

Framework Concept for Satellite Operations

Miguel Angel Molina, Jose Prieto

GMV – c/ Isaac Newton, 11
PTM Tres Cantos
28760 Madrid
SPAIN

Tel: +34 918072100; Fax: +34 918072199

mamc@gmv.es, jjpm@gmv.es

Gonzalo Garcia, Theresa Beech

GMV Space Systems
1 Research Ct, Suite 450
Rockville, MD 20850
USA

Tel: +1 301 216 3840; Fax: +1 301 519 8001

US_marketing@gmvspacesystems.com

ABSTRACT

GMV has developed an internal project, called *FOCUS*, to produce a new generation multi-satellite and multi-mission framework. Project *FOCUS* came through successful requirements definition during the first half of 1999 and the development phase started in August 1999. *FOCUS* has very ambitious goals in mind, in particular the development of a truly generic operational framework for all types of satellite missions (including GEO, LEOP, LEO, satellite formations, constellations, etc.) to be commercialised as a COTS product. This led in the last few years to *FOCUSuite*, a suite of operational products that cover all these types of missions.

1.0 INTRODUCTION

The design of new satellite control systems is driven by the imperative need to reuse legacy software, exhaustively exploited during years of operations which have qualified them as “flight proven”, and the desire to use modern software technologies which dramatically boost system usability, accessibility and stability. In order to fulfill those objectives, the “framework” concept appears as the required support for future on-ground satellite systems development.

Framework concept is thus oriented to minimize program development schedule, costs and risks, and at the same time to improve the efficiency of operations (minimizing operations workload) and reducing the risk of human errors. In order to meet these ambitious goals, the framework should provide a number of ready to use components for data manipulation and visualization as well as event logging, overall framework architecture, communications layers, process management, automation and reporting. Nevertheless, the real power behind the framework lies in its ability to integrate external components. While the integration of disparate software is usually a hard task plagued with difficulties (programs not conceived to act as components, wealth of protocols and formats, unclear interfaces, etc.), a framework makes this issue straightforward. The framework should also provide well-designed, high performance, robust, customizable, extensible, flexible and *coherent* Graphical User Interfaces (GUIs), promoting graphical views as much as possible.

Such a framework would allow integration of flight dynamics algorithms with unprecedented ease. Those flight dynamics functions may come from different origins in terms of programming language. Interface between the framework and the flight dynamics algorithms should be based on the implementation of an API (Application Programming Interface) in order to manage the processing and the data interfaces. Management of new entities (e.g. satellites) in the system should be performed by simply modifying editable configuration files. Those configuration files can be interpreted by the framework system in order

Angel Molina, M.; Prieto, J.; Garcia, G.; Beech, T. (2005) Framework Concept for Satellite Operations. In *Integration of Space-Based Assets within Full Spectrum Operations* (pp. 10-1 – 10-12). Meeting Proceedings RTO-MP-SCI-150, Paper 10. Neuilly-sur-Seine, France: RTO. Available from: <http://www.rto.nato.int/abstracts.asp>.

Framework Concept for Satellite Operations

to create the required database structure, to construct the overall GUI and to generate the required functionality including additional (if needed) generic functions like plotting or data acquisition and pre-processing. The issue of users and privileges needs also to be managed by the framework, in addition to automation which deserves special attention.

focusSuite includes a suite of **generic operational satellite control** systems (initially devoted to Flight Dynamics activities) sharing the same architecture and many common components. Each of them has a different computational layer, all of them based originally on flight proven ESA's packages like PEPSOC, NAPEOS and LEOPOLD. The following stand-alone **products** are included within focusSuite:

focusGEO is targeted to Geostationary satellites (see later on this paper).

