Data Vault: providing simple web access to NRAO data archives

Ron DuPlain[*a], John Benson[b], Eric Sessoms[a]
[a]National Radio Astronomy Observatory, 520 Edgemont Rd, Charlottesville, VA, USA 22903-2475;
[b]National Radio Astronomy Observatory, P.O. Box O, Socorro, NM, USA 87801-0387

ABSTRACT

In late 2007, the National Radio Astronomy Observatory (NRAO) launched Data Vault, a feature-rich web application for simplified access to NRAO data archives. This application allows users to submit a Google-like free-text search, and browse, download, and view further information on matching telescope data. Data Vault uses the model-view-controller design pattern with web.py, a minimalist open-source web framework built with the Python Programming Language. Data Vault implements an Ajax client built on the Google Web Toolkit (GWT), which creates structured JavaScript applications. This application supports plug-ins for linking data to additional web tools and services, including Google Sky. NRAO sought the inspiration of Google's remarkably elegant user interface and notable performance to create a modern search tool for the NRAO science data archive, taking advantage of the rapid development frameworks of web.py and GWT to create a web application on a short timeline, while providing modular, easily maintainable code. Data Vault provides users with a NRAO-focused data archive while linking to and providing more information wherever possible. Free-text search capabilities are possible (and even simple) with an innovative query parser. NRAO develops all software under an open-source license; Data Vault is available to developers and users alike.

Keywords: data archive, web application, free-text search, model-view-controller, python framework, web.py, Google Web Toolkit (GWT), rapid application development, agile development

1. INTRODUCTION

All data from telescopes of the National Radio Astronomy Observatory (NRAO) becomes public after a pre-determined proprietary period (usually 12 months). Like most observatories, NRAO hosts public data in an archive for on-demand access. Since 2003, NRAO has provided a web-based search tool and interface for accessing public and proprietary data (when properly authenticated) via the internet.

Over the course of a few years, as the archive added new features, rising sophistication of web users demanded additions that were outside of the original requirements. For users, constructing a query with the advanced search form became ever more involved and somewhat daunting, especially for new users. For the NRAO staff, new search parameters and features increased the maintenance burden of the web application. Although a form-based interface is still required for advanced searches, the NRAO began to investigate new approaches to searching and browsing contents of the archive, with three primary goals: 1) create a simple "Google-like" free-text search box for searching contents of the archive, 2) provide access to all NRAO telescopes through such a search box, and 3) link various information services to the search box and its query results. This is the Data Vault project.

2. EVOLUTION OF THE NRAO DATA ARCHIVE

A centralized web interface to NRAO data has only recently been available. Prior to that, data was accessible in a rather ad-hoc fashion.

2.1 Ad-hoc access

Legend has it that in order to access NRAO data in the twentieth century, a researcher required connections. Accessing data files required knowing someone who knew the location of the archive, how to navigate it, and had the time available to fulfill a demand for data. Perhaps it was not such a cloak-and-dagger system, but certainly, someone

[*] rduplain@nrao.edu; phone +1 434-244-6845; www.nrao.edu

seeking data had to be familiar enough with those who maintained the archive. At the very least, one must know where to start asking questions.

2.2 Data Archive System

NRAO launched the Data Archive System in 2003, and provided a web portal telling users exactly where to start on the quest for data. The Data Archive System, which consists mostly of Java web technologies, allows users to submit queries via the web for data browsing and retrieval, stores a copy of observing data for disk storage, which is isolated from telescope data collection, and extracts and catalogs metadata to support user queries. As of the beginning of 2008, the Data Archive System allowed users to retrieve public and approved proprietary data for the following:

- all Very Large Array (VLA) data and images
- all Very Long Baseline Array (VLBA) data and images dating back to 1997
- limited Robert C. Byrd Green Bank Telescope (GBT) data

The data archive system archives new data from the VLA and VLBA telescopes daily. The system authenticates users based on the current user database and the NRAO Proposal Submission Tool. The archive system has a search tool that is available through a standard web browser at http://archive.nrao.edu and delivers data to users through FTP for public data and HTTPS for proprietary data.

The archive system search tool provides forms for basic queries using fields such as NRAO proposal or observing project ID, observer's name, NRAO telescope, observing date (range). Advanced queries can include info such as data type, source name, source coordinates, sensitivity and resolution, and frequency ranges.

