

# Scheduling Spitzer: The SIRPASS Story

**David S. Mittman**

Jet Propulsion Laboratory  
California Institute of Technology  
4800 Oak Grove Drive  
Mail Stop 301-250D  
Pasadena, California 91109-8099  
[David.S.Mittman@jpl.nasa.gov](mailto:David.S.Mittman@jpl.nasa.gov)

**Robert Hawkins**

Space Telescope Science Institute  
3700 San Martin Drive  
Baltimore, Maryland 21218-2410  
[rhawkins@stsci.edu](mailto:rhawkins@stsci.edu)

## Abstract

NASA's Spitzer Space Telescope was launched on August 25, 2003 from Florida's Cape Canaveral Air Force Base. Drifting in a unique Earth-trailing orbit around the Sun, Spitzer sees an optically invisible universe dominated by dust and stars. Since 1997, the Spitzer Integrated Resource Planning and Scheduling System (SIRPASS) has helped produce spacecraft activity plans for the Spitzer Space Telescope. SIRPASS is used by members of the Observatory Planning and Scheduling Team to plan, schedule and sequence the Telescope from data made available to them from the science and engineering community. Because of the volume of data that needs to be scheduled, SIRPASS offers a variety of automated assistants to aid in this task. This paper will describe the functional elements of the SIRPASS software system — emphasizing the role that automation plays in the system — and will highlight lessons learned for the software developer from a decade of Spitzer Space Telescope operations experience.

## Introduction

In this introductory section, we describe the project environment in which the Spitzer Integrated Resource Planning and Scheduling System (SIRPASS) was developed and used in operations. We include high-level descriptions of the Spitzer project, spacecraft facility, and both flight- and ground-based systems; much detail has been excluded, but can be found by consulting the references cited in the text.

### Spitzer Space Telescope

The Spitzer Space Telescope (Werner et al. 2004) is the fourth and final of NASA's great observatories, designed to take images and spectra of astronomical objects in the infrared. Spitzer consists of a spacecraft, a 0.85-meter

telescope and three cryogenically cooled science instruments: the Infrared Array Camera (IRAC), the Infrared Spectrograph (IRS), and the Multiband Imaging Photometer for Spitzer (MIPS), as shown in Figure 1.

Launched from Cape Canaveral, Florida, on August 25, 2003, the mission plan called for a 60-day In-Orbit Checkout (IOC) followed by a 30-day Science Verification (SV) phase. Spitzer employed an innovative warm launch architecture: the telescope is located outside the cryostat — rather than being encapsulated within it — and was at ambient temperature at Launch. The telescope baffle was cooled by helium vented from the cryostat, a resource that lasted until May 2009 and gave the spacecraft nearly 5.5 years of prime mission lifetime. After exhausting its supply of helium, Spitzer became too warm to conduct scientifically useful observations using either IRS or MIPS, but was still sufficiently cold to perform some IRAC observations. Spitzer was designed to perform scientifically useful observations for at least 2.5 years, but continues to do so almost ten years after launch.

Spitzer is in an earth-trailing solar orbit, slowly drifting away from Earth at rate of approximately 0.64 AU every five years. Although this orbit was chosen primarily for thermal and launch mass considerations, it also vastly improves the simplicity and efficiency of operations. As a cryogenically cooled spacecraft, Spitzer is constrained to keep its solar arrays and sun shield pointed toward the Sun by restricting its ability to pitch and roll. As a result of these pointing restrictions, Spitzer can only observe targets within a narrow annulus that rotates about the sun once per year, but ends up covering all inertial targets for at least 40 days each year.

## Normal Operations Process

At the completion of SV, Spitzer transitioned into its normal operating mode; Spitzer's three science instruments are operated in a discrete number of observing modes (seven specified at launch, with an eighth added later), which are characterized as Astronomical Observation Templates (AOTs). In order to design an observation, the astronomer, using the Spitzer Planning Observations Tool (Spot), chooses a template and specifies a number of parameters that are specific to it (Laine et al. 2006). These inputs produce a complete recipe for the commanding needed to carry out the observation on board Spitzer. The resulting fully specified observation is called an Astronomical Observation Request (AOR) and the parameters are stored in the Spitzer Science Operations Database (SODB) at the Spitzer Science Center (SSC) along with estimates of the duration of the observation, the volume of data it will generate, and other important attributes derived from the input parameters. In addition to AORs, the SODB also stores similar types of data for Instrument Engineering Requests (IERs) and Spacecraft Engineering Requests (SERs). The AOR/IER Resource Estimator (AIRE) calculates resource estimates for both AORs and IERs, while either SIRPASS itself or the Observatory Engineering Team provides the SER resource estimates. The AORs, IERs, and SERs — along with their

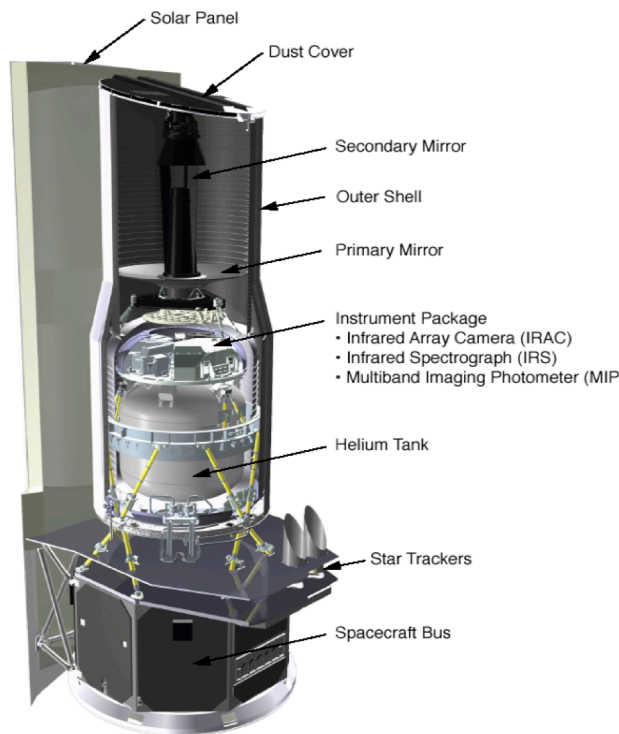


Figure 1. Spitzer Space Telescope

resource estimates and constraints — are the fundamental inputs to the planning and scheduling process (Barba et al. 2006).

## Operations Organizations and Interfaces

The Spitzer flight operations organization has two components: the Science Operations System (SOS) and the Mission Operations System (MOS). The Observatory Planning and Scheduling Team (OPST) is in the domain of the SOS, which is managed by the SSC and located at the California Institute of Technology (Barba et al. 2005).

The SSC is responsible for evaluating and selecting proposals and is also responsible for providing technical support to the science community, performing mission planning and science observation scheduling, instrument calibration and instrument performance monitoring, data processing and production of archival quality data products, and funding science research. The MOS is comprised of teams located at both the Jet Propulsion Laboratory (JPL) and Lockheed Martin in Denver.

At the SSC, the OPST is responsible for the generation of one-week schedules and sequences consisting of both science and engineering activities, which are then sent to the MOS for detailed command generation, modeling, validation, and radiation to the observatory. Data are received from the observatory by the Deep Space Network (DSN) and transferred to JPL where relevant data is extracted from the spacecraft telemetry. Science and instrument engineering data is then sent to the SSC for science data processing, science product generation, and archiving. The main MOS teams that the OPST interfaces with are the Mission Sequencing team (MST) at JPL and the Observatory Engineering Team (OET) at Lockheed Martin in Denver (Barba et al. 2006).

## Observatory Planning and Scheduling Team

The OPST at SSC uses SIRPASS for creating the long-range plan (LRP) and short-term schedule (STS) for Spitzer science, instrument engineering and spacecraft engineering activities. SIRPASS performs both an LRP and an STS function, and completes the generation of spacecraft plans by creating a set of mid-level sequence products ready for delivery to MST for expansion, validation and translation to uplink products. To support its LRP function, SIRPASS incorporates a version of the Science Planning Interactive Knowledge Environment (Spike) software adapted by Laurence Kramer and Robert Hawkins at the Space Telescope Science Institute (STScI).

