A Status Report on the Development of the JWST Long Range Planning System

Mark E. Giuliano, Robert Hawkins and Reiko Rager

Space Telescope Science Institute, 3700 San Martin Dr., Baltimore, MD 21217, USA
{giuliano, rhawkins, rager}@stsci.edu

Abstract

The development of the James Webb Space Telescope (JWST) long range planning system started in the fall of 2009 and is ongoing. The planner is based on the SPIKE system that has been used to plan Hubble Space Telescope (HST) observations for the past 20 years. This paper gives a status report on JWST long range planning development. The paper first describes how long range planning fits into the larger JWST planning and scheduling architecture, and shows how lessons learned from HST are being applied to JWST. The paper then describes the high-level software components and processes being used to build the JWST long-range planner while supporting HST operations. The final section of the paper outlines several improvements being made to the planner including potential improvements to the least commitment resource model.

Introduction

The James Webb Space Telescope (JWST), often referred to as the successor to the Hubble Space Telescope (HST), is an infrared-optimized space observatory with a 6.5-meter mirror. Launch is currently planned for 2014 to an orbit 1.5 million kilometers from Earth at the Lagrange point 2. JWST's science operations process will be similar to HST's, from observation selection to data distribution. Based on 20 years experience of operating HST, a Planning and Scheduling System is under development for JWST that builds upon and improves what has worked for HST.

This paper focuses on the SPIKE long-range planning component of the JWST planning and scheduling system. The SPIKE long range planner uses a least-commitment approach of assigning windows of possible start times to each task, rather than assigning a discrete start time to each task, while taking preferences and resource usage into account. SPIKE has been used for the last 20 years for HST long range planning, and will be re-used in the same role for JWST. While SPIKE has been used for other Astronomy applications aside from HST, the least-commitment component has not been utilized outside of HST. The code is therefore specialized towards HST and presents an interesting challenge in order to support the JWST mission while maintaining functionality and testability for the HST mission.

This paper gives a status report on on-going JWST development for SPIKE and is organized as follows. The first section presents lessons learned from the HST mission. The next section gives background on the JWST planning and scheduling system, and shows how we have addressed items in the lessons learned section. The next section describes the high level design of the JWST long-range planner and the software engineering processes for refining the HST SPIKE legacy code to support the JWST mission. Following this, we describe enhancements to SPIKE for JWST including potential changes to the least-commitment resource model.

Learning from HST experience

In [Hawkins et. al. 2009], we summarized some lessons learned from the Hubble experience:

1. Provide scheduling information as early as possible, at the time of telescope time allocation, to achieve higher scheduling efficiency later.
2. Provide observation preparation software that enables astronomers to construct feasible programs that do not need to be reworked during planning and scheduling.
3. Implement a process that encourages astronomers to craft efficient (i.e. increased scheduling efficiency) programs.
4. Organize teams to increase effectiveness.
5. Use a two phase planning and scheduling approach of long-range planning and short-term scheduling to achieve efficient usage of the telescope while maintaining a stable plan for observers.
6. Identify efficiency drivers and model them appropriately.
7. Keep the spacecraft model in one place. Instead of modeling spacecraft constraints in multiple

subsystems, try to make one subsystem responsible for computing them.

8. Design systems to be able to easily incorporate changes.

Some of these lessons have already been applied to the HST process (i.e. 2-6), which we are using as a foundation for JWST. Others are concepts that weren't necessarily applied to HST, but are being developed for JWST (i.e. 7, 8).

JWST Planning and Scheduling

The JWST planning and scheduling architecture is given in Figure 1. Observers prepare observations at a remote site using the Astronomer's Proposal Tool (APT) and submit their observations to the planning and scheduling database at the Space Telescope Science Institute (STScI). The Proposal Constraint Generator reads observation data from the database, calculates windows for the observation's physical and astronomer specified constraints, and writes the resulting windows to the database. JWST scheduling is handled in a two-phase process by separate long-range planning and short-term scheduling systems. Both the long-range planner and short-term scheduler read observation data from the shared database and use the constraint windows produced by the proposal constraint generator.

