Data Modelling with ASN.1 for Space Applications

ESA/ESTEC frame contract n° 4000104809

Thanassis Tsiodras, Dr.-Ing
NeuroPublic S.A.
ASN.1? What is that?

It's a "secret" weapon of the aeronautical, security and telecommunication domains - a simple language describing data structures, and offering multiple ways (trade-offs between CPU usage/space) to encode them:

```plaintext
DataView DEFINITIONS ::= BEGIN
T-THRUSTER-INDEX ::= INTEGER( 1 .. 10 ) -- Allowed: 1 to 10
T-REAL ::= REAL( -10000.0 .. 10000.0 ) -- Allowed value range
T-ACCELERATION ::= SEQUENCE (SIZE(3)) OF T-REAL -- Array of 3
T-ACS-CMD ::= SEQUENCE {
  set-thruster-on BOOLEAN,
  set-thruster-index T-THRUSTER-INDEX,
  set-thruster-data T-ACCELERATION
}
END
```
ASN.1? What is that?

ASN.1 compilers automatically generate your messages' type definitions, as well as your encoders and decoders – you don't write (and test!) them manually. You can also easily choose between encoding trade-offs (e.g. PER: packed encodes but more CPU, etc)
Not so secret, really

• You all used it today. At least 100 times.
• OK, not you - your phone. “Dear local GSM cell tower, I am alive and well, you can find me here”.
• Your browser used it when you accessed any HTTPS-enabled site from your laptop / tablet / ...
• Your bank used it to send information about your account's balance yesterday.
• The local telecom provider in Noordwijk (Vodafone NL) used it to send your mobile phone's roaming charges to your home provider (e.g. Vodafone DEU)
• Etc... Billions of ASN.1 msgs exchanged per day...
Airplanes use ASN.1 for Air Traffic Control => ESA became interested, and a study with Astrium back in 2008 showed many potential ASN.1 space applications...

It's not just encoders and decoders. We can automatically generate *many* things from ASN.1 – because the type information “drives” many things.

For starters: why write Interface Control Documents (ICDs) in Word/Excel? They can be automatically generated (with hyperlinks, too) from ASN.1 specs...
### Message(SEQUENCE) **ASN.1**

<table>
<thead>
<tr>
<th>No</th>
<th>Field</th>
<th>Comment</th>
<th>Optional</th>
<th>Type</th>
<th>Constraint</th>
<th>Min Length (bits)</th>
<th>Max Length (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Preamble</td>
<td>Special field used by PER to indicate the presence/absence of optional and default fields.</td>
<td>No</td>
<td>Bit mask</td>
<td>N.A.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- bit1 == 1 = <strong>isReady</strong> is present</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>msgId</td>
<td></td>
<td>No</td>
<td>INTEGER</td>
<td>(0 .. 255)</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>myflag</td>
<td></td>
<td>No</td>
<td>INTEGER</td>
<td>(0 .. 1000)</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>value</td>
<td></td>
<td>No</td>
<td>REAL</td>
<td>(0.000000E+000 .. 9.999000E+003)</td>
<td>8</td>
<td>104</td>
</tr>
<tr>
<td>5</td>
<td>szDescription</td>
<td></td>
<td>No</td>
<td>OCTET-STRING</td>
<td>(SIZE(10))</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>6</td>
<td>isReady</td>
<td></td>
<td>Def</td>
<td>BOOLEAN</td>
<td>N.A.</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### File: DataView.asn

```asn1
MY-MODULE DEFINITIONS AUTOMATIC TAGS ::= BEGIN
Message ::= SEQUENCE {
    msgId INTEGER (0..255),
    myflag INTEGER (0..1000),
    value REAL (0.0..9999.0),
    szDescription OCTET STRING (SIZE(10)),
    isReady BOOLEAN DEFAULT TRUE
}
```

• Automatically generated runtime “translators” of messages between code written in different languages (C, Ada, Python) and/or different tools (Simulink/RTW, SCADE, PragmaDev RTDS, etc)

```c
typedef struct {
    GU_RG_51 10 fd_height;
    GU_RG_50 9  fd_latitude;
    GU_RG_49 8  fd_longitude;
    GU_SEQOF_52 11 fd_subtypearray;
} GU_T_POS;
```

```c
typedef struct {
    real_T longitude;
    real_T latitude;
    real_T height;
    Subtypearray_type subtypearray;
} T_POS;
```

• You write code in Simulink/RTW and “glue” it to your projects manually? There's no need to suffer that!
DATA MODEL 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),
}  

VR-Arm Configuration ::= SEQUENCE {
  ...
ASN.1 in space – TM/TC GUIs, tests (3/6)

- TM/TC GUIs: we generate them automatically since 2010 (i.e. 0% manually written code), and they allow us to communicate with our running systems - and both monitor and control them easily.

