VTS: a long-term approach for synchronization of all visualization software

Thomas Crosnier[(1)], Mathieu Joubert[(2)], Quentin Minster[(2)]

CNES, Centre spatial de Toulouse, CNES[(1)]
18 avenue Edouard Belin, 31401 Toulouse Cedex 4, France
thomas.crosnier@cnes.fr

Spacebel SAS[(2)]
Technoparc 8, rue Jean Bart, 31670 Labège, France
mathieu.joubert@spacebel.fr
quentin.minster@spacebel.fr

INTRODUCTION

Visualization of data, results and telemetry is a large and omnipresent topic in the daily activities of space agencies and their partners. While some representations like graphs are transversal, others are more domain-specific: coverage analysis, sensor camera simulation, 3D geometry around the Earth, etc.

Without any agency-wide policy, the "visualization" functionality is implemented into each tool which needs it (Fig. 1). The implementation is thus highly dedicated to a specific environment: programming language, framework and design.

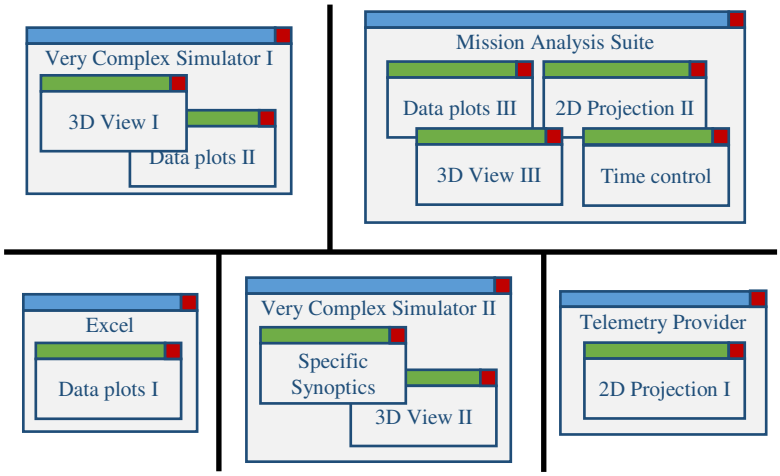

Fig. 1: Pre-VTS organization of visualization tools

In the first place, application designers may feel the need for a visualization component dedicated to their environment. Custom software can appear as a perfect answer to complex requirements. Indeed, the final visualization functionality is fully integrated with the application and exactly addresses the user's needs.

But this development model has many drawbacks. Overall cost is obviously very high since each project has to re-implement 2D, 3D, plots, etc. from scratch. Development may also be carried out by non-experts, which further increases costs, especially for 3D components which have an unforgiving learning curve for developers.

The first answer to these issues is the development of a common visualization facility (Fig. 2). This way, development is in theory made once for all applications.

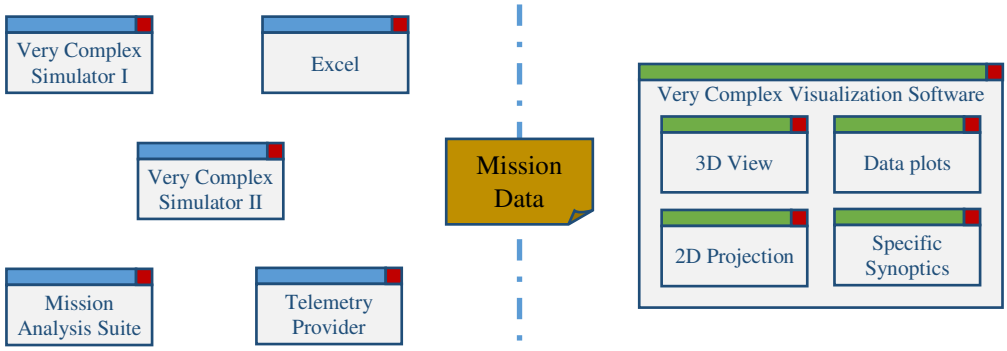

Fig. 2: Common visualization facilities

In spite of this unification effort, some factors lead to the cyclic emergence of new visualization tools:

- Indecisive communication within the agency: talented and enthusiastic engineers produce their own specific tools, mainly 3D
- New technologies or software make tools permanently obsolete
- Industry partners propose new tools all the time

This instability prevents building up a capital of skills and standard processes, both in the long term and in the diversity of a space agency's activities.

