



A Data-Driven Command and Telemetry System

2016 Workshop on Spacecraft Flight Software

Richard D. Hunt
Sandia National Laboratories
P.O. Box 5800 M/S 0513
Albuquerque, NM 87185-0513
(505) 844-3193
rdhunt@sandia.gov

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

A decorative graphic in the top-left corner consisting of a large grey star and several overlapping red and white stripes.

Purpose of this Research

- ◆ Create a system for managing command and telemetry parameters in an Electronic Interface Control Document (eICD) to ensure better coherence between flight and ground systems
- ◆ Develop software that uses the eICD in a data-driven fashion to automate software generation and operation within flight and ground systems
- ◆ Show how COTS based tools can be leveraged by flight system, ground system, and analysis teams to process the eICD and improve data sharing
- ◆ Demonstrate representative flight and ground system software using an eICD and associated tools

Key Terms

- ◆ Payload Parameters – Artifacts that define command and telemetry values within payload hardware interfaces and applications
- ◆ XML – Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable
- ◆ XML Schema – A language for expressing constraints about XML documents
- ◆ Binary XML – A compact representation of ASCII XML data in a binary form
- ◆ XTCE – XML Telemetric and Command Exchange (XTCE) is an XML schema standard for space systems that is managed by OMG and CCSDS
- ◆ XML Data Binding – The process of creating a software data objects from an XML document based on a specific XML schema
- ◆ Data Model – The software object model created as a result of the XML data binding process
- ◆ Data Objects – Data model objects accessible to software applications in a specific programming language
- ◆ Binary Serialization – The process of creating a serialized encoded data stream from a data model
- ◆ Encoding Format – Electronic format used in binary serialization process
- ◆ Decoding Format – Electronic format used in the binary deserialization process
- ◆ Metadata – Additional data provided with data values that describes it (size, type, ...)
- ◆ Electronic ICD – An Electronic Interface Control Document (eICD) is an electronic file that defines payload parameters for use by software applications

What is an Electronic ICD?

Command and Telemetry parameters were defined in an ASCII file called a "Config File"

PE
SDP 3
0x48
0x1234
0x4321
0xDEAD
0xBEEF

TSE
FIB
0x51
0x1111
0x2222
0x3333
0xAA55

Subsystem **PE**
 Table **123** for **SDP 3**
 Field Frame Rate
 Field Enable Output
 Field Enable Framing
 .
 .

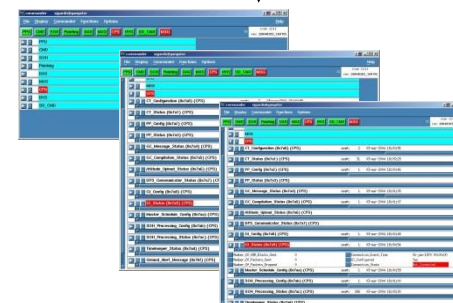
Subsystem **SE**
 Table **456** for **SVO 14**
 Azimuth 256 Temp
 ARS 0 Timeout
 .
 .
 .

A compiler used the Config File to generate data structures for storing parameters within the flight code

CVPP
Compiler



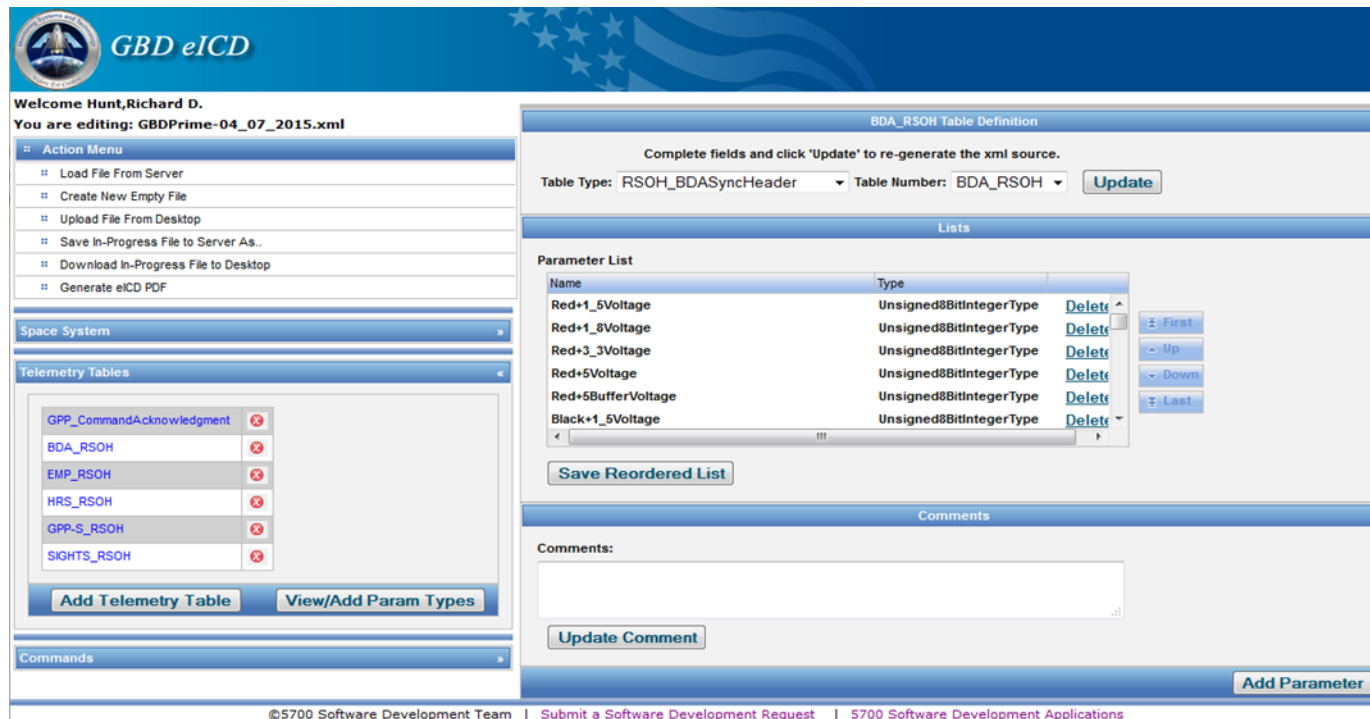
Commands and Status



Flight and Ground software used the eICD directly and became "tightly coupled"