As of Dec 2007, over 700 users from 250 institutions have downloaded nearly 9 TB of telescope data (60,000 data files). The download data rate has climbed to over 200 GB per month (1600 data files per month). Data files over one year old are in the public domain and account for over one-half of the download volume. The data files reside on a hard disk array, provide the archive users with fast access, and download via FTP and HTTP. The total size of the on-line archive is 33 TB. The Data Archive System is currently setup to use the Next Generation Archive System (NGAS).[1]

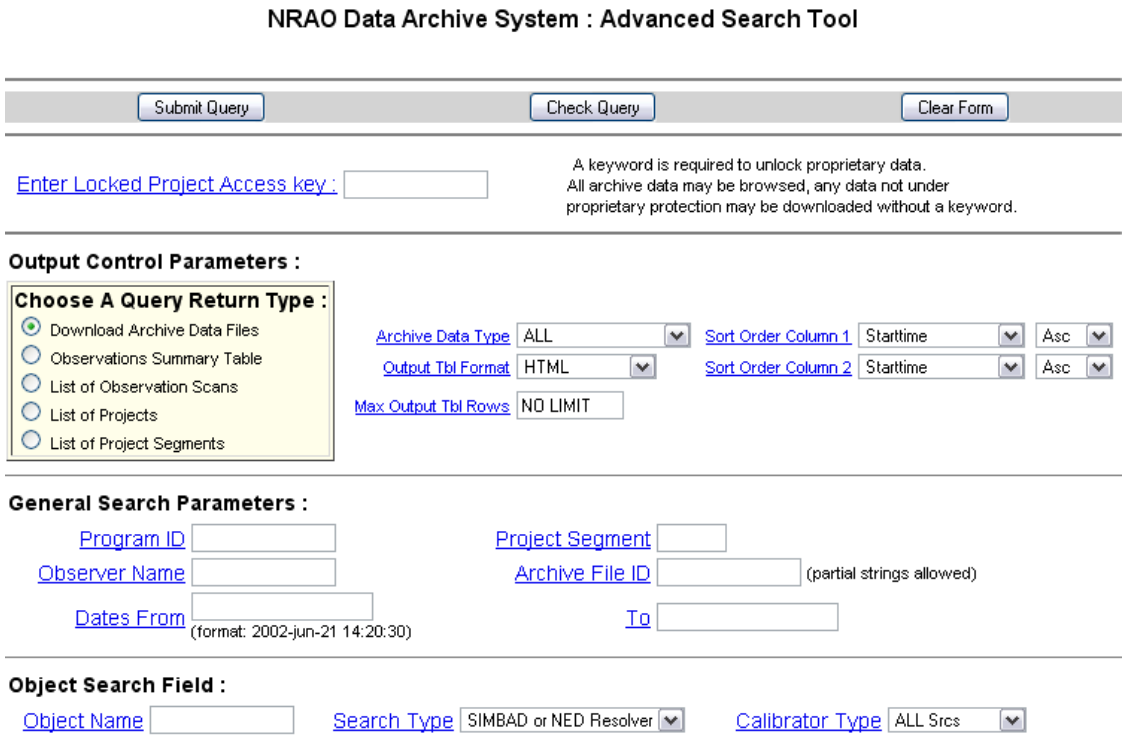


Figure 1. Above is a screenshot providing a glimpse of the Data Archive System's advanced search tool as of May 2008.

Among various archive requirements, the Expanded Very Large Array (EVLA) requires that the Data Archive System support all collected data (science, testing, and calibration), environmental data, monitor and control (M&C) data, project data – including proposals, observing scripts, logs, and papers – and data products such as calibrated data.[2] The Data Archive System plans to support these requirements.

