType==  3 pusSubType== 27 ],
    tc-6-2-load-memory        [present-when pusType==  6 pusSubType==  2 ],
    tc-6-5-dump-memory        [present-when pusType==  6 pusSubType==  5 ],
    tc-6-9-check-memory       [present-when pusType==  6 pusSubType==  9 ],
    tc-6-129-transfer-image   [present-when pusType==  6 pusSubType==129 ],
    tc-210-3-reset-dpu        [present-when pusType==210 pusSubType==  3 ],
    tc-210-4-enable-watchdog  [present-when pusType==210 pusSubType==  4 ],
    tc-210-5-disable-watchdog [present-when pus
    tc-210-6-boot-iasw        [present-when pus
    tc-197-2-report-boot      [present-when pus
}
```

T-tc-applicationData(CHOICE) ASN.1 ACN	min = 0 bytes	max = 1010 bytes

=============================
Telecommand application data
=============================
List of all available TCs categorized by their respective pus(-sub)types
Definition of actual payload data is done in respective Types below
In the ACN-file this type is used to automatically assign the pustype and subtype fields
in encoding and determine the packet type from pustype and subtype in decoding
Types defined as T-NULL have no actual payload data besides the fields
for pustype and subtype.

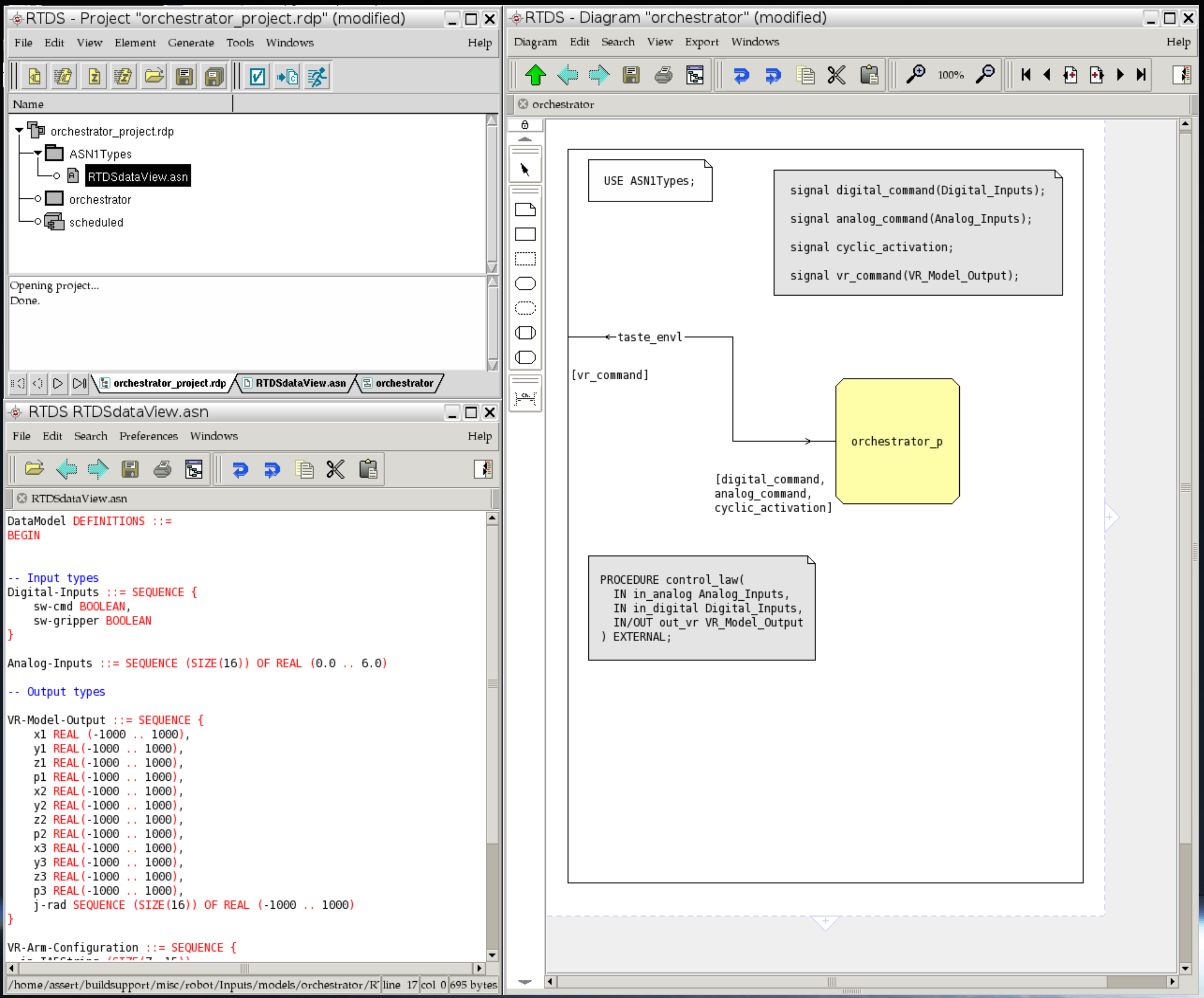
No	ACN Parameters what is this?	Type
1	pusType	T-uint8
2	pusSubType	T-uint8

No	Field	Comment	Present	Type	Constraint	Min Length (bits)	Max Length (bits)
1	tc-3-27-update-hk-period		pusType=3 AND pusSubType=27	TC-UPDATE-HK-PERIOD	N.A.	32	32
2	tc-6-2-load-memory		pusType=6 AND pusSubType=2	TC-LOAD-MEMORY	N.A.	112	8080
3	tc-6-5-dump-memory		pusType=6 AND pusSubType=5	TC-DUMP-MEMORY	N.A.	80	80
4	tc-6-9-check-memory		pusType=6 AND pusSubType=9	TC-CHECK-MEMORY	N.A.	72	72
5	tc-6-129-transfer-image		pusType=6 AND pusSubType=129	TC-TRANSFER-IMAGE	N.A.	80	80
6	tc-210-3-reset-dpu		pusType=210 AND pusSubType=3	T-NULL	N.A.	0	0
7	tc-210-4-enable-watchdog		pusType=210 AND pusSubType=4	T-NULL	N.A.	0	0
8	tc-210-5-disable-watchdog		pusType=210 AND pusSubType=5	T-NULL	N.A.	0	0
9	tc-210-6-boot-iasw		pusType=210 AND pusSubType=6	TC-BOOT-IASW	N.A.	80	80
10	tc-197-2-report-boot		pusType=197 AND pusSubType=2	T-NULL	N.A.	0	0

