Astronomical Data Analysis Software and Systems IX
ASP Conference Series, Vol. 216, 2000
N. Manset, C. Veillet, and D. Crabtree, eds.

The Transition from VMS to Unix Operations for STScI's Science Planning and Scheduling Team

D. S. Adler, D. K. Taylor

Science Programs, Computer Sciences Corporation, Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD, 21218 USA

Abstract. The Science Planning and Scheduling Team of the Space Telescope Science Institute currently uses the VMS operating system. SPST began a transition to Unix-based operations in the summer of 1999. The main tasks for SPST to address in the Unix transition are: (1) converting the current SPST operational tools from DCL to Python; (2) converting our database report scripts from SQL; (3) adopting a Unix-based code management system; and (4) training the SPST staff. The goal is to fully transition the team to Unix operations by the end of 2001.

1. Introduction

The Science Planning and Scheduling Team (SPST) of the Space Telescope Science Institute (STScI) is responsible for the planning and scheduling of all HST observations. The group creates and maintains the multi-year long range plan, generates the weekly science calendars (Science Mission Specification - SMS), and generates the command loads which are uplinked to HST.

Currently, SPST works with the VMS operating system. Due to an Institute wide shift away from VMS platforms, SPST began a transition to Unix-based operations in the summer of 1999. The operational software (Fortran/C/C++), maintained by the Planning and Scheduling software team (PST), is nearly 100% Unix-compatible. For the transition, SPST must: (1) convert the current operational tools from Digital Command Language (DCL) to Python; (2) convert our existing database report scripts from Sybase SQL; (3) adopt a Unix-based code management system; and (4) train the SPST staff in Unix, Python, and the database report language so that all team members can support and develop tools in the future. SPST must continue the mission-critical function of planning and scheduling HST while transitioning to Unix-based operations.

SPST currently consists of twelve members with a wide range of Unix comfort and compatibility. Most team members have done some sort of tool scripting, mostly in DCL. An important consideration in the choice of scripting language and database report language is that while a couple of team members have a background in computer science, we are *primarily a group of astronomers.*

With few outside resources available, the transition to Unix-based operations must fit into our regular schedule. On a rotating basis, one team member is assigned the task of working on the transition full-time. Other team members who are not building calendars can be used for work on the project assuming no other special projects take priority. While we have access to other groups at STScI for advice, the project is SPST's to design, develop, and implement.

2. Design

2.1. The Choice of a Primary Scripting Language

A Scripting Language Team (SLT) was formed to investigate three general-purpose, cross-platform scripting languages - Perl, Python, and Tcl. The SLT translated an existing DCL script from the SPST operational tool catalog into the three candidate languages and compiled a list of pros and cons about each. Their comments were then compiled, the scripts reviewed, and a recommendation was made to SPST management by the Transition Manager.

The SLT found that as a string-based language, Tcl is less powerful and less flexible than the other two choices. Perl was found to be quite powerful, with a high degree of built-in functionality. However, the SLT felt that efficient Perl code is difficult to read and maintain. Python's ease of use and readability was admired by most team members. While Python code might be less efficient than similar Perl code, it was found to be more than adequate for the needs of SPST. Python's combination of power and readability makes it the top choice for our primary scripting language.

2.2. The Choice of a Database Report Language

Under VMS operations, SPST uses Sybase ISQL to pass database information between tools and Smartstar SQL to print formatted reports. The ISQL reports work fine under Unix, but the Smartstar SQL has to be converted. We currently have a Database Report Language Team (DRLT) investigating several options. The first option is to convert all existing SQL scripts to SQR, which requires SPST team members to learn a new database report language. The second option is to convert all SQL to ISQL, use these reports to produce an intermediate (unformatted) file, and then use a Python script to parse this output and produce a formatted report. Preliminary results with this method are encouraging, but the extra step slows down processing. Other options include database readers in Perl and Tcl. PST is also investigating the creation of a direct link from Python to the database.

2.3. The Choice of a Code Management System

For software maintenance, SPST currently uses the VMS code management system (CMS). Since other groups at STScI are also looking to move away from VMS operations, a CMS Replacement Group (CRG) was formed to look into Unix-based systems. The CRG is currently investigating RCS, CVS, Razor, and Perforce. For ease of maintenance, SPST will go along with the recommendations of the CRG.