Spitzer's science instruments can only be used one at a time, and — for reasons of efficiency and maximizing the observatory's prime mission duration — there is a preferred ordering (IRAC, then MIPS, then IRS) and duration (approximately 14 days) of instrument usage. In

the LRP, this information is folded together with the information about the AORs in the SODB to produce a Baseline Instrument Campaign (BIC) for an extended period of time (usually one year) that allocates windows of time when each of the three instruments is planned to be available for observations.

Once a BIC is established, the LRP also allocates what are called plan windows for each observation. The LRP function of SIRPASS — provided by the Spitzer Spike subsystem — assigns “plan windows” to each request in the SODB by intersecting target visibility as a function of time with the availability of a particular instrument (per the BIC), and any constraints associated with the observations. For example, constraints may include requests by observers to acquire data on specific dates, in a specific order, or at specified time intervals. Plan windows allow the scheduler to identify which science requests are available to schedule in the given time period. Plan window updates are usually done once per week in order to keep pace with the frequent modifications and additions to observing programs in the SODB, as well as to take into account observations that have been scheduled.

Filling each week on the observatory timeline with engineering and science activities consistent with the BIC is the responsibility of the STS process. SIRPASS uses a variant of the Greedy algorithm to produce an optimized schedule of science observations (Samson 1998). The SSC Director approves the weekly schedule and then the sequence files are transferred to the MST at JPL for generation of command products, Mission Manager approval, and uplink to the spacecraft for execution (Barba et al. 2006).

## Spitzer Integrated Resource Planning and Scheduling System

SIRPASS (Figure 2) is the last known adaptation of the Plan-IT II planning and sequencing tool first developed by William “Curt” Eggemeyer in the 1980s (Eggemeyer et al. 1997). The Galileo, Mars Pathfinder and DATA-CHASER (Chien 1999) flight projects each used subsequent versions of Plan-IT II as part of their planning and scheduling systems. The lead author developed the Plan-IT II

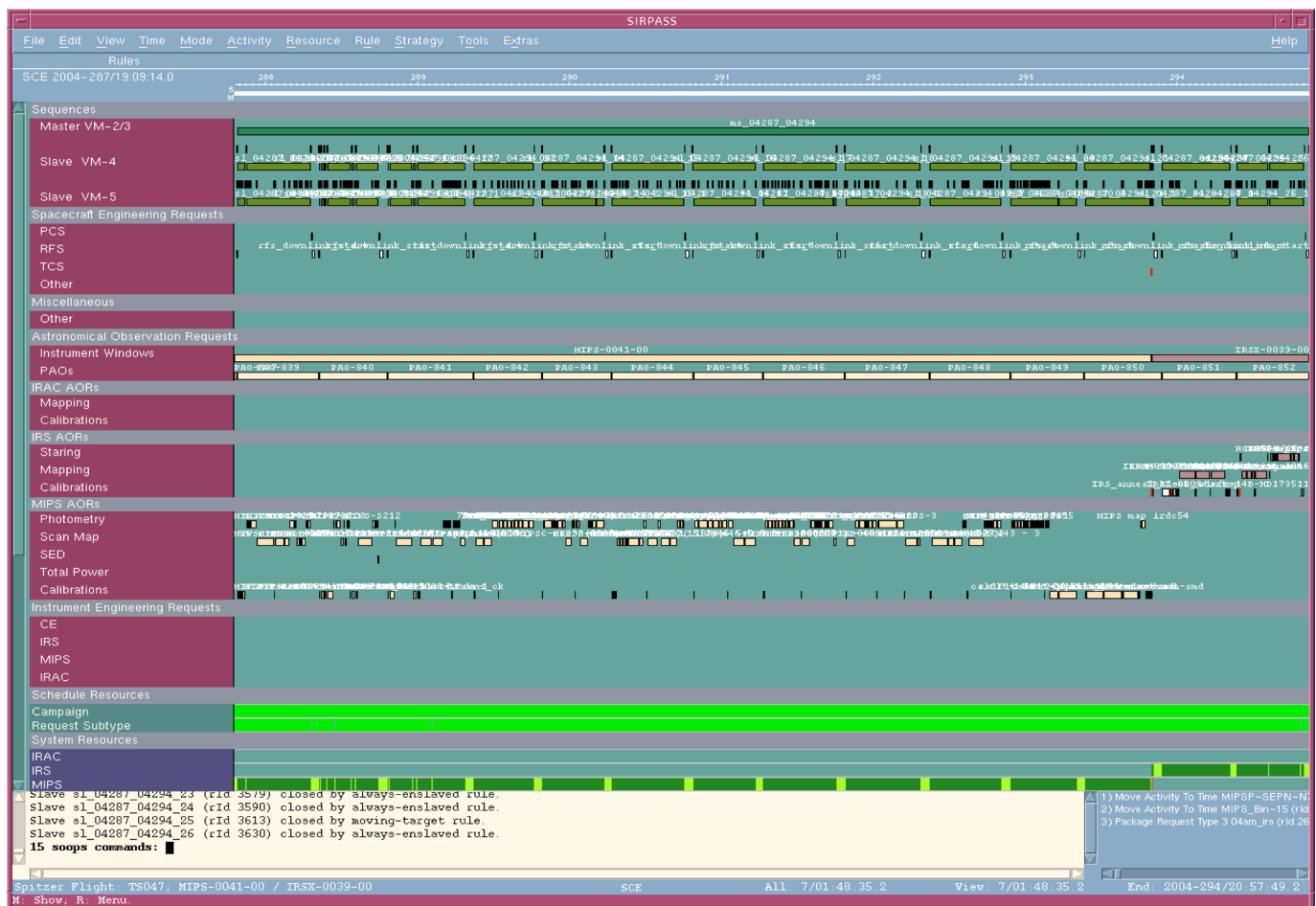


Figure 2. SIRPASS

adaptations for both Mars Pathfinder and the Spitzer Space Telescope.

SIRPASS is an interactive software application for the planning and scheduling of Spitzer activities. The application is designed to be a software-based assistant to the members of the OPST, who are experts in the planning and scheduling of Spitzer observations. The application is a Decision Support System that provides an integrated platform for assessing the quality of Spitzer scheduling options. The application aids in scheduling instrument selection, assigns schedule times to specific observation requests, and generates stored sequence products destined for execution on Spitzer.

### **Plan-IT II Core**

SIRPASS is an adaptation of Plan-IT II, a JPL-developed spacecraft activity planning software application. The Plan-IT II software application has a long history of use at the Jet Propulsion Laboratory, including use on the Mars Pathfinder and Galileo projects. Plan-IT II supports the modeling of spacecraft activities and their impacts upon a variety of resources. Because the Plan-IT II software architecture utilizes a highly object-oriented design, the core software can be easily extended for specific scheduling problem domains. Plan-IT II is developed in Allegro CL, a dynamic object-oriented development environment for ANSI Common Lisp from Franz, Inc.

### **Spitzer Science Planning Interactive Knowledge Environment**

A survey of astronomical observatory planning and scheduling tools conducted in 1998 identified Spike as having the greatest degree of applicability to the difficult scheduling problem presented by the Spitzer Space Telescope. Similarities between the Hubble Space Telescope (HST) and Spitzer, including characteristics of the astronomer observer community, made Spike — first developed for HST — the natural choice for providing the required LRP function in SIRPASS.

#### **Introduction to Spike**

Effective utilization of space-based resources such as astronomical observatories is critical to supporting NASA's mission of advancing and communicating scientific knowledge and understanding of the Earth, the solar system, and the universe. Part of the success of the HST mission has been the creation of systems that allow HST to achieve high science efficiency. HST has been able to supersede the pre-launch predicted efficiency of 35% with average efficiencies up to 50% during normal operations, with peaks into the 55% range (Adler and Workman 2008). A significant portion of this increase is due to ground system support software and operational improvements that were developed post-launch (Hawkins

2009). Spike is one of the major software components that contributed to this increased efficiency (Johnston and Miller 1994). Over its 20+ years of development, Spike has evolved significantly, and has been applied successfully to a number of astronomic missions, in addition to Spitzer.

#### **History of Spike According to Hubble**

HST is a general-purpose space observatory that provides support for near-infrared, visible, and ultraviolet frequencies. In contrast to Spitzer, HST's Low Earth Orbit (LEO) dominates the scheduling of observations. The two major constraining factors that LEO contributes to HST scheduling are that the target cannot be observed when occulted by the Earth or near the bright Earth limb, and, in addition, HST may not observe when passing over the South Atlantic Anomaly (SAA). Aside from the LEO, the main physical constraint on HST observations is that the target selected by the observer must not be scheduled within a minimum angular separation from the sun or moon. Finally, a user can place other requirements on an observation including absolute time windows and roll angles for observations, as well as timing and roll links relative to other observations.

