

# An Approach for Automation the Satellite of Routine's Operation and Procedures<sup>1</sup>

Marinalva Soares, Maurício Ferreira, Adenilson Tomé, Francisco Júnior, Jaqueline Clivelaro, Vinícius Oliveira and Wagner Silva

*Satellite Control Center, National Institute for Space Research, Sao Jose dos Campos, Sao Paulo, Brazil*

The routine operation phase of a satellite occurs after its launch and initializing settings of the satellite in orbit. The basic operational functions are routine operations that include telemetry monitoring, sending remote command and the implementation of tracking measurements, such as range and range rate, and execution of procedures, which consists of actions that are taken when the execution of a task. Currently, in the area of satellite control, there is a general interest in automating as a way of reducing in-orbit satellite maintenance costs, specially the high cost with teams operations. Besides the cost reduction, the automation can minimize errors that may be introduced by repetitive tasks. The growing number and diversity of satellites launched often leads to the increasing and complexity of the operations. This has a negative impact on safety, efficiency and operations costs. With automation, the same operators' team can control more satellites and they can use their time on other activities; human errors can be reduced by repetitive and fast execution of planned operations; a large number of verifications can be made in a short time. With the prospective of launching new satellites and reducing of financial resources, the National Institute for Space Research (INPE) has concerns about solutions that enable the operations of multiple missions keeping the same operation team. In this sense, INPE has developed automated solutions for the routine operations involving the control and tracking of satellites. The automation is evolving toward to controlling both the current missions as well the future missions. Thus, it is necessary to build a flexible architecture that easily adapt to new missions with minimal changes in the programming code. In this context, this work describes the architecture and its requirements for automating the routine tasks for different missions. The requirements were defined based on two main purposes: a) automation of routine tasks: antenna calibration, range and range rate measurements, send command, telemetry monitoring, backup, file transfer and creation of unforeseen activities; b) procedures execution: procedures can be defined as a set of activities (or actions) that are executed in sequence or a set of actions which are taken (based in conditions) when a fault occurs during the activity execution. The major prospects with automation in development at INPE are: to enable the efficient handling of a series of similar satellites; to facilitate the coding and maintenance of procedures; to favor the experience and knowledge in the field of automation technology; to ensure safe operations by reducing the human risk factor and to reduce costs with operators.

## I. Introduction

With the growing number and diversity of satellites launched by the space agencies, the automation of the routine operations for satellite control have been the solution to save costs with human operators and to increase their productivity keeping the same operators team. The basic operational functions are routine operations that include antenna calibration, telemetry monitoring, sending remote command and the implementation of tracking measurements, such as range and range rate, and execution of procedures, which consists of actions that are taken when the execution of a task. These operations are repetitive tasks and the operators have good understanding about them, which makes them candidates for automation. The operations are executed in different time periods, for example: connection with ground station and antenna calibration are executed few minutes before satellite visibility, telemetry monitoring is executed during the satellite visibility and disconnection with ground station is after the end of the tracking.

However, future missions can present particular activities and procedures. Thus, to be an effective benefit, the automation should be generic enough to meet both the current missions as well as future missions, and this is one of the challenges of automation. In this context, INPE is developing a solution based on a multilayer architecture whose focus is to minimize the changes in the programming code to meet new missions. Automation is an issue that

---

<sup>1</sup> Sponsored by FAPESP

has been investigated and applied in missions of important space agencies like the National Aeronautics and Space Administration (NASA) and European Space Agency (ESA).

The operations of satellite control are organized in the Flight Operation Plan (FOP), which is generated weekly for each satellite. At INPE, the control activities are planned, coordinated and executed from the Satellite Control and Tracking Center (CRC). The CRC, during periods of satellite visibility, connects to the ground station by a network able to receive telemetry and send telecommand to the visible satellite in real time by using a Satellite Control System (SATCS) developed by INPE. SATCS is a real-time software application and requires direct interaction with the satellites operators during successive visibility periods. This interaction becomes infeasible as the number of satellites increases. Details about the INPE satellite operation and how the activities are organized in the FOP can be seen in [5].

Researches related to the automatic generation of a mission plan operation and automation of routine operations can be found in [1] [2]. But it is important to note that the automation described in this paper is not related to the generation of the FOP, but the automatic execution of the operations contained in the plan. Commercial solutions such as FlexPlan [3], developed by GMV Space Systems, and STK Scheduler [4], developed by Orbit Logic, have been used by some space agencies. For example, ESA, NASA and EUMETSAT (European Organization for the Exploitation of Meteorological Satellites) have used FlexPlain in some of their missions.

