Automated Sequence Processor – Something Old, Something New

Barbara Streiffert[1], Mitchell Schrock[2], Forest Fisher[3], and Terry Himes[4]
NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. 91109 USA

Operations teams are comprised of highly productive, dedicated, and talented people. The use of scripting is one way operations teams have maintained their productivity. At Jet Propulsion Laboratory, a set of scripts called the Automated Sequence Processor (ASP) is used to automatically check, process, and uplink commands that have been tested and guaranteed not to harm the spacecraft. This software is a significant time saver. However, the software is difficult to maintain because only portions of ASP have been designed from the ground up for its current functionality. Instead, it was written in several scripting languages by many people over more than a decade. Moreover, it does not utilize modern proven technologies, standards, or third party commercial-off-the-shelf software. Work is underway to revitalize ASP. Initially, it is necessary to understand the current state of the software and its capabilities. The next step is to look at appropriate newer technologies including work flow engines and newer programming languages. In addition, the software team must develop a strategy for incorporating the revitalized software while maintaining the existing ASP software at the same time. In order to accomplish both of these goals the software team and the operations teams maintain close communications to ensure that operations needs are met. This paper will describe the work that is being performed to create a new system while supporting an old one.

I. Introduction

High productivity is required for any operations team due to aggressive schedules and the need to ensure that risk is kept to a minimum. One approach used by operations teams at Jet Propulsion Laboratory (JPL), and probably by other teams in other companies, is the use of scripting. Scripting automates operations processes and, in turn, increases productivity. Multiple scripting languages exist and they are generally easy to learn, easy to obtain, and quick to implement. Unfortunately, these same properties may sometimes mean that scripts are difficult to control because they can be modified by anyone. The Automated Sequence Processor (ASP) is a set of about 200 scripts that have been put together over the last ten years. This software performs a necessary function and is required for operations teams to meet their deadlines. The approach and process that has been taken to determine the revitalization effort for this essential software is examined in the following paragraphs.

II. ASP Introduction

ASP has been created because operations teams require rapid automated processing of pre-approved commands available to them twenty-four hours a day, seven days a week. ASP performs several functions, but its main function is to process commands that typically come from instrument teams and pre-approved real time commands from Spacecraft teams. These commands, referred to as non-interactive commands, have been thoroughly checked out

[1]Software System Engineer, Multi-Mission Planning & Sequencing, 4800 Oak Grove Dr. Pasadena CA. 91109 M/S 301-250D,
[2] Cognizant Design Engineer, Multi-Mission Planning and Sequencing, 4800 Oak Grove Dr., Pasadena, CA. 91109 M/S 301-250D.
[3] Multi-Mission Seqgence System Engineer, Multi-Mission Planning and Sequencing Team, 4800 Oak Grove Dr., Pasadena, CA. 91109 M/S .264-235.
[4]Sequence Team Lead, Deep Impact/Epoxi Operations Team, 4800 Oak Grove Dr., Pasadena, CA. 91109 M/S 264-235.

using hardware and software simulators to prove that they will not cause anomalies on the spacecraft. This check-out process occurs in flight operational test beds before the commands are sent through ASP.

Figure 1 below shows the general ASP process pictorially. In order for ASP to process commands, they are first sent to a repository at JPL, and a message is queued to signal ASP that there is data to process. ASP processes that message and retrieves the files associated with these commands from the repository. Initially, they are checked to make sure that they are non-interactive commands. If they pass that check, then they continue through the work flow process determined by a series of scripts that comprise ASP. Once initial checks are passed, the relevant command files are processed by a software discrete event simulator called Seqgen. If no errors are produced by the simulator, the command products are packaged for radiation. Afterwards, operations teams are notified that the command products are ready to be radiated to the spacecraft. After the commands have been radiated, ASP automatically post-processes the radiated command information and stores this information in the repository for complete accountability. The final information is checked by both science and spacecraft operations teams.

