Developing Web Interfaces for Scientific Data Archives

Thomas C. Stein [(1)], Keith J. Bennett [(2)], Daniel M. Scholes [(3)]

[(1)] Washington University in St. Louis
NASA Planetary Data System Geosciences Node
Campus Box 1169, 1 Brookings Drive, St. Louis, MO 63130 USA
EMail: stein@wunder.wustl.edu

[(2)] EMail: bennett@wustl.edu, [(3)] EMail: scholes@wunder.wustl.edu

ABSTRACT

Web interfaces are the primary tools for accessing archived scientific data sets. Developing these interfaces can be a challenging task, especially when the data users are not the data producers. An effective interface reflects a symbiotic relationship between the data producer, the web developer, and the user community. Key to creating a useful interface is an understanding of what the data measure, how the data are represented, and how the user will search for and use the data. The developer must bridge the gap between producer and user.

For ten years, the Geosciences Node, a discipline-based node of the Planetary Data System (PDS), has developed web interfaces for data from NASA's missions to the terrestrial planets. Two flagship web tools, the Analyst's Notebook and Orbital Data Explorer, provide access to Node data archives. Based on this experience, we have established standard approaches and techniques for developing web interfaces to scientific data archives.

Keywords: web interface, data archive, design, development

INTRODUCTION

The widespread availability of high-speed networks to scientific researchers has made electronic access the primary means for locating and disseminating archived science data sets. The ubiquitous nature of web browsing provides a common foundation for interfacing between the researcher and the data that he or she seeks.

Several models exist for providing web-based access to science data to users. The most common is a browser-based interface. Some interfaces merely supporting directory browsing via FTP or HTTP protocols. More complex is the use of web services that allow client applications connections, such as for ArcGIS and Google Earth. In some cases, client applications are developed for searching, acquiring, and working with specific science archives.

Developing web interfaces to scientific data archives requires planning and foresight, and, as shall be seen, specialized knowledge of information systems, data archiving, science data, and user design. Most importantly, developers must understand that interface development is an iterative process that usually requires compromises in allocating available resources such as system infrastructure, developer time, and functionality.

Who's in charge?

A web interface may be produced by one of several entities, such as data producers, data archivists, technical specialists, and user interface design specialists. In fact, each of these groups has specialized knowledge that is required for a robust interface.

Data producers often provide simple access to data, but in many cases the responsibility for providing access to data archives falls to another entity. At first glance, the data producer may seem like the best choice to provide access because of first-hand knowledge and experience using the data. In reality, data providers are rarely funded to support other scientists' requests for data, especially in the long term. Their expertise lies with the data, not with making the data available.

Data archivists have expertise working with data—organizing data stores, creating search engines, and delivering data to users. Technical specialists understand well information architecture and the infrastructure that underlies web interfaces. User interface design specialists understand the subtleties in developing human-computer interactions that are intuitive and functional, although lessons learned from e-commerce interfaces may not be applicable to science interfaces.

Entities developing interfaces to science data must incorporate expertise from all four of these knowledge domains (figure 1). Interface developers that start in one domain will expend resources to gain expertise in other areas. This may include improved communications with a data producer and a better understanding of the scientific theory behind the data. Technical abilities and capabilities may need updating. In some cases, research and training can give development teams sufficient knowledge, though adding team members with specialized knowledge can be most useful.