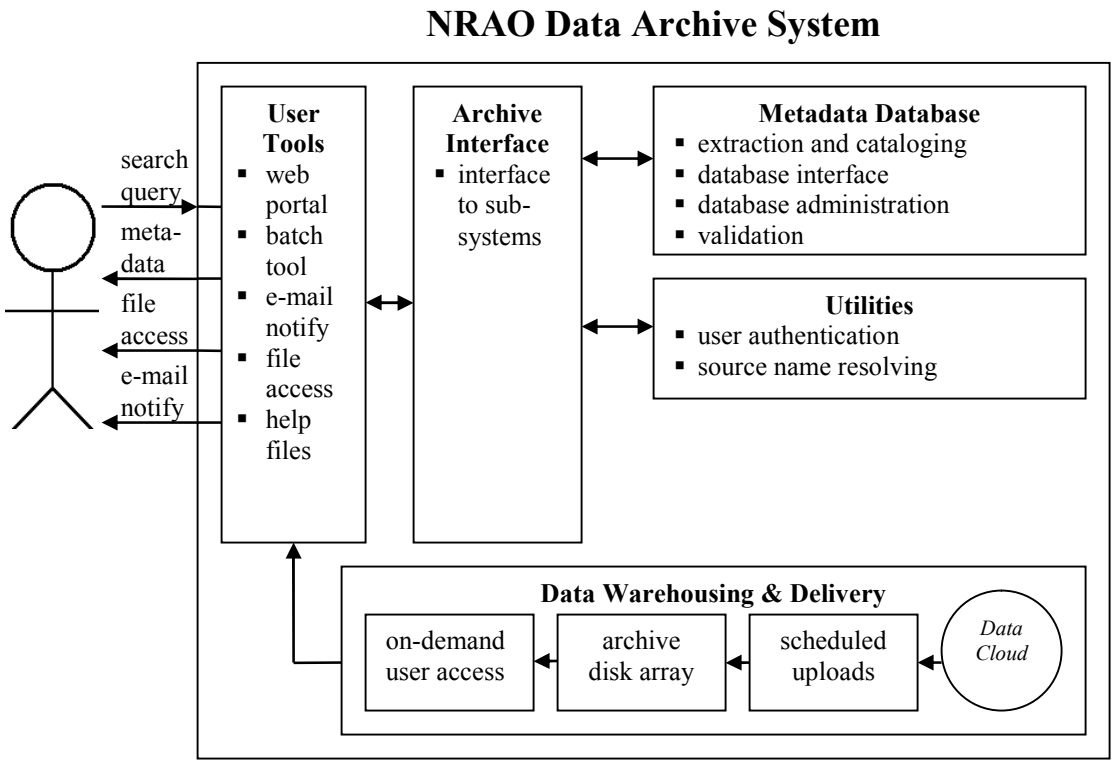


Figure 2. Above is basic block diagram showing Data Archive System information flow from the point of the view of the user, as of fall 2007.

2.3 Data Vault

Data Vault began by providing full support for archiving and accessing GBT data. The first public beta of Data Vault in fall 2007 provided two search boxes, one for VLA and VLBA data, and one for GBT data. The VLA & VLBA search box simply parsed the query request and sent users to the Data Archive System results as though they had used the advanced search form, which Figure 1 illustrates.

However, the GBT search box worked entirely independently of the Data Archive System, since GBT data was not available by those means. Instead, the Data Vault itself produced a table of results, linked to GBT proposal data, and experimented with various ways to browse the results and narrow down to target data. This version of the Data Vault was an excellent proof of concept, but since it did not provide a single search across all NRAO telescopes, a completely rewritten version replaced it in spring 2008.

As of May 2008, the Data Vault consists of a web.py-built controller, a Google Web Toolkit (GWT) browser-based client, and a MySQL database. DataVault aims to provide its users with a modern web experience. If successful, Data Vault will make the user feel as though the application is not a software application at all, but an easy-to-use tool for accessing various forms of content through a responsive UI that is a one-stop entry point to many interesting services.