However, due to the peculiarities of each mission and the difficulty of integrating of those solutions with the satellite control system already in use by the organization, the adoption of a commercial solution is not always feasible. Thus, it is important that the space organizations themselves have means to develop solutions to meet their needs more precisely. INPE is planning to launch a lot of satellites in the future. In this context, INPE is going toward to the domain of knowledge in the field of automation technology beginning with the developing a architecture to automate routine operations and procedures for current and future missions. The approach adopted by INPE in the development of the architecture is described in next section.

## II. REQUIREMENTS FOR A FLEXIBLE ARCHITECTURE TO AUTOMATE THE SATELLITE CONTROL AT INPE

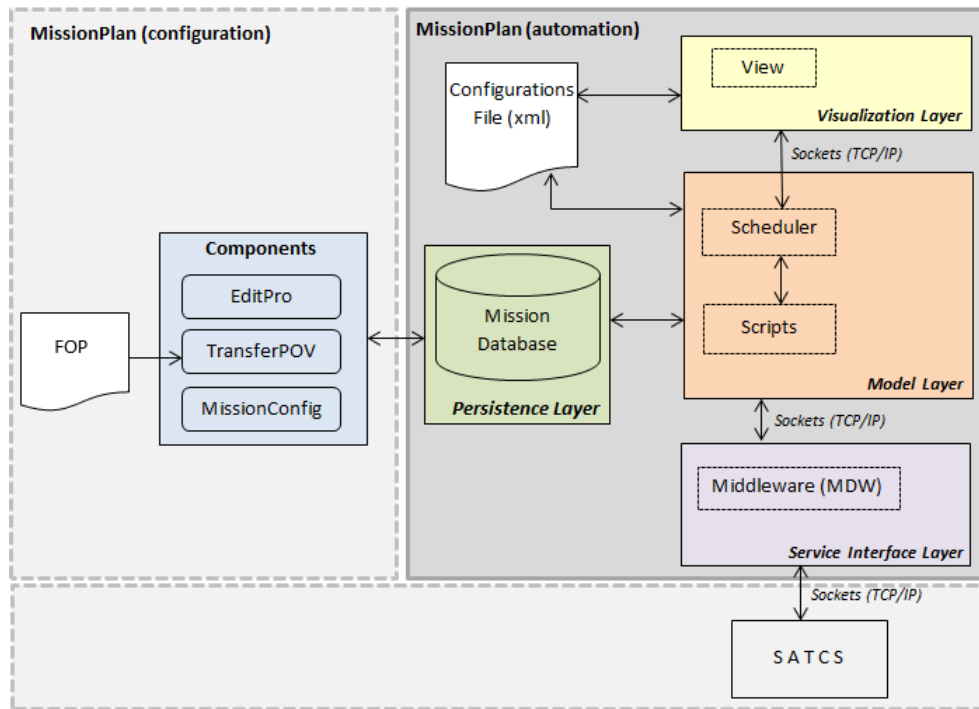
In order to automate routine tasks and procedures execution for both the current missions as well the future missions, it was necessary to create an automation architecture with the following requirements: a) a data source easily accessible, multi-satellite and multi-mission; b) an environment to write, to compile and to configure procedures for execution; c) a scheduler for automatic loading and execution of activities and procedures of the passage.

The architecture created, named MissionPlan, was divided in two parts: Configuration and Automation (as illustrated in Fig. 1). The MissionPlan Configuration is composed by independent programs responsible for providing the mission data and their configurations for execution. These programs are independent of each other and do not make exchange of data among themselves. On the other hand, the MissionPlan Automation has three main layers that communicate among themselves by exchange message using the TCP/IP (Transmission Control Protocol/Internet Protocol) protocol. These layers are: persistence, view (presentation), domain model and communication. The persistence layer is a database Mysql that stores in a structured way the activities and procedures configuration for execution, which access interface is SQL command; the domain model implements the automatic control of the activities before, during and after the passage through the scheduler module; the view layer manages the display of information; the communication layer makes the data interchange between SATCS and domain model.

