DATA MODELLING IN PROBA3 ASPIICS PAYLOAD

Lessons learned and way forward

Konrad Grochowski

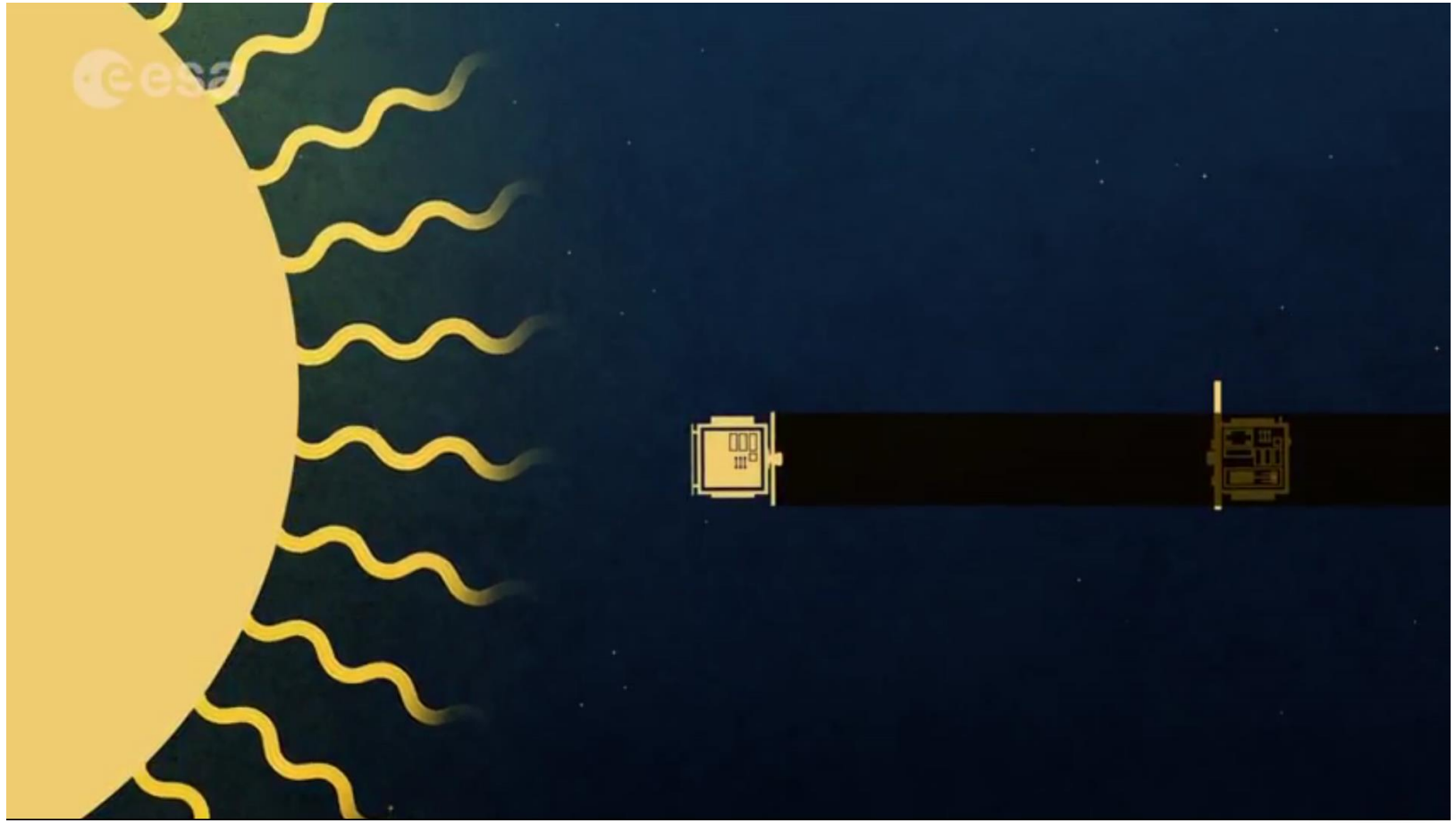
kgrochowski@n7mobile.com

2016

Agenda

- Short introduction
- PROBA-3 Mission
- CCB Software
- Data Modelling for CCB
- Small ASN.1/ACN example
- Lessons learned
- Future

