ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/281462825

Demonstrating Autonomous Mission Operations Onboard the International Space Station

CONFERENCE PAPER · AUGUST 2015

READS
34

11 AUTHORS, INCLUDING:

Jeremy Frank
NASA
103 PUBLICATIONS **1,785** CITATIONS
SEE PROFILE

David L. Iverson
NASA
37 PUBLICATIONS **158** CITATIONS
SEE PROFILE

Kara M. Pohlkamp
NASA
7 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Kerry McGuire
9 PUBLICATIONS **36** CITATIONS
SEE PROFILE

Available from: Jeremy Frank
Retrieved on: 28 March 2016

Demonstrating Autonomous Mission Operations Onboard the International Space Station

Jeremy D. Frank * David Iverson * Christopher Knight * Sriram Narasimhan *

Keith Swanson * Michael S. Scott *

May Windrem *

NASA Ames Research Center, Mail Stop N269-3, Moffett Field, California 94035-1000, U.S.A.

Kara M. Pohlkamp † Jeffery M. Mauldin † Kerry McGuire ‡

Haifa Moses ‡

NASA Johnson Space Center, 2101 NASA Parkway, Houston, TX 77058, U.S.A.

The NASA Autonomous Mission Operations (AMO) project conducted an experiment to turn over operation and management of selected International Space Station (ISS) systems to the on-board crew. ISS crews managed two different spacecraft systems: the Total Organic Carbon Analyzer (TOCA), a water quality analyzer, and Station Support Computers (SSC) laptops, which are non-critical crew computer systems. These systems were selected because they are representative of systems a future crew may need to operate autonomously during a deep space mission. The crew autonomously operated these systems, taking on mission operations functions traditionally performed by support teams on the ground, using new software tools that provide decision support algorithms for planning, monitoring and fault management, hardware schematics, system briefs, and data displays that are normally unavailable to the crew. The experiment lasted seven months, during which ISS crews managed TOCA and SSCs on 22 occasions. The AMO software processed data from TOCA and SSCs continuously during this seven month period. The combined performance of the software and crew achieved a 88% success rate on managing TOCA activity, the system for which ground-truth was available.

I. Introduction

For over 50 years, NASA's crewed missions have been confined to the Earth-Moon system, where speed-of-light communications delays between crew and ground are practically nonexistent. The close proximity of the crew to the Earth has enabled NASA to operate human space missions primarily from the Mission Control Center (MCC) on the ground. This ground-centered mode of operations, with a large, ground-based support team, has had several advantages: the on-board crew could be smaller, the vehicles could be simpler and lighter, and the mission performed for a lower cost.

NASA is now investigating future human spaceflight missions[a] that include a variety of Martian destinations and a range of Near Earth Asteroid (NEO) targets. These possibilities are summarized in Figure 1. The table shows the approximate distance between the destination and the Earth, where the control center will be located, and the one-way light-time delay between the destination and Earth.

*Intelligent Systems Division, NASA Ames Research Center, Mail Stop N269-3, Moffett Field, California 94035-1000, non-Member.

†Flight Operations Directorate, NASA Johnson Space Center, 2101 NASA Parkway, Houston, TX 77058, non-Member.

‡Human Health and Performance Directorate, NASA Johnson Space Center, 2101 NASA Parkway, Houston, TX 77058, non-Member.

[a]http://www.nasa.gov/sites/default/files/files/NextSTEP-EMC-Reference.pdf

$Destination$	$Distance\,(km)$	$Time\,delay\,(seconds)$
ISS	435	ϵ
$Lunar$	$38,400,000$	1.3
$NEOs(close)$	$variable$	$variable$
$Mars(close)$	$545,000,000$	181.6
$Mars(opposition)$	$4,013,000,000$	1337.6

Figure 1. Destinations.

As is evident from Figure 1, missions beyond the Moon will be of much longer duration, and put crews much further from Earth, than today's missions. Accordingly, NASA has recently funded a number of projects[1] to develop and test operations concepts for these future missions. Of significant importance is the balance between crew autonomy and vehicle automation. Future crews need both the authority to make decisions without inefficient communication back and forth with ground-based mission control. They will also need sufficient information and vehicle capability to make, and implement, those decisions. However, small crews cannot take on all functions performed by ground today, and so vehicles must also be more automated in order to reduce the number of tasks that crews are responsible for performing.

Previous work[2] has been done to evaluate autonomy concepts in terrestrial simulations; additional work[3] has been performed to develop best practices in creating human spaceflight procedures for crews to perform with little or no assistance from ground. However, to date, no experiments have been conducted in which a spacecraft crew has autonomously managed complex space systems with no assistance from ground. Terrestrial simulations are insufficient to advance the state of the art; the transition of actual mission operations tasks, respecting the constraints of flight hardware, and integration of novel technology with operational spacecraft, are all needed to demonstrate transition of responsibility to onboard crew.

The NASA AMO project conducted an experiment to turn over operation and management of selected ISS systems to the on-board crew. The systems selected spans two types of ISS hardware: the Total Organic Carbon Analyzer (TOCA), a water quality analyzer, and Station Support Computer (SSC) systems, non-critical crew computer systems. These systems were selected because they are representative of systems a future crew may need to operate autonomously during a deep space mission. The crew autonomously operated these systems, taking on mission operations functions traditionally performed by ground. They did so with the aid of new software tools that provide decision support algorithms for planning, monitoring and fault management, hardware schematics, as well as system briefs, and data displays that are normally unavailable to the crew. The resulting experiment lasted seven months, during which ISS crews managed TOCA and SSCs on 22 occasions. The AMO software processed data from TOCA and SSCs continuously during this seven month period. The combined performance of the software and crew achieved a 88% success rate on managing TOCA activity, the system for which ground-truth was available.

This paper will describe the design and performance of the software used during the experiment. The paper is organized as follows. Section II describes the experiment concept of operations and deployment environment onboard ISS. Section III describes the AMO software design and components in more detail. Section IV describes software integration and the behavior of the AMO software during different operating scenarios. Section V describes the AMO ground system and how it mirrors the behavior of the on-orbit software. Section VI describes the operational complexity of the experiment. Section VII describes the quantitative results of the on-orbit experiment, both in terms of how the ISS crew used the software, and in terms of rating the correctness of both the software and crew with respect to ground-truth results for TOCA. This section also contains lessons learned for the design of similar systems to enable autonomy. Finally, we conclude and discuss potential future work.

II. Autonomous Operations Hardware and Concept of Operations

In this section we describe the experiment in more detail. We first describe the concept of operations and crew responsibilities. We then describe the ISS subsystems operated by the ISS crew. Finally, we describe the environment in which the decision support software used by the crew is deployed onboard the ISS, and its principle interfaces to other ISS on-board software systems.

II.A. Crew Autonomous Operations Activities

Each week, the crew used the AMO software to provide the ground with a recommendation of the TOCA activities required for the next planning cycle (two weeks out). The software kept track of activities required by time or use counts (some activities are required every month, some were required every X runs, etc.). The crew had the ability to override software recommendations, not request activities, and request other activities not recommended by the software, as required. Crew recommendations were compared to the activity requests provide by flight controllers to determine if the crew's recommendations were correct.

After each TOCA analysis, the crew used the AMO software to determine if TOCA performed nominally, and if the Total Organic Carbon (TOC) in the water was within or out of trend. If the crew had a question regarding TOCA they were asked to first consult AMO software to see if they can answer their own question prior to calling the ground. The AMO software includes technical references, schematics, and just-in-time training to assist the crew with evaluating TOCA performance. If either water quality or TOCA hardware performance was deemed off-nominal, the crew was asked to use AMO software to provide their next step recommendation. AMO software provides its own evaluation of the above questions to aid the crew.

The AMO software also collects and analyzes performance parameters from each SSC onboard, and flags off-nominal performance. For detected off-nominal performance, the AMO software provides the crew with a troubleshooting recommendation. The crew was directed to look for SSC alerts when using the software for TOCA tasks, or to consult the software when SSC performance was off-nominal.

In the next sections, we describe these ISS systems and their management in more detail.

II.B. Total Organic Carbon Analyzer (TOCA)

The TOCA is used to analyze the crew's potable water supply. It measures the TOC in the water as a proxy for water quality. The TOC is distinguished from Total Inorganic Carbon (TIC) during the analysis phase. Water quality activities are performed either by the crew manually opening a valve to the water supply, or by filling a small bag with water from the 'sink', called the Potable Water Dispenser (PWD). A single water analysis activity takes approximately three hours, during which the TOC is measured three times. A single analysis comprises 22 intermediate processing states. One or two water quality activities are performed weekly. Water quality results are provided to the crew via a display on the TOCA hardware, but in-depth analysis both of water quality trends and TOCA hardware performance (fault and anomaly detection) is performed by MCC. TOCA management also requires scheduling of maintenance activities, including calibration and replacement of some components. TOCA produces hardware performance data for dozens of parameters at a rate of 1 Hz. The responsibility for managing the TOCA activity schedule is with ground, as is water quality analysis, TOCA hardware health assessment, and any fault isolation or recovery.

II.C. Station Support Computer Laptops

The ISS crew has access to 23 Lenovo T61P computers; these are referred to as SSC laptops. These laptops are used for a variety of purposes, including access to the crew's daily mission plan, operational procedures, non-critical operational tools, and a variety of crew personal uses (email and entertainment). Flight controllers on the ground are responsible for all network and computer management functions to maintain these computers, including routine software maintenance, performance monitoring, and trouble shooting.

II.D. Deployment Environment

The ISS deployment environment and major interfaces are shown in Figure 2. The ISS Operations LAN consists of a network of T61P laptop computers, Apple iPads, and other devices. One of these (LS1) runs a Debian Linux server. A number of Virtual Machine (VM)s act to serve various applications, one of which is the AMO software. AMO consists of a variety of components that provide data to Web clients via an Apache web service. Clients run on either T61P laptops running Windows 7, with the Internet Explorer web browser, or Apple iPads running the Safari web browser. Another VM hosts the free, off-the-shelf Icinga[b] network monitoring system for collecting SSC health and status. Icinga consists of collection and analysis services and a Postgres database. AMO indirectly interfaces with Samba[c], a set of "daemons" on LS1, which provides a number of network services.

