Limitless Automation: ANY CONOPS

J. Noguero[1], E. Fraga[2], Luis Blanco[3], and Andres Pizarro[4]
GMV, Tres Cantos, Madrid, 28760

Having the most powerful operations automation solution in the market is not enough. Concepts of Operations (CONOPS) and automation needs appear in radically different shapes and flavors across organizations. Accordingly, fulfilling these concepts of operations is an overwhelming challenge. This paper describes specific techniques and solutions that allow fulfilling the widest variety of operations concepts. The core automation need for spacecraft operations is the execution of flight procedures delivered by the spacecraft manufacturer. This is often achieved through domain-specific procedure languages. On the other hand, some operations teams are ready to embrace the power of new scripting technologies for flight procedure execution. This is being increasingly considered by the space industry. Scripting does not necessarily rule out the deployment of domain-specific procedure languages. Both approaches can be combined by taking elements from a scripting language in order to create a domain-specific procedure language. Modern scripting languages can be extended and adapted to achieve this. Finally, some operations teams require higher-level abstractions beyond what can be achieved through procedure languages and scripting, for example, by representing procedures in ad-hoc views (e.g. flowcharts). Such requirement can also be combined to consistently support the triplet procedure/script/view by deploying core support for procedure/script execution and then deploying ad-hoc views as add-on user interface presentation layers through plug-in technologies such as Eclipse/RCP. These capabilities allow deploying homogeneous cross-sectional automation (e.g. from Assembly Integration and Verification, up to Operations Preparation and Mission Execution) whilst fulfilling any CONOPS.

Nomenclature

CONOPS	=	Concept of Operations
RCP	=	Rich Client Platform
SPEL	=	Spacecraft Procedure Execution Language
TC	=	Telecommands
TM	=	Telemetry

I. Introduction

As part of GMV's activities in the deployment of operational ground systems over the last 25 years, we have placed a strong emphasis in ground systems automation, addressing a wide variety of ground systems and associated automation solutions.

In recent years, our research and development efforts have led to the deployment of breakthrough solutions in this area. Such solutions are nowadays flight-proven as part of GMV's operational deployments worldwide, and have allowed to fulfill a wide range of radically different concepts of operations (CONOPS).

The most essential element in these achievements is the fact that all these different concepts of operations have been possible by evolving and improving GMV's solution for satellite control, hifly®, together with its native procedure automation component: autofly.

Beyond this relevant achievement, hifly® has further allowed operational deployments where different automation paradigms may even coexist (e.g. procedure/script/view).

[1] Ground Systems Consultant, Satellite and Mission Control, Isaac Newton 11, 28760 Madrid (Spain).
[2] Business Unit Director, Satellite and Mission Control, Isaac Newton 11, 28760 Madrid (Spain).
[3] Ground Systems Engineer, Satellite and Mission Control, Isaac Newton 11, 28760 Madrid (Spain).
[4] Ground Systems Engineer, Satellite and Mission Control, Isaac Newton 11, 28760 Madrid (Spain).

In the following lines, we describe available elements allowing to fulfill any concept of operations taking benefit from these operational solutions.

II. Shaping the operations automation concept

Operations automation is nowadays at the heart of today's concepts of operations. The choices to be made at the early stages in the shaping of operations automation will have major implications in the cost-effectives of the solutions that will eventually be deployed.

Over the years, GMV has experienced that there are very different possible approaches to satellite operations and ground systems automation. Even within the same organization, operations responsibles will sometimes have very different opinions on how things should be. Keeping an open mind and being able to challenge our own (often strong) views, is a valuable attitude towards seeking the best solution.

Today's automation solutions, provide a very wide range of possibilities, a key element in making the right choice is to get the solutions provider involved early on in order for all relevant stakeholders to be aware of available solutions and the associated implications. Bringing it all together allows to achieve cost-effective scalable solutions, either in scenarios where operations are migrated from an existing system (e.g. legacy flight procedures may need to be supported or migrated), or in scenarios where the operations team faces an entirely new deployment.

III. Core services

The first element enabling to fulfill different automation concepts is the set of core services available in the target system (e.g. the spacecraft control system). These services are used by the automation component to automate the target system. Therefore, the automation system can only be powerful if enabled by the target system and its available services.

