ISS Operations Cost Reductions through Automation of Real-Time Planning Tasks

Timothy A. Hall
Planning and Software Integration Lead, NASA, Johnson Space Center Houston TX

Dr. William J. Clancey
Intelligent Systems Division, NASA, Ames Research Center, Moffett Field CA

Aaron McDonald
Planning and Software Integration Team, NASA, Johnson Space Center Houston TX

Jason Toschlog
Planning and Software Integration Team, United Space Alliance, Johnson Space Center Houston TX

Tyson Tucker
Planning and Software Integration Team, United Space Alliance, Johnson Space Center Houston TX

Ahmed Khan
Planning and Software Integration Team, United Space Alliance, Johnson Space Center Houston TX

Steven (Erik) Madrid
User Applications, Facilities Development and Operations Contract, Lockheed Martin, Johnson Space Center Houston TX

ISS Operations Cost Reductions through Automation of Real-Time Planning Tasks

A. Abstract

In 2007 the Johnson Space Center's Mission Operations Directorate (MOD) management team challenged their organizations to find ways to reduce the cost of operations for supporting the International Space Station (ISS) in the Mission Control Center (MCC). Each MOD organization was asked to define and execute projects that would help them attain targeted cost reductions goals by 2012. The MOD Operations Division Flight Planning Branch responded to this challenge by launching several software automation projects that would allow them to greatly improve console operations and reduce ISS console staffing and in turn reduce operating costs. These tasks ranged from improving the management and integration mission plan changes, to automating the uploading and downloading of information to and from the ISS and the associated ground tasks that required multiple decision points. The software solutions leveraged several different technologies including customized web applications and implementation of industry standard web services architecture; as well as engaging a previously TRL 4-5 technology developed by Ames Research Center (ARC) that utilized an intelligent agent-based system to manage and automate file traffic flow, archive data, and generate console logs. These projects to date have allowed the MOD Operations organization to remove one full time (7 x 24 x 365) ISS console position in 2010; with the goal of eliminating a second full time ISS console support position by 2012. The team will also reduce one long range planning console position by 2014.

B. History

In 2007, the Johnson Space Center's Mission Operations Directorate (MOD) management team challenged each of their divisions to find ways to reduce the cost of operations in the Mission Control Center (MCC) supporting both the International Space Station (ISS) and the future Constellation missions. Each MOD organization was asked to define and execute projects that would help them attain a goal of a 30% reduction in operating costs by 2012. The MOD Operations Division Flight Planning Branch responded to this challenge by launching several software automation projects that would allow them to greatly improve console operations and reduce console staffing.

The MOD Operations Division serves as the global point of integration for the overall mission plan which includes all of the MCC ISS systems and console positions as well as inputs from the global flight control team (FCT) supporting ISS. In order to produce the integrated products utilized for execution by the flight control team as well as the astronauts onboard the ISS; the Operation Division is required to manage the operational data and products sent to and received from the ISS crew. These products include such things as the integrated ISS mission schedules and the procedures and messages for the crew and ground teams to execute the daily mission tasks. The Operations Division also manages the Space Station Computer (SSC) Operations LAN (Ops LAN) resources and the file traffic uplinked to and downlinked to and from the ISS. With these roles, the Operations Division Flight Planning Branch is also tasked with design, development and deployment of customized software applications needed in order to produce these integrated products for execution.

Managing all these areas requires the integration of inputs and products produced by other MOD divisions, Johnson Space Center (JSC) directorates, and NASA Centers as well as the International Partner's control centers around the world. Because of this diverse set of "suppliers", information comes in via a variety of formats and via several avenues of communication. As assembly of the ISS proceeded over the past few years and new modules and hardware along with the International Partner control centers came online, new issues arose in this dataflow process. These included the globalization of operations requirements, language differences, integration of various messaging and data standards as well as supporting and growing internet and intranet needs to suit this global flight control community. These international needs are somewhat unique in NASA history, since past programs have historically operated in a largely domestic environment. All this new capability and diversity drives inefficiencies into the planning integration cycles and makes streamlining and automating these processes an even greater challenge.

For the Flight Planning Branch, there are three main areas where a significant amount of manual labor was being invested to support the ISS global integration. These areas include: 1) managing the mission plan and associated plan changes from several months prior to a mission through execution, 2) creating and managing the approval and uplink of crew messages and, 3) managing the electronic uplink and downlink of a wide range of information that the crew accesses via their SSC laptops onboard ISS. This electronic uplink and downlink includes the information mentioned previously such as crew mission plan updates, procedures and messages; but also includes the transfer of other information like crew email, imagery taken onboard, crew personal information (news and media), as well as updates to the laptop applications and operating systems.

When the Flight Planning Branch first looked at ways to decrease operating costs, the teams looked to these manually intensive areas (plan management, message generation and file transfer operations) to potentially automate these highly administrative tasks, which could reduce the staffing needs on console and in turn reduce operating costs. Each of the three areas was broken into separate smaller automation initiatives that were managed as formal projects by the MOD. The following sections break out in detail how we approached each area and the solution that was implemented into the MCC and real-time ISS operations.