[b] https://www.icinga.org/
[c] https://www.samba.org/

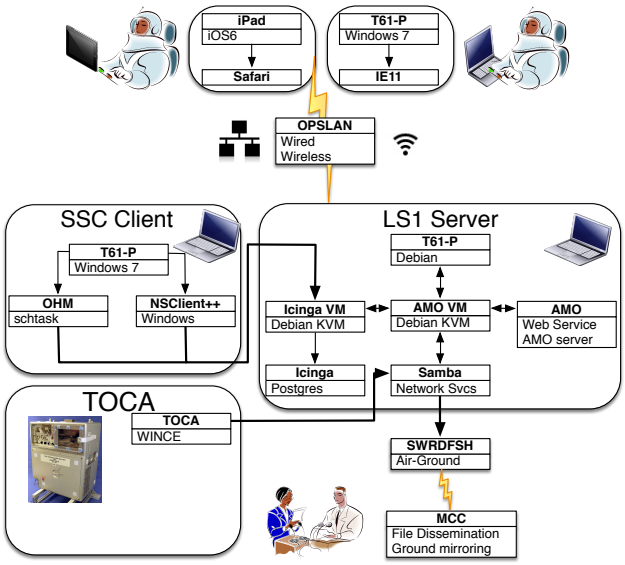


Figure 2. AMO Deployment Environment and Major ISS Interfaces

The TOCA hardware is connected to the Operations LAN via hardline ethernet to communicate files after each water processing run. AMO accesses these via the Samba fileshare services. SSC data is collected from multiple sources, including NSClient [d] and Open Hardware Monitor (OHM), an open-source tool for monitoring computer performance parameters (temperatures, disk speeds, etc) [e]. This data is polled by the Icinga software running in the Icinga VM on LS1; AMO, in turn, pulls data from the Icinga VM.

AMO both produces files to send to ground, and reads files sent to the ISS from ground during its operations. A more complete description of this interface are provided in Section V.

Finally, the AMO software has embedded web links to the International Procedure Viewer (IPV), an ISS Web-based application. This application runs on a second server computer, ISS-Server-1, and is not shown on Figure 2.

III. AMO Software Design and Architecture

The AMO VM consists of a variety of components that perform analysis and provide data to the User Interface (UI), via the Web service, in response to a variety of events. These events include user activity on the AMO UI via the Apache web service, the creation of files on the shared file system, or internal clocks. This architecture is described in more detail in Section III. The components perform analysis functions such as anomaly detection fault detection, scheduling, monitoring limits, pulling data, and writing logs. Each component is described in more detail below.

III.A. The User Interface

The AMO User Interface (UI) is implemented as a single-page application presented to the user via a Web Browser. The elements of the UI each serve one of the functions described in section II. In the following sections we describe the server elements that produce dynamically generated content for the UI, or generate or react to files.

The AMO UI is a Web application. As a result, the UI benefits extensively from the use of HTML links. These links allow the user to rapidly navigate from one function within the AMO UI to another. The AMO UI is driven by a number of static and dynamically generated files. The static files are HTML; the

[d] http://www.nsclient.org/
[e] http://openhardwaremonitor.org/

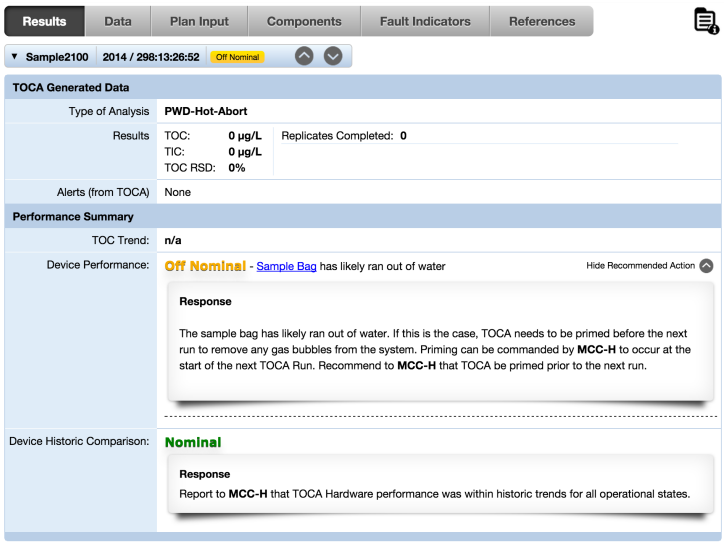


Figure 3. TOCA UI screenshot, showing the main UI function tabs across the top, and the Results of an off-nominal sample showing a fault and recommended actions.

dynamic content is JSON formatted, and is designed to be read by specific parts of the UI. Each JSON file is produced by server components described in the next section. Early in the design process, it was clear that the AMO UI must be designed to be presented either on a T61P or an iPad. The physical form factor of each device is different, and the iPad can be reoriented to portrait or landscape mode. The touchpad interface on the iPad presented design challenges for the layout and usability of the UI. Finally, factors such as the amount and complexity of data to present, font size and color constraints, and number of required operations (menus, scrolling, etc) desired to access information, all influenced the final design. Extensive user studies were invaluable in honing in on the final design.

III.A.1. TOCA UI

The TOCA UI contains functions that present data to the crew, allowing them to manage TOCA. Each of these functions are accessible via 'tabs'; the tabs are organized hierarchically.

The Results tab presents the results of each TOCA analysis activity. A quick-look pane shows the summary results of each sample, any hardware generated fault messages, and also shows nominal and off-nominal status for TOC, device performance (faults), and device historic comparison (anomalies). Each sample is deemed Nominal if each of the above assessments reports nominal, and Off-nominal otherwise. Water quality is determined to be off-nominal if the TOC trend, as measured over consecutive samples, is out of a predefined range; this range is changed periodically based on expert judgement of TOCA engineers and flight controllers. If the TOC is above the potable limit, any of the three TOC measurements are above the potable limit, or if the relative standard deviation of the samples is too high, this is noted for the sample. Faults or anomalies are detected from a combination of TOCA software fault messages and the output of AMO server components. If any element of a TOCA analysis activity is deemed off-nominal, the UI presents one or more Recommended Actions. Examples of all of these features are shown in Figure 3. These actions provide descriptions of the steps the crew can take to either identify a fault, or understand an off-nominal result. Entries in the Results tab can be sorted by date, sample type, sample number, and nominal-off-nominal result.

The Data tab presents data plots showing accumulated results over all uses of the TOCA hardware (Trend), as well as data for individual TOCA analysis activities (Per-Run). TOC measurements from each sample are grouped into series based on the type of analysis performed (Hose, Hot PWD, Ambient PWD, and Calibration Check). Error bars are provided for the Calibration Check series. The TOCA device cannot detect carbon below a certain threshold, and the carbon content cannot exceed a threshold for the water to be safe to drink; these limits can be visualized on the trend plots. The TOCA processing states can be visualized on the Per-Run plots. All plots have a zoom-in zoom-out and plot overview feature. Where applicable, sensor readings are plotted with a nominal performance region overlay, showing the historical

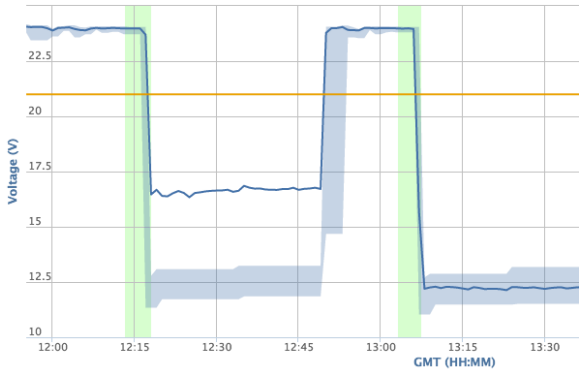


Figure 4. TOCA data plots. Nominal Performance Regions are shown in blue.

behavior of these sensor values throughout a run; this allows easy visual determination of deviations from the normal region over time. Figure 4 shows a plot with the nominal performance region shown in blue. The remaining per-run plots have upper and lower limits beyond which software fault messages or hardware faults are generated; these are shown on the plots. All plots also have a description of how the plot should look if TOCA is behaving as expected. Figure 5 shows a list of data from TOCA that is provided to the crew in plot form.

Parameter	Per-Run/Trend
TOC	Trend
Liquid Flow Average (React TOC)	Trend
Oxidizer Voltage	Trend
Liquid Flow Rate	Per-Run
Maximum Oxidizer Voltage	Per-Run
Minimum Oxidizer Voltage	Per-Run
Oxidizer Voltage	Per-Run
Temperatures (9)	Per-Run
Pressures (4)	Per-Run

Figure 5. TOCA Data presented to crew in plot form.

The Plan Input tab allows crew to either request the scheduling of tasks that are recommended by the scheduler, or manually recommend tasks. The time horizon for the plan is three weeks, including the current week. The Plan Input tab shows, for each task, its status (Recommended, Requested, Scheduled, Completed), a link to the procedure for each task, a rationale explaining why each task is requested, a user-editable crew note, and action buttons allowing crew to take actions based on the task status. Recommended tasks can be Requested; Requested tasks can be Canceled. In addition an Activity History button shows all previously performed tasks; the activity history can be sorted by activity type, procedure, sample number, due date and execution date.

The Components tab includes schematics and photos showing TOCA with and without the exterior panel removed. In addition, a photo and short functional description of each major part of TOCA is provided. Each part has a closeup photo and a description of the part. Due to the small screen size, each schematic has a magnifier allowing the user to zoom in on the schematic. The Components tab also includes a description of the TOCA processing states, and shows what elements of TOCA are active during those states.

The Fault Indicators tab allows the crew to identify TOCA hardware faults. The hardware faults trip in the event TOCA experiences a fault that causes it to power down. There are four fault indicators forming a 4-bit fault pattern. The crew can use the Fault Indicator Tab to replicate the fault pattern and see what condition may have caused this problem.

The References tab is a searchable list of each TOCA procedure, component name, processing phase, hardware and software fault messages, plot descriptions, and links to instructional videos.

