Activity profile module specifications

April 2019

1 Introduction

NANOSTAR is a European SUDOE project. One of its goals is to develop a software suite to enable student teams and university staff to perform preliminary analyses and design small satellites.

Nanostar Software Suite (NSS) needs a **visualization module** that allows the team to communicate internally, to check and validate mission analysis and scenario and to get pretty communication outputs. This visualization module shall interact in terms of input/output with other software in the NSS.

The goal of this module is

1. to facilitate satellite mode management (the activity profile of the spacecraft during the mission)
2. to ease NSS interaction with VTS, a multi-purpose visualization tool developed by CNES.

2 Client

The client for this contract is Universidad Carlos III de Madrid. Not withstanding this, ISAE Supaero shall be the main point of contact regarding clarifications, questions, and validation of the software, documentation, examples, etc produced in the framework of this contract.

3 The Nanostar Software Suite

For a whole descritpion of NSS, please refer to NANOST-OUT-058 document[1]

[1]https://extranet.aerospace-valley.com/Extranet/NANOSTAR

4 The VTS software

As stated in the VTS official webpage[2], *"The main purpose of VTS is to animate satellites in 2D or 3D environments. The architecture of VTS has been designed as an extensible platform, able to work with an unlimited number of compatible applications. [...] VTS is a helpful tool for all activities implying space data. It can be used as a graphical validation method, as an educational tool, or as a communication support for all exchanges between space experts."*

VTS is cross-platform and, even if it is not open-source, VTS is distributed as a freeware and provide connection interfaces with classical languages (Java, Python...).

5 Visualization module capabilities

The activity profile module aims at helping the engineering team to deal with the activity profile management and interfacing VTS with NSS. This interface can be seen as a third-party application (Cf. Fig. 1) that will make easier the connection between NSS-Core and VTS broker. For the user, VTS-broker can be seen as the module taking care of the time line management (bottom frame on Fig. 2). It contains a front-end part (for user interaction) and a back-end part which ensures interconnection between NSS-Core and VTS.

For a whole description of NSS-Core, please refer to NANOST-OUT-058 document. For more detail on VTS-Broker, please refer to VTS documentation.

6 Module inputs

- CCSDS standard OEM (Orbit Ephemeris Message) data (in a json format - Cf. listing 1)
- CCSDS standard AEM (Attitude Ephemeris Message) data (in a json format - Cf. listing 2)
- modes listing and description (json format) - Matrix of equipment power consumption verse satellite mode (ex: Listing 3)

7 Usecase

Let's consider that we have already the orbit, no propulsion system, and 2-3 orbit as OEM file. From there, you can build your activity profiles (get compulsory sequences, with the wet thumb, 6 hours of detumbling, after that, operational mode engage, alternating "gathering payload data" and "com mode" (for instance). A first activity profile is ready. From it, EPS check consumption of equipments. Ho, no: there is not enough power ! And Data budget is not

[2]https://logiciels.cnes.fr/en/content/vts

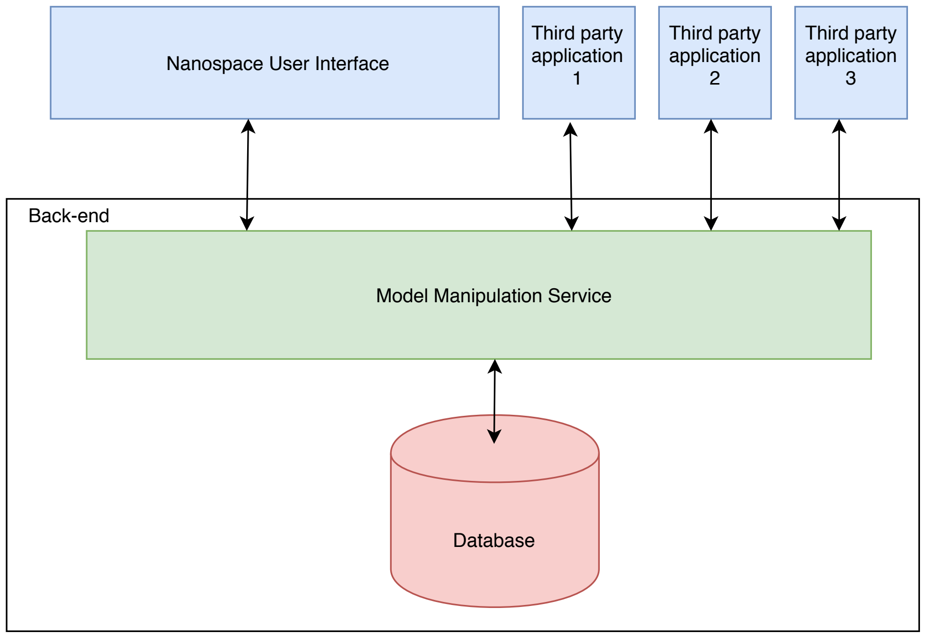
Nanospace User Interface
Third party application 1
Third party application 2
Third party application 3
Back-end
Model Manipulation Service
Database

