Strategies and Techniques for Automation as Implemented in EO-1 Flight Operations

Russell DeHart[1] and Baran Sahin[2]
Honeywell Technology Solutions Incorporated, Greenbelt, Maryland, 21054

The Earth Observing-1 (EO-1) spacecraft was developed as part of the NASA New Millennium Program (NMP). In December 2001, NASA Headquarters approved the EO-1 program to begin an Extended Mission operations phase. The Extended Mission seeks to maximize the infusion of EO-1 technology through increased utilization of the satellite and reduce the cost of operations through a Continuous Improvement Program (CIP). A key objective of the CIP is to reduce operations costs by increasing automation use. The three-person EO-1 Flight Operations Team (FOT) is responsible for four main tasks: flight dynamics processing, mission planning, real-time engineering, and engineering analysis. Both automation and meta-automation—using an additional layer of automation to control separate automated processes—are utilized by the FOT. Techniques and strategies for implementing automation are discussed. Criteria are presented to judge the appropriateness of a procedure for automation. They are grouped into categories: ease of implementation, return on investment, and degree of feedback. Factors that affect the design of an automation scheme include: available operating systems, strengths of commercial and in-house tools, tool familiarity, desired error checking structure, and desired nature of operator feedback. Unit testing, acceptance testing, parallel operations, documentation, training, and configuration management are necessary to implement automation in flight operations. Automation lowers labor costs, but also benefits an FOT by standardizing procedures. However, negative consequences can occur as a result of implementation, including loss of knowledge, operators trusting process over output, inappropriate use of automated procedures, and increased system administration issues. The automation implemented by the EO-1 FOT is also discussed. For flight dynamics processing, the orbit determination and prediction procedure has been automated piecewise. Manually, this procedure required approximately three hours; automated, the procedure lasts 30 minutes, during which the user can intermittently leave the workstation to perform other tasks. Automation has been applied to the command load generation procedure, where appropriate. Most importantly, the real-time support procedure has been fully automated and can receive, process, deliver, and confirm commands without human input. The system also monitors telemetry and sends notifications to the FOT, signaling nominal or anomalous operations. While conducting engineering analysis, the FOT performs operations requiring extensive human judgment and custom processes, including analyzing telemetry, designing spacecraft maneuvers, and designing new command sequences. The time savings obtained through automation allows the FOT time to perform these more dynamic tasks. Throughout the course of implementing automation, the EO-1 FOT has learned lessons useful to other flight operation teams. These include the importance of each of the following: 1) configuration management; 2) redundancy in FOT being capable of repairing automation; 3) tool familiarity; 4) adequacy of new employee training; 5) occasionally conducting manual processes; 6) extensive documentation; 7) combining tools of increasing complexity; 8) the ability to automate GUI input; 9) continuing process improvement throughout the mission; and 10) not viewing automation purely as a labor-saving device.

[1] Systems Engineer, Mission Operations and Mission Services—Flight Operations, NASA GSFC M/S 428.2, Non-member.
[2] EO-1 FOT Lead Engineer, Mission Operations and Mission Services—Flight Operations, NASA GSFC M/S 428.2, Non-member.

I. Background

THE Earth Observing-1 (EO-1) Mission was a consequence of the Land Remote Sensing Policy Act of 1992 (Public Law 102-55) wherein NASA was charged to ensure Landsat data continuity through the use of advanced technology. Consequently, EO-1 was designed to flight-validate breakthrough technologies applicable to Landsat follow-on missions. More specifically, the EO-1 design included: multispectral imaging capability that addressed the traditional Landsat user community; hyperspectral imaging capability with backward compatibility that addressed the Landsat research-oriented community; calibration test bed to improve absolute radiometric accuracy; and atmospheric correction to compensate for intervening atmosphere effects.

The EO-1 spacecraft was developed as part of the NASA New Millennium Program (NMP). The NMP was charged to: develop and flight-validate revolutionary technologies; reduce development risks and life cycle costs of future science missions; enable highly capable and autonomous space systems; and promote nationwide technology teaming and coordination. As part of the NMP, the mission of the EO-1 spacecraft was to develop and validate a number of instrument and spacecraft bus breakthrough technologies[1]:

1) Multispectral Imaging Capability
2) Wide Field, High Resolution, Reflective Optics
3) Silicon Carbide Optics
4) Hyperspectral Imaging Capability
5) Atmospheric Corrector
6) X-Band Phased Array Antenna
7) Wideband Advanced Recorder and Processor
8) Enhanced Formation Flying
9) Lightweight Flexible Solar Array
10) Carbon-Carbon Radiator
11) Pulsed Plasma Thruster
12) LA-II Thermal Coating

In addition, another mission objective was established under the NASA Research Announcement NRA-99-OES-01, issued jointly by NASA and the U.S. Geological Survey (USGS). This objective was to evaluate the ability of the instruments to produce images suitable for performing defined science validation investigations. As a result, 30 principal investigators were selected to form a Science Validation Team (SVT).

EO-1 was launched on November 21, 2000 into a polar orbit with an equatorial crossing time of 10:03 a.m. (descending node), an altitude of 705 km, an inclination of 98.2 deg., and an orbital period of 98 minutes. The mission had a design life of 18 months and a nominal life of 12 months.

A. Extended Mission

The EO-1 program completed its baseline mission requirements successfully after one year of operations on November 20, 2001. In December 2001, NASA Headquarters approved a plan to permit the EO-1 Program to embark on an Extended Mission operations phase. The objective of the Extended Mission is to maximize the infusion of EO-1 technology by simultaneously increasing utilization of the on-orbit resource and to reduce the cost of operations through a Continuous Improvement Program. Two of the main objectives of the improvement program are to serve as a test-bed for new technologies, such as SensorWeb technologies, and to reduce the cost of EO-1 operations by increasing the use of automation[2].

B. Spacecraft Operations

The EO-1 spacecraft continues to collect useful science data well past its design life. All three instruments—the Advanced Land Imager (ALI), the Hyper-Spectral Imager (HSI), and the Atmospheric Corrector (AC)—continue to functional nominally. In fact, only two spacecraft anomalies currently have any negative impact on EO-1 operations. First, a failure of the ALI solar calibration aperture selector (utilized to vary sunlight on the solar calibration diffuser plate located within the instrument) causes any ALI solar calibrations to collect sunlight at a constant illumination level rather than over the entire dynamic range. Second, a failure of the interface from the Command and Data Handling (C&DH) processor to the S-band communications subsystem causes an inability to playback engineering data recorded between contacts[1].

To maintain an orbit conducive to imaging, all remaining fuel is being used to conduct periodic inclination burns. With the aid of these burns, the project plans to preserve the quality of the science collected by maintaining orbit altitude and an equatorial crossing time of 10:00 a.m. (descending node). A de-orbit waiver was granted by

NASA Headquarters in October 2007. The EO-1 mission has been granted an extension until 2009, with the possibility of an additional two-year extension.

C. Flight Operations

The three-person EO-1 Flight Operations Team (FOT) is responsible for four main tasks: flight dynamics processing, mission planning, real-time engineering, and engineering analysis. The main procedures included in each of these tasks are described briefly below.

1. Flight Dynamics Processing

The EO-1 FOT is responsible for determining the current spacecraft orbit and to predict the spacecraft orbit for the next 35 days. Using this prediction, the FOT generates and distributes event windows, such as station in-views and eclipse times. The FOT also generates and delivers orbit path data for image planning. In addition to this routine processing, the FOT also designs orbital maneuvers and instrument calibration maneuvers. Lastly, analysis of spacecraft attitude is performed for any special images or anomalies. Autoproducts, Autocon, MATLAB, and Satellite Tool Kit (STK) are used to process flight dynamics data.

