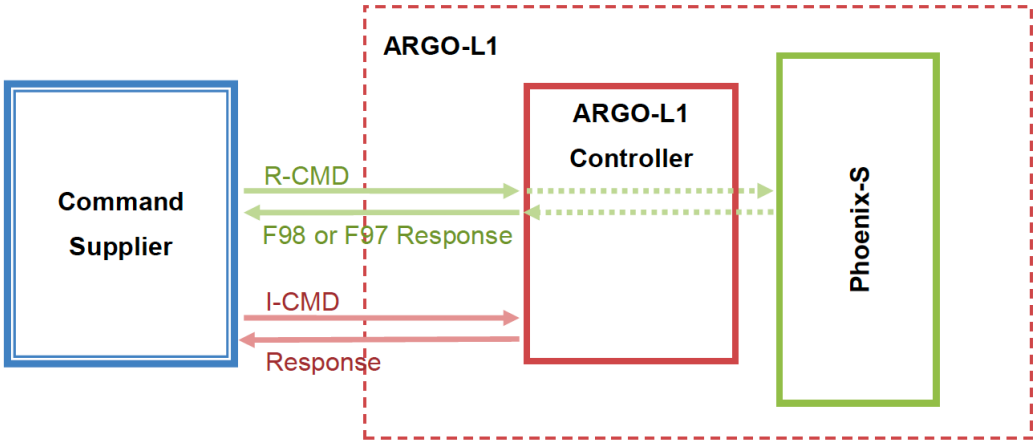
Appendix
Data-driven Mutation Testing:
LUXSPACE
ESAIL-GPS FAULT MODEL

This document describes the procedures to execute data-driven mutation testing on the communication between OBS and the ARGO-L1 GNSS Receiver system.
In Section 1 we provide a brief overview of the case study. Then, in Section 2, we describe the structure of the periodic output messages and command responses that will be object of the data-driven mutation procedure and the corresponding fault models.

1) OVERVIEW OF THE CASE STUDY

ARGO-L1 (where ARGO stands for Astro Redundant GNSS Orbital Determination) is a receiver for the Global Navigation Satellite System (GNSS).
F40 is the ID of a GPS receiver telemetry data (R-TLM) output message issued periodically from the ARGO-L1 GNSS Receiver.
The software of ARGO-L1 can be divided in two parts: the Phoenix-S GPS receiver's software and the interface circuit board software, which act as a data communication interface between the GPS receivers and the satellite communication bus.
It encodes GPS receiver commands (R-CMD) in CAN protocol Process Data Objects and decodes then in UART protocol to communicate with the Phoenix-S GPS receivers.
It also handles the decoding of GPS receiver telemetry data (R-TLM) and R-CMD responses (encoded in UART protocol) and its subsequent encoding in CAN protocol PDO.



The specific operational details of this component are described in its user manual, the ESAIL-AFW-MAN-PFO-6023 document.
In the following table the R-TLMs are listed. They can be divided in output periodic messages (R-CMD IDs F13-15, F40-44, F48, F52 and F62) and command responses (R-CMD IDs F97-98).

Msg. ID	Characters	Description
F13	120	Satellite almanac data
F14	304	Satellite ephemeris data
F15	144	Ionospheric/UTC model data
F40	104	Cartesian navigation data
F41	337	Pseudorange and range-rate (smoothed)
F42	505	Pseudorange, carrier phase, range rate (raw)
F43	253	Channel status
F44	71	Clock data
F48	48	Configuration and status parameters
F52	95	User Spacecraft Mean Elements
F62	601	Extended raw data message
F98	variable	Native receiver command responses
F97	9	Simulated receiver command response. It ensures a command response even in the case the when the particular command has no corresponding native response (i.e. F98).

R-TLMs are linked to specific GPS receiver commands (R-CMD) which are issued by the command supplier (in this case the satellite control software) to the ARGO L1 controller and in turn to the Phoenix-S GPS receiver.
According to LuxSpace, **the most interesting command to be considered for data mutation is F40.** We are interested in mutating the **response** generated by this command, as reported below.

2) R-TLM STRUCTURE AND FAULT MODEL

The receiver telemetry (R-TLM) sentence format is shown in the following tables.
All bytes contained in the data field (X) are printable 7 bit characters (ASCII 32-127).

R-TLM	<STX>	'F'	n	n	x	x	x	…	x	x	H	H	<ETX>

<STX>	Protocol character – Start of Transmission (ASCII char. 0x02)
<ETX>	Protocol character – End of Transmission (ASCII char. 0x03)
'F'nn	Output message identifier (3x ASCII char.) were, 'F' is the fixed initial character and n a decimal digit (0,…,9)
CC	Command identifier (2x ASCII char.) were, C is an uppercase alphabetic character
HH	Sentence checksum (2x ASCII char.) to guard against transmission errors-hexadecimal (uppercase) representation of the exclusive-or of all data field characters and message/command identifier characters in the sentence, excluding <STX> and <ETX> characters
X	Data field (1 BYTE)

R-TLM F40-CARTESIAN NAVIGATION DATA

The F40 message provides the Cartesian state vector (position and velocity) of the host vehicle in the Earth-fixed WGS84 system.