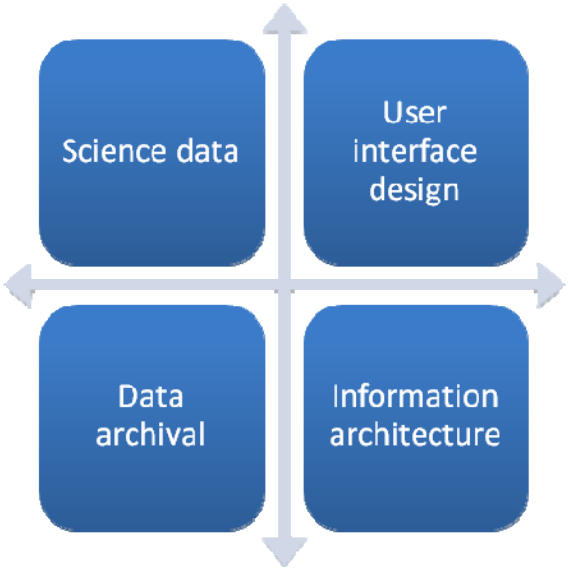

Figure 1: Knowledge domains that provide input to interface development.

The Planetary Data Systems Geosciences Node is primarily a data archiving entity. However, several members have extensive, first-hand knowledge in other domains of science data, information architecture, and user interface design. Advantages to this approach enable development of web interfaces that support cross-mission and cross-instrument capabilities. In order to integrate related data from multiple sources, additional specialized knowledge of data interactions is required.

GEOSCIENCES NODE EXPERIENCE

The Geosciences Node of NASA's Planetary Data System was formed in 1988 with the charge of producing, maintaining, and disseminating archives of geosciences-related mission data for Mars, Venus, Mercury, and planetary moons. Prior to the late 1990s, digital data volumes were small enough that entire data sets could be effectively distributed on CD-ROM or DVD-ROM disks. Scientists were able to use index tables included on the disks to locate data products of choice.

As the new millennium approached, a number of events changed this approach. First, data volumes were becoming sufficiently large that physical media were no longer cost effective. Anecdotal evidence showed that in the case of the 42-volume Viking Orbiter CD-ROM set from the early 1990s, a majority

of the disks remained unopened by researchers because only a portion of the entire data set was used by a given scientist.

From 2000 to 2007, Geosciences Node archive holdings grew a hundred-fold, from 0.2 TB to 20 TB. The data archive is expected to exceed 200 TB in the next five years (figure 2), not including data storage requirements for processing. In addition, data storage required to support web interfaces is expected to be 35 TB in 2014.