The ground system parsed the Config File upon startup and dynamically generated command and telemetry displays

The New Electronic ICD



The screenshot shows the GBD eICD web interface. At the top, it says "Welcome Hunt, Richard D. You are editing: GBDPrime-04_07_2015.xml". The interface is divided into several sections:

- Action Menu:** Includes options like "Load File From Server", "Create New Empty File", "Upload File From Desktop", "Save In-Progress File to Server As...", "Download In-Progress File to Desktop", and "Generate eICD PDF".
- Space System:** A dropdown menu.
- Telemetry Tables:** A list of tables including "GPP_CommandAcknowledgment", "BDA_RSOH", "EMP_RSOH", "HRS_RSOH", "GPP-S_RSOH", and "SIGHTS_RSOH". Each table has a red 'X' icon next to it. Below the list are buttons for "Add Telemetry Table" and "View/Add Param Types".
- Commands:** A dropdown menu.
- BDA_RSOH Table Definition:** A section for defining the BDA_RSOH table. It includes a "Complete fields and click 'Update' to re-generate the xml source." instruction. Below this are dropdowns for "Table Type: RSOH_BDASyncHeader" and "Table Number: BDA_RSOH", followed by an "Update" button.
- Lists:** A section containing a "Parameter List" table. The table has columns for "Name", "Type", and "Delete". The parameters listed are:

Name	Type	Delete
Red+1_5Voltage	Unsigned8BitIntegerType	Delete
Red+1_8Voltage	Unsigned8BitIntegerType	Delete
Red+3_3Voltage	Unsigned8BitIntegerType	Delete
Red+5Voltage	Unsigned8BitIntegerType	Delete
Red+5BufferVoltage	Unsigned8BitIntegerType	Delete
Black+1_5Voltage	Unsigned8BitIntegerType	Delete

 To the right of the table are navigation buttons: "First", "Up", "Down", and "Last". Below the table is a "Save Reordered List" button.
- Comments:** A section for adding comments. It includes a "Comments:" label, a text input field, and an "Update Comment" button. At the bottom right of this section is an "Add Parameter" button.

At the bottom of the interface, there is a footer: "©5700 Software Development Team | Submit a Software Development Request | 5700 Software Development Applications".

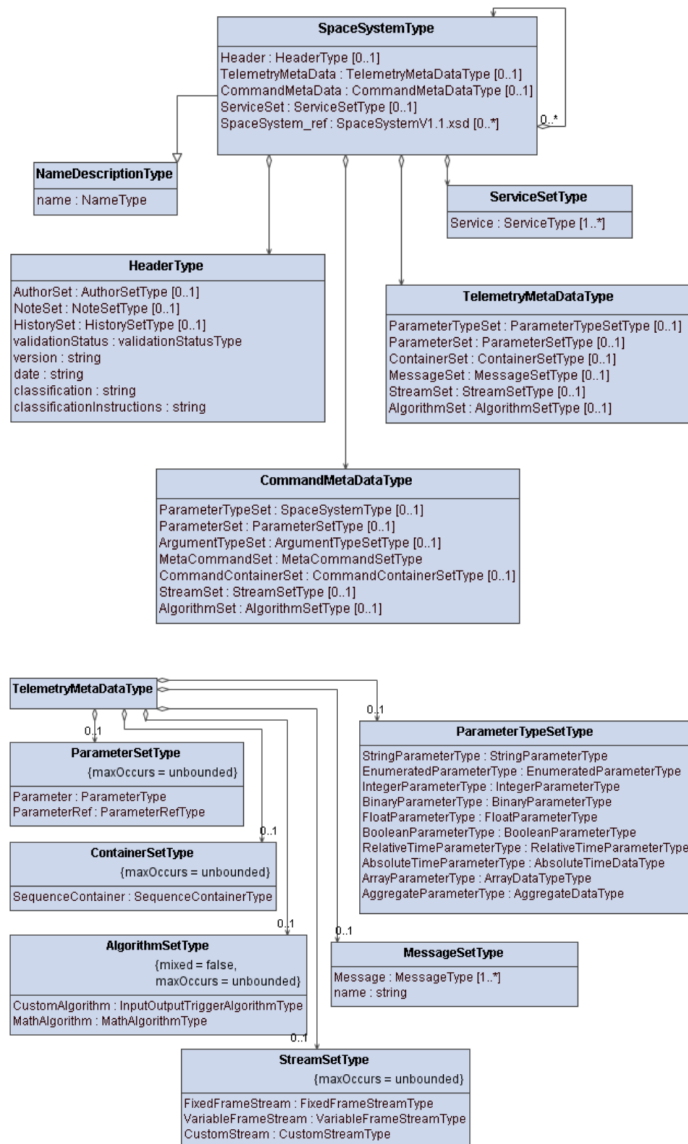
- ◆ Uses **XML** with the **XTCE Schema** to define command and telemetry parameters
- ◆ Uses **XML Data Binding** tools to allow software to use the parameters in a **Data-Driven** fashion
- ◆ Operates through a web based interface to provide convenient management of content

XML and XTCE

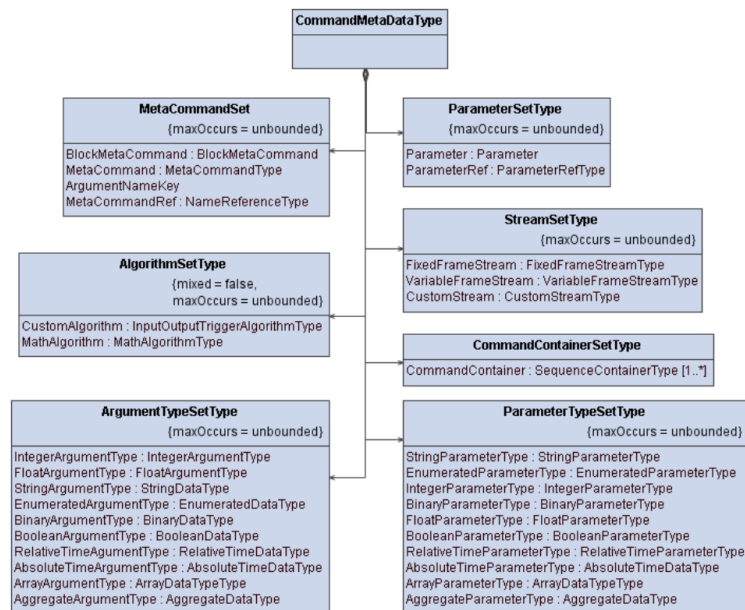
- ◆ Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable
 - XML 1.0 was first published in 1998
 - A textual data format that was designed for simplicity, generality and usability across the internet
 - Open standard for defining electronic documents (e.g. MS Office)
 - Plethora of COTS tools for processing XML data

