Calvin Klatt et al.: A Next Generation Geodetic Experiment Scheduling Tool? (SKED++), IVS 2002 General Meeting Proceedings, p.77–80
http://ivscc.gsfc.nasa.gov/publications/gm2002/klatt2

A Next Generation Geodetic Experiment Scheduling Tool? (SKED++)

Calvin Klatt, Mario Bérubé, Anthony Searle, Jason Silliker

Geodetic Survey Division, Natural Resources Canada
Contact author: Calvin Klatt, e-mail: cklatt@nrcan.gc.ca

Abstract

A number of the software tools used in geodetic VLBI were designed twenty years ago and written in FORTRAN. Despite their good performance, the software requires maintenance and new features are desirable. This presentation is a proposal to the geodetic VLBI community to think about re-writing one of the essential utilities, namely SKED. A new version of SKED could take advantage of new programming techniques, such as object-oriented (OO) programming to ease the maintenance of the software. Such a new OO software design would enable the optimization routines to be improved by a variety of users, with the best features of SKED remaining. This presentation is intended to get feedback on this proposal and generate further discussion about the status of crucial VLBI operational software.

1. Essential Software

SKED plays an important role in all VLBI experiments and in the vast majority of experiments the schedules are very well optimized.

1.1. Why Change?

1) SKED is aging. The core was written many years ago and has been repeatedly modified as VLBI technology has evolved. As new hardware appears, more *if* statements are added - in many places - making changes and debugging difficult. The distribution of SKED/DRUDG now contains hundreds of FORTRAN files plus C and Java. The user-interface is a mix of text menus plus a Java GUI.

2) A dynamic future involves changes. The formation of the IVS was intended to further involve the entire VLBI community in how VLBI is performed. Open-source flexible software tools could be part of this effort. Small antennas (CTVA, Mizusgsi) are currently difficult to schedule with other antennas that are significantly more sensitive. Some new systems' features are not fully utilized (e.g. continuous tape motion). In the near future much greater data recording rates, shorter scans and real-time VLBI will come into practice. Finally, we should use real-time/adaptive scheduling to make the most of the huge investments at the antennas.

1.2. Why Bother?

There are a number of arguments against fixing a tool that isn't broken. SKED could be patched indefinitely, and, given this, the community's resources may be better spent elsewhere. Some of the new features, such as adaptive scheduling may not be realistic in the next few years. Another concern is who will maintain a new version of SKED (over the long term)?

2. A Dynamic Future

2.1. Programming Languages

Continued reliance on aging FORTRAN tools cannot continue forever, and the longer we avoid this change the higher the price we pay. We will have difficulties attracting new staff when programmers are not being trained in the FORTRAN language. Furthermore, we have an obligation to students to give them up-to-date and marketable skills: Both students and new employees need training that is useful to their careers.

New software development could take advantage of modern software development environments and could take advantage of Object Oriented Programming. The VLBI scheduling task lends itself to OO programming extremely well: New equipment would only require the creation of a new derived class and the addition of necessary virtual functions. The existing OO code would remain untouched.

If the VLBI community wants to have a dynamic future it must respond to changes and be up to date.

2.2. Design Goals

The following are the design goals we feel are most important in re-writing SKED:

- Learn from the past: copy or improve on the existing features. Separate the source-selection algorithm from the detailed construction of the objects involved.
- Open source optimization: enable anyone to write optimization software without extensive interaction with the basic equipment functionality (antenna slewing, tape status, frequency, modes, etc.).
- Take advantage of available software development tools on the most common hardware platform (PC).
- Ease the maintenance of the software by utilizing existing software tools such as Excel.
- Use an up-to-date, friendly user interface.
- Maximize flexibility of the antennas and data systems (e.g. allow for multi-beam, real-time systems).
- Enable real-time or adaptive scheduling. The schedule should respond if one antenna fails mid-way through an experiment.

3. A Preliminary Version

In the initial stages of the software development we have two main deliverables, the first being software suitable for one VLBI system (S2) but not including the optimization routine. The architecture is designed for all VLBI systems, obviously. The second deliverable is a stand alone optimization routine that will try to duplicate as closely as possible the existing routine in SKED.