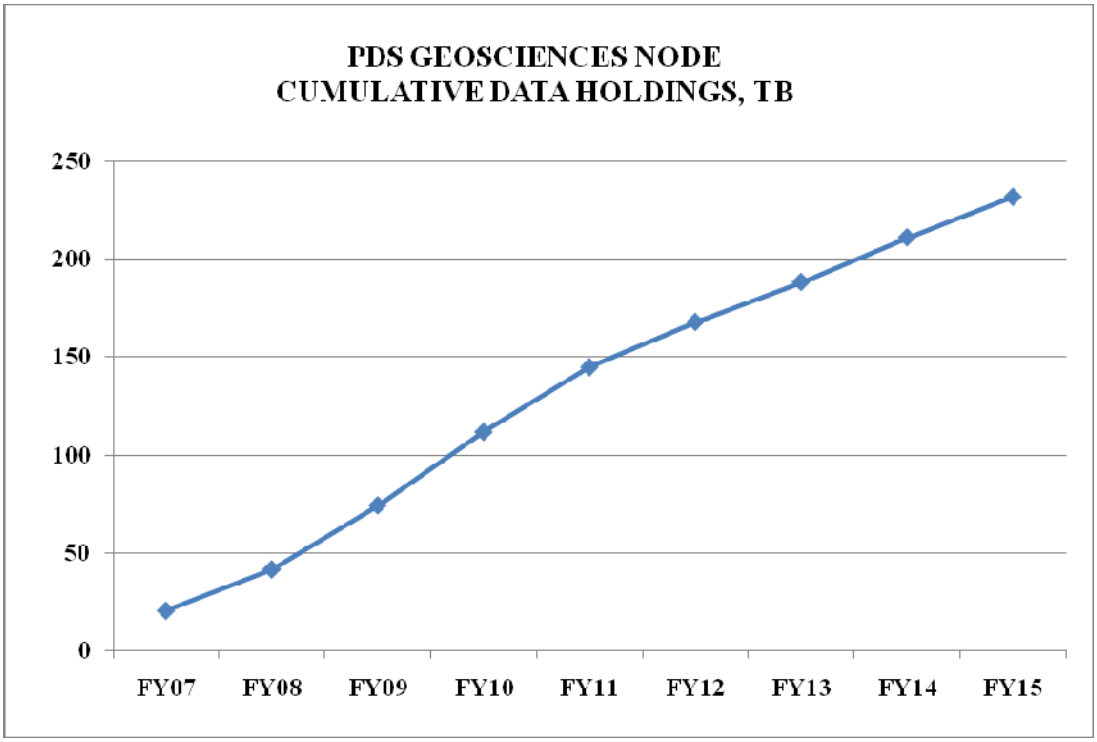

Figure 2: Expect increase in Geosciences Node archive size over time.

Second, archive updates were costly to disseminate with the use of physical media distribution. Data providers (scientists within the missions) were required to release archived data on a regular basis, typically within six months of collection. The calibration algorithms of some instruments (e.g., MRO CRISM) are refined over a period of years. As a result, entire data sets may be updated several times during the lifetime of a mission. With a data volume of 10s of TB, data set updates are not a trivial process.

Third, web interface technology now supported simple search capabilities and electronic data delivery to most end users. Although it was (and still is) not feasible to reasonably deliver the largest data sets in the archive electronically, individual product downloads were becoming routine. At the same time, tools were available to assist with developing web interfaces beyond the early hand-coded static pages.

Finally, the nature of the data archives was changing. As different instruments from numerous missions collected studied the same targets, researchers needed a way to locate and access correlated data. In addition, landed missions, which would be nondeterministic by nature, were expected to produce not only instrument data, but associated documents and other metadata that detailed the decision-making process behind the data collection.

Early web interfaces were simple, allowing users to traverse archive volume directory structures and download individual files. Existing archive documentation could also be viewed. As development continued, users were presented with static product pages that include browse images, but metadata remained accessible only by viewing index tables or downloading and opening data products.

Search and retrieval schemes involving data from multiple sources raised additional challenges. For the Geosciences Node, cross-instrument/mission search allowed users to find related data from multiple sources for a given location or time. For example, a user may wish to find all images for a given crater on Mars. This might involve images from recent spacecraft such as the Mars Reconnaissance Orbiter (MRO) HiRISE, CTX, and CRISM imaging instruments as well as older Mars Global Surveyor (MGS) Mars Orbital Camera instrument. There are a number of caveats for the developer. The data must be "normalized" across the instruments to allow common searches. In this example, the location data from the MRO instruments is in a different coordinate system than the MOC instrument. The data archive may be located at multiple sites around the world (three different sites in this example). In addition, the data may be stored in subtly different data forms or archive structures based on the data providers needs. Finally, there was the challenge of how to provide a simple interface for users to make such cross-instrument/cross-mission searches.

As a result, the Geosciences Node has developed two web interfaces to its archived science data. The Analyst's Notebook allows users to explore recent landed NASA missions Mars Exploration Rovers (MER) and Mars Phoenix landed missions. The Notebook combines sequence information, engineering and science data, and documentation to provide replay of these non-deterministic missions. The Orbital Data Explorer (ODE) supports cross-instrument and cross-mission data search and delivery functions for Mars, Lunar, and Mercury NASA and ESA missions, including Mars Reconnaissance Orbiter (MRO) and Mars Express (MEX).

The remainder of this paper discusses the design decisions, methodology, and lessons learned from implementing these two systems. The Analyst's Notebook development began as part of the 1999 Mars rover field test. MER, Phoenix, and Apollo Notebooks are now available, and additional Notebooks are in development for LCROSS and Mars Science Laboratory. Current Mars, Lunar, and Mercury versions of the ODE contain data from MRO, MEX, MGS, Clementine, Lunar Prospector, and MESSENGER.