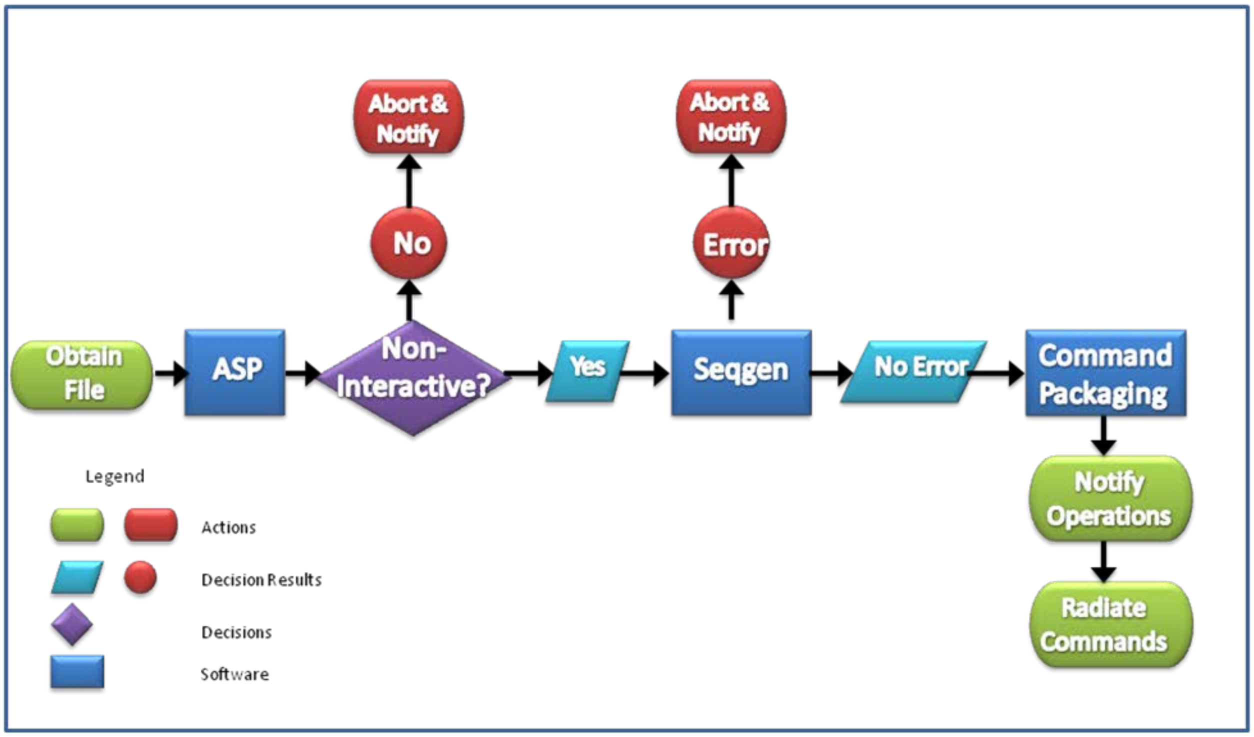


Figure 1. ASP processing of Non-interactive Payload Commands

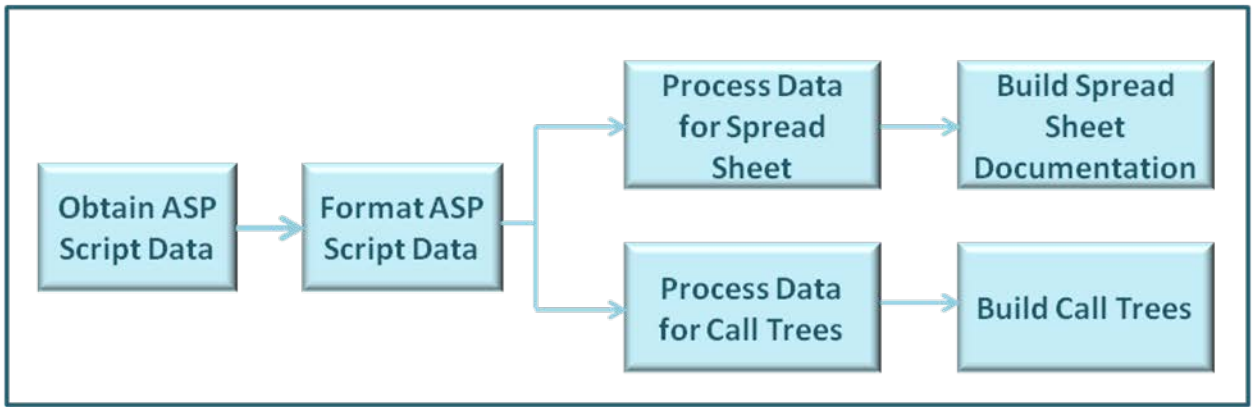
III. Documenting and Analyzing ASP

Initially, ASP began as a grass roots evolutionary development effort by multi-mission operations teams. The effort used scripting to automate manual tasks. New scripts often used smaller, existing scripts to create the automated process. Due to the significant savings of operations costs, but the difficulties in maintenance, it has been determined that this software needs to be revitalized. Since ASP has been developed by multiple operations teams over a significant amount of time to meet changing requirements, the documentation is sparse. Therefore, in order for the development team to be able to document and analyze ASP, the initial instruction must take place by word of mouth. Understanding ASP took two forms. The first was a series of classes taught by a very knowledgeable member of the multi-mission operations team. This class had some training materials as well as power point presentations and used live demonstrations to help with the learning effort. The informal classes were held once or twice a week for an hour or two. This hands-on learning process took about a month.

The second form involved actually documenting the major scripts in ASP. The most important scripts were identified, and a spread sheet was formed to hold information about those scripts. The spread sheet contained the name of the each script, scripting language used, a brief description of the script's major functions, the inputs and outputs of the script, a list of scripts which call the script in question, and a list of scripts which the script in question calls. There were over 100 scripts identified and documented during this process. Figure 2 shows two rows of the spread sheet. The