A FUTURE-PROOF DESIGN

Instead of developing once more a brand new product that integrates standard features for spatial work like 3D in Cosmos, Mercator view, orbit propagation, etc., CNES decided on the conception of an open framework able to integrate various software components, including already existing ones. The core idea was to provide a synchronization protocol between heterogeneous applications in order to leverage them together for the visualization of common data (Fig. 3).

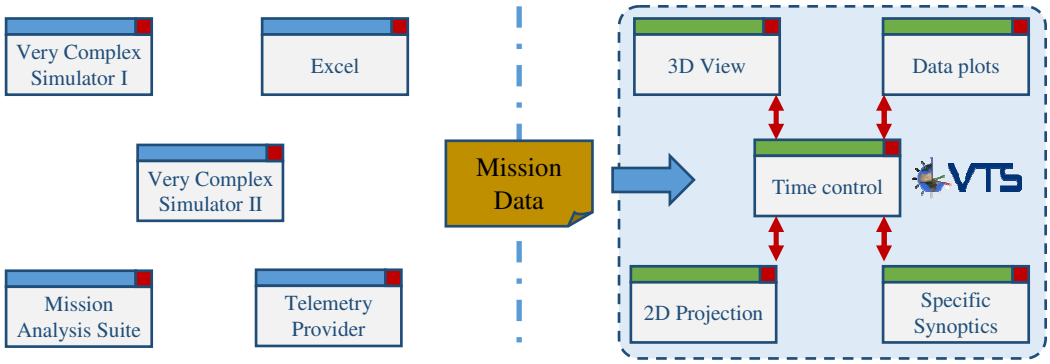

Fig. 3: Modular design of VTS

This approach can federate all visualization activities at CNES. Acceptance of the VTS toolkit is increased by its flexible design: existing software can be interfaced with VTS with little modification.
Highly specialized software particularly benefits from being integrated with the toolkit, as this allows it to focus on its core features rather than having to develop generic visualization capabilities.

With this paper, CNES wants to provide the Space community with feedback on how VTS benefited all its visualization activities.

SOFTWARE ARCHITECTURE

Time synchronization and control across multiple applications is the backbone of the VTS system. In both real-time and replay modes, applications connected to the VTS central component (the "Broker") share the same visualization context and date.

This strongly correlates the various data sources available, making important events more noticeable by coalescing several symptoms: an outlying point in a plot, a spacecraft orientation shown in 3D, a geographic position on a planisphere, or a particular event in a specialized synoptic (Moon transit across a sensor's field of view, etc.).

Drawing from the many and free software tools shipped with VTS, plus user-made compatible applications, a VTS user can compose a software suite that closely matches his needs.

TECHNICAL CHOICES

In order to successfully achieve the unification of various visualization activities, a number of technical choices were made. They are presented here below.

Single XML configuration file

The VTS configuration file holds configuration information needed by visualization components to set up the visualization: spacecrafts, ground stations, sensors…

The configuration file is written in XML format. This format emphasizes understandability of the file contents by human readers, while being extremely well supported by most programming languages. This allows third-party software to easily generate its own visualization configuration files.

The configuration file is also comprehensive: it contains configuration information for all synchronized applications and all entities taking part in the visualization. This makes sharing and duplicating visualization contexts amongst VTS users a breeze.

Synchronized applications need not be modified to parse the VTS configuration file. Interoperability with VTS can be provided through application-specific proxy utilities (the "Launchers") with the ability to extract required information from the VTS configuration file, and prepare the visualization environment of their applications accordingly.