DEFINE REQUIREMENTS

Before development on a web interface begins, it is imperative that developers have a clear understanding of the interface purpose and how it relates to the science archive. This can be accomplished by addressing these questions: what is the intent of the interface? What type of information is to be included? How will the information be acquired? Who are the end users? What are end users expectations?

Creating a usable science archive interface starts with requirements. These may be defined by a combination of the directing agency, the developers, and the end users. Requirements should cover data sources, functionality, deliverables, security, hardware/software limitations, maintenance, and cost. Ideally, requirements contain specific, attainable goals, such as "the user should be able to preview images and graphs representing the data prior to download."

For small projects, the requirements may be somewhat broad and flexible, becoming more defined and detailed over time. For large projects, requirements may go into considerable detail and be comparably lengthy from the start.

Use cases are useful in creating the functional requirements that detail the services, tasks, and functions of the interface [9]. Each use case describes a sequence of interactions between the components (or *actors*, in the language of use cases) of the system that accomplishes a goal. Examples are a user who getting documentation for a specific data set in the archive, and ingestion of archive data into the interface database.

An end-to-end design flow should emerge from the requirements. The interface development continues with the following phases: determine data sources, develop information architecture, choose development environment, design the interface, implementation, testing and feedback, maintenance, and enhancement (figure 3).

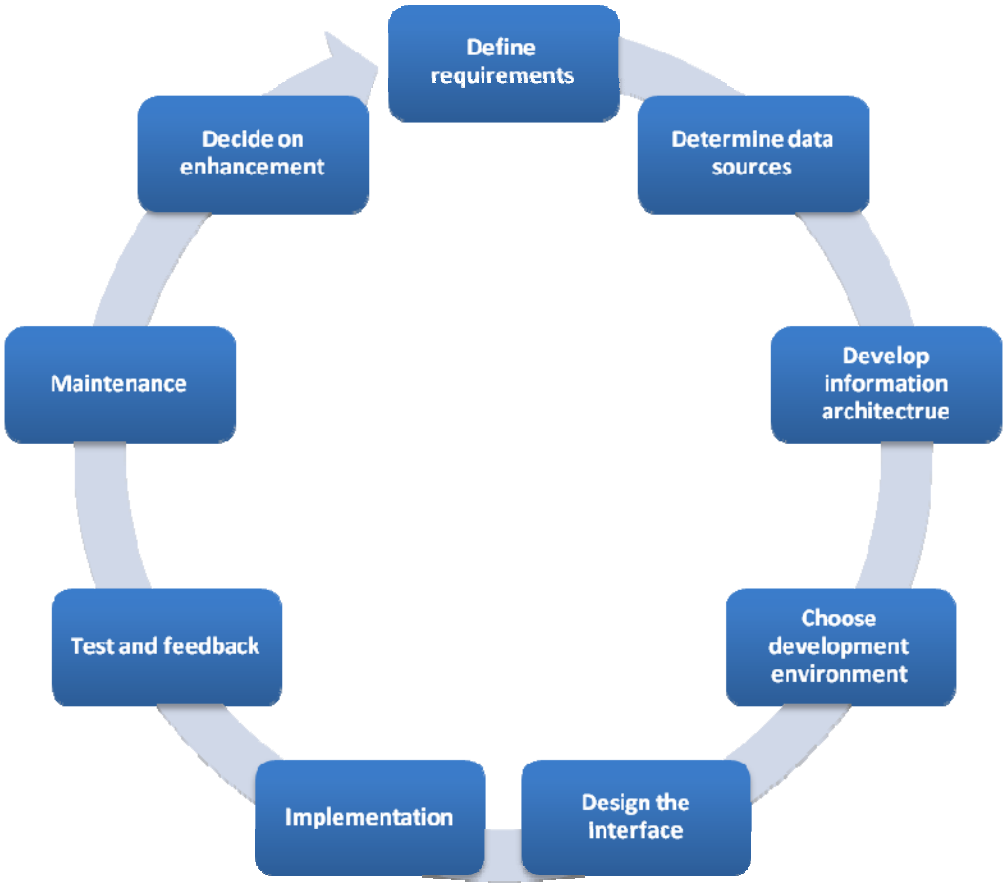

Figure 3 : Phases of interface development. Note that the process is, in reality, an iterative process.