first row identifies the data for each cell. The second row provides an example script entry. Entries were color-coded according to the scripting language used. In order to collect the ASP script data for documentation and to build call trees, software had to be written. Figure 3 shows the software process that has been used to collect the data.

	Details			**Parameters**		**Communication**	
Identifier	**Name of Script**	**Type**	**Functional Description**	**Inputs**	**Outputs**	**Called By**	**Calls**
[1-5] Used for color coordination of "type" column.	Main file name	perl, cshell, shell, awk, text, other	Brief description of purpose of script.	What the input names are	What the script outputs	Scripts that call this script Δ = none	Scripts that this script calls Δ = none
2	add_cmd	c-shell	Adds commands to the spacecraft activity sequence file (sasf)	step_num offset offset_type command param sasf_name	command appended to "sasf_name.sasf" file	file_load	Δ

Figure 2. Spread sheet containing information about ASP scripts.


Obtain ASP Script Data
Format ASP Script Data
Process Data for Spread Sheet
Build Spread Sheet Documentation
Process Data for Call Trees
Build Call Trees
Figure 3. Software process used to create documentation and build call trees.

In addition to the spread sheet holding information for several hundred scripts, call trees have been created for each of the scripts so that the information could be assessed graphically. For some scripts the call trees were very simple. For others, they were extremely complex. For the more complex trees, there is probably a main thread that is used most of the time, but the graphical depiction has been both daunting and a revelation. Using the call trees, scripts that were no longer being used could be easily identified. Figure 4 shows several call trees ranging from simple to complex.