2. Mission Planning

The EO-1 program currently utilizes two systems for mission planning. One is the Automated Scheduling and Planning Environment (ASPEN), which is currently operated at the Jet Propulsion Laboratory (JPL). ASPEN utilizes flight dynamics input from the FOT, image requests from mission scientists, and a technique called "iterative repair" to formulate daily activity schedules for the EO-1 spacecraft[3]. In addition, JPL utilizes ASPEN in conjunction with the Continuous Activity Scheduling Planning Execution and Replanning (CASPER) onboard flight software to generate primary spacecraft command loads[4]. References 3 and 4 provide further details on these systems.

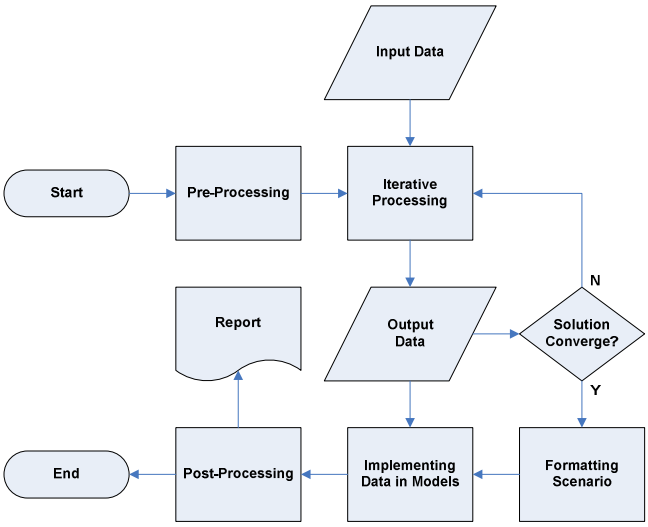

Figure 1. Example Process before Automation.

The other mission planning system is the Mission Operations and Planning SubSystem (MOPSS). Some primary command loads and all backup command loads are generated by the FOT using MOPSS. In addition, the FOT is responsible for updating and deconflicting the spacecraft schedule, generating and delivering daily spacecraft activity reports, and maintaining a database of spacecraft activities.

3. Real-Time Engineering

The FOT is responsible for operating all real-time supports. This includes performing all commanding of the spacecraft, managing spacecraft memory and flight software, and monitoring spacecraft telemetry. Thirty-five months after launch, the FOT transitioned to lights-out operations. The EO-1 mission utilizes the NASA Advanced Spacecraft Integration and System Test Software (ASIST) for the real-time command and control system.

4. Engineering Analysis

The FOT analyzes and trends telemetry received from the spacecraft. Any anomalous data are investigated and necessary actions are coordinated with subsystem engineers. In addition, the orbit is analyzed and spacecraft maneuvers are designed to maintain desired equatorial crossing times and altitudes. The FOT also designs and implements new commanding sequences as requested by mission scientists, such as developing new instrument calibrations. Lastly, any anomalies that occur in the ground systems are investigated and resolved.

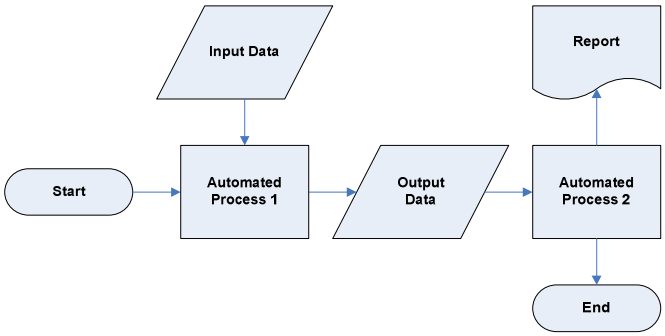

Figure 2. Example Process after Automation.

D. Automation Definitions

In this paper, the term "automation" is defined as the controlled operation of a process by computer software that replaces human labor. Example process maps for a procedure before and after implementation of automation are illustrated in Fig. 1 and 2.

An automated process, such as that shown in Fig. 2 in which automation has been applied to the process in Fig. 1, can be started either directly by human operators or automatically by software. In the case of automatic execution, tasks can be started either at a set time, or *absolute time referenced*, or after a certain criteria is satisfied, or *relative time referenced*.

During the life of a mission, process improvement initiatives may lead to the desire to automate a group of tasks that are currently individual automated processes. In this paper, the term "meta-automation" is used to describe the use of an additional layer of automation to control separate automated processes. Figure 3 shows the resulting process map after meta-automation is implemented on the procedure provided in Figure 2. Often meta-automation requires feedback from the automated sub-tasks in order to appropriately control the entire procedural flow.

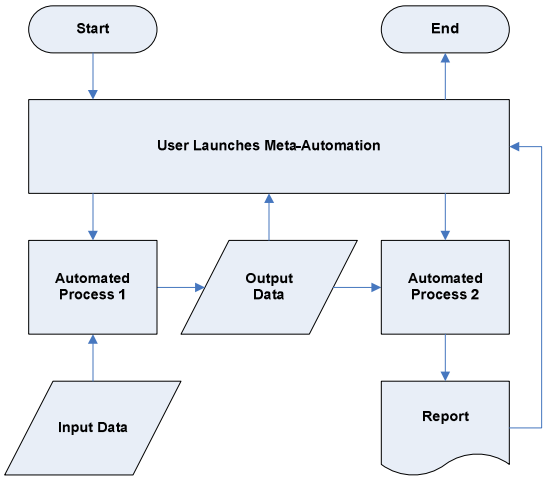

Figure 3. Example Process after Meta-Automation.

E. Tool Classification

The definition for automation provided in the previous section describes software as being the controlling agent. The types of software used can be further broken down into four main groups. They are, in increasing order of sophistication: Operating System (OS) tools, stand-alone scripts/languages, coding languages resident in applications, and automation software. Examples are provided below:

1) OS tools: crontab (Linux/Unix), Windows services (Windows)
2) Stand-alone scripts/languages: Perl, C/C++, AutoIt
3) Resident languages: STOL (ASIST), VBA (MS Applications)
4) Automation software: Autoproducts

An automation scenario will often implement a combination of the above tools, such as in the case of using a cron job to start a Perl script. The next remainder of this paper discusses: characteristics of procedures that are suitable for automation; techniques for implementing automation; advantages and disadvantages of implementing automation in a flight operations environment; the manner in which automation has been applied to the EO-1 mission; and the lessons learned by the EO-1 FOT.

II. Method

A. Identification of Target of Procedures

Many motivations drive flight operation teams to implement automation, including the goal of reducing labor costs. An initial investment in labor is required, though, to implement automation. Hence, when deciding which procedures to automate as part of flight operations, procedures should be prioritized by the degree to which they are amenable to automation. Below are criteria by which to judge the appropriateness of a procedure for automation. They can be grouped into categories: Ease of Implementation (items 1 – 4), Return on Investment (items 5 and 6), and Degree of Feedback (items 7 and 8).

1) Minimal variation in process: Procedures that consist largely of the execution of stable, well-defined tasks are well-suited for automation. The automation of these procedures will not require extensive logic or branching of algorithms.
2) Predictable execution: Procedures that commence at designated times (absolute time referenced) or upon the satisfaction of some criteria (relative time referenced) are amenable to high levels of automation. Execution times can be programmed in advance so that procedure execution is transparent to the user.
3) Localized process: Procedures that are executed on a single platform or location are more suitable for automation than those executed across multiple platforms. Localized processes generally include an easily identifiable nucleus in which to control the process.