In reality, the greatest single factor driving the requirements is almost always resources, in terms of both time and money. After the latest models have been considered and the best practices have been studied, the biggest decisions will usually be based on available resources. In terms of schedule, a great tool in the future is of less value than a good tool now.

DETERMINE DATA SOURCES

The information required to support a functioning interface is more than simply a collection of data files. Better, more complete interfaces require the integration of a number of data sources, including data product metadata, archive documentation (about both the data set and the archive), and non-archive documentation from data providers and others with first-hand knowledge. Tracking down these sources can be challenging. It may be advantageous to work with data providers prior to and during data collection when possible.

At this point in development, political or ethical considerations may need to be made. Perhaps the developer decides to include browse (sample) versions of the data, but the data provider has not created any. If the developer chooses to create the browse versions, what obligation does he or she have to working with the data provider?

It may be that data may have a validation or proprietary period during which they might be available to the interface developer but not for public release. In the case of the Analyst's Notebook, team versions containing unreleased data and documents are password protected to limit access. When data are publicly released, they also are made available in the public versions.

A particularly important area for U.S. developers is the International Traffic in Arms Regulations (ITAR) that controls export of defense-related articles. This is a concern for both of our example interfaces (ODE and the Analyst's Notebook) because all spacecraft are covered by this regulation. For example, some original calibration documents were found to contain sensitive material and had to be modified to allow public release. These documents required editing and additional review in order to be publicly released.

Data

The foundation of the interface is the data. While users will require supporting documentation, the user is primarily seeking data. The interface serves as a portal between the archive (an extension of the data producer) and the user. In fact, users may fail to distinguish between the interface and the archive. When a user experiences difficulty with the interface, they may transfer their resulting emotions to the archive, and perhaps even to the data producer.

Data quality is of prime importance. Because the data are part of an archive, one would expect them to be well-formed and well-documented. Many times this is not the case. The fault may lie with the data producer or archivist (or both). Regardless, interface developers may have to invest considerable resources making the data ready for inclusion in the interface.

For example, a major aspect of developing a unified searchable data system involves ensuring all data in the system are based on the same definition. A case in point is the Mars location data stored in ODE, which uses the areocentric coordinate system used by MRO instruments such as the HiRISE high resolution camera, CTX camera, and CRISM hyperspectral imager. This is a planetocentric, positive east coordinate frame. All instruments from MRO use this coordinate system. But including other, older Mars datasets requires care to ensure that location data are stored in the same coordinate system. For example, the Mars Global Surveyor's Mars Orbital Camera (MOC) data is stored in planetographic, positive west coordinates, and thus must be reprojected in the planetocentric, positive east coordinates before storing in ODE. As a result, users can search for both MRO and MOC products simultaneously. Users who spend most of their time working with MOC data must convert their search inputs into the new coordinate system, however.

When data are recast (have a different format, are subset into smaller parts, are joined to form a larger data product), the transformation should be clearly labeled and described. In most cases, the user should have the option of accessing the original data as contained in the archive.

Metadata

Often defined as "data about the data", metadata describe the science data, and are used primarily to support search queries and to facilitate use of the data. The metadata of an image, for example, may include its size in pixels, the data type, and time of acquisition. In a rich data set, the metadata may include instrument state, environmental conditions, data processing steps, and applied calibration, among others. Some metadata may be considered scientific data on their own.