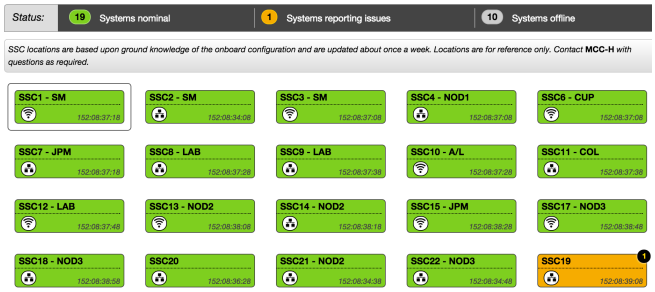


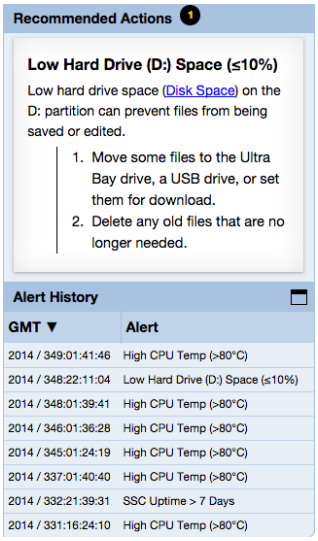
Figure 6. The SSC Overview page showing 20 active SSCs. One SSC (19) is reporting one problem, shown in yellow.

III.A.2. SSC UI

As with the TOCA UI, the SSC UI contains functions to allow the crew to manage SSCs. The SSC UI is organized in a similar manner to the TOCA UI.

Figure 7 shows the performance parameters collected and analyzed for each SSC, and limits used to determine if an SSC is off-nominal. The Overview tab shows the status of each of the 23 SSCs onboard ISS at a glance. As with TOCA results, each SSC is either deemed Nominal or Off-Nominal. Off-nominal SSCs are colored yellow in the Overview tab, and nominal ones are colored green; the color of an SSC indicates whether the last retrieved data shows off-nominal behavior. An SSC can also be considered Offline if it has not reported three or more of the parameters for a period of time. An icon also indicates whether an SSC is on the wired, wireless or both networks. If an SSC is considered off-nominal, the number of problems is shown in the upper right corner. The date and time of the last reported state is also shown. SSCs displayed in the Overview tab can be sorted by their name, location onboard ISS, last update, or their status. The Overview tab is shown in Figure 6.

The Details tab shows more information about the state of each SSC. Each SSC reports network latency, uptime, hard drive space usage, memory usage, CPU utilization, and temperatures for each CPU core and the hard drive. Each of these parameters has a nominal performance limit, shown in Figure 7; if these limits are exceeded, an Alert is generated. A list of historical alerts is shown to the right. Figure 7 shows examples of several SSC alerts as they appear in the UI. If the SSC's current state triggered an alert, a list of recommended actions is also provided; one such recommendation is also shown in Figure 7. The crew can navigate to different SSCs using a drop down menu or arrows.



Parameter	Limit
Uptime	7 days
Latency	500 ms
CPU Util.	100%
Memory	80 %
Hard Drive	90 %
CPU Temp	80 °C
HD Temp	60 °C

Figure 7. SSC alerts and SSC thresholds used to generate alerts.

The Data tab allows the crew to drill down into plots of each of the parameters. As with TOCA, the plots allow zooming, have an overview, and a description of how the plot should look if SSCs are behaving as expected. AMO stores 5 weeks of SSC data for presentation to the crew. The threshold for nominal

performance is shown on each plot.

The References tab contains a searchable list of AMO procedures, recommended actions, plot descriptions, definitions, and help videos.

III.A.3. Help UI

The Help UI contains a quick-reference menu on the right. This menu is divided into Videos, TOCA help, and SSC help. Five videos are provided; one AMO overview video, and four videos describing various TOCA operations. Links are provided to short overviews of the principle TOCA and SSC management functions described above.

III.B. SSC Data Collection

The ISS program uses NSClient to collect network latency, uptime, hard drive space usage, memory usage and CPU utilization data. While the ISS program analyzes this data on the ground, AMO performs a simpler analysis onboard to present a simpler form of SSC state and recommendations to the crew. The OHM software was deployed on each SSC to collect additional information about SSC performance, specifically CPU core and hard drive temperatures. Each parameter is collected once every 10 minutes ($\frac{1}{600}$ Hz). As mentioned previously, this data is processed and stored in Icinga onboard, and then pulled for processing by AMO.

III.C. Air to Ground Link

AMO sends and receives files over the Air-Ground link using a NASA developed system called SWRDFSH. There are three main uses of the air-ground link: schedule synchrony, AMO log downlinks, and AMO configuration uplinks. When the crew decides to either manually add a task or recommend a task, this information is recorded and sent to the ground. Since TOCA is a critical ISS system, TOCA activities are scheduled by flight controllers on the ground. The TOCA specific activities are extracted from the Short Term Plan, the crew's daily activity plan, and are uplinked daily to ISS. This ensures that scheduled TOCA activities are accurately reflected in the AMO UI Plan Input tab. In addition, all AMO UI usage is logged for analysis; these logs are sent to ground as well. This downlink is performed three times daily. Finally, the AMO UI Configuration file allows extensive reconfiguration of the AMO system. Examples of configuration changes include the TOCA and SSC thresholds and Scheduler frequencies. This file is uplinked on an as-needed basis.

III.D. The AMO Server Components

The AMO UI displays a variety of dynamically generated data produced by AMO server components. These components interact with the files and data produced by SSCs and TOCA, as well as responding to data updates (either schedules or configuration file changes) from the ground delivered over the Air-Ground link. The AMO server also generates a number of data products that are delivered to ground for display and analysis purposes. This section describes the principle software functions of the AMO server. The server architecture is shown in Figure 8. The functional components of the server are described in the following sections.

III.D.1. Monitor

The AMO Monitor watches the shared file space for new TOCA result files. When new files are detected, the AMO Monitor invokes the Anomaly Detection, Fault Detection and JSON Writer functions to process the TOCA files. The resulting data files are written to populate the TOCA Results and Data tabs. The Monitor retrieves SSC data once every 60 minutes by invoking the AMO SSC Cache function. After retrieval, Limit Checks are performed, and again JSON Writer is called to generate data files to populate the SSC Overview, Details and Data tabs. Once a day, a new schedule is uplinked. The uplink occurs at varying times since it depends on when the authoritative ground schedule is made available. This file is retrieved by the Monitor, and subsequently processed by the Scheduler. Whenever a crew person interacts with AMO via the UI, user activity is logged for later analysis. These logs are recorded and downlinked daily; more information about the logs is presented later in the paper. Three times a day (0800, 1200 and 1600 GMT) AMO writes a consolidated log of all files that have changed since the last time logging was performed. These files are moved to the shared file space, and retrieved over the Air-Ground link. Each time the AMO Monitor performs these

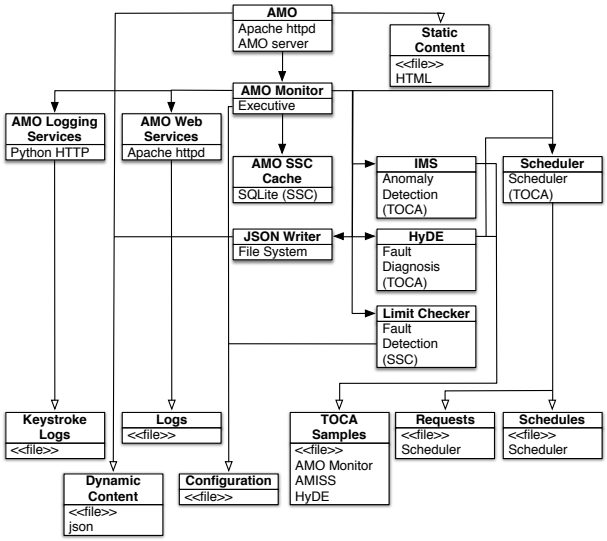


Figure 8. The AMO Server Architecture.

operations, it reads the AMO Configuration file anew. This way, any configuration changes are automatically reflected in the processing performed by any of the components, and the newly generated dynamic content for the UI. However, no retrospective processing of historical data is performed when configuration changes are made. AMO ensures that no two TOCA samples are processed at the same time. It ensures any new TOCA samples are processed, regardless of the sample number. It ensures TOCA and SSC processing is not performed at the same time. It also enforces timeouts to terminate the processing of jobs if they take too long.

Activity	Timing
SSC retrieval	60 minutes
Downlink log	0800,1200,1600 GMT
IMS Timeout	5 minutes
HyDE Timeout	5 minutes
JSON Writer timeout	5 minutes
SSC Processing timeout	30 minutes

Figure 9. Timing of AMO Monitor activities.

III.D.2. Scheduler

The Scheduler takes as input the constraints on the water sampling and maintenance activities, and the prior history of performed activities, and produces a three week TOCA sample and maintenance activity schedule. Constraints on maintenance activity are expressed either in calendar terms, or in terms of the number of uses of TOCA. The Hose sample activity is to be performed weekly; the Bag Sample is performed monthly, but actually consists of bi-monthly analyses of hot water and ambient water temperature samples. Some of the constraints result in activities scheduled roughly every month (e.g. the Waste water bag changeout), while others can be much longer (the Buffer container changeout ends up being an annual activity). The Calibration Check activity actually requires running a Hot and an Ambient samples, and therefore uses TOCA twice. Any instance in which TOCA is turned on but fails to complete an analysis for any reason counts as usage

when considering maintenance activities. Of all the activities, only the Calibrate is not regularly scheduled; a Calibrate is needed only if a Calibration Check fails. The Scheduler provides an interface for other AMO components to make recommendations based on their TOCA processing results. In the case of Calibration Check activities, the failed Cal-Check is detected by Hybrid Diagnosis Engine (HyDE), which instructs the Scheduler to recommend a Calibration activity.

A number of other considerations apply when determining the order in which recommended activities should be performed. TOCA has a duty cycle constraint; it can operate for at most five hours a day. This effectively constrains TOCA activities to one a day, since processing a sample can take as much as three hours. It is possible that multiple activities can be recommended by the Scheduler to occur on the same day. When this happens, the Scheduler will recommend the least frequently scheduled activity prior to the more frequently scheduled activities. Because of the different ways that frequency is expressed in the scheduling constraints (by weeks vs by number of uses of the TOCA), this rule is implemented by determining which recommended activity was actually performed longest ago. Also, recall that Calibration Checks require both a Hot and Ambient sample, and these checks must be scheduled on different days; the Ambient temperature activity is scheduled before the Hot activity. There is one remaining nuance to scheduling, which is how to treat either Requested activities (those a crew person has tentatively placed on the schedule), and Scheduled activities that have not yet been performed. These activities are treated as Completed for the purpose of looking ahead and recommending activities for the future weeks.