Figure 1: Nanospace overview. The back-end is the NSS-Core, while third party application ensure the consistency of each module of the NSS. The module describe here is a Third Party applcation.

valid anyway : we are producing too much data flow ! Ok, let's go back to the activity profile : we are going to "sleep" during eclipse (convenient to have this kind of data available i, activate payload during "sunny" moment... and produce half of the previous data. Thanks to the Activity profile module, it's very easy to update! Before, we were modifying an horrible excel sheet to do so... A new activity profile is generated. EPS check, Data budget check. We can generate AEM data of the whole orbit thanks to the preliminary ADCS module, and check the feasibility. Let's suppose it's ok : now, we have a valid activity profile, valide OEM data, valid AEM data. We can then feed VTS with it for visualization. Ho no, during visualization, we detect that payload is not activated on the best moment in the orbit.. so we can go back to the Activity profile module, and update it. At this point, remains some interrogation : to what extend can we dynamically re-generate the whole process leading to the update of VTS visualization, I'm not sure. But in any case, the whole previous process is valid. To answer you: uou can see AEM as a final input that will be provided to VTS...

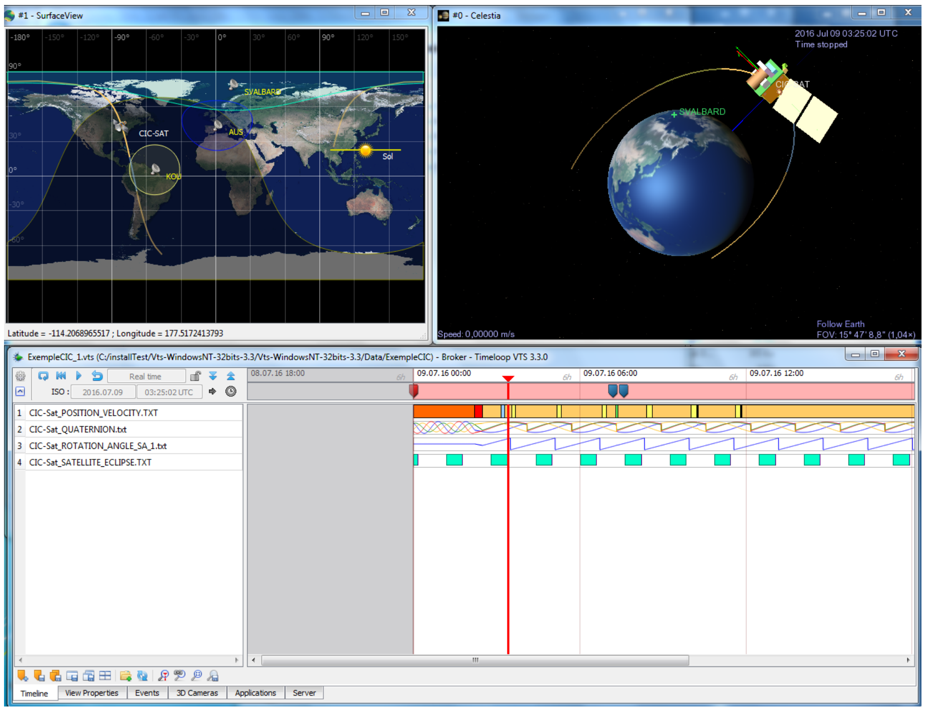
#1 - SurfaceView
#0 - Celestia
SVALBARD
CIC-SAT
AUS
KOU
Sol
2016 Jul 09 03:25:02 UTC
Time stopped
Latitude = -114.2068965517 ; Longitude = 177.5172413793
Speed: 0,00000 m/s
Follow Earth
FOV: 15° 47' 8,8" (1,04x)
ExempleCIC_1.vts (C:/installTest/Vts-WindowsNT-32bits-3.3/Vts-WindowsNT-32bits-3.3/Data/ExempleCIC) - Broker - Timeloop VTS 3.3.0
Real time
ISO : 2016.07.09 03:25:02 UTC
08.07.16 18:00
09.07.16 00:00
09.07.16 06:00
09.07.16 12:00
1 CIC-Sat_POSITION_VELOCITY.TXT
2 CIC-Sat_QUATERNION.txt
3 CIC-Sat_ROTATION_ANGLE_SA_1.txt
4 CIC-Sat_SATELLITE_ECLIPSE.TXT
Timeline
View Properties
Events
3D Cameras
Applications
Server