More powerful services allow more comprehensive automation. Powerful core services can be made available under the following circumstances:

1. The target system is powerful and rich in functionality (*core functions*).
2. The target system has flexible means to expose such functionality through a proper *service interface*.

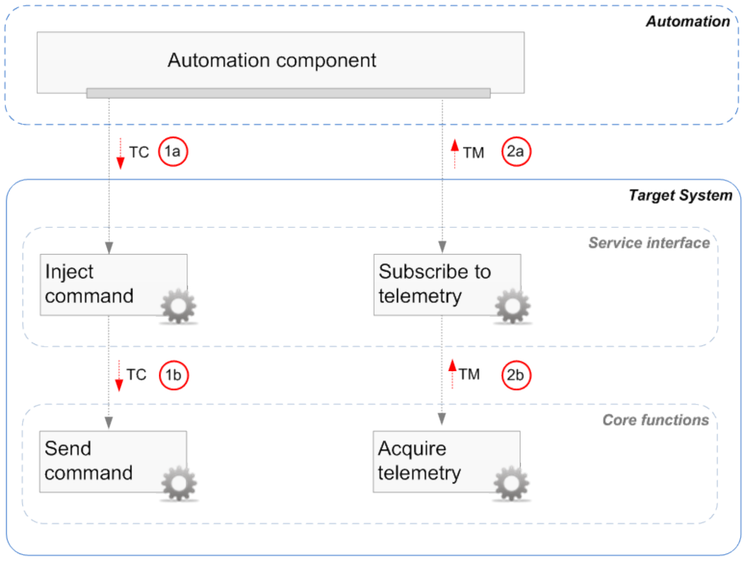

Figure 1. Core services for automation. *Automation is able to automate a target system by making use of the target system capabilities which are made available through a service interface.*

In addition to these basic requirements, different concepts of operations will further impose demanding requirements onto the *core services* in many areas and often specifically in terms of:

1. Availability (e.g. ability to transparently and automatically recover the service after connection outage).
2. Performance (e.g. high data throughput in data provision).

A very good example of how different concepts of operations impose very different requirements on the target system is the available hifly® services for procedure automation in the area of out of limits management (for telemetry parameters monitoring). In this particular case, we have the following services which serve different purposes in different deployments:

1. Modification of telemetry parameter out of limits definitions.- limits defined in the spacecraft database are modified (e.g. relaxed) and later restored during flight procedure execution consistently with the operations in-progress.

2. Session level masking of telemetry parameter out of limits definitions.- limits are masked and later restored during flight procedure execution consistently with the operations in-progress. The limits are not modified but masked at procedure level so they are still visible within the core processing chains.
3. Modification of selected limit sets to be applied.- allows to keep original definitions unmodified in the spacecraft database by dynamically setting a different set of predefined limits.
4. Live notification of out of limits violations.- allows automation procedures to manage unexpected limit violations (e.g. monitoring out-of-limits, pausing execution, etc).

Finally, since hifly®'s core is specifically designed to maximize common/generic support, this principle is also applied to its core services, that is, core services are generic for all heterogeneous spacecraft buses being supported. This allows homogeneous multi-mission operations in the context of heterogeneous spacecraft fleets.

IV. Building up powerful automation capabilities within the scripting layer

Once services are available, they need to be captured as native capabilities (e.g. *native functions*) within the automation component. Typical functions to automate flight procedures include: acquisition of spacecraft telemetry and ground telemetry, injection of spacecraft commands and ground commands, modification of ground system configuration (e.g. out of limits definitions), etc.

In autofly, these *native functions* are then wrapped as native python functions (www.python.org) within autofly's python-based scripting automation layer. These native functions follow the SPEL open source specification developed by SES and GMV[1] which establishes a clear reference for standardizing python-based flight procedures.

The autofly scripting layer is an important element underpinning the overall automation concept. Its main asset is the fact that it is a superset of other existing flight procedure languages and therefore allows supporting the associated operations concepts. This has been possible as follows:

1. The target system (hifly®) is rich in *core functions*
2. A comprehensive set of automation *native functions* has been deployed. This includes:
   a. The encapsulation of the target system core functions (e.g. Send command).
   b. The deployment of additional native automation functions (e.g. user prompting and interaction, time management, etc).
3. A set of *extended python functions* deployed at the python scripting layer (e.g. combining the *Send* and *Verify* within a single *Send and verify* python function).

