Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

Design of a Remote Proposal Submission System

A. Richmond, M. Duesterhaus, S. Yom

Hughes STX, GSFC/NASA, Greenbelt, MD 20771

E. Schlegel, A. Smale

NASA, Research Scientist, USRA, Greenbelt, MD 20771

N. E. White

NASA/Goddard Space Flight Center, Laboratory for High Energy Astrophysics, Greenbelt MD 20771

Abstract. Many astrophysics and astronomy tools have been developed utilizing NCSA Mosaic and the World Wide Web (WWW). Several tools have been developed at the High Energy Astrophysics Science Archive Research Center (HEASARC) at NASA's Goddard Space Flight Center, including a Remote Proposal Submission[1] (RPS) system for NASA satellite missions. The RPS provides facilities for submitting Observation Request forms. It is an automated service, designed to support several missions. Currently only the *XTE* mission is supported. Other projects that will be supported in the future include *ASCA*, *GRO*, and *ROSAT*.

1. The High Energy Astrophysics Science Archive Research Center

HEASARC was created by NASA in 1990 as a site for X–ray and Gamma-ray archive research. The total data volume will be of the order of 1 Terabyte by 1995, available on-line for immediate access. HEASARC provides a multi-mission archive for data from the *Röntgen Satellite*, the *Compton Gamma-Ray* Observatory, the *Astro 1 Broad Band X–ray Telescope*, *Advanced Satellite for Cosmology & Astrophysics*, and *XTE* missions. This co-exists with the archival data from past missions such as *HEAO 3*, *OSO 8*, *SAS 2* and *3*, *Uhuru*, *Vela5B*, the second *High Energy Astrophysics Observatory*, and the first *High Energy Astrophysics* Observatory. Data from non-US missions (e.g., the *European X-ray Astronomy Satellite* and *Ginga*), is also provided as international agreements allow, along with data from *Ariel-V*, *COS-B*, *EUVE*, *HST*, *IRAS*, *SMM*, and *TD1*. The HEASARC is located at the Goddard Space Flight Center and is a collaboration between Goddard's Laboratory for High Energy Astrophysics (LHEA, responsible for the science content of the archive), and the National

[1] http://heasarc.gsfc.nasa.gov/cgi-bin/RPS.pl

Space Science Data Center (NSSDC, responsible for the data archive management).

2. What is XTE?

XTE is the *X–ray Timing Explorer*, categorized by (1) 1 microsec time resolution, (2) 2–200 keV energy band covered, (3) 1.06 microJy sensitivity at 5.2 keV, (4) 1° FWHM field-of-view, (5) all-sky monitor capability, (6) having more than 93% of sky visible, and (7) near real-time operation.

The WWW was chosen as the user interface to the *XTE* RPS to promote

Familiar interfaces. The learning curve is very small. NCSA Mosaic is free and many users already have it installed. The WWW interface has point-and-click functionality making it very user-friendly.

Support All Platforms. The WWW was chosen to allow us to support all platforms with minimal effort. This minimizes the amount of software that needs to be maintained. And since the server is centralized, it does not have to be ported to every type of machine and every operating system version used in the astronomy community.

3. The Major Implementation Problems: User/Browser Side

Loss of Control. When we were writing GUI's we had complete control over our interface and could simply create user-side code as we saw fit, e.g., to pop up some dialog. We have given up that flexibility, for the time being, because of the significant advantages we gained. But we hope that ultimately there will be some client-side equivalent to server scripts, so that we can, for example, interface data analysis tools to browsers.

Layout. One of the major implementation problems was getting satisfactory layout. We experimented with this and finally settled on parsing a LaTeX template to get position information and making the screen resemble the actual paper form by using preformatted layout. And if you do not—well, the widget and its label may get separated by a line break.

Lack of Browser Uniformity. Another problem is variations in presentation via different browsers and platforms. We really would like to see more uniformity between browsers. Often one may spend considerable time getting the layout to look good on one (with all defaults selected), only to find that it is a horrible mess on another.

4. The Major Implementation Problems: Server Side

The major limiting characteristic of HTTP is statelessness. For our StarTrax interface we want to engage in a dialog with the user which progresses from nothing, and builds up to a state where we have identified a target satisfying given constraints, and can offer associated data products. The WWW offers the opportunity to "mix and match" reusable components in a rapid prototyping approach; we were astonished in our early days (only 1 year ago!) at how quickly we could deliver useful functionality. Later we began to realize that some of our early optimism was perhaps too naïve: the need for good software engineering practices is probably even greater than before. There are many more components being interconnected and interfaced, and exponentially more failure points. Perl helps you to hack up something real fast without worrying too much about the engineering aspects until much too late.

5. General Observations

In spite of the sometimes severe disadvantages, we believe the WWW is a very effective platform for rapid information systems development in astronomy and astrophysics. The WWW provides a substantial pre-existing infrastructure supporting a client-server architecture on the Internet, so one can deliver application functionality at a much faster rate—say, an order of magnitude faster. In the "classical" mode—in which we started this project—the developer(s) spend a great deal of time wallowing in relatively low level code. We have not yet achieved the goal of "reusability," in spite of the promises of modern software engineering methodologies. Using WWW and HTML (Hypertext Markup Language) (and ideally, Perl), since you build on that infrastructure, you can deliver functionality very quickly. This not only delivers the promises of rapid prototyping (throw one away), but also of rapid development. The iteration cycle time was often of the order of minutes, rather than hours or days, as in the classical approach. This was partly due to the pre-existing WWW functionality, but to our choice of Perl rather than C. Perl is ideally suited to this kind of work, because it, too, provides a great deal of ready-made high-level functionality. For example, we converted some 30 lines of C code, for decoding URLs, into 3 lines of Perl.