4) Well-documented procedures: Often, a Standard Operating Procedure (SOP) or Local Operating Procedure (LOP) document that defines a process can serve as a flowchart to organize the automation code that needs to be developed. In addition, generating the documentation for the automated process is often assisted by referencing these documents.
5) Frequent execution: The labor savings attained by automating a procedure is dependent in large part on the frequency with which that procedure is executed. Little advantage is gained by automating procedures that are rarely implemented.
6) Prolonged execution: Likewise, labor savings are maximized if procedures that require a large investment of labor are automated. Such procedures often require extensive user feedback; however, such tasks can often be addresses piecemeal as described below.
7) Minimal feedback: The use of feedback, whether in the form of ingesting output from other processes or in the form of more complex human interaction, complicates an automation scheme. When human judgment is not required—such as using an ephemeris generated by another process—the implementation is straightforward and simply necessitates additional automation code (e.g., checking for completion of ephemeris generation and performing basic quality assurance). When human judgment is required, such as determining if a calculated drag coefficient is reasonable, other techniques are often required. One technique is breaking the overall procedure into compartmentalized processes.

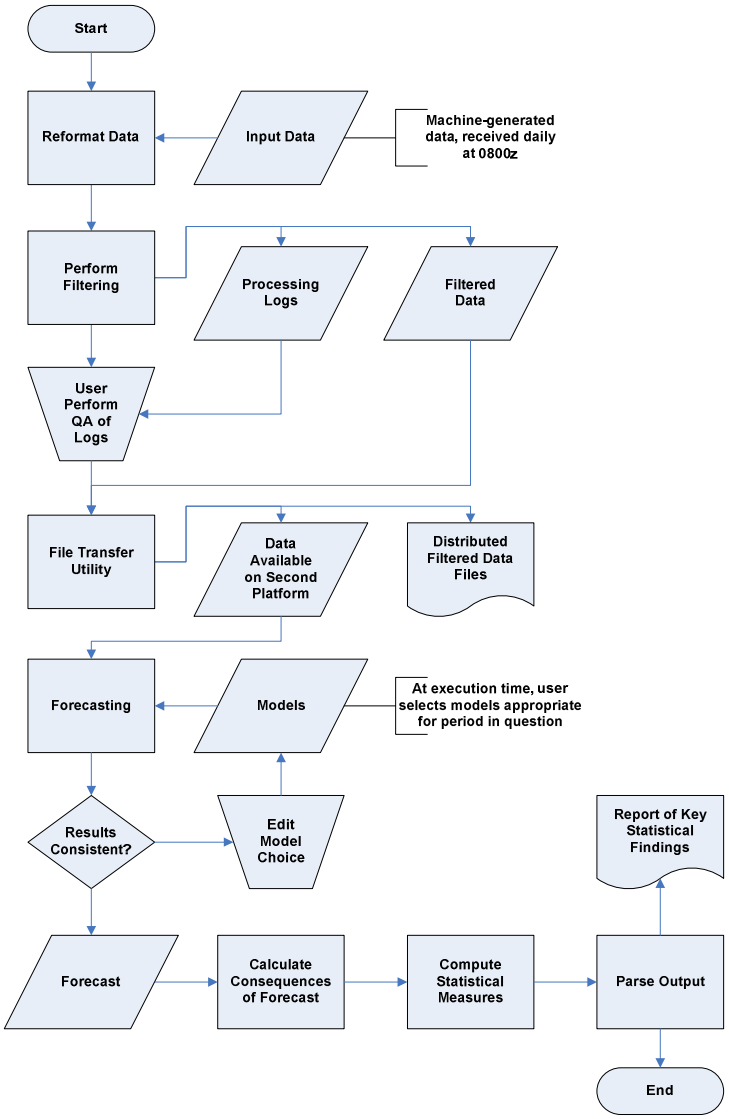

Figure 4. Example Process Map.

8) Compartmentalization of process: The next section of this document describes the method of defining a procedure by developing a process map. After the development of a process map, it is often determined that a procedure is very extensive or has many junctures at which human input is required. In both cases, the process map can help identify portions of the procedure that can be grouped into semi-independent processes, which are amenable to automation. In the case of lengthy procedures, sub-processes can be automated in a piecemeal fashion. In the case of human input, the points in the procedure where operator input occurs can serve as the breaking points between automated sub-processes. If a procedure does not appear to be easily compartmentalized, this procedure may not be suitable for automation.

B. Automation Design

1. Procedural Definition

As was introduced in the previous section, often the nature of a procedure (for example, many instances of feedback) will dictate the design of an automation scheme. It is beneficial to have a blueprint that defines the procedure to be automated on an operational level. A *process map* is a diagram that visualizes the flow of work throughout a procedure, which can vary in detail but typical includes at a minimum: start and end points; inputs from outside the procedure; outputs from the procedure; the tasks performed at each step in the procedure; and any feedback loops from one task in the procedure to another[5]. When identifying feedback loops, be sure to include any error captures. Furthermore, for the sake of automation design, it is also beneficial to label all external inputs as human-generated or machine-generated and as absolute time referenced, relative time referenced, or unpredictable. An example process map is provided in Fig. 4.

The procedure illustrated in Figure 4 is most likely too unwieldy to entirely automate as one single automated procedure. Instead, using the concepts discussed in the previous section, the procedure can be improved by breaking the procedure into automated sub-processes, where possible. The iterative loop in which the user selects models and revises those choices based on output provides a major barrier to automation. The beginning of the procedure, though, contains two sequential tasks that require no human input. Notice that the input data for the two tasks are received at the same time each day. In this situation, it is appropriate to use a "cron" or "at" job to start execution of a script that performs these two tasks. The beginning automation could be extended further, if user review of the output is delayed until after the file transfer utility is executed; in this case, the beginning of this automation script would reformat data, perform filtering, and transfer files. However, most often quality assurance is implemented before files are distributed. In this example, then, the timing of human feedback is limiting the extent of automation. The file transfer utility stands separate from the beginning automation script and feeds into the iterative model selection loop. The end of the example process provides another opportunity to use automation. Three sequential tasks are performed without human input. The timing of the receipt of the input data—the forecast—is uncertain. These three tasks can be automated together as one script, launched by the user after satisfactory model selection. More sophisticated methods of launching the process include establishing a listener that waits for data to populate a directory. Regardless of the method, the resulting overall design for the new procedure is shown in Figure 5.

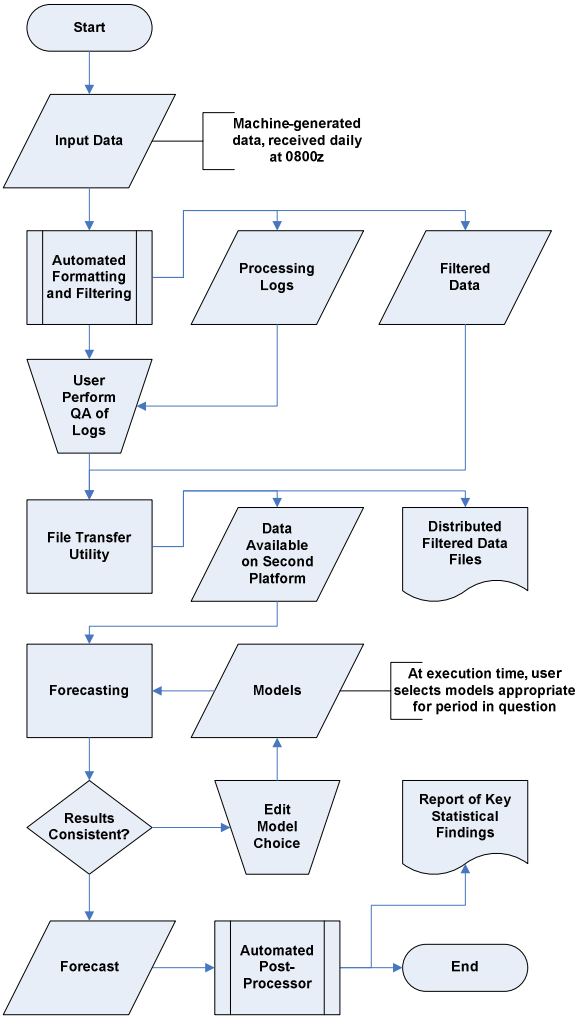

Figure 5. Process Map for New Procedure Design.

