SpaceOps-2025, ID # 1 (103)

Test the digital Space Systems by the tools engineering

Jamel Metmati[a]

[a] *Cyber Science Institut, ISAE-SUP AERO, France*, *Djamel.METMATI@ext.isae-supaero.fr*

Abstract

The Space as a service provides the capacity to deliver products and an access easier to the orbits. Moreover, the way to produce a satellite out of the box by the size, by the functions expected, the potential data from system design, the programming and the format data evolution. These external conditions provide a digital support to test the operation concepts including the operation security on-line and off-line through the operation engineering with more tools to ensure the same goals of safety, performance applicable for Space. In this process, some steps shall be included to improve the security of Space systems and subsystems during the design and the verification phase. This step should be an digital environment to test the robustness of the systems and subsystems by development environment. The tools engineering provide the capacities to test the digital Space systems by unique computer on board while considering the data expected for the other subsystem. It considers a information model applicable to the computer on board to test the functions and the data processing.

Keywords: system, test, cybersecurity, safety

Acronyms/Abbreviations : Assembly Integration Testing, Model Base Systemn Engineering, Application Programming Interface, Integration Verification Validation Qualification.

1. Introduction

The Space systems gathers all the componements on input and output able to form communication module. They find them on the ground segment for the operationnal center and on the Space segment on satellite on board. This manages the Space operations and others systems based on Space as services support the data dissemination for applications. It means the infrastructure are critical to maintain the satellites on orbit and to use use the data collected by the sensors in the spacecraft. All these requirements provide the capacity to a spacecraft to survive in Space or Deep Space in the harsh context in which the engineers must mix the safety and the security of the systems.

The Voyager I and Voyager II probe far away from never ever spacecraft in Space is still in live thanks to the robustness of its systems and the capacity to sprint code to manage the computer on board. To catch the purpose, the phase of design produce the requirements of the systems and the subsystems then the technologies are choosen. At last, all is tested on the ground and assembled by the AIT management to deliver a satellite. The process is industrialized for the major satellites. Nevertheless, the arrival of smallsat by private compagnies need support and environnement to ensure the safety and the security of the systems.

2. Material and methods

The context is to be able to apply safety and security concepts to the operation of on-board systems for Small satellites with the low means (figure 1). The simulated model is the approach choosen from the optimised MBSE [1] to test the system hardening requirements. The methods considers the features of the test through the capacity to loop the IVVQ phase over shorter times, the data generation for handling anomalies and incidents, the requirements management and compliance matrix management, the assistance in designing test scenarios. The design phases integrates the feasibility studies, the objectives, the needs, the constraints, the technical specifications for systems and subsystems, the management of internal and external interfaces, the performance, the safety, the reliability, the mission concepts and possible architectures, the associated risks and cost-based solutions, the mission concept selection.

The preliminary design review considers the checking requirements, the design evaluation, the risk analysis, the performance validation. The detailed design takes the design validation, the further analysis, the full documentation. Then, the robustness means the safety to keep the system in orbit, the security for anomaly and incident management and the survival of the system plus the subsystem. The On Board Computer [2] simulates safe operation of on-board processor to support development phases. The focus on the content are the technical documents, the analysis report, the development plan, the risk management strategy, the continuous simulation analysis report without waiting for the preliminary design review. The figure 1 describes the function of the SmallSat [3] and the blocks that could be taken account in the simulated model. Each one should be simulated depending on the purpose of the tests from the functional architecture of SmallSat on the figure 1 and the hardware components on the figure 2.