And use the code

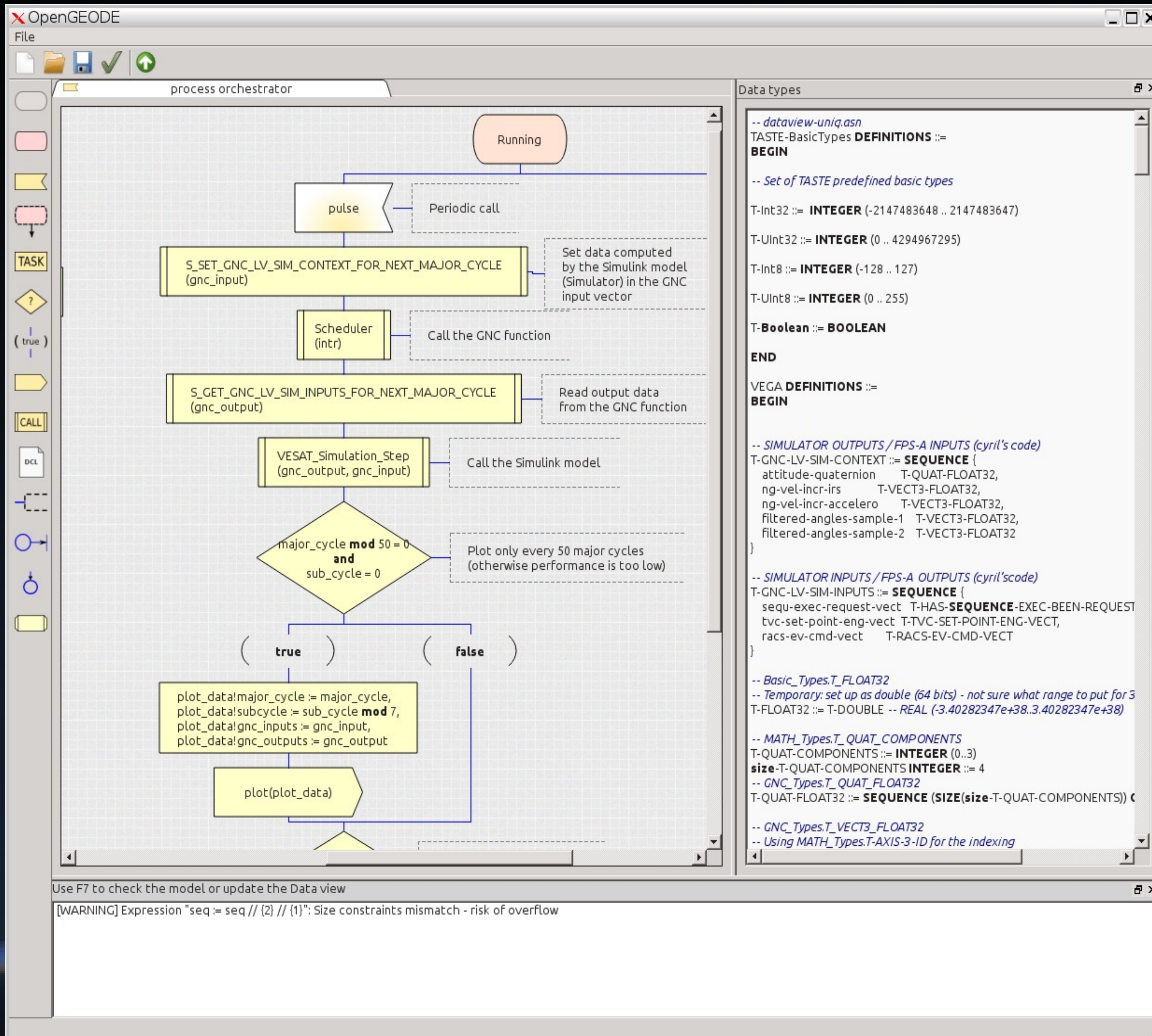
```c
typedef struct {
    enum {
        T_tc_applicationData_NONE,
        tc_3_27_update_hk_period_PRESENT,
        tc_6_2_load_memory_PRESENT,
        tc_6_5_dump_memory_PRESENT,
        tc_6_9_check_memory_PRESENT,
        tc_6_129_transfer_image_PRESENT,
        tc_210_3_reset_dpu_PRESENT,
        tc_210_4_enable_watchdog_PRESENT,
        tc_210_5_disable_watchdog_PRESENT,
        tc_210_6_boot_iasw_PRESENT,
        tc_197_2_report_boot_PRESENT
    } kind;
    union {
        TC_UPDATE_HK_PERIOD tc_3_27_update_hk_period;
        TC_LOAD_MEMORY tc_6_2_load_memory;
        TC_DUMP_MEMORY tc_6_5_dump_memory;
        TC_CHECK_MEMORY tc_6_9_check_memory;
        TC_TRANSFER_IMAGE tc_6_129_transfer_image;
        T_NULL tc_210_3_reset_dpu;
        T_NULL tc_210_4_enable_watchdog;
        T_NULL tc_210_5_disable_watchdog;
        TC_BOOT_IASW tc_210_6_boot_iasw;
        T_NULL tc_197_2_report_boot;
    } u;
} T_tc_applicationData;

#define T_tc_applicationData_REQUIRED_BYTES_FOR_ENCODING       1007