2. Identification of Driving Factors

Other factors will also affect the design of an automation scheme. Some affect what automation tools to select whereas other deal more directly with the structure of the automation design. Some factors are provided below.

1) Operating System: The OS being used may dictate what tools are available, both in terms of the capabilities of the OS and in terms of what COTS software may be available.
2) Commercial versus In-House Tools: Code generated in-house will often be less restrictive than commercial software; however COTS software may provide less development time. The need for an open-ended or easily expandable solution should be measured against the need for quick implementation.
3) Tool Familiarity: When selecting a tool to assist with the development of automation, the familiarity of the FOT with available tools should be taken into consideration, particularly in the case of a small FOT. During

the life of the mission, the automation will likely need to be updated, expanded, or corrected. Ideally, all members of the FOT would be capable of performing these actions.

4) Extensive use of error checking in automation is advised. Though this may add complexity to the original procedure process map, this approach will ensure smooth execution and ease during troubleshooting. The decision also needs to be made as to whether to automatically terminate the process upon entering an error state or to allow the operator to correct the condition as it occurs. In the first case, the procedure can be run without an operator; in the second case, the procedure would be executed with an operator in the loop (OITL).
5) Likewise, the timing and nature of operator feedback must be decided. Onscreen feedback can be provided to the operator, but detailed information is best reserved for reports. If all the information can be presented to the operator as log files at the end of execution for inspection, then such procedures can be run automatically. If instead the operator is to act on this data at points in the procedure, the automation must be built in a manner that allows for this OITL interaction.

C. Implementation and Testing

In line with customary software development practice, unit testing—the testing of individual portions of software—should be conducted before automation code is integrated into one large process. This facilitates the identification of errors early in the process, when discovering the root cause of software bugs is most easily performed. After the individual portions of the automation scheme have passed unit testing, the package is ready for integration and acceptance testing. If the automated procedure functions as desired, the FOT can begin performing parallel operations, in which both the original and the automated procedures are performed, with the results being compared to identify any automation errors.

During parallel operations, all conditions in which the procedure is to be performed should be tested. This typically requires the FOT to develop a test plan that lists all the functions and conditions to be tested during parallel operations. Certain operations may be conducted by the FOT very infrequently (such as planning orbit-correcting burns). These operations must be performed for the sake of testing the automation, even though the results will never reach the spacecraft. Upon satisfactory completion of parallel operations, the automated procedure is ready to be implemented as part of nominal operations.

Three other actions should be occurring in conjunction with the abovementioned testing process. First, documentation of the automated process needs to be developed. Often, developing documentation in parallel with the development process ensures that documentation is generated on a timely basis. Second, members of the FOT must be trained to perform the automated procedure. If members become familiar with the automation as it is being developed, they will more likely understand the underlying processes (as opposed to simply knowing how to operate a black-box system). This understanding will allow FOT members to troubleshoot, repair, and improve the automation later in the mission. Lastly, all development, repair, and improvement of the automation should be conducted under a configuration management regime. At a minimum, this should include a repository for the accepted, configuration-controlled software, a suite of test cases for use in testing updates to that software, a method of tracking changes and versions of the software, and documentation for the software.

D. Use of Meta-automation

Process improvement initiatives may lead to the desire to automate a group of tasks that are currently individual automated processes. In fact, this development should be considered during the design of automation. The potential to control the procedure in question by another higher-level procedure should be examined. Accessible entry and exit points for future meta-automation should be established. Incorporating the possibility of meta-automation early in the process provides labor savings later when meta-automation is implemented.

Flight operations includes the use of many different tools across many different platforms. Some tools, such as those that only accept input through a Graphical User Interface (GUI) may at first appear unsuited for automation. However, tools exist that facilitate the use of meta-automation, including the case of automated input to GUIs. Some of these tools will be discussed later in this paper. The implementation of automation can continue throughout the life of a mission, with higher and higher levels of automation being built.

E. Practical Considerations for Use of Automation in Flight Operations

Flight operation teams often implement automation to generate labor and cost savings. However, potential pitfalls are present with the use of automation. Both the advantages and disadvantages should be kept in mind when formulating an automation plan. If the potential disadvantages are kept in mind, preventative steps can be taken to

help avoid these pitfalls. Some of the major advantages and disadvantages to implementing automation are provided below.

1. Advantages

The use of automation helps lower labor costs, but also has the benefit of standardizing the procedures executed by the FOT. This process standardization yields two benefits. First, the likelihood of operator errors can be decreased (though the next section describes manners in which automation opens new opportunities for operator errors). Second, process standardization reduces variability of product, which may assist other agencies in implementing automation of their own. Extensive use of automation can also help the FOT weather temporary shortages in available labor, such as those due to changes in staff or due to inclement weather.

2. Disadvantages

Most of the disadvantages discussed here are exacerbated by improper training of FOT. However, all of these potential pitfalls are present in some form, even in a well-trained FOT. Firstly, the possibility for the loss of knowledge exists, especially if the FOT experiences a high turnover rate. If not properly trained, operators may only know how to operate the automation as a black box. If the automation fails, such members would not know how to conduct the procedure manually without the existence of extensive supporting documentation. Secondly, due to the perceived infallibility of a standardized process, operators may come to trust process over output. In other words, operators may be more likely to ignore problems in output or, even worse, not look for problems because of the perception that the tool does not make mistakes. Thirdly, FOT members may be inclined to use the automated procedures when inappropriate. The likelihood is increased when operators do not understand the underlying processes performed by the automation. Lastly, the expanded use of software, code, and OS tools opens more opportunities for system administration problems. These risks should be considered during the design, rollout, and use of automation.

III. Results

Four categories of automation tools were presented in Section II: Operating System (OS) tools, stand-alone scripts/languages, coding languages resident in applications, and automation software. All four levels are utilized by the EO-1 FOT. The tools being used in each classification are listed below:

1) OS tools: at jobs (Linux/UNIX), cron jobs (Linux/UNIX), shell scripts (Linux/UNIX), batch files (Windows)
2) Stand-alone scripts/languages: Perl, C/C++, AutoIt
3) Resident languages: STOL (ASIST), PSTOL (ASIST), M-code (MATLAB), STK/Connect (STK), VBA (MS Applications)
4) Automation software: Autoproducts

The following sections provide details regarding how these tools are implemented and which FOT procedures are automated.

A. Automation Implemented by EO-1 FOT

As suggested in Section II, procedures that have minimal variation in process and require minimal human feedback are extensively automated. This allows the small EO-1 FOT time to perform more dynamic tasks, such as those described earlier under the heading of “engineering analysis.” The following sections describe the automation implemented in each of the main responsibilities of the EO-1 FOT.

