Astronomical Data Analysis Software and Systems VII
ASP Conference Series, Vol. 145, 1998
R. Albrecht, R. N. Hook and H. A. Bushouse, eds.

Astro-E's Mission Independent Scheduling Suite

A. Antunes and A. Saunders
Hughes-STX, Goddard Space Flight Center, Greenbelt, MD 20707, Email: antunes@lheamail.gsfc.nasa.gov

P. Hilton
Hughes International/ISAS, 3-1-1 Yoshinodai, Sagamihara, Kanagawa 229, Japan

Abstract. The next generation of Mission Scheduling software will be cheaper, easier to customize for a mission, and faster than current planning systems. TAKO ("Timeline Assembler, Keyword Oriented", or in Japanese, "octopus") is our in-progress suite of software that takes database input and produces mission timelines. Our approach uses openly available hardware, software, and compilers, and applies current scheduling and N-body methods to reduce the scope of the problem. A flexible set of keywords lets the user define mission-wide and individual target constraints, and alter them on-the-fly. Our goal is that TAKO will be easily adapted for many missions, and will be usable with a minimum of training. The especially pertinent deadline of Astro-E's launch motivates us to convert theory into software within 2 years. The design choices, methods for reducing the data and providing flexibility, and steps to get TAKO up and running for any mission are discussed.

1. Scheduling Defined

The basic concept of scheduling is simple. You take a list of targets, and make a calendar. The rest is efficiency and bookkeeping, which is akin to saying "to make a painting, just put paint on a canvas." In reality, it is a bit more complicated. The three main factors to consider are calculated quantities (i.e., unchangeable facts of the situation), derived quantities (soft constraints, ratings of good and bad events, and politics), and capacity constraints (such as time and telemetry).

For mission scheduling, the general calculated quantities are based on the platform. For satellites, this includes: sun angle, orbital position, earth limb angle, roll angle, and others. For ground based, several analogs are day/night, latitude/longitude, and elevation angle. These have no particular weight to them, but are simply physical facts about the observing situation.

From these, we create the derived quantities, which determine whether a given observation is feasible. This includes (for satellites) issues like allowable sun condition, occultation, thermal profile, in-SAA region, bright/dark earth, ground-station contacts, and star tracker acquisition. Some of these are func-

tions of the calculated quantities, and others are qualitative opinions based on the calculated quantities. Some derived quantities can be entirely political, and imposed from the outside (scientific priority, for example).

Capacity constraints include, first and foremost, time, generally defined as "you can only look at one thing at a time." Telemetry is a large concern for satellites, and other user-defined resources vary from mission to mission.

Tracking all of these is the principal task of scheduling, that is, to place targets such that no constraints are violated and all resources are used to maximum efficiency without overbooking. The goal is ultimately to maximize the scientific routine.

So tools to manipulate this output not just "a schedule", but also an evaluation of its overall suitability (and stability) for the problem at hand. The interface must present all the quantities above to the user when hand-editing is required. Finally, the automatic scheduling routines (ranging from simple "good time bad time" limiting, to AI routines that make sequences) must interface with the editing options available to the user.

TAKO ("Timeline Assembler, Keyword Oriented") is our multimission software suite for achieving this goal while simplifying training and operations (and thus saving time and expenses). Our goal is that TAKO will be easily adapted for many missions, and will be usable with a minimum of training. Rather than having a single monolithic program, we are building an integrated suite (similar to the FTOOLs/XANADU approach) that handles all input, parameterization, scheduling, and output. This provides flexibility both for a variety of missions, and for changes during an mission lifetime.

Further, our design is modular to allow customization of relevant sections for different missions. Feedback from the Astro-E science team and from other schedulers has been crucial in determining the necessary capabilities for this suite. The first benchmark goal for the Astro-E first edition is to be able to schedule at least 400 items over a 1-year baseline at 1 minute resolution, and to manipulate this schedule in real time without alienating the user.

2. TAKO Design

The focus for users is the intuitive GUI for interaction and scheduling. Scheduling is an intensely visual task, and much of the development time is for this GUI. The actual scheduling engine deals only with the generic time-ordered quality function and header, without having restrictions on what the header elements are. The intermediary code between this flexible GUI and generic scheduler is what makes TAKO specifically useful for satellite and observational scheduling.