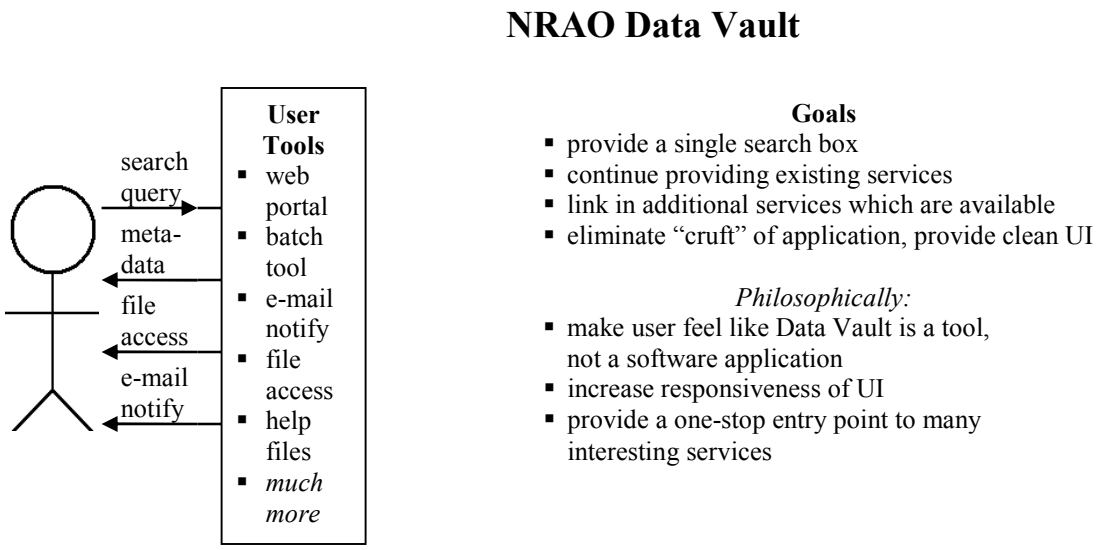


Figure 3. Data Vault aims to provide an entry point to the Data Archive System's services, while providing a single search box, which gives access to all available NRAO telescope data. If successful, users will not notice all of the background services required to keep the Data Archive System running.

Data Vault's UI goal is largely to simplify the user experience. With growing demands on the form-based search tools, various error messages sometimes show up in query results of the Data Archive System that may confuse users, since at times, the error message is the result of an underlying support tool which the user would not otherwise see. Simplified, Data Vault can focus on providing clear messages to the user. Figure 4 illustrates Data Vault's home page as of May 2008.

Figure 4. Above is a glimpse of the Data Vault, as of May 2008; it is a tabbed application, which begins by presenting users with a single search box.

2.4 Supporting the Virtual Observatory

Integration with the Virtual Observatory (VO) and the services it provides is critical for all modern observatories. "The VO aims to provide astronomers with seamless access to online resources... The VO is the next step, providing new resources and seamless access to them. New data and tools are already here and will be continue to be added."[3] NRAO is committed to supporting VO activities and aims to integrate VO search tools into the Data Vault application in the upcoming year.

3. DATA VAULT ARCHITECTURE

Data Vault implements a model-view-controller design pattern,[4] with MySQL databases backing the model, a web.py controller between the model and the view, and a Google Web Toolkit browser client to provide the view.

3.1 Metadata registry

The metadata registry contains key information about the data in the archive, and provides search terms for the application. Each search term has at least one corresponding field in the registry, and available search terms include telescope, project ID, frequency, band, receiver, detector, observer, source, coordinates, and date. The Data Vault metadata database is effectively read-only except when importing new data. The import can run independently of the application. Hence, to the Data Vault, the registry database is read-only.

Free-text searching requires that metadata has high integrity to function properly. Data Vault runs its import by checking that all input data matches fields in the database. If even one field is out of place, the importer will reject a scan in order to preserve proper function at runtime.

Data Vault uses MySQL to provide a familiar database management system for interested stakeholders at NRAO.

3.2 Stand-alone query execution server

Data Vault uses a web.py controller to connect the model (i.e. the database in MySQL) to the view (i.e. the code executed in the web browser). The web.py framework is a minimalist web application framework in Python.[5] It provides powerful URL mapping with regular expressions and simple templating, making web.py ideal for outputting XML or basic HTML to the web browser based on the results of small Python methods. The controller accepts as input a method name and various arguments (as a URL), and returns simple XML as output.

The controller calls on a query parser when it receives a request for the search method. The search method requires at least a query ("q") argument, which corresponds to a free-text query. The query parser identifies all possible permutations of the request, identifying whether the user asked for data relating to an observer, a source, both, etc. For example, a search of "w3" could mean the source by that name or it could be part of a project ID (e.g. "W308B"). In that case, the query parser determines that "w3" could be either a source or a project, with the source receiving higher rank.

On multi-term searches, the query parser will continue to provide permutations and will become decreasingly relevant to the search request. For example, searching on a source and an observer will return results with the source **and** the observer, just the source, and just the observer. The search method accepts a relevancy argument ("rel") to avoid long-winded search results.

The search method can also accept arguments for the number and offset of records for return, which allows for tailored paging in the results. The controller response is XML containing records, which correspond to the fields determined by the query parser, returned in order of rank. The search method also supports which fields a search query should use, which in effect will change the number of records returned, where there are no duplicate rows in the results. For example, if a search returns two rows that differ only by source, removing the source field from the search will reduce those two rows into one. A download request for that one resulting row is equivalent to two download requests – one for each of the previously listed rows.