#define T_tc_applicationData_REQUIRED_BITS_FOR_ENCODING        8049
#define T_tc_applicationData_REQUIRED_BYTES_FOR_ACN_ENCODING   1010
#define T_tc_applicationData_REQUIRED_BITS_FOR_ACN_ENCODING    8080
#define T_tc_applicationData_REQUIRED_BYTES_FOR_XER_ENCODING   2272

void T_tc_applicationData_Initialize(T_tc_applicationData* pVal);
flag T_tc_applicationData_IsConstraintValid(const T_tc_applicationData* val, int* pErrCode);
flag T_tc_applicationData_ACN_Encode(const T_tc_applicationData* val, BitStream* pBitStrm, int* pErrCode, flag bCheckConstraints);
flag T_tc_applicationData_ACN_Decode(T_tc_applicationData* pVal, BitStream* pBitStrm, int* pErrCode, T_uint8 pusType, T_uint8 pusSubType);
#ifndef ERR_T_tc_applicationData_unknown_choice_index
#define ERR_T_tc_applicationData_unknown_choice_index           1037  /**/
#endif
```

SDL and ASN.1

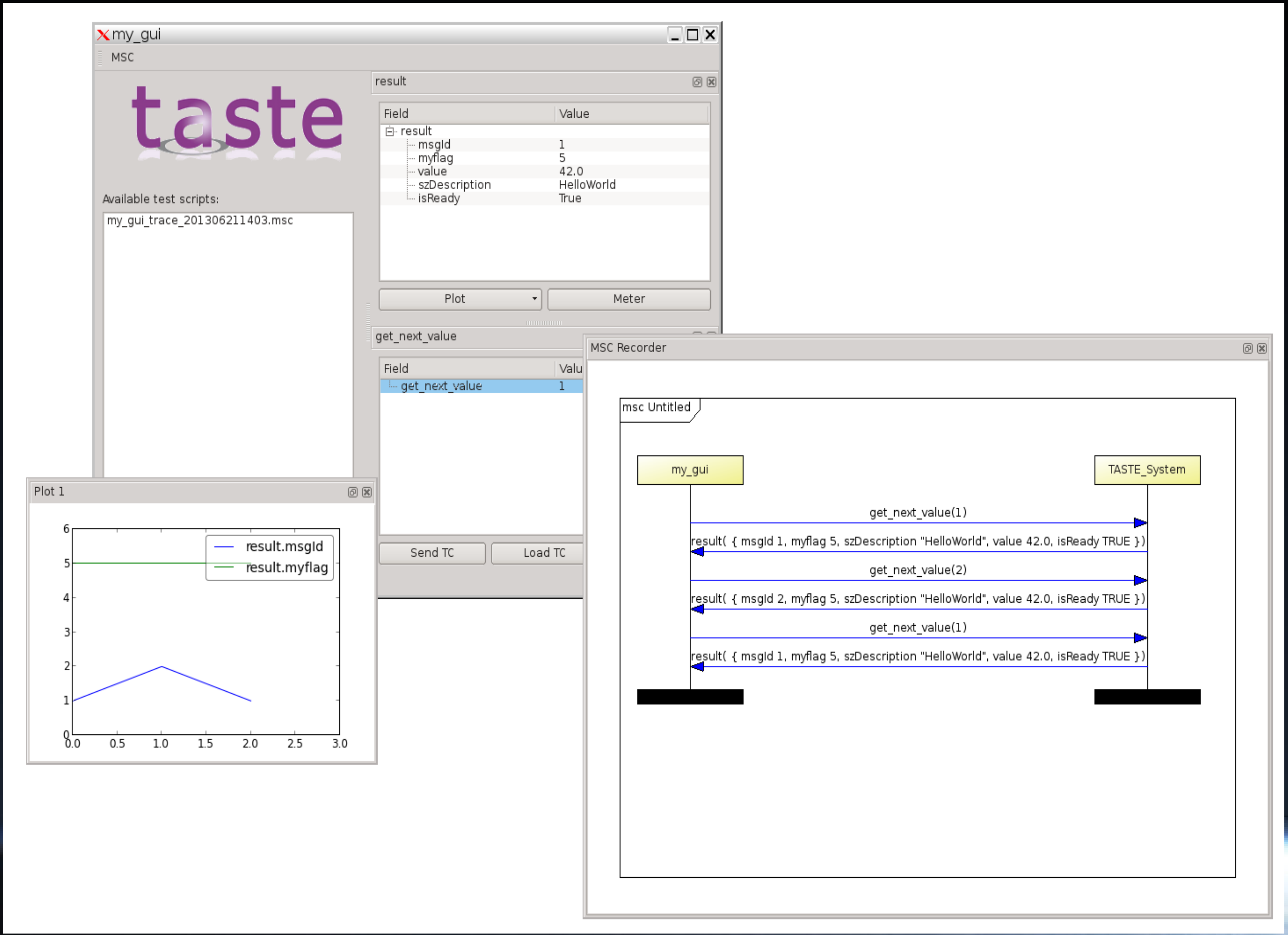


RTDS - Project "orchestrator_project.rdp" (modified)
File Edit View Element Generate Tools Windows Help
Name
orchestrator_project.rdp
ASN1Types
RTDSdataView.asn
orchestrator
scheduled
Opening project...