Byte	Span	Format	Type	Description	Fault Class
0	1	x	HEX	<STX>	IV(V!= 0x02)
[1…3]	3	xxx		Message Id (=F40)	
[4…7]	4	xxxx		GPS week	
[8…19]	12	xxxxxx.xxxxx	GG	GPS seconds of week [s] (of navigation solution)	HV(Value=100) SS(Delta=60) VAT(T=604800, D=1)
[20…21]	2	xx		GPS-UTC [s]	none
[22…33]	12	sxxxxxxxx.xx		x (WGS84) [m]	ASA(T=0, D=100000, V=2) SS(D=100000)
[34…46]	12	sxxxxxxxx.xx		y (WGS84) [m]	ASA(T=0, D=100000, V=2) SS(D=100000)
[47…58]	12	sxxxxxxxx.xx		z (WGS84) [m]	ASA(T=0, D=100000, V=2) SS(D=100000)
[59…70]	12	sxxxxx.xxxxx		vx (WGS84) [m/s]	ASA(T=0, D=1000, V=2) SS(D=10000)
[71…82]	12	sxxxxx.xxxxx		vy (WGS84) [m/s]	ASA(T=0, D=1000, V=2) SS(D=10000)

[83…94]	12	sxxxxx.xxxxx		vz (WGS84) [m/s]	ASA(T=0, D=1000, V=2) SS(D=10000)
95	1	x		Navigation status (0=no-Nav, 1=First Fix, 2=continuous 3D-Nav)	IV(V=3) simulating an illegal value IV(V=0) to transform 2s in 0s
[96…97]	2	xx		Number of tracked satellites	SS(Delta=-2)
[98…101]	4	xx.x		PDOP	SS(D=20) IV(V=1) IV(V=0)
[102…103]	2	xx		Checksum	BF?
104	1	x	HEX	<ETX>	IV(V!= 0x03)

NOTES:

- "." denotes the position of the decimal point,
- leading digits may be blank and,
- "s" indicates that the first non-blank character contains the sign of the respective quantity, either "+" or "–".

EXAMPLE:

```
<STX>F401139180006.0000013 -1070289.69 -1135280.39 +6636840.38
-3777.36103 -6389.28606 -1696.253242 9 2.172<ETX>
```

3) PROBES INSERTION

The mutation probes were manually inserted in the method *GpsReceiver::SendF40,* which collects the GPS data and produce the package to send to the OBSW.

The method is defined in the file *GpsReceiver.cpp,* contained in the folder *Svf/Models/CAN/src/Pdhu/GPS*.

The probe insertion strategy was carried out as follows.

```
1.  void GpsReceiver::SendF40()
2.  {
3.
4.      ::Smp::UInt16 gpsWeek = 0;
5.      ::Smp::Float64 gpsSecondsOfWeek = 0.0;
6.
7.      Generic::Utils::GetGpsTime(getCurrentDateTime(), gpsWeek, gpsSecondsOfWeek);
8.
9.      //MANUALLY INSERTED PROBE
10.     std::vector<double> gps_buffer;
11.
12.     gps_buffer.push_back(gpsWeek);
13.     gps_buffer.push_back(gpsSecondsOfWeek);
14.     gps_buffer.push_back(cartNavData.x);
15.     gps_buffer.push_back(cartNavData.y);
16.     gps_buffer.push_back(cartNavData.z);
17.     gps_buffer.push_back(cartNavData.vx);
18.     gps_buffer.push_back(cartNavData.vy);
19.     gps_buffer.push_back(cartNavData.vz);
20.     gps_buffer.push_back(cartNavData.navigationStatus);
21.     gps_buffer.push_back(cartNavData.nbOfTrackSats);
22.     gps_buffer.push_back(cartNavData.pdop);
23.
24.     mutate_FM_F40( &gps_buffer, fm );
25.
26.     gpsWeek = gps_buffer[0];
27.     gpsSecondsOfWeek = gps_buffer[1];
28.     cartNavData.x = gps_buffer[2];
29.     cartNavData.y = gps_buffer[3];
30.     cartNavData.z = gps_buffer[4];
31.     cartNavData.vx = gps_buffer[5];
32.     cartNavData.vy = gps_buffer[6];
33.     cartNavData.vz = gps_buffer[7];
34.     cartNavData.navigationStatus = gps_buffer[8];
35.     cartNavData.nbOfTrackSats = gps_buffer[9];
36.     cartNavData.pdop = gps_buffer[10];
37.     //PROBE END
38.
39.     std::stringstream ss;
40.     ss << "F40" << std::setw(4) << gpsWeek << std::fixed << std::setw(12) << std::setp
    recision(5) << gpsSecondsOfWeek
41.        << std::setw(2) << (unsigned)cartNavData.gpsUtc << std::showpos << std::setw(1)
     << std::setprecision(2)
42.        << cartNavData.x << std::setw(12) << std::setprecision(2) << cartNavData.y << s
    td::setw(12)
43.        << std::setprecision(2) << cartNavData.z << std::setw(12) << std::setprecision(
    5) << cartNavData.vx
44.        << std::setw(12) << std::setprecision(5) << cartNavData.vy << std::setw(12) <<
    std::setprecision(5)
45.        << cartNavData.vz << std::noshowpos << std::setw(1) << (unsigned)cartNavData.na
    vigationStatus << std::setw(2)
46.        << (unsigned)cartNavData.nbOfTrackSats << std::setw(4) << std::setprecision(1)
    << cartNavData.pdop;
47.
48.     Trace(1, "F40: %s", ss.str());
49.
50.     std::string response = ss.str();
51.     ::std::vector<::Smp::UInt8> responseData(response.begin(), response.end());
52.     SendMessage(40, responseData);
```

```
53. }
```