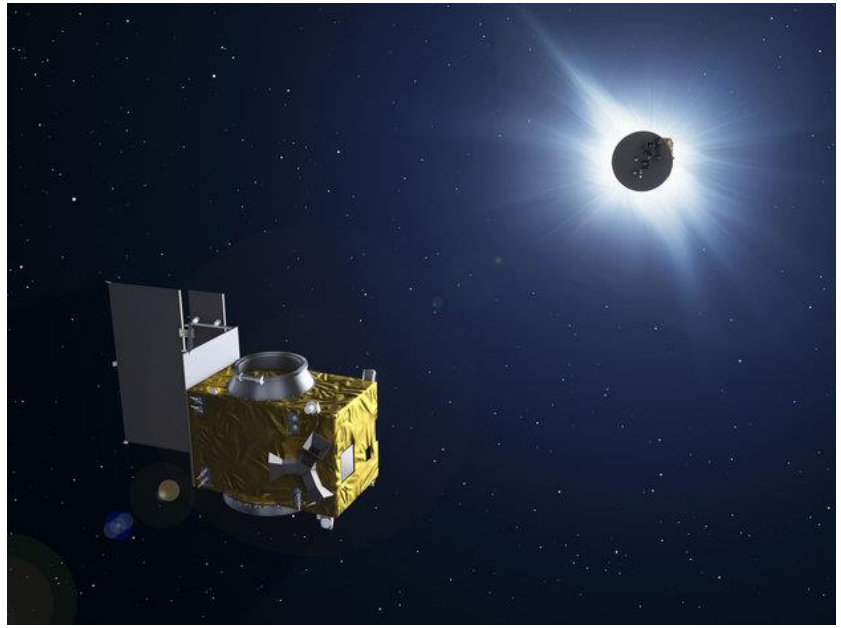
PROBA-3 – Formation Flying & ASPIICS



©ESA–Proba-3: Dancing with the stars

CCB Software - responsibilities

- BSW
  - Low-level initialization (Leon3)
  - Booting Application SW
  - Updating Application SW
  - TC/TM interface
- ASW
  - Shadow Position Sensor
  - Image acquisition
  - Mechanical devices control
  - Image compression and upload to platform
  - TC/TM interface

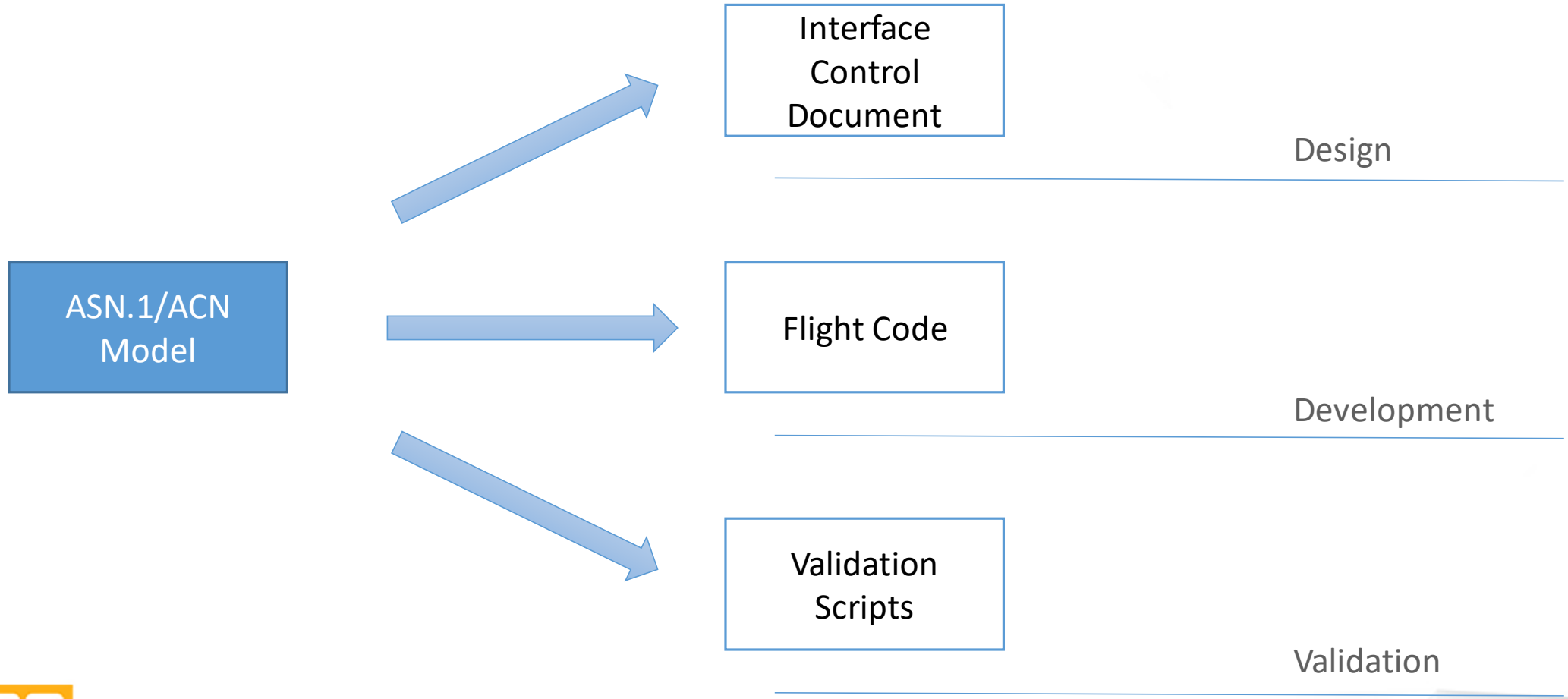


Data Modeling for CCB – platform exchange

- PUS over CCSDS
- ASN.1 data model
- ACN encoding model
- asn1scc compiler from TASTE
  - Generating flight code
  - Generating ICD
- DMT tools from TASTE
  - Generating verification code