The two phase scheduling process for JWST will be similar to the process used for the Hubble Space Telescope [Giuliano 1998]. In the first phase, the long-range planner assigns observations to overlapping least commitment plan windows that are nominally 60 days long. Plan windows are a subset of an observation's constraint windows and represent a best effort commitment to schedule within the window. In the second phase, plan windows are used to create successive short-term schedules for 22-day upload periods. This two phase process allows a separation of concerns in the scheduling process: plan windows globally balance resources, are stable with respect to schedule changes, and provide observers with a time window so they can plan their data reduction activities. The short-term scheduler provides efficient fine-grained schedules to the telescope.

We now describe how this architecture addresses some of the concerns described in lessons learned 7 and 8.

Shared constraint modeling

In HST, scheduling constraints involving the telescope's orbit or engineering restrictions are calculated by three different scheduling subsystems including SPIKE. These systems were developed over time and have different users and fidelity requirements. Aside from the extra cost of maintaining three separate constraint calculation systems, there is potential for inconsistencies between the systems. Inconsistent constraint data can cause rework in operations often at the last minute when rushing to prepare an observation for flight. For JWST, there is a single subsystem called the Program Constraint Generator (PCG) that calculates constraint windows for all the scheduling tools, including the program preparation tool (i.e. APT), the long-range planner (i.e. SPIKE) and the short-term scheduler. The PCG has detailed models of the telescope and its orbit and calculates all physical and observer specified constraints for a single observation. In other words, PCG calculates what the possible start times are for an observation as if it is the only observation to schedule on the telescope. With PCG, there will be no modeling mismatch between subsystems. This approach also frees JWST SPIKE from the detailed constraint modeling it has to do for HST.

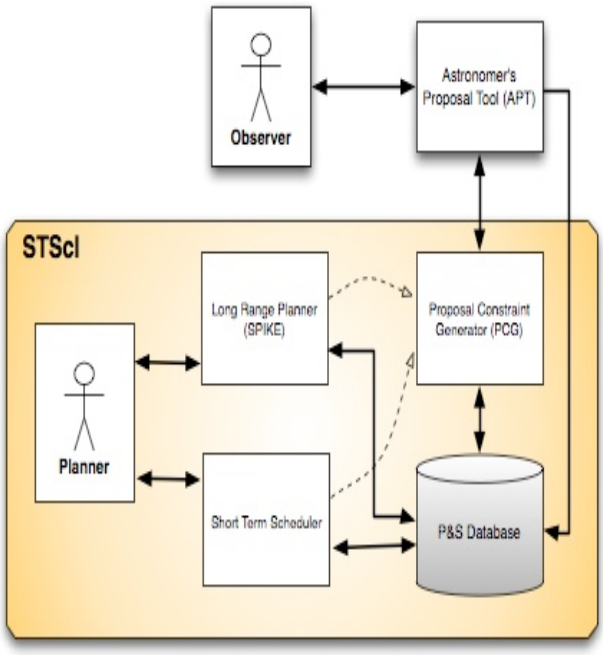

Figure 1 : JWST Planning and Scheduling Architecture

Another benefit of using PCG is the ability to give observers more detailed information on physical constraints involving guide stars. A guide star is a star near the target coordinate that can be tracked using the Fine Guidance Sensor (FGS) to keep the telescope stably pointed at the target region. Guide stars may only be