The first five years of experience using Spike to support service mode LRP illustrated three criteria that were recognized as important for service mode observing plans: Efficiency, Stability and Mutability (Kramer 2000). During those first five years of operations, Spike produced plans that were not successful in meeting these three criteria. The plans that Spike produced led to low efficiency short-term schedules, were unstable, and were very resistant to incremental change. The first Spike LRP planned observations to weeklong bins for input to a short-term scheduler. By design these bins had to be well oversubscribed so that there would be an adequate mix for the scheduler. This frequently meant that half of the observations planned for a given week's bin would get bumped to another bin, once the schedule for the week was generated, which was frustrating for both scheduling staff and observers.

To address these problems, Spike was redesigned to incorporate a new planning and scheduling operations concept using "plan windows," which are typically 4–8 week windows that are a subset of the observation's constraint windows (Giuliano 1998). A plan window represents a best effort commitment to schedule in the window. Plan windows from different observations can overlap and the windows for a single observation can be non-contiguous. Spitzer Spike was based upon this model, rather than the original bin-based model that other missions had used.

## **Adapting Spike**

### *Spitzer Space Telescope*

Initial development of Spitzer Spike focused on one of the primary differences in the Spitzer mission – the sequential nature of instrument usage described above. This led to the development of an extension to the Spike plan window concept, “instrument windows.”

Instrument window campaigns (the BIC) were constructed using a layered strategy. This strategy focused on placing the most constrained “absolute time” observations on the timeline first, which output a skeleton BIC determined by these windows. The second step was to fill gaps between these based on a variety of criteria, including desired window size and the defined ordering described previously. Finally, the remaining observations were given plan windows where their constraint windows intersected appropriate instrument windows, modifying the BIC only when necessary. (Kramer, 2000)

### *James Webb Space Telescope (JWST)*

The JWST will be a large, infrared-optimized space telescope, designed to find and study the first galaxies that formed after the Big Bang. JWST will have infrared sensitive detectors and a 6.5-meter segmented primary mirror that allows it to also look through interstellar dust clouds to see and study the formation of stars and planets. The telescope will have a lifetime of 5 to 10 years and will be placed about 1.5 million km from Earth in an orbit around one of the semi-stable Lagrange points in the Earth-Sun system (L2).

JWST has many similarities to both HST and Spitzer. Like HST, JWST will be operated in service mode, have similar observer constraints and allow some parallelism in instrument usage. More akin to Spitzer, JWST will be optimized for infrared observation, and not have the highly constraining earth orbit. JWST is also limited by non-replaceable resources and not repairable, like Spitzer, so optimization of resource usage will be a key focus of LRP.

Development of JWST Spike began after most work on Spitzer was concluded. Our experiences with Spitzer clarified some concepts in software design related to specializing the HST code base for new missions and helped us focus the early JWST work on a re-architecting of Spike (Giuliano 2011) to allow for more coherent Object Oriented design, and further extension for future missions.

### *Far Ultraviolet Spectroscopic Explorer (FUSE)*

FUSE was launched into a low Earth orbit on June 24, 1999, and was designed to perform high-resolution far ultraviolet spectroscopy. FUSE was similar to HST in LEO dynamics, but also had many physical constraints that differentiated it from HST (some of which were only apparent well after launch).

The initial FUSE Spike implementation was developed prior to the advent of the plan window concept discussed above, and thus used the bin-based problem solving capabilities of Spike (Calvani et al. 2004). This version of FUSE Spike used many “repair based” algorithms from the Spike toolkit (Johnston and Miller 1994). Due to issues discovered after in-orbit checkout, a new constraint on reducing the number of slews across the orbital plane needed to be put in place. Since the generic algorithms weren’t easily tunable to handle this problem, early LRPs had to be developed by hand, rather than in Spike. Eventually, the Spike team developed a Campaign Scheduler for FUSE that was loosely based on the concept that had originally been deployed for Spitzer to build the BIC, despite the two missions being implemented using different core Spike models. For FUSE, “hemisphere campaigns” were laid out, around highly constrained observations, in a similar fashion to the instrument campaigns of Spitzer.

## **Spike Lessons Learned**

The Campaign Scheduling concept developed for Spitzer proved to be a very useful concept for LRP, and was deployed successfully for FUSE, as described above.

Throughout the development of Spike, we have noted that missions never end up being “as planned” and that one should develop with post-launch change in mind (Johnston and Miller 1994) and (Hawkins 2009). This was highlighted in our experience with Spitzer just as it was with HST and FUSE: post-launch, we found that, at times, the mission requirements weren’t quite as expected pre-launch, or that manual overrides were necessary due to some wrinkle (or bug) that wasn’t anticipated. Examples include manual instrument campaign overrides and new “instruments” (not physical but virtual).

## **Plan-IT II Adaptation**

SIRPASS, the Spitzer adaptation of Plan-IT II, incorporates a full complement of system-level models and activities. The software includes activity type definitions for all of the approved Spitzer request types, models for each resource whose utilization impacts the schedule, and a variety of integrated models and interfaces that support the tasks and decisions required of the OPST. LRP is supported by the Spitzer Spike module from STScI along with a variety of reports that detail the often-complex constraint relationships between requests. STS, which forms the majority of the week-by-week OPST tasks, is facilitated through the use of an automated scheduling algorithm derived from the HST scheduling operations. Finally, SIRPASS supports the weekly production of formal schedule review and stored sequence products.

## Development Environment

SIRPASS is implemented following the ANSI Common Lisp standard using Allegro CL, version 8.2, from Franz, Inc. The integrated development environment includes the XEmacs editor, incremental compilation, dynamic linking and loading of shared libraries, foreign function interfaces to C and other languages, and runtime patching. An Allegro-supplied ODBC module, AODBC, which supports either the version 2.0 or the version 3.0 ODBC definitions, manages connections to the SODB. Source code is configuration managed at the SSC using CVS, the Concurrent Versions System.

## Plan-IT II Extensions

SIRPASS uses a number of available extensions to the Plan-IT II core system to support its Spitzer functionality. The SIRPASS team developed many of these extensions as generally useful extensions to the core system, including:

*CSPICE*: Provides access to NAIF CSPICE Toolkit functionality for calculating light time between Earth and spacecraft for timing downlink and uplink activities and for converting between NAIF body IDs and names. Access to additional CSPICE capabilities is available through bindings to the SpiceZfc library.

*DSMS*: Provides access to JPL's Deep Space Mission System artifacts, such as Command Definition Language files containing Spitzer's Command Dictionary, definitions of DSN complex and antenna resources, and DSN schedule files such as Station Allocation Files and Viewperiod Files. The DSMS extension also provides the capability to save Plan-IT II activity schedules in Spacecraft Activity Type File (SATF) and Spacecraft Activity Sequence File (SASF) formats.

*Emacs*: Provides additional Allegro CL XEmacs IDE integration with Plan-IT II for editing files as part of the planning and scheduling process and for scripting and otherwise controlling Plan-IT II from a programmable editor.

*Flat*: Provides support for reporting planning and scheduling information in CSV and other flat file formats for integration with common spreadsheet analysis software.

*Gnuplot*: Provides access to the popular graphing utility for plotting arbitrary planning data such as time series data or other data types.

*HTML*: Provides support for producing many types of planner output in HTML format, including Plan-IT II reference documentation, timeline summary and activity detail information.

*Model Parameter:* An extension for defining models and the parameters that influence their evaluation. Includes support for runtime modification of the parameter values and automatic reevaluation of the model. The end-user can

model different operations scenarios by saving and loading predefined sets of parameter values.

*Reports:* A windowed report-generating system that supports nicely formatted columnar reports with headers and footers. Reports can be viewed on screen, refreshed with updated data, saved to text files, and printed to PostScript printers. Adapters can easily define new reports, and many reports allow for end-user customization of content and layout.

*Test:* An extension of the Allegro CL test harness to support interactive and scripted unit testing and reporting for Plan-IT II and adapted systems.

## Integrated Models

In order to obtain accurate predictions in support of efficient scheduling, SIRPASS includes a number of independently developed models.