1. Flight Dynamics

The EO-1 FOT is responsible for determining the current spacecraft orbit and to predict the spacecraft orbit for the next 35 days. The FOT uses this prediction to generate and distribute event windows, such as station in-views and eclipse times and orbit path data for image planning. This entire procedure is conducted three times each week and is a very well-defined and stable process. However, the procedure is broad and involves processes on multiple operating systems. This procedure is well-suited for automation, but built in piecewise fashion. The FOT has implemented automation extensively for this procedure. A process map for the orbit determination and prediction procedure is supplied in Fig. 6.

The procedure shown in Fig. 6 consists largely of automated subtasks. First, the OS tools "at" and "cron" are used to control the execution of FDSS_Daily_Script and FDSS_TDPP_Script. The FDSS_Daily_Script collects the input data needed to determine the spacecraft's orbit. The FDSS_TDPP_Script preprocesses the tracking data supplied by the ground stations. Next, a UNIX shell script preprocesses this data, starts STK, sends commands to STK to determine the orbit, and generates logs for the user to examine as part of quality assurance. After the user validates the orbit determination, processing moves to a Windows workstation. A COTS tool named Autoproducts is used to automatically generate the input files needed in the next step of the procedure. Next, the user manually uses the COTS software FreeFlyer to generate the future predicted orbit, along with Improved Inter-Range Vector (IIRV) and Extended Precision Vector (EPV) files. A Perl script is then used to validate the IIRV and EPV. Based on the predicted orbit, Autoproducts is used again, this time on a UNIX system, to automatically send commands to STK to generate the event windows and orbit path data. After the user manually verifies the prediction output, two UNIX shell scripts are used to make file deliveries. Lastly, the user manually delivers the IIRV to the Network Control Center (NCC) for TDRS. Throughout this process, files are shared between the UNIX and Windows platforms via a mapped, shared drive.

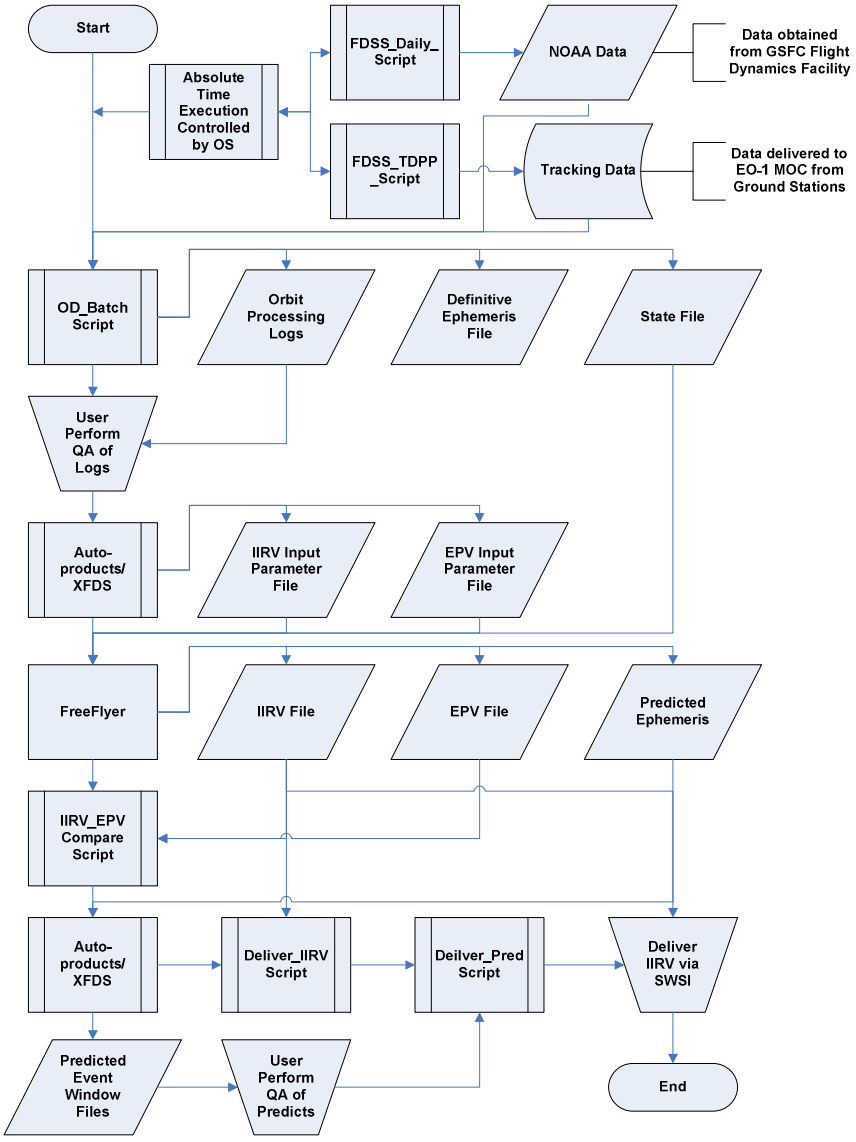

Figure 6. Orbit Determination and Prediction Procedure.

Before the FOT implemented the automation outlined above, all the input file preprocessing, STK commanding, product validation, and file deliveries were performed manually. This entire procedure required three hours. As currently implemented, the procedure lasts 30 minutes, with some of this time being available for the user to leave the workstation to perform other tasks. Since this process is conducted so regularly (three times each week), the FOT experiences large time savings.

As was mentioned in Section I, later in a mission, meta-automation can be implemented to bring together separate automated subtasks. The FOT is currently working on implementing meta-automation to simplify the process shown in Fig. 6. Auto-It is a coding language that allows the user to emulate the keystrokes and mouse operations in a Windows environment. The code structure affords the user the opportunity to automate processes once thought difficult or impossible to automate (such as sending commands through a GUI). The resulting process map for the orbit determination and prediction procedure after the implementation of Auto-It is shown in Fig. 7.

As shown in Fig. 7, all of the tasks performed on the Windows workstation (input file generation using Autoproducts, ephemeris generation using FreeFlyer, and product validation using the IIRV_EPV_compare script) are automated. Auto-It is used as an overlay to send commands to these applications, replacing the need for an operator during these steps. Though this will not appreciably decrease the amount of time needed to complete the whole procedure, it will leave the operator free to perform other tasks during execution of this procedure. As a long-term goal, the EO-1 FOT aims to migrate all these tasks onto a Windows workstation in order to fully automate the entire process using AutoIt.

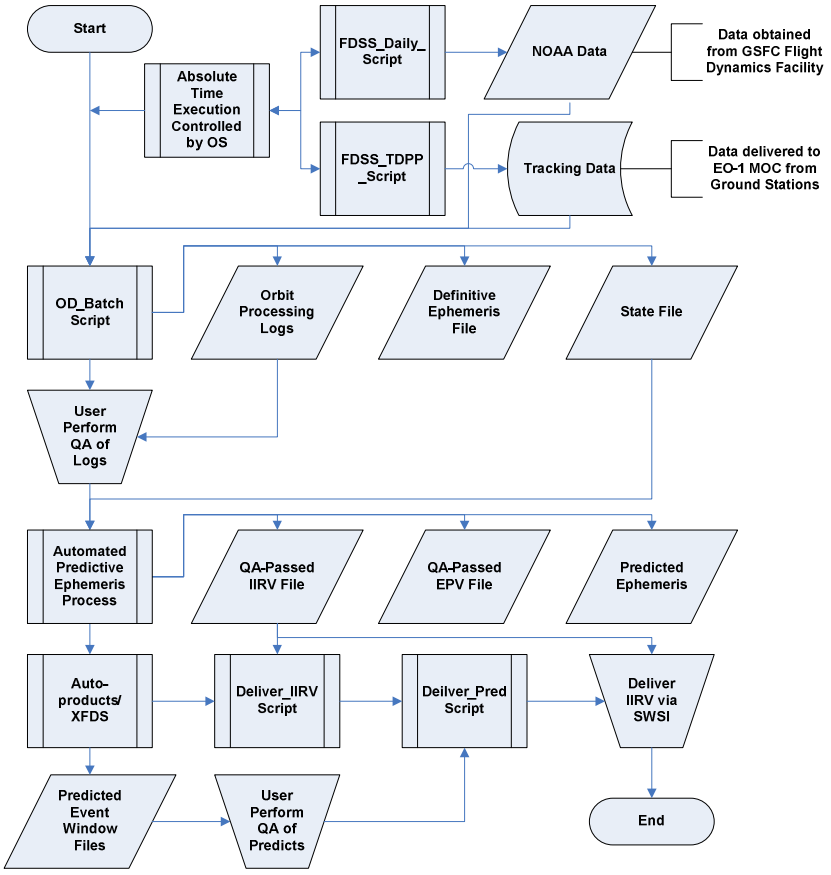

Figure 7. Orbit Determination and Prediction Procedure after Meta-Automation.