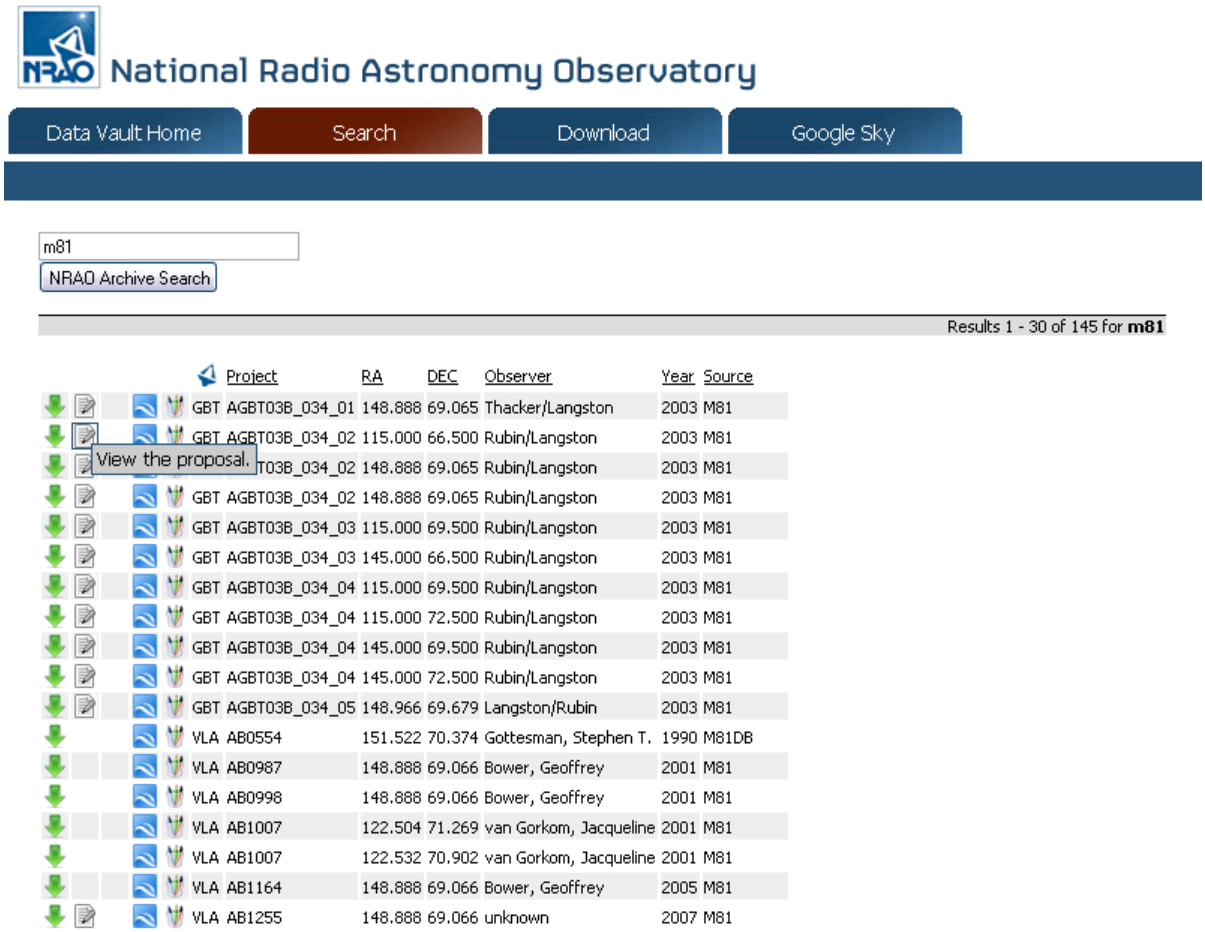
Beyond searching, the controller also supports retrieving the number of records and fulfilling download requests (through info and download methods). All methods return XML, which the requesting browser can easily parse, or another web service or stand-alone program can use.

Stackless Python provides for advanced process management, which Data Vault uses to cache queries; that is, Data Vault cache queries as Stackless threads.

3.3 Rich web client

Users who visit the Data Vault directly in the browser never even see the input (URL request) and output (XML response) of the controller. When a user loads the Data Vault web application, the web browser executes Google Web Toolkit entry point. GWT is a toolkit for developing rich internet applications, and involves writing Java code which compiles to JavaScript. This allows many developers to produce highly responsive web applications using traditional software engineering practices.[6]

GWT provides the view of Data Vault. Users interact with the GWT-enabled Data Vault code that executes in the web browser. This allows users to interact heavily with Data Vault without ever reloading the page. Whenever the user requests a search or a similar service, the web client will start an Ajax request to the server, which loads into the page in a matter of seconds without reloading the entire page. This enables the user to interact with several tabs concurrently in the Data Vault tabbed application design. For example, a user can request to see a source in Google Sky, and immediately return to the search results through the Data Vault tab interface, without requiring a page refresh or a new search.