These constraints are described more concisely in Figure 10 below.

Activity	Type	Frequency
Hose Sample	Nominal	1 week
Bag Sample (Hot)	Nominal	2 months
Bag Sample (Ambient)	Nominal	2 months
Calibration Check	Maint	3 months
Waste Bag Changeout	Maint	6 Runs
Buffer Container changeout	Maint	47 Runs
Calibrate	Maint	Cal.Check failed

Figure 10. TOCA Activity Constraints.

The computational complexity of the resulting scheduling problem is linear in the number of prior activities. The algorithm works as follows:

1. Scan backwards through the activity history to find the latest execution date of each activity type with a calendar based activity frequency, d_a. Simultaneously count the number of TOCA usages u since the last Waste Bag Changeout and the last Buffer Container Changeout.

2. Let t be the date of the last day of the current week. For each calendar frequency based activity type let f_a be the execution frequency in Figure 10. For instance, if the activity a is Hose Sample, then f_a is 1 week. If $t - d_a \geq f_a$ then activity type a is recommended for the current week.

3. If the number of TOCA usages u exceeds the frequency of either a Waste Bag changeout or Buffer Container changeout, recommend these activities.

4. Once all activities for this week are known, order them such that the least frequently performed activities (according to Figure 10) are recommended first. Update the latest execution dates d_a of all activities recommended this week, and update u. These updates ensure activities recommended in a prior week are correctly accounted for when recommending activities for the next two weeks.

5. Repeat steps 2-4 for the next two weeks.

The Scheduler can determine the completion status of activities from TOCA log data. This can be done only for sampling activities. The Scheduler also determines the completion status from the daily plan updates. These may sometimes override the status in the TOCA log data.

Whenever the crew recommends or cancels a recommendation, or a new schedule is uploaded, the schedule is recomputed.

Status	Procedure	Rationale
This Week (GMT 2014/342 - 2014/348)		
Requested	ISTAR Total Organic Carbon Analyzer: TOCA - Waste Water Bag Changeout (Med Ops 6.3.350)	Required every 6 runs and prior to next run
Scheduled	ISTAR Total Organic Carbon Analyzer: TOCA - Water Sample Analysis Using TOCA Water Sample Hose (Med Ops 6.3.250)	Required weekly
Next Week (GMT 2014/349 - 2014/355)		
Requested	ISTAR Total Organic Carbon Analyzer: TOCA - Water Sample Analysis Using TOCA Water Sample Hose (Med Ops 6.3.250)	Required weekly
Future Week (GMT 2014/356 - 2014/362)		
Recommended	ISTAR Total Organic Carbon Analyzer: TOCA - Water Sample Analysis Using TOCA Water Sample Hose (Med Ops 6.3.250)	Required weekly
Recommended	ISTAR Total Organic Carbon Analyzer: TOCA - Water Sample Analysis From TOCA Sample Analysis Bag (Med Ops 6.3.300) (for Hot PWD)	Required monthly (alternating between the hot and ambient ports)

Figure 11. The TOCA Plan Input Tab, showing a combination of Scheduled, Recommended, and Requested activities.

Figure 11 shows the Plan Input tab. In this figure, there is a Waste Bag changeout, one Bag Sample activity, and three hose sample activities. The Waste Bag changeout activity is ordered first, since of the tasks in this three week window, it was last executed longest ago. The first Hose Sample activity is Scheduled, while the second has been Requested, and the third has been Recommended but not yet Requested. The Bag Sample activity is scheduled last, and has also been Recommended but not Requested. The crew can Request the two hose samples that are presently Recommended; when ground decides to schedule them, they will be displayed as Scheduled.

The Scheduler takes as input the current schedule of activities, and updates the Plan Input UI to show which activities are scheduled. Doing this requires some processing of the crew's Short Term Plan, an input to other ISS operational tools, to extract and interpret the specific activities we need for our purposes. For example, activity names in the plan are used to find TOCA related activities, but often it is the procedure name, which is referenced in the activity, that informs us of which specific TOCA activity is scheduled. To determine water temperature for PWD sampling we look for a separate water collection activity scheduled within 24 hours previous to the sampling activity. The water temperature is parsed from the Execution Notes field for the collection activity. To find Calibration activities, we look for two separate activities named Hi-Cal and Low-Cal scheduled on consecutive days.

III.D.3. Anomaly Detection

TOCA anomaly detection is performed using two algorithms. The Inductive Monitoring System (IMS) [4] is used to learn a model of the nominal performance of TOCA using recorded data gathered from prior runs. This process is referred to as 'clustering'. Different clusters are build for each TOCA processing state. At each instant or time step, the behavior of the system as represented by a set of parameters can be considered a vector of numbers. The Euclidean distance $\delta(v_1, v_2)$ on two vectors of length j is $\sqrt{\sum_j (v_1^j - v_2^j)^2}$. A cluster c is represented by c_h^j and c_l^j, the highest and lowest values respectively of parameter values of the vectors assigned to the cluster. Define $c_a^j = \frac{c_h^j - c_l^j}{2}$, and denote c_a as the vector of c_a^js. In order to build the clusters from device performance data, the following algorithm is used:

- Each vector v is first scaled and normalized.
- Suppose there are C clusters, the closest cluster $c' = \min_{c \in C} \delta(v, c_a)$ is identified.
- If $\delta(v, c) \leq \epsilon$ then v is added to c, and c_h and c_l and c_a are updated, otherwise a new cluster is created.

The value ϵ can be used to tune the number and sizes of the clusters.

Once the clusters are created, new instances of TOCA device behavior, denoted b, can be compared to the previously learned behavior, and the distances $\delta(b, c)$ are computed. In previous incarnations of IMS, $\min_{c \in C}(\delta(b, c))$ was computed, and compared to a limit; if the new vector b exceeded the limit, an anomaly was declared. The clusters IMS builds fit the TOCA data well, but distances vary considerably

with time, even within a processing state. First, the *weighted* average (based on the number of training instances in a cluster) of the distances to the n closest clusters in C is built for each training instance; denote this quantity $\delta(v, n_C)$. This weighted average distance is computed using the same training samples used to build the clusters. An adjunct to IMS, the Meta Monitoring System (MMS) system, uses a more robust means than a simple limit on the distance to detect anomalies. Denote the probability distribution of weighted average distances $\delta(v, n_C)$ for processing phase r by $p(\delta(v, n_C, r))$. Then the cumulative distribution $F(\delta(v, n_C, r)) = \sum_{\delta'(v,n_C,r) \leq \delta(v,n_C,r)} p(\delta'(v, n_C, r)$ is computed. As with building the IMS clusters, this can all be performed offline. For a new vector of system behavior b observed during processing phase s, $\delta(b, n, s)$ is computed. The quantity $F(\delta(b, n_C, s))$ is then determined from the appropriate cumulative distribution $F(\delta(v, n_C, r))$ with $s = r$ [f]. If this quantity exceeds a threshold (currently .975) a sufficiently large number of times, an anomaly is declared.

Recall that the TOCA hardware produces data at a rate of 1 Hz. The TOCA hardware processing time varies, but is typically under three hours. TOCA performs three independent measurements of the TOC, and hence some processing states are repeated three times. The duration of each state varies from a few seconds to tens of minutes. In discussions with the TOCA hardware engineers, the TOCA Liquid Loop was considered the most likely TOCA component to exhibit off-nominal behavior. Six TOCA performance parameters characterize Liquid Loop performance. Most, but not all, of the parameters were available for display on the Data plot. As noted above, TOCA has many processing states; separate IMS and MMS models are constructed for each processing state. Some processing states are repeated multiple times, but all data from those states is used to build the IMS and MMS models for that state. The data was created from 46 TOCA data analyses performed prior to the deployment of the software onboard ISS. The number of clusters per phase ranged from about 1300 to 15000.

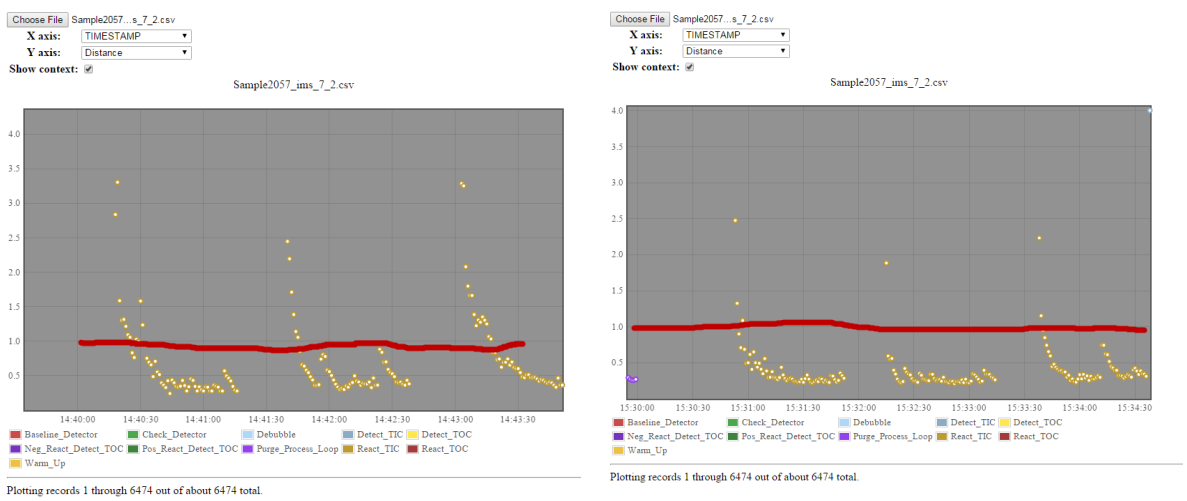


Figure 12. Anomalous behavior, as shown by weighted average distance, of ReactTIC Phase (left) vs nominal behavior (right). Gold points show the weighted average distance ofTOCA; the red line shows the threshold for anomalous behavior.

Figure 12 shows an example of IMS output. As described previously, IMS determines whether TOCA is behaving anomalously, and indicates the specific processing phase in which the anomaly was exhibited. In this case, 12 shows two examples of the weighted average distance $\delta(b, n, s)$ to the closest clusters of one phase (React TIC) of TOCA processing. In the figure on the left, the ReactTIC phase is behaving more anomalously; a large number of the measured distances (shown in Gold) are above the 'nominal' behavior distance (shown in red). By contrast, the right hand figure shows $\delta(b, n, s)$ during a nominal ReactTIC phase; there are significantly fewer data points whose distance exceeds the 'nominal' distance.