*The Spitzer Slew Model:* An efficient schedule allocates no more time than necessary to the slewing of the telescope from one location to the next. Should too little time be allocated to slewing, the telescope would be in danger of not finishing an observation before an established deadline. The Pointing Control System (PCS) slew model brings the on-board control algorithms into the realm of the ground data system, enabling a variety of ground-based software applications to accurately predict Spitzer slew times. Slew time, and the time it takes to settle upon a target, is non-deterministic, and therefore the PCS slew model provides a best guess as to the actual time these activities will take during on-orbit execution. The slew model is implemented in the C language and integrated into SIRPASS.

*The Spitzer PCRS Catalog Tool:* The Spitzer Pointing Calibration and Reference Sensor (PCRS) Star Catalog Tool is ground-based software used in the Spitzer Uplink Process to select appropriate stars from the PCRS Guide

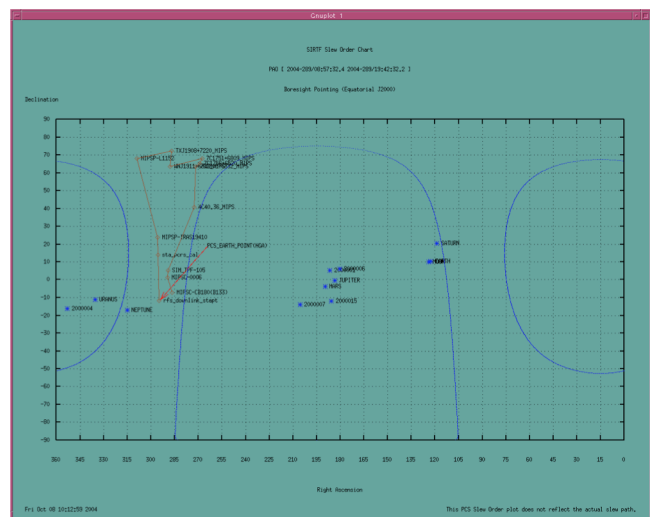


Figure 3. Plotting Slew Order

Star Catalog (GSC). PCCRS allows Spitzer to obtain more accurate pointing by referencing the locations of well-known stars. PCRS calibration activities must be performed periodically during the schedule. In order to not significantly decrease the efficiency of the schedule, SIRPASS uses the Spitzer PCRS Catalog Tool to select a calibration star that is close to the scheduled slew path.

*The Spitzer Spike Module:* SIRPASS supports the LRP process through its integration of a Spitzer-adapted Spike from STScI. Spitzer Spike operates in one of two major modes. In both modes, Spitzer Spike calculates a Plan Window for each request. A Plan Window is a series of time intervals where a request may be scheduled. The request's Plan Window is consistent with the BIC and all constraints in which the request participates. In some cases, the Plan Window is empty indicating that the request cannot be scheduled.

Although Spitzer Spike can produce an optimal BIC from a set of provided requests, the provided BIC is often difficult to justify due to its somewhat uneven allocation of time between instruments and its tendency to occasionally drop an instrument out of regular rotation. Instead of using the Spitzer Spike mode that produces a BIC, the OPST instead handcrafts a BIC for the second major Spitzer Spike mode, which utilizes the provided BIC as input to the LRP process.

*The Mars Pathfinder Heritage Data Model:* Several recent spacecraft have inherited the Mars Pathfinder (MPF) telemetry model that is partially based upon the Consultative Committee for Space Data Systems (CCSDS) concept of Application Process Identifiers (APIDs), wherein an APID is associated with an onboard application process that generates telemetry. For instance, packets designated as APID 20 for Spitzer contain IRAC instrument data (from AORs and IERs) specified by the ground to be sent first (for instance, calibration data). All downlink data are packetized and assigned to APID queues, from which data is downlinked in FIFO (first in first out) order.

Within a single downlink session, APIDs are prioritized according to a two-dimensional priority matrix called a Downlink Priority Table (DPT). The DPT is used to make sure the most important data gets in the front of the downlink stream, regardless of when it is acquired, with the proviso that downlink from individual queues is FIFO. In the DPT, APIDs can be assigned to completely override others in priority (that is to say, completely prevent other APIDs from getting any downlink so long as any data is left in the higher priority APID), or they can be assigned to share a priority level on a percentage-of-bits basis. Different downlink sessions can be governed by different DPTs, and within limits, the DPT organization is negotiable, although generally once a set of DPTs is tested, it is not modified during flight (Mars Pathfinder 1996).

The algorithms for applying a DPT to a volume of data categorized into APIDs is a general one, and applicable to most missions that have adopted the convention. The Spitzer model for APID/DPT use is essentially unchanged from the Mars Pathfinder implementation. The APID/DPT specification format is simple, however the system-level effects of the specification are often not fully understood until some flight experience has been gained through normal operations and, possibly, safe-mode or standby operations. Where the APID/DPT system becomes highly complex is in its effects on activity-specific data downlinks. The APID/DPT system makes it difficult to answer questions such as “when does data from activity X arrive on the ground” and there are currently no scheduling

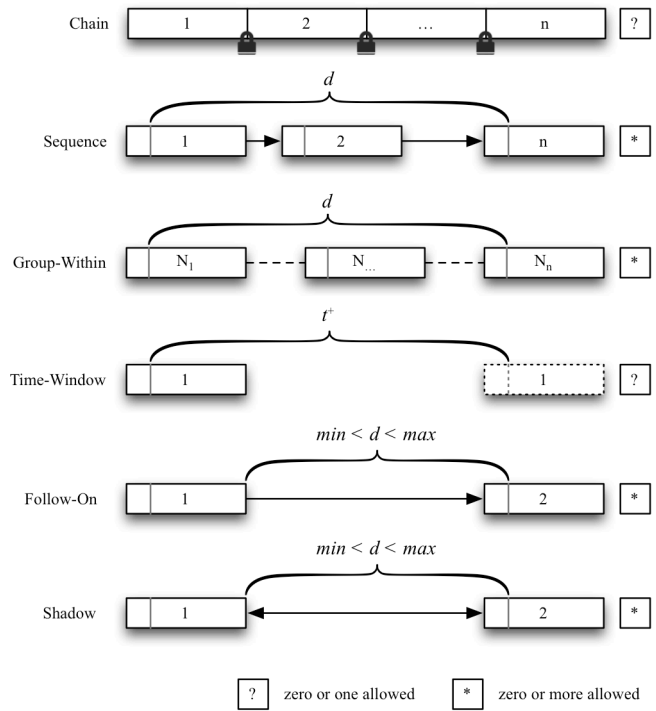


Figure 3. Request Constraints

tools available which can satisfy a constraint expressed in the form “schedule this activity so that its data arrives on the ground no later than time X.”

### Observer-Imposed Constraints

Spitzer allows observers to request constraints on their observations. Because observer-imposed constraints — combined with operational constraints such as target visibility and telescope roll limitations — make it more difficult to schedule observations in an efficient manner, it is essential that observers keep constraints to a minimum. It is recognized that some scientific programs can only be accomplished through use of observer-imposed constraints; however, Spitzer makes it a priority to ensure that these



constraints are thoroughly and soundly justified in the observing proposal.

SIRPASS supports a constraint language similar to that defined by the Space Telescope Science Institute. The constraints (Figure 3) available to observers are:

*Chain, an ordered, uninterruptible group:* The AORs will be executed in the order specified with no interruptions in the chain. The chain may not exceed the maximum allowable duration for a single AOR.

*Sequence, an ordered, interruptible group:* A sequence constraint is similar to, but less stringent than, a chain constraint. The AORs will be executed in the order specified and a duration in which they should be completed is specified. The sequence constraint should only be used when the science requires sequential ordering of the AORs. For AORs in which the order of observation is not important, a “group-within” constraint should be used instead of a sequence constraint.

*Group-within:* A group-within constraint specifies that a group of AORs will be executed within a specific length of time but with no particular starting date or time constraint. Once the first AOR has been executed, the rest of the AORs in the group will begin within the specified time interval. They may be executed in any order within the time interval. This is similar to a sequence constraint, but the observations may be executed in any order.

*Time-window:* Time-window constraints consist of defining a window or series of windows for the start time of an AOR. If the open and close times of the window are specified to be identical for a moving target, then the AOR will be scheduled as an absolute time observation at that time, and will be executed at that time or no more than three seconds later. Spitzer’s scheduling architecture generally operates on relative time; inertial target AORs will simply run one after the other. Timing constraints for inertial target AORs are macroscopic (days, weeks, months), not microscopic (seconds, minutes, hours).

