Product Verification on Mars Express – Routine Validation to Ensure Routine Success

Thomas Ormston[1]
VEGA Space GmbH, Darmstadt, Germany

Daniel Lakey[2]
SciSys, Darmstadt, Germany

and

Michel Denis[3]
European Space Agency, Darmstadt, Germany

Modern space missions have increasingly complex operations, both on-board and on ground. Mars Express is no exception to this rule, with frequent special science operations and a bespoke plan for each individual orbit. While well validated and competent planning systems are the basis on which such operations are built, a routine system of validating the output of such systems is an important component to ensuring safe and successful operations. In this sense we never stop validating on Mars Express and by doing this we are able to ensure a high level of success and minimize and catch errors long before they reach the spacecraft or the ground station.

In order to be able to do this for complex and varied operations we employ a multi-layered approach featuring both manual and automatic checks. This paper will describe what these different levels of checking entail and how the decisions have been made on whether or not to automate. This is a dynamic process with automation being consistently introduced when it can provide more efficient and cost-effective than manual checking. One of the major new introductions on Mars Express is a highly configurable state machine engine that is flexible enough to perform a wide number of checks that traditionally have been manual. The design of this tool and its potential will be explored in the paper.

As well as the methods and type of checks that are employed on Mars Express; the paper will go into some detail on the underlying reasons for why we consider that careful validation of routine operations products is key to a safe and successful mission. The paper will detail the importance of the concept of independent checking – that validation must be separate from generation, whether manual or automated. We will also discuss the balance between delivery validation of a system that will produce ground and space commanding products and the continued validation of products that system produces even after acceptance. Not only is this continued verification valuable and critical in a complex operations environment but it is also very important to eliminate sources of error not covered by delivery validation, including human error and the use of the system beyond its original design purpose.

[1] Mars Express Spacecraft Operations Engineer, HSO-OPM, ESOC, Robert-Bosch-Strasse 5, 64293 Darmstadt, Germany
[2] Mars Express Spacecraft Operations Engineer, HSO-OPM, ESOC, Robert-Bosch-Strasse 5, 64293 Darmstadt, Germany
[3] Mars Express Spacecraft Operations Manager, HSO-OPM, ESOC, Robert-Bosch-Strasse 5, 64293 Darmstadt, Germany

Through all of these factors this paper will present the routine operations practice on Mars Express of constant validation of products and how this has ensured that such a complex mission can be conducted safely and efficiently.

I. Introduction

THE Mars Express mission is a complex mission to operate that requires intricate planning processes to maximize the performance of the spacecraft and its science output. Both ground and space operations for the mission are highly complex and often feature unique activities and opportunities. Managing this is not just a case of careful implementation and planning systems, but also a case of careful validation of all deliverables governing the mission execution. This high level of validation ensures that the mission is conducted with maximum efficiency and safety.

This paper will briefly cover the different types of deliverables that are checked as part of the routine validation process on Mars Express. It will go on to detail why a high level of routine checking and validation is considered necessary for these products. The remainder of the paper will focus on how this checking is performed in an efficient and safe manner. The definition of a routine check is described, followed by a summary of the different types of checking, manual and automated. For each of these types of checking, a description of why each is appropriate is given, along with details on the tools and methods that can be used to perform them. Although the paper focusses on the lessons learned and processes in place on Mars Express, it is intended to be applicable to any complex mission as a “checking philosophy” for safely managing complex operations.

II. Mars Express Command Products

There are several different types of products that are delivered by the Mission Planning team of Mars Express to schedule the operation of the spacecraft and ground systems. These are the files which are validated before being routed to their final destinations. The reference against which to validate them must also be carefully selected to ensure that any failure in the generation chain cannot have impacted both the product and the reference.