Metadata are found in a variety of sources. Most often, they are contained with individual data products as an internal or external label, or in an index table. Developers must carefully consider how to use the available metadata. It is likely that only a subset of the metadata are needed for query and other interface functions. Conversion of metadata, such as in the reprojection example cited above, should be done in a way that accessibility to the original values is maintained.

Documentation

Creating documentation is low on most data providers' action item lists, perhaps higher only than creating the actual archive. Without good documentation, users will be at lost when attempting to understand the data. Before that point, though, the interface developer will have intimate knowledge of both the archive and data documents as part of preparing the interface. As data and metadata are loaded into the interface information infrastructure, numerous documents will be scoured for file formats, data content, and other clues needed to populate the interface database.

Most developers will encounter several types of documentation. Data product documents (including calibration and format information) are produced by the data provider. Archive documents may be created by the data provider or archiving entity. Additional documents may be generated during the data collection phase by a number of people directly and indirectly involved. In the case of the Phoenix Analyst's Notebook, daily reports from these non-deterministic missions were captured to preserve the knowledge and intent behind the decisions. After the mission ended, some scientists prepared temporal summaries of major processes such as coordinated observations and soil digging that were incorporated into the interface.

It can be difficult to capture all documentation related to the archive. Fostering good working relations and communications between the developer and the documentation producers usually leads to a more complete document set within the interface. This was accomplished for the Phoenix Analyst's Notebook by embedding a developer within the mission science team and allowing the team to use the interface before public release. In cases like this, care must be taken that restricted documentation used to create the interface is not revealed to the user.

Additional inputs

The amount of non-archive information used to support archive interfaces in increasing. This can include observation notes and additional data products. Traditional archive interfaces provide access to data, but do not contain information that provides intent—answers to questions about what was happening during data acquisition. Why was the observation made? What special circumstances caused the change in operations? What happened that allowed higher quality data than originally planned?

As the interface matures and developer-data producer interactions improve, opportunities for gathering these additional inputs increase. When possible, collaborations should occur as early as possible. Such information can be difficult to find, but harder—and sometimes impossible—to regenerate.

Developers should expect increased inputs for interfaces that support data from multiple sources. For example, both the Analyst's Notebook and ODE feature tools for locating and viewing data from coordinated observations. Considerable effort was expended to obtain sufficient input from science team members to support the interface.

Another example of is the daily activity timeline in the Phoenix Analyst's Notebook. The activity sequence plans were provided by the science team and ingested into the interface database. Collaboration between the data providers, archive producers, and developers improved the metadata within the data products to facilitate associate between the planned activities and the resulting data. In turn, the web interface allows the user to easily discover the data products acquired for any given activity in the timeline.

A final example is that of browse images in ODE. While some image datasets include browse versions, many do not. Initially, the MRO SHARAD ground penetrating radar data set did not include browse images. In this case, interface developers worked with data providers to implement the processing algorithms that matched the providers' requirements. The developers were then responsible for generating the browse images. In other cases, the web developers generated browse images directly from data without involvement of data providers. This is typical when the data are older and the providers are no longer available.

DEVELOP INFORMATION ARCHITECTURE

There are technical aspects of the interface that must be considered prior to implementation. Once the requirements have been defined, developing the information architecture lays the foundation for other steps leading to implementation. Information architecture, in this case, models the system in which the interface is developed and made operational, thereby facilitating planning, management, and implementation of the system.

For smaller projects, the architecture may be largely focused on the functional and deployment views (as defined in ISO 42010 [5]). Larger projects will likely produce more formal architectures and also incorporate more complete views of code, process, and feedback.

The stakeholders—developers, end users, data providers, and others with an interest in the interface—interact with the system in their own way. Considering the system in terms of relationships to the stakeholders drives the architecture, which in turn gives direction to the developer.