Human-readable input data format

Input data such as (but not limited to) position and attitude ephemerides of visualized spacecrafts must be provided to synchronized applications.

Data files must be written in the "CIC" format. It is a compact text format inspired by and resembling the CCSDS OEM and AEM formats. It is human-readable and easy to understand, with a well-defined header describing the data. Data can be either numerical values (integers or real numbers) or text strings, enabling any serializable data to be stored in a CIC file.
This makes CIC data files easy for humans and software alike to generate or parse, facilitating activities such as rapid prototyping, heavy data generation for VTS from third-party tools…

Synchronized applications either need to be updated to handle the CIC format, or may retain their specific data formats. In the latter case, interoperability with VTS must then be provided through proxy utilities (the "Launchers") with the ability to convert CIC data files into application-specific data files.

```
CIC_OEM_VERS  = 2.0
CREATION_DATE = 2012-03-29T11:35:51.706
ORIGINATOR    = CNES - DCT/SB/MS

META_START
OBJECT_NAME = CIC-Sat
OBJECT_ID   = CIC-Sat
CENTER_NAME = EARTH
REF_FRAME   = ICRF
TIME_SYSTEM = UTC
META_STOP

57578  0.00000 -648.783 -6953.936 -15.097
57578 10.00000 -658.843 -6952.662  59.783
57578 20.00000 -668.826 -6950.573 134.657
57578 30.00000 -678.731 -6947.671 209.514
```

Fig. 4: Sample attitude CIC file

Extensible text-based synchronization protocol

Synchronized applications communicate with the VTS Broker in order to receive the visualization time and various visualization-related commands.

Communications follow a client-server model through a standard TCP socket, using a custom text-based protocol. This simple design enables easy integration with existing third-party software.

The synchronization protocol is stateful and versioned. It carries time synchronization messages, time control commands, and a variety of miscellaneous visualization-related commands. It is also used to broadcast data streams in real-time mode, as well as base64-encoded raw data (e.g. screenshots).

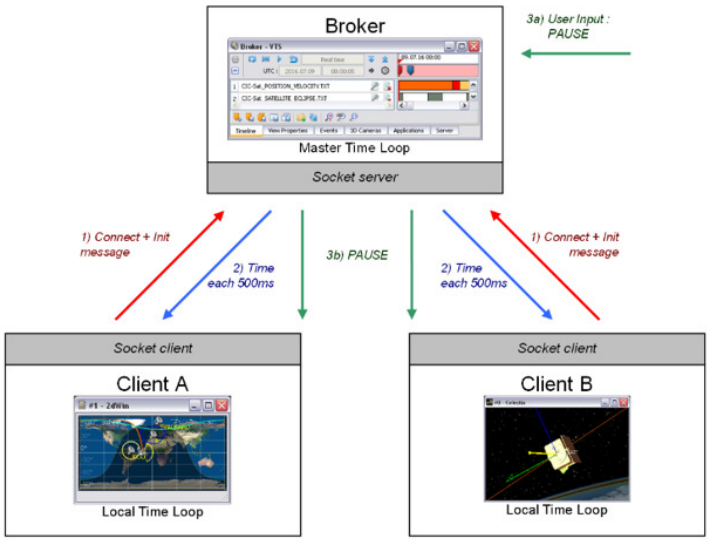

Fig. 5: VTS synchronization protocol

The protocol can be extended to transmit application-specific messages, or pre-declared data values using a variety of supported data types. Applications only need provide VTS with a standard INI file describing the data values they may receive through the synchronization protocol during a visualization.
Actual data may then be sent to synchronized applications manually from the GUI or using the scripting features of the VTS Broker. This enables fine-grain control over the visualization parameters for all compatible applications, making VTS ideal for generating movies or preparing live demos.