There are two main types of product that are subject to routine validation:

1. **Ground station commanding products**
   These products give instructions to the ground stations from the mission on what configurations to set and when to time passes and activities. These also contain information relevant to the ground station on the status and activities of the spacecraft, such as when signal is expected to be acquired and lost and what configuration the on-board TT&C equipment will be in. The products require checking to ensure they accurately reflect what will actually be commanded on board and that they fit within the defined boundaries of ground station capabilities and scheduling. Any errors in these products could lead to problems with receiving telemetry or sending telecommands to the spacecraft. On ESTRACK stations, the ground station products, after an automated conversion stage by validated software is run by the station operations group, directly trigger the software jobs that pilot the antenna and back-end equipment via the Station Computer.
2. **Spacecraft commanding products**
   The spacecraft commanding products are more critical because any errors here could have a direct impact on spacecraft safety. The spacecraft commanding products consist of all the commands that the spacecraft will execute in a given time period, including the parameters on these commands and their time-tags. These require careful checking to first ensure they fit the planned activities of the spacecraft and secondly to ensure that spacecraft safety or operational safety is never at risk. This involves checking which commands execute and when they execute with respect to the plan and with respect to each other, and that the modes and states they establish on the spacecraft and instruments are compatible for a certain, condition-dependent duration. At best any error in these products would lead to missed or incomplete operations. At worst an error in the spacecraft commanding could have serious implications for spacecraft safety.

This is not an exhaustive list and there are many other products which are subjected to a higher level of checking, although this at least forms the core of what is routinely checked as part of Mars Express operations. These are given as an indication of what is checked by the processes described in the rest of this paper, and the implications of any errors, should they occur and not be caught before delivery of the concerned product.

III. Routine Validation versus Delivery Validation

All of the products which are described in the previous section are generated using operationally validated systems and personnel. Therefore, assuming validation was completed thoroughly, they should never exhibit any errors. However, experience has shown that even with a well validated system errors can still occur.

This is often due to manual modifications and/or implementation of combinations of activities that were considered beyond the scope at the time the system was validated. Two arguments could be presented here – that the operations should be scaled back to fit within the scope of the validation or that further validation is required to safely execute the operations. The latter approach gives far greater flexibility and efficiency, even though it drives the requirement for the routine validation described below.

"*What has not been tested does not work*", says a cautionary although constantly reconfirmed operational wisdom. "*Did you test it on the simulator?*" asks the technical manager before approving a new flight procedure. "*Did you validate it on the simulator?*" asks the higher management when something has gone wrong or before authorizing a special recovery.

This operational paradigm is clear and proven for spacecraft operations that are to be done once in special phases, or a few times - or, for most contingency cases, never. These cases are just a small part of the lifetime of the spacecraft - the routine mission, the mission phase normally called *exploitation*, normally represents the vast majority of the time and activities in flight. In this case a simulator focused on the spacecraft, and simulations focused on technical procedures or staff training are of limited help due to the vast amount of diverse individual cases encountered. Traditionally, it is good practice to set up and run on the simulator one or several "representative routine orbits"; this establishes confidence in the correctness of the ground interfaces, the readiness of the team - and often, ironically, the robustness of the simulator. However, depending on the orbit period, the variability and the density of operations, and depending also on the accuracy and modifiability of the "environment" simulated beyond the satellite proper, a few full orbit tests may be of little help to answer regularly, efficiently and in advance the question "*Is the running mission profile possible and safe?*"

Of course, the spacecraft has been designed to function in a flight envelope that the operators are in charge of ensuring. But what if implementation flaws, failures in flight, changes in environment or mission requirements make this flight box evolve often and with significant impacts on spacecraft operability or even safety? Of course, the activities during each spacecraft revolution display a largely repetitive character in routine operations. But what if the changes in the flight envelope, combined with large, time-dependent combinations of spacecraft configuration make every orbit at best similar to the others, but never the same? One scientist of the Mars Express community once said seriously: "*Every orbit of Mars Express is a new mission*". Despite this, the majority of operations of more than 10,000 routine orbits (at the time of writing) have been fully validated before execution.

There were several possible answers to the challenge of how to manage the routine validation of so many orbits and operations and their deviation from the example test cases performed during initial delivery validation. An interesting concept is the "*Mission Simulator*", sometimes proposed by industry as an option in the spacecraft development program. In theory this is elegant: the spacecraft manufacturer plugs together various design models, or downsized versions "*good enough for operations*", in order to represent the overall spacecraft and its environment. In the middle resides the "core" spacecraft simulator, emulating real on-board software and modeling the platform hardware – a simulator developed by the spacecraft manufacturer, or re-using the customer's spacecraft simulator. At the other end of the chain the manufacturer also provides the original spacecraft database, hosting the commands that the ground teams parameterize and combine to operate the spacecraft. Overall this creates a black box that allows the real mission to be played in advance with the best available models, the true spacecraft software, and the real command bits, all in one place!

