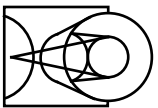


SPACE TELESCOPE SCIENCE INSTITUTE

**NASA P&S Workshop
Miller, et. al.
28 October 1997**

Adaptation and Evolution of the Spike Planning and Scheduling System

**Glenn Miller, Mark Giuliano, Larry Kramer,
Tony Krueger, Peg Stanley
Space Telescope Science Institute**



SPACE TELESCOPE SCIENCE INSTITUTE

NASA P&S Workshop
Miller, et. al.
28 October 1997

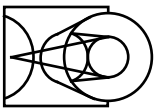
Overview

Spike is a general framework for planning and scheduling originally developed for the Hubble Space Telescope by STScI.

This talk reviews major architectural components of Spike, adaptation of Spike to new observatories and highlights lessons learned.

In addition we make some general observations:

- No one model of planning and scheduling applies to all observatories.**
- An architecture needs to be flexible to encompass different processes.**
- Challenges for the future**

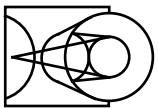


SPACE TELESCOPE SCIENCE INSTITUTE

NASA P&S Workshop
Miller, et. al.
28 October 1997

Conceptual Foundations of Spike

- Suitability Functions:
 - express strict constraints as well as preferences (“soft” constraints)
 - mathematically sound representation
 - allows simple and efficient propagation of constraints and preferences
 - Constraint Satisfaction Problem (CSP)
 - “multistart stochastic repair”
 - Window Planner
 - stable plans in an unstable world
 - Quantitative Schedule Quality Measures
 - successful schedules require a balance of many factors
- Created with reuse and extensibility as design criteria, not afterthoughts



SPACE TELESCOPE SCIENCE INSTITUTE

NASA P&S Workshop
Miller, et. al.
28 October 1997

Core Architecture of Spike

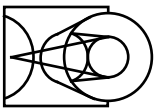
"Micro-Spike": Constraint Representation,
calculation and propagation

Utilities

Database Interface

Astronomical Objects: Sun, Moon, Spacecraft, ...

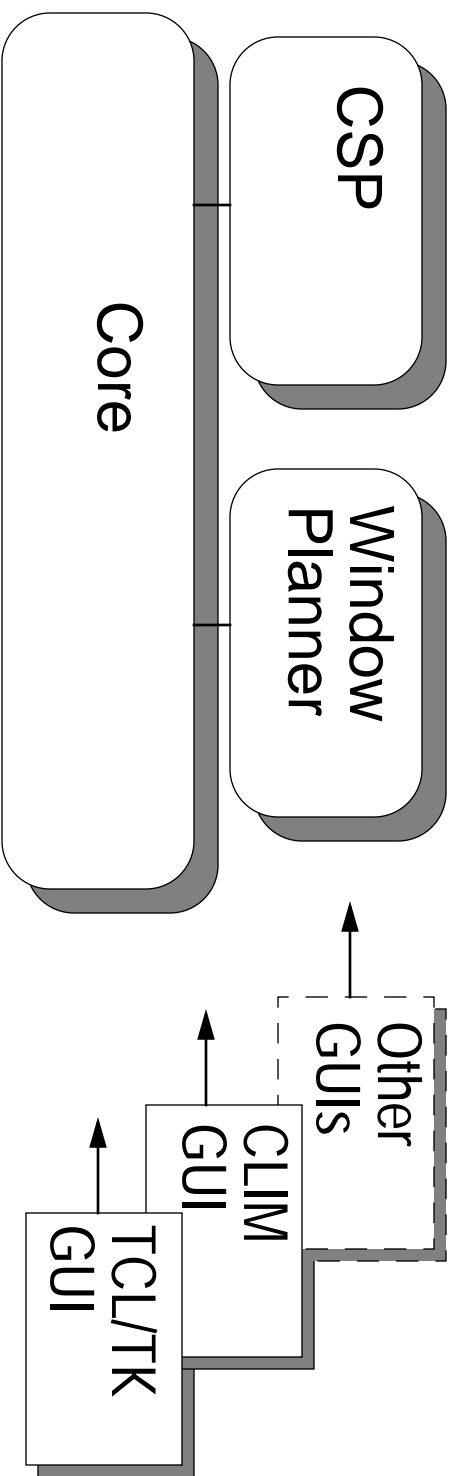
General, mathematical, astronomical date, time