[f]In fact a weighted sum of the recently encountered percentiles is computed, but these details are less important than the essential idea.

III.D.4. Fault Diagnosis

HyDE is a model-based fault diagnosis engine that diagnoses discrete faults.[5] HyDE takes as input a model that describes the combined discrete and continuous behavior of the components of the system. The model consists of discrete modes of operations, relevant variables and parameters of the modes, and how these variables relate to each other in different modes of operations. HyDE uses sensor data from the system and the model of the system behavior to deduce the evolution of the state of the system over time, including changes in state indicative of faults. Transitions between modes are based on either commands or conditions on the internal variables. Faults are represented as special unknown event transitions. HyDE supports modeling of variables and relations in different domains like Boolean, enumeration, real-valued and interval-valued. HyDE's reasoning algorithm tries to determine the modes of all components, values of all variables, and the occurrence of one or more unknown events in the system. The reasoning starts with the assumption that no unknown events have occurred, and all modes and variables have their initial values. Any available sensor observations are compared against predictions from the model. Inconsistencies between predicted and observed states generates conflicts; each conflict is a set of unknown event transitions that contribute to each inconsistency. A search over the space of unknown events driven by these conflicts results in the generation of one or more fault states that would resolve the inconsistencies. The number of possible faults is configurable, which limits the possible explanations. Further simulation with these potential candidates would determine which potential candidates become actual diagnoses. This process continues over time as more and more observations become available.

AMO used a HyDE model of the TOCA Liquid Loop. All components (like Sample Bag, Hose, Valves, Oxidizer, etc.) have corresponding components in the HyDE model. Each sensor is also modeled as a component. Variable values in the model are simplified to represent nominal, high or low sensor measurements; these categories are derived from the TOCA fault tables, as described further below. The relations over these variables then correspond to how these variables are affected by the underlying physical relations. Modes of operation of components include nominal modes plus failure modes that were determined by an analysis of faults already exhibited by the system and the common failures for components in the system. The TOCA HyDE model includes a failure mode for each sensor and failure modes for Sample Bag Underfilled, Hose Valve closed, a failure mode each for the Oxidizer, Valves, Pump, and Waste Container, and two Chiller failures (high and low temperature). Figure 13 enumerates the TOCA faults HyDE can detect.

Fault Type	Number of Faults
Sensor Faults	27 of Faults
Valve Stuck	12 (6 valves, open/closed)
Sample Bag	1 (Underfilled)
Hose	1 (Closed Valve)
Waste Container	1 (Full)
Acidic Buffer Dispenser	1 (Faulty)
Sample Circ Pump	1 (Failed)
Oxidizer Reactor	1 (Faulty)
Chiller	2 (Under Chill/Over Chill)
Pipe	1 (Clogged)

Figure 13. TOCA Fault diagnoses.

The HyDE server component reads the TOCA data file, maps columns to corresponding HyDE variables and then converts to the values to the enumeration domain using the fault table file provided by TOCA experts. The fault table identifies whether the values are nominal, high or low in different modes of operation of TOCA. The TOCA logs also note the change of state of valves (open to closed); HyDE issues an appropriate command to the valve in the model in response. After data at each time step has been presented to HyDE, its reasoning is invoked to update the diagnosis. When the data generated by TOCA includes off-nominal values, HyDE uses the reasoning described earlier to diagnose faults. Typically, when only one value is off-nominal, the corresponding sensor is considered to be suspect. This can be achieved by tuning HyDE to pick sensor faults over single component faults first. If more than one sensor if off-nominal, then the conflict directed

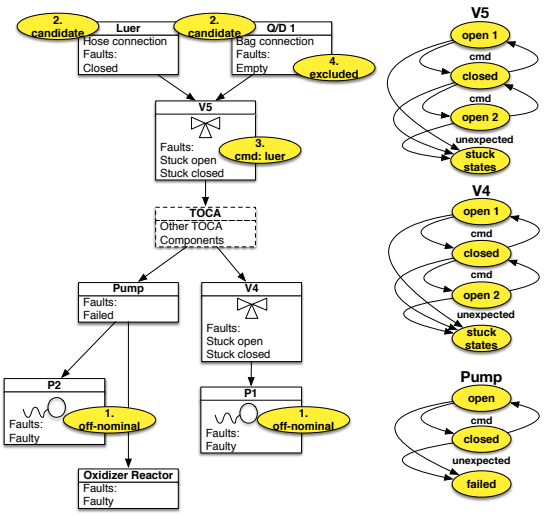


Figure 14. HyDE Model. The left side of the figure shows the components and faults; the right shows the modes and possible transitions between modes for a subset of the components. The numbered ovals show the evolution of the set of candidate faults explaining the observed system behavior.

search describe above would result in finding the component that affects all off-nominal observations. After all data has been presented the final diagnosis is written for further processing by the JSON Writer. Figure 14 shows a fragment of the Liquid Loop model, and a situation when both the P1 Sensor and P2 Sensor (bottom) are off-nominal. The steps of the search for explanations of the off-nominal readings in the sensors. In this case the search might identify faults in both the Bag (top middle) and Hose (top left) as possible candidates, but based on the state of valve V5 (middle), as determined by the actual TOCA data, the Bag fault is excluded, leaving the Hose fault as the only explanation.

The HyDE component also has the additional task of determining if a Calibration run on TOCA failed. In order to do this HyDE first checks whether the current run is a Calibration run based on the TOCA sample file. Then it checks if TOCA performance is nominal based on the results of HyDE and IMS. If TOCA performance is nominal and the current run is a Calibration run, the application reads additional TOCA sample log data to determine the expected calibration value of the TOC. The TOC value is read from the data files and a comparison with expected value determines whether the calibration failed or not. In the case of a Calibration Check fail, a Calibration activity request is sent to the Scheduler.

III.D.5. JSON Writer

The JSON Writer is responsible for generating the bulk of the dynamic content for the AMO TOCA UI. This includes the files that drive the TOCA Results Tab (each individual TOCA result plus the drop down menu), the TOCA Data Tab (each TOCA data plot plus the historical plots), and the TOCA Plan Input Tab (Calibration activities in Activity History). The AMO Monitor invokes the JSON writer after all other TOCA data processing is complete.

III.D.6. SSC Limit checks

The SSC limit checks are performed once every 60 minutes. The AMO monitor queries the Icinga VM and retrieves SSC data, which is placed into an SQLite cache. The data for each SSC is then processed to determine whether it is offline, whether there are any alerts and when they occurred. It also generates the plot data for each SSC parameter. The number of off-nominal SSCs is determined and dynamic data is generated for the SSC Overview and Details Tabs as well as the plots. The SSC Limit checking component creates the JSON files after processing the SSC data. Note that while the UI shows whether SSCs are simultaneously connected to the wired and wireless networks, this is not considered an Alert and is not stored in the Alert history.

IV. AMO Server Component Integration

AMO Server components are integrated at a technical level by the AMO Monitor. As described previously, this component is responsible for invoking each server side component in response to events. In the next section we will describe how the components work together in each key scenario.

AMO component integration is also accomplished by knowledge representing what actions the crew must take if a fault takes place, limits are exceeded, or an off-nominal or anomalous situation is encountered. This knowledge is explicitly represented in the AMO UI, but the UI behavior is driven by the output of AMO Server components. For example, if IMS detects an anomaly in TOCA processing, the AMO UI Recommended Action drop-down in the Results tab will direct the crew to the Data tab, and indicate the processing state in which the anomaly took place. However, the UI depends on the output of IMS to report the processing state in which the anomaly took place. Similarly, if HyDE detects a fault, the AMO UI Recommended Actions drop-down shows the correct recommended action, but based on HyDE output. SSC action recommendations are driven by the output of the Limit Checker. Finally, the HyDE component can recommend the Calibrate action to the Scheduler if the Calibration Check action fails. These forms of knowledge reflect the expertise that a human flight controller would normally bring to bear; they are codified in the complete AMO software system to provide a decision aid to the crew that stands in for flight controllers who may not be available in a timely manner.

We now describe how the AMO architecture processes data when a TOCA data file is recieved.

1. A TOCA sample file is delivered to the shared file system.
2. The AMO Monitor component detects this file, determines it is new, and performs some sanity checks.
3. The AMO Monitor invokes IMS.
4. Once IMS is complete, the AMO Monitor invokes HyDE.
5. Once HyDE is complete, the AMO Monitor invokes the JSON Writer.
6. Once the JSON Writer is complete, the dynamically created content is ready for processing by the Web server and for display on the AMO UI.

The steps performed by AMO in this scenario are shown in Figure 15. The other scenarios are omitted in the interests of brevity.

V. AMO Ground System

The AMO system onboard produces files that are downlinked daily and then moved to multiple ground-based instances of AMO. Since AMO is a web service, each of these instances is a version of the same single-page Web application, accessed via a URL, just as it is onboard ISS. Once the proper JSON files generated onboard are moved to the right place, ground will see what there crew sees. In addition, once the TOCA samples are downlinked from ISS for analysis by ground, we can re-process the TOCA samples to validate the behavior of AMO onboard. Ideally, we would do the same with Icinga database, but we did not have access to this data for reprocessing. In this section, we describe the process of moving data from ISS to the ground-based instances, and the purpose of each instance.

The Air Ground Link consists of NASA developed software called SWRDFSH. SWRDFSH essentially acts like ftp between ISS and the MCC, with additional scripting capability to collect files and move then to proper directories. Additional NASA developed software called OCAMS[6] allows automatic transfer of files according to a schedule. We scheduled downlink of files three times a day; downlink products included all dynamically created content. Once downlinked data was resident in the MCC, we automatically retrieved it from the MCC. Uplinks were nominally scheduled hourly, in order to uplink TOCA schedules extracted from the crew's plan as soon as they were available. We manually pushed schedule updates to the MCC because of the need to manually ensure the extracted schedule had all the needed information in it.