- focusLEOP is a complement to focusGEO providing support for the Launch and Early Orbit Phase for geostationary satellites
- focusLEO is targeted to Low Earth Orbit satellites
- focusCⁿ is targeted to satellite constellations
- focusPlanet is targeted for interplanetary missions
- focusReentry is targeted for re-entry missions
- focusRdV is targeted for rendez-vous and formation flying missions

2.0 DESIGN DRIVERS

The following paragraphs describe at a high level functionalities of focusSuite which are common for all the members of the focus families:

- A **computation and data layer** based on the extensive reuse of existing and improved software. A large variety of satellite platforms in different orbit conditions (e.g. GEO, LEO, MEO, constellation, reentry, interplanetary, formation flying, etc.)
- A **client/server architecture**: All data and functionalities reside on a server which is accessed via a client GUI. Communications between client and server are done via TCP/IP. The possibility to work over the Internet with the appropriate degree of security has been foreseen.
- **Database driven**: All important mission data is stored in a database residing on the server side. The possibility to plug a commercial RDBMS like ORACLE into focus has been foreseen, although this is not mandatory being a proprietary database by default.
- An **advanced GUI**: The GUI integrates advanced widgets and a design philosophy based on commercial desktop applications for the office: "everything-in-one-working-area" and "all-one-click-away" (tabs). The GUI implementation is based on a proprietary toolkit called Tkforms that allows a development through configuration files rather than through code.
- Procedures automation capability through the AutoFOCUS extension, which is able to produce an automatic sequence and execution of modules required to fulfill a particular procedure. This automation is based on a dedicated language: SOL (Spacecraft Operations Language) also developed and integrated by GMV.
- Advanced **graphical capabilities**: Dedicated widgets for generic X-Y plots and Events visualization have been developed for focusSuite.
- **On-line help**: Complete User Manual available on-line in PDF format, with hypertext and navigation capabilities.

- **Portability (UNIX/Windows NT):** Both the client and the server part run under UNIX and Windows-NT. Any combination of the two operating systems is possible (UNIX/UNIX, UNIX/Windows-NT, Windows-NT/Windows-NT, Windows-NT/UNIX). Virtually all flavours of UNIX are supported, including Linux. At the client side any of Windows 95/98/2000/NT/XP are also supported.
- **Extensibility:** Any extra functionality following certain I/O rules can be easily integrated within focusSuite via configuration files. A server Interface Description and a graphical Server Manager are available.
- Capability to perform **unlimited Undo/Redo** operations: Any data editing and function execution operation can be undone and redone an unlimited number of times. This allows a quick and user-friendly operating of the product while maintaining at the same time the user confidence and the data integrity.
- **No licenses** of external products are required.

3.0 ARCHITECTURE

focusSuite is a database-driven framework, built as a three-layer architecture (GUI Layer/Computation Layer/Data Layer). Focus is also conceived as a client-server system, where multiple clients may be accessing concurrently the same server and database.

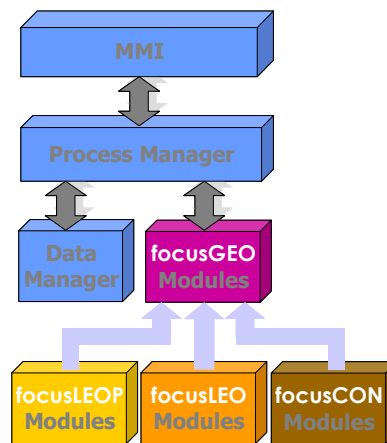


Figure 1 focusSuite Components.

focusSuite integrates many common components shared across the suite:

- GUI
- Events Logging
- Automation
- Process Management
- Data Management

A **client/server architecture** is considered as baseline for data transmission. All data and functionalities reside on a server which is accessed via a client GUI. Communications between client and server are done via TCP/IP. The possibility to work over the Internet with the appropriate degree of security has been foreseen.

Framework Concept for Satellite Operations

The **Computation Layer** is the bridge between user and data services. It responds to requests from the user (or other computation processes) in order to execute a process. This protocol insulates the user from direct interaction with the database. Two different types of processes are included in this layer:

- The Process Manager is a single process, receiving requests from the clients and starting and controlling the computation processes.
- The Computation Processes are multiple processes that can run concurrently and perform the computations needed by the system.

The **computation and data layer** are based on the extensive reuse of existing and improved software (e.g. PEPSOC, NAPEOS). The **Data Layer** maintains, accesses, and updates data. It also manages and satisfies requests to manipulate data that are initiated by computation processes. Separation of data services allows the data structure and access mechanisms to be maintained, modified, or, if necessary, even redesigned without affecting the computation or user layer.

Two different single processes running permanently can be identified in this layer:

- The **Database API Back-end** handles all the requests received from the computation processes.
- The Relational database (RDBMS).