While this is a brilliant system concept, there are several major issues with this approach:

1) The adequacy and usability of design models when applied to routine operations
2) The time required by the operations team to set up and simulate each planned operational set before loading the real spacecraft, and its compatibility with mission duty cycles
3) The procurement and running costs of such a tool and the hardware to support it
4) The initial validation of this complex system and the continued maintenance required to keep it up-to-date throughout many years of mission (experience shows how difficult this can be for the more static, smaller scope spacecraft simulator)
5) The need for a dedicated process to manage this tool and its operation.

These issues rendered such a system unfeasible or at least unaffordable for Mars Express. However, by taking inspiration from the goals, and challenging the potential issues of this system, it was soon proposed that the mission simulator could be a *process*, rather than a *system*[1]. This is achieved by interleaving validation components with Mission Planning and the transformation of spacecraft activity plans into commands ("sequencing"), although with the independence required to ensure robust validation. This *Mission Simulator Process* is therefore robust enough to ensure all the goals of detailed orbit-by-orbit validation.

However, as a process, the Mission Simulator has to cope with paradoxes, an Authority Paradox and a Time Paradox:

- **Authority Paradox**
  To ensure that the validation is run routinely, it is natural to include this process with the various mission planning activities: science planning, mission planning, spacecraft sequencing. This ensures regularity and completeness of the checks. On the other hand, this ultimate level of validation must remain absolutely independent from the planning and scheduling tools; the design, the implemented software and even the operator must be different. For instance, checking operational products by simply re-running the tool that generated them would be insufficient as the same systematic errors (either human or machine) would occur in both cases.
- **Time Paradox**
  The output of the checks, should they fail, has to be reported as soon as possible to the generator of the products: minimizing the latency for correction is a must to keep the correction costs under control, or even to save the planned operation. However, some critical checks have to be performed as late as possible in the chain, applied to "*real spacecraft products*" that will be as little and as automatically as possible transformed into the commanding bit string that the spacecraft will receive and process to execute its mission. Managing this is critical to ensure that errors found can still be acted upon.

Having to cope with these two paradoxes has strongly influenced the implementation of the mission simulation process. To solve the Authority Paradox, it was decided to implement the validation tools within the operations team: this allows a completely independent check of the outputs generated by the core software (mission planning or mission control system), procured from external companies and configured with their help by the Mission Planners. The development of validation tools by spacecraft engineers has multiple advantages: it can be done in a prototyping approach and within record times; it is an efficient use of the workforce to develop tools to avoid the problems rather than monitor or rescue them; it allows integration of the semi-empirical predictive models of the environment or spacecraft behavior that result from the latest studies which are also performed by the spacecraft operations engineer - for instance which thermal power has been and will be consumed under given conditions. Finally, it keeps the motivation and mental agility of the spacecraft operations engineers, when, by exception, the regular operations become really routine.

The Time Paradox, logically enough, has influenced the timeline of the mission simulation process. Contrary to the ideal mission simulator black box mentioned above, it has been solved with a series of checks between planning and authorizing commands. This gives the overall mission simulator process "*on delivery*" several different stages, with each check on the output of a given planning level allowing it to become an input to the next stage: Science Planning, Flight Dynamics, Mission Planning, Sequencing. The various validation stages before the final sequencing stage are described in Ref. 2. The final stage, to be performed "*as late as possible, on the real spacecraft products*" before authorizing them to "*leave the control room*" and be loaded on the spacecraft for execution (or leave the planning room to be loaded on the station computer) is described in the rest of this paper.

IV. Defining the Checks to be Performed

By establishing a set of constraints to which the spacecraft and its operations must obey then we can begin to form the foundation of the checks. These eventually become a master set of requirements for safe and valid operations that can then be applied either manually or automatically to the deliverable products.

There are a number of steps which typically occur during the development of a checking rule for Mars Express. These are described below, although given experience of these steps, it is more and more possible for us to immediately jump to the end of the process and define clear, general rules up front.