- ◆ XML Telemetric and Command Exchange (XTCE) is both a CCSDS and OMG standard schema
 - XTCE 1.0 was first published in 2005; XTCE 1.2 is coming soon
 - Dictionary exchange standard oriented towards space mission operations which describes properties including:
 - Commands, arguments, and other aspects of commanding
 - Telemetry, mnemonics, limits and calibrators
 - Packaging: such as packets or major/minor frames
 - And so forth

XTCE Hierarchy Example



- ◆ Provides containers for organizing parameters into packets or frames
- ◆ Provides metadata for defining command and telemetry parameters



XML Data Binding

```
<object>
  <name>Red Hill</name>
  <position>
    <lat>-33.69</lat>
    <lon>18.83</lon>
  </position>
</object>
```

```
object* obj = ...; // Parse XML.
const char* name = obj->name ();
position& pos = obj->position ();
float lat = pos.lat ();
float lon = pos.lon ();

delete obj;
```

- ◆ Process of generating software data objects in computer memory from XML data
- ◆ XML data binding tools (<http://xmldatabinding.org/>)
 - Provide parsing, serialization/deserialization, and validation functions
 - Support many common programming languages
 - C++ – Code Synthesis XSD
 - Java – Java Architecture for XML Binding (JAXB)
 - Support a variety of workstation and embedded operating systems
 - Unix, Windows, VxWorks, ...



Data-Driven Software

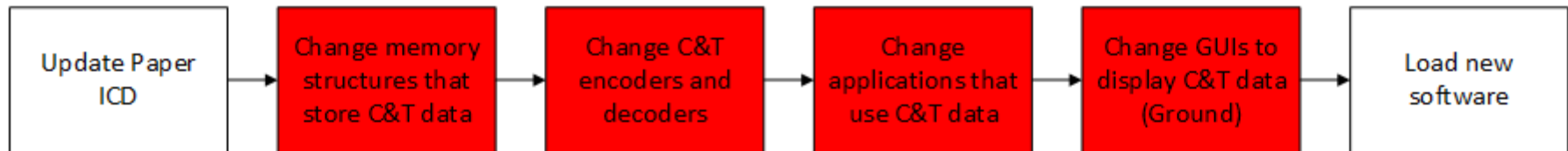
- ◆ Software that is automatically generated or operates on the metadata defined within the eICD
 - An example would be to allocate memory data structures based on a command or telemetry parameter name, size, and type

- ◆ Software that can be auto-generated from metadata
 - Memory structures and interface functions
 - Database structures and interface functions
 - GUI Displays and interface functions

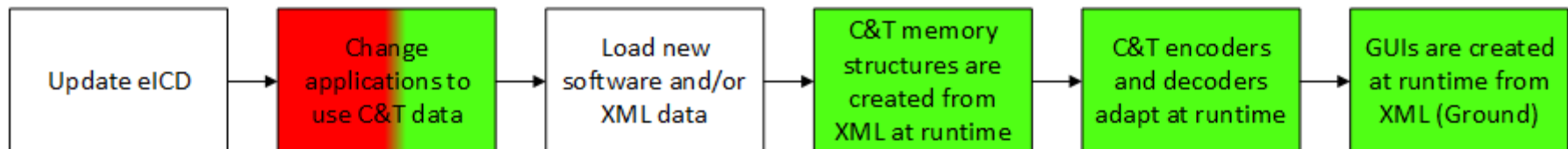
- ◆ Software functions that operate on metadata
 - Telemetry collection
 - Data validation
 - Database population
 - GUI population
 - Communication link encoding/decoding

Developer Driven vs. Data Driven

Developer Driven – Developers must update software to reflect changes in ICD definition