Although the automation described here does not include the FOP generation, one important requirement is to make the architecture independent of the FOP current format (which is the main data source to control the satellite). This requirement facilitates the adaption of the architecture to future missions, because the basic access interface to the data does not change. This independence was reached by storing of the FOP in a structured database that is multi mission and multi satellite. Furthermore, one important characteristic of the database is the storing of the configurations for procedures execution during the passage of the satellite, which are scripts written in Python. The procedures are also important data for the control satellite. In most cases, these procedures are actions that are executed when a remote command fails. The information about the procedures that are stored in the database are script identifier (the procedures are written as a script), path, parameters, execution period, activities to the which the script is triggered, conditions for execution and so on.

As the FOP is in a format text, it was necessary to develop programs to transfer it to the database and to manipulate the database. These programs are in the MissionPlan Configuration: TransferPOV, an application that reads the FOP from a text file and write it to the database; MissionConfig, an application with graphical interface

created to manipulate the database; EditPro, an application that integrates the idle Python with the mission database to write procedures, which the second requirement of the automation architecture.



**Figure 1 – Multilayer architecture for automation**

### A. Mission Database

One of the main requirements for the development of a flexible architecture is the independence of the data source. The current format of the FOP is not appropriate to incorporate changes in the case of the new missions. In this sense, it was created a database multi mission and multi satellite, which access is by SQL commands. The database was structured to store the FOP (the TransferPOV reads the FOP and stores it in the database), the configurations and restrictions for execution of the activities and procedures of each passage. Each activity in the database has an associated a script written in Python, which contains command to execute this particular activity. These script is called automatically by scheduler when its identifier is loaded from database. The database stores activities known and, activities not foreseen in future missions, can be written as a procedure, stored and configured for execution in a passage in the database. A procedure is stored in database such as an activity and also has a script that execute it. The information about the script (such as identification, path, description and a flag to execute or not execute the script for that passage) are stored in database to enable the schedule trigger them automatically when load the activities of the passage. An activity can be configured to execute for a period including several passage of a same satellite. These configurations and restrictions data enable that new activities and procedures can be scheduled for execution in specific passages without changes in model layer.

Fig. 2 and Fig. 3 show a snapshot to create a new activity and configure it for execution in the database, respectively.

The screenshot shows a software window titled "ACTIVITIES". At the top, there are two tabs: "CREATE" (which is active) and "EDIT/SEARCH". Below the tabs, there are several input fields and controls:

- Name:** A text input field.
- Type:** A dropdown menu.
- Routine:** Two radio buttons labeled "YES" (selected) and "NO".
- Execution:** Two radio buttons labeled "A" (selected) and "M".
- Subsystem:** A dropdown menu.
- Description:** A large text area.
- Script:** A dropdown menu.
- Automation script:** A dropdown menu.
- Responsible:** A text input field.

At the bottom of the window, there are two buttons: "SAVE" and "CLOSE".

**Figure 2 – Activity creation**

In Fig. 2 can be seen that when creating an activity, must be associated to it a script two scripts: the label “Script” is the script which contains command to execute the activity. The label “Automation script” is when the activity has a procedure to be executed by certain conditions when the activity fails.

The screenshot shows a software window titled "CONFIGURATION OF ACTIVITY EXECUTION". At the top, there are two tabs: "CONFIGURATION" (which is active) and "EDIT CONFIGURATION". Below the tabs, there are several input fields and controls:

- Activity:** A dropdown menu.
- Satellite:** A dropdown menu.
- Ground Station:** A dropdown menu.
- Period:** Two date pickers (represented as // //) with a "to" label between them.
- Time:** Two time pickers (represented as : : ) with a "to" label between them.