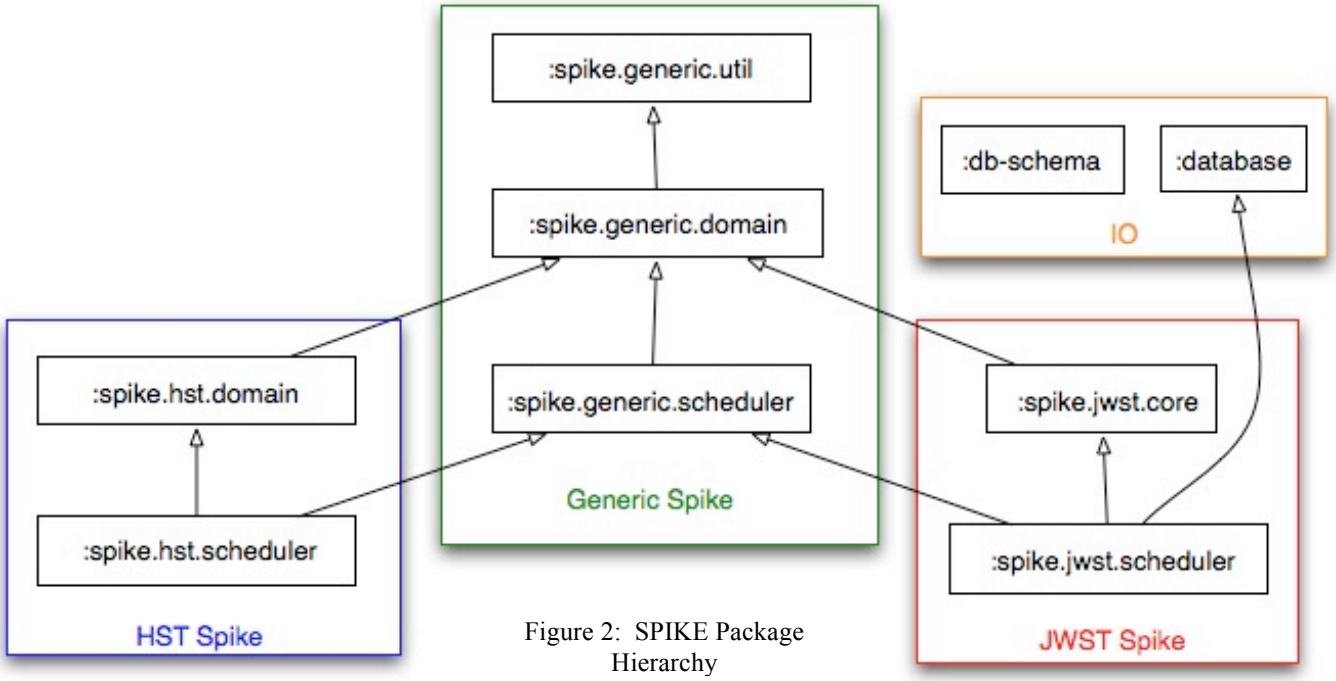

Figure 2: SPIKE Package Hierarchy

available at certain telescope orientations resulting in a set of time windows that constrain the observation. With HST, the availability of guide stars for a particular observation is not checked until the observation program has been submitted to STScI and the operation staff runs the guide-star system. With JWST, guide-star availability is checked by PCG. An observer will be able to access guide star availabilities from PCG through APT. If no guide star is available, the observer can modify his or her observation to fix the problem prior to submission.

Shared database interface

For the HST planning and scheduling systems, the interface between different subsystems is implemented inconsistently - through files, databases and other media. Each system has a partial view of the state of an observation. In order to collect the data on one observation, staff may need to look in various directories as well as query databases. Some data are redundant, as different formats are used in the interface between each subsystem, and so the same data may appear in different files or database tables.

For JWST, the decision was made that the interface between planning and scheduling subsystems would be through one shared database. Each planning and scheduling subsystem reads a consistent view of the observation specification and writes their output products to the database. This strategy removes redundancy in file formats, and removes the potential for inconsistency. While we believe the potential benefits of a shared database are significant, sharing database tables introduces increased complexity in coordinating database schema changes. As such, tools and processes have been developed to implement schema changes more smoothly. All the information on a database table, such as field names, type, description and the usage, are kept in a XML file that is version controlled. A tool then uses these XML files as input to automatically install database schema changes and web documentation.