taste
The Assert Set of Tools for Engineering

Single Model – Multiple uses



ASN.1 model

```
-- List representing all possible data structures contained by TM.
-- Only single item from this list can be present in TM at once.
-- Present item is determined by reporting service type and sub-type.
BSW-TM-SourceData ::= CHOICE
{
  -- Data in PUS(1,1) report - Telecommand Acceptance Report - Success.
  ackSuccess                    TM-PUS-1-1-AckSuccess,
  -- Data in PUS(1,2) report - Telecommand Acceptance Report - Failure.
  ackFailure                    TM-PUS-1-2-AckFailure,
  -- ...
  -- Data in PUS(17,2) report - Link Connection Report.
  connectionReport              TM-PUS-17-2-LinkConnectionReport
}
```

ACN Encoding

```
BSW-TM-SourceData<Base.UInt8:type, Base.UInt8:subType> []
{
  ackSuccess                  [present-when   type == 1    subType == 1],
  ackFailure                  [present-when   type == 1    subType == 2],
  -- ...
  connectionReport            [present-when   type == 17   subType == 2]
}
```

Generated part of ICD

BSW-TM-SourceData (CHOICE)	Min: 0 bytes	Max: 1030 bytes

List representing all possible data structures contained by TM.
Only single item from this list can be present in TM at once.
Present item is determined by reporting service type and sub-type.

No	ACN Parameters	Type
1	type	UInt8
2	subType	UInt8

No	Field	Comment	Present	Type	Min Bits	Max Bits
1	ackSuccess	Data in PUS(1,1) report - Telecommand Acceptance Report - Success.	type=1 AND subType=1	TM-PUS-1-1-AckSuccess	32	32
2	ackFailure	Data in PUS(1,2) report - Telecommand Acceptance Report - Failure.	type=1 AND subType=2	TM-PUS-1-2-AckFailure	40	40
…	…	…	…	…	…	…
11	connectionReport	Data in PUS(17,2) report - Link Connection Report.	type=17 AND subType=2	TM-PUS-17-2-LinkConnectionReport	0	0

Generated code usage

```
void TmBuilder_Pus17_build(PusCoder_Telemetry* const tm)
{
  PUSCODER_TM_INIT(tm);
  tm.packetDataField.sourceData.kind = connectionReport_PRESENT;
}
```

Python bindings usage in SVF

```
import aut


class Pus17Test(aut.TestSuite):

    ## Sends PUS(17,1) request and expects application to respond with PUS(17,2) TM.
    #  @SRS SRS-BSW-FUN-PUS-17-10
    #  @SRS SRS-BSW-FUN-PUS-17-15
    def test_applicationRespondsToPus17Request(self):
        self.pusInterface.sendTelecommand(aut.asn.TC_PUS_17_1_PerformConnectionTest())

        msg = self.pusInterface.expectTelemetry(
            expectedTmTypes=[aut.asn.TM_PUS_17_2_LinkConnectionReport],
            timeout=5)

        self.assertTmDataEqual(aut.asn.TM_PUS_17_2_LinkConnectionReport(),
                               msg)
```

Lessons learned

- Some inconvenience when using generated C code
- Indirect access to ‘hidden’/implicit layers

- Huge costs reduction
- Developers confidence improved (consistency!)
- Models become source code

Generated C code inconvenience

```
tm.packetDataField.sourceData.kind = ackFailure_PRESENT;
tm.packetDataField.sourceData.u.ackFailure.errorParam.kind = pus1_AckErrorCode_AppError_PRESENT;
tm.packetDataField.sourceData.u.ackFailure.errorParam.u.pus1_AckErrorCode_AppError = errCode;
```

- Ada?
- C++?
- Some helper functions/macros?

Workload reduction due to code generation

	BSW	ASW
PUS Services	1, 5, 6, 8, 17	1, 3, 5, 6, 8, 9, 17
ASN.1	691	3332
ACN	316	946
C Source	18108	83132
C Headers	2259	8052

Includes unit tests with 100% coverage!

Future

- asn1scc is being constantly enhanced
- N7 Mobile started development of dedicated IDE for ASN.1/ACN
- PUS-C – ASN.1 + ORM

Wish-list

- Dedicated Python generator
- Enhancements in developer-experience

Conclusions

- Models reduce costs and increases dependability
- Single model ensures consistency from design, documentation, implementation to validation
- New PUS-C standard can make ASN.1/ACN models popular

Thank you for your attention

Konrad Grochowski
kgrochowski@n7mobile.com
+48 693050711
www.n7mobile.pl/Space

N7 Mobile Sp. z o.o.
Łowicka 19/10
02-574 Warszawa
Polska