All important mission data is stored in a database residing on the server side. In the standard version of Focus the relational database has been implemented as a set of standard ASCII files. However the interface between the database and the Computation Layer is done through a dedicated API, in such a way that the database data can be easily ported to a commercial relational database system (RDBMS) such as ORACLE.

Both the client and the server part run under UNIX and Windows. Any combination of the two operating systems is possible (UNIX/UNIX, UNIX/Windows, Windows/Windows, Windows/UNIX). Virtually all flavors of UNIX are supported, including Linux. At the client side any of Windows 95/98/2000/NT/XP are also supported. This allows complete portability of the system.

4.0 FOCUSSUITE GUI

focusSuite GUI is largely based on a proprietary toolkit for rapid GUI development called tkforms that has been developed by GMV. The tkforms toolkit is a piece of middleware that allows GUI development based on configuration files rather than on code (more details are provided in the following sections).

The most outstanding features of the focusSuite GUI are:

- **Efficiency and ergonomics:** Current Flight Dynamics GUIs do not fully exploit capabilities for easing the access of the data and facilitating the use of the system. The figure below shows a screen shot of the focusSuite GUI and highlights some of the advantages that the GUI brings.
- **Additional functionality, reliability, efficiency:** The new GUI not only brings substantial improvements in terms of efficiency and ergonomics but also provides additional functionality, including:
 - Detailed Events Logging mechanism including capabilities to visualise key results characterising the execution of the programs and therefore allowing for very efficient monitoring of station keeping planning procedure behaviour both in manual and automatic mode.
 - Capability to perform unlimited Undo and Redo operations which represents a dramatic improvement of the system in terms of reliability. Every data modification and program execution operation can be undone and redone.

- Advanced data management capabilities including the possibility to manage data sets (duplication, modification, deletion). The GUI includes the concept of private and public data sets thus reinforcing the reliability of the operations performed with the system and enabling far more efficient concurrent use of the system. Also capability for the user to perform on the fly duplication of operational data for performing trial executions before bringing data operational.
- Improved mechanisms for work groups (concurrent system access), dramatically improved locking mechanisms.
- Capability of integrating new pieces of software without the need of performing software rebuild, making maintenance a lot easier and dramatically less costly.
- **Ease of use and learning curve:** from the screen shot shown below it can be observed that the focusSuite GUI represents a dramatic improvement in terms of ease of use and learning curve of the system. The system offers two levels of grouping of tasks that can be customised for each customer, therefore being capable of adapting to different operational approaches. Moreover, the system can be customized such that at any time the user is confronted with the normal sequence of operations making the system extremely easy to use.
- Additional **graphic capabilities** to the system will dramatically improve the productivity of the operations staff and provide a user-friendlier environment. A better output would also provide better view and better understanding of the results. In summary productivity and reliability will both be considerably improved. There are three types of graphical information to be presented to the user, each of them visualized by a dedicated tkforms widget:
 - X-Y plots (for example orbital elements versus time or orbit determination measurements residuals) provide by focusGrafos component
 - Events (for example Orbital Events or a Timeline) managed by the
 - 2D/3D orbit visualization using the VisualFocus component