Done.
orchestrator_project.rdp
RTDSdataView.asn
orchestrator
RTDS RTDSdataView.asn
File Edit Search Preferences Windows Help
RTDSdataView.asn
DataModel DEFINITIONS ::=
BEGIN
-- Input types
Digital-Inputs ::= SEQUENCE {
sw-cmd BOOLEAN,
sw-gripper BOOLEAN
}
Analog-Inputs ::= SEQUENCE (SIZE(16)) OF REAL (0.0 .. 6.0)
-- Output types
VR-Model-Output ::= SEQUENCE {
x1 REAL (-1000 .. 1000),
y1 REAL(-1000 .. 1000),
z1 REAL(-1000 .. 1000),
p1 REAL(-1000 .. 1000),
x2 REAL(-1000 .. 1000),
y2 REAL(-1000 .. 1000),
z2 REAL(-1000 .. 1000),
p2 REAL(-1000 .. 1000),
x3 REAL(-1000 .. 1000),
y3 REAL(-1000 .. 1000),
z3 REAL(-1000 .. 1000),
p3 REAL(-1000 .. 1000),
j-rad SEQUENCE (SIZE(16)) OF REAL (-1000 .. 1000)
}
VR-Arm-Configuration ::= SEQUENCE {
/home/assert/buildsupport/misc/robot/Inputs/models/orchestrator/R line 17 col 0 695 bytes
RTDS - Diagram "orchestrator" (modified)
Diagram Edit Search View Export Windows Help
100%
orchestrator
USE ASN1Types;
signal digital_command(Digital_Inputs);
signal analog_command(Analog_Inputs);
signal cyclic_activation;
signal vr_command(VR_Model_Output);
taste_envl
[vr_command]
orchestrator_p
[digital_command,
analog_command,
cyclic_activation]
PROCEDURE control_law(
IN in_analog Analog_Inputs,
IN in_digital Digital_Inputs,
IN/OUT out_vr VR_Model_Output
) EXTERNAL;

SDL, MSC and ASN.1

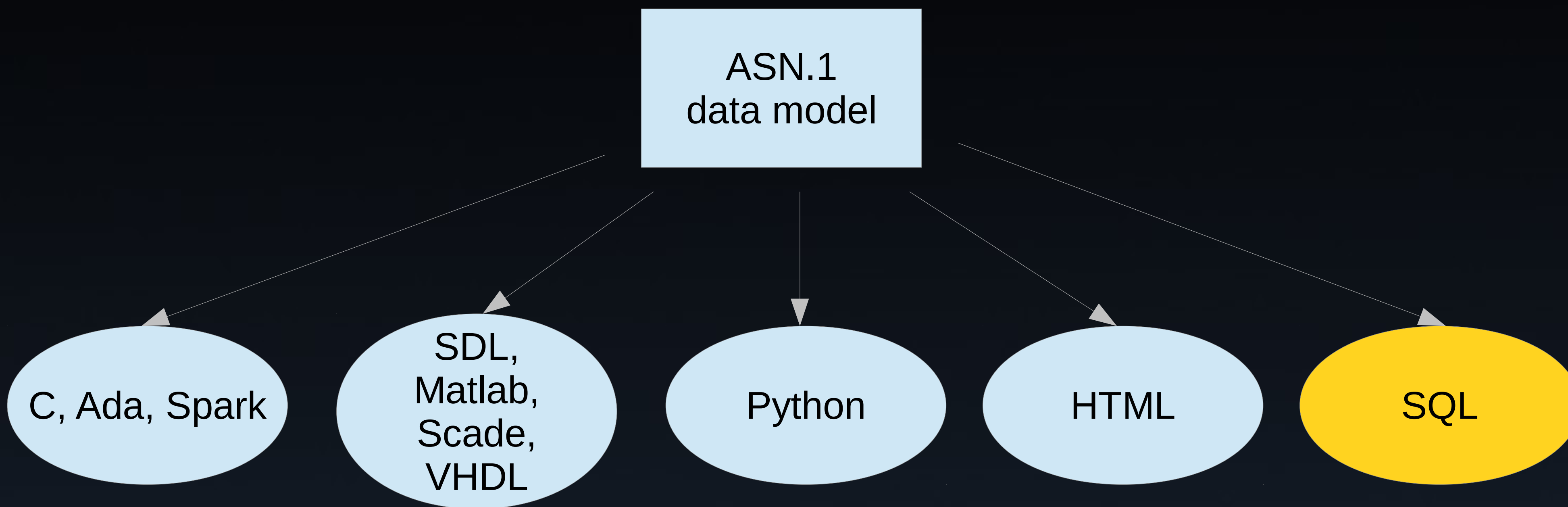


MSC and ASN.1



my_gui
MSC
taste
Available test scripts:
my_gui_trace_201306211403.msc
result
Field
Value
result
msgId 1
myflag 5
value 42.0
szDescription HelloWorld
isReady True
Plot
Meter
get_next_value
Field
Valu
get_next_value 1
Send TC
Load TC
Plot 1
result.msgId
result.myflag
MSC Recorder
msc Untitled
my_gui
TASTE_System
get_next_value(1)
result({ msgId 1, myflag 5, szDescription "HelloWorld", value 42.0, isReady TRUE })
get_next_value(2)
result({ msgId 2, myflag 5, szDescription "HelloWorld", value 42.0, isReady TRUE })
get_next_value(1)
result({ msgId 1, myflag 5, szDescription "HelloWorld", value 42.0, isReady TRUE })