The ground system comprises a total of four mirrors of the state of AMO onboard ISS, each with a different purpose. The first instance duplicated the state onboard ISS by simply moving the JSON files generated onboard into the UI, without actually processing either TOCA or SSC data. A second instance replicated the processing performed onboard once the TOCA sample files are received; this instance mirrors

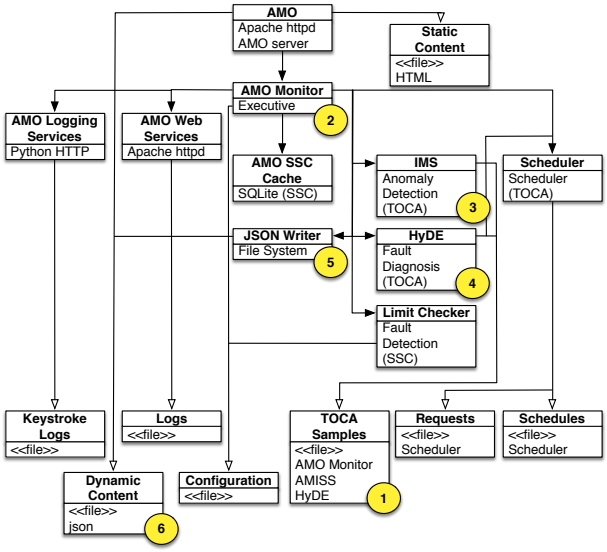


Figure 15. AMO Server component order of invocation after arrival of TOCA sample file on filesystem. The numbered circles show the order each component is invoked by the Monitor after a TOCA sample file is detected on the filesystem.

the TOCA UI, but not the AMO UI. Both of these instances are maintained for situational awareness by ISS flight controllers; even though the instances are 'fully functional', allowing e.g. Requesting tasks, adding Crew Notes, and so on, they are not manipulated in this way. Both of these instances are hosted on rack-mounted servers and are available as a Web service, on any computer.

We also maintain two 'development' instances for testing and evaluating problems. One such instance is on a rack-mounted server, and another is maintained on SSC computers in a laboratory, in the event that a problem cannot be replicated with the rack-mounted server system (due to operating system or browser incompatibility.)

We maintain an experiment tracking Web page that uses the TOCA JSON files for schedule requests and TOCA analysis results to help track crew recommendations, analysis results, and actual TOCA performance.

In addition to the JSON files that populate the Web pages, changes to the schedule, including requests made by the crew, are downlinked in a single JSON file. The contents of this file are formatted in such a way that the flight control team can easily read them, and emailed to the team.

VI. AMO Operational Complexity

Parameter	TOCA	SSC	Total	ISS
Data items	22	161	183	170K
Displays	13	185	198	5K
Procedures	12	7	19	4K
Plan Steps	6	0	6	.5K
Constraints	10	0	10	.2K
Faults	70	207	277	13K

Figure 16. Quantification of AMO Operational Complexity, measuring key operational parameters. Estimates of the corresponding ISS operational parameter measurements are included for comparison.

As described earlier, the AMO concept involved transitioning operations responsibility, such as making plans, executing those plans, monitoring data, and responding to faults, from MCC to the crew. The difficulty

of these operational tasks is quantified by assessing the 'size' of the problem, in terms of task parameters. In this section we discuss those parameters, and how they were measured for the AMO experiment. This provides a basis for understanding how much of a hypothetical future space mission's future operations were demonstrated by this experiment. A summary of the parameters is shown in Figure 16, with a description of each item in the following paragraphs.

The number of individual *data* items produced by the systems is a simple measure of how much work is required to maintain awareness of the system's state. This measure is somewhat misleading, as the data rate, data delivery frequency, and frequency with which the data can change are not measured, but serves as a good first assessment of the difficulty of keeping tabs on these systems. For TOCA the number of data items is the subset of TOCA parameters selected for display to the crew; for SSC it is the number of data items for all SSC computers onboard. The large number of SSC data items reflects the fact that there are 7 data items per SSC, and 23 SSCs onboard ISS.

The number of *displays* organizing the data is another measure of how much work is required to maintain awareness of the system's state. Since multiple data items are combined on single displays, and a data item may appear on multiple displays, there is a complex relationship between the number of data items and the number of displays. For TOCA, the displays includes the Results tab and Activity History, plus each plot in the Data tab. For SSC, the displays includes the Overview tab, each SSC Details page, and a plot for each SSC parameter; since there are 23 SSCs, the total number of displays is higher than the number of SSC data items.

The number of *procedures* for operating the systems is a measure of the complexity of executing plans for the systems. Procedures are step-by-step sets of instructions for operating systems; procedure steps typically take minutes to perform. The procedures for TOCA and SSC that the crew may be called upon to perform are easily counted. However, of these procedures, only a small number are nominal procedures (7 for TOCA, none for SSC), and the remainder were not used during the experiment. Similarly, the procedures may vary widely in the number of steps, and not all steps may be executed.

The number of *plan steps* is also a measure of the complexity of creating plans for the systems. A plan step for TOCA corresponds directly to performing a single procedure. As described, plans span three weeks, with between one and three activities a week. The typical plan averaged two activities a week. There were no plans for SSC management during the experiment.

The number of *operational constraints* that limit the set of acceptable plans is an additional measure of complexity of creating plans. To quantify the number of plan constraints for TOCA plans, we use those constraints in Figure 10, plus the TOCA duty cycle constraint, priority rule for ordering activities in the same day, and the constraint that Calibration activities must occur on consecutive days.

Finally, the number of *faults* and fault responses measure the complexity of managing systems in the presence of faults. For TOCA, we include TOC Off nominal, anomalous TOCA historical comparison for each of the 22 processing states, and the faults described in Figure 13. This count is not completely realistic; the two most common failures of TOCA are the empty sample bag and sensor faults, and it is somewhat misleading to consider anomalies in each state separately. For SSCs we include faults for each SSC parameter threshold, plus two additional conditions (SSC offline, and SSC connected to two networks).

These numbers can be compared to the overall operational complexity of the ISS as a whole; hundreds of thousands of data items, thousands of displays and procedures, hundreds of operational constraints, plan steps of multiple hundreds per day, and tens of thousands of faults and responses. It is also important to realize that the ISS crew managed TOCA and SSC systems for at most 10 minutes per week, while they performed their other tasks. When viewed this way, the AMO experiment appears quite modest. However, the demonstration is important because it required ISS crews to manage critical ISS systems, in the flight environment. This required accurate characterization of all of the operational constraints, commands, and data, as well as seamless integration of AMO software with ISS systems. It is, therefore, a complete test of crew autonomy.

VII. AMO Software Performance and Lessons Learned

In this section we describe the AMO software usage statistics, performance, and lessons learned. For software usage statistics, we describe which parts of the software were used, and how often. By performance, we refer to how well the software components performed compared to flight controller inputs, or assessment of faults and anomalies. Specifically, for the Scheduler, we analyze how often the scheduler recommended an

System	Screen	Uses	% Total
Main	-	32	15%
TOCA	all	149	67%
TOCA	Results	48	32%
TOCA	Plan Input	41	27%
TOCA	Data	35	23%
TOCA	References Procedures	14	9%
TOCA	Fault Indicators	4	2.5%
TOCA	Components Functional Overview	3	2%
TOCA	References Software Fault Messages	3	2%
TOCA	References Analysis Plot Descriptions	1	.5%
SSC	all	36	17%
SSC	Overview	21	58.3%
SSC	Details	13	36.3%
SSC	References Procedures	1	2.7%
SSC	Data	1	2.7%

Figure 17. On-orbit usage of AMO UI elements.

activity that was also recommended by flight controllers; for IMS we assess how often an anomaly was truly considered anomalous; for HyDE we analyze how often faults were detected and whether they were truly faults; and finally, we analyze whether or not faults were diagnosed correctly.

VII.A. AMO On-Orbit Usage Statistics

On orbit keystroke data was logged every time the software was used for the entire duration of the experiment. Each interaction with a specific component of the AMO UI was logged to enable analysis of design features. The logs contained time-stamped entries for every interaction with the software that indicated the device used to view the software (SSC, iPad), the browser (Internet Explorer, Safari, etc.), and the software feature used.

Before the logs could be analyzed, the data needed to be processed. Some features of the software would create multiple entries for one action. For example, whenever a user selected a plot from the drop down list on the TOCA Data tab, one entry would record the plot that the user selected, while the following entry would log activity for the previous plot. If the data was counted without being cleaned, the frequency count for selecting a plot from the drop down list would be double the actual frequency. A similar problem with the raw keystroke logs occurred was seen when users interacted with charts. Abstracting usage of the sliders to zoom required scrubbing 11 log entries, and abstracting a single click and drag to zoom required scrubbing 8 log entries. During the experiment 22 sessions were recorded and analyzed over the duration of the experiment. Crew used iPads and SSCs equally to access the software. Figure 17 summarizes the uses of the main UI elements of AMO. The percentages for usage for Main, TOCA - all and SSC - all sum to 100%, and the percentages within each category are calculated compared to the total number of times each UI element was used for each subsystem.

From this usage data, it is plain that the bulk of the time spent by crew was on TOCA functions. Given that TOCA was the centerpiece of the experiment and hence instructions to the crew, and the high complexity of TOCA operations, this is unsurprising. For TOCA, the bulk of the time is spent analyzing results, drilling down into data, or managing the plan. A small amount of the activity involves referencing procedures or looking up TOCA component descriptions or errors. By contrast, for SSC, the bulk of the time spent was on either the Overview or Details pages.

VII.B. AMO Performance

Overall, the AMO software performed well during the experiment. Qualitative analysis from interviews with flight controllers and crew are still being analyzed, but the preliminary results from all users have been very positive. The crew used the software to analyze TOCA performance 15 times, and on 14 of those occasions correctly identified TOCA performance. In four of those 15 cases, TOCA was off-nominal; the sole crew error was incorrectly identifying one of these cases as nominal. As noted in Section VII.A, crews also used the software to manage SSC laptops; unfortunately, their use of AMO was limited.

Before quantitatively analyzing performance during the experiment, we describe one use case on-orbit, and some reactions of the crew to the software. During the crew's first use of AMO to analyze a Bag Sample activity, the bag was underfilled and TOCA aborted the analysis. The crewmember successfully used AMO to diagnose the failure, and called down with the correct recommended next action. The conversation during this failure was very clear, without multiple back and forth questions between MCC and the crew, as witnessed during previous instances of this failure. This streamlined communication would be considerably simpler in a time-delay scenario. Flight controllers in the MCC also reported referencing the AMO software in real time to help diagnose the problem. Post-flight crew debriefs were conducted with each crew member. While this data is still being analyzed, both crews and flight controllers felt that AMO helped crew manage TOCA effectively. The bulk of the crew responses pertained to TOCA situational awareness and fault management functions. A representative quote: "Understood the underfilled bag case immediately, all made sense to me. I looked at the bag, and sure enough the bag was empty, I knew exactly what was going on." Crew also commented that the software design should be extended to other ISS systems, and that this was the appropriate direction for enabling autonomy. A representative quote: "This is the way we ought to be heading. I know it is difficult to gather all that info, but if you could do it for other systems, it would be great!"