JWST SPIKE Development

JWST Spike long range planning development began in earnest in the fall of 2009 and is planned to complete in 2013 to support pre-launch verification and validation testing. The system is being developed in a series of formal builds to the JWST project:

- 10/2010 – Model JWST domain
- 10/2011 – Define generic SPIKE planner
- 10/2012 – Implement JWST planning capabilities
- 5/2013 - Release to verification and validation

We have completed the JWST domain model and are now working on refactoring the SPIKE planner.

An initial issue tackled was the relationship between JWST and HST development. How should the two missions share code? Ideally, our approach should maintain the stability of the HST SPIKE Planning System while

allowing JWST SPIKE enhancements to be used in HST when possible. A related issue was whether to separate the HST and JWST SPIKE applications, or to have them use one shared model and run concurrently as a single application. We chose to separate the applications as this would decouple the build and test procedures, and there was no compelling use case to support both missions in a single application. To support sharing of code between missions, we will utilize an object oriented approach and will use a multi-step code refactoring process as described below.

A first step was to implement a package hierarchy that enables controlled sharing of functionality between the core generic components, and the HST and JWST SPIKE applications. As shown in Figure 2, the package hierarchy includes a core suite of generic code (green box) that is further broken down into domain, scheduler and utilities sub-packages. The domain sub-package contains models of objects in the space-mission planning domain: observations, targets and constraints as well as supporting infrastructure like link-constraint propagation. The scheduler sub-packages contains planning and scheduling routines, resource management and interface support. The package structure for applications is parallel to that of the generic core. The initial applications envisioned are HST (blue) and JWST (red). Additionally, database support routines are separated into a package outside this hierarchy (orange box).

Once the initial package hierarchy was defined, the HST code was factored into its internal sub-packages of domain, scheduler and utilities (i.e. the green box in Figure 2). The majority of the utility code was sufficiently generic and was moved into the generic utilities sub-package.

After completing the initial component breakdown for the HST SPIKE application, the domain components were refactored, moving core classes (or creating superclasses) and supporting functionality into the generic domain sub-package. The domain features described above were refactored, allowing the modeling of any space telescope mission using this generic domain. During this refactoring, the HST-specific functionality was carefully maintained in the HST domain object models.

Following the delivery of HST SPIKE with this initial refactoring, the development of the JWST SPIKE domain model began, reversing the process and creating the appropriate subclasses and structures required to model the specific needs of the JWST application domain.

Further developments in the first year of JWST SPIKE added new features to generic SPIKE that were not part of HST. This included a new database package that checks the consistency between the database schema and the SPIKE code accessing the database. Several additional developments are described later in the LRP enhancements section.

Currently, refactoring efforts are focused on moving the core plan window assignment routines to the generic SPIKE scheduler package. These routines find the best possible plan window assignments for sets of observations linked by timing constraints. The existing routines march over time examining potential plan windows and selecting the best potential window. The process uses three software mechanisms: critics, criteria, and filters [Giuliano 2008].

- Critics heuristically ensure that the high-value portion of the search space is explored. The search space for plan windows is huge and theoretically consists of every subset of a observation's constraint windows. Critics limit the potential solutions to be considered to those that are most likely to give a high value solution.
- Criteria are used to compare potential plan window assignments for an observation. The search process will select the potential plan window with the highest criteria value.
- Filters provide for efficient execution by partitioning the search space into a list of time intervals such that if a solution is found in one time interval the next interval is pruned from the search and is not examined.