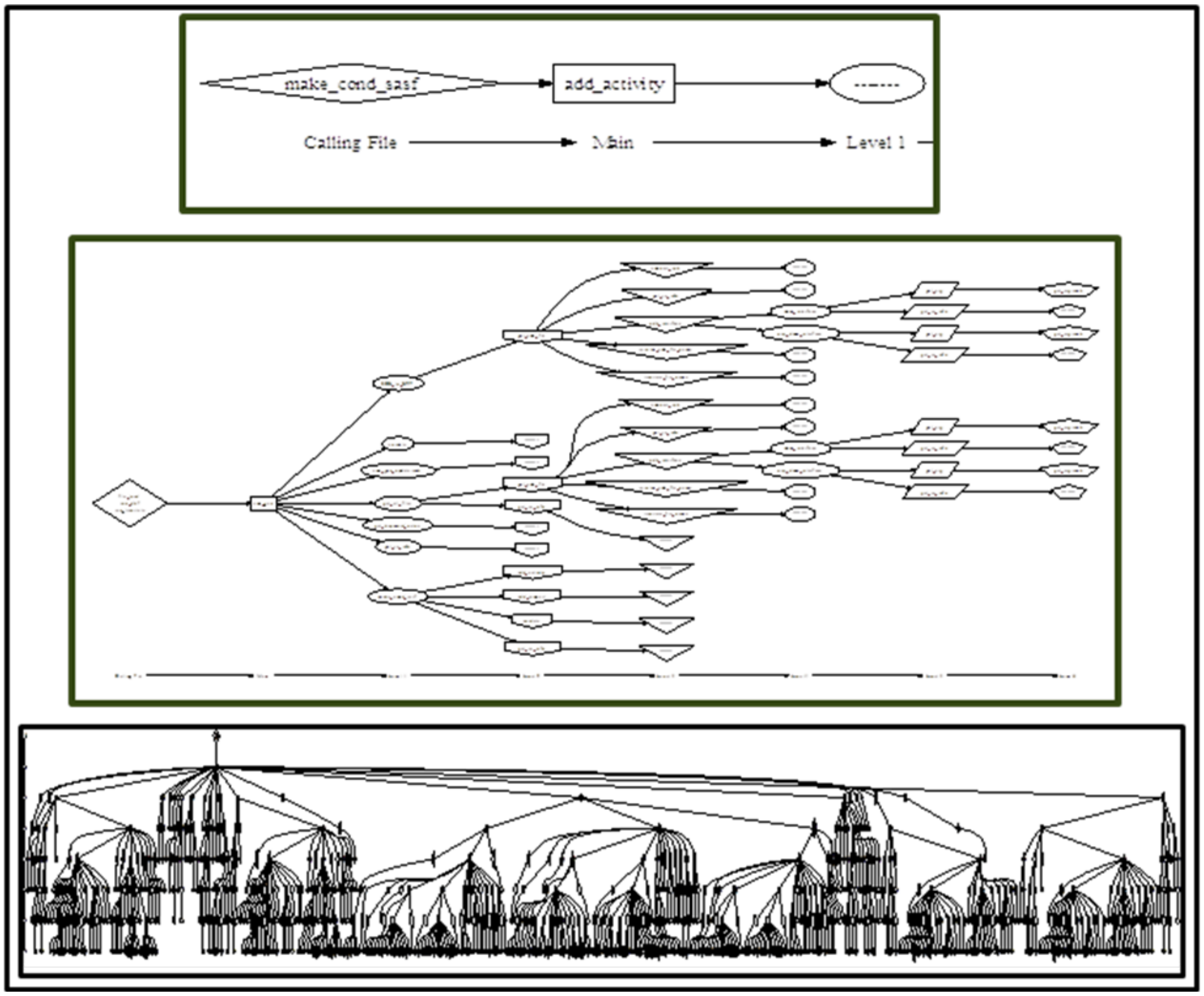


Figure 4. Graphical representations of call trees of varying complexity

IV. ASP User's Working Group

After completing the initial ASP training and documentation, it was time to begin discussions with ASP users. When considering this revitalization effort it was important to make sure that the users' needs and desires have been considered. If the revitalization effort has not been aligned with users' needs, the resultant software would not be used and the effort would have been in vain. Even before the initial ASP training, an ASP User Group had been formed. The initial meeting was simply an introduction of all the participants, a discussion of the goals of the revitalization, and a determination of the logistics of the meetings. At this initial meeting it was decided that the group would meet once every other month and that the next meeting would be after the learning activities had been completed. Having operations users guide the revitalization effort makes a significant difference in the quality of the product.