Finally, the communication channel is a two-way link, enabling external applications to send visualization commands or time synchronization messages towards VTS, effectively taking control of the visualization.
This allows VTS to be used in real-time mode where time synchronization is provided by an external application (e.g. a spacecraft simulator).

High-accuracy time synchronization engine

VTS features a sophisticated time synchronization engine tasked with sending time synchronization messages ("ticks") to synchronized applications at a rate of 500 ms (wall clock time). This engine supports high-accuracy restitution (currently millisecond-level in visualization time, with nanosecond-level planned) of scripted commands and smooth video capture, even at very high visualization speed or on old hardware.

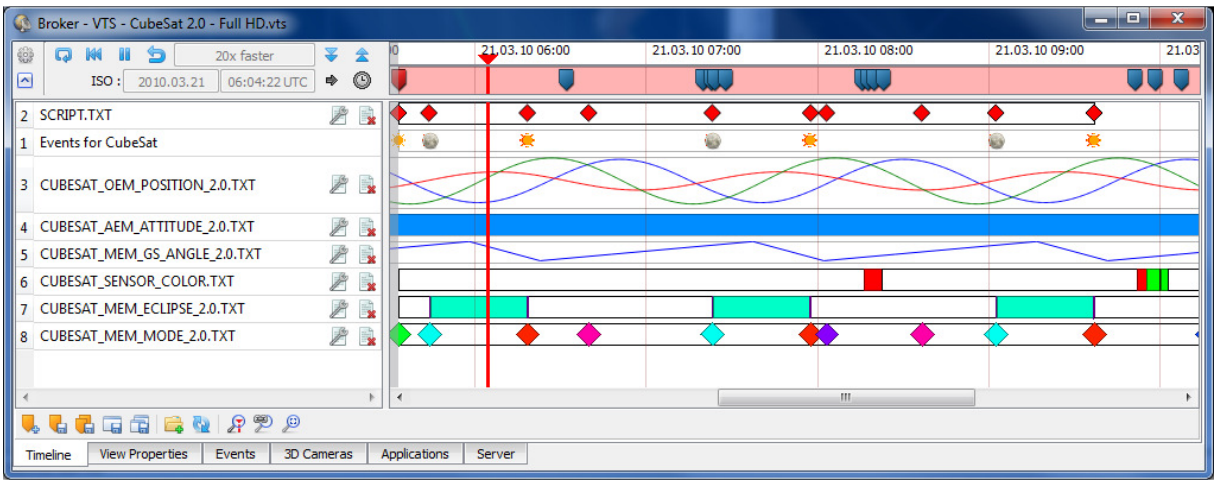

Fig. 6: Advanced time synchronization engine

Stable interfaces

All interfaces to external applications are fully documented, with examples, in the VTS user manual. They are versioned and, as much as possible, kept stable over time. Backwards compatibility is assured for VTS-owned formats, i.e. the synchronization protocol and the XML configuration file.

LIMITATIONS

In order to remain consistent with its design philosophy, the functional scope of VTS is purposely limited to visualization only. Data production activities are kept to a bare minimum. It is up to the user to provide the data, using the tools of his choice.

In particular, VTS does **not** offer the following features:

- Orbit propagation
- Attitude generation
- Environment models
- Mission analysis / models
- Satellite simulation
- Electrical and thermal simulation

However, there are a few known use cases for data production with VTS. Such cases include Celestia-based computations of the insulation for solar arrays on Rosetta's Philae lander, and Celestia-based computations of LIDAR depth maps for LIDAR simulations. These cases stand for R&D projects and are not publically released.

Event when handling user data, VTS specifically ignores as much as possible about the data contents, except for core entity configuration data such as position or attitude ephemerides. The preferred policy for non-core data is to provide the data to the applications and let them handle it if they can. VTS only gives special status to core data such as position and attitude ephemerides of spacecrafts and their mobile components.

These limitations, which result on continuous design choices, make VTS a pure visualization software. VTS does not imply to replace existing space domain software, and it will never go through its functional scope.