*Follow-on:* A follow-on constraint executes the “follow-on” AOR within a specified time range after a particular initial AOR has been executed. (In other words, from the end of the first AOR body to the beginning of the second AOR body.) Follow-on constraints are useful for periodic observations of a target where the interval between observations is relatively short (hours to a small number of days).

*Shadow:* The shadow constraint is a special case of the follow-on constraint, and is used to obtain background measurements for moving targets. The primary AOR is executed as specified. The shadow AOR will be executed to repeat the track of the primary observation. The selected AOR parameters must be identical in the two AORs. The shadow may be executed before or after the primary AOR. Note that the shadow does not re-observe the target at a

later date, but rather the background of the primary observation (Spitzer Space Telescope 2012).

These observer-imposed constraints — along with spacecraft and operations constraints — are considered during both LRP and STS processes. During LRP, Spitzer Spike considers constraints while generating rough plan windows for requests that are consistent with the BIC; during STS, SIRPASS performs detailed, recursive tree-walking time arithmetic to calculate the schedulable time interval for each constrained request.

### Science Operations Database

The Science Operations Database (SODB) at the SSC stores information about all requests for Spitzer observations. The SODB is central to all science operations and contains a summary of all tracking and status information related to a given request. SIRPASS obtains information about requests for observations from the SODB and provides schedule and status updates to the SODB. The SODB interface was a major new addition to SIRPASS.

The SODB classifies requests for observations into three categories:

*Astronomical Observation Request (AOR):* A request to observe a target using one of Spitzer’s eight observing modes. Observers generate AORs by providing parameters to one of eight Astronomical Observation Templates. AORs are used for science observations and most instrument calibration activities.

*Instrument Engineering Request (IER):* A request to operate a Spitzer instrument in a way that cannot be accomplished through one of the Astronomical Observation Templates. Spitzer engineers use IERs to perform instrument operations in a way that is not available to observers, generating some IERs by parameterizing Instrument Engineering Templates (for example, some instrument calibration requests). Other IERs are fixed sequences that do not require parameterization (for example, instrument turn-on and shut-down).

*Spacecraft Engineering Request (SER):* A request to perform a spacecraft activity not requiring the use of instruments (for example, data downlink). Spacecraft engineers generate some SERs by parameterizing complex activity templates (for example, Wide Angle Sun Sensor calibrations), while other SERs are fixed sequences that do not normally require parameterization (for example, data downlink).

We can categorize the information stored by the SODB thusly:

*General:* Includes title, program and observer affiliations, comments, instrument, and observation type.

*Resource Estimates:* Includes predicts of execution duration and telemetry data volume.



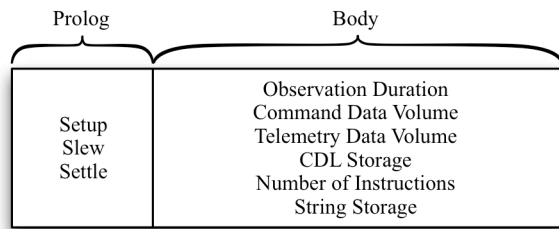


Figure 4. Request Estimates are provided by both SIRPASS (prolog) and AIRE (body).

**Target Specification:** Includes specification of initial and final telescope pointing location for the observation.

**Scheduling Information:** Includes scheduled request start and stop times and inter-request constraints.

The AIRE system calculates resource estimates (Figure 4) that are used later in the scheduling domain. Targeting is the responsibility of the slew model, used by both SIRPASS and AIRE. Request expansion to library calls is the responsibility of AIRE; this keeps SIRPASS highly interactive without bogging it down expanding activities to individual commands. Request expansion to

suitable for scheduling an observation or is not suitable. There are few situations in which one specific possible scheduling of an observation is considered “better” than any other. Because suitability is a simple yes/no proposition, SIRPASS converts these propositions into *time windows* and performs complex time window set operations and calculations in order to determine the appropriate location for the observation. Because of the complex *constraints* between sets of observations, SIRPASS employs a recursive tree-walking algorithm that accounts for the position of the observation within its constraint as well as the schedule state (scheduled or unscheduled) of the constrained observations.

Each request may participate in either “zero or one,” or “zero or more” of each type of constraint. A request may participate in zero or one Chain and Time-Window constraints, and in zero or more Sequence, Group-Within, Follow-On, or Shadow constraints. A request cannot participate in both a Sequence and a Chain constraint; if this occurs, the Chain constraint is ignored.

It is common for programs with hundreds of requests to be fully constrained, that is, all the requests in the program participate in at least one constraint. These systems of

Young Stellar Object Variability (YSOVAR): Mid Infrared Clues to Accretion Disk Physics and Protostar Rotational Evolution (pID 60014 go)

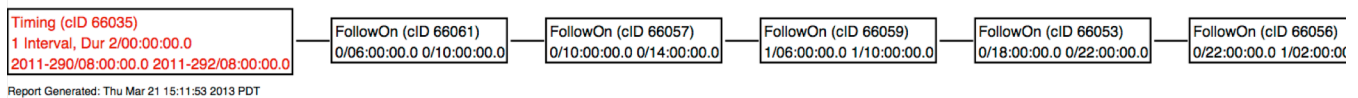


Figure 5. Depicting Constraints

spacecraft commands and the modeling of those commands is the responsibility of SEQGEN, a computationally demanding process.

Resource estimates are generated several times during the planning and scheduling process, as additional information becomes known about the timing of the observation. Requests are divided into estimation regimes: the Prolog, which is dominated by the acquisition of the observation target, and the Body, which is dominated by the acquisition of instrument data while pointing at or mapping the target. Request body resource estimates can be generated with some accuracy, without precise knowledge of execution timing, although the geometry of the target with respect to the observatory can influence the duration of the many slews that make up larger mapping-type observations. The execution time required by the request prolog, however, is heavily determined by the previous request in the schedule, and cannot be precisely determined until the request is assigned a tentative execution time on the SIRPASS timeline.

### Constraint Satisfier

Most constraints to scheduling Spitzer observations are binary: a segment of time, or time interval, is either

constraints can become quite complex and are difficult to visualize. Rather than depict the requests themselves and their relationships to other requests in the constraint, SIRPASS renders the constraints and their relationship to other constraints via the involved requests (Figure 5).

To improve performance, SIRPASS caches intermediate and final results of its tree-walking algorithm, and employs a “just in time” approach to requesting information from the SODB.

### Greedy Scheduler

After filling the timeline with those activities whose timing is mostly prescribed, such as some types of SERs (for instance, downlinks), and some types of IERs, (for instance, instrument transition requests), the rest of the schedule is ready to be filled with any of the available unscheduled AORs that have been assigned to the current week during the LRP process. In order to fill the timeline in the most efficient manner possible, SIRPASS makes use of a variant of the Greedy algorithm first described in (Samson 1998). In the SIRPASS variant, the timeline is analyzed for openings where no activity currently is scheduled. At the start of each opening, SIRPASS

calculates a cost for scheduling each available unscheduled request at that time and selects the least costly activity to start at that scheduled time. The process continues until all available unscheduled requests have been scheduled, or no additional available unscheduled requests can be scheduled. The cost for scheduling a specific request at a specific time is the sum of several terms; each term is multiplied by its own coefficient and scaling factor. Amongst the terms used are costs associated with slewing to the target from the previously established location on the sky, a (very high) cost for breaking a chain constraint, costs associated with how much overall time is left to schedule the request and whether or not the request is considered highly constrained. Evaluating the cost formula yields a single number, and the request with the lowest cost number is selected for scheduling at the considered time.

Because the scheduling of different weeks occurs in parallel, the scheduling process emphasizes the need for OPST members to communicate impacts to the schedule to those team members scheduling adjacent weeks. Scheduling a request in week  $n$  may result in its follow-on request needing to be scheduled in week  $n+1$ . When this occurs, it is termed a “must go” request in week  $n+1$ ; SIRPASS supports this process by generating “must go” and other constraint-related reports as part of the scheduling process. Should constraints need to be broken, constraint waivers can be obtained from the SSC’s Observer Support Team.

### **Scheduling Rules**

SIRPASS extended the Plan-IT II core with a new way to instantiate activities on the timeline: the New Activity by Rule command. The user invokes the New Activity by Rule command on any number of predefined activity types, and SIRPASS arranges for Adapter-supplied rules to be applied in a predetermined order.