1. Operational Data History

Since the beginning of the United States Human Space flight, the crew and flight control teams have required a significant updates to operational information during real-time mission execution. From Mercury, Gemini, Apollo, Skylab, and ASTP through Shuttle and ISS, the flight control teams have had some method of distributing operational updates, both to the crew onboard and to the flight control team on the ground during missions. From voice calls and simple teletype print outs on the vehicles to laser printers and now electronic data and files sent directly to applications located on laptops onboard ISS and the Space Shuttle; several generations of technologies have accomplished this same task.

Since this operational information (schedule updates, procedures changes, etc) is not considered critical to crew and vehicle safety, the systems that manage this information have been allowed to evolve in the moderate security arena of the MCC and not be tied to the vehicle avionics and other mission critical software. This has allowed the method for collecting, creating and overall management of this operational information to more aggressively leverage industry trends in office automation and information sharing. For example, this has permitted the process used to generate crew messages for the Space Shuttle to evolve from a proprietary NASA system in the 80's, to a Windows-based platform utilizing Microsoft Office products and modern printers in the 1990's, to web- based tools utilized onboard and the ground and updated electronically in the early 2000's. This flexibility continues to allow the Flight Planning Branch to leverage current industry capabilities and apply automation to several processes on the ground and onboard ISS.

C. Message Automation

2. Overview

When looking for potential operating cost reduction projects, the Flight Planning Branch knew that the message generation and management process was labor intensive but mostly administrative related tasks. The concern was the complexity of the process involved due to the large numbers of users and wide range of inputs received from this diverse global user base. The message generation process had a long history in the Flight Planning organization and automating this process to shift the focus to the external user or message content owner would be a paradigm shift for all parties involved. Despite these concerns, the Flight Planning branch could see the potential cost savings and reduced complexity and decided to launch an automation project to streamline the overall management of the message generation process. The goal was to reduce or eliminate the need for a dedicated console position to support message management.

3. Message History

For almost the entire history of the human space program, the flight planning organization in the MCC has been the console team that generates, manages the approval of, and uplinks operational messages to the crew. The most recent incarnations of the console positions that manages this effort are called the RPE Support (Real-time Planning Engineer Support) pronounced "Rip-E Support" for ISS and the MATS (Message and Timeline Support) console position for the Space Shuttle in the MCC.

The RPE Support and MATS positions developed messages on daily basis that include information for the crew and FCT to perform daily tasks. Examples of the messages include: 1) Daily Mission Summary for the crew (overall picture of the crew activities for that day and other technical data), 2) summaries of the science activities to be performed, 3) updates to existing procedures as well as new procedures to be performed, 4) big picture summaries on upcoming mission events (such as space walks, robotic arm operations, vehicle docking and undocking, etc), 5) stowage information associated with tasks and inventory, and 6) information on Public Affairs events. Since 2000, these teams have uplinked over 60,000 messages to the astronauts onboard ISS and the Space Shuttle.

The process of managing these messages has always been labor intensive and completed by a centralized console position on the planning team in the MCC like the RPE Support position. Since this has always had a "human in the loop" to process inputs from a wide range of personnel supporting the space missions, the human mind has always been involved to process the data, reconcile formats, consolidate thoughts, and overall maintain the quality and consistency of the messages uplinked to the crew and shred with the FCT. This meant that the process was not consistent and could have multiple unique steps that were tailored to each "supplier" of message information. For example, we could accept inputs in a wide range of formats including: Microsoft Word files, ASCII and .txt formats, as well as yellow stickies with poorly written notes on them (think of a doctor's prescription). The initial analysis in 2008 of message generation process for this automation project indicated the RPE and MATS positions were accepting message inputs in over 70 different formats. Luckily, a few standard formats (e.g. MS Word) represented the majority of inputs, whereas a wide array of less commonly used formats were used for a smaller number of message inputs.

When the planning organization targeted message management as an automation project, it was obvious that a new level of consistency for the message inputs would have to be developed. As with almost any automation project, the level of flexibility in the final product (essentially choices or decisions a user can make) will drive up the cost and complexity of the automated solution. For some projects, this flexibility is critical to meet client needs. In the case of message generation, we found most unique formats were accepted because of the flexibility allowed by the

human in the loop, and not because the supplier required this format. The new automated system would have to address the remaining unique formats as part of the process.

4. Message Automation Project

The Message Automation project was initiated in 2009 and was a planned 2 year effort. This project was focused on the ISS message generation and management process since the Space Shuttle retirement was planned for 2010. The goal of the project was to automate much of the administrative role the RPE Support position had in generating, reviewing, approving and uplinking the messages to the ISS crew. By automating this process and allowing the "supplier" of message inputs to create and manage their own messages, along with streamlining the approval and uplink process, the planning organization could eliminate the need for a dedicated message management position on console. There was concerns expressed by the FCT and the message "supplier" community that the results of this project would be to push the current message creation and management process out to their organizations, and simply off load the Flight Planning organization of this task. But, through initial analysis of the new process, the message automation team felt strongly that if done correctly, the message content "supplier" and the flight planning branch personnel would both have less overall workload and less dependence on one another to complete the new message generation cycle.

The Message Automation project would modify a legacy message system called JEDI (Joint Execution Package Development and Integration) to include this new assisted message generation capability, or "Message Wizard" , along with utilizing an Application Programming Interface (API) in order to develop web services to share data with other applications. The legacy JEDI system is a web-based application that has been used on both Space Shuttle and ISS for several years. The JEDI tool was completely refactored in 2006-2008 from the classic ASP language to modern C#.NET utilizing AJAX and other modern web technologies. This prepared JEDI for increased automation and adaptation of a future services architecture for web tools used in the MCC.