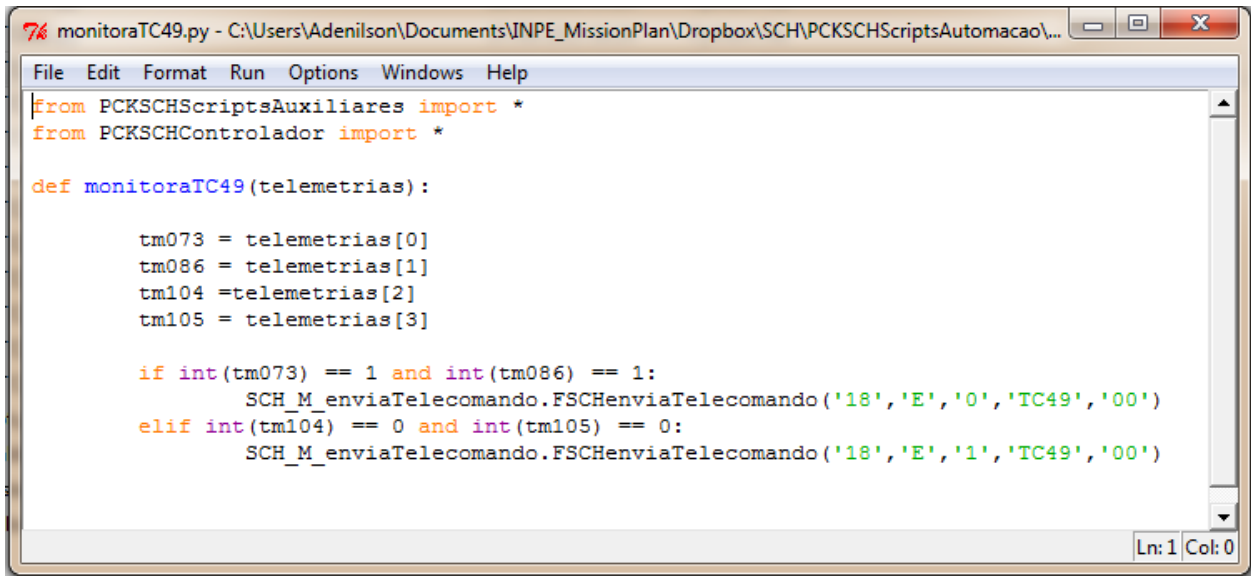
At the bottom of the window, there are two buttons: "SAVE" and "CLOSE".

**Figure 3 – Execution configuration of activity**

When configuring the execution of an activity, should be informed the satellite, ground station and its period of execution (initial and final date, initial and final hour).

## B. EditPro – an environment for procedures edition in Python integrated with the mission database

One of the requirements of the adopted architecture is the development of an environment to write and to compile procedures. This requirement was implemented as a program in the MissionPlan Configuration. Fig. 4 shows an example of a procedure created for execution when the telecommand execution fails. This procedure is written in a Python language using a traditional idle Python, an integrated development environment for Python.