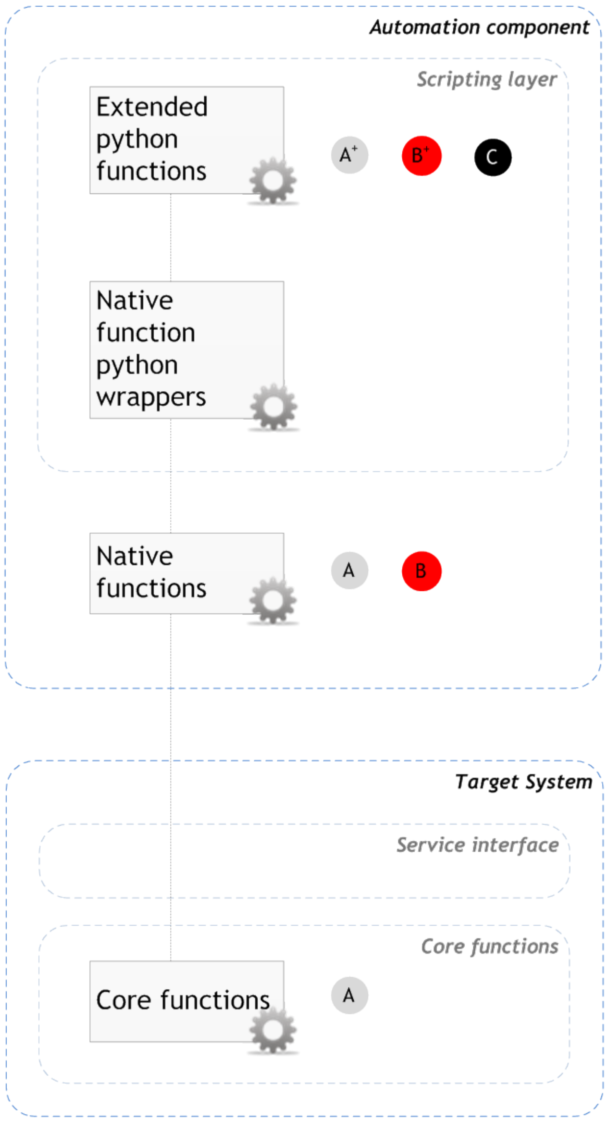

Figure 2. Empowering the scripting layer. *The scripting layer includes target system capabilities (A), native functions (B), and extended python functions based on A and B (A$^+$ and B$^+$). Furthermore, high-end python capabilities are available (C) by combining all available functions with the power of python scripting.*

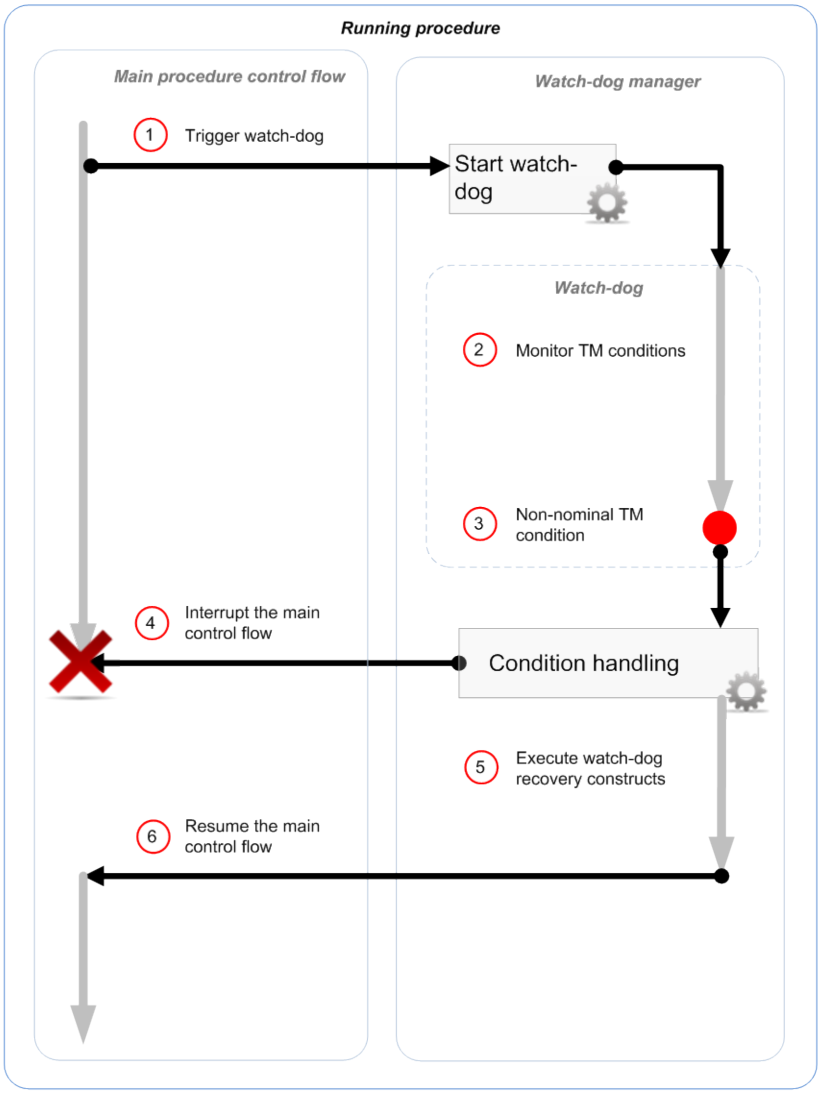
Running procedure
Main procedure control flow
Watch-dog manager
1 Trigger watch-dog
Start watch-dog
Watch-dog
2 Monitor TM conditions
3 Non-nominal TM condition
4 Interrupt the main control flow
Condition handling
5 Execute watch-dog recovery constructs
6 Resume the main control flow