For example, during the Nominal mission, schedulers used the New Activity by Rule command to schedule Cryogenic Telescope Assembly (CTA) makeup heater commands according to a complex model of thermal performance. Early in the mission, spacecraft engineers made rules designed to keep the telescope at a steady temperature, and the CTA makeup heater-scheduling rule ensured that heater levels were switched each time an instrument was powered on or off. Later in the nominal mission, engineers decided that on-board cryogen could be more efficiently utilized if there were a single pulse of heater use (which caused the cryogen to sublime and on its escape through the shroud of the telescope, cool the

baffle) followed by a period where the telescope’s temperature was allowed to float. For simplicity, parameterization of scheduling rules was not exposed through the operations interface, but rules could be adjusted to accommodate temporary changes in desired behavior.

Scheduling rules were also available for data accountability housekeeping, IRU calibrations, momentum desaturation, downlink, and star tracker pointing reference calibrations.

### **Decision Support Tools**

The *ViewingZones* tool was developed by Mark Garcia, a JPL Mission Planner, and provides a quick way to visualize the complex relationship between the OPZ, Earth and other important celestial bodies, and the observation targets for a given Period of Autonomous Operations (PAO), the schedule time between downlinks. The *ViewingZone* tool is written in Quick, a scriptable interactive, programmable desktop calculator that is part of the Mission Analysis Software Library (MASL) tools suite developed by JPL’s Mission Analysis Section, and a precursor to the current MONTE system.

SIRPASS includes many other decision support tools, including time calculators, schedule content summary reports and request constraint visualization reports.

### **Sequence Architecture**

Spitzer employs a “master” and “slave” sequence architecture wherein the master sequence controls the behavior of the overall sequence load, spawning slave sequences at appropriate times. There may be one or many master sequences per week, but only one master may be executing at a time. Slaves may be as small as one AOR or can be several AORs, IERs, or other activities packaged together. Spitzer also uses a set of functions called sequence blocks, which are parameterized, reusable relative-timed sequences. Parameterization allows execution of the block to occur differently with each use. Blocks may accept parameters, return values, or both, and — when loaded into a virtual machine (VM) engine — may be used by the master or slave sequences in AORs, IERs or SERs. To solve certain VM storage issues, there may also be a construct called the slave library. This is similar to a block library but contains only single-use activities at the AOR and IER level. Slave sequences may contain all the commanding necessary for the execution of their activities or they may call blocks from the block library and slaves from the slave library.

Before any request can be executed in flight, it must first be packaged into logical units that will control the timing and structure of the plan. In SIRPASS, the bundling of requests into this structure is called *sequence packaging*. There are several sequence packaging structures used in the mission, but one structure predominates the nominal mission phase.

Packaging is based upon a series of rules which determine which types of requests can co-exist in the same sequence and which must be in separate sequences. For the purposes of packaging, the following types of requests exist:

- Fixed-Target AORs
- Moving-Target (Absolute) AORs
- Power-Transition IERs
- Other IERs
- True SERs
- Pseudo SERs

A graphical representation of the rule set is shown in Figure 6. In addition to packaging according to the compatibility of adjacent request types, the packaging of sequences was also constrained by the need to keep slave library sizes compatible with flight system requirements. For instance, a single VM module is limited to storing 80,000 bytes of command codes. The AIRE system is responsible for estimating the number of command codes required by a request, and SIRPASS factors in a running total of packaged command codes when determining if additional requests can be packaged in the current module. Other module-related packaging restrictions include limits on string storage and total number of VM instructions.

## Spitzer Development and Testing

Development of SIRPASS began in 1997 with the selection of Plan-IT II as the core science planner system. During development, SIRPASS was used for a number of reference mission studies and in testing of the Spitzer ground data system both at the SSC and at JPL. In many ways, the development of SIRPASS mirrored the development of the mission system, and many of the capabilities of SIRPASS were developed to aid in the development and testing of the mission system.

The Model Parameter extension, for instance, was used during the planning of the initial phases of the mission to globally adjust the duration of IOC activities as part of a worst-case scenario designed to place an upper bound on

		Next Request is...					
		SER		IER		AOR	
		Pseudo SER	True SER	Power Transition	Other	Moving Target	Fixed Target
Previous Request is...	Pseudo SER	✓*	✓*	✓*	✓*	✓*	✓*
	True SER	✓*	✓	✗	✗	✗	✗
IER	Power Transition	✓*	✗	✓†	✗	✗	✗
	Other	✓*	✗	✗	✓	✗	✓
AOR	Moving Target	✓*	✗	✗	✓	✗	✓
	Fixed Target	✓*	✗	✗	✓	✗	✓

\* Only if one or both of the Pseudo-SERs are always enslaved.  
† Only if both Power Transition IERs are for the same instrument.

Figure 6. Request Packaging is a Pairwise Function

the length of IOC. During an extensive period of learning how to operate the Spitzer Spike module, the Model Parameter extension was used to define and enforce Spike performance parameters; parameterized studies were performed to characterize the performance of the system under different sets of underlying assumptions about instrument ordering, maximum instrument campaign length, etc. These types of studies continued throughout the mission as the performance of the observatory changed over time and the population characteristics of the proposed observations also changed.

## Command and Data Handling

Early in ground testing of Spitzer's Command and Data Handling (C&DH) subsystem, it was discovered that the Fast Uplink/Downlink Card (FULDL) was not capable of reliably processing transfer frames for downlink at 2.2 Mbps, Spitzer's highest data rate, nominally scheduled for use during the prime mission. As a result, the C&DH team created the Storage Unit (SU), a data structure holding sixteen transfer frames. The on-board data storage overhead associated with SUs was minor and so did not invalidate SIRPASS's data models. However, SUs and certain idiosyncratic behavior of the flight data storage subsystem necessitated the use of extra ground analysis tools to ensure that on-board data storage was not exceeded. Tools were developed to track on-board data storage usage and procedures were developed to free on-board storage space when the associated data had been received and processed on the ground (Sarrel et al. 2006).

SIRPASS was extended to help make this process more manageable, but a completely satisfying solution was never achieved. The resulting system was a balance between process and tool support and will likely remain so until the end of the Spitzer mission.

## Flight Operations Experience

Experience in operating the flight system invariably leads to changes in procedures and capabilities that are reflected in changes to ground tools. The manner in which these changes are accommodated is often determined by time, budget and the flexibility of the automated systems being used to support mission operations.

### Slewing Efficiency

Due to several operational requirements such as solar avoidance and solar panel illumination, Spitzer is constrained at all times to point the telescope no closer than 82.5 degrees toward and no further than 120 degrees away from the Sun. This defines the Operational Pointing Zone (OPZ) of the Observatory. The OPZ was reduced by 2.5 degrees in January 2004, which allowed for an increase in the maximum slew rates used by Observatory.

### Nominal Operations

It was during the Nominal Operations mission phase that the science operations team discovered the potential for *bright objects* to leave latent images on the instrument sensors. To alleviate the effects, it was decided that observations of bright objects should be placed immediately before downlink activities. The definition of a bright object was complicated. Latent images were caused by a combination of factors including the inherent brightness of the observed target, the selection of instrument modes and the duration of the observation. The science instrument teams defined the requirements of a special software module to be run during observation expansion and resource estimation that would flag bright objects at the SSC. The flags were then available to SIRPASS for use in scheduling the observations in locations on the timeline that were unlikely to cause problems for subsequent observations. Because the population of bright object observations was thought to be low, it was decided to not offer special automated tools for scheduling such observations. Instead, bright object observations were brought to the attention of the schedulers and were placed on the

timeline manually early in the STS process, before other observations were added to the timeline.

## Extended Mission and Warm Mission

The observatory operated extremely well and efficiently throughout its nearly 5-year 9-month cryogenic lifetime from launch on 25 August 2003 until the helium was exhausted on 15 May 2009. Without helium, radiative cooling allowed the mirror and cryostat temperatures to stabilize at about 26 K, too warm for operation of either IRS or MIPS, but sufficiently cold for IRAC observations at 3.6 and 4.5  $\mu\text{m}$  (Mahoney et al. 2010). SIRPASS was able to accommodate the new Warm Mission phase with minimal change by eliminating the use of IRAC instrument windows and by changing the layout of the software interface to remove display space devoted to MIPS and IRS.