For a given mission, the input file currently specifies how to custom-build single observations in a highly flexible way. It uses ASCII files for storage and output, allowing filters to be written and input/output to be integrated with other packages (including spreadsheets). Thus the entire suite can be dropped as a single unit into the mission's science operations center, with pre- and post-processors written to make the linkage.

TAKO uses standard hardware, operating systems, and compilers (i.e., any workstation with gcc and Tcl/Tk 8.0). And, TAKO is modular by design, to allow for use with different missions. Switching missions or handling updated

requirements requires adjustments to a handful of the discrete modules, rather than an entire code overhaul. And much of the mission-specific specifications can be done at the user level, in the input files, rather than requiring a recompile.

To improve communication, all our documentation is produced as the modules are written, in HTML, so that the entire package design can be understood and easily accessed by Web. The design work and user manuals alike will be available from the Web. After integration, training manuals will also be available. In the absence of dense technical details within this paper, we therefore refer you to these Web pages (URL given below).

3. TAKO Implementation

For TAKO, we set several basic design precepts. The GUI is an on-screen editor that ties it all together. Keyword-value pairs are used to define data elements and relations generically. Each data element has a header, and acts like an object. This object-oriented design means that new elements can be defined on the fly, much as with a database. And, parameters can be adjusted during runtime or before each run, with saved definitions allowing mission customization.

Mission-specific code is largely in orbital prediction routines ("calculated quantities"). Derived quantities are specified in terms of these, using the keyword-value pair schema. There are four primary types of structures ("target information", "constraints", "resources", and "event list"). "Target information" is the information from the science proposals. "Constraints" is the interpretation of the science objectives into operations terms, i.e., "avoid SAA" in people-speak becomes an on-screen curve showing what times are allowed. Constraint curves are very similar to SPIKE "suitabilities" (for those familiar with HST's package). "Resources" includes items like the actual schedule (how much time is spent on target) and telemetry. And "event lists" are outputs like the mission timeline itself, lists of calibration observations, and so on.

4. TAKO Buy-In

To work for many missions, a scheduling tool must be flexible, accept post-launch changes, be able to handle many situations, and be better/faster/cheaper[1]. In the past, it took as long to customize an old tool as to write a new one, and most solutions required post-launch customization. Also, each mission required new training for new tools, often at a programmer level, increasing costs and time spent doing everything except actually scheduling. So TAKO is designed to be easily adapted for many missions, and to require a minimum of training at the user level (substituting "common sense" instead)[2].

Buy-in assumes that the software is subject-oriented rather than focusing on the specific algorithms or performance benchmarks. In short, the user comes

---

[1] NASA motto

[2] *It takes an engineer to design a car and a mechanic to fix it, but anyone can drive.* R. A. Heinlein

first. A good design is presumed to already use appropriately chosen algorithms and to achieve requisite benchmarks; what matters most is that the software not merely function, but be genuinely usable. Therefore, the problem it solves is "maximize science return"– to consistently produce good mission schedules over a long period of operations.

Looking at it from a system perspective, again focusing on the user base, we find that it must install easily and require no special software, must provide immediate functionality, and be easily integrated with familiar tools and files. From a long-term perspective, it should be flexible, adaptable, and easily modified. And, it should be evolutionary (taking the best of previous packages) rather than implementing new ideas simply to be different– there should be a familiarity to the overall methodology.

To achieve these goals, we've defined two approaches, "internal" and "external". Internally, the base-level algorithm and programming work is being done with standard professional methodologies, to select the best approach for the given precepts and implement it (nothing terribly novel there). Externally, the GUI is the predominant way the user will interact with TAKO, and is being designed by a programmer/analyst (p/a), receiving constant feedback from three different mission schedulers. Thus the interface is essentially being designed by the users, with the p/a acting as adjudicator and implementer. TAKO was designed in detail before coding began, so that ad hoc coding and integration problems are generally minimized.

Astro-E launches early in 2000. For more information, visit the Astro-E URL (http://lheawww.gsfc.nasa.gov/docs/xray/astroe) and the TAKO subpages (http://lheawww.gsfc.nasa.gov/docs/xray/astroe/tako).

Acknowledgments. We are grateful to Pamela Jenkins for scheduling advice, Larry Brown for programming support, and Glenn Miller for moral support.