All AMO software performance analysis is performed for TOCA related functions. The TOC and TOC trend analysis was quite straightforward, and classified TOC performance (both individual measurements and trends) correctly 100% of the time; as a consequence, we do not discuss this aspect of the software performance further. While the SSC function ran on-orbit, and generated dozens of alerts, as noted above, it was not heavily used by the crew. Also, unlike for TOCA, the alerts were not easily associated with faults or anomalies where a 'ground truth' for correctness of performance was available. The remainder of this section is devoted to analyzing software performance on TOCA scheduling and fault management functions.

Function	Analyses	Correct	Score
Scheduler	31	28	90 %
IMS	59	52	88 %
HyDE (detect)	59	59	100 %
HyDE (diagnose)	59	57	96 %

Figure 18. AMO Performance.

VII.C. Scheduler performance

The Scheduler performance was very good. The Scheduler recommendations were incorrect in only 3 of 31 cases. In one instance, the Scheduler recommendation was incorrect because the TOCA activities were not uplinked to ISS. In this case, the problem was a 'process' issue rather than a software design problem or bug. The second instance occurred because MCC needed to schedule a Calibration Check one week early due to a full timeline the week it was due (according to the constraints). This extra early run of TOCA would also cause the waste water bag to become full earlier than the software predicted. Thus, for this week, the AMO software and the crew recommended only a Hose analysis when a Calibration Check, Waste Water Bag changeout, and a Hose analysis was required for the future week. It turned out that once the new schedule was uplinked, the Scheduler processed it and created the recommendations as expected, but there was no crew use of the software after the schedule uplink. Hence, strictly speaking, this case involves another 'process' issue rather than a design flaw or bug.

The third Scheduler issue was different. In the last week of January, the Scheduler recommended an extra Bag sample for 1/30/2015. When making recommendations for a given week, the scheduler does not look beyond the end of that week while looking for the last completed or pending execution of a given activity.

The last Bag sample was done on 12/31/2014. The end of the week in which 1/26/15 occurred is 1/31/2015, and there was a Bag sample scheduled for 2/2/2015. According to the Bag sample constraints (every 30 days), the Scheduler recommended the Bag sample for 1/30/2015 and 'ignoring' the scheduled activity on 2/2/2015, because 2/3/2015 is in the 'next' week. Notice the Scheduler does not maintain a 'moving' notion of a week ahead, but 'rolls' the week on Friday. This case hilights a design flaw in the Scheduler that leads to the incorrect recommendation; the solution is to extend the 'lookahead' of the scheduler or roll the week to cover this case.

Throughout the experiment, there were a few instances where a task was moved a few days earlier or later which bumped it into a different "week". These instances were marked as correct via crew and software recommendations because both the crew and software had correctly identified the need for the activity and the when aspect of scheduling an activity was out of scope for this experiment. In addition, there was one unusual case that arose in pre-flight testing that the Scheduler would not handle correctly, in which two Hose samples were scheduled in the same week. In practice, this almost never happens, but a temporary plan caused the Scheduler to produce unexpected results when two Hose samples were scheduled in one week. After this, we introduced a check to ensure no plan uplink with two Hose samples in the same week was uplinked.

VII.D. Anomaly and Fault Detection performance

In this section we discuss the combined performance of the anomaly and fault detection functions. These functions were designed to warn if TOCA was not behaving as expected, or to identify faults if one occurred. When TOCA experienced a problem, it would sometimes abort processing of a sample. On other occasions, however, processing would complete and report TOC results, but TOCA would enunciate a fault message. Since TOCA enunciates faults, it is worth knowing when these enunciated faults occur in conjunction with IMS and HyDE results. Also recall that the UI contains nominal performance regions for most (but not all) of the per-run TOC parameters used by IMS to detect anomalies. Since the UI can direct attention to the TOCA processing phase and time at which unusual behavior takes place, it is also interesting to know when IMS results are correlated with deviations from the nominal performance region. A summary of the samples in which some evidence of a fault or anomaly is shown below in Figure 19.

Sample	Aborted	IMS	HyDE	TOCA Fault	Perf. Region	SME off-nominal
2100	Y	N	Y	N	N	Y
2089	Y	N	Y	Y	N	Y
2115	N	Y	Y	Y	Y	Y
2091	N	Y	Y	Y	Y	Y
2088	N	Y	N	N	Y	Y
2096	N	Y	N	N	Y	N
2134	N	Y	N	N	Y	N
2119	N	Y	N	N	N	Y
2090	N	Y	N	N	N	Y
2131	N	Y	N	N	N	N
2101	N	Y	N	N	N	N
2097	N	Y	N	N	N	N

Figure 19. AMO Off-nomimal Performance for TOCA samples.

We begin our discussion of performance with the two samples in which TOCA processing was aborted. In these cases, HyDE reported a problem but IMS did not. When TOCA aborts, logging of performance data from the hardware terminates and IMS has no data points to process after the abort. On the one hand, this makes it less surprising that IMS does not identify an anomaly in these abort cases. However, since TOCA halts operation for some reason, one might expect IMS to report something wrong. Of particular interest is the case in which TOCA produced a fault message (2089); it is worth considering whether IMS should have reported an anomaly in this instance. The conservative conclusion is that these are 'false negatives' for IMS;

however, it is possible that a TOCA component other than the Liquid Loop is responsible for the abort.

There were ten samples on which IMS flagged an anomaly; as noted above, none of these were cases in which TOCA aborted. Of these ten, there were two cases in which IMS detected an anomaly, and HyDE also detected a fault. As expected, IMS is more sensitive than HyDE, and detects off-nominal situations without an obvious fault. In five of the ten cases in which IMS detected an anomaly, one TOCA parameter was also outside the nominal performance region. The remaining five cases are those in which IMS is the only component that is reporting something off-nominal. The use of IMS to determine anomalies and guide the users to the right part of the plot is a potential benefit, since there are many plots, and deviations from the nominal performance can be hard to find due to plot resolution. Recall that two TOCA parameters are not available for display on the Data Tab (due to a recommendation made by the Subject Matter Experts (SME)s early in design); it is possible these data values were outside their nominal performance regions when these IMS anomalies were detected.

VII.E. IMS performance

IMS performance was good. In Figure 18 above we see that IMS was consistent with SME assessment of TOCA behavior 88% of the time. Of the ten IMS identified anomalies, the flight control team agreed they were anomalous on five occasions. Thus, there were five IMS errors that were 'false positives', that is, IMS indicated a TOCA run was 'nominal' when flight controllers or analysts considered it 'off-nominal'. In five cases (samples 2131, 2101, 2097, 2096, 2134), IMS reported an anomaly but SMEs still deemed TOCA performance acceptable. Two of these cases (samples 2096 and 2134) also showed at least one TOCA parameter deviated from the nominal performance regions. As noted in Section VII.D, there were two aborts in which IMS did not report anomalies. These are considered IMS 'false negatives'. The seven IMS errors are the combination of the false positive and false negatives.

Tuning IMS was challenging for several reasons. During IMS configuration, it was observed that transients between states were hard to train for IMS. There were typically transients between states that simply had to be ignored. IMS could have been retrained on new datasets throughout the experiment, and it is possible that fewer false positives would have resulted if we had done so. However, this was not operationally easy to do for two reasons. First, it was difficult to actually change the AMO software load without significant overhead. Second, it was not possible to re-analyze all previously analyzed TOCA samples and revise their anomaly scores for presentation to the crew or flight controllers, which could have raised questions about discrepancies that we felt would complicate the experiment.

VII.F. HyDE performance

HyDE performance was very good. In Figure 18 above we see that HyDE had a perfect score in terms of detection of faults; that is, HyDE detected faults every time a TOCA fault occurred, and never announced a fault if no faults occurred. HyDE's diagnosis rate is lower, indicating it misdiagnosed faults. Specifically, HyDE correctly diagnosed two of four faults. We discuss this performance more thoroughly below.

On Sample 2091, the sample was aborted and HyDE TOCA detected a fault. The Crew also reported an off-nominal performance and recommended the same response as the software recommendation: that TOCA be checked for water leaks. The actual cause of this off-nominal performance was a bad buffer container. This failure mode is not included within AMO software as it was never envisioned during software development. Hence, even though HyDE correctly detected the fault, the diagnosis was incorrect. Interestingly, sample 2091 was a Hose sample, but the diagnosed fault was an empty Bag; this type of fault should not occur in conjunction with this activity, and therefore could be addressed by the model.

On Sample 2115 HyDE diagnosed a fault with the chiller, due to a high temperature alert. This diagnosis was incorrect, because (unbeknownst to HyDE), a failure outside of TOCA caused high temperatures in the cabin, which in turn led to the TOCA temperature trip. In this case, HyDE did not have this fault mode modeled, although this possible explanation was actually present elsewhere in the AMO software (specifically in the plot descriptions). Once again, HyDE successfully detected the problem, but could not diagnose the true fault, since it was not modeled. Unlike the previous HyDE diagnosis error, this incorrect behavior is due to an 'external' event; cabin temperature data was not easily available to the AMO application, and furthermore it would be difficult to successfully limit the scope of external events or environmental conditions to model in HyDE.

Some other notes concerning anomaly and fault performance are worth mentioning. A total of eight abort cases were available to test HyDE, including cases observed prior to and during the experiment. In one case observed prior to the experiment start, HyDE should have reported at least a potential sensor failure in the case of an abort, revealing a second problem with the HyDE model. In addition to the external event seen in Sample 2115, we saw one additional case prior to the experiment in which an external event caused a TOCA fault condition. In this case, the unexpected shutoff of the nitrogen line to TOCA during operations resulted in an under-pressure event that was misdiagnosed as a TOCA fault.

VII.G. Additional Lessons Learned

In addition to the specific flight performance related lessons learned, there were several other useful lessons in deployment of the technology that we mention here.