Figure 3. Watch-dogs. *Procedures may include watch-dogs that are triggered (1) in order to monitor operational consitions ni parallel to the procedure main control flow (2). When the watch-dog detects a non-nominal condition (3), autofly is able to interrupt the main control flow (4), execute the watch-dog recovery constructs (5), and resume the procedure main control flow (6).*

Such powerful scheme has allowed extending autofly to incorporate high-end capabilities such as procedure watch-dogs.

Watch-dogs are procedure constructs that may execute in parallel to the procedure main control flow and are able to monitor nominal operational conditions during procedure execution (e.g. expected trends in telemetry).

In case of non-nominal conditions, the watch-dog interrupts the main procedure control flow (if necessary), executing the necessary recovery actions (which may include the sending of commands to the spacecraft), and resuming procedure execution (automatically with or without user intervention as needed).

Accordingly, autofly has evolved to incorporate the necessary functions and capabilities to support high-end operations concepts. All these elements have been smoothly included as add-ons which are extensions with respect to the core language constructs.

At this stage, it is very important to mention that an essential automation paradigm is to provide the means to deploy automation support through simple means. This implies that autofly is very powerful at the core without needing to make use of sophisticated features. However, some concepts of operations do demand high-end complex capabilities for very specific scenarios. The ability to achieving ANY CONOPS is exactly meaning the need to cover up to the most demanding needs.

The autofly watch-dog support is comprehensive and even allows several parallel overlapping watch-dogs to be executed at the same time. Watch-dogs are meant to be able to monitor procedure execution and take recovery actions as a human spacecraft operator would do. Therefore, algorithms for TM trend analysis are incorporated in order to establish whether actual TM trends are nominal or not.

On the other hand, watch-dogs benefit from all general scripting layer support for procedure monitoring and for the execution of the necessary recovery actions.

All these features are available under a consistent scheme allowing manual operations (e.g. step by step execution), semi-automated operations (e.g. stop at every step/stage), and full lights-out operations.