1. *Reactive Rule*

The first stage is a reactive rule, which is how routine validation on Mars Express began, with certain rules being put in place to react to well known or frequently occurring issues. These were the most common issues and putting in place the reactive rules allowed these errors to be caught and corrected before being delivered. However, the reactive approach is limited when it comes to catching new or infrequent errors. If a new error occurs that is not covered by one of the reactive cases then it may be missed. An example from Mars Express of a reactive rule would be:

"*Check that a ground station signal is indicated in the products after the spacecraft exits a radio occultation*"

While this catches one case very specifically, it would not be guaranteed to detect other errors in ground station signal indication. In addition, it is not applicable in all cases, for example if a ground station was not available – this leads to false triggering of the rule, which can cause real triggering to be ignored. Therefore further development is required, into a proactive rule.

2. *General Proactive Rule*

The reactive rule can then be developed into a proactive rule. This is done by examining why the rule needs to be in place – in the case indicated it is to ensure that a signal is indicated when one is present. Proactive rules should represent general principles that must be true in order for the products to be valid rather than specific cases. This inevitably leads to higher level rules covering complex cases, making diagnosing the exact issue slightly more complex. However, the higher level rules mean a far higher proportion of errors are detected and therefore corrected. To develop the previous example into a proactive rule would turn it into the following:

"*Ensure a ground station signal is indicated whenever all of the follow are true – no occultation, ground station available, transmitter on, spacecraft earth-pointed*"

While longer and more complicated, this type of rule then catches all cases of errors rather than once specific check.

3. *Specific Proactive Rule*

The final stage in evolving the original rule is to simplify the initial general proactive rule, making it easier to check and clearer to determine the cause of failure. This does ultimately result in an increase in the number of rules to be checked but this is an acceptable cost given that each individual rule becomes easier to check and less prone to error.

One way to reduce the complexity of each rule is to allow certain rules to rely on others being successful. This is a question for the person defining the rules as to whether this approach is valid, although it can allow a massive simplification of the rules. It does however enforce an all-or-nothing approach to the checking – where any one failed rule could have implications for other rules. Continuing the example from above, the rule mentioned can be further split and simplified using these approaches, resulting in the following rules:

1. "*Check that transmitter is never on during an occultation*"
2. "*Check that transmitter is never on without a ground station available*"
3. "*Check that transmitter is never on without the spacecraft being earth pointed*"
4. "*Check that ground station signal is indicated whenever the transmitter is on*"

Although this splits the rule into four parts, each is very straightforward to check, and the sum of all of these rules completely covers the original rule. It also allows the exact cause of an error to be more properly targeted. Feeding back to the original reactive rule, the exact wording is not covered completely by either of these rules. However, the combination of 1 and 4 ensure that any errors would still be caught, along with many more.

By working through these steps in the development of a rule, a simple reactive response to an error can be turned into an overarching set of rules that would capture multiple different errors and would indeed be capable of catching problems well beyond the original rule. Ideally the specific proactive rules should be specified directly, but given the experience on Mars Express it has been much simpler to evolve the rules as more experience is built up on the mission. Having performed this process several times it becomes easier to jump straight to defining specific proactive rules.

V. Manual Checking

Manual checking of rules is the initial stage of almost any rule implementation on Mars Express. It allows confidence in the rule to be built and helps validate the rule itself as usable. It also always remains as the backup in case the automated checking systems fail. In some cases manual checks are maintained in preference to automating them.

This manual stage is both a constraint and an opportunity of the *Mission Simulator Process* used for routine validation, and described in Section III. A constraint because the operations engineers cannot (initially) rely on any press-button software to confirm that the planned operations are possible and safe; an opportunity because the process forces them to define adapted rules and regularly review the current possibilities and constraints of the mission. A routine mission is never fully static; on the negative side, a spacecraft may underperform in some areas, or simply be aging; on the positive side, new methods, new ideas or results of data analysis may show that more is possible than envisaged thus far. The money of the taxpayers and the trust of the scientists deserve the creativity and care of the operators - and a bit of "*manual*" work.