In addition to this orbit processing, the FOT also designs orbit maneuvers and instrument calibration maneuvers. The specification of these maneuvers is highly variable in nature and is not performed as routinely as the orbit processing (approximately once each month). Additionally, analysis of spacecraft attitude is performed for any special images or anomalies. These are custom-defined tasks that are performed very infrequently. Since all these tasks are unpredictable and uncommon, these procedures have not been automated. Instead, time savings achieved from the orbit processing and other automation is applied to tasks that require more human input for decision making, such as the design of spacecraft maneuvers.

2. Mission Planning

Each week, the EO-1 FOT works with mission scientists, JPL, and the White Sands Complex Ground Network Scheduling Office (WSC-GNSO) to establish the activity schedule for EO-1. The requested images, ground station in-view windows, and ground station availability are used by personnel at JPL, who utilize the Automated Scheduling and Planning Environment (ASPEN) to formulate the schedule for the EO-1 spacecraft.

Two methods of generating command loads are utilized by the EO-1 mission. In one method, JPL utilizes ASPEN in conjunction with the Continuous Activity Scheduling Planning Execution and Replanning (CASPER) onboard flight software to generate spacecraft command loads, named “goals.” In the other method, the EO-1 FOT uses the Mission Operations and Planning SubSystem (MOPSS) to generate Absolute Time Sequence (ATS) command loads. The FOT generates command loads on a daily basis. The process of producing command loads includes the ingestion of several input files and the generation, validation, and distribution of command loads and activity schedules.

Since the command load generation procedure is conducted daily and consists of some steps that are quite regular (e.g., file ingestion and file delivery), the FOT has applied automation to this procedure, as appropriate. Figure 8 illustrates the current FOT command load generation procedure. First, the mission planning software, MOPSS, receives input data such as ephemeris data, ground station inviews, and eclipse times. These are input using the FDSS_Ingest shell script. Next, if any special maneuvers are being conducted, the operator must manually ingest these files. This task has not been automated due to its erratic nature. Image requests from mission scientists are

processed as well prior to ingestion into MOPSS using the ALI MATLAB tool. Once the final schedule has been established, MOPSS can be used to produce an Unprocessed Daily Activity Plan (UDAP). This UDAP is used by the Command Management System (CMS) to create the command load. Next, the ATS_Load_Check Perl script performs all of the validation checks that were once performed manually. Lastly, the Daily_Activity Auto-It script implements meta-automation to conduct several, once individual, tasks: performs necessary file transfers between operating systems; executes VBA macros in Excel to produce activity spreadsheets; executes VBA macros in Access to enter spreadsheet data into an activity database; and sends emails containing the spacecraft schedule to interested parties.

The main barrier to further automation of this procedure is the nature of the operation of MOPSS and CMS. Both applications require input from a GUI and reside on Linux or UNIX workstations. The tool used by the FOT to automate GUI applications, Auto-It, only functions in Windows environments. The FOT is currently investigating other methods to improve this process. One avenue is to locate and test similar utilities that function in Linux and UNIX environments. Another approach is to eliminate the use of CMS by incorporating updated versions of the ground system command and telemetry software currently in use (ASIST) that can build command loads from the UDAPs generated by MOPSS.

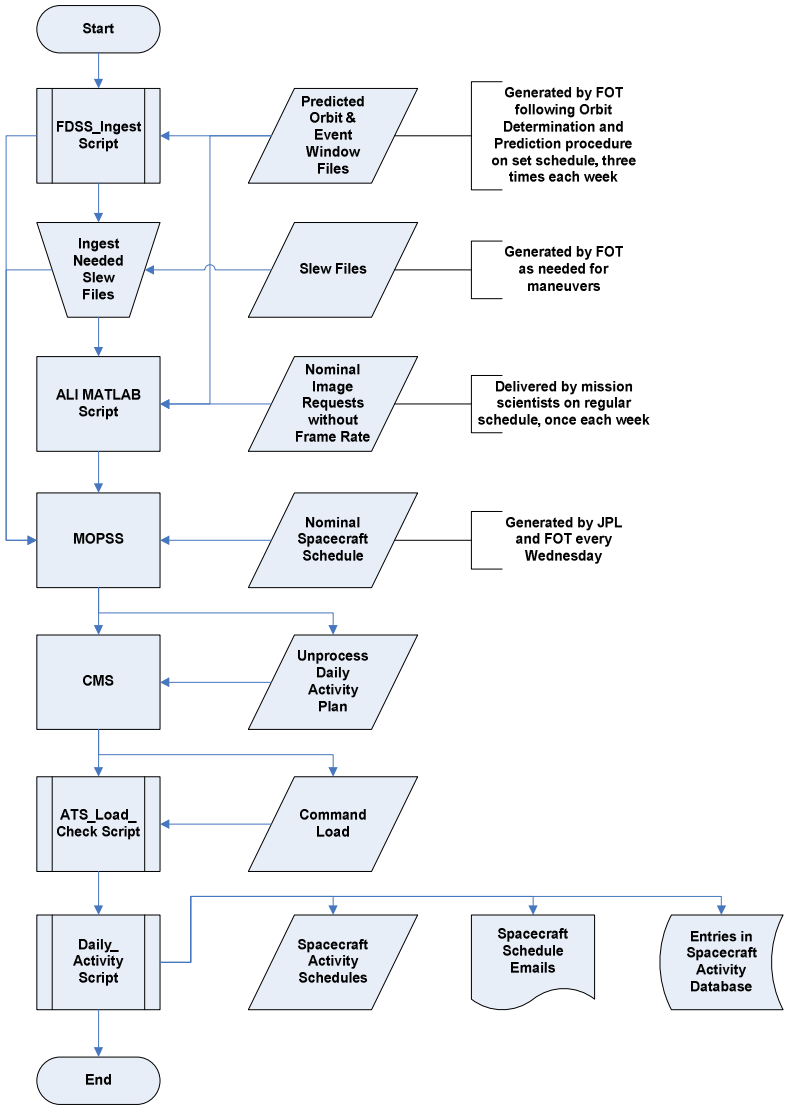

Figure 8. Command Load Generation Procedure.

The FOT is also responsible for updating and deconflicting the spacecraft schedule, due to events such as ground stations becoming unavailable. This work is sporadic and often involves exercising judgment as to when additional supports should be scheduled and contacting all involved parties. Consequently, the implementation of automation would not be appropriate for this responsibility.