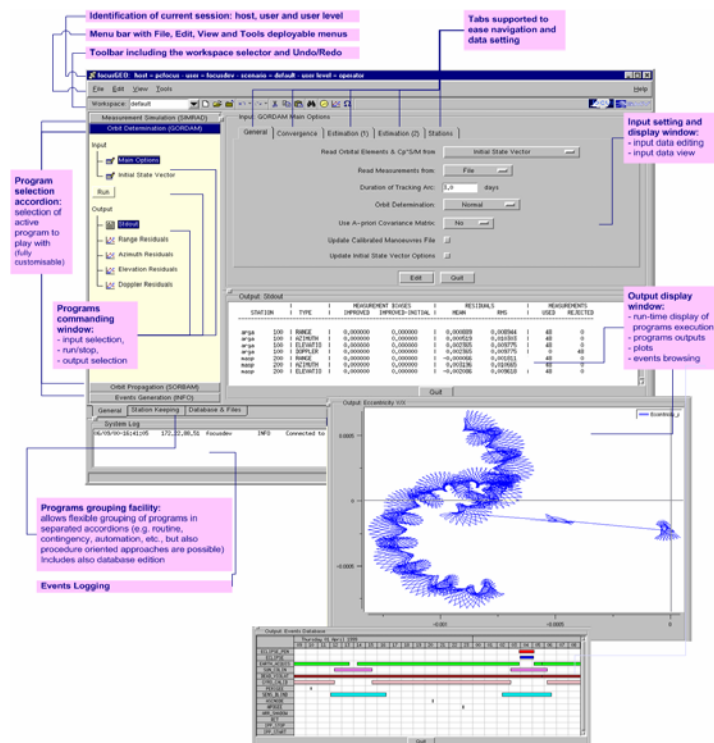


Figure 2 System GUI.

Framework Concept for Satellite Operations

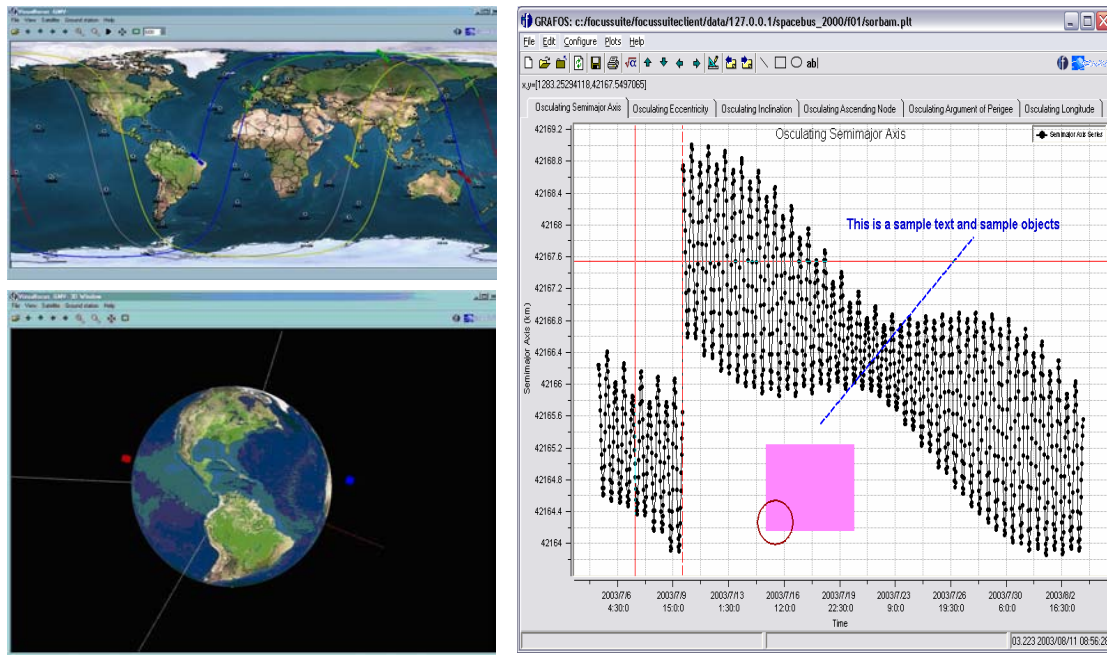


Figure 3 2D/3D Visualization capabilities (VisualFocus/ focusGrafos).

5.0 HOW INTEGRATE APPLICATIONS

On the following we are listing the requirements on any given application (program) to be integrated into the focusSuite framework:

- focusSuite data is organised in data structure composed of three levels (system, scenario and workspace). The data of the external application shall be adapted to this structure.
- The developer shall create the configuration file of the program (COF file), which defines the environment of the program, containing the definition of datasets (inputs and outputs) required to run the associated program by the process manager.
- The program shall have at least one main input dataset and one Autofocus specific dataset, the latter one being required only if we want to invoke the corresponding program from Autofocus. There is also a specific dataset naming convention to be applied.
- Both the main input dataset and the Autofocus specific dataset shall be compatible with the Tkforms component, which provides the data access services used by the data manager component. This is a crucial requirement if we want to get access to the inputs and output of a program from Autofocus.
- The developer shall create the Client GUI configuration files. These files also shall be compatible with the Tkforms components. The client GUI allows the user to get data access, to command program execution and to browser program results. Access to these functions is enabled through configuration files as described later.