3. Development

3.1. Tool and Report Triage

SPST has over 400 operational DCL tools, some dating back to 1985. There are also almost 300 ISQL/SQL reports. A triage was performed to determine which tools are still in use and which could serve a purpose in the future. During the triage, we grouped the tools into ten distinct categories. We evaluated the need for each tool using the following criteria:

- Is the tool currently used in our daily calendar-building operations?
- If not, is it still used on a regular basis?
- If not, is it still used by other groups at the Institute?
- If not, could it have a use to any Institute group in the future?

If the answer to all of these is "no," the tool will not be translated. Most of the obsolescent tools are due to process improvements and retired HST instruments.

Table 1. SPST Tools and Reports

GROUP	Description	TOOLS total (to convert)	REPORTS total (to convert)	EXECUTABLES total (to convert)
SYS	System Functions	52(42)	3(2)	4(4)
UTIL	Multi-group Utilities	71(25)	73(39)	4(3)
PPREP	Proposal Preparation	93(40)	33(25)	5(2)
CPREP	Calendar Preparation	19(13)	61(15)	9(3)
CBLD	Calendar Building	34(24)	53(14)	1(1)
CCHK	Calendar Checklist	34(17)	20(11)	0(0)
EPHEM	Ephemeris Processing	15(13)	16(13)	1(1)
SMS	SMS Generation	26(19)	26(13)	4(3)
SCHK	SMS Checklist	21(12)	6(4)	2(2)
PASS	PASS Processing	57(36)	6(5)	14(8)
TOTAL		422(241)	297(141)	44(27)

The first three groups to be translated will be the calendar-processing groups, since these are used early in the calendar-building process. To help create efficient code, a team member went through this set of tools and identified code best suited for a utility library. Once this basic set of utilities was created, the tool conversion started.

3.2. Coding Style/Standards and Code Reviews

We started with the C/C++ coding manual from PST and adjusted it for our needs. In particular we address topics such as complete, consistent header information (for all tools, modules, classes, and functions); comment style (brief but descriptive); variable/constant naming conventions; and isolation of platform-specific code into utilities.

4. Implementation

4.1. Unix and Python Training

Training is weighted towards the beginning of the project. All SPST members are attending an STScI-wide Unix course. We are using "Learning Python" (Lutz & Ascher 1999) and "Programming Python" (Lutz 1996) as well as tutorials found at the Python home page[1].

One team member is always in a 12-week "software development" position. Since the first three team members are already familiar with Unix and Python, they needed very little training. During this 36-week period, other team members are learning Unix and Python during off-shift weeks. In this way, the majority of the software development shift will be spent on code translation rather than training.

4.2. The Operational Transition from VMS to Unix

The goal is to have the Python tools work under both VMS and Unix operations. In this way, current VMS operations will serve as a testbed before the move to Unix. The DCL scripts will be available as an emergency fallback. When a complete suite of Python tools is confirmed to work under VMS, it will be completely tested on Unix. Once it has been confirmed to work there, the operational work for that group of tools will be switched to Unix. PST will then drop their support for that suite of VMS tools.

5. Autumn 1999 Progress Report

Two SPST team members have completed their software development shifts. After administrative startup, the actual code translation began in early September.

A utility library with over 70 functions has been created. There are two main modules - "spss," which includes classes on calendar-building basics, and "time_tools," which includes classes on various time functions in the SPST software. As of this writing, several operational tools have been translated, including "calopt.py," which addresses scheduling possibilities for science programs being considered for a calendar, and "ask_for_pars.py," which creates and manipulates lists of crafted parallels for primary science.

The short-term goal is to translate the calendar-building tools and install them under VMS by January 2000. While being used under VMS operations, they will be tested on Unix with the goal of switching them to Unix operations by the spring 2000.

References

Lutz, M. 1996, Programming Python (Sebastapol: O'Reilly)

Lutz, M. & Ascher, D. 1999, Learning Python (Sebastapol: O'Reilly)

[1] `http://www.python.org`