Data Driven – Software automatically adapts to changes in ICD definition



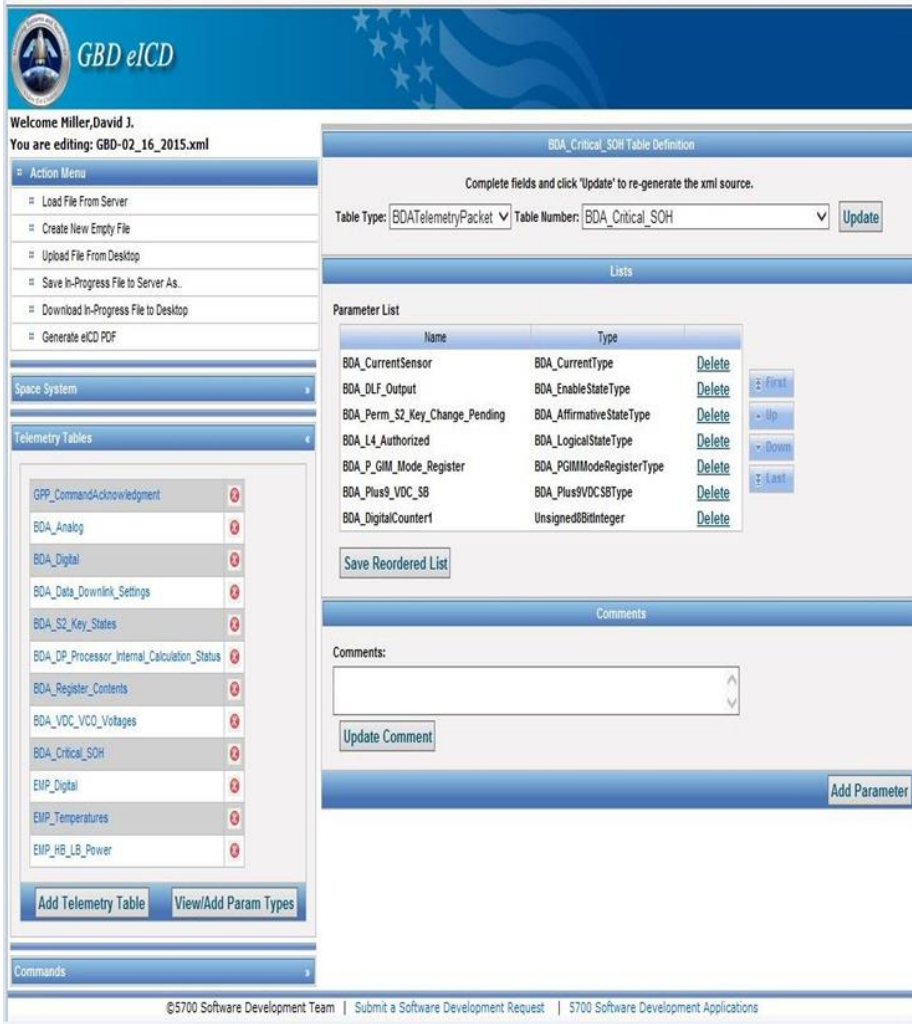
Manual

Requires developers to change software, test and verify

Automatic

Software is auto-generated or adapts at runtime; no development and minimal testing and verification required

eICD → XML



GBD eICD

Welcome Miller, David J.
You are editing: GBD-02_16_2015.xml

Action Menu

- Load File From Server
- Create New Empty File
- Upload File From Desktop
- Save In-Progress File to Server As...
- Download In-Progress File to Desktop
- Generate eICD PDF

Space System

Telemetry Tables

- GPP_CommandAcknowledgment
- BDA_Analog
- BDA_Digital
- BDA_Data_Download_Settings
- BDA_S2_Key_States
- BDA_DP_Processor_Internal_Calculation_Status
- BDA_Register_Contents
- BDA_VDC_VCO_Voltages
- BDA_Critical_SOH
- EMP_Digital
- EMP_Temperatures
- EMP_HB_LB_Power

BDA_Critical_SOH Table Definition

Complete fields and click 'Update' to re-generate the xml source.

Table Type: BDAtelemetryPacket Table Number: BDA_Critical_SOH

Lists

Parameter List

Name	Type	
BDA_CurrentSensor	BDA_CurrentType	Delete
BDA_DLF_Output	BDA_EnableStateType	Delete
BDA_Perm_S2_Key_Change_Pending	BDA_AffirmativeStateType	Delete
BDA_L4_Authorized	BDA_LogicalStateType	Delete
BDA_P_GIM_Mode_Register	BDA_PGIMModeRegisterType	Delete
BDA_Plus9_VDC_SB	BDA_Plus9VDCSBType	Delete
BDA_DigitalCounter1	Unsigned8BitInteger	Delete

[Save Reordered List](#)

Comments

Comments:

©5700 Software Development Team | Submit a Software Development Request | 5700 Software Development Applications

```

<xtce:SequenceContainer name="BDA_Critical_SOH">
  <xtce:EntryList>
    <xtce:ParameterRefEntry parameterRef="BDA_CurrentSensor" />
    <xtce:ParameterRefEntry parameterRef="BDA_DLF_Output" />
    <xtce:ParameterRefEntry parameterRef="BDA_Perm_S2_Key_Change_Pending" />
    <xtce:ParameterRefEntry parameterRef="BDA_L4_Authorized" />
    <xtce:ParameterRefEntry parameterRef="BDA_P_GIM_Mode_Register" />
    <xtce:ParameterRefEntry parameterRef="BDA_Plus9_VDC_SB" />
    <xtce:ParameterRefEntry parameterRef="BDA_DigitalCounter1" />
  </xtce:EntryList>
  <xtce:BaseContainer containerRef="BDAtelemetryPacket">
    <xtce:RestrictionCriteria>
      <xtce:ComparisonList>
        <xtce:Comparison parameterRef="BDATableNumber" value="BDA_Critical_SOH" />
      </xtce:ComparisonList>
    </xtce:RestrictionCriteria>
  </xtce:BaseContainer>
</xtce:SequenceContainer>

```

Example of a telemetry table defined in the eICD and the XML file that was generated

eICD → PDF

Edit Integer Parameter

Edit parameter and click 'Save' to write to the xml source.

*Parameter Type Name: BDA_CurrentType Base Type:

Parameter Size In Bits: 32 Initial Value:

Signed: ☐ True ☒ False

Encoding Type: Integer Data Encoding

*Encoding Size In Bits: 8

Encoding: UNSIGNED

Unit Set

Unit: Unit Abbv: Add Unit

Unit	Unit Abbv	Delete
Ampere	A	Delete

Polynomial Calibrator

Coefficient: Exponent: Add Term

Coefficient	Exponent	Delete
0.0	0	Delete
1.0	1	Delete

Manage Alarms:

Min Violations: 1

View/Edit Static Alarms

View/Edit Alarm Conditions

Comments:

The next three types define examples of BDA subsystem parameter types that illustrate the

Save Close Window

Name	Base Type	Size	Initial Value	Signed	Data Encoding (Size)	Calibrator List	Static Alarm Ranges	Alarm Conditions	Comments
Unsigned1BitInteger		1	0	false					These define an example set of generic parameter types that can be used for both telemetry and commands.
Unsigned2BitInteger		2	0	false					
Unsigned3BitInteger		3	0	false					
Unsigned4BitInteger		4	0	false					
Unsigned5BitInteger		5	0	false					
Unsigned6BitInteger		6	0	false					
Unsigned7BitInteger		7	0	false					
Unsigned8BitInteger		8	0	false					
Unsigned11BitInteger		11	0	false					
Unsigned12BitInteger		12	0	false					
Unsigned13BitInteger		13	0	false					
Unsigned14BitInteger		14	0	false					
Unsigned16BitInteger		16	0	false					
Unsigned24BitInteger		24	0	false					
Unsigned32BitInteger		32	0	false					
Unsigned56BitInteger		56	0	false					
Unsigned64BitInteger		64	0	false					
Signed8BitInteger		8	0	true					
Signed16BitInteger		16	0	true					
Signed32BitInteger		32	0	true					
Signed64BitInteger		64	0	true					
BDA_CurrentType		32		false	Integer Data Encoding (8)	Coeff: 0.0 Exp: 0 Warn: Critical	6.4 3.2	16.4 19.6	The next three types define examples of BDA subsystem parameter types that illustrate the use of units, calibrators, and alarms.