- We can also watch (and graph) the TM/TC message exchanges in real-time MSC diagrams – again, the necessary code is generated automatically.

- The system's integration tests – why not write them in a nice scripting language? We automatically generate Python mappings for your ASN.1 messages – so system testing can be scripted, added to nightly regression-checks, etc.
Databases – automatically store your ASN.1 messages (TM/TC, whatever)

SQL mappers – they map your mission's ASN.1 messages to semantically identical database tables' definitions.

But that's not all - we also generate runtime mappers for Python, addressing all major open-source database engines: If you use SQLite, MySQL or PostgreSQL, you can store and restore your message data with one-liners... No need for manual tinkering of any sort.
The SQL mapper generates a table for each ASN.1 type, with a “sequence” (autoincrement-ed) primary key.

For primitive types (INTEGER, REAL, etc) the table has a “data” field of the appropriate type, with the necessary constraints:

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

```
CREATE TABLE "MyInt"
(
    iid serial NOT NULL,
    data integer NOT NULL,
    CONSTRAINT "MyInt_pkey" PRIMARY KEY (iid),
    CONSTRAINT "MyInt_data_check" CHECK (data >= 0 AND data <= 20)
)
```
- For SEQUENCE types (records), the mapper generates detail tables, and adds appropriate foreign keys.
- For SEQUENCE OF types (arrays), the mapper generates index fields as well.

```
CREATE TABLE "MySeq"
(
  iid serial NOT NULL,
  "fk_anInt_iid" integer NOT NULL,
  "fk_anotherInt_iid" integer NOT NULL,
  CONSTRAINT "MySeq_pkey" PRIMARY KEY (iid),
  CONSTRAINT "MySeq_fk_anInt_iid_fkey"
    FOREIGN KEY ("fk_anInt_iid")
    REFERENCES "MyInt" (iid),
  CONSTRAINT "MySeq_fk_anotherInt_iid_fkey"
    FOREIGN KEY ("fk_anotherInt_iid")
    REFERENCES "My2ndInt" (iid)
) 
```
class TypeNested_SQL(Base):
    __tablename__ = 'TypeNested'
    __table_args__ = (UniqueConstraint('iid'),)
    iid = Column(Integer, primary_key=True)

@staticmethod
def loadFromDB(session, iid):
    return session.query(
        TypeNested_SQL).filter(TypeNested_SQL.iid == iid).first()

@property
def asnl(self):
    if hasattr(self, '_cache'):
        return self._cache
    pyObj = TypeNested()
    self,assignToASN1object(pyObj)
    self._cache = pyObj
    return pyObj

def assignToASN1object(self, pyObj):
    state = pyObj.GetState()
    pyObj.Reset(state)
    self.intVal.assignToASN1object(pyObj.intVal)
    pyObj.Reset(state)
    self.int2Val.assignToASN1object(pyObj.int2Val)
    pyObj.Reset(state)
    self.int3Val.assignToASN1object(pyObj.int3Val)
    pyObj.Reset(state)
    self.intArray.assignToASN1object(pyObj.intArray)
    pyObj.Reset(state)
    self.realArray.assignToASN1object(pyObj.realArray)
    pyObj.Reset(state)
    self.octStrArray.assignToASN1object(pyObj.octStrArray)
    pyObj.Reset(state)
    self.boolArray.assignToASN1object(pyObj.boolArray)
    pyObj.Reset(state)
    self.enumArray.assignToASN1object(pyObj.enumArray)
    pyObj.Reset(state)
    self.enumValue.assignToASN1object(pyObj.enumValue)
    pyObj.Reset(state)
SELECT * FROM test.MySeq;
But... is it safe? This is safety-critical code!

Meet our compiler (https://github.com/ttsiodras/asn1scc): Open-source, written in F# (OCaml-derivative), a language where many programmer errors are caught at compile-time (option types, pattern matching on type forms, etc). Translation: a compiler where large categories of errors are impossible.