Each of these mechanisms was initially coded as part of the HST scheduler and is currently being refactored so that they can be used in both missions. As stated above, a goal in this refactoring work is to maintain current SPIKE behavior when possible. However, we do not want to hamper the addition of new and improved capabilities. Refactoring these routines is a particular challenge with respect to maintaining current behavior, as these routines are the core of the planner. To mitigate this issue, the task of refactoring the planner core has been broken down into a series of incremental steps that start with infrastructure that is unlikely to change behavior and end with the potentially more volatile portions of the planner. The idea is to be able to regression test as much of the porting as possible against the current HST behavior.

SPIKE LRP Improvements

While refactoring SPIKE for JWST, we are making multiple improvements to SPIKE including:

- Making SPIKE more transparent as to what actions it has taken.
- Adding a multi-objective component to SPIKE.
- Making improvements to SPIKE's least commitment resource model.

Some of these improvements will be coded generically and will benefit HST, while others will remain JWST only. The main features of the improvements are discussed below.

Making SPIKE more Transparent

The SPIKE system includes multiple steps to plan observations including making an initial assignment, resource leveling, and adjusting linked observations for execution times. Each of these steps has numerous parameters that can be set per observation or per session including criteria weights, nominal plan window size, and planning start or end dates. While SPIKE currently provides tracking at a coarse level, more detailed information is needed by end-users to understand why a given action was or was not performed. In order to solve this problem, several new tracking capabilities are being added to SPIKE.

First, we are adding a command logger that keeps a history of all high-level user commands. In addition, the logger keeps track of all plan window assignments and un-assignments including the source of the assignment (e.g. manual, initial assignment, resource repair, adjusting links). This history log spans a SPIKE user session (which can be saved and restored), and allows users and other SPIKE software components access to the key steps taken during a scheduling session.

Second, we have formalized the definition of SPIKE planning parameters. A *planning scenario* is a named set of parameters stored in the planning database that includes all user options for controlling SPIKE. When executing the SPIKE planner, users specify a named planning scenario to be used. Many times, users want to use a named scenario except for a given parameter (e.g. changing the nominal plan window size from 60 to 50 days). To simplify this process, the system allows users to modify a parameter within a SPIKE session and for SPIKE to automatically store a new planning scenario with the modified planning parameter(s).

Finally, we are updating our long-range plan status tracking relations. The SPIKE long range planning process occurs throughout an HST or JWST observing cycle. At the start of a cycle, the long-range planner creates a plan integrating all the new observations for the cycle with any unexecuted observations from previous cycles. During the execution of the cycle, the long-range planner is run daily and will modify the plan based on new observations added to the plan, changes to existing observations, and execution times of planned observations. The long range planning process can be put into an equation:

Observations to plan + Input Long range plan = Output Long range plan

In other words, SPIKE takes as input a set of observations to plan and an input long-range plan and uses planning algorithms and interactions with the end user to produce a new long-range plan as output. Both observation data and long range plans are stored in database relations for JWST. A primary goal of the planning process is to maintain stability. In general, if an observation is planned in the input plan, it should have the same plan window in the output plan. Although stability in the plan is a goal, changes from plan-to-plan will occur and SPIKE needs to provide for tracking. These database records will allow users to trace the history of an observation's plan window and to know the planning scenario used to create each window.

Multi-Objective Scheduling

SPIKE currently provides a mechanism to evaluate the quality of a plan window assignment for an individual observation. The JWST long range planning requirements dictate that SPIKE provide the ability to evaluate a long range plan as a whole with respect to a set of criteria including minimizing momentum buildup, minimizing schedule gaps, grouping observations from the same program together, and balancing resources. To meet these requirements, SPIKE will support a multi-objective approach to scheduling. In a traditional single objective approach, individual criteria are combined in a weighted fashion. In contrast, in a multi-objective search, individual criteria are kept separate during scheduling. The result of a multi-objective search is a Pareto-surface of schedule solutions where no solution on the surface is strictly dominated by another solution in terms of all criteria. A multi-objective infrastructure has been added to generic SPIKE that supports global plan criteria and maintaining a Pareto-surface of potential solutions. Tools and techniques for displaying and reasoning about Pareto-surfaces are described in [Giuliano and Johnston 2010, Giuliano and Johnston, 2011].