Not all rules are defined a long time before they are used, and it may take time and some experimentation until they are ready to implement in software. Experience shows that it is possible to operate safely and efficiently before all rules are frozen in software, and often even safer to do so provided a conscious process is put in place and control team members empowered with the responsibility of success[3]. It is necessary to start new activities or new configurations on a small scale, even if they have been technically commissioned, proceduralized and could in principle be produced in large quantities. This low density of novelties is necessary as long as they are checked "*manually*" to ensure care is taken over each.

"*Manual checks*" are established during discussions between the concerned experts of the requesting payload and the impacted spacecraft subsystems. The resulting agreement must be documented at technical level, and if necessary escalated to a formal process called a "*Resource Allocation Board*" where the Spacecraft Operations Manager and Mission Manager give a final conclusion. The retained rules are written in the "*Cockpit Manual*", a Control Room reference document maintained permanently and accurately[4]. The signed handwritten changes get progressively integrated in a formal version of the document and from there can be included as much as possible in automated tools to systematize their application and ensure their stability or controlled evolution via configuration files.

Before implementation of a rule, a trade-off is made between a manual check and an automated check based on the following metrics:

- The frequency of occurrence of a particular operation
- The complexity for a human operator to check
- The risk of error during a manual check
- The difficulty of implementation in a software tool

After eight years of routine operations, some checks have deliberately not been automated because of an unfavorable cost/benefit ratio. Also, certain checks have been kept "*manual*", or rather human-assisted, to optimize the use of "*brainware*": for instance, graphical operations timelines can be prepared by visualization software such that the human brain and eye, extremely efficient in detecting "*visual anomalies*" (an ancient advantage in the struggle for life!) can make a correct assessment in a matter of seconds, while programming this level of "*pattern recognition*" would be complex and costly.

A huge help to manual checking can be the use of timeline tools, such as the one show in Figure 1 below:

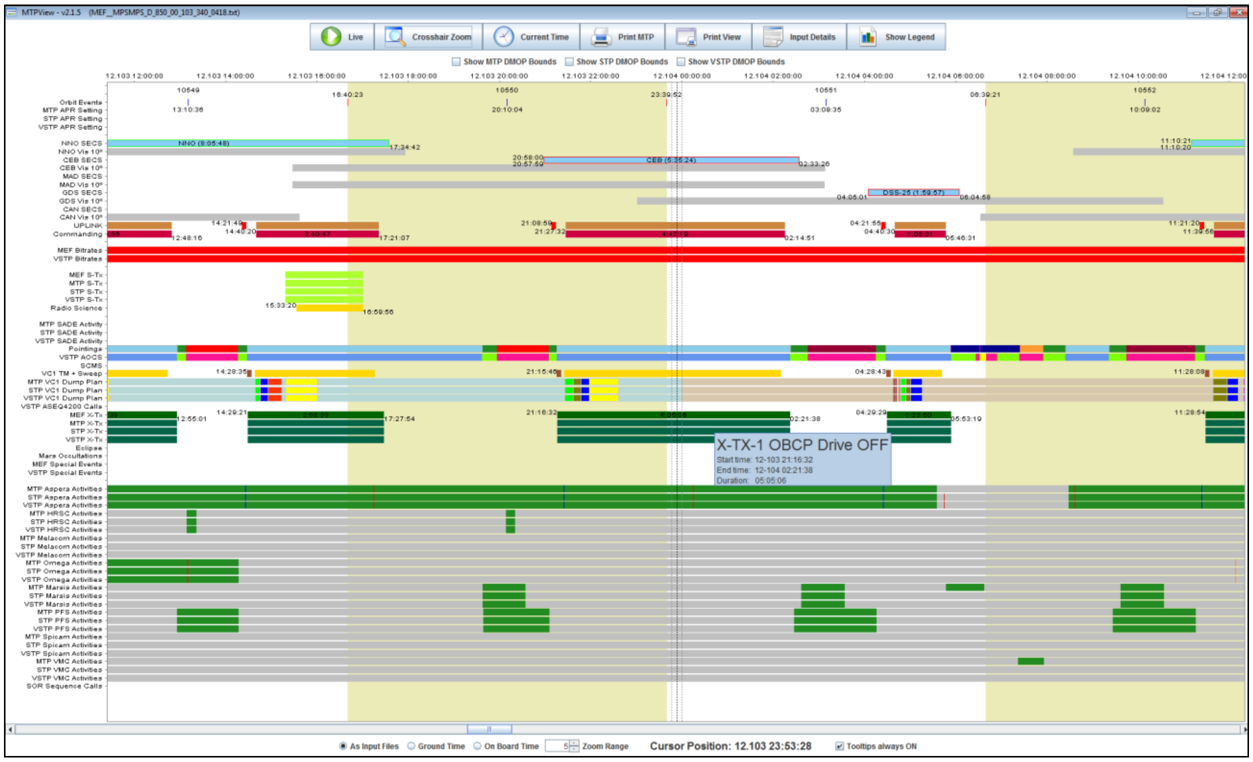

Figure 1. Screenshot of the Mars Express MTPView timeline visualization tool, showing one day of Mars Express operations.

This view represents different activities and events from the source files as bars and markers on a timeline. When tied with the Finite State Machine described in Section VI below it can also provide a visual representation of the checking process, to aid diagnosis of problems.

The timeline tool on Mars Express has been developed to the stage where it can take any kind of input file and display the states or events on the timeline. This allows it to be expanded or simplified as required just by changing the configuration files. The timeline also has uses beyond checking – either electronically or in printed form it is a handy guide to the ongoing activities of the spacecraft.

When used in checking, the timeline view allows very rapid visual checks of rules to be performed. Taking one of the rules previously as an example – that the transmitter must be off during an occultation – it is possible to look at bars representing occultations and bars representing transmitter activity, and check they do not overlap. Although a check to high accuracy is harder to perform, it provides a very rapid way to visually assess the outcome of a rule. When used in conjunction with automatic rule checking, any violations could be flagged on this timeline view, allowing quicker investigation of where the problem lies.

VI. Automated Checking

Automated checking is the ultimate form of most rules applied on Mars Express. Automatic checking requires careful coding but results in a much faster way to analyze rules, particularly where they are repetitive. Automated checking also improves the time accuracy to which a check can be performed and the consistency with which it is applied.

One thing to note about automated checking is that it does not completely exclude human error. Human error in manual checking is more direct but in automated checking an error can still be made when coding the rule. This is overcome by making the assumption that a double human error in the same area will not occur. This means that the checking system should be designed and coded independently of the system generating the products that it checks. This follows the solution to the Authority Paradox described in Section III and is already applied for manual checks where the same engineer should not create and then sign off on their own products. This applies directly to software, and allows confidence to be had in software checking the results of another piece of software.

On Mars Express various tools have been developed to perform automated checking of products and experience of these has allowed a generic "*checking core*" to be developed. This is still undergoing implementation and eventually should be able to replace most, if not all, bespoke checking tools on the mission. The key to this generalization of checking on Mars Express is that almost all rules are defined in terms of states and the transitions between them. This can either mean one system, with a number of states, which must occur in a certain order or at a certain time or it can mean the combination of several states that could be mutually exclusive or vice versa. In one of the rule examples in Section IV, "*Check that transmitter is never on during an occultation*", this can be considered as requiring two states. One state, the Transmitter On state, runs from a transmitter on command until a transmitter off command. The other state, the Occultation state, runs from a predicted occultation start until a predicted occultation end. With these states established, an exclusion rule can be put in place that will flag any case where the states are both active at the same time as being an error.

In computational terms this represents a Finite State Machine. This is what has been coded as the "*checking core*" of the tools for Mars Express, although by design would be applicable to any mission. In practice this core is built from a number of models, each of which is written in an XML-based markup language. A model defines all the known states for that model, the transitions between those states and the events that trigger them. Each model can have one or more sub-models, which themselves can have sub-models, and so on. This allows a complex tree to be built covering all the different aspects of the spacecraft system and its environment, along with ground elements of the system. The lower level models track simple state changes and then in combination with other models can build up a detailed system-level model of the mission. This approach allows all of the models and their interactions to be contained within one unified modeling space. All the models and sub-models in this space can see the different states, parameters and changes of other all other models, allowing changes in one model to trigger a reaction of a state change in another.