Tkforms toolkit provides all data access services including GUI panel edition and data access libraries to be used from computation programs. Notice however that the computational programs do not necessarily have to use the Tkforms library to access the data, though this is highly recommended. A common data file format (TKF) and the use of this toolkit improve commonality throughout the system and allows faster

development of new components and easier integration of new components (such as new computation programs). In particular, the GUI, the DataManager and [AutofOCUS](#) use this toolkit.

The idea behind [Tkforms](#) is to split data content and data visualisation in two (or three) different files:

- Configuration file: written in very simple language and containing the details on how the data will be accessed and visualised. These details consist of the type, the layout, format and description of the parameters, etc. A typical definition of a piece is

```
integer
<
name `ITEM1'
label `An Integer:'
format `i7'
units `deg'
iulim 1000
illim -1000
help `This is an integer number'
>
```

- Data file: contains the actual data either as a sequence of pairs identifier/value or as list of values. When a program or application requires a data file with more complex structure than the standard, it is necessary to combine this data file with the wildcard file.
- Wildcard file: this is used for non-standard [Tkforms](#) data files and works as a template of the data files. This file is a copy of the data file in which it has replaced the values by @.

The types currently supported by standard [Tkforms](#) are:

- **Separator**: This item does not correspond to any data element. Only it is used by graphical modules (i.e focusPanel which generate a horizontal line).
- **Tabs**: This item does not correspond to any data element. Only it is used by graphical modules (i.e focusPanel which generate a new tab page).
- **Integer**: This item corresponds to an integer datum. It is a elementary item.
- **Real**: This item corresponds to a real datum. It is a elementary item.
- **String**: This item correspond to a string datum. It is a elementary item.
- **Epoch**: This item corresponds to a date datum. It is a elementary item.
- **Array**: This item correspond to a vector of dimension 1xN. It is a elementary item.
- **Yesno**: This item correspond to a logical datum. It is a elementary item.
- **Menu**: This item correspond to a enumerate datum. It is a elementary item.
- **List**: This item correspond to a list of records. It is a composed item, where each record is a finite number of elementary items.
- **Matrix**: This item correspond to a matrix NxM of real, integer or string data. It is composed item.

The consistency and coherence between the above set of files is crucial for the GUI and the rest of the system to work properly. [Tkforms](#) is the main mechanism that guarantees this consistency. Then additional modules translate the information stored in the [Tkforms](#) files in new components, which are integrated easily in high-level applications or computation programs.

Framework Concept for Satellite Operations

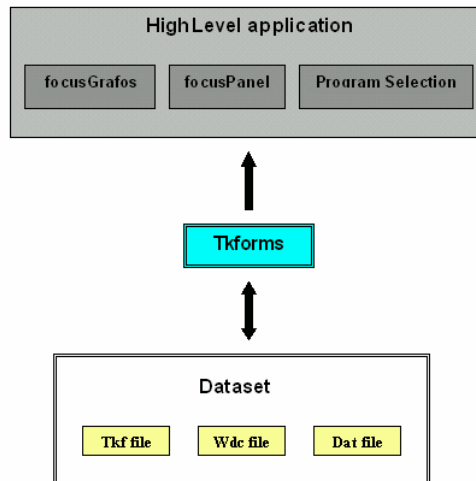


Figure 4 Tkforms concept.

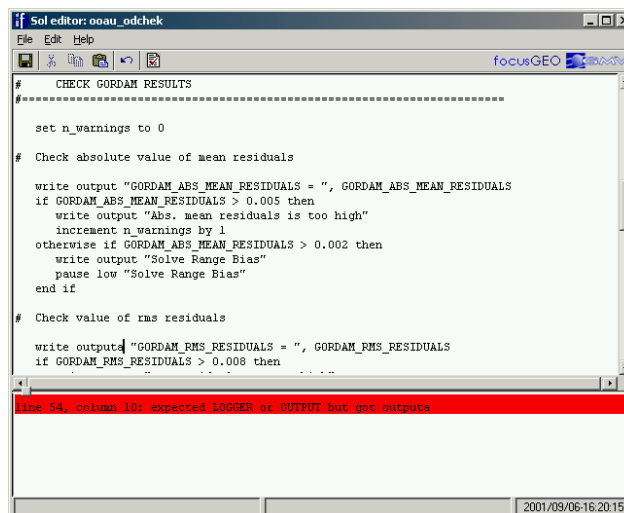
6.0 PROCEDURES AUTOMATION: AUTOFOCUS

Autofocus is an advanced component that delivers full automation support for hands-off operations to the entire **focusSuite**. The solution is based on an integrated set of tools for the definition, scheduling, execution, monitoring, and control of high-level user-defined operational procedures.