Least Commitment Resource Models

A primary goal of long range planning is to distribute resource usage evenly across a planning interval so that efficient short-term schedules can be created. SPIKE currently uses two techniques to measure resource conflicts during long range planning called *classical plan window resources*, and *validation scheduling* of observations to orbits. While investigating prototype resource metrics, it was observed that the existing models have issues and that other models are possible. The existing models are described below and issues with the models are discussed. An alternate model that falls in-between the two models is

then presented. For ease of presentation the discussion below is given in terms of orbits as applied to HST. The same also applies to scheduling to precise times as will be done for JWST.

Classical plan window resources are tracked at a granularity of a day. The resources used by an observation are divided equally across all the days in the plan window. For example, Figure 3 gives an example of resource calculation for three observations O_1 (2 orbits long), O_2 (4 orbits long) and O_3 (3 orbits long). The plan windows of the observations stretch from day 2 to 3, day 2 to 5 and day 3 to 5, respectively. Here, we assume there are two orbits per day. The orbits required for each task are divided equally across the number of days in their plan windows. The bottom two rows of the chart show the amount of resources consumed and available for each day. The example shows that day 3 is oversubscribed, days 1, 6 and 7 are undersubscribed, whereas days 2, 4, and 5 are subscribed to full capacity. Although the classical plan window resource model captures the resource load over time and tracks resource failures, it allows both false positives and false negatives as illustrated below.

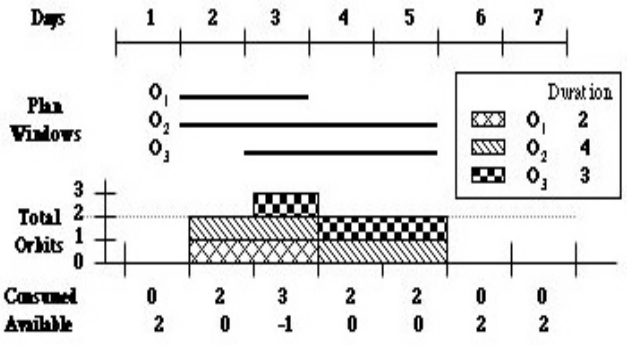

Figure 3: Example of classical resource model

Figure 4 shows an example of classical plan window resource consumption with a false positive. Here, a false positive means that the model shows no resource oversubscription yet the scenario is not schedulable. The figure shows the resource consumption of four observations across their respective plan windows. The total orbit limit for each day is 4 and the total consumption shows no oversubscribed days. However, an actual schedule of observations cannot be generated. The problem lies with observation O_2. There are not enough orbits to schedule it either before or after task O_1. The intuition behind this false positive is that the resource model implicitly allows observations to be executed in a non-contiguous manner during scheduling. However, this is not the case for HST or JWST scheduling.

Figure 5 gives an example of classical resource consumption with a false negative. Here, a false negative means the model shows resource over-subscription yet the scenario is schedulable with no over-subscription. Each day has 2 total orbits. Day 3 and day 4 are oversubscribed in the classic resource model. However, a schedule can be obtained by placing O_1 in days 2 and 3, O_2 in day 1, and O_3 in day 4. The false negative occurs because the resource model assumes that the resource consumption is evenly divided across the plan window.