The legacy JEDI-based message generation process started with users or "suppliers "of message inputs submitting the inputs into another web based application called Electronic Flight Notes (EFN), often called the "Flight Notes" system. EFN is a core tool for the FCT and is where a large percentage of all console issues and actions are documented and tracked. The message input supplier would open an EFN with a brief description of the content and need for the message, and also attach a file with the technical message content. The FCT, lead by the Flight Director, would review and approve the attached message inputs in EFN. The EFN and attachment may be revised or "rev'd" during this approval cycle by the input supplier. Once the EFN attachment was approved, the RPE Support position would use the JEDI tool to take the EFN attachment or "approved" message content and place the information into a standard message template and verify proper formatting for an ISS message. The RPE Support would then manage a very similar approval process for the JEDI message with the supplier and then with the entire FCT and Flight Director. The JEDI message may also be revised or "rev'd" during this approval cycle by the RPE Support position. Obviously, this process was inefficient by requiring two review cycles with the FCT for the same message content.

The Message Automation project team first performed a process improvement analysis on the full process. The resulting process combined these two approval cycles and took the RPE Support out the process almost completely. The new process allows the message input "supplier" to create the message in JEDI as they create the initial EFN. So now when the FCT reviews the EFN inputs they are actually reviewing the JEDI message itself, and approving it at one time for both content and formatting. This was accomplished by creating interfaces (via APIs and web services) between the JEDI and EFN tools which allows the end user (message input supplier) to work the message generation through the nominal interface, the EFN tool.

The message initiator now opens the EFN as nominal, and indicates they want to create a message. The message initiator is then automatically walked through a Message Wizard, (which is actually part of the JEDI tool), that prompts the user for the required JEDI message information and content. This new JEDI message automation process then applies the correct formatting and saves the message in the JEDI ready for FCT review. Once the FCT reviews and approves the message, it is automatically placed into the queue for uplink to the ISS (by the OCAMS tool described later in the paper). This greatly simplifies the process for both the message suppliers, but the entire FCT by completing one review cycle for each message.

The new JEDI Message Wizard not only helped automate the creation of the message and the approval cycle, but helped automate several of the ancillary steps in the process that the RPE Support position had to manage manually. For example, before the FCT can submit an EFN for content review and message generation, the document must contain a unique JEDI message identity (or simply called a message number), which is incrementally created by the JEDI tool. In the past, message input "suppliers" would contact the RPE Support console position via multiple contact methods (MCC voice loops, phone calls and email) to create a message number "placeholder" for them. Certain information is required for the placeholder that the "supplier" would provide. The message automation project now provides a JEDI web interface that gives a "supplier" the ability to generate and manage JEDI message number placeholders without any interaction required with RPE support. The release of the Placeholder Manager further reduces the dependence of "suppliers" on RPE support position and further streamlined the process. The new message automation process also handles a unique message that is frequently received from the Russian Space Program called the Russian Radiogram. In the past, the message content for this message was delivered via email to the RPE Support position. The message content was always in a consistent format with very little variation. The RPE Support would manually retrieve the Radiogram input, and using the JEDI tool, manually create a message using the proper Radiogram template. The message automation project has made it so the Radiogram inputs can be automatically created when the input file is received with little or no contact from the RPE support position. Again, taking the human out of the loop for processing this data.

5. Message Automation Technical Approach

The message automation relies heavily on the altering of Microsoft Word documents, the most common message input format. Because of licensing arrangements, Microsoft Office cannot be installed on a server without an end-user license for each person that connects to said server. The number of users connecting to the MCC web servers is on the order of thousands during anyone month and they are dispersed among many facilities, companies and networks. Thus, it was quickly determined that another solution was necessary to allow the message input suppliers to alter Word documents as they created and managed messages. The first version of message automation utilized a COTS product to locally mimic MS Word capability to allow users to open and alter documents for the required message formatting. The specifications of the COTS product indicated it would be able to perform this task affectively. This solution with the COTS software was tested thoroughly and put into operations.

After a short time, the ISS crew and FCT began to notice that some messages were inconsistent with respect to the standard formats (even though initially input the correct way). Moreover, in some rare cases, documents lost content after being modified by the COTS product, which lead to operational concerns over losing technical content. The issues appeared inconsistently and this software "bug" was escalated to the COTS vendor. The team worked with the COTS vendor to try and resolve the issues, but it became apparent that their patch development schedule was not going to fit into the Message Automation project needs. It was decided at that time a proprietary solution was necessary for the customization and proper scaling of message automation. The ISS team went back to the legacy process for developing messages until this could be fixed.

Approximately six months after the initial release of message automation solution, the team had developed and released a solution which operated off of the local client's Microsoft Word license. This solution meant that the word documents could now be altered using the native Microsoft Interop API instead of a non-customizable COTS product. This is a unique use of the Interop API and a unique combination of both web app/client applications. Traditionally, web applications (like JEDI) do not communicate with client (formatting) applications. Internet Explorer employs the notion of an ActiveX plugin as an avenue for which the browser can invoke a small subset of MS Windows applications. However, with the JSC domain in the Federal Desktop Core Configuration (FDCC) settings' "trusted sites" list, this meant that ActiveX content (formatter) could be launched from the JEDI web application, allowing more client app like capability for the users. To ensure this approach was viable, the JEDI team worked with MOD and NASA network and security teams to verify this solution met all policies and guidelines for web applications. This limited use of ActiveX enabled the JEDI team to replace the COTS product with a proprietary and fully-customizable solution for the message generation process.