### Spitzer Observing Efficiency

For the purpose of accounting, we categorize efficiency as the sum of the amount of time spent executing science observations, calibrating science instruments and slewing between science targets, as a fraction of all observatory time. As shown in Figure 7, beyond the initial “learning curve” of operations and notwithstanding spacecraft anomalies that halt all science activities, Spitzer has achieved efficiency ratings of 90% to 95%.

### What the Future Holds

SIRPASS performance was originally designed with a 500-request per week timeline and an operations process that took four weeks to shepherd one weeks’ worth of

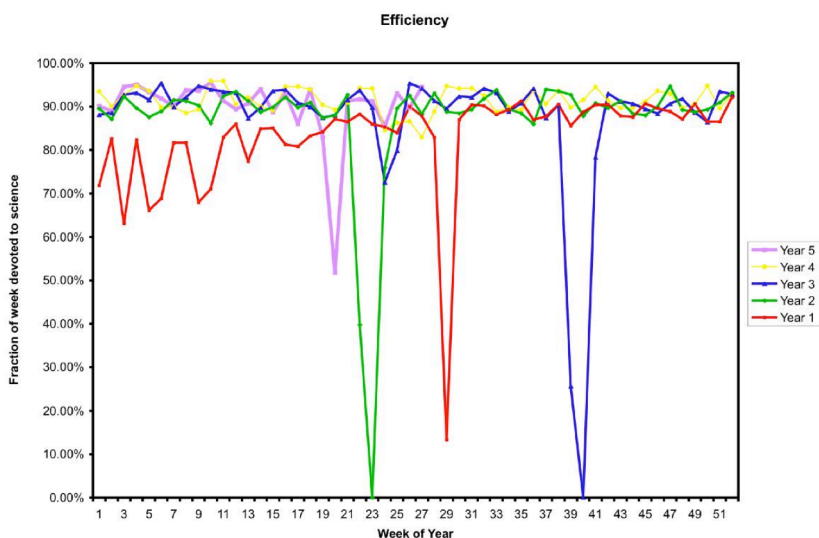


Figure 7. Spitzer Efficiency Through May 2008

observations through the entire planning, scheduling, sequencing, and uplink process with a staff of four schedulers at the SSC. As of this writing, during the Warm Mission, the scheduling process is occurring over two-week intervals using a staff of three schedulers. Over the course of the mission to date, Spitzer has returned over 12.5TB of science data and executed over 78,000 observations over 63,000 hours. Although most weeks have fewer than 500 observations, SIRPASS has scheduled some weeks with up to 600 observations and others with as few as 60 observations.

In late 2013, spacecraft operational constraints must change for the mission to continue. At this time the geometry between Spitzer, Earth, and the sun would cause a violation of the current OPZ when the spacecraft attempts to point its high-gain antenna (HGA) towards Earth for downlink. Considering the narrow beam-width of the HGA, if the OPZ were left unchanged downlinks would not be possible after the 30° limit was reached. Therefore when this limit is reached the OPZ will be expanded to 38° in pitch for downlinks only. All other OPZ constraints will remain unchanged and all science observations shall remain inside the original OPZ constraints (Mahoney et al. 2012). Before approving the change in OPZ limits, the spacecraft engineering team first have to perform tests to characterize and model the Observatories thermal and pointing sensor performance during dwells beyond the current limits.

## Conclusions and Lessons Learned

As a wise man once said, “The unexamined life is not worth living.” On that basis, in this section we take a look at the life of the Spitzer Space Telescope project through the lens of SIRPASS, from its initial development through its use today in Spitzer’s Extended (Warm) Mission.

First off, many software and process design choices are made in the context of assumptions that seem reasonable at the time. However, when those assumptions eventually change and the design choice is revisited, it is not unusual to find that it is very costly to change the design and implementation to accommodate the change in assumption. There are a number of reasons for this, including the build-up of process and software external to the system under consideration that result in a ripple effect expanding from the original system to many other linked systems. Sometimes simply knowing that a “limit” is not truly a limit, but a current knowledge boundary, can help drive a design that can more easily accommodate a change to the limit. I refer to this as the “Here be Dragons” effect, a phrase used to denote unexplored areas on medieval maps.

Secondly, don’t be surprised — or shocked — at the longevity of any piece of software or hardware used in

mission operations. Budgetary and safety constraints will invariably lead to limits a project’s ability to upgrade ground data system hardware and software. Personnel turnover (including, but not limited to, terminations, promotions and death) will change the cost / schedule / risk equations used to determine whether to keep, scrap or modify any single piece of software or hardware. Plan for the full range of possibilities: your software may never get used in operations (for example, in the case of a launch failure) or it may be used for far longer than you ever anticipated. When balancing development priorities and design, consider three longevity cases: failure to launch, nominal mission duration, and order-of-magnitude mission duration. In the case of Spitzer, the nominal mission duration was 2.7 years, so the “order-of-magnitude” mission duration would be about 27 years. Considering the longevity of spacecraft, Voyager being a case in point with mission duration of over 35 years, Spitzer mission duration of 27 years is not completely whacky.

As flight projects age, the cost of maintaining a high-fidelity integration and test facility for ground software becomes costly and is often an early target for the inevitable budget cuts that are associated with extended mission operations. Differences between operations and integration and test environments can cause false positives during regression testing of ground software. It is often the case that the highest fidelity test environment is the operations environment itself. When designing software systems, consider the possibility of testing the systems on a “non-interference” basis within the operations environment. For instance, designing a “global switch” that prohibits the writing of data to operations data stores, but instead simulates the operation without affecting any operations system. A software system so isolated can be safely tested in an operations environment. Another approach is to consider the strategic addition of test assets to an operations environment. For instance, a replicated database running on a test server in the operations environment can act as a stand-in for the operations database in many cases.

As originally envisioned, the Spitzer uplink process was to be executed by a single, integrated team, the Integrated Mission Planning and Scheduling Team (IMPST). The prototype for IMPST was the Mars Pathfinder scheduling team which, using MPF Plan-IT, created sequences for the Mars Pathfinder lander and processed those sequences through the complete uplink process. In 2001, at or around project Critical Design Review (CDR), it was decided to split the Spitzer uplink process between two teams, one at the SSC that would be responsible for assembling the spacecraft sequences, and one at the Jet Propulsion Laboratory that would be responsible for detailed modeling of the sequences and conversion to binary products for transmission to the spacecraft.

## Virtual Machine Language

In 1999, the Spitzer project adopted the use of Virtual Machine Language (VML), a procedural sequencing language that offered a number of advancements over more traditional declarative sequencing languages, including the support for conditional branching and function block return values (Grasso 2003). VML, although in its early stages of development as a flight system component, was seen as a much better approach to accommodating Spitzer's non-deterministic execution due to slew-and-settle uncertainties, than the more traditional sequence execution approaches (Grasso and Lock 2008). Given the rich set of traditional programming language capabilities offered by VML, one might be excused for thinking that the proper course of action would be to adopt this new style of procedural sequencing for Spitzer. However, our ground tool capability for verification and validation was no match for the advanced capabilities of VML, and we found that if we used the full complement of VML capabilities on board, we would have no fast and effective way of modeling the flight system behavior on the ground. Therefore, it was decided to severely limit our use of advanced VML capabilities in ways that 1) could be reasonably modeled on the ground, and 2) could be shown to provide improvements in system operability without undue complexities. It is often a wiser course of action to restrain the whole-hearted adoption of new technology for a more measured and balanced approach.

One early use of VML was in avoiding the over-filling of spacecraft mass memory, a situation that would have consequences for spacecraft safing procedures. Each AOR was expressed in VML with a logical wrapper that compared a ground-generated estimate of the data volume generated by the AOR against the flight system's record of remaining free memory. If the comparison revealed insufficient free memory, the observation was skipped. Since this observation skipping behavior was consistent with the overall operations concept for the telescope, no special accommodations for modeling and verification were required on the ground. The skipped AOR was simply repeated at a later time. To date, only 10 to 20 AORs have been skipped in this way, and a subsequent update to AOR definitions has replaced the original logic with new logic that will skip individual data collection events (DCEs) within an AOR rather than skipping the entire AOR.

## Acknowledgement

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The authors wish

to gratefully acknowledge the contributions of William "Curt" Eggemeyer, Laurence Kramer, and Tatiana Goldina to the creation of SIRPASS.