- O U O -

Page 6 of 26

Telemetry parameter in the eICD and associated PDF file including conversion values and alert/alarm thresholds

A decorative graphic in the top-left corner of the slide, featuring a large, stylized star with a red and white striped pattern and a blue outline, set against a background of red and white stripes.

Summary of Benefits

◆ Benefits of an eICD

- Allows content to be conveniently managed through a web based interface
- Helps payload and ground system applications stay synchronized through a centralized management approach
- Enables sharing of data between different teams and organizations through standard electronic document formats
- Allows content can be automatically validated against a schema for correctness
- Data can be outputted into a variety of electronic formats (XML, PDF)

◆ Benefits of using a Data-Driven approach

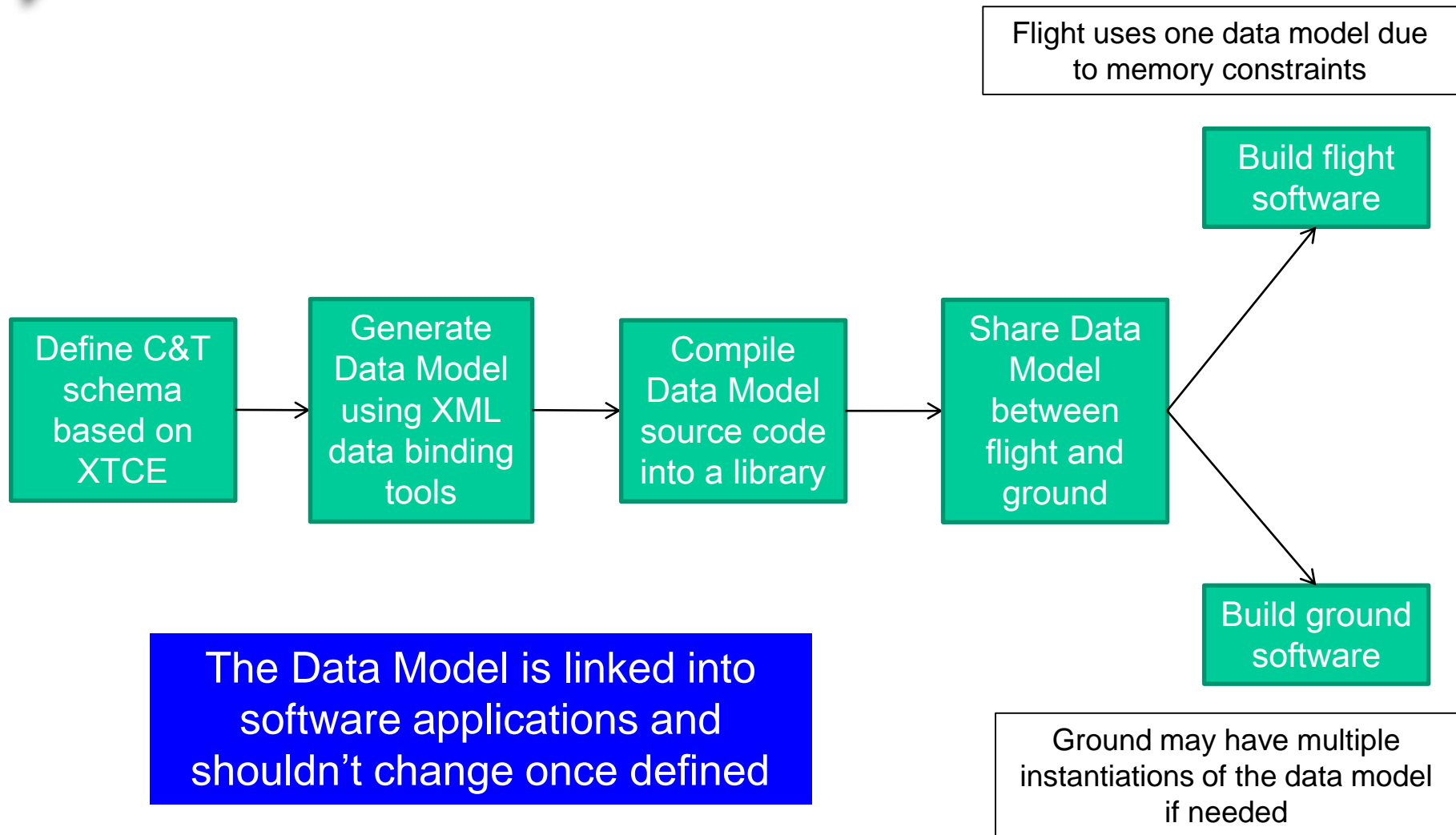
- Reduces time to develop, integrate, and test software due to auto-generation of software
- Reduces risk of programming errors from manual implementation
- Reduces the risk of large application uploads to the payload since content is managed externally from software