The JEDI Message Wizard is in use today for ISS and the team continues to roll out this new capability to more and more message input "suppliers". Although the Flight Planning Branch still creates some of the ISS messages directly, the branch no longer schedules a RPE Support position on console as part of the Op Plan support team in the MCC.

D. Orbital Communications Adapter Automation Using OCAMS – Orbital Communications Adapter Management System

6. OCA Overview

As mentioned previously, a significant amount of operational updates are generated for the crew and flight control teams during a space mission. But this operational data is just one of many types of information that needs to be sent to and received from the crew during a mission. Other official and unofficial data and information are sent to the crew and downlinked from the crew on a daily basis. These include experiment data and imagery, as well as crew email and personal support items, such as news, webcasts, and videos. All of these are sent to and from the ISS via the OCA (Orbital Communications Adaptor) system located both in the MCC and on the ISS Operations LAN (OpsLAN) SSC's (Space Station Computers) onboard. The MCC console operator for the OCA system is known as the OCA officer and is part of the ISS Ops Plan console in the planning organization.

Since the beginning of ISS assembly, the OCA Officer has been the console position that managed all the OCA file traffic to and from the ISS crew on a daily basis. With the ever-increasing size and complexity of ISS and its crew, the OCA Officer role was getting more and more taxing and complex to execute. In 2006-2007, as part of the effort to implement automation on console and reduce operating costs for the ISS, the planning organization launched a project in conjunction with the Ames Research Center (ARC) Intelligent Systems Division to develop an automated system based on a previous software agent concept developed at ARC [ref MAA, MA].

The OCA system resides in the MCC on a dedicated moderate security platform called the OCA LAN (Local Area Network). The OCA Officer was required to perform the uplink and downlink of information to and from ISS (via a connection to the ISS high security network) while working on the OCA LAN. Once the transfers with ISS were complete, the OCA Officer was then required to repeat the file transfer process on the MCC Mirror LAN (a ground copy of the ISS Ops LAN) to keep it in synch and then distribute the information to multiple parties from another moderate security network known as the MCC Automation System (MAS). The information was then distributed from the MAS network across JSC as well as to parties at other NASA centers and the International Partners Control Centers across the globe. This information was also distributed via multiple communication paths, including secure FTP, thumb drives, and email. The distribution of the information also required notification of the parties involved. The OCA officer would update their request status via MCC web based tools such as EFN, along with emails and calls via the DVIS voice loops to other console positions or phone calls to external parties. This very complex process for collecting, distributing, and notifying a diverse group of users on multiple platforms and security levels made the role of the OCA officer very difficult and labor intensive.

Although the OCA processes were complex, the actual tasks were mostly administrative in nature (file transfers, emails, status updates, verification of transfers, logging, etc), so the Flight Planning branch believed automation of the more simple tasks would go a long way toward reducing the OCA officer's workload.

7. OCA Automation Project

In 2005-2006, through a joint JSC and ARC effort, the ARC MODAT (Mission Operations Design and Analysis Toolkit) had been identified as a possible solution for improving console productivity by helping automate many of the flight control positions in the MCC. The MODAT solution employs the Brahms multi-agent language developed at ARC in the Work Systems Design and Evaluation group to automatically execute tasks based on a wide range of predefined criteria or parameters [ref Brahms, MODAT]. By deploying Brahms Agents to automate OCA tasks and leverage network infrastructure to automatically distribute information or provide communication, the ARC team believed they could automate a significant amount of the OCA Officers role.

The ARC MODAT teams approach was to utilize the Brahms agent technology to simulate both the current and future environments to verify the future state and demonstrate gains or proposed solution outcomes. To do this, the ARC team invested significant effort into auditing the OCA console operations through meetings and real-time observations to document the current OCA Officer Roles and responsibilities, along with the complex nature of the OCA network and the collection and distribution of information. This included aspects beyond the standard written procedures that the OCA Officer was executing, but truly using ethnography to document the OCA environment and exactly what the OCA officer had to deal with during each shift.

By accurately baseling the current OCA environment and OCA officer's tasks, the team was able to build a detailed simulation of the OCA world. For the OCA project the ARC team utilized actual data from OCA ISS operations to simulate the agents' tasks. The ARC team was able then to use this current baseline to implement changes and determine the best future state for the automation of the OCA environment. Once the future state simulations are determined to meet the goals of the task automation, the simulation agents can then be used to build operational solutions. This allows for streamlined transition from requirements definition and system design straight to development and building a valid solution for testing and deployment [ref ESAW07].

The ARC and OCA teams decided to phase in the automated solutions to the OCA environment to both test the new technology and allow the teams time to assimilate the use of automation in support of operations. The teams worked together and identified the tasks of updating the ISS Mirror LAN located in the MCC to be the best target first set of tasks to automate. The ISS Mirror LAN is a set of ISS laptops located in the MCC to literally "mirror" the ISS Ops LAN onboard. All actions taken to the SSC's onboard ISS OpsLAN are then applied to the SSC's on the ISS Mirror LAN on the ground. This keeps the SSC's and two LANs in synch and permits the FCT to troubleshoot issues on the ground without impacting the onboard OpsLAN. At this time the ARC OCA automation solution was named OCA Mirroring System or "OCAMS".