Autofocus is fully compatible with today's operations based on procedures, but assists the human operator by means of an agent that handles procedures written in SOL, the spacecraft operations language. **Autofocus** provides two user environments,

- one for procedure definition (the SOL factory),
- another for procedure execution (the procedure AGENDA).

SOL is a GMV automation language specifically designed to support spacecraft operations. Oriented towards spacecraft operators (not programmers), SOL is a very high-level language with natural language-like syntax in which the number of language elements has been minimised.



```

Sol editor: ooau_odcheck
File Edit Help
focusGEO
# CHECK GORDAM RESULTS
-----
set n_warnings to 0

# Check absolute value of mean residuals
write output "GORDAM_ABS_MEAN_RESIDUALS = ", GORDAM_ABS_MEAN_RESIDUALS
if GORDAM_ABS_MEAN_RESIDUALS > 0.005 then
  write output "Abs. mean residuals is too high"
  increment n_warnings by 1
otherwise if GORDAM_ABS_MEAN_RESIDUALS > 0.002 then
  write output "Solve Range Bias"
  pause low "Solve Range Bias"
end if

# Check value of rms residuals
write output "GORDAM_RMS_RESIDUALS = ", GORDAM_RMS_RESIDUALS
if GORDAM_RMS_RESIDUALS > 0.008 then
  ...

line 54, column 10: expected LOGGER or OUTPUT but got outputa
2001/03/06:16:20:15
  
```

Figure 5 Autofocus SOL editor.

SOL is a procedural language which features numeric, text, Boolean and date (relative and absolute) data as well as list handling.

A rich set of mathematical (trigonometric and hyperbolic functions, logarithms, power, etc.) and logical expressions (equal to, less than or equal to, logical **and**, **or** and **not**, etc.) is available in SOL. Date expressions and arithmetic operations are also supported, such as in:

```
set endEpoch to today + maneuverDuration
increment eclipseDuration by 0.5 hours
```

In the above example **today** gets substituted by today date.

SOL features the ability to call **procedures** from within other procedures. Procedure call, via the **execute** construct, allows to explicitly enumerate the inputs and outputs of a procedure for improved legibility:

```
execute orbitDetermination
  & inputs are
  &   set satellite to "H02"
  &   set epochStart to 22-jul-1999
  & outputs are
  &   set newStatus to status
  &   set newOrbit to orbit
```

SOL control structures include branching via **if/otherwise/end if** and flexible looping: repeat a block of instructions a fixed number of times (**repeat for/end repeat**), repeat while a condition is true (**repeat while/end repeat**) and iterate over the elements of a list (**repeat for each/end repeat**). For example:

```
repeat while residuals > 1.0e-3
  # Perform computation here
end repeat
```

SOL supports the whole focusSuite for full focusSuite framework automation. It is possible, for example, to activate external focusSuite components from a SOL procedure, collect its output and process it afterwards.

The Autofocus integrated procedure definition environment allows the user to write, validate and test procedures.

- Writing SOL procedures: The Autofocus SOL editor includes all features common in text editors (search, search and replace, copy, paste, ...) as well as providing SOL syntax coloring and contextual help and examples for all SOL syntax elements.
- Validating SOL procedures: SOL has been designed so as to allow extensive static procedure validation, both at the syntactic (keywords incorrectly spelled, incorrect expressions, a missing **end repeat** construct, ...) and semantic (executing a procedure with an incorrect set of inputs, trying to add two absolute dates, ...) levels. The validation feature ensures that procedures scheduled for execution are free of all but run-time execution errors (such as a division by zero).

Errors found when validating SOL procedures are displayed in a dedicated window: click on an error and the editor automatically jumps to the offending line. Contextual help for all errors is available.

- Testing SOL procedures: The Autofocus integrated procedure definition environment allows to test validated SOL procedures in a safe way so as not to modify operational data.