BENEFITS

The VTS approach to visualization ensures continuity in the long term for visualization activities at CNES, as evolving requirements will be covered thanks to the interchangeability of applications within the VTS framework.

VTS does encourage reuse of existing applications, which reduces investments in software development as only new requirements need to be addressed. Even then, if alternative software emerges instead of one of the main applications of VTS like the 3D view Celestia, it can be adapted to VTS and gradually and seamlessly replace it. While VTS is not the definitive answer to the recurring software development costs of visualization tools, as its acceptance grows it helps foster interoperability between all visualization tools.

VTS also benefits its users by indirectly sharing ideas from users all over CNES, due to its development model: as CNES projects contribute new features to VTS, those features are made available to all VTS users, providing them with possibly new tools and ways to visualize and analyze their own data.

VTS USERS

In 2015, having been developed for six years on, VTS is widely used at CNES: ATV, Galileo, all operational activities, the Rosetta probe (Fig. 1), spacecraft simulators, concurrent engineering, etc. It is also gaining traction outside CNES, e.g. at Airbus Defense & Space.

Historically, VTS users have always been very involved in the direction taken by VTS, mostly through feature requests supporting specific workflows but also as validation aids during development of those new features. This strong interaction with its user base has ensured solid adequacy of the VTS toolkit within CNES.

As a particular example, CNES's Centre d'Ingénierie Concurrente (CIC, or Concurrent Design Facility) has been from the early days a strong partner and advocate of VTS, using it and pushing it further in all phases of spacecraft design.

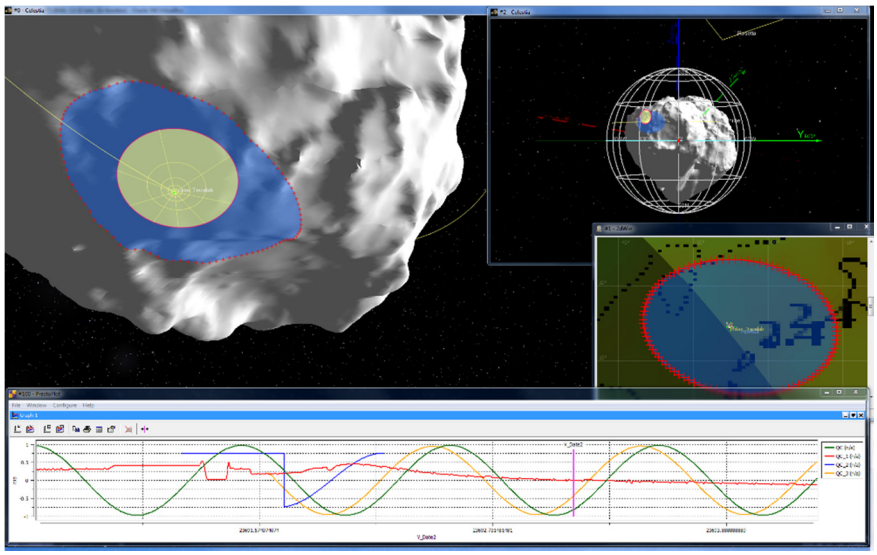

Fig. 7: Rosetta mission

CONCLUSION

With its minimalistic yet rock-solid foundation as a visualization software coordinator, VTS has become a reference point at CNES for visualization activities. Its focus on letting each tool perform its specialty and keeping feature bloat to a minimum has proven successful and has gathered great support from various projects at CNES.

As software becomes increasingly cheap to produce and diversified, and as the Space industry gradually enters the era of Big Data, the VTS approach of enforcing interoperability of its specialized components becomes increasingly vital to successful analysis of the huge amounts of data involved in complicated interactions aboard spacecrafts.

The ability to assemble a software suite from several specialized components to answer a particular question is one of the main strengths of VTS, and as such, it is our belief that the VTS philosophy will succeed in helping engineers solve tomorrow's most complex challenges in spacecraft design and engineering.