A screenshot of a Python IDE window titled '76 monitoraTC49.py'. The window contains the following Python code:

```
from PCKSCHScriptsAuxiliares import *
from PCKSCHControlador import *

def monitoraTC49(telemetrias):

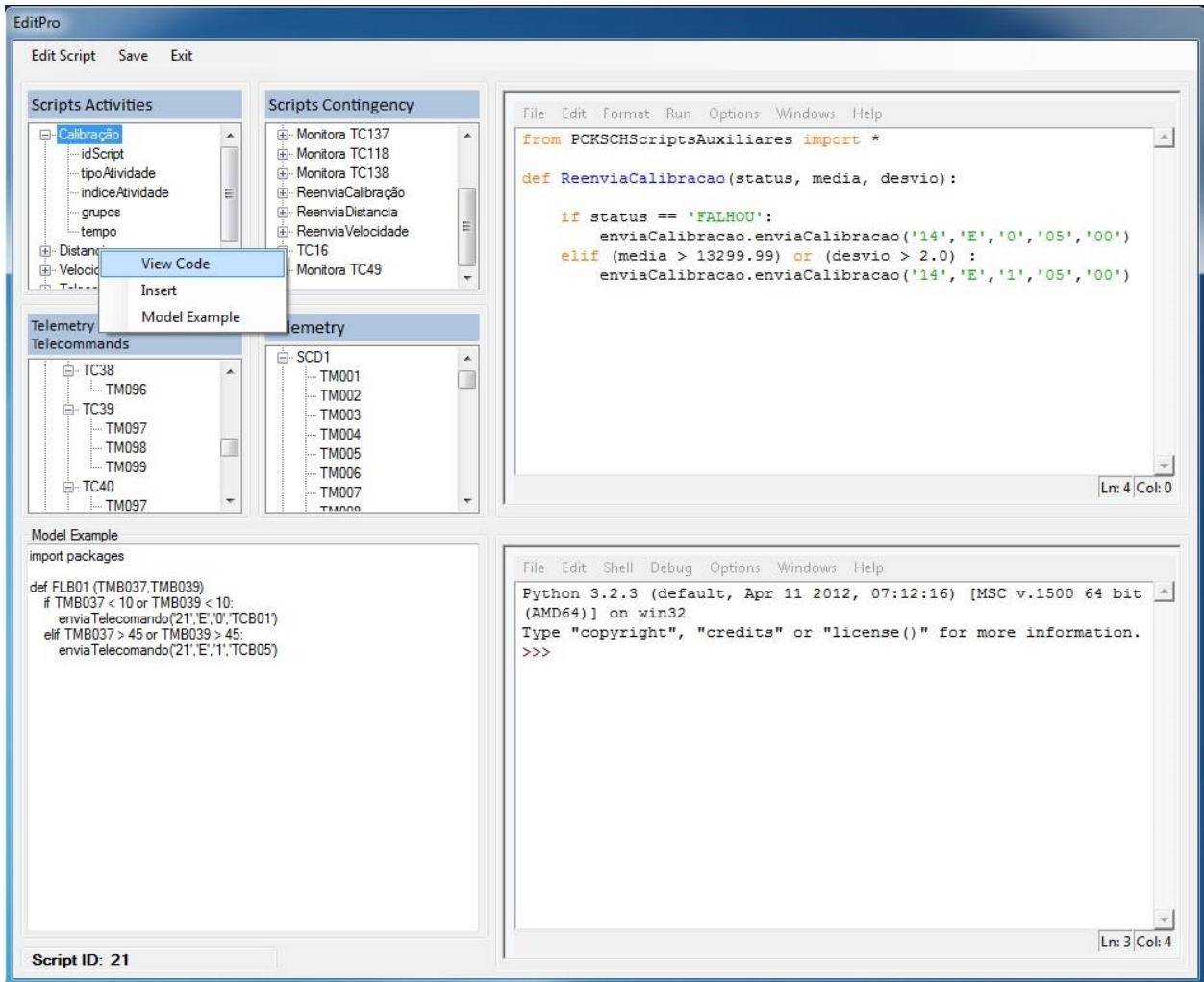
    tm073 = telemetrias[0]
    tm086 = telemetrias[1]
    tm104 =telemetrias[2]
    tm105 = telemetrias[3]

    if int(tm073) == 1 and int(tm086) == 1:
        SCH_M_enviaTelecomando.FSCHenviaTelecomando('18','E','0','TC49','00')
    elif int(tm104) == 0 and int(tm105) == 0:
        SCH_M_enviaTelecomando.FSCHenviaTelecomando('18','E','1','TC49','00')
```

The status bar at the bottom right shows 'Ln: 1 Col: 0'.

**Figure 4 – Example of a procedure written in a traditional idle Python**

However, when writing a procedure, the user must know the telecommand and its telemetries and the scripts developed for activities execution for use in the procedures. Thus, it was necessary to create an environment that integrates the idle Python with the mission database. This environment was named EditPro and is illustrated in Fig. 5.