## References

- Adler, D. S. and Workman, W. M. III. 2008. Maximizing Scientific Return for the Hubble Space Telescope in a Post-SM4 World. In *Observatory Operations: Strategies, Processes, and Systems II. Proceedings of SPIE (7016)*. Bellingham, Washington: SPIE.
- Barba, S. J.; Garcia, L. J.; McElroy, D. B.; Mittman, D. S.; O'Linger, J. C.; and Tyler, S. R. 2006. Planning and Scheduling the Spitzer Space Telescope. In *Observatory Operations: Strategies, Processes, and Systems. Proceedings of SPIE (6270)*. Bellingham, Washington: SPIE.
- Barba, S. J.; Garcia, L. J.; Levine, D. A.; McElroy, D. B.; Mittman, D. S.; O'Linger, J. C.; and Tyler, S. R. 2005. Spitzer Space Telescope Observatory Planning and Scheduling Team. *Proceedings of the IEEE Aerospace Conference*. Manhattan Beach, CA: IEEE Aerospace Conferences.
- Bliss, D. A.; Voskanian, V.; and Weise, T. 2006. Spitzer Space Telescope Sequencing Operations Software, Strategies, and Lessons Learned. *SpaceOps: Rome, Italy*.
- Calvani, H. M.; Berman, A. F.; Blair, W. P.; Caplinger, J. R.; England, M. N.; and Roberts, B. A. 2004. The Evolution of the FUSE Spike Long Range Planning System. In *Proceedings of the Fourth International Workshop on Planning and Scheduling for Space*. Darmstadt, Germany.
- Chien, S.; Rabideau, G.; Willis, J.; and Mann, T. 1999. Automating Planning and Scheduling of Shuttle Payload Operations. *Artificial Intelligence Journal* 114:239-255.
- Eggemeyer, W. C.; Grenander, S. U.; Peters, S. F.; and Amador, A. V. 1997. Long Term Evolution of a Planning and Scheduling Capability for Real Planetary Applications. In *Proceedings of the First Workshop on Planning and Scheduling for Space*. Oxnard, California.
- Giuliano, M.; Hawkins, R.; and Rager, R. 2011. A Status Report on the Development of the JWST Long Range Planning System. In *Proceedings of the Seventh International Workshop on Planning and Scheduling for Space*. Darmstadt, Germany.
- Giuliano, M. 1998. Achieving Stable Observing Schedules in an Unstable World. In *Astronomical Data Analysis Software and Systems VII*, 271-274. San Francisco, California: Astronomical Society of the Pacific.
- Grasso, C. A. and Lock, P. d. 2008. VML Sequencing: Growing Capabilities over Multiple Missions. *SpaceOps: Heidelberg, Germany*.
- Grasso, C. A. 2003. Techniques for Simplifying Operations Using VML (Virtual Machine Language) Sequencing on Mars Odyssey and SIRTf. *Proceedings of the IEEE Aerospace Conference*. Manhattan Beach, CA: IEEE Aerospace Conferences.
- Hawkins, R. E.; Jordan, I. J. E.; and Giuliano, M. E. 2009. Applying Lessons Learned in Planning and Scheduling the Hubble Space Telescope to the James Webb Space Telescope. In *Proceedings of the Sixth International Workshop on Planning And Scheduling for Space*. Pasadena, California.
- Johnston, M. D. and Miller, G. E. 1994. Spike: Intelligent Scheduling of Hubble Space Telescope Observations. *Intelligent*

- Scheduling*, Fox, M. and Zweben, M. eds. San Francisco, California: Morgan Kaufmann Publishers.
- Kramer, L. A. 2000. Generating a Long Range Plan for a New Class of Astronomical Observatories. In *Proceedings of the Second International Workshop on Planning and Scheduling for Space*. San Francisco, California.
- Laine, S.; Silbermann, N. A.; Rebull, L. M.; and Storrie-Lombardi, L. J. 2006. Spitzer Space Telescope Proposal Process. In *Observatory Operations: Strategies, Processes, and Systems II*. Proceedings of SPIE (6270). Bellingham, Washington: SPIE.
- Mahoney, W. A.; Effertz, M. J.; Fisher, M. E.; Garcia, L. J.; Hunt, J. C. Jr.; Mannings, V.; McElroy, D. B.; and Scire, E. 2012. Spitzer Operations: Scheduling the Out Years. In *Observatory Operations: Strategies, Processes, and Systems IV*, Peck, A. B.; Seaman, R. L.; and Comerón, F., eds. Proceedings of SPIE (8448). Bellingham, Washington: SPIE.
- Mahoney, W. A.; Garcia, L. J.; Hunt, J. C. Jr.; McElroy, D. B.; Mannings, V.; Mittman, D. S.; O’Linger, J. C.; Sarrel, M.; and Scire, E. 2010. Spitzer Warm Mission Transition and Operations. In *Observatory Operations: Strategies, Processes, and Systems III*, Silva, D. R.; Peck, A. B.; and Soifer, B. T., eds. Proceedings of SPIE (7737). Bellingham, Washington: SPIE.
- Mahoney, W. A.; Comeau, S.; Garcia, L. J.; McElroy, D. B.; Mittman, D. S.; O’Linger, J. C.; and Tyler, S. R. 2008. Spitzer Scheduling Challenges: Cold and Warm. In *Observatory Operations: Strategies, Processes, and Systems II*. Proceedings of SPIE (7016). Bellingham, Washington: SPIE.
- Mars Pathfinder. 1996. Mars Pathfinder & Mars '96 Lander Science Opportunities (website), <http://mars.jpl.nasa.gov/MPF/nasa/pipfaq.html>. California Institute of Technology, Jet Propulsion Laboratory, Pasadena, California.
- Miller, G. E. and Stanley, P. 1998. Applying the Lessons Learned from HST Operations to New Missions. *Observatory Operations to Optimize Scientific Return*. In *Observatory Operations to Optimize Scientific Return*. Proceedings of SPIE (3349). Bellingham, Washington: SPIE.
- Samson, R. J. 1998. Greedy Search Algorithm Used in the Automated Scheduling of Hubble Space Telescope Activities. In *Observatory Operations to Optimize Scientific Return*. Proceedings of SPIE (3349). Bellingham, Washington: SPIE.
- Sarrel, M. A.; Carrion, C.; and Hunt, J. C. Jr. 2006. Managing the On-Board Data Storage, Acknowledgement and Retransmission System for Spitzer. *SpaceOps: Rome, Italy*.
- Sasaki, T.; Kosugi, G.; Hawkins, R. E.; Kawai, J. A.; and Kusumoto, T. 2004. Observation Scheduling Tools for Subaru Telescope. In *Optimizing Scientific Return for Astronomy through Information Technologies*. Proceedings of SPIE (5493). Bellingham, Washington: SPIE.
- Skinner, D. L. 1989. QUICK, An Interactive Software Environment for Engineering Design. In *Proceedings of the 7th Computers in Aerospace Conference*: Monterey, California.
- Spitzer Space Telescope. 2013. NASA Spitzer Space Telescope (website), <http://www.spitzer.caltech.edu>. California Institute of Technology, Jet Propulsion Laboratory, Pasadena, California.
- Spitzer Space Telescope. 2012. Warm Spitzer Observer’s Manual (website), <http://ssc.spitzer.caltech.edu/warmmission/propkit/som/>. California Institute of Technology, Jet Propulsion Laboratory, Pasadena, California.
- Spitzer Space Telescope. 2011. Spitzer Telescope Handbook (website), [http://irsa.ipac.caltech.edu/data/SPITZER/docs/spitzer\\_mission/missionoverview/spitzertelescopehandbook/](http://irsa.ipac.caltech.edu/data/SPITZER/docs/spitzer_mission/missionoverview/spitzertelescopehandbook/). California Institute of Technology, Jet Propulsion Laboratory, Pasadena, California.
- Tyler, S. R., O’Linger, J. C.; Comeau, S.; Garcia, L. J.; Mahoney, W. A.; McElroy, D. B.; and Mittman, D. S. 2008. Rapid Replacement of Spitzer Space Telescope Sequences, Targets of Opportunity and Anomalies. In *Observatory Operations: Strategies, Processes, and Systems II*. Proceedings of SPIE (7016). Bellingham, Washington: SPIE.
- Werner, M. W.; Roellig, T. L.; Low, F. J.; Rieke, G. H.; Rieke, M.; Hoffmann, W. F.; ...and Cruikshank, D. P. 2004. The Spitzer Space Telescope Mission. In *Astrophysical Journal Supplement Series*, 154:1–9. American Astronomical Society: Washington, D.C.