	Project	RA	DEC	Observer	Year	Source
	GBT AGBT03B_034_01	148.888	69.065	Thacker/Langston	2003	M81
	GBT AGBT03B_034_02	115.000	66.500	Rubin/Langston	2003	M81
	T03B_034_02	148.888	69.065	Rubin/Langston	2003	M81
	GBT AGBT03B_034_02	148.888	69.065	Rubin/Langston	2003	M81
	GBT AGBT03B_034_03	115.000	69.500	Rubin/Langston	2003	M81
	GBT AGBT03B_034_03	145.000	66.500	Rubin/Langston	2003	M81
	GBT AGBT03B_034_04	115.000	69.500	Rubin/Langston	2003	M81
	GBT AGBT03B_034_04	115.000	72.500	Rubin/Langston	2003	M81
	GBT AGBT03B_034_04	145.000	69.500	Rubin/Langston	2003	M81
	GBT AGBT03B_034_04	145.000	72.500	Rubin/Langston	2003	M81
	GBT AGBT03B_034_05	148.966	69.679	Langston/Rubin	2003	M81
	VLA AB0554	151.522	70.374	Gottesman, Stephen T.	1990	M81DB
	VLA AB0987	148.888	69.066	Bower, Geoffrey	2001	M81
	VLA AB0998	148.888	69.066	Bower, Geoffrey	2001	M81
	VLA AB1007	122.504	71.269	van Gorkom, Jacqueline	2001	M81
	VLA AB1007	122.532	70.902	van Gorkom, Jacqueline	2001	M81
	VLA AB1164	148.888	69.066	Bower, Geoffrey	2005	M81
	VLA AB1255	148.888	69.066	unknown	2007	M81

Figure 5. Above is a view of Data Vault's search results. Placing the mouse over a proposal icon instructs the user that clicking will allow the user to view the proposal for the project.

4. IMPLEMENTING A SERVICE MODEL

So far, developers and early reviewers like the "services model" of the Data Vault application, where various web services, scripts, and the like are combined into a single web application that performs well (i.e. loads quickly & is responsive) and has the new NRAO look & feel (which was rolled out in early 2008). Pushing this forward, Data Vault may enter a get-things-done mode of operation where developers will rapidly implement new simple-but-valuable features by integrating hacks into a single clean, beautiful application.

4.1 From search to data delivery

The Data Vault's primary purpose is to provide a single "Google-like" free-text search box to allow users to find and download data products they seek, and further to identify and download relevant data products they didn't know existed.

Current developments have a functional cross-telescope free-text search for NRAO telescopes given current resources, with searches taking < 1 second (best case) to no more than 1 minute (worst case, uncommon) with several opportunities for query optimization as features stabilize.

In order to go into production for retrieval of raw telescope products, Data Vault met the following needs:

- For each telescope, identify:
    - maintainer of science data metadata/registry
    - maintainer of physical storage for science data products
    - maintainer of data download service, linking a web browser to some form of data retrieval
- For the Data Vault application, identify:
    - maintainer of free-text search execution
    - maintainer of user interface
    - helpdesk or equivalent

The key is sustainability. If the Data Vault is to succeed, it ought to have responsible, capable, interested maintainers and a mechanism by which to manage all of these processes (e.g. helpdesk). Some of these roles have obvious candidates, while others should choose candidates carefully. Some may even need to be broken down later into sub-processes.

4.2 Value-added services

The Data Vault's secondary purpose is to link to and combine value-added services in the form of maintained web services, prototypes, and just plain dirty hacks.

As an early Perl-based GBT scheduling block exercise indicates, the Data Vault can easily integrate simple, high-impact services very quickly. This is exactly the type of service professionals can contribute to the Data Vault: a small scope script/application based on a few parameters readily available in the search results of the archive.