Process for Using an eICD

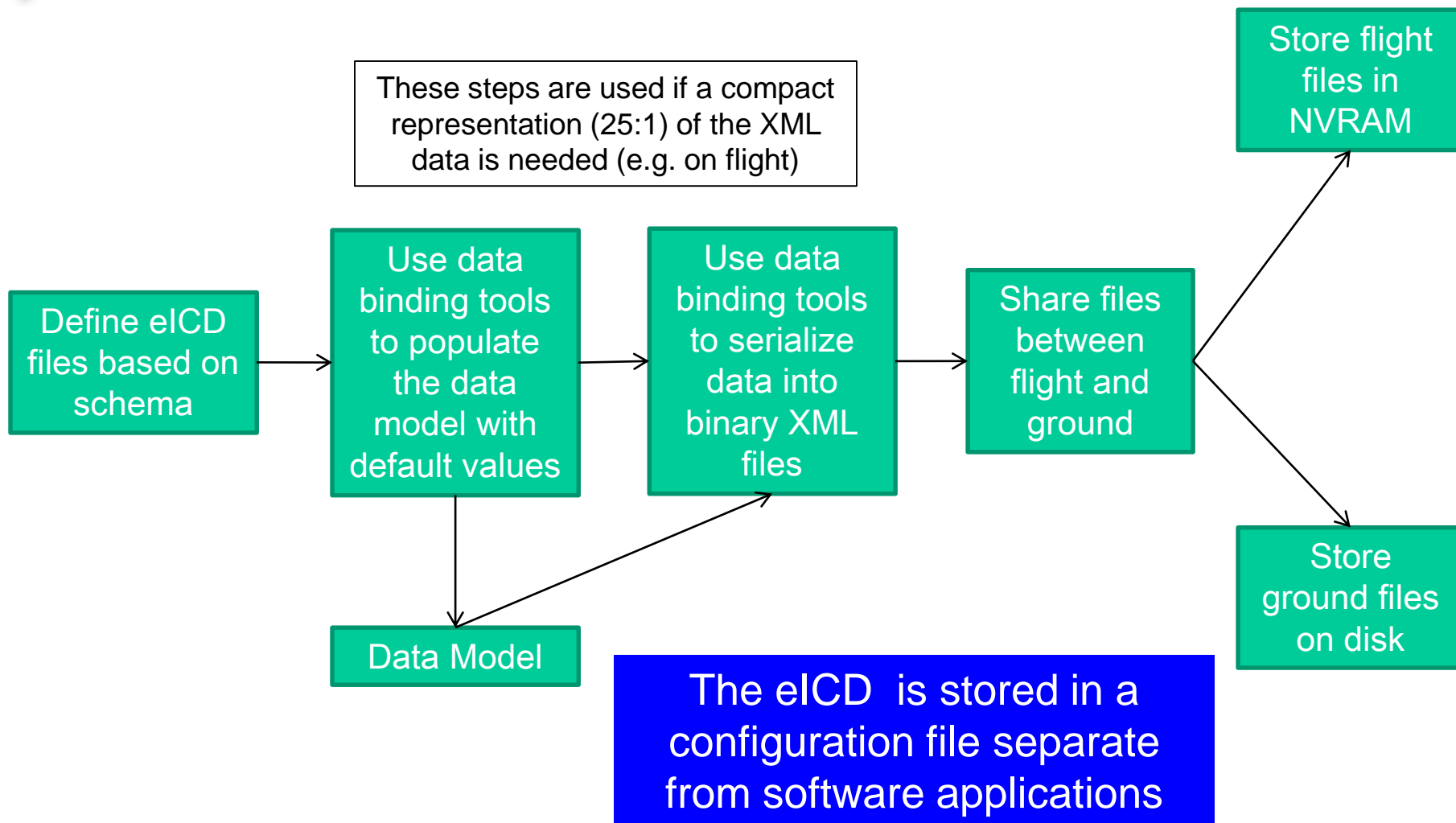
Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.