The end user will be affected by requirements placed on them. For example, users of the Spitzer Science Archive are required to download client software ("Leopard") to access data [1]. A number of interfaces exist for the European Southern Observatory (ESO) Science Archive. One interface is browser-based, although users must be registered in order to retrieve data. The VirGO visual browser for ESO science archive connects users of Stellarium virtual planetarium software to the science archives [2]. Data providers will care about how the data are represented in the interface. On the other hand, the developer will be concerned about getting proper documentation from both the data provider and the archive entity.

Longevity and availability are also important drivers for the information architecture. Because the interface will provide access to archive data, developers must consider how well the architecture will support the interface for the long term. Windhouwer and Dimitriadis have developed a method with an archive-specific API that in turn supports a generic user interface [15]. Updating the interface still would require resources, though, maintaining interfaces that supported real-time, interactive processing would remain be especially challenging.

CHOOSE DEVELOPMENT ENVIRONMENT

Given the longevity expected of an archive interface, selection of the proper development environment is important. Factors in this decision include selected information architecture, size of development team, and the technologies required to support planned interface. The maturity of the development environment, like that of the technology infrastructure, can increase long term stability of the interface while minimizing the effort required maintaining functionality.

In most cases, the developer will benefit from a well-established integrated development environment (IDE) that is easy to use, help with code development (real-time syntax checking, type-assist functions that help with code completion), provide robust testing/debugging, and support multi-developer projects.

A concern often raised about developers is their tendency to embrace new, often immature, technologies. This can happen in both development and implementation. It is recommended that developers approach use of web technologies as early adopters or early majority. (Rogers defines this position as trying out new ideas, but in a careful way [12].) By doing so, developers can take advantage of recent innovations while creating an interface that is stable, usable across a variety of client platforms, and easy to maintain. The long operational life of archive interfaces offers—in reality, demands—that developers look to incorporate newer technologies during the maintenance and enhancement phases. Thus a technology that is not mature enough upon the initial release of an interface may be incorporated in future releases.

DESIGNING THE INTERFACE

Although it may seem that all of the difficult work has been completed at this point (defining requirements, selecting appropriate development and operations platforms, etc.), interface design should be carefully considered. In some cases, the web interface under development may provide the only access to a given archive, but that should not be taken as license for poor design.

The web interface is the first part of an archive data set that many users see. Bad page design can impede users, even if there is useful functionality underneath. More importantly, it may turn users away from quality data. Because the highest adoption rate for new innovations occurs via peer-to-peer recommendations [12], users' initial perception is a significant factor in the success of the interface. Not only will a dissatisfied user fail to recommend the interface, but is likely to disparage the interface

whenever the opportunity arises. In fact, users may fail to distinguish between the interface and the archive. When a user experiences difficulty with the interface, he or she may transfer their resulting emotions to the archive, and perhaps even to the data producer.

Functional requirements and user scenarios provide strong boundaries to minimize scope creep and focus interface design. But how should the interface appear to the user, and what will the interface affect the user experience? These questions are addressed by considering the design's readability and usability.

Readability

Readability covers a broad range of factors that can be the difference between a marginal interface and a successful one. The factors include design elements such as color and images (including icons). In general, readability elements may be considered hygiene factors [16]. As an analog to Herzberg's hygiene-motivational workplace theory [3], Zhang showed that good use of these hygiene factors contributes to basic website functionality, but poor use creates user dissatisfaction.

Without question, color affects the user's attitude toward the interface as well as the usability of the site. Color theory provides certain principles of correct use: blue is a background rather than a foreground color; green text on red background is unreadable. Physical factors, such as color blindness, play a role. Cultural color associations may merit consideration, though they are less important in web design. Finally, standard practices dictate some color choices (e.g., users expect blue text to represent web links).