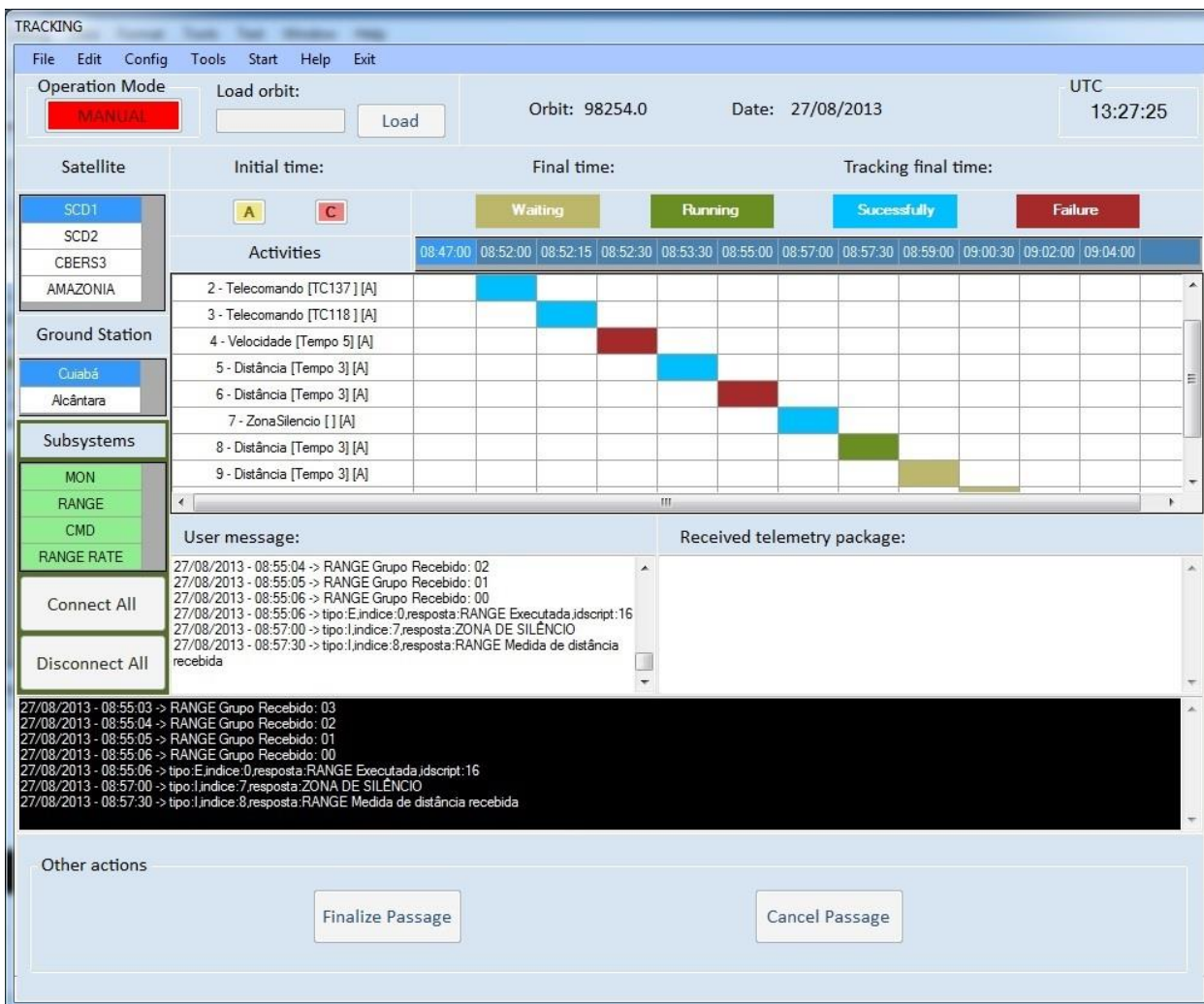
**Figure 5 – EditPro – idle Python integrated with the mission database**

On the left side of EditPro, the user can see the scripts that can be used to write procedures and call them in the code. The code of each script can be visualized by hovering the mouse over the script. The list of telecommands and their telemetries are also provided to the user. In this environment the procedure can be only compiled, but not tested or executed. The script can be executed only by the scheduler when loading the passage for which the script has been configured to execution. The script must be registered in the database and configured for execution by the MissionConfig program or directly in the database.

### **C. Scheduler - Module of passage loading and automatic control**

The domain model layer implements the scheduler that trigger the activities and procedures to be executed before, during or after the tracking. The scheduler controls the time of trigger each activity and invokes the script responsible to send the activity to the SATCS. SATCS has a subsystem for each type of activity. The main datasource from scheduler is the database. Thus, all the activities and procedures that were included and configured for execution in a certain passage are loaded by the scheduler in model layer. When the scheduler load the passage, it identify all the subsystems of the SATCS that are required to perform the activities scheduled for that passage. The time of connection with the ground station is calculated and executed automatically by the scheduler. Procedures of reconnection may be executed if the passage it was configured for this. The monitoring of the execution of activities can be made in the graphical interface that was implemented in view layer by the following status (Fig. 6): green:

executing; red: failure; brown: waiting; blue: success. The status of the activities is updated in real time during the passage.



**Figure 6 – Visualization interface of a passage**

The scheduler has two operation modes: automatic and manual. The operation mode of the scheduler can be changed by operator through the graphical interface (the view). When the operation is in the automatic mode, no interaction is allowed, except for the failed activities. When one activity fails, the operator can trigger it manually and change its parameters. To interact with other activities, the operator can change the operation mode by the view module for manual at any time. Any change in the parameters of the activities should be made in the manual mode. If the operator change the operation mode to automatic again, the changes performed may or not be repassed to the scheduler, this is criteria of the operator.