Step 1: Create and Share Data Model

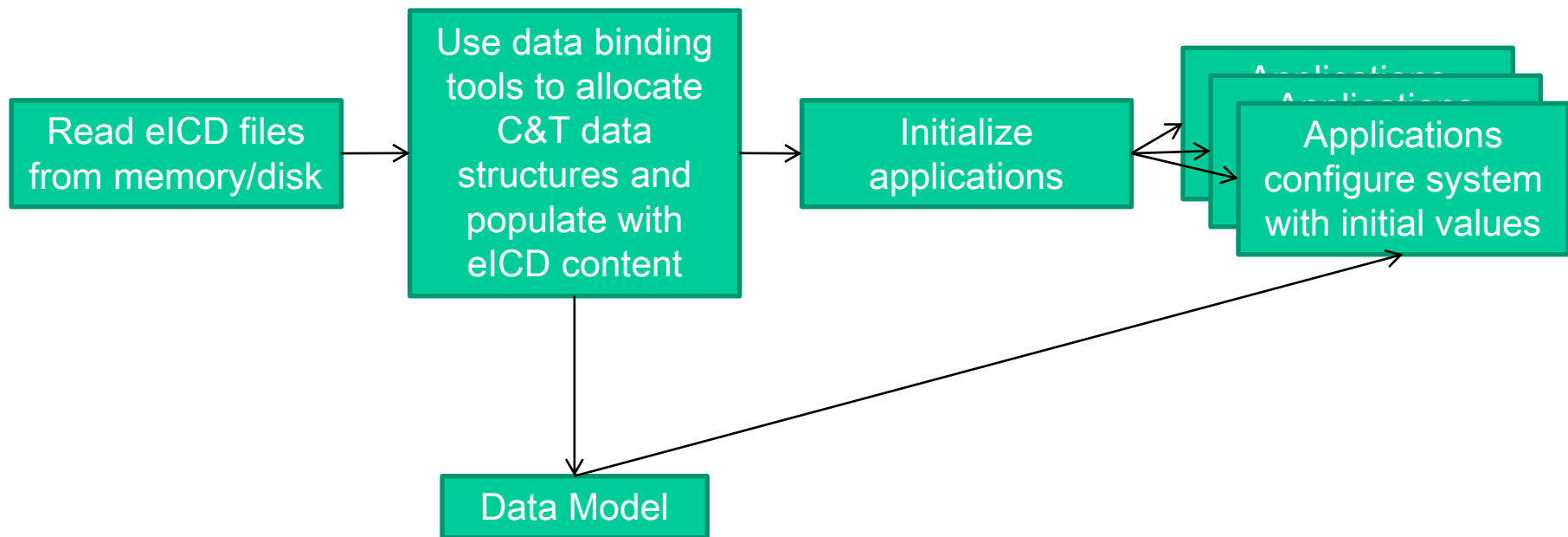


Step 2: Create and Share eICD

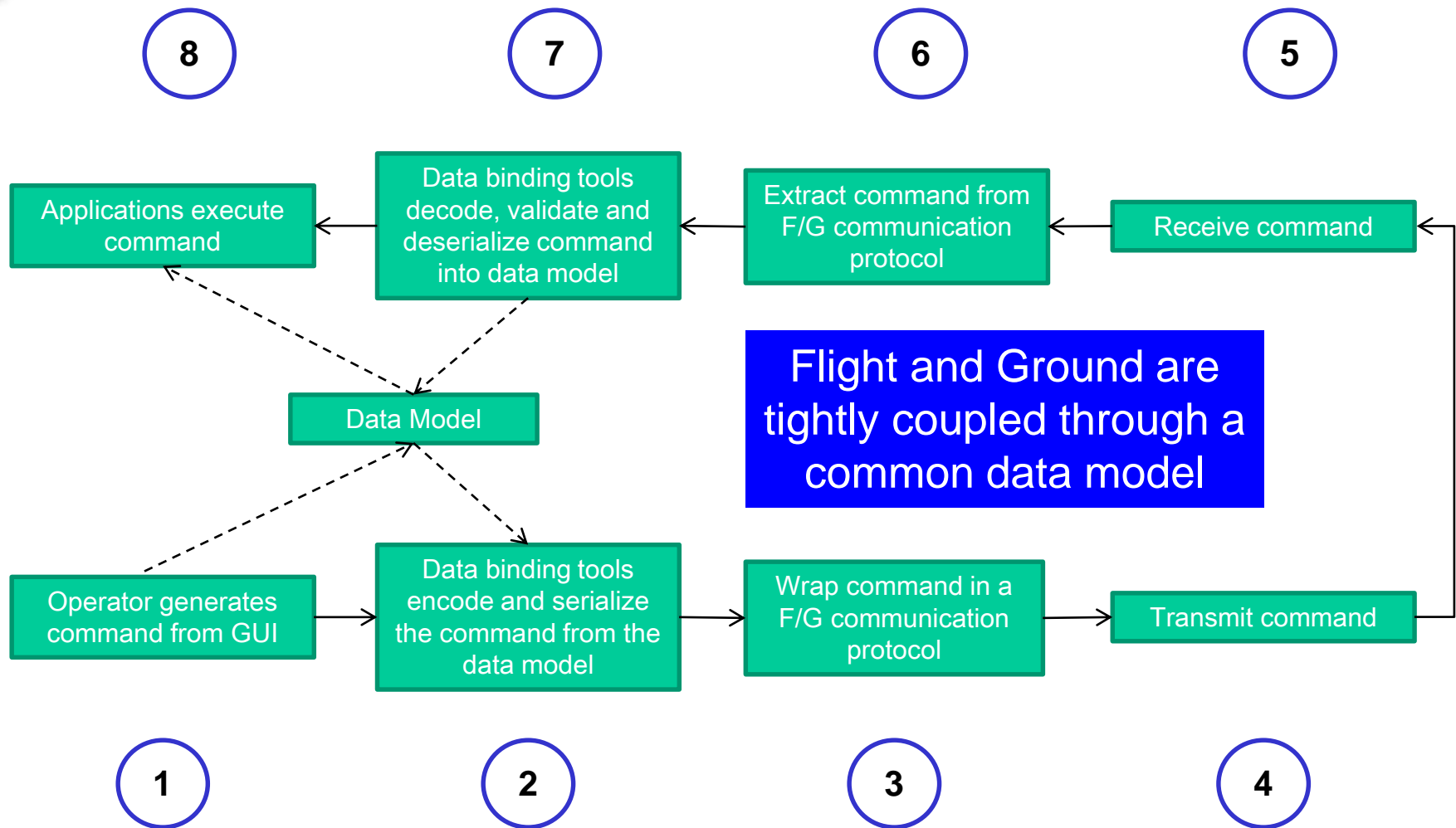


Step 3: Application Initialization

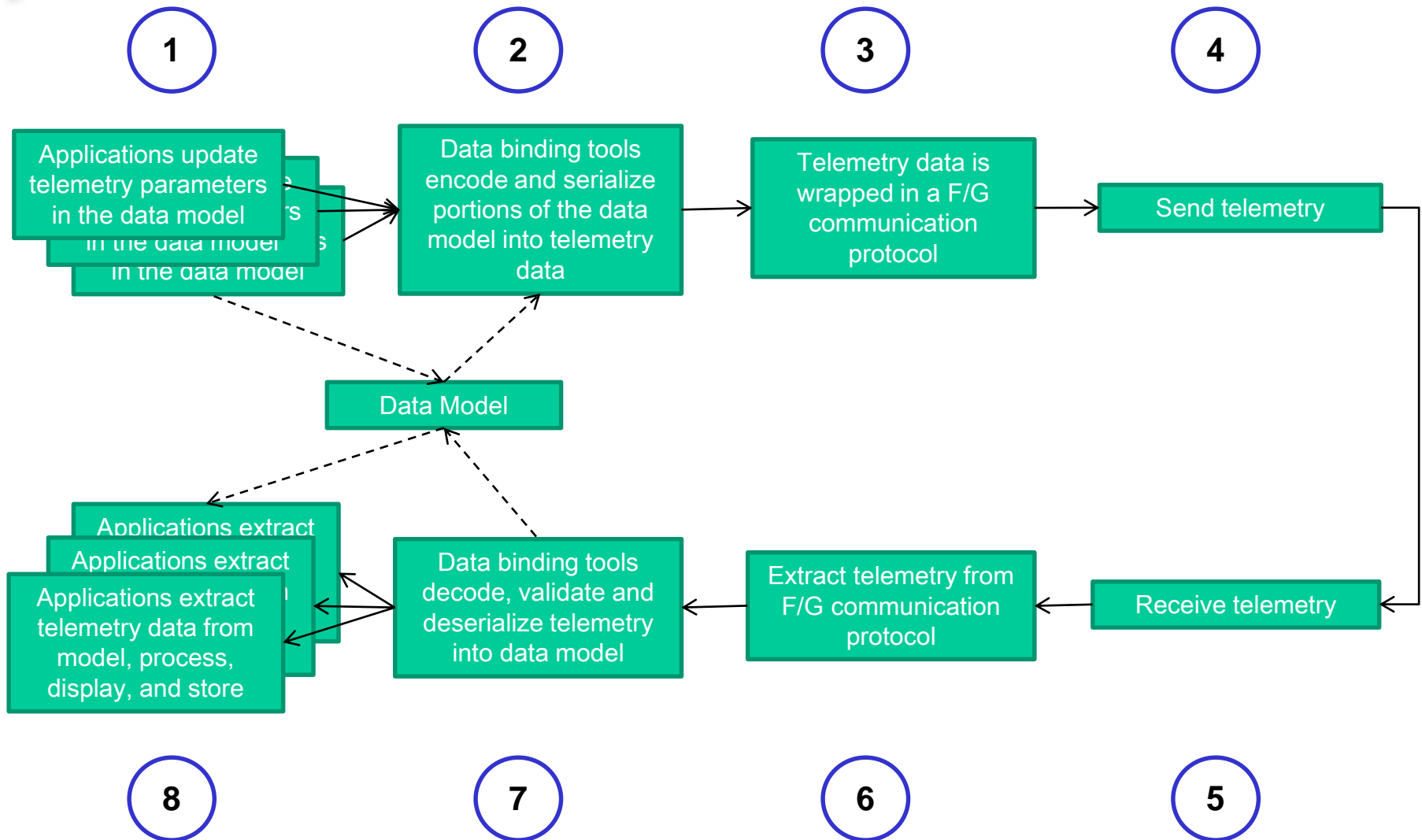
Flight and ground systems use the same initialization process; applications can be of any type: data processing, displays, drivers, ...