In July of 2008, OCAMS 1.0, the first implementation of the OCA agent-based system, was deployed into ISS operations, automating the OCA Mirroring process after each uplink/downlink session with ISS. The initial deployment of the agents was limited to a separate OCAMS PC/Server called the Mirror LAN Staging Machine (MSM) residing on the OCA LAN. This was done because the OCA PCs were older and near an equipment replacement (ER) cycle. Adding the OCAMS software to the existing OCA PCs was not viable option due to their age and limited processing capability of the units. For this reason the OCAMS team decided to phase in the distribution of the OCAMS agents to the OCA PC's and other MCC networks after the OCA ER was complete. This also had two side benefits 1) this allowed the teams to isolate OCAMS if required during a troubleshooting session in case it was suspected of causing issue with the OCA environment. 2) this also appeased the MCC security personal and the operations management teams that were still skeptical of the role automaton could play in the MCC operations environment. The automation of the mirroring tasks proved very valuable to the console position and saw 100% adoption by the OCA officers within a few weeks of deployment.

In March of 2009, OCAMS release 2.0 successfully went into ISS operations. This new release distributed the agents on multiple computers and added additional capabilities for the OCA officer via automated archiving and deletion of files; it included a prototype of the OCAMS Rule editor, which allowed OCA Officer to change and update rules used to process files. This was a significant first step towards allowing the OCA Officer and ops team to manage the changes to the OCAMS rules setup inside the software. Without this interface, changes would have to come through the OCAMS developers, which would extremely limit the flexibility of the system and likely mean it would become unusable to the OCA team. With the expansion of the OCAMS role on the console, the OCAMS acronym was reinterpreted as the "OCA Management System."

In September 2009, OCAMS release 3.0 was implemented into ops and provided automated delivery of products to OCA customers, as well as automated notifications via email and updates to the EFN system indicating the status of the requested activity (e.g. indications that a file was successfully uplinked or downlinked). The integration with EFN via an API established a key interface for the primary OCA customers. The EFN tool is the main interface where an ISS Flight Controller begins most requests for data or starts a formal discussion of technical issues with the rest of the FCT. By making OCAMS work with the EFN tool in the same way that the message automation/JEDI solution also utilizes EFN as the user front end, the team has kept the user training impacts to a minimum and allowed for a smooth adaptation of these solutions into daily operations. With the delivery of 3.0, the OCAMS software had automated approximately 80-90% of OCA Officers' ground-related tasks. With the 3.0 version, the OCAMS software is analyzing all the file traffic sent to and from ISS today to determine the correct actions required for processing. OCAMS does not take actions on every file, but does assess on average 2,300 files (18 GB) for uplink/week and 7,600 files (50 GB) for downlink/week.

The release of OCAMS 4.0 was deployed into operations in the summer of 2011 and automated most of the onboard tasks performed remotely by the OCA Officer. This includes completing uplinking and downlinking files automatically as they are requested by the users, and approved by the Flight Director and any other required MCC

discipline. The uplink and downlink is accomplished by leveraging proprietary software recently developed for the ISS program specifically for transferring files over the ISS data link called "Software to Ready Data Files to Send Hastily" or SWRDFSH. The OCAMS and SWRDFSH development teams worked closely together to optimize the interface between these two tools. With the 4.0 release, an OCA customer submits an EFN request to OCAMS, completing discipline-specific templates, which OCAMS uses to validate against a library of pre-defined procedures. Each procedure specifies allowable operations (e.g., Uplink, Delete File), directory paths, and file name patterns. For example, a procedure may specify uplinking a command file to control an onboard camera and then downlinking resulting photographs and logs. Procedures specify who may submit a request and what approvals are required to execute the request. A follow-on release of OCAMS (4.1) at the end of Fiscal Year 2011 is planned to make updates to the Integrated Procedure/Rule Editor to include Handover Log rules along with a more powerful procedure language and execution process. This will allow the ground teams to more easily establish automated procedure operations based on OCA requests.

8. OCA Automation Technology

In summary, OCAMS consists of a set of distributed, co-operating agents (or actors), which are designed to perform specific tasks, like archiving, generating email messages, processing procedures, or staging files for mirroring. These agents can be reused, for example, an email agent can be used in multiple locations if required to generate different emails. The agents use a variety of attributes to "match" or trigger actions to take on a certain file. These can range from operation type (uplink, downlink, copy, move, etc.) to file name or extension to things like source directory (the drop box location or SSC location onboard).

The OCAMS agents deployed in the MCC to support OCA operations are written in the Java language. To maximize performance, the Brahms logic was compiled down into Java for OCAMS Release 4.0 and above. OCAMS also uses the ARC-developed Collaborative Infrastructure (CI) for inter-agent communication. Agents communicate using structured messages (CommunicativeActs) based on the FIPA specification. With OCAMS Release 4.0, agents will use the open source spring layered Java/J2EE application framework. This improves agent/application design and decouples component implementations used by agents. This also allows for configuring an agent's components to enable it to provide its specific services (e.g. File Service, Archiving Service, User Interface, Hibernate). OCAMS persistence is managed using Spring and the open source Hibernate object-relational mapping (ORM). This abstracts away the interface to the database (SQL) and provides the ability to change DBMS without changing application code

The implementation of OCAMS in the MCC will allow the MOD Flight Planning branch to eliminate the dedicated OCA console position in 2012. Any OCAMS administrative tasks that need to be completed will be executed by other planning team members on console or managed via OCA representatives in the office. By removing one 7x24 console position, this project will reduce the staffing needs for ISS planning by multiple EP.