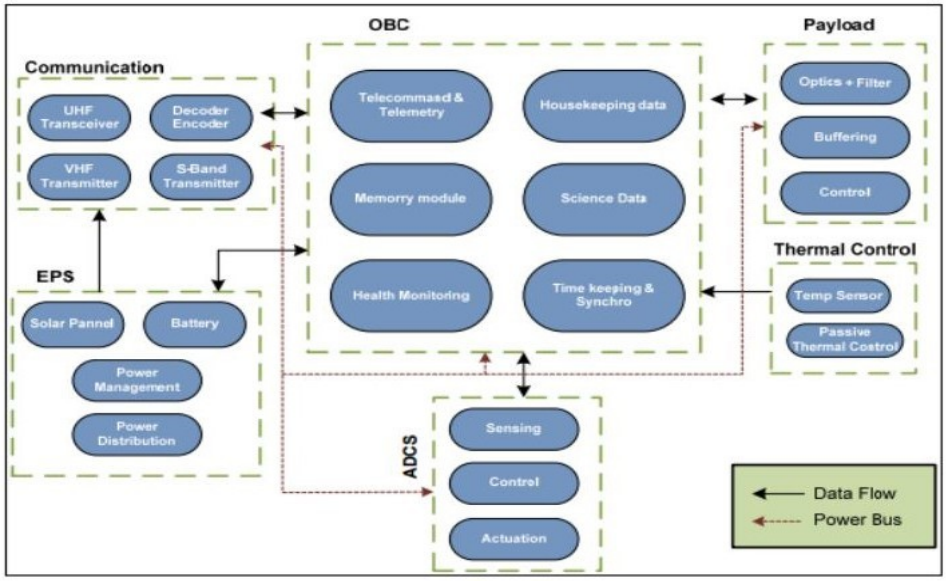

Figure 1: system model

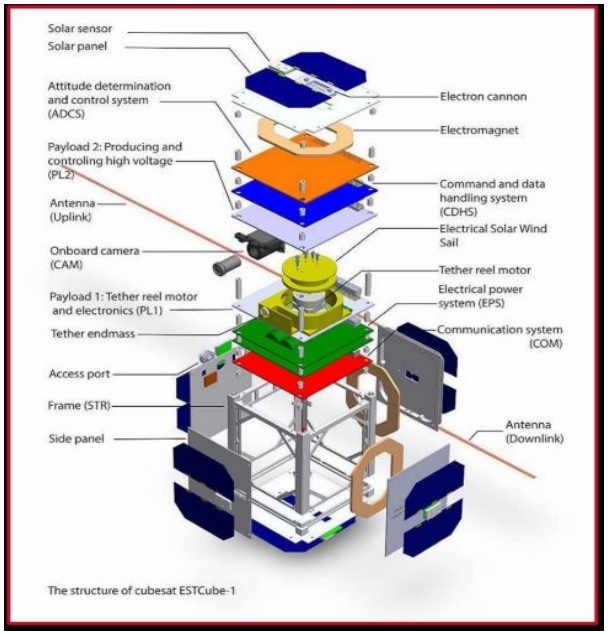

Figure 2 : architecture of the SmallSat

3. Theory and calculation

The value of the digital test considers the operationnal value of the requirements [4]. The requirements associate the value *n* with the IS-s *ID*.

The value 1 names "to carry out the IVVQ phase upstream" with a security system requirement "the on-board computer must be able to check the feasibility of operational concepts.

The value 2 names "data generation for anomaly and incident processing" with a security system requirement "the on-board computer must be able to test for anomalies and incidents".

The value 3 names "management and monitoring of architecture and test requirements" with a security system requirement "the on-board computer must be able to provide data for drafting requirements".

The value 4 names "help in designing test scenarios with a security system requirement" the on-board computer must be able to test scenarios for system design and validation.

The system methodology shall integrate the parameters of the choice of optimized MBSE model, the capacity to visualize optimal variants of the OBC subsystem relationship, the parametric diagram to check the blind spots between the sub-system processing. The software methodology considers the blocks approach from the agents, the containers, the virtual machines with the possibility to introduce an AI engine if the data sets from uses cases are available on the ground segment. The input and the ouput from the system and the sub-system considers the bus from the pipelines via the API. The bonus of the digital tool takes the possibility to add the simulation tool to improve the capacity of the tests. At last, the software approach on the on-board computer is viewed as the software block where the definition of the parameters is linked with the code value.

This baseline must be re-word in a scenario of test from the operational value. For the test, the pre-scenario considers the data following : the clock is onfigured for high frequency as 16 MHz, the I/O data transfer is evaluated by 5 kbits to energy subsystem, the UART communication considers 6 kbits to ground with 4-bit MTU frame authentication for the transmission, the calculation time is reduced to 2 ms. The on-board computer is represented by the msp430 on the figure 3 and the parameters are displaying on the figure 4 and 5.

```c
#include <msp430.h>
#define MTU_AUTH_MASK 0x0F  // Masque de 4 bits pour l'authentification

void configClock() {
    // Configurer DCO (Digitally Controlled Oscillator) pour 16 MHz
    BCSCTL1 = CALBC1_16MHZ;
    DCOCTL = CALDCO_16MHZ;
}

void configUART() {
    P1SEL = BIT1 + BIT2;    // P1.1 = RXD, P1.2 = TXD
    P1SEL2 = BIT1 + BIT2;
    UCA0CTL1 |= UCSSEL_2;   // SMCLK
    UCA0BR0 = 1666;         // Baud rate = 6 kbps @ 16MHz
    UCA0BR1 = 0;
    UCA0MCTL = UCBRS0;      // Modulation UCBRSx = 1
    UCA0CTL1 &= ~UCSWRST;   // Initialiser le module USCI
    IE2 |= UCA0RXIE;        // Activer l'interruption pour RX
}
```