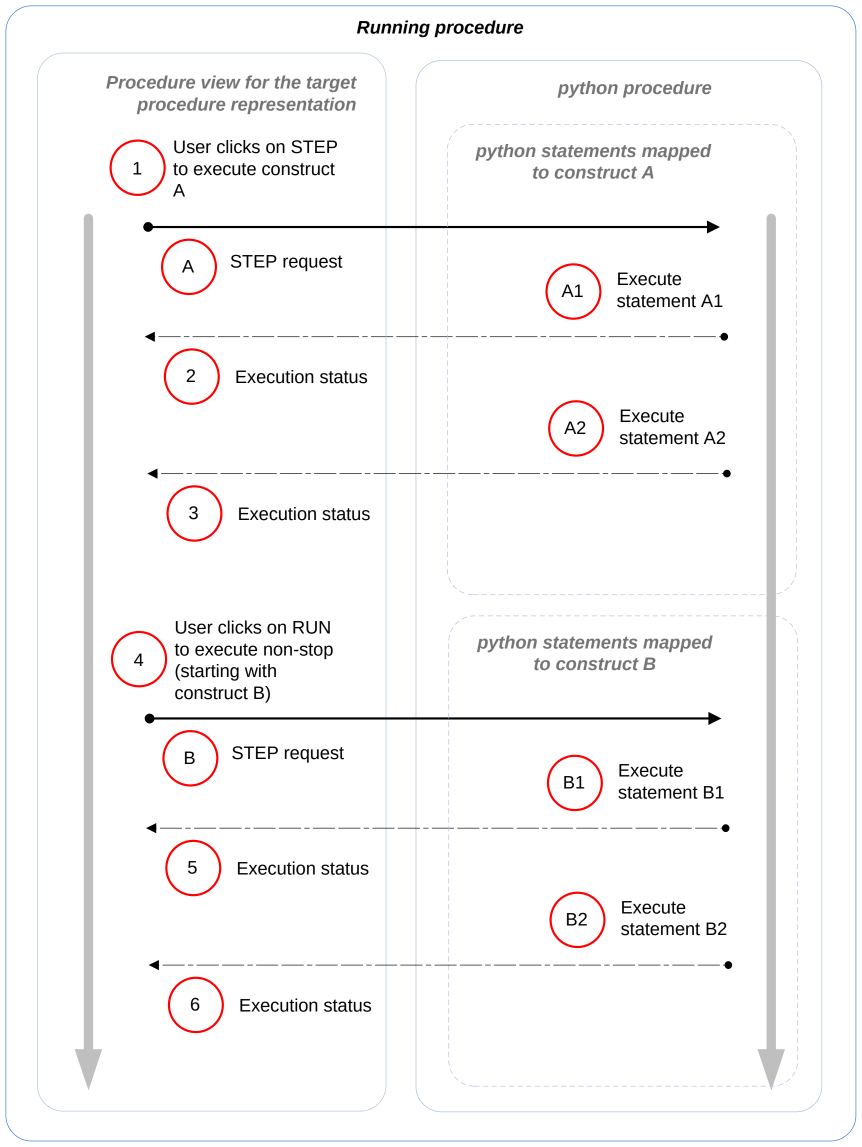
Running procedure
Procedure view for the target procedure representation
python procedure
python statements mapped to construct A
1
User clicks on STEP to execute construct A
A
STEP request
A1
Execute statement A1
2
Execution status
A2
Execute statement A2
3
Execution status
python statements mapped to construct B
4
User clicks on RUN to execute non-stop (starting with construct B)
B
STEP request
B1
Execute statement B1
5
Execution status
B2
Execute statement B2
6
Execution status

Figure 4. Procedure mapping technology. *When the user clicks STEP on the target procedure representation view (1), the system will request to the scripting layer the execution of such construct (A). The mapping technology allows executing the python statements that correspond to that construct (A1 and A2) and sends execution information after the execution of each python statement (2 and 3).*

V. User interface and the procedure mapping technology

One of the elements that are significantly subject to the target concept of operations is the user interface.

In autofly we have innovated by deploying the concept of procedure representation **mapping**.

This is a breakthrough achievement since it allows deploying and executing operational procedures available in different representations (flow-chart, commercial operations language, etc) by transparently making using of autofly's core scripting layer.

That is, the user may view the preferred procedure representation during procedure execution, but execution actually takes place in the form of python procedures.

Procedures may be modified and authored in any of the supported procedure representations (flow-chart, commercial flight procedure language, Microsoft Excel® format, python, etc) and their execution may be monitored and controlled in any of the supported formats as procedures execute construct-by-construct.

With this approach it is not necessary to force a single procedure representation on the operations team, since several views may coexist as enabled by GMV's mapping technology and the Eclipse/RCP user interface plug-in architecture.

The mapping technology establishes:

1. Flexible notification scheme providing the different procedure views with the necessary information to keep their status up-to-date.
2. The correspondence between elements in the preferred procedure view and the python procedure, so manual execution (e.g. step by step) can be controlled according to the preferred procedure representation (e.g. language constructs). For example, a powerful construct in a specific procedure language, may map to several constructs in python or vice-versa.