E. Mission Planning Automation

9. Planning Overview

Since the early days of human space flight, the Flight Planning organization has managed the development of the crew mission schedules or "Flight Plans" for execution of daily tasks on while humans are in orbit. These integrated flight or mission plans are the result of months of meetings, communications and collaboration to combine inputs from a wide variety of areas including: vehicle systems, payload and science, crew and life sciences, and program managers. The flight plans used for ISS and Space Shuttle missions today have evolved over the years to include all aspects of the mission, from crew and ground activities, to robotics and automated sequences completed by ground and onboard systems. With the ever increasing complexity of the human space flight vehicle and ground systems, these plans require a higher level of integration between the systems, flight control team and the astronaut crew onboard than ever before.

Even though each mission plan details are coordinated with all subject matter experts and updated frequently starting many months from execution; a consistent amount of change occurs as we approach the actual execution of the mission. For human spaceflight, and in particular ISS, the politics and diversity of key stakeholders involved, along with the limited resources and the complexity of operations, will continue to drive plan changes all the way up to the day of execution. With that reality in mind, the Flight Planning Branch knew that in order to reduce ISS console operating costs for planning, the process for managing these plan changes had to be streamlined and

automated as much as possible. The team realized that not only did the process need to be more efficient, but the planning teams' capability to effectively manage large volumes of change had to increase.

10. Next Generation Planning System (NGPS)

For this reason, in 2006-2007 the Flight Planning branch launched a project to develop a Next Generation Planning system (NGPS). This would replace the legacy planning systems utilized around the globe by all ISS International Partner (IP) planners. This system would be a collaborative project between several NASA centers that included ARC, Jet Propulsion Laboratory (JPL) and JSC Flight Planning branch. The impetus for developing this new system was in several areas, including:

1) To gain the efficiencies of a new consolidated system by replacing the combined 15 legacy planning applications that are used to generate flight plans today.
2) Develop a planning system that could leverage new services capabilities that allow data to be shared affectively between applications, and eliminates cumbersome manual transfer of data used with legacy tools.
3) Gain new capability and flexibility by leveraging the advanced planning the tools developed by ARC for use on the previous Jet Propulsion Laboratory JPL planetary missions (e.g. MER, Phoenix missions).

As the NGPS project was initiated, it was determined that the only feasible way to transition the global planning community to a new system was to make the NGPS tool suite compatible with the primary legacy systems used on ISS today. This included compatibility with several of the legacy planning system called CPS – Consolidated Planning System, developed in the early 1990's, along with the short range plan execution web based application called OSTPV – Onboard Short Term Plan Viewer developed in 2001, as well as the current Change Management (CM) tool utilized globally for ISS plan changes called PPCR – Planning Products Change Request developed in 2002. The CPS tool is used by the global planning teams for long range planning and integration, starting several months from execution. The OSTPV tool is also utilized by the global flight control team and the astronaut crew onboard ISS for short range planning and execution of daily activities. Both of these applications would be replaced by the NGPS suite of tools within 5 years (2014). By making NGPS compatible with CPS, OSPTV and PPCR, it would allow for a smooth transition to the new planning world for the international planning community, but would also drive additional work in to the NGPS project and even the legacy tools.

11. Change Management Integration Across Planning Tools

One of the key compatibility pieces for NGPS and the legacy planning systems would be the integration with PPCR, the planning CM tool for ISS. PPCR is a complex web-based tool that has been in use for ISS since the early 2000's. It is utilized by the global ISS FCT to submit changes and updates to the ISS plan. PPCR was upgraded in 2006 to a new modern web C#.net application so that it could benefit from the latest web services and data sharing technologies being developed in industry. As the NGPS project began, the team realized the huge benefits that would be gained by having NGPS and PPCR highly integrated and sharing planning data.

To this point in the ISS planning world, the plan changes were collected and managed separately from the legacy planning systems. Once a change was approved, the team would manually transition the approved change into the planning tools. The goal was to have the tools share this approved data and automatically update the planning system with this updated data. But, only applying this new process to the new tool, NGPS, meant the benefits for the FCT wouldn't be realized for years. Since PPCR would be the CM tool for both the new and legacy planning tools, the decision was made to integrate PPCR with the legacy planning tools as a first step toward overall planning process modernization. The goal was to have all the planning tools share data with PPCR and each other, allowing for seamless integration of plan information. The high level architecture for this concept was baselined, and the required changes to PPCR, NGPS, CPS and OSTPV to allow each to share plan data and improve console operation were outlined.

12. Legacy PPCR Process

The ISS plans are placed under formal CM at approximately 7 days from real-time execution. From that point, changes are required to be reviewed and approved by the planning team, lead by the Ops Plan console with the Flight Director and FCT in the MCC. There are some exceptions to this rule, but for the most part all changes require a PPCR after this point in development.