Each project that uses ASP provided an operations person to represent that team. Group members included team members from Mars Odyssey (ODY), Mars Reconnaissance Orbiter (MRO), Juno, Spitzer, Gravity Recovery And Interior Laboratory (GRAIL), and the Multi-Missions Operations Team (MPST), as well as members of the development team. ODY, MRO, Juno, SMAP, Spitzer, and Grail are currently flying JPL spacecraft. MPST oversees several missions including Odyssey, Juno and Grail. General discussions included modifications required on the existing ASP, existing requirements that ASP fulfills, and the current revitalization efforts. The working relationship between the operations team members and the development team members has been both helpful and successful in bringing the groups together to create a cohesive and consistent effort. Bringing the two groups together into a single team has been one of the key elements in being able to maintain the old and generate the new. In some sense the cognizant engineer for ASP has become an ad-hoc member of operations. Even when there hasn't been a meeting, he stays in contact with various operations team members to get their feedback and find times for testing. The biggest hurdle for the user group is finding a time when everyone can meet. Figure 5 identifies the participants in the working group.

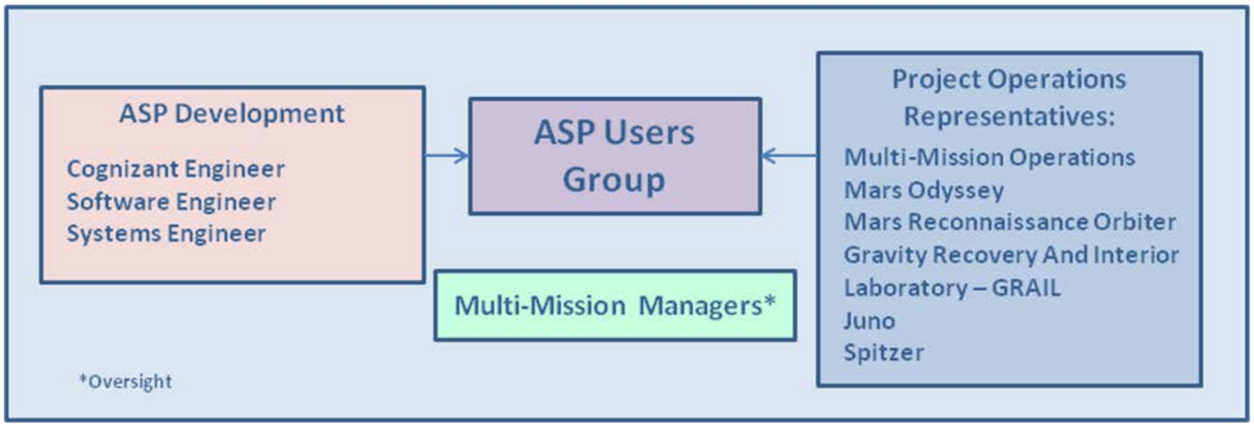


Figure 5. ASP User Group

V. Decisions on Revitalizing ASP

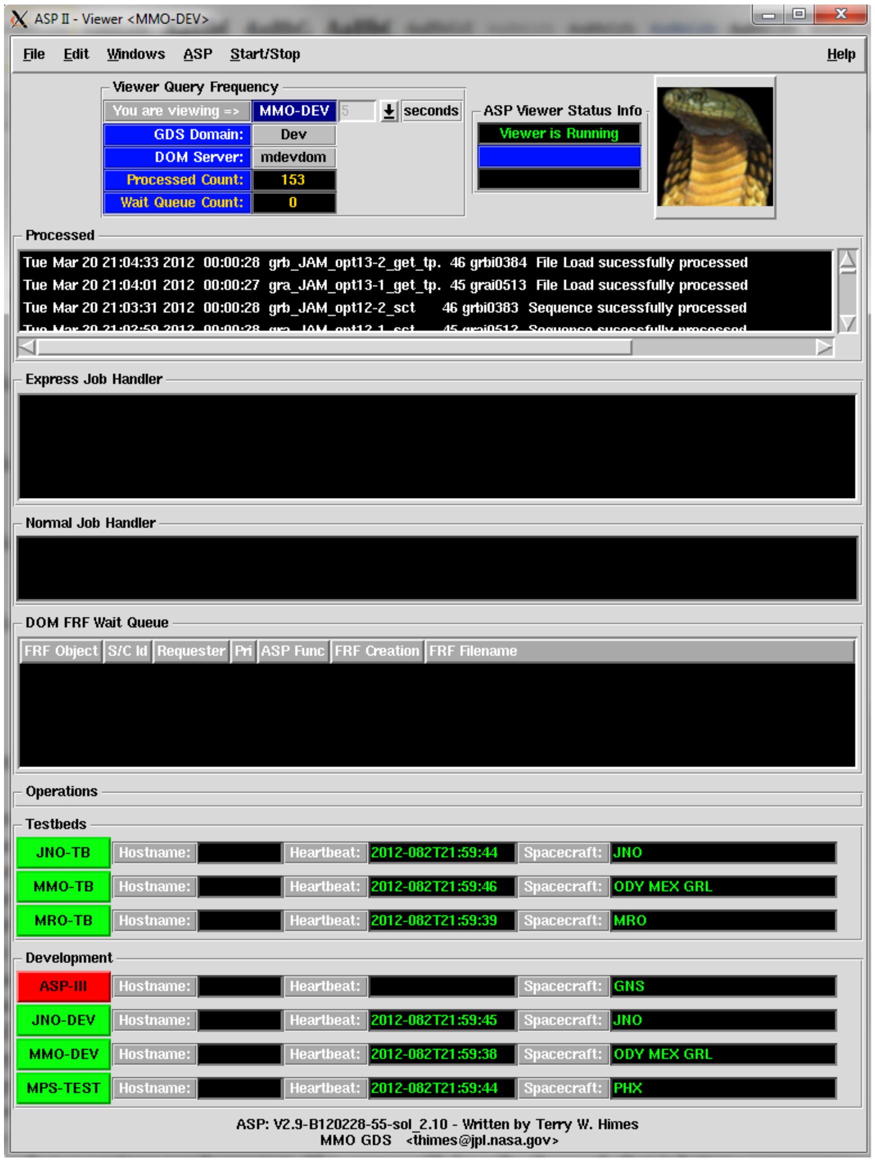
As with any software project performing a revitalization effort there are four basic possibilities:

1. Start from scratch including obtaining a new set of requirements
2. Upgrade the current system using the existing design
3. Use portions of the current system with a new design
4. Use a combination of the old requirements along with new requirements that determine the system's capabilities, but develop a new design and new software.