The models are fed by "*Events*", which are a certain event, with a given time, and several parameters. In the case of the Mars Express checking tools, these are parsed from a variety of input files - files being checked and files used as reference. A flexible parser has been developed that uses regular expressions to allow parsing of a variety of different input formats. These fall broadly into two groups – flat text files and XML files – although theoretically any kind of input is possible, as long as it can be transformed into a series of "*Event*" objects which are then fed into the state machine to trigger transitions. In the example above, "*Transmitter On*" and "*Transmitter Off*" are "*Events*" that would be parsed from a spacecraft command file. These are fed into the model and matched by the "*Transmitter State*" model, causing the transitions between the Transmitter On state and the Transmitter Off state. Together with this, the "*Start Occultation*" and "*End Occultation*" events from an orbit prediction file would drive the "*Occultation State*" model. As these are then all in the same modeling space, the check can be easily performed, despite the information coming from diverse sources.

The models are allowed to interact with each other by sending events either to other models or outside to the entity running the state machine, allowing it to also interact with the environment around it and, in the case of a checking tool, report errors found during the checking. The models can also have properties, known as parameters, which can be set to show various values that might need to be tracked as part of the modeling process. A further option for more complex interaction is the use of "*External Data Providers*". These act as plug-in scripts which can be added to a model to perform more complex operations or checks beyond the scope of the state machine itself.

The fact that the system is generic allows the designer of the rules to focus more on defining the rules and how they should behave, rather than concerning themselves with the underlying mechanics of the state machine. This makes the tool flexible enough that it can even be expanded beyond the checking field and into a tool for building up a model and then producing an output using that model. Here of course the Authority Paradox of Section III is encountered and can be overcome by ensuring the engineer coding models used to produce outputs is not the same as the engineer coding models to check those outputs. This is acceptable even though the core state machine is common, because it should be treated more as an environment than part of the process itself.

VII. Conclusion

Ensuring a high level of high quality output from a complex mission is a difficult task and is not purely related to what activities can be scheduled. Even more so, it is related to ensure that the activities which are scheduled are valid and safe. Only by ensuring this can the mission output be expanded and developed with confidence. As this process often exceeds the base validation of systems, personnel and software, a rolling process of validation is required throughout a mission lifetime to deliver the highest level of confidence in mission activities. The best way of doing this is to effectively simulate every operation performed by the spacecraft as accurately as possible. By integrating this philosophy throughout the mission, a robust process of mission simulation can be established.

This leads to the requirement of carefully defined checks that must be applied to ensure safe and successful operations. These checks may begin in reaction to specific problems but can only realize their true potential when generalized and simplified to ensure the core constraints and requirements of the system are obeyed. These checks can then initially be tested with a manual implementation, where graphical timeline tools can help with the completion of the checks. Depending on the difficulty of implementation, versus the benefits, a tested check can then be implemented as an automated rule. The use on Mars Express of a finite state machine core has greatly improved the ability to code an implement these state based rules in software.

All of these efforts mean that even with a dynamic and complex mission, flexibility and performance can be ensured through the confidence that comes from consistent and effective routine validation.

Appendix A
Acronym List

ESA	European Space Agency
ESTRACK	ESA Tracking Station Network
TT&C	Telemetry, Tracking & Commanding
XML	Extensible Markup Language

Acknowledgments

The authors would like to thank the whole Mars Express team and mission management for allowing the development of this concept and for ensuring its execution over the years of the mission.

References

[1]Schulster, J., Denis, M., Moorhouse, A., Rabenau, E., “From Mission Concept to Mars Orbit: Exploiting Operations Concept Flexibilities on Mars Express,” *SpaceOps*, Rome, 2006

[2]Rabenau, E., Denis, M., Altobelli, N., “From Mission Concept to Mars Orbit: Exploiting Operations Concept Flexibilities on Mars Express,” *IEEE Aerospace Conference*, Big Sky, 2012

[3]Denis, M., et al. “Human Centered Operations - Reduced Cost and Improved Performance for ESOC Missions,” *SpaceOps*, Heidelberg, 2008

[4]Mounzer, Z., Denis, M., Moorhouse, A., Shaw, M., “Flying Mars Express - A Day in the Cockpit,” *SpaceOps*, Rome, 2006