Figure 4 : OBC parameters

```c
void sendDataToEnergySubsystem(const char* data, unsigned int length) {
    // Simuler le transfert de données de 5 kbps vers le sous-système énergi
    while (length--) {
        P2OUT ^= BIT0;  // Envoyer un signal (exemple simplifié)
        __delay_cycles(3200); // Délai pour simuler le débit de 5 kbps
    }
```

Figure 5 : OBC and power subsystem

The parameters from figure 5 are integrated in the docker container as system function and can be ready to be executed with the others potential sub-system in the figure 6

```
# Construire l'image Docker
docker build -t msp430-satellite .

# Exécuter le conteneur
docker run --rm msp430-satellite
```

Figure 6 : OBC system in the docker container

The OBC system from the figure 6 is linked with the software bus by a bridge represented supporting by the name of a network, the number of ports to connect the other containers. The operating system works on the linux through the Debian distribution displaying on the figure 7 where the use case is llined with the msp430-satellite processor.

```
version: '3'
services:
  msp430-satellite:
    build: .
    networks:
      - my_network
    ports:
      - "8080:8080"

  other-container:
    image: debian
    networks:
      - my_network

networks:
  my_network:
    driver: bridge
```

Figure 7 : Bridge of the container

4. Results and Discussion

4.1 Build the environment

Considering three technologies : Radio Frequency, optical, Quantum, the simulation is built after the preliminary design system encompassing the subsystems from the models already existing. The Space segment is linked with the computation of the orbits parameters at the moment where the signal is available to run or to test in the real condition. For this, the OPS-SAT is the best solution for the system and the subsystem for the connection is already semi-opened. The Ground segment requires a control center with one or several stations to send and to receive the uplink and the downlink data from the satellites in Space and the interaction with the systems of the operational control center. The building phase needs modules, software, hardware integrating in a plateform or simply in a development environment. The first methodology uses the lab in Space through the OPS-SAT features from the assembling of hardware on the physical platform. The main list of hardware to be considered may follow the examples of satcatalog depending of the test required.

The generic components include the ACDS like GNSS/GPS, the CDH with the on board computer, avionics packages, atome clocks, the processors and the memory, the power through the batteries energized by solar panel or solar cell, the communication system, the propulsion, the instrument like camera and the structure element. The critical components should choose the platform via the CDH with a focus on the on board computer. This manages the other subsystems fo the satellite.

Depending on the manufacturers, an alternative solution is to get a processor including several functions with the way to simulate data back to Earth like the microcontroller ATMEGA 326. The brain of the satellite can be tested by this unit with many sensors by copying the code to the microcontroller. Other methodologies would insert the code by its image within a container. The software is based on the micro-services architecture through kubernetes with the

use of image of each susbsystem. The image is built from the data expected for the SmallSat or from data of a foundation model. An app generation is the last brick to the test factor and it can be completed by other tools to improve the features of the simulation. The ground segment can gather YAMCS, YAML files, Python libraries, Open C3, Open MCT. Or else, the OPC via the sensors is built through PiCam and Zybo. The on board computer can works via Basilik and Vizard. All must follow the architecture of Satellite manage via the nanosat-mo-framework for the use case including test on apps running by computer on board.
The other is to buid the ground segment through each subsystem then with the connection composed of the all hardware components.The interaction doesn't exist yet because these steps orient the technologic choices depending on the mission requirements.
The point at this step is the design of the API to implement the links between the simulated systems and sub-systems. And the target of the test is the computer on board able to manage by the bus the behavior of the systems. The software architecture is based on micro-service with dockers, kubernetes and other tools as Open MCT YAMCS and Open C3. The point is to create an environment development able to provide telemetry from Spacecraft. The code is available here : https://github.com/nasa/openmct-tutorial/blame/part-b-step-1/example-server/spacecraft.js#L5. The file is on Jason and would integrate in the Open C3 via an API.
The computer on board is included on the object to be created to represent the fictional subsystem of spacecraft. The run consists in connecting each module to validate the work of the system from electrical model. Each module shall be simulated through the application layer from the design orientation. It can be improved by data containers with the foundation model from each modules. As the data expected already exists, the run can be applied directly on the data from the containers at each phase of sub-system delivery.