To support installed services, Data Vault met the following needs:

- For Data Vault services, identify:
    - application architect(s) to lead moderation and integration of new services
    - maintainer for each installed service
- Identify early Data Vault services in development:
    - Google Sky
    - VLA/VLBA Image Search
    - Proposal Viewer
    - GBT data (GO) browser

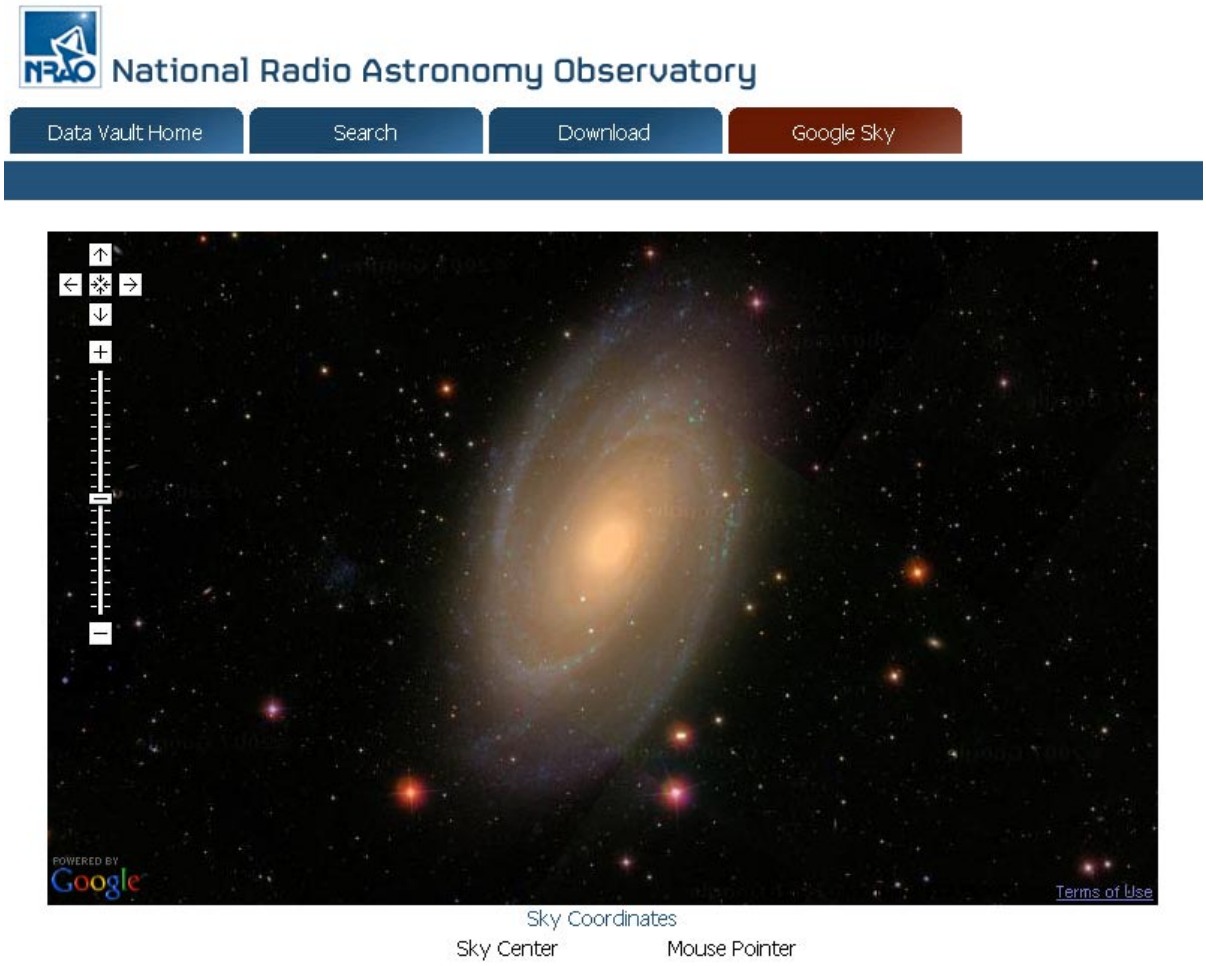


Figure 6. Google Sky embeds nicely into the tabbed Data Vault application.

An installed service can be anything from a web CGI script to use of a public API to a useful download for a specific application or software package. For example, Google provides public access to the Maps API, which includes Google Sky; Figure 6 illustrates Google Sky's integration into the Data Vault.

Most installed services will start as a simple link to some URI (Uniform Resource Identifier), and they could easily remain in this state indefinitely. Optionally, these can fully integrate into the Data Vault where the service produces an XML response to a URI request and the service maintainer (or a GWT developer) develops client-side GWT support.

As GWT has a learning curve, the logical intermediate step is to produce a style template which service developers use, or style can be ignored leaving users to view 90s era web pages.