Each of these options was studied as part of the initial effort. Most of the above options are not viable for ASP because of specific ASP characteristics. As mentioned in the Introduction, ASP consists of many scripts written in many different scripting languages, including Perl, C Shell, AWK, and others, and has been built over a period of approximately ten years. When a new capability was needed it was simply added, usually by someone without knowledge of the rest of the system and without regard to overall system cohesion or design. Some scripts are no longer used, while other scripts are used by many different processes, not just ASP. A design was created for the queuing and display portions of ASP and that part of ASP has been recently updated, but the actual workflow portions have not been reworked in recent years. ASP is difficult to maintain with few people knowing how to make corrections or additions. New third party software, new scripting languages, and new development methods have been implemented in the last few years, but ASP does not utilize any of them. Based on these criteria it has been decided to maintain the current ASP at a low level for flying missions but design and develop new software that provides the same capabilities for the future.

VI. Required Design Characteristics

Since ASP has been designed and developed by operations personnel, one critically important aspect of creating a new ASP is making sure that it has the same functionality as the old one in terms of operations processes, procedures, and flexibility. The current ASP allows users to monitor the state of ASP, what is running in each of the queues, etc. Figure 6 is a screen shot of the current graphical user interface. The top portion of the display gives the state of ASP (running or not running) as well as the computer used for processing. The middle portion indicates the states of various job queues. The lower portion identifies the various operating environments, the computers used to process ASP requests in those environments, and the specific missions for which they process requests. For improved safety and security reasons, the operations environment is separate from the testbed and development environments. This picture shows testbed and development computers which perform ASP processing. In addition to this display, an iPhone app has been built which provides simple ASP state information in a stoplight display (red, yellow, green).