We feel that future developments in schedule optimization should be separated from the basic station operations, such as antenna slewing, horizon masks, tape control, etc. The project is designed such that the scientific aspects that are embodied in the optimization routine can be addressed by a maximum number of people (including students).

3.1. Assumptions Made

- Bill Gates' dream of a PC on every desktop has nearly come to pass.
- These PCs run Windows with MS Excel installed.

3.2. Decisions Taken

- The software is written in the C++ language within Borland Development Environment, Builder C++ 5.0. Further developments would not be dependent on this choice.
- Data (catalogs of station characteristics, source flux, etc.) are stored in an MS Excel file. The software accesses data from the Excel file through an Excel Com server.

3.3. Objects

Figure 1 illustrates the objects in our design and their interrelationship.

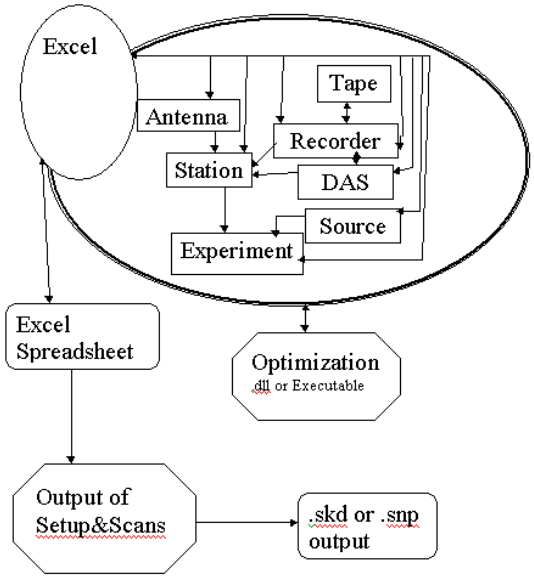

Figure 1. **Objects**

- **Experiment** A list of stations and sources at a defined epoch. Uses a baseline approach with two antennas and a source.
- **Station** Pointers to Antenna, DAS, Recorder.
- **Antenna** Different classes for axis, slewing rates, sensitivities.
- **Data Acquisition System (DAS)**
- **Recorder** Pointer to the Tape object.
- **Tape**
- **Source**

3.4. Program Flow:

At this time our design involves a program flow as follows:

- The user chooses stations, sources, DAS setup, recorder and optimization scheme. An alternative is to load a saved list of selections. These are used to create an experiment class.
- The user defines the start epoch and length, the optimization or autosked++ routine, and starts the creation of a schedule.
- The optimization routine queries the experiment class of the main program for available baselines, plus other parameters.
- The optimization routine generates instructions for each station that are time stamped.

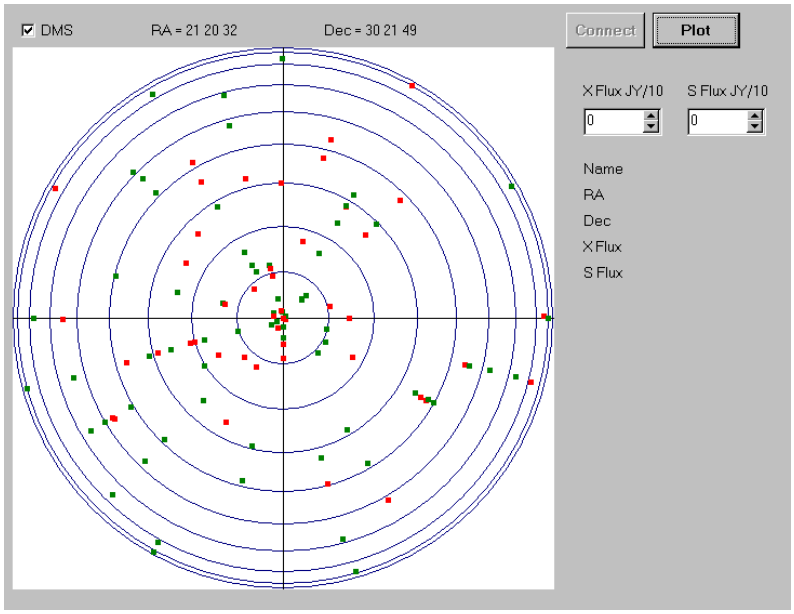

Figure 2. Source Selection Screen Capture