Approved PPCRs are processed for the 7 day out plan, known as the Weekly Look-ahead Plan (WLP) into the CPS system. Each day of the WLP is broken into a Short Term Plan (STP) as the team works daily changes to the WLP. At approximately 3 days from execution the official plan management is transitioned from the CPS tool to the OSTPV tool for the real-time console teams to manage last minute changes and then finally execute. But, due to the processes followed by the global planning teams that also utilize CPS, the MCC planners must keep both OSTPV and CPS up to date all the way through execution of the plan. This means the changes identified in each PPCR are made to both CPS and OSPTV as you proceed through real-time execution. Double work for each change.

Up until the planning automation project, these changes were manually entered into these tools. The tools did allow for users to make changes in CPS and generate a file to update OSTPV which helped; but, the problem came because certain changes in preparation of execution could only be done in OSPTV (e.g. coloring of activities, attachments, etc) so if a new CPS file was generated, it would overwrite the recent changes made in OSPTV. This lead to manually updating both tools to avoid overwriting of data. Also, all initial PPCR inputs were generated from scratch by flight controllers looking at OSTPV and manually typing changes into PPCR. This lead to many mistakes and issues with proper syntax and nomenclature on the PPCR inputs that had to be detected and corrected by the planning teams during review cycles. To summarize this process,: 1) the MCC planning teams would work hard to make each PPCR or change perfect, 2) then get it reviewed by the global FCT, make updates as required, then 3) once the change was approved, the planning team would have to manually put the data in one or both legacy planning systems, 4) and the plan was reviewed again to confirm the changes. Keep in mind that as many as 100 changes might be submitted the day before execution, and it's up to the planning team to implement all the changes approved by the FCT. This was obviously a very inefficient and labor intensive process.

13. The new Automated Planning Process

As part of the planning automation improvements, the planning tools team developed an architecture where every tool could consume or produce planning data via an XML file with a defined schema. In order to make this architecture possible, each planning tool had to be able to recognize a unique identifier assigned to each plan activity. This would allow a tool to receive inputs, integrate the changes, and make new changes and export the information back to the other tools. Without this unique identifier, the systems would choke on data not native to their own system, and in turn not be able to share data with other planning tools. This also meant each tool had to be capable of generating this unique identifier so it could independently create new activities when necessary. After many hours of design and discussion, a process and architecture to apply a unique planning identifier to each was developed. One by one, the planning tools were modified to be able to manage receiving as well as producing these identities and generating the XML file.

This has had a significant impact on the world of ISS planning. It has not only increased the planning organizations ability to handle large volumes of planning changes, but it has improved the quality of the changes being submitted and reduced the effort required to make the changes are correctly implemented. These changes have also improved console efficiencies across the global ISS Control Centers for planners as well as all flight controllers.

For example, a PPCR user (flight controller) can now open a PPCR, request it to be auto populated by the OSTPV tool (previously only manually entered data was possible for PPCR). The user simply selects the activity from a read- only query of the actual approved OSPTV data to auto populate. The user makes the required changes (traditional CM From: and To: fields) and submits the change for review and approval by the FCT. The planning team reviews the PPCR (which is a higher quality input since it was from an activity already approved in OSTPV) and validates the changes are viable and can be integrated with the rest of the plan. The Flight Director and flight control team review and approve the PPCR. The planning team can now mark the PPCR approved and via selecting a button, can generate all changes for export to OSTVP and CPS. The changes are automatically applied to the plan for OSTPV and integrated quickly into CPS via another process. Little or no time is spent verifying inputs to the plan since they eliminated the chances for human error. This has taken what could literally be several minutes of work updating a plan manually for each PPCR approved (remember 20-30 changes per day are not unusual), to simply pushing a button for multiple PPCRs at one time This improved integration and automation has allowed the planning teams to focus on optimizing the ISS mission plans, versus just focusing on just getting the multitude of changes implemented.

The information being shared between PPCR tool and the legacy planning tools (CPS and OSTPV) is being accomplished via XML files. A standard planning XML schema was developed to include all of the required planning information to allow the tools to move this information from one tool to another. This required changes to all the tools in order to synchronize the use of this file format. The interaction between NGPS and PPCR is being done via a REST services interface. PPCR has an established a REST service capability that allows it to share data with new planning system, NGPS. In fact the integration between PPCR and the NGPS Score tool (the scheduling component of NGPS), has allowed development team to create a PPCR interface that is displayed inside the NPGS tool. Planners will be able to manage plan changes from one single standard interface vs. two separate tools.

As the NGPS components come on-line and replace CPS and OSTPV, we fully expect the high level of integration we have been able to produce between PPCR and NGPS. In fact the goal is to take advantage of a new service-oriented architecture being developed for the future MCC platform that will allow all tools to leverage these services between tools. This should allow console positions throughout the MCC to leverage this data sharing architecture to streamline operations and reduce the data management overhead that each console experiences today.

14. Implementation of Standards for Planning Integration

Where possible in this modernization process, COTS oriented software design processes and technologies were used to develop and integrate these web tools, PPCR for example utilizes Microsoft's Entity Framework Object Relational (O/R) Modeling to abstract business objects above the relational database layer. On top of this model, a series of restful services were added that allows direct integration with the future NGPS suite of tools.