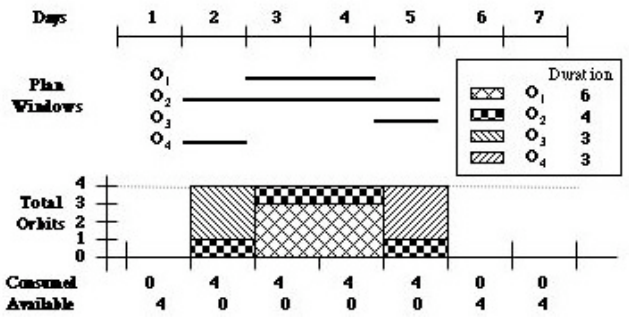

Figure 4: Classical resource model with false positive

The classical plan window resource model shows a probabilistic picture of demand for resources, but may result in false positives and false negatives. With the classical plan window resource model, it is hard to pinpoint which observations are likely to fail due to over-commitment, or which time of the year HST orbits are underused. To account for the false positives and negatives, the SPIKE tool creates a separate resource validation schedule that assigns observations to specific orbits [Ferdous, 2006]. This process identifies times where the plan windows cannot be realized as a short-term schedule. The main problem with the validation approach is that it removes the least-commitment nature of the plan, as it only shows that a single scheduled time is good and does not validate the entire plan window.

As noted above, the classical SPIKE plan window resource model makes two assumptions:

- Observations can be broken into as many pieces as desired.
- Resources are divided evenly across the days in a plan window

In contrast, the validation schedule makes neither of these assumptions. The first assumption causes false positives in the classical resource model as it ignores the bin packing aspect of making a schedule. The second assumption causes false negatives in the classical resource model due to observations with short plan windows. The HST and

JWST scheduling domains were examined with respect to the causes of false positives and false negatives. HST is in low Earth orbit and is impacted by the South Atlantic Anomaly (SAA) in 9 of its 15 orbits for a day. The SAA impacts create a complex bin-packing problem as an

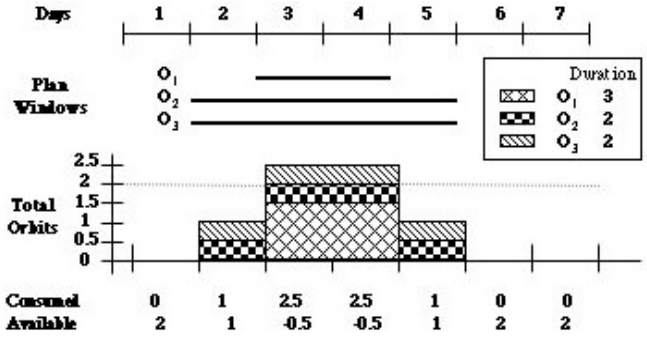

Figure 5: Classical resource model with false negative

observation can be scheduable in the 6 SAA free orbits and in some, or none, of the SAA impacted orbits. In contrast, JWST, at L2, will not have earth occultation and SAA impacts. As a result of the orbit differences, JWST is expected to have less bin packing problems than HST resulting in less false positives in the classical resource model. JWST, like HST, will have observations with short plan windows resulting in false negatives in the classical model. Since JWST will have less opportunities for false positives, but will still have false negatives, we investigated and coded a third resource model that supports removing the false negatives found in the classical model.

When refactoring the SPIKE resource model, we added a resource *assignment map* that specifies how resources are consumed over a plan window. By default, resources are divided evenly, but the assignment map provides the ability to explore alternative resource allocations over time. This implementation leads to a resource assignment problem. Given a set of plan windows for observations, can resource assignment maps be defined for all observations so that no resources are oversubscribed? For example, Figure 6 shows a resource-conflict free distribution of resources for the example given in Figure 5. Even for this simple example, there are many ways in which the resources could be assigned to remove conflicts. The figure shows a distribution where observations O_2 and O_3 both reduce their usage to 0.25 orbits in days 3 and 4 and increase their usage to 0.75 in days 1 and 4. The idea here is that we can redistribute resources yet keep the window nature of the model. We are currently exploring heuristic methods for incrementally solving the resource assignment problem.

If a resource assignment problem is not solvable then we know that the corresponding resource validation orbit assignment problem is unsolvable. If we cannot solve the problem allowing observations to be broken into arbitrary pieces, then the problem cannot be solved if each observation has to be executed in a non-interruptible fashion. On the other hand, solving a resource assignment problem does not ensure that we can assign observations to orbits with no resource conflicts. A solvable resource assignment problem may not have an orbit assignment due to bin packing concerns.