You are viewing =>	MMO-DEV
GDS Domain:	Dev
DOM Server:	mdevdom
Processed Count:	153
Wait Queue Count:	0

```
Tue Mar 20 21:04:33 2012  00:00:28  grb_JAM_opt13-2_get_tp.  46 grbi0384  File Load sucessfully processed
Tue Mar 20 21:04:01 2012  00:00:27  gra_JAM_opt13-1_get_tp.  45 grai0513  File Load sucessfully processed
Tue Mar 20 21:03:31 2012  00:00:28  grb_JAM_opt12-2_sct      46 grbi0383  Sequence sucessfully processed
```

FRF Object	S/C Id	Requester	Pri	ASP Func	FRF Creation	FRF Filename

JNO-TB	Hostname:		Heartbeat:	2012-082T21:59:44	Spacecraft:	JNO
MMO-TB	Hostname:		Heartbeat:	2012-082T21:59:46	Spacecraft:	ODY MEX GRL
MRO-TB	Hostname:		Heartbeat:	2012-082T21:59:39	Spacecraft:	MRO
ASP-III	Hostname:		Heartbeat:		Spacecraft:	GNS
JNO-DEV	Hostname:		Heartbeat:	2012-082T21:59:45	Spacecraft:	JNO
MMO-DEV	Hostname:		Heartbeat:	2012-082T21:59:38	Spacecraft:	ODY MEX GRL
MPS-TEST	Hostname:		Heartbeat:	2012-082T21:59:44	Spacecraft:	PHX

ASP: V2.9-B120228-55-sol_2.10 - Written by Terry W. Himes
MMO GDS <thimes@jpl.nasa.gov>

Figure 6. ASP Viewer Graphical User Interface application

In addition to monitoring capabilities of ASP processor states, operations teams also have the flexibility of modifying existing scripts in real time to meet immediate needs. Often the team doesn't have time to wait for a software engineer to modify the software, test it, and repackage it for delivery. The wait for redelivery could involve missing uplink windows needed to radiate commands and, as a result, cause the loss of valuable science opportunities. The new ASP must include this flexibility or provide a better, faster develop and delivery cycle.

ASP is basically a workflow engine that moves the command data from one software element to the next. The software elements check the commands for constraints and flight rules. If there are no violations, the commands are packaged into the format that's ready to be radiated to the spacecraft. In order to check out the viability and flexibility of using commercially available work flow engine software, a trade study has been performed. First, the criteria required for the workflow engine software to be viable were identified. Next, commercially available software was evaluated based on the criteria. The developed criteria for the commercial-off-the-shelf (COTS) workflow engine included:

1. Process progress represented graphically
2. Customization

3. Mechanism for Creating/Using Custom Code (for example, ReST Interface)
4. Web Interface
5. Ability to use open source development environment (for example, Eclipse IDE)
6. Documentation and Community Support
7. Open Source (if possible)

The COTS workflow engines that have been explored include jBPM, Activiti, Intalio, Apache ODE, Open Business Engine, Open for Business, jFlower, and Pegasus. jFlower was dropped from the evaluation because it only had a download page and little other information. In addition, it has only seven reviews and the rating from those reviews was only 50%. The following table (Table 1) shows the results of the trade study:

Table 1. COTS workflow trade study assessment based on supplier website information

Workflow Engine	Customization	ReST Interface	Web Interface	Documentation	Community Support
jBPM	Y	Y	Y	Y	Y
Activiti	Y	Y	Y	Y	Y
Intalio	N	N	Y	Y	Y
Apache ODE	Y	Y	Y	Y	Y
Open Business Engine	Y	N	N	Y-	N
Open for Business	Y	Y	N	Y-	N
Pegasus	Y	N	N	N	Y

During the trade study if it was not possible to determine if one of the criteria existed, then it was assumed that it did not exist. Pegasus only works on Unix based systems and is not available for PCs. Activiti has dependency conflicts with Eclipse. Of the three work flow engines that met the criteria, jBPM seemed the most mature and implements Business Process Markup Notation (BPMN) 2.0 which is quickly becoming the industry standard. The next step was to build a prototype of the ASP process to see if the work flow engine would work for this application. The prototype was built using the ASP process outlined in Figure 1 and stubs were used for the software elements. Figure 7 shows the graphical representation of the jBPM work flow that was implemented.