This core support is combined with editing and authoring tools that are specific to the target procedure representation to be supported as needed for procedure preparation and static validation.

VI. Performance and availability

No matter how procedures are to be represented, today’s concepts of operations often place strong requirements in terms of performance and availability.

We jointly address these requirements, since their fulfillment is increasingly demanding when combined.

In terms of performance, we have encountered the need to execute procedures extremely fast. In these deployments, extensive use is made of python scripting for flight procedures spanning across thousands of lines of code.

Here we achieve what we term as *machine-gun effect*, where the current line showing procedure execution moves extremely fast. In these scenarios, the user is still in full control and can immediately stop the procedure at any time. autofly has specific capabilities allowing to properly managing such powerful capability.

In terms of availability, autofly needs to continuously and consistently store the state vector in non-volatile memory in order to resume execution (after outage) from the very last construct that was being executed when the outage occurred. It is very important to highlight that such capability is part of hifly®’s comprehensive support for high availability and hot-redundancy. That is, autofly includes its own redundancy support consistently with the overall hifly® prime server / backup server state vector alignment policy.

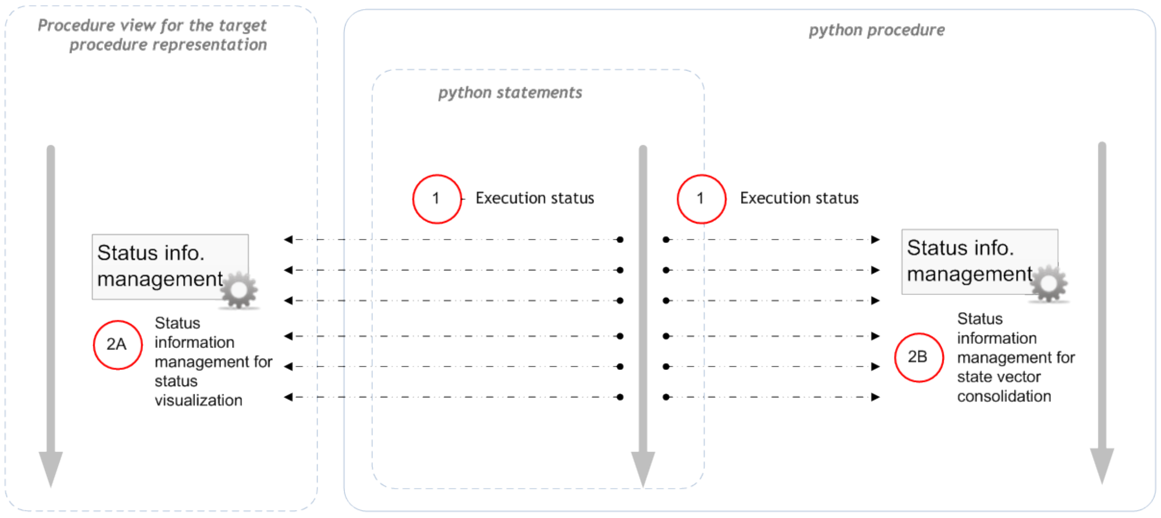
Procedure view for the target procedure representation
python procedure
python statements
1
Execution status
1
Execution status
Status info. management
Status info. management
2A
Status information management for status visualization
2B
Status information management for state vector consolidation

Figure 5. Performance and availability. *The python scripting layer generates execution status information very rapidly (1). Status information is managed for efficient high-performance status visualization in the procedure views (2A). Status information needs to also be managed very efficiently, in order to consolidate the procedure execution state vector without hampering the procedure execution speed.*

VII. Conclusion

hifly® provides comprehensive automation support to achieve the widest range of concepts of operations and includes high-end capabilities to fulfill the most demanding requirements.

hifly® is based on modern scripting and user interface technologies (python and Eclipse/RCP respectively) which facilitate the deployment of different automation paradigms whilst maximizing the common support to the different solutions.

Breakthrough innovation by GMV, such as the procedure mapping technology and other significant achievements, allow fulfilling virtually any concept of operations as needed by different organizations, and further allow different automation paradigms to coexist and be used as the same time in a consistent manner.

References

[1]Morelli, G., et al, “Unleashing the full power of today's technologies for flight procedures automation”, SpaceOps 2010, AIAA-2010-2286.