When the passage has any script to be executed, it is loaded and shown to the user to monitor the execution (Fig. 7). The status of execution is the same described above for the activities. All the scripts to be executed in a passage are read when the scheduler load the passage. These scripts are showed in another window. However, no interaction with the execution of the scripts is allowed yet. And the interaction with the scrip during its execution is a important key, because unforeseen events may occur and may be necessary to stop the execution of the script and continue later.

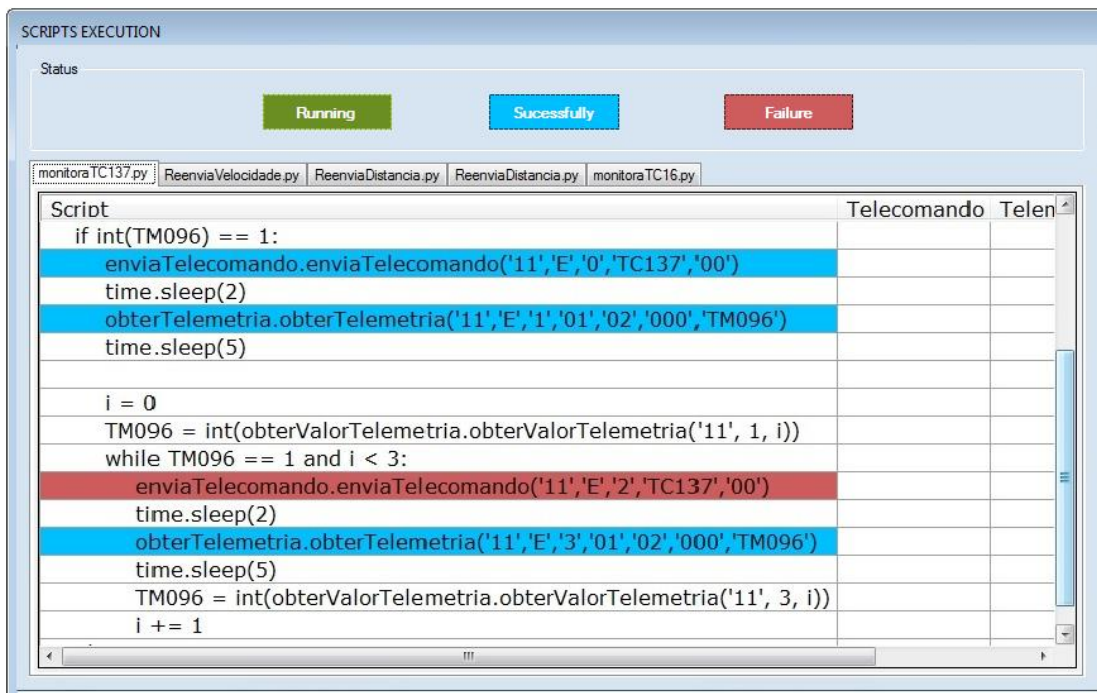


Figure 7 – A procedure (written as a script) in execution

The same script can be visualized by graphs of execution, as illustrated in Fig. 8.