The resource assignment problem is a mid-point between the classical plan window resource model and validation scheduling to orbits. An advantage of the resource assignment model is that it retains the window nature of the classical plan window resource model. There may be many resource usage assignments that do not violate any resource limits. The resource assignment search mechanism will have a metric that measures how even the resource usage is over the plan window. The search will use the metric to prefer solutions that keep resource usage evenly spread out over the observation plan windows as much as possible as opposed to solutions that fragment resource usage for an observation. In this way, the resource assignment model retains a least-commitment component.

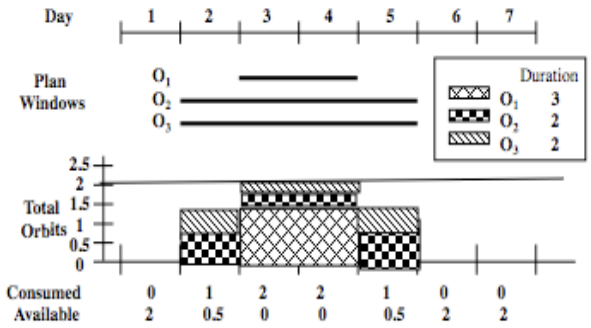

Figure 6 : False negative example repaired by adjusting resource usage

The three resource models are compared and contrasted in Table 1. The ability to assign alternative resource maps has been coded for the resource assignment model. However, only simple engines to generate alternate maps have been prototyped. The results have been promising and we are currently investigating alternative search techniques for making resource assignments. Based on our findings, we will decide which resource models will be used in JWST operations.

Conclusions

Development of JWST long range planning capabilities for SPIKE is ongoing and expected to complete in May of 2013. This paper gave a status report describing the context for JWST development, a software engineering process that allows code to be shared with HST SPIKE, and improvements to core planning capabilities for least commitment planning.

References

Hawkins, R., Jordan, I. and Giuliano, M.E., 2009. Applying Lessons Learned in Planning and Scheduling the Hubble Space Telescope to the James Webb Space Telescope. *International Workshop on Planning And Scheduling for Space (IWPSS)*. Pasadena, California.

Ferdous, N., and Giuliano, M.E., 2006. Validating Resource Usage in Least Commitment Planning. *International Workshop on Planning And Scheduling for Space (IWPSS)*. Baltimore, Maryland.

Giuliano, M.E., and Johnston, M.D. 2010. Visualization Tools For Multi-Objective Algorithms. Demonstration: *International Conference on Automated Planning and Scheduling (ICAPS)*, Toronto, Canada.

Giuliano, M.E., and Johnston, M.D. 2011. Developer Tools for Evaluating Multi-Objective Algorithms. *Interatnional Workshop on Planning and Scheduling for Space (IWPSS)*. Darmstadt, Germany

Giuliano, M.E. 1998. Achieving Stable Observing Schedules in an Unstable World. In *Astronomical Data Analysis Software and Systems VII*. 271-274.

Giuliano, M.E., 2008. Handling Oversubscribed Orbital Resources in Hubble Space Telescope Operations. *Workshop on Oversubscribed Planning and Scheduling, at the International Conference on Automated Planning and Scheduling (ICAPS)*, Sydney Australia.

Model Name	Description	Pros	Cons
Classical Plan window	Resources are spread evenly among the days in a plan window	Low runtime overhead. Distributes resources over the whole plan window	False positives and false negatives in the model
Resource assignment	Resource usage over plan window can be assigned	Removes false negatives. Can detect some unschedulable situations. Retains window nature.	Requires search to determine resource usage. Model has false positives
Validation scheduling	Assign observations to individual orbits	Removes both false positives and negatives.	Requires search to determine resource usage. Only verifies the single assignment not the entire window.

Table 1: Comparison of least commitment resource models