Figure 2: Illustration of some VTS frames.

8 Outputs

- Activity profile CSV file (timestep - mode)
- Activity profile data (json formatted)[3]
- VTS[4] file (.vts) (respecting VTS input/output format)[5]

9 Requirements

9.0.1 Requirements typology

- Requirement that should be respected "as is" will be designated with a bullet (•) in this document.

[3]Note that the activity profile can be refined during the process, so it will also be considered as an input

[4]https://logiciels.cnes.fr/en/content/vts

[5]in order to ease modularity

- Requirement that can be adapted, in accordance with the development team, will be designated with a circle (○) in this document (i.e nice to have).

9.1 Global requirements

In every module, we have these following requirements:

- independent library for any domain specific tools that can be easily called by third-party software;
- well documented API on this library;
- *Python* API for domain specific calculus;
- Python command line example using this API;
- Backend that implements this API in REST;
- Front-end en Angular qui implémente l'interface REST;
- *Angular* based front-end which implements the REST interface;
- modules must be open-source, with an AGPL license[6];
- each module must have a stand alone version, working in a local way;
- code quality including test unit for each methods, integration and performance tests for whole module;
- ISAE-SUPAERO's Sourceforge platform will regroup source code deliveries and documentation, organized in one global project, subdivided in sub-projects. Each sub-project includes a git deposit[7].

9.2 Deliverables

In every module and for NSS core and modules, we have the following requirements:

- mock-up for validation and approval by the client; then, actual software
- source code, organized in a standard project folder structure:

```
.
├── doc/
├── README.md
├── LICENCE
└── src/
```

[6]https://www.gnu.org/licenses/agpl-3.0.en.html - accessed March, 1st 2019
[7]e.g. https://sourceforge.isae.fr/projects/nss_activity_profile

- working stand-alone example;
- quickstart guide;
- REST API documentation;
- REAMDE.md (markdown) file describing dependencies, compilation process, running example, running tests and limitations;

9.3 Specific Module requirements

9.3.1 Usage

The module GUI should allow the user to:

- read different OEM file (spacecraft, ground stations...)
- read an OEM data in JSON format (Cf. Listing 1)
- read an AEM file and data in JSON format
- read alternative data such as eclipse schedule in JSON format (Cf. Listing 2), ground station visibility or other types of orbital events (sky observation window, any orbital event...)
- define different activity profiles for a same mission (e.g : different profile on different orbital cycle, or tabbing different profile on the same orbital cycles)
- update spacecraft's modes definition and associated parameters
- allow to update dynamically parameters of these two types of data (e.g power consumption may depend of the mode, etc.)
- provide tools to visualize these profiles in an efficient and ergonomic way
- provide tools to edit these profiles in an efficient and ergonomic way (drag and drop existing modes, manage a scenario timeline, etc.)
- provide tools to visualize simple matrix (time-step / value) to allow specific data visualization (Ex: DoD data, data coming from another module, like payload data).

```
{ "data" : [
{"epoch": "2019-02-22 20:09:59",
"vx": -1398.2534207,
"vy": 4700.0781515,
"vz": 5804.1131447,
"x": 6831358.3554319,
"y": 810625.9372051,
"z": 1001039.6648682},
{"epoch": "2019-02-22 20:10:00",
"vx": -1406.3577919,
"vy": 4699.11358,
"vz": 5802.9219982,
"x": 6829956.0496841,
"y": 815325.5335352,
"z": 1006843.183013}
],
"meta": {
"CENTER_NAME": "Earth",
"INTERPOLATION": "lagrange",
"INTERPOLATION_DEGREE": 5,
"OBJECT_ID": "Brick-Sat",
"OBJECT_NAME": "Brick-Sat",
"REF_FRAME": "EME2000",
"START_TIME": "2019-02-22 18:30:00",
"STOP_TIME": "2019-02-22 20:10:00",
"TIME_SYSTEM": "UTC",
"USEABLE_START_TIME": "2019-02-22 18:30:00",
"USEABLE_STOP_TIME": "2019-02-22 20:10:00"}}
```

Listing 1: Example of OEM data, in json format

```
1,0,0,0,
0.9757431,-0.0408880,0.0997367,-0.1905415,
0.9871960,0.0295592,-0.0726097,0.1389177,
0.9960149,0.0163814,-0.0405589,0.0777235,
0.9957313,-0.0167863,0.0419298,-0.0804943,
0.9947715,-0.0183712,0.0463393,-0.0891340,
0.9910906,-0.0236726,0.0603569,-0.1163442,
0.9978004,-0.0116284,0.0299989,-0.0579588,
```

Listing 2: Example of AEM CSV file. Here time step is implicit.

```
Equipment,Launch,Nominal,Transmit,Heating,BUS
Gomspace AX100 UHF/VHF Transceiver,0,0.396,2.805,0.396
ISIS VHF UHF Antenna,0,0.06,0.06,0.06
ISIS VHF UHF Antenna Dipole Configuration Base,0,0.06,0.06,0.06
NanoPower BP4 G Option,0.3,0.3,0.3,0.3
NanoPower P31u E Option,0.16,0.16,0.16,0.16
Interface CU,1,1,1,1
Gomspace - Fine Sun Sensor,0,0.01155,0.01155,0.01155
NINANO,2,2,2,2
ISIS Magnetorquer,0,1.2,1.2,1.2
Heater,0,0,0,0
PAYLOAD,MODES
Equipment,Edmon,Radmon,Mission,Off,BUS
```

Table 1: Human format for Listing 3

Equipment	Launch	Nominal	Transmit	Heating
Transceiver	0	0.396	2.805	0.396
ISIS Antenna	0	0.06	0.06	0.06
ISIS Antenna Op 2	0	0.06	0.06	0.06
NanoPower G Option	0.3	0.3	0.3	0.3
NanoPower E Option	0.16	0.16	0.16	0.16
Interface CU	1	1	1	1
Gomspace - Fine Sun Sensor	0	0.01155	0.01155	0.01155
NINANO	2	2	2	2
Magnetorquer	0	1.2	1.2	1.2
Heater	0	0	0	0
PAYLOAD MODES				
Equipment Radmon	Mission	Off	Dual	
Payload Edmon	10	0	10	
Payload Radmon	0	0.3	0.3	

```
Payload Edmon,10,0,10,0
Payload Radmon,0,0.3,0.3,0
```