Step 4: Send Commands



Step 5: Send Telemetry

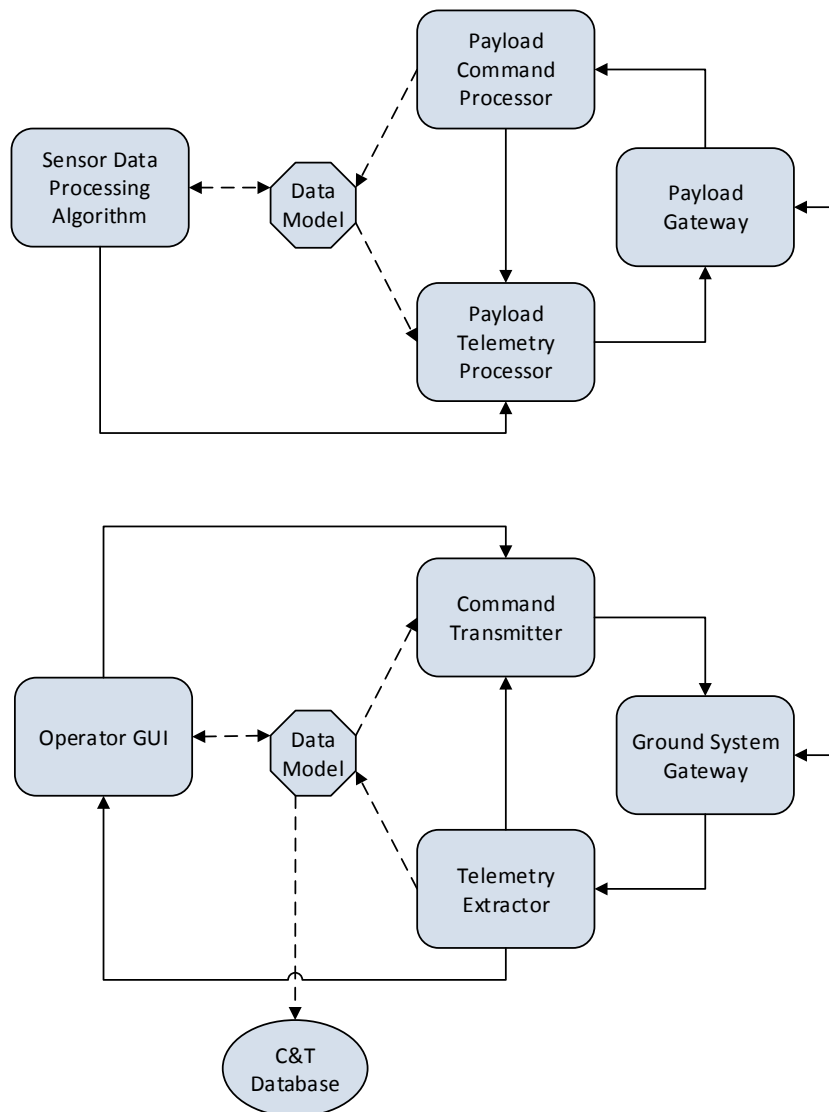


Demonstration

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.



Demonstration Software



Flight Processes

Sensor Data Processing Algorithm – Generates telemetry data, validates, stores in the data model, and signals telemetry processor.

Payload Command Processor – Receives commands from the ground, decodes, validates, stores in data model, and executes. Generates acknowledgements for sending to the ground

Payload Telemetry Processor – Reads data from the data model, encodes, serializes and sends to the ground.

Payload Gateway – Implements the communication protocol between flight and ground.

Ground Processes

Ground System Gateway – Implements the communication protocol between flight and ground.

Telemetry Extractor – Receives telemetry data, validates it, stores it in the data model, and signals GUI to display. Provides command acknowledgements to Command Transmitter.

Operator GUI – Generates commands, validates, stores in the data model and signals command transmitter. Reads telemetry data from the data model and displays.

Command Transmitter – Receives commands from the GUI, serializes and sends them to the payload. Processes acknowledgements from the payload.

C&T Database – Receives command and telemetry data from the data model and populates a data base.

Development Environment

- ◆ Flight software ran on an embedded processor
 - LEON 3-FT Processor @ 80MHz (SPARC V8 RISC)
 - 32 MB SRAM
 - 8MB NVRAM
 - Communication link was simulated over Ethernet interface to Linux Workstation
 - C++ application running on VxWorks 6.7 RTOS
 - Code Synthesis XSD/e for XML Binding

- ◆ Ground software ran on a Linux workstation
 - Java application using the JDK
 - Java Architecture for XML Binding (JAXB)
 - JavaFX & ControlsFX for GUIs
 - Remote Method Invocation (RMI)
 - Allows communication between JVMs