4.2 Run the environment
The way to run the ground segment depends of the test strategy and the targer choosen regarding the critical aspect viewed by the project. It can be the sensors, the antenna, the electrical module. The point is to use tools for listening the satellite from the orbits parameters and the software processing go back on Earth. The orbits parameters from the Two Line Element verify the positions in Space used by NASA plus NORAD and the links with the ground by the swath width to run the test in the slots. The software processing should take account of the requirements of Space data link security protocol. Through the consultative committee for Space data systems, it provide the methodology to apply authentication and the confidentiality on the data processing. The running of the ground from the connection to the computer on board is used for the test. This is used from NanoSat framework which is useful to generate apps on the computer. Then, the NanoSat framework to run the ground to Space by apps can be improved by the micro-services solution on full software. The data containers from the data foundation model are created to be closer of the signal processing to Space. For example, they work the telemetry, the flight parameters, the memory and the connection to the sensors.
To make the test, it needs to go the NanoSat framework [5]. From the https link, copy and past on your own github and clone it. It may be possible to open via visual code.
The advantage of the environment is to make separate API for the subsystem wanted to the test and to install the SatDump or other tools to listen the exchanges of the system containers with the satellites on real. The containers would be configured from the code from the NanoSat framework. For example, the framework gets already the code structure for the scheduler, the OBDWParameter, the sensor as the camera.

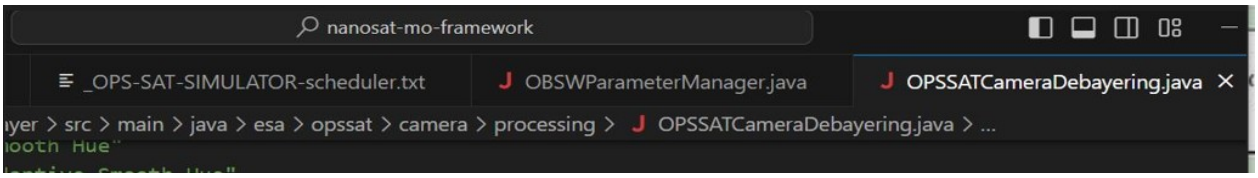

Figure 8 : the package available from nanosat-mo-framework

The visual code environment proposes extensions available to built the container via dockers and kubertenes and make the potential instance via YAML extension. The most uselful is docker file. The step uses the instruction "add" to insert the files expected for example : "OBSWParameterManager.java" in the container created.

```
1 ADD . /app/
```

Then, once the container built, the image is already to be integrated in the architecture to run with the data concerning the satcamera with the ports identified. These steps can be repeated for each subsystem of the satellite. The figure 9 shows the the function for the camera.

```
docker run -d -p 2460:2460 satcamera
```

Figure 10 : Docker image for camera function

The process can be extended via docker hub if you need to make the test on the bigger system. The test can be launched by the docker compose connecting the containers created for each subsystem. The monitoring may be deployed by other tools like Open C3 with its monitoring and observability function applied on the logs of the ports from the containers created. Each instructions are available to test the scenarios to visualize the behaviour the satellites and its capacity to response thanks to a good configuration. For instance, on a docker file, the test would introduce an abnormal temperature. By the command on line, the test will be to check the temperature on software board or to exploit a misconfiguration to re-write the right level. The simulation doesn't replace the test on real on the chambers once the satellites assembled. It provides support on the process between the preliminary design and the assembly, integration and test phase, the operational activity in Space. The simulation gives a environment test to monitor and control the satellites to understand anomalies to be study and to be corrected on a short times. The case of Voyager I and II probes which was updated by the fortran language directly on the computer on board from billions miles demonstrate the robustness test well executed on simulation would provide good execution result in Space.

6. Conclusions

The test the digital Space Systems by the tools engineering offers the capacity to support the design of the Space system to test the requirements planned for the hardware before to engage the means. This approach considers the need to discuss and to configure the functions with the parameters expected for the mission, the performance, the safety and the security by design.

References

Reference to a regulation publication:
[1] ISO-42010. 2011. "ISO 42010- System and software engineering - Architecture Description". New York : s.n., 2011.

Reference to a book:
[2] "Multiplexed networks for embedded systems: CAN, LIN, Flexray, Safe-by-Wire...". s.l. : John Wiley & Sons, 2007.
[3] Small Satellite Mission for Earth observation : new developments and trends, Rainer Sandau, 2024.
[4] "Software product line engineering : Foundations, principles and techniques.". s.l. : Springer, Pohl, K., Böckle, G., & van Der Linden, F. J., 2005.

Reference to a journal publication :
[5] Van Huong, P., & Binh, N. N. 2012. "Embedded system architecture design and optimization at the model level". International Journal of Computer and Communication Engineering. 2012, Vol. 1, 4, p. 345.

Reference to a website:
[6] https://github.com/esa/nanosat-mo-framework