It generates code for C/C++ and SPARK/Ada. Well, SPARK encoders and decoders come with code verification – the message encoders and decoders are proven:

```
PROCEDURE BitStream_AppendByte (  
  Strm       : IN OUT BitStream;  
  ByteValue  : Unsigned_8;  
  Negate     : IN Boolean) IS  
  
  --# pre Strm.CurrentBit+8>Strm.Data'SIZE+1  
  ByteVal   : Unsigned_8 := ByteValue;  
  BEGIN  
    IF Negate THEN
```
ASN.1 in space – Safety Critical (5/6)

• Automatically generated test cases for your ASN.1 grammars, that provide 100% coverage of the code (gcov)
• No dynamic memory (heap) or syscalls – portable code
• Run-time library (RTL) is open, too - no black boxes, minimal and optimal.
• In benchmarks, there's less than 10% speed difference with the top commercial ASN.1 compiler
• Legacy encodings? Compatibility with the past? Sure – ACN allows you to completely control the serialization format used in the binary streams
 ASN.1 in space - FPGAs (6/6)

• You work with FPGAs? We can help.

• Our code generators automatically map a system's ASN.1 grammar to ...
  
  − (a) VHDL and SystemC skeletons for your designs, with all the interface declarations ready-made for you (just fill in the body of the logic)
  
  − (b) device drivers that automatically interface with the FPGA, communicating with it at runtime. We prototyped this over a USB accessible Xilinx Spartan3, and then tried it on an FPGA accessible at runtime over Leon's PCI bus.
ASN.1 in space - FPGAs (6/6)

VHDL Skeleton

```vhdl
architecture archivhdl_aes of vhdl_aes is
begin
    process(clk_vhdl_aes,rst_vhdl_aes)
        variable run : std_logic;
        begin
            if rst_vhdl_aes='0' then -- Asynchronous reset
                finish_vhdl_aes <= '0';
                -- write your resets here
                run := '1';
            elsif clk_vhdl_aes'event and clk_vhdl_aes='1' then
                if start_vhdl_aes = '0' then
                    finish_vhdl_aes <= '0';
                    run := '1';
                elsif run = '1' then
                    -- write your logic to compute outputs from inputs here
                    -- and when your results are ready, set...
                    --
                    -- run := '0';
                    -- finish_vhdl_aes <= '1';
                end if;
            end if;
        end process;
end archivhdl_aes;
```
ASN.1 in space - FPGAs (6/6)

Automatically generated device driver:

```c
if (var_T_VHDL_Arg.kind == t_vhdl_aes_arg_set_key_PRESENT) {
  unsigned tmp = 1;
  ESAWriteRegister(BASE_ADDR + 0x4, tmp);
  {
    unsigned tmp = var_T_VHDL_Arg.u.t_vhdl_aes_arg_set_key.t_arg_key_length;
    ESAWriteRegister(BASE_ADDR + 0x8, tmp);
  }
  {
    unsigned tmp = 0;
    tmp |= ((unsigned)var_T_VHDL_Arg.u.t_vhdl_aes_arg_set_key.t_arg_key_content.arr[0]) << 0;
    tmp |= ((unsigned)var_T_VHDL_Arg.u.t_vhdl_aes_arg_set_key.t_arg_key_content.arr[1]) << 8;
    tmp |= ((unsigned)var_T_VHDL_Arg.u.t_vhdl_aes_arg_set_key.t_arg_key_content.arr[2]) << 16;
    tmp |= ((unsigned)var_T_VHDL_Arg.u.t_vhdl_aes_arg_set_key.t_arg_key_content.arr[3]) << 24;
    ESAWriteRegister(BASE_ADDR + 0xc + 0, tmp);
  }
```
Summary: a single data definition to ensure consistency everywhere
Meet TASTE (taste.tuxfamily.org)

• There's much more. These were just the highlights!

• You are cordially invited to download the TASTE VM, a Virtual Machine that contains all the tools you saw - built over the last 6 years. Use this VM with the free VMWARE Player (Windows, Linux) or VirtualBox (Linux, OS/X)

• The VM auto-updates itself, once you boot it, via a simple “UpdateTASTE.sh” script – or a simple double-click on a desktop icon :-) You are always up to date in terms of our tools.

• The technology is quite mature – join us! We can build a mission together, even up to a complete satellite.
http://taste.tuxfamily.org

Questions?
http://taste.tuxfamily.org

Questions?