3. Real-Time Engineering

The FOT is responsible for operating all real-time supports. This includes performing all commanding of the spacecraft, managing spacecraft memory and flight software, and monitoring spacecraft telemetry. Thirty-five months after launch, the FOT transitioned to lights-out operations. As currently implemented, the EO-1 FOT real-time automation is allowed to conduct all regularly scheduled ground station supports. Even during staffed periods, operators only observe automation during passes. If ad-hoc passes are conducted, such as performing a blind acquisition contact to deliver a supplemental command load, FOT may conduct the support manually. Many automation components are required to perform this procedure. The process is made more complex by the fact that commands for the spacecraft are generated at two facilities (GSFC and JPL). Moreover, commands in the form of

goals are received from JPL intermittently throughout any given 24-hour period. Hence, this procedure must be fully automated and be able to receive, process, deliver, and confirm commands without human input. This automated real-time support procedure is illustrated in Fig. 9.

First, the operator executes the Automation script which parses the output generated by the mission planning procedure shown in Fig. 8. The script proceeds to generate a shell script and an entry in the at job queue of the Linux OS for each scheduled contact. The at job is designed to execute the shell script six minutes before scheduled Acquisition of Signal (AOS). The shell script, in turn, contains Spacecraft Test and Operations Language (STOL) directives that configure ASIST for the contact. This includes launching the Station_AOS_Auto STOL procedure, which is the controlling procedure for the pass. The AOS procedure calls other procedures which are responsible for specific aspects of the pass. For example, the EO1_Limon_Ops procedure establishes the constraint limits for telemetry.

Another procedure which is executed is GN_CMD_Auto. This procedure controls all commanding during the contact.

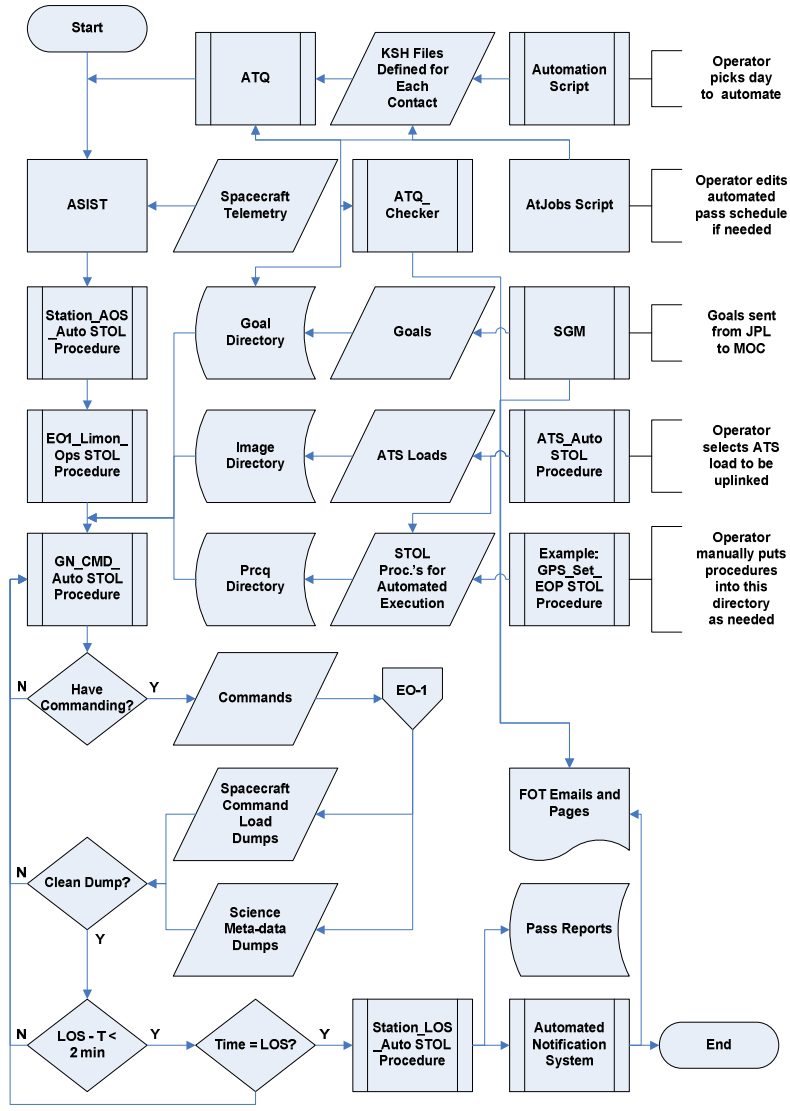

Figure 9. Real-Time Support Procedure.

Verification of commanding status is performed before any command is executed. Also, dumps and compares of command loads are conducted to ensure the commands were delivered successfully before committing to an activity schedule. Commands originate from one of three sources. First, GN_CMD_Auto itself attempts to conduct downlinks of science metadata (e.g., science data directory listings). Second, command loads ("goals") are delivered to the EO-1 MOC from JPL on a continual basis. The Sensor Goal Monitor (SGM) receives and processes these files, sending notifications to the FOT. It also builds a STOL procedure that will uplink the command load to the spacecraft and places the procedure in an archive directory. After the science metadata is downlinked, GN_CMD_Auto checks the directory and executes any goal file uplink procedures, with the oldest procedure being executed first. Third, a queue directory is utilized to place STOL procedures that are to be conducted autonomously by GN_CMD_Auto. Operators or automation can place procedures here. For example, every Sunday, maintenance of the onboard GPS is conducted by the automatic placement of the GPS_Set_EOP procedure in this directory. Another use of the queue directory is the delivery of FOT-generated ATS command loads. The operator places an ATS_Auto procedure in the queue directory. When GN_CMD_Auto executes this procedure, the ground system obtains the desired ATS load and uplinks the command load. Finally, GN_CMD_Auto also verifies that sufficient time remains in a pass to conduct commanding; if the time remaining is less than two minutes, no further commands are sent.

At the scheduled Loss of Signal (LOS), the Station_LOS_Auto STOL procedure is executed, which generates all log files for the contact. Finally, the Automatic Notification System (ANS) parses through these logs and sends notifications to the FOT, signaling either nominal or anomalous operations. Warnings are also sent to the FOT by the ATQ_Checker script that checks the contents of the at jobs queue and the goal directory. For instance, if the at jobs queue is nearly empty (signifying only a few passes remain to be conducted), notifications are sent to the FOT.

The implementation of this automation yields large labor savings for the EO-1 FOT. Before the transition to automated real-time operations, six operators were employed solely to conduct manual passes. The FOT now numbers three individuals, who conduct all the procedures outlined in this document. In addition, the use of automation for real-time supports greatly reduces the likelihood that erroneous commands are accidentally delivered to the spacecraft via operator error.

4. Engineering Analysis

In the domain of engineering analysis, the FOT performs operations that require extensive human judgment and custom processes. These operations include analyzing telemetry, designing spacecraft maneuvers, and designing new command sequences. The only area suitable for automation is the process of ingesting telemetry into the trending database. The Data Trending and Analysis System (DTAS) is currently used by the EO-1 mission for trending telemetry values. The FOT is presently investigating methods to automate the population of the telemetry database, including the use of different trending software.