The data-driven nature of IMS and the model-based HyDE facilitated software reuse, which dramatically reduced software development time. The Scheduler was implemented from scratch rather than reusing existing model-based planning or scheduling software due to the relative simplicity of the scheduling problem at hand. However, it was implemented in a data-driven manner to facilitate changes to scheduling frequencies. It is notable that the project went from concept to implementation in roughly 12 months, including the integration of these components which had never been integrated before. This schedule would not have been possible without the existence of these highly reconfigurable reasoning engines.

AMO logs and downlink data were moved to a specific folder on LS1 (refer to Figure 8) for downlink. These files were moved on a schedule consistent with the three times a day downlink schedule. The logs and on-orbit data produced by AMO was large enough that if there was a problem, we could store up to four downlink files a day. We found that on weekends, and occasionally during the week, a combination of factors (short downlink windows plus MCC coverage) led to exceeding the file system limit, leading to corrupted downlinks. We negotiated an increase to the size of the folders and a change in the strategy for downlinks (repeat tries to reduce the chances that a communication outage would cause us to miss our window), and this resolved the problem.

The temperature of a bag sample was manually entered in a portion of the activity description called the Execute Notes in the crew's plan. Specifically, the temperature was not part of the activity or procedure name. This critically impacted the Scheduler: without a reliable way of knowing the temperature of the Bag sample, we would be unable to correctly schedule the next Bag sample. The temperature usually could be automatically extracted from the Execute Note, but on occasion it was not present due to failures of process. We either added the temperature manually, or used default rules to guess which temperature was needed. Other inconsistencies in activity names in the plan occurred as well, e.g. with Buffer Changeout activities.

During development, we learned there would be a change in the TOCA flight hardware prior to the start of the the experiment. This change most profoundly affected IMS and HyDE, since the change altered both the behavior of the device, and therefore the anomalies, but also the TOCA data log format; this meant significant changes to the code that processed TOCA logs and generated inputs for IMS and HyDE. The change also impacted the Scheduler; previously, TOCA analyses were short enough that two could, conceivably, be performed in a single day. However, the new generation hardware was only able to process a single TOCA sample per day. In addition, a small number of scheduling rules changed as well, e.g. the number of TOCA samples needed prior to a buffer changeout increased. Significant changes were needed to retrain IMS and MMS, and to revise the JSON writer to generate the plots correctly. While this change did cause some reworking of the software, on balance the model- and data-driven foundations of the server side components required little reprogramming.

A concurrency bug in TOCA occasionally resulted in each TOCA activity record being duplicated. This led to some IMS false alarms during development, in which TOCA samples that should not have exhibited anomalies did so. The source of the bug was not identified until after operations (it only manifested during certain types of TOCA activities). The replicate records behavior occurred during the experiment, however, to our knowledge, none of the IMS 'false alarms' we experienced during the experiment were caused by this behavior.

As noted, AMO requires processing the crew plan to extract TOCA activities for use by the Plan Input tab. Early in development, we had planned to perform this function onboard ISS; the crew's plan is uplinked to LS1, and the data file and format were sufficiently well characterized that on onboard function was feasible. However, other software development work was planned that would change the file format at roughly the same time as the experiment, so we opted to be conservative and do our processing on the ground.

The AMO logic to detect that an SSC was simultaneously wired and wireless wasn't perfect. An Icinga command called "check_arp" that looks up addresses and sees if the address returns multiple MAC addresses using the "arp" command. Both DNS and ARP cache their data so it may be that the cache is stale, but it should give an indication that the crewmember either switched to WiFi without running the windows program to reconfigure the network, or that they've left both the wireless and ethernet active, sometime in the recent past. Unfortunately, we found that SSC laptops were often left untended for long periods of time, and that the results were sometimes inaccurate.

A combination of unexpectedly high AMO component processing times and a process kill bug led to problems. Processing times on orbit were sometimes as much as six times larger than were anticipated or recorded during testing on the ground. Processing times varied significantly throughout the experiment; while the reboot of the computer our software runs on would sometimes lead to a reduction in processing times, they would then increase again, sometimes in as little as 24 hours. As a result of this unexpectedly high processing time, the AMO Monitor sometimes issued a process termination to sub-processes that were taking too long. Unfortunately, this revealed an unforeseen bug in the software package that issued the termination notice. Once this was uncovered, it was decided to change the timeouts in order to allow processes to complete normally. AMO was designed to read its configuration file constantly, so changes to the configuration (once we figured out what we needed) were trivial. In addition to this problem, downlink processing required a process running on a computer at ARC to retrieve and push the downlinked data to our mirror servers. On occasion, this process crashed and needed restarting. It was sometimes not clear which element of the data processing pipeline had failed.

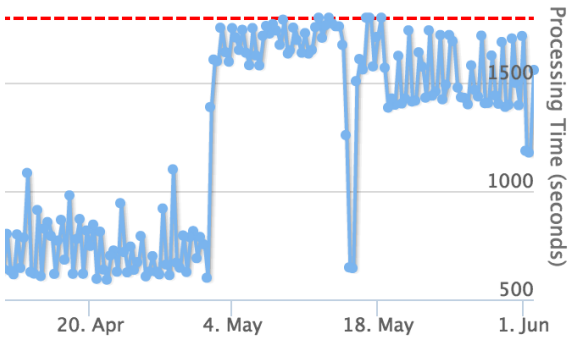


Figure 20. Changes in ICINGA processing times between April and June.

VIII. Conclusions and Future Work

The AMO experiment conclusively demonstrated that an autonomous crew can use decision support software to manage plans, monitor system performance, and detect and respond to off-nominal and fault conditions. The AMO software designed and developed for this experiment included a combination of existing and custom components providing the needed planning, monitoring and fault management functions. These functions were implemented using highly reconfigurable software components, making a very aggressive development schedule feasible. As shown in the usage statistics (Figure 17), all elements of the AMO software were used during the experiment. The performance results for TOCA showed that software accuracy, across all functions, exceeded (88%). After resolution of the software concurrency bug, AMO ran on-orbit problem free for seven months. Qualitative analysis from interviews with flight controllers and crew are still being analyzed, but the preliminary results from all users have been very positive. The crew used the software to analyze TOCA performance 15 times, and on 14 of those occasions correctly identified TOCA performance. In four of those 15 cases, TOCA was off-nominal; the sole crew error was incorrectly identifying one of these cases as nominal.

The bulk of the issues discovered that impacted performance were process related (e.g. Scheduler uplink) or environmental conditions or faults that were ground-ruled out for the experiment (e.g. for HyDE). It is possible that changing IMS during the experiment would have led to increased performance, but this operational change was also out of scope of the experiment.

As described in Section VII.B, overall AMO software performance on TOCA management activities was very good. Individual components performed quite well, but not perfectly. Lessons learned from the problems and errors in the software can be categorized as follows:

1. The IMS error rate was partially a function of the existing training examples. New IMS models could have been generated from more on-orbit data, and could potentially have lowered the error rate. However, the other contributor to the error rate, the transition between processing states, is harder to address, and it is likely that some false positives cannot be eliminated.

2. The recommended action from IMS could have identified the Liquid Loop parameter maximally contributing to the anomalies. However, this was not feasible due to timing constraints, and also due to the early decision not to make all Liquid Loop parameters available to the crew. The solution to this problem is to ensure all data used in analysis can be seen by the crew.

3. Fault modeling deficiencies led to low HyDE success rate in diagnosis. The HyDE modeling problems require examples of the faults for regression testing. However, the case of external cabin temperatures requires identifying cabin temperature data, which was not easily available to the application due to the deployment configuration, as well as augmentation of the model.

4. Algorithm design caused one Scheduler error. This problem can be addressed by extending the Scheduler's 'lookahead' distance to correctly identify scheduled activities.

5. Process problems caused two Scheduler errors. These problems can be addressed by improved operations process.

6. While SSC lessons learned are scarce due to lack of ground truth and limited use, the lack of accuracy of the wired and wireless connection state is an obvious area of improvement.

As noted in section VI, the tasks and systems used in this experiment represent a small part of the operational needs of a future exploration spacecraft. Further experiments with ISS and ground analogs are needed to continue fleshing out autonomy enabling technologies, concepts of operations, system designs, and lessons learned.

Acknowledgements

This project was the work of a large team of dedicated individuals; the authors gratefully acknowledge the hard work and dedication of Ilya Avrekh, Vicky Byrne, Lionel Delmo, Peter Morgan-Dimmick, Tyler Doubrava, Travis Fitzgerald, Nicholas Fritz, Jayleen Guttromson, Elisca Hicks, David Korth, Jenessa Lin, Harry Litaker, Brian Martin, Jason Mintz, Lee Morin, Chad Morrison, Jayanta Ray, Landon Sommer, Hao Thai, Abe Velazco, Shawn Wolfe, and all of the flight controllers and crew who flew the ISS during this experiment. This work was funded by the NASA Advanced Exploration Systems (AES) Program.

References

[1]Rader, S. N., Regan, M. L., Janoiko, B., and Johnson, J. E., "Human-in-the-Loop Operations Over Time Delay: NASA Analog Missions Lessons Learned," *Proceedings of the AIAA International Conference on Environmental Systems*, 2013.

[2]Frank, J., Spirkovska, L., McCann, R., Wang, L., Pohlkamp, K., and Morin, L., "Autonomous Mission Operations," *Proceedings of the IEEE Aerospace Conference*, 2013.

[3]Beisert, S., Rodriggs, M., Moreno, F., Korth, D., Gibson, S., Lee, Y. H., and Eagles, D., "Development and Execution of Autonomous Procedures Onboard the International Space Station to Support the Next Phase of Human Space Exploration," *Proceedings of the AIAA Space Conference*, 2013.

[4]Iverson, D., "Inductive System Health Monitoring," *Proceedings of the International Confernce on Artificial Intelligence*, 2004.

[5]Narasimham, S. and Brownstone, L., "HyDE - A General Framework for Stochastic and Hybrid Model - Based Diagnosis," *Proceedings of the 18th International Workshop on the Principles and Practices of Diagnosis*, 2007, pp. 162 – 169.

[6]Clancey, W. J., Sierhuis, M., Seah, C., Buckley, C., Reynolds, F., Hall, T., and Scott, M., "Multi-Agent Simulation to Implementation: A Practical Engineering Methodology for Designing Space Flight Operations," *Engineering Societies in the Agents' World VIII*, edited by A. Artikis, G. O'Hare, K. Stathis, and G. Vouros, Springer, 2008, pp. 108–123.