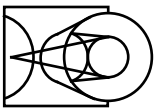


SPACE TELESCOPE SCIENCE INSTITUTE

NASA P&S Workshop
Miller, et. al.
28 October 1997

Spike Architecture





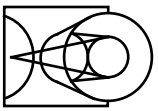
SPACE TELESCOPE SCIENCE INSTITUTE

NASA P&S Workshop
Miller, et. al.
28 October 1997

Implementation

- **Object Oriented: Common Lisp Object System (CLOS)**
- **Concise**
 - Basic system is ~22K lines of code in a dozen files
 - Complete system (mission specific interface) is ~40K lines of code in ~30 files.
- **Fast**
 - Efficient representation of constraints and propagation of effects
 - Caching for compute intensive calculations
 - Efficient search strategies
 - Schedules hundreds of observations in tens of seconds on Sparc 2 class machine

Originally implemented on TI Explorer Lisp Machines (late 80s), and ported to Suns in early 90s. Included moving from Flavors to CLOS.



CSP Scheduler

Constraint Satisfaction Problem (CSP)

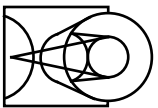
- set of variables, each with a domain of discrete values and a set of constraints which limit a variable's value based on the assigned values of all variables.

Problem:

- Assign a value to all variables such that all constraints are satisfied.

Spike Scheduling CSP:

- variables represent tasks to schedule
- values represent time intervals
- solution is assigning a time interval (value) to each task (variable)
- further require that solution maximize preferences

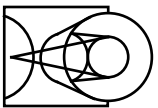


SPACE TELESCOPE SCIENCE INSTITUTE

NASA P&S Workshop
Miller, et. al.
28 October 1997

Spike CSP Toolkit

- Each variable instance records preference and constraint conflicts for each domain value
- Variables are permitted to have assignments which have constraint conflicts. This is important in guiding the search process.
- Variables can be locked to a value
- Variables can be ignored
- Constraints can be weighted
- Constraints can be cached - yields performance improvement and also allows for an explanation of constraint violations
- Capacity (Resource) constraints



SPACE TELESCOPE SCIENCE INSTITUTE

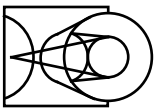
NASA P&S Workshop
Miller, et. al.
28 October 1997

Origin

Original scheduler was implemented as a neural network - the “Guarded Discrete Stochastic” (GDS) neural network.

GDS neural network performed very well, and detailed analysis of why it worked led to the development of the “min-conflicts” strategy for solving CSP problems.

Neural network is no longer used in Spike as the CSP formulation has a much smaller overhead (particularly smaller space overhead).



SPACE TELESCOPE SCIENCE INSTITUTE

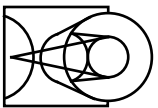
NASA P&S Workshop
Miller, et. al.
28 October 1997

Multistart Stochastic Repair

Uses constraint violations to guide search. Steps:

1. *Trial Assignment* - Make an initial guess. Resulting schedule will generally have constraint violations and resource capacity overloads.
2. *Repair* - Apply heuristic repair techniques to eliminate violations until there are no conflicts or a pre-established level of effort has been reached.
3. *Deconflict* - Eliminate any remaining conflicts by removing tasks or relaxing constraints.
4. *Fill In* - Add “filler” tasks where possible, without introducing constraint violations.

Entire process can be repeated (with different heuristics or randomization) and best solution chosen. Parameters in steps 1&2 are easily customized (min conflicts, high priority, early or late greedy, etc.)



SPACE TELESCOPE SCIENCE INSTITUTE

NASA P&S Workshop
Miller, et. al.
28 October 1997

Long-Range Planning

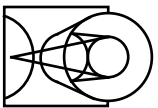
STScI uses multi-year Long-Range Plan (LRP) to manage HST observing program.

- inform observers when observations will occur
- plan for servicing missions (new instruments and repairs)
- respond to changes in instruments or spacecraft

Originally used CSP scheduler to assign observations to a single week for execution, as short term schedules are made weekly.

To achieve high efficiency, short term scheduler required ~100% oversubscription. Thus LRP was highly unstable and required additional effort to reschedule ~half of every week's observations.