SOL procedures can be run in debug mode, providing controlled execution (line by line, jump to a given line or breakpoint, etc.) and introspection capabilities (showing the list of defined procedures with their inputs and outputs, the list of defined variables and their contents, etc.). A

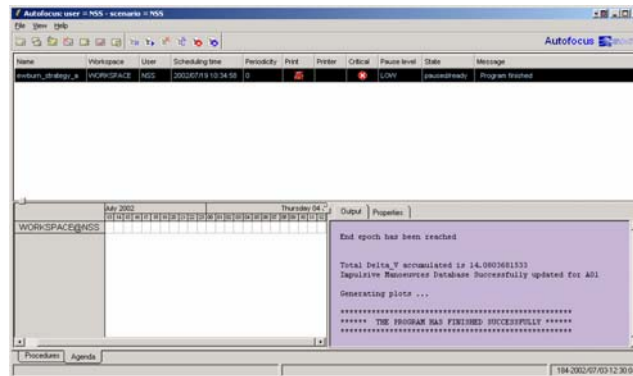


Figure 7 Autofocus procedures agenda.

7.0 OPERATIONAL EXPERIENCE: FOCUSGEO

focusSuite has been progressing in a phased approach and the several operational members of this family have been developed, in particular focusGEO product has been conceived in order to provide a new generation Flight Dynamics product for geostationary satellites.

This focusGEO version is now available and operational at HISPASAT and EUTELSAT. This tool is used as part of the end-to-end activities from Flight Dynamics point of view. In particular the following operations are executed using focusGEO:

- Routine and contingency operations over more than 20 satellites under control by EUTELSAT and over 4 satellites under control by HISPASAT. Routine operations include at least the following capabilities:
 - orbit determination and manoeuvres estimation,
 - orbit propagation,
 - manoeuvres computation,
 - manoeuvres calibration,
 - manoeuvres implementation and TC computation,
 - events computation,
 - reporting,
 - TM retrieval.
- Collocation routine operations, including combined ionic and chemical propulsion.
- Collocation cluster initialization.
- De-orbiting.
- Longitude relocation.
- Inclined orbit control (free inclination drift).
- Ionic propulsion management.
- Mass evolution and lifetime prediction.

focusGEO integrates the following elements as part the common focusSuite infrastructure:

- Mission independent SW inherited originally from PEPsoc, but highly modified by GMV internal developments and improvements of the original control algorithms, in particular

Framework Concept for Satellite Operations

optimization of the orbit control under critical constraints (e.g. reduced control window, high cross-coupling, etc.) Multi-satellite platform system: Astrium's Eurostar 2000/2000+/3000

- Alcatel's Spacebus 3000/3000B
- NPO/PM's SESAT (ionic propulsion)
- Alenia's SATELCOM
- HP376 Boeing (spinning satellite)
- FS1300 SS/Loral
- Collocation Station Keeping and Inclined orbit (in collocation also) Multiple reference frames
- Ion thrusting
- Reporting capabilities
- Contingencies and AOCS support
- LEOP support (launch window, transfer optimization)

From computational point of view we can highlight the following strategies:

- Orbit determination is based on a weighted least squares method able to estimate orbit, station biases, CpSM, delays, maneuvers.....using ranging, angular, doppler, turn-around, interferometry data.
- Numerical orbit propagator is based on a Runge Kutta, taking into account impulsive and continuous maneuvers.
- Maneuver computation is based on minimum fuel (several options depending on the cycle length,...) inclination control.
- Maneuver computation is based on sun pointing perigee strategy for eccentricity and longitude control. Additional eccentricity/longitude control strategies implemented for severe satellite constraints (reduce window, high cross-coupling).

REFERENCES

- [1] GMV S.A: "Introduction to focusSuite" v2.5. May 13th 2004.
- [2] GMV S.A.: "focusGEO User's Manual".
- [3] EUTELSAT FD maintenance handbook and operational procedures.
- [4] HISPASAT FD maintenance handbook and operational procedures.

BIOGRAPHY

Miguel Angel Molina is a Senior Aeronautical Engineer currently responsible of the Flight Engineering Business Unit at GMV S.A. This Business Unit is mostly dedicated to Flight Dynamics and Mission Analysis activities.