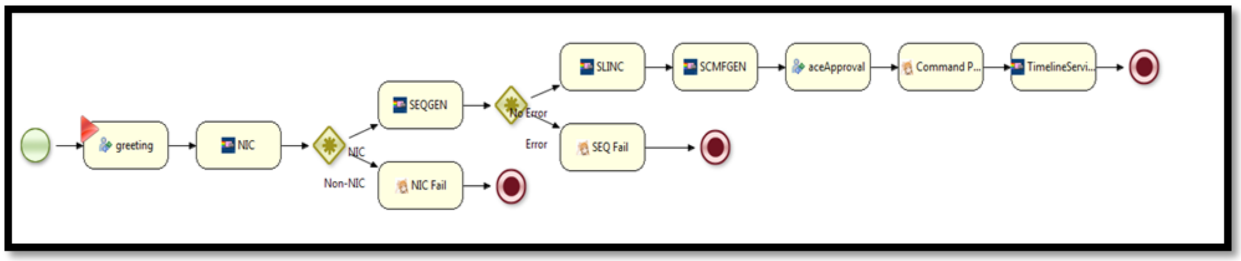


Figure 7. Prototype ASP process in jBPM

The jBPM workflow graphical display is intuitive. The red arrow icon indicates the currently executing element in the process. The diamond icons indicate decision points. The blue square icons indicate process software elements and the person icon indicates human interaction at that particular point. The red circle icons indicate endpoints in the process. The prototype proved that it is possible and even desirable to use a COTS work flow engine. The jBPM work flow engine meets all the criteria identified as necessary. It has both flexibility and configurability. At this time further study on the usability and viability of jBPM is underway. Over the next few months it will be determined if this software will meet the requirements and the desires of operations as well as meeting that one quality, "ease of use," that is always difficult to determine. In addition to building a more fully implemented prototype instead of stubs in jBPM, other work flow engines will be prototyped as well, especially the ones that also met the initial criteria.

VII. Conclusion

At this time, work in this area is a two-pronged effort consisting of maintenance of the old ASP for currently flying missions and development of a new ASP for the future. The ASP User Group meets regularly to discuss the current progress on both fronts, but focuses on changes required to the current ASP. Typically, these changes consist of bug fixes and small modifications to its capabilities. This maintenance work is kept to a minimum, but strives to meet the operations users' needs by providing justifiable improvements. As to the future ASP, it has been determined that it cannot be backward compatible with the current ASP due to the inherent complexities and design problems of the current system. Despite this fact, the new system will still have similar characteristics as those of the old system and must fulfill both the requirements of the old system and new requirements levied against it. In other words, the future will have an Automated Sequence Processor that has "Something Old" and "Something New."

Acknowledgments

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. This work is funded by Multi-Mission Ground Systems and Services (MGSS).

References

[1]Gladden,Roy, Fisher, Forest, Khanampornpan, Teerapat, Waggoner, Bruce, Thomas, Reid and Wenkert, Daniel, "NIPCs: The Operational Sandbox of Science Commanding," *AIAA Meeting Papers on Disc* [CD-ROM], Vol. 1, AIAA 2006-5901, SpaceOps 2006, Rome, Italy, 2006.

[2]Himes, Terry, "Automated Sequence Processor II," *AIAA Meeting Papers on Disc* [CD-ROM], Vol. 1, AIAA 2006-5901, SpaceOps 2006, Rome, Italy, 2006.

[3]Himes, Terry, "Lights-Out GDS – Reliable Unattended Operations Using Automated Processes," *AIAA Meeting Papers on Disc* [CD-ROM], Vol. 1, AIAA 2006-5901, SpaceOps 2006, Rome, Italy, 2006.

[4]Streiffert, Barbara A., and O'Reilly, Taifun, "The Evolution of Seqgen," *AIAA Meeting Papers on Disc* [CD-ROM], Vol. 1, AIAA 2008-3523, SpaceOps 2008, Heidelberg, Germany, 2008.