As a result, STScI developed *Plan Window* concept

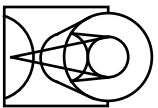


SPACE TELESCOPE SCIENCE INSTITUTE

NASA P&S Workshop
Miller, et. al.
28 October 1997

Window Planner

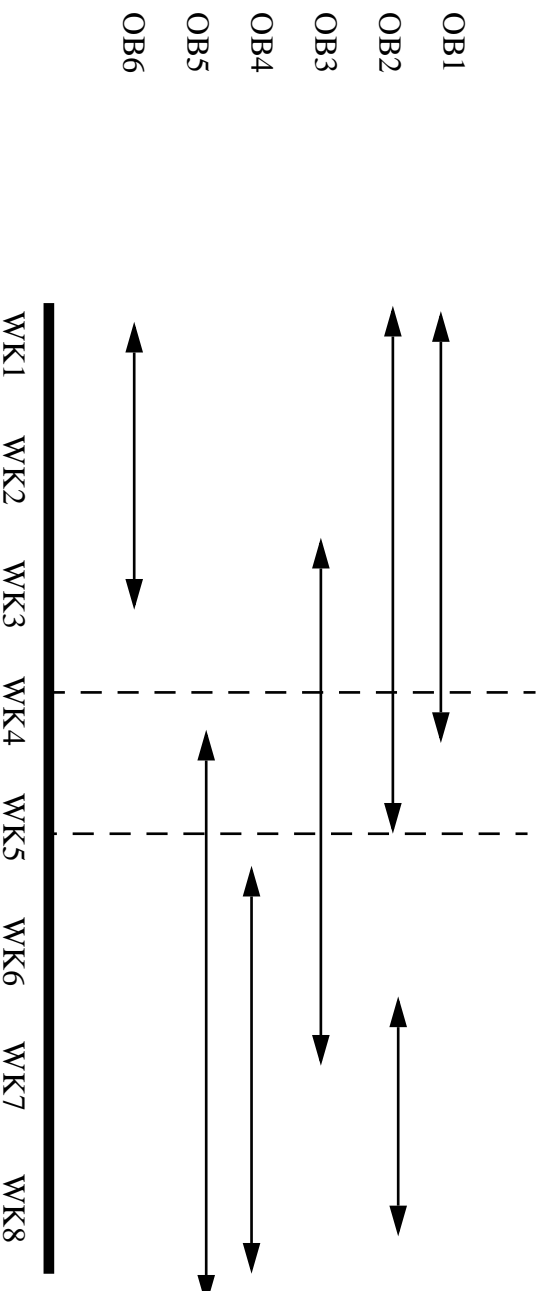
- **Plan window is a subset of an observation's constraint window**
constraint window reflects the physical and scientific constraints
- **Plan windows are selected to optimize:**
 - **Observatory resources: observing time, SAA avoidance, ...**
 - **Scientific quality: orbital visibility, low background, ...**
- **Represents a best effort commitment to schedule in the window**
- **Nominal window duration is ~8 weeks**
 - **Provides sufficient resolution for user planning**
 - **Builds in flexibility and oversubscription needed for stable and efficient plan**



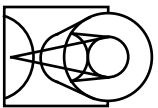
Example Plan Windows

Plan windows for observations 1-6 in weeks 1-8.

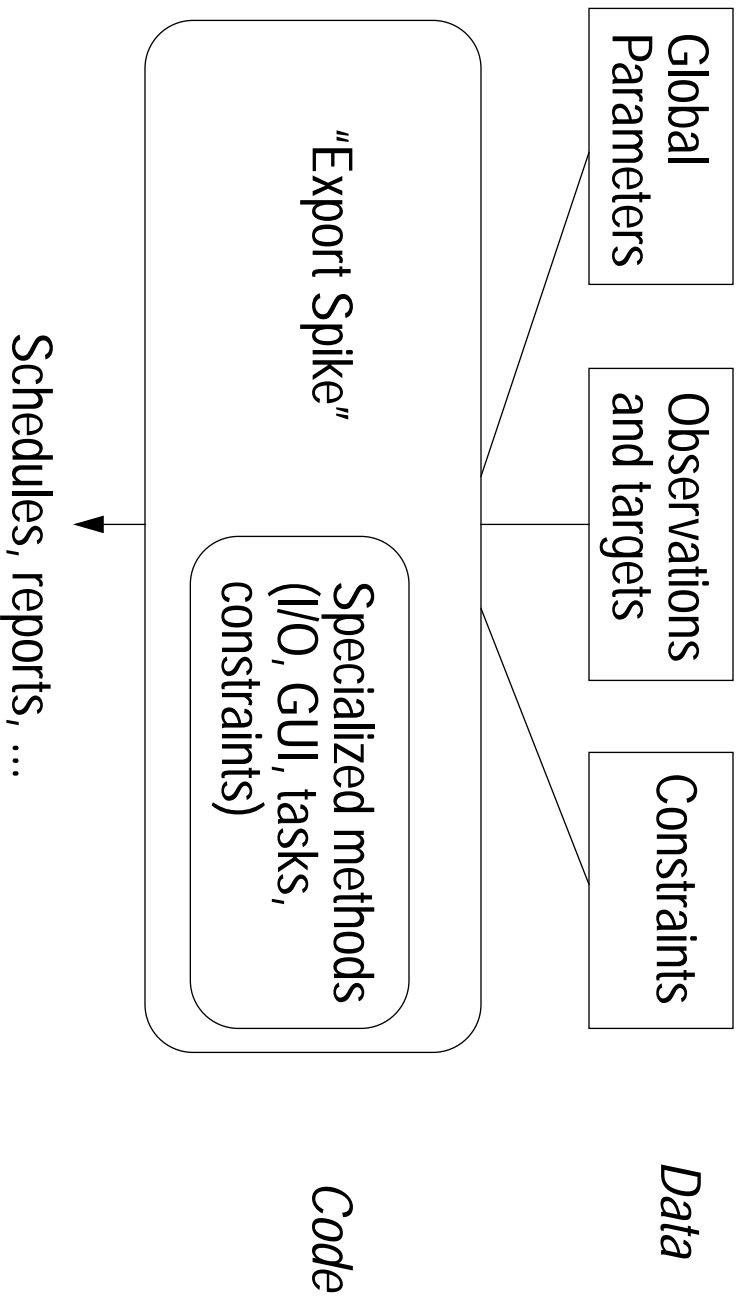
- Plan windows are overlapping and possibly non-contiguous