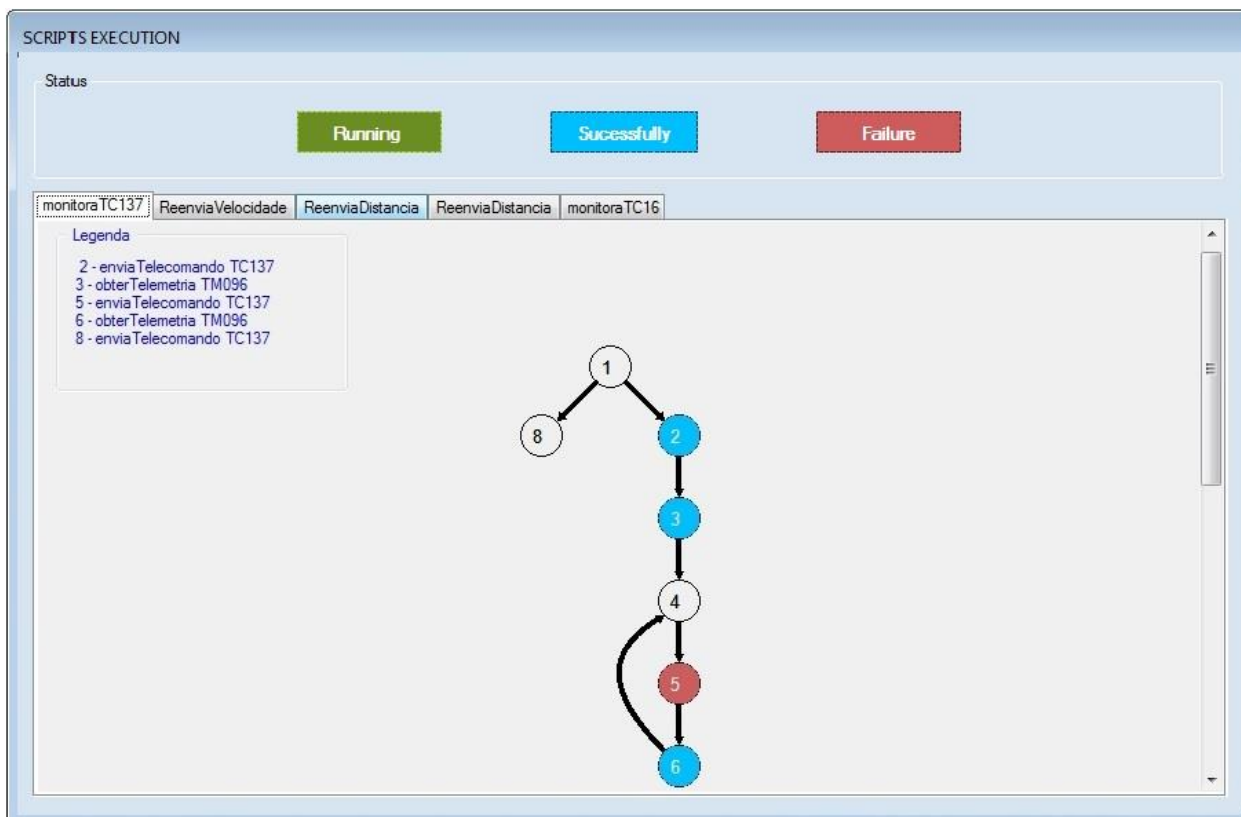


Figure 8 – Visualization of the procedure execution by graph



### III. CONCLUSIONS

With the automation, INPE can operate more satellites with the same staff. Thereby, human errors by repetitive tasks can be decreased and the staff can devote more time in more important and complex tasks. The time of make decision when a fail occurs can be decreased with the automation of the procedures. More complex procedures require the presence of an expert, but those more simpler and known are strong candidates to automation. The approach adopted to automate the routine's operations and procedures of the current missions controlled by INPE as well of the future missions was to develop an architecture independent of the data source creating a multi mission and multi satellite database. Thus, if the FOP generation and its final format change, reading by the MissionPlan Automation does not change. One of main characteristics of database is to allow to configure the scripts execution per satellite and ground station. Moreover, the execution of new activities by writting procedures that can be configured in database makes the scheduler more easily adaptable to the specific characteristics of new missions. The script that executes the procedure must be included in the same package of the scheduler. Furthermore, one script can call other scripts recursively. The operator can monitor the script execution, but does not interact. As future work, INPE is planning to automate the FOP generation and also to allow interact with the script.

### References

- [1] C. R. Haddow, K. Adamson, B. Sousa and G. Whitehead. Mission Planning - Establishing a common concept for ESOC's missions. In: Proceedings of SPACEOPS 2010 Conference, Huntsville, Alabama, 2010.
- [2] Michael Koller, Vemund Reggestad and Kate Adamson. ESOC Earth Observation Missions and the Automation of Operational Routine Tasks. In: Proceedings of SPACEOPS 2010 Conference, Huntsville, Alabama, 2010.
- [3] L. G. Gutierrez, J.A. Tejo, A. Cendrero, M. Pereda, J. Saiz, C. Hernandez, D. de Miguel, T. W. Beech. Off-the-shelf mission planning is possible: FlexPlan does it. In: Proceedings of Aerospace Conference, 2005 IEEE, pp. 4008 - 4016.
- [4] STK Scheduler. Available at <http://www.orbitlogic.com/products/stkscheduler.php>. Visited in April, 16 2013.
- [5] M. D. Soares, M. G. V. Ferreira, A. J. A. Tomé, F. E. Júnior, J. Clivelaro, V. C. Oliveira, W. R. A. Silva, A Multilayer Architecture for Automation of the Satellite Control Routine Operations. In: IEEE International Conference on Computer Science and Automation Engineering, 2013, Guangzhou, China.