For inter-tool communication, a new standard for ISS web tools Web Services were developed, which utilized some of the latest trends in service architecture including the use of a RESTfully oriented message query model, using Microsoft's open source Open Data (ODATA) modeling and REST interfaces for PPCR and OSTPV. One benefit of using standard RESTful models was the limited ramp up time and quick development cycle to get these services online. This allowed one or two developers to quickly stand up customized service interfaces to meet the needs of new tools and still maintain old legacy interfaces on multiple URLs. This approach will continue to pay off as other tools are modernized and can utilize this service architecture to maximize data sharing and reduce development costs for unique single tool solutions.

F. Summary

When the Flight Planning Branch first initiated these projects in 2006-2007, none of the branches planning tools affectively shared data and substantial amount of manual labor was being invested to move information from one specific tool or application to another. With the implementation of the JEDI Message Automation, the OCA automation, and PPCR/NGPS automation projects, the planning teams now have data moving seamlessly between several tools and has greatly improved console operators efficiency.

This is accomplished utilizing industry standards such as XML file transfers as well as the use of RESTful services and ODATA Web services, as well as OCA Agents that communicate using structured messages (CommunicativeActs) based on the IEEE Foundations of Intelligent Physical Agents (FIPA) specifications. These projects have also lead to modernization of several tools and laid the ground work for future increased use of web services capability between several of the mission support tools. Several other web- based tools are now standing up a web services architecture that will allow all the tools to share data seamlessly. This has helped spur many new ideas for improving processes and leveraging this collaborative environment in the MCC. These automation projects will likely pay off for several years in the future with increased flexibly and reduced manual tasks for many of the MCC console positions.

It is also worth noting that all these project were accomplished utilizing extremely close communication between the development organization and the operational or users organization. Although the projects utilized several software development methodologies (including classic Waterfall and Iterative development approaches), each team managed close communication between the developers, project leads and users. In the case of Message Automation and Planning Automation, the teams literally had "imbedded" programmers that were physically co-located in the ops organization during development. This allowed the development organization to gain and understanding of the user operations environment and concepts; allowing them to make educated decisions while developing code that normally would have slowed development (to ask questions) or worse yet, lead to bad decision made independently

which lead to the wrong implementation of a requirement. This approach lead to more productive and accurate releases of each software product as they worked through the requirement set with direct user involvement.

Together, these three projects will lead to the reduction of two 7x24 ISS console positions in the MCC, along with the reduction one console day shift position in the long range planning room for the Flight Planning branch. Because of the nature of staffing 7x24 positions, this equates to multiple equivalent persons (EP) that are no longer required to staff shifts in the MCC for the ISS. The exact total return on investment (ROI) for these projects is hard to gage since it includes several factors including: 1) costs incurred to develop the automated solutions, 2) the required sustaining costs for the automation software that was implemented, 3) the reduction in KSLOC and associated sustaining costs in the areas where modern code base was implemented, 4) reduction in console personnel and overall required EP through 2020, and 5) the overall console efficiencies gained by the planning team as well other flight control positions. A conservative estimate of the ROI for all three would be in the range $8-$10M of MOD operating costs reductions in support of ISS through 2020.

Acknowledgments

The Authors would like to thank the MOD, the Operations Division and the Flight Planning Branch management for leadership to challenge us to improve, the courage to invest in automation for real-time operations, and the patience and support of all the projects as they charged ahead, struggled at times, and pressed ahead to finally succeed.

The authors would like to thank ARC Intelligent Systems Division for their innovative solutions to operations issues and their complete dedication and team work displayed on the OCA and Planning automation projects to make them a complete success.

The authors would like to thank the programmers from the User Applications organization of the Flight Operations Development Contract (FDOC) for their expertise and commitment to the MOD organization to make the Message and Planning Automation projects a success.

References

MAA: Clancey, W. J., Lowry, M., Nado, R., Sierhuis, M. 2011. **Software Productivity of Field Experiments Using the Mobile Agents Open Architecture with Workflow Interoperability,** IEEE Space Mission Challenges for Information Technology, August 2011, Palo Alto

MA: Clancey, W.J., Sierhuis, M., Alena, R., Berrios, D., Dowding, J., Graham, J.S., Tyree, K.S., Hirsh, R.L., Garry, W.B., Semple, A., Buckingham Shum, S.J., Shadbolt, N. and Rupert, S. (2007). **Automating CapCom using Mobile Agents and robotic assistants**. NASA Technical Publication 2007-214554. Washington, D.C.

Brahms: Clancey, W.J., Sachs, P., Sierhuis, M., & van Hoof, R. (1998) **Brahms: simulating practice for work systems design. *Int. J. Human-Computer Studies*,** 49, 831-865.

MODAT: Sierhuis, M., W.J. Clancey, Seah, C., and Trimble, J., and Sims, M.H. (2003) **Multiagent modeling and simulation in human-robot mission operations work system design**, *Journal of Management Information Systems* 19(4) 85-128. (M.E. Sharpe,Inc. URL: http://www.stern.nyu.edu/jmis)

ESAW07: Clancey, W.J., Sierhuis, M., Seah, C., Buckley, C., Reynolds, F., Hall, T., Scott, M. (2008) **Multi-agent simulation to implementation: A practical engineering methodology for designing space flight operations**. In A. Artikis, G. O'Hare, K. Stathis, & G. Vouros (Eds.), *Engineering Societies in the Agents' World VIII.* Athens, Greece, October 2007. Lecture Notes in Computer Science Series, Volume 4870. Heidelberg Germany: Springer, pp. 108-123.