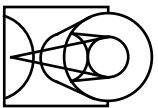


Candidates for Week 4 are 1, 2, 3 and 5. Obs 1, 2, 3 may have been executed previously.



Adapting Spike to a New Mission





Example Specialization

`task-to-task-overhead:`

- method to compute overhead time between two tasks

Parameters:

- Two task instances. Task class is specialized for an observatory, e.g. FUSE-TASK, ESO-TASK

Default method

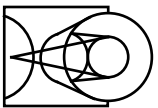
- No overhead

FUSE

- `slew-time (task1, task2)`

VLT

- `max (instrument-overhead-time (task1, task2), slew-time (task1, task2))`

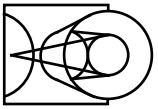


SPACE TELESCOPE SCIENCE INSTITUTE

NASA P&S Workshop
Miller, et. al.
28 October 1997

Spike Applications

HST	Operational since 1989, used for proposal preparation (RPS2), implementation, and long-range scheduling
FUSE	Will be used for long- and short-term scheduling.
ESO/VLT	Long-range (semester) and short-term (monthly) scheduler: Currently being tested on NTT.
AXAF	Will be used for long-term scheduling
EUVE	Operational since 1991, used for long-range scheduling of pointed observations
ASCA	Operational since Nov 1992, long-range scheduling
XTE	Operational since Dec 1995, long-range scheduling
GSOC/DLR	German Space Agency has used Spike to schedule several MIR experiments
Interval Logic Corporation is applying Spike technology to semiconductor manufacturing	



SPACE TELESCOPE SCIENCE INSTITUTE

NASA P&S Workshop
Miller, et. al.
28 October 1997

Additional Capabilities

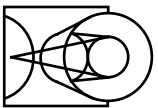
Examples of features added to Spike:

Far Ultraviolet Spectroscopic Explorer (FUSE)

- variable task duration (duration depends on time of execution)
- interruptible observations
- spacecraft slews, ground station contacts, ram avoidance

European Southern Observatory Very Large Telescope (ESOVLT)

- moon phase, airmass
- coordinated scheduling on multiple telescopes
- minimization of instrument changes
- calibration observations



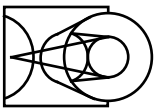
Lessons Learned

It is possible to build an architecture for planning and scheduling for a broad domain which is both general enough to be easily extended but also efficient enough to handle the task.

This is achieved by:

- general purpose constraint representation and propagation
- libraries of domain-specific objects and methods
- modular design and object-oriented implementation
- specialized scheduling/planning engines as needed

Software must support the actual scheduling process used by observers and observatory. No one process fits all observatories.



SPACE TELESCOPE SCIENCE INSTITUTE

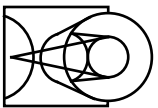
NASA P&S Workshop
Miller, et. al.
28 October 1997

Future Possibilities for Spike

**Use CORBA to better integrate with other components of ground system
(in progress)**

Port from Lisp to C, C++, Java, ... ?

Use commercial software for event calculation (e.g. Satellite Toolkit from Analytical Graphics)



Challenges

Challenge for designers of planning and scheduling systems is to stop reinventing the wheel. Possible steps:

- ❑ More modular designs to be able to plug-in components from a variety of systems
- ❑ More use of commercial software for basic functions such as orbital events
- ❑ Interoperability via CORBA, scripting languages
- ❑ Develop multi-mission format for interchange of scheduling data
- ❑ Publish benchmark problems and solutions for realistic mission planning and scheduling