Listing 3: Example of mode listing, in CSV format, linked with power consumption

```
[{"start": "2011-12-01T18:09:01.455",
"end": "2011-12-01T18:41:30.556"},
{"start": "2011-12-01T19:48:49.577",
"end": "2011-12-01T20:21:19.340"},
{"start": "2011-12-01T21:28:37.701",
"end": "2011-12-01T22:01:08.124"},
{"start": "2011-12-02T12:26:50.896",
"end": "2011-12-02T12:59:27.180"},
{"start": "2011-12-02T14:06:39.038",
"end": "2011-12-02T14:39:15.964"},
{"start": "2011-12-02T15:46:27.181",
"end": "2011-12-02T16:19:04.748"}]
```

Listing 4: Example of eclipse timeline data, in json format

```
[{"date": "2011-12-01T18:09:01.455",
"mode": "1", "modeHumanReadable": "idl"},
{"date": "2011-12-01T20:21:19.340",
"mode": "1", "modeHumanReadable": "idl"},
{"date": "2011-12-01T22:01:08.124",
"mode": "2", "modeHumanReadable": "detumbling"},
{"date": "2011-12-02T14:06:39.038",
"mode": "2", "modeHumanReadable": "detumbling"},
{"date": "2011-12-02T14:39:15.964",
"mode": "2", "modeHumanReadable": "detumbling"},
{"date": "2011-12-02T15:46:27.181",
"mode": "3", "modeHumanReadable": "operation"},
{"date": "2011-12-02T16:19:04.748",
"mode": "3", "modeHumanReadable": "operation"}]
```

Listing 5: Example proposition of activity profile, in json format

9.3.2 Technical proposition

- should be totally compatible with VTS broker (TCP sockets connections, VTS contains full documentation about it)
- data formatted as json string and CSV files should be provided

9.3.3 Tasks to be achieved

- formalize an activity profile data - list of couple (time, type of activity) - loop-back with Nanostar Development team;
- formalize satellite mode data - loop-back with Nanostar Development team;
- propose a user interface (front-end, web page) for manipulating a schematics representation of an activity profile (Cf. VTS broker visual on Fig. 2). Loop-back with Nanostar Development team.

  Suggestions consist in:

  - time-line creation, manipulation,
  - drag and drop of satellite modes on the time-line;
  - visualize or hide eclipse scheduled on the time-line;
  - navigate through the time line

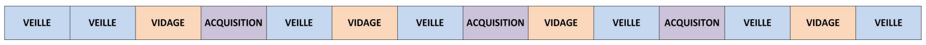

Figure 3: Example of a visual timeline with satellites modes. Imagine then to drag and drop "mode" boxes...

- implement it as an angular web interface making easy to describe an activity profile;
- define and implement an angular web interface making easy to modify the different satellite modes;
- provide a library implementing functionality for manipulating it;

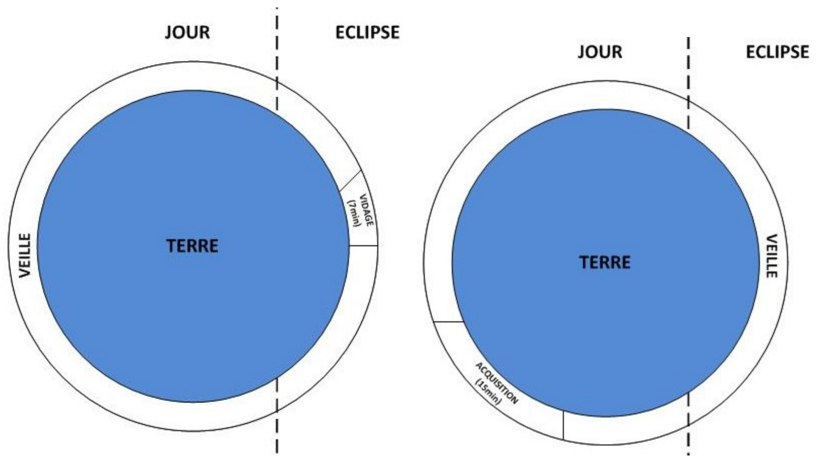

Figure 4: Visualization of two kind of orbital activities.

- provide a REST API according to this library;
- update VTS broker when navigating on the time line.