4.3 Data Vault: web pages or a web application?

When choosing web technologies for the Data Vault, one question persisted: "Is this a web application or a collection of dynamically generated web pages?" Today, the answer is "Both." Data Vault at its core is a web application. It is a load-it-once, fast-response application complete with Ajax and a model-view-controller pattern.

Through installed services, though, Data Vault is an exercise of classic web development, a collection of small-to-medium size pages, both static and dynamic. This identity mix-up may cause for embarrassment if not handled properly.

4.4 Keeping things clean under the rug

Too many dirty hacks will compromise the esteem of respectful engineer, so why is this approach a good idea? The NRAO staff seems rather amenable to development of individual software packages and services. The Data Vault can combine these types of services and perform very well, **if** regular testing and validation is performed **and** an architect (possibly architects) ensure progress.

The following practices will help achieve just that:

- enforce unit tests on all services, and run these tests daily
- maintain concise documents on service interfaces, scope, and limitations
- combine all small-to-medium NRAO services into a single server (archive.cv.nrao.edu), unless special circumstances prevail
- run all background/daemon services through a web-based process management tool (such as Supervisor[†])
- enforce active ownership of all other installed services
- maintain a plan of record

4.5 Different is good

As witnessed to date, the VLA/VLBA and GBT archive search tools are very different in feature sets and philosophy. These differences can be a very good thing in order to try different approaches to satisfying user needs. For example, the VLA/VLBA download tool e-mails users when data is available and the GBT download tool grabs data on-demand for immediate download. These download services cross-fertilize, and the future holds a hybrid approach for the Data Vault. This may provide for rapid development of new services.

4.6 A Win-Win-Win situation

Too many services can be frustrating to the user. Perhaps the Data Vault should support user options for which services to include and which to ignore, through cookies and/or user login. This way, users can tailor the Data Vault experience to the services best suited to their needs.

Further, this will allow Data Vault developers to rapidly integrate new features and for non-Data Vault developers to contribute in meaningful ways (thereby becoming Data Vault developers). This is a recipe for more off-budget development time, higher user traffic, and hopefully, higher user satisfaction.

5. CONCLUSIONS

The Data Archive System remains quite useful. Through a service model, Data Vault is able to link in many of the services that the Data Archive System provides. Familiar users may prefer to use the pre-Data Vault tools for certain needs.

Building a model-controller-view application with a minimalist server component, web.py, and a heavyweight client component, GWT, provides a unique opportunity. As of Spring 2008, GWT does not have strong XML parsing capabilities, but once advanced XML parsing is available (or custom-built, e.g. Data Vault), an agile server framework with web.py provides quick results while GWT greatly expands potential for application usability.

By shifting the focus of the Data Vault to network management, the development team is setting itself up for substantial contributions where any interested party can submit or integrate a service into the Data Vault infrastructure. Through this approach, the Data Vault application is significantly increasing its participative capacity to get users involved in the services. If approached carefully, this design methodology – provide a central service that can link together various services based on core information (project IDs, observer names, source names, etc) – could lead to faster expansion and higher usability.

[†] Supervisor is a Python-based process management tool; see http://supervisord.org.

ACKNOWLEDGEMENTS

Thanks to Nicole Radziwill, Assistant Director, Office of End to End Operations at NRAO, for technical contributions and continued leadership of the Data Vault and to Dana Balser and Anthony Remijan at NRAO for scientific support.

Thanks to Ryan Scranton and the Google Sky team for their ongoing collaboration with NRAO.

The National Radio Astronomy Observatory is a facility of the National Science Foundation, operated under cooperative agreement by Associated Universities, Inc.

REFERENCES

[1] ESO Data Management Division, "Next Generation Archive System," http://www.eso.org/projects/dfs/dfs-shared/web/ngas/, (2006).

[2] Butler, B., and Myers, S., "EVLA e2e Science Software Requirements," http://www.nrao.edu/~smyers/evla/evla_requirements.pdf, (2003).

[3] Norris, R., Andernach, H., Eichhorn, G., Genova, F., Griffin, E., Hanisch, R., Kembhavi, A., Kennicutt, R., Richards, A., "Astronomical Data Management," XXVIth IAU General Assembly, (2006).

[4] Reenskaug, T., "MVC XEROX PARC 1978-79," http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html (1979).

[5] Swartz, A., "web.py 0.23 documentation (web.py)," http://webpy.org/docs, (2008).

[6] Dewsbury, R., [Google™ Web Toolkit Applications] Prentice Hall, (2007).