B. Automation Lessons Learned

Throughout the course of implementing process improvements through automation, the EO-1 FOT has learned many lessons that are of use to other flight operation teams. Some of these lessons are ways to avoid potential pitfalls. Others are techniques that repeatedly prove their usefulness. A summary of the most important lessons learned is given below:

1) Configuration management procedures must be strictly enforced. The change of just one line of code could have considerable impact on a mission. Since such minor changes are transparent to the user, a method must be in place to ensure the operator is implementing the correct version of a procedure. Proper configuration management is critical to continuing reliable operations.
2) Ensure multiple members of an FOT are capable of updating or repairing automation code. Though many or all members may understand the underlying steps, not all members may be familiar with the language or software used to automate the procedure. If such knowledge rests in the hands of only a few (or one) member, the FOT could possibly lose the ability to maintain automation with the loss of just one employee. For small FOTs, this may indicate that every member be familiar with the automation code.
3) More familiar solutions may be a better choice than more eloquent solutions. As a consequence of lesson #2, if presented with multiple choices of tools to implement automation—such as when choosing a coding language—choose the tool most familiar to the FOT. In a flight operations environment, reliability of operations is far more important than the style in which those operations are conducted.
4) Ensure that new employees are trained in performing both the automated and manual versions of a procedure. Otherwise, a mission could eventually reach the point where no members are familiar with manual operations. If the automation fails or needs updating, the FOT will be ineffective.
5) Occasionally conduct automated procedures in parallel with the manual process. Compare the output of both processes to ensure the quality of the product that is delivered and to ensure that knowledge of the manual process is indeed functional. This tactic may be implemented as part of lesson #4, though all FOT members should be included, not just new employees.
6) Ensure that all processes, automated or manual, are extensively documented. Documentation helps not only ensure that knowledge is not lost, but the information contained therein can also serve as a foundation for process maps and algorithms. Therefore, documentation protects established procedures and helps in the development of new ones. In addition, documentation in the form of runtime logs assist in the identification of automation errors. When implementing meta-automation, ensure that all logs of sub-processes are carried through the whole parent procedure.
7) Certain combinations of tools are extremely useful. For instance, the Linux/UNIX at job queue is used extensively in EO-1 FOT automation in conjunction with shell or Perl scripts. These scripts often act as a control agent that launches other more complex scripts. In some cases the subscripts then send commands to software, such as using the STK/Connect module to send commands to STK or using STOL to send commands to ASIST. In this fashion, a broad network of automated tasks can be built with increasing levels of complexity, moving from at jobs, to shell scripts, to Perl scripts, and finally to COTS software.

8) The collection of tools available to an FOT should include tools that enable automation of GUI input. Tools, such as Auto-It, which emulate keyboard and mouse input, allow the user to automate input to any software being used on a platform. This in turn allows the user to bridge gaps in older automation, gaps that existed due to breaks in the process that required GUI input. Bridging these gaps can bring many processes under the control of one meta-automation script.
9) Flight operations teams should continue process improvement through the lifecycle of a mission. If budgetary concerns begin to threaten missions, those missions that deliver useful science in the most cost-effective manner are likely to be extended. One key method to increasing the efficiency of an FOT is the use of automation.
10) Automation should not simply be viewed as a labor saving device. Such a viewpoint may lead FOT members to undervalue the additional work (such as configuration management) that is required to properly implement automation. Automation is an investment that at first requires increased time and labor but ultimately pays off with increased reliability and efficiency.

IV. Conclusion

The EO-1 spacecraft was developed as part of the NASA New Millennium Program. As part of the NMP, the mission of the EO-1 spacecraft was to develop and validate a number of instrument and spacecraft bus breakthrough technologies. In December 2001, NASA Headquarters approved a plan to permit the EO-1 Program to embark on an Extended Mission operations phase. The objective of the Extended Mission is to maximize the infusion of EO-1 technology by simultaneously increasing utilization of the on-orbit resource and to reduce the cost of operations through a Continuous Improvement Program. A key objective of the improvement program is to reduce the cost of EO-1 operations by increasing the use of automation.

The three-person EO-1 Flight Operations Team is responsible for four main tasks: flight dynamics processing, mission planning, real-time engineering, and engineering analysis. Both automation and meta-automation, the use of an additional layer of automation to control separate automated processes, are utilized by the FOT. Both strategies for implementing automation, and the results obtained by the EO-1 FOT, are discussed.

Criteria by which to judge the appropriateness of a procedure for automation can be grouped into categories: ease of implementation (minimal variation in process, predictable execution, localized process, well-documented procedures); return on investment (frequent execution, prolonged execution); and degree of feedback (minimal feedback, compartmentalization of process). After a procedure has been identified as appropriate for automation, the design often begins with the development of a process map. The process map aids in visualizing the flow of work throughout a procedure. The map often identifies factors that affect the design of an automation scheme. These include: available operating systems, strengths of commercial and in-house tools, tool familiarity, desired structure of error checking, and desired nature of operator feedback.

Automation eventually provides labor savings, though a large amount of work must be performed initially. After automation has been built, unit testing, acceptance testing, parallel operations, documentation, training, and configuration management are all necessary to implement the automation into flight operations. The use of automation eventually helps lower labor costs, but also has the benefit of standardizing the procedures executed by the FOT. However, risks are also present when implementing automation, including the possibility of the following: loss of knowledge, operators trusting process over output, inappropriate use of automated procedures, and increased system administration problems. These risks should be considered during the design, rollout, and use of automation.

The EO-1 FOT utilizes many tools for automation, including: OS tools (at jobs, cron jobs, shell scripts, and batch files), stand-alone languages (Perl, C/C++, AutoIt), resident languages (STOL: ASIST, M-code: MATLAB, STK/Connect: STK, VBA: MS Applications), and automation software (Autoproducts). For flight dynamics processing, the orbit determination and prediction procedure has been automated in piecewise fashion. Manually, this procedure required three hours. As currently implemented, the procedure lasts 30 minutes, with some of this time being available for the user to leave the workstation to perform other tasks. Automation has been applied to the command load generation procedure, where appropriate. Most importantly, the real-time support procedure has been fully automated and can receive, process, deliver, and confirm commands without human input. The system also monitors telemetry and sends notifications to the FOT, signaling either nominal or anomalous operations. In the domain of engineering analysis, the FOT performs operations that require extensive human judgment and custom processes, including analyzing telemetry, designing spacecraft maneuvers, and designing new command sequences. The time savings obtained through automation allows the small EO-1 FOT time to perform these more dynamic tasks.

Throughout the course of implementing process improvements through automation, the EO-1 FOT has learned many lessons that are of use to other flight operation teams. Issues that are important to other missions include the importance of each of the following: 1) configuration management; 2) redundancy in FOT being capable of repairing automation; 3) tool familiarity; 4) adequacy of new employees training; 5) occasionally conducting manual processes; 6) extensive documentation; 7) combining tools of increasing complexity; 8) the ability to automate GUI input; 9) continuing process improvement throughout the mission; and 10) not simply viewing automation as a labor saving device, but rather as an investment.

References

[1]Young, J. et al., "Baseline Mission," *Earth Observing-1 preliminary technology and science validation report* [online database], URL: http://eo1.gsfc.nasa.gov/new/validationReport/ [cited March 2008].

[2]Young, J. et al., "Continuous Improvement Plan," *Earth Observing-1 preliminary technology and science validation report* [online database], URL: http://eo1.gsfc.nasa.gov/new/validationReport/ [cited March 2008].

[3]Rabideau, G., Knight R., Chien S., Fukunaga A., Govindjee A., "Iterative Repair Planning for Spacecraft Operations in the ASPEN System," *International Symposium on Artificial Intelligence Robotics and Automation in Space*, SP-440, ESA, Paris, pp. 99-106, 1999.

[4]Tran D., Chien S., Rabideau G., Cichy B., "Flight Software Issues in Onboard Automated Planning: Lessons Learned on EO-1," *International Workshop on Planning and Scheduling for Space*, ESA-ESOC, Darmstadt, Germany, 2004.

[5]Pyzdek, T., *The Six Sigma Handbook, Revised and Expanded*, 2nd ed., McGraw-Hill, New York, 2003, Chap 8.