ASN.1 to SQL / Working with databases



TASTE relies on ASN.1 to ensure consistency of data at each level of the process :
Engineering, processing, testing, documentation, communication, data storage and retrieval.

ASN.1 to SQL magic

- Use the same ASN.1 model to create SQL schemas ⟶ keep consistency (one SQL table per ASN.1 data type is created by the toolchain, automatically)
- Use case : telecommand/telemetry storage
  - Describe TM/TC data format in ASN.1 and ACN
  - Use C/Ada binary encoder/decoders in flight code
  - Use ICD generator to document format at binary level
  - Pick TC/Store TM in the SQL database for post-processing – field format is correct by construction
- Very flexible : using SQLAlchemy to be compatible with Oracle, SQLite, PostgreSQL...
- Python interface

A simple API

```
MyInt ::= INTEGER (0..20)
```

```python
# Can work with any DB. Here is an example with PostgreSQL
engine = create_engine(
    'postgresql+psycopg2://taste:tastedb@localhost/test', echo=False)

# Create data using the ASN.1 Python API
a = MyInt()
a.Set(5)

# Add the value to the SQL table called MyInt
aa1 = MyInt_SQL(a)
aid1 = aa1.save(session)
```

A simple API – Retrieve data

```python
# Data is retrieved using SQL queries, or SQLAlchemy API

# Retrieve ALL records in the MyInt table
all_values = self.session.query(MyInt_SQL)

for record in all_values:
    # The magic : data is transparently converted back to ASN.1
    print record.asn1.Get()
```

Query data with the full power of databases. It will be converted automatically to ASN.1 structures.

Use case :
Query all TC with type=XX and subtype=YY (1 line of code)
Select the ones you are interested in
Encode them with ASN.1/ACN to a PUS packet (1 line of code)
Send them to the satellite (1 line of code)

Check the results

- Demo of the complete features in `/home/assert/tool-src/DMT/tests-sqlalchemy`
- Run `make` (password for the db is *tastedb*)
- Run pgadmin3