Images, especially in the form of icons, can provide subconscious cues to users and improve readability. When possible, developers should employ icons conventionally. (For example, a pair of scissors is commonly used to represent the "cut" function.) By doing so, the amount of end user training required is reduced. In addition, judicious use of images (appropriate size, metaphor, and consistency) can have a large impact on how a web interface is received. Lightner's research shows that use of too many images damaged users' perceptions of professionalism [8]. As a result, usage often declines.

Readability is affected by a number of other factors as well, including white space, navigational aids, text, font choice, and layout. With science archives, developers often face the challenge of displaying a relatively large amount of information without confusing or "overloading" the user. In the Phoenix Analyst's Notebook interface, one user view (figure 4) shows a list of data products on the left side of the screen, and detail of a single product on the right side. Within the view are several areas for navigating the interface: main navigation tabs at the top, sol (Mars day) selector, report selector, and list of products, and product view navigation menu. The example view employs several techniques to enhance readability despite the dense page coverage: use of background color to link related elements; use of font weight and size to express hierarchy among elements; consistent use of icons within the site (the golden product header background color matches the golden "S" icon used to identify all data from this instrument); white space is used to promote readability within the product list; the question mark icon provides a link to system help.

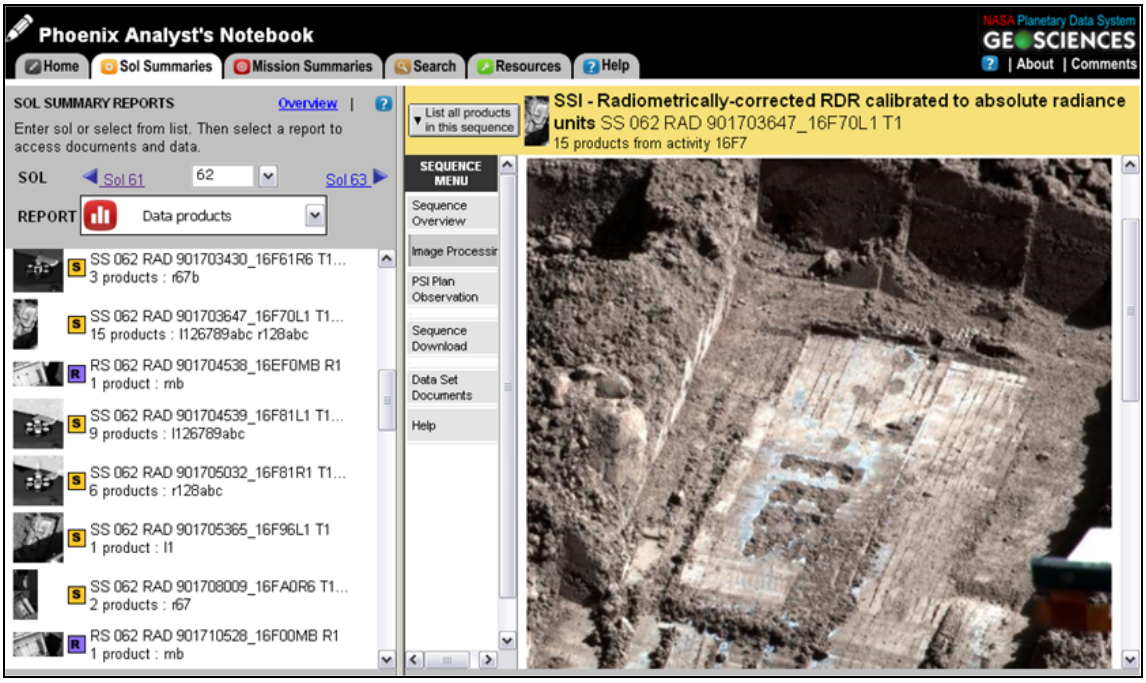

Figure 4. Example user view from the Phoenix Analyst's Notebook. Despite the dense coverage of information, a number of techniques are employed to improve readability. Standard interface practices, such as tabbed menus at the top of the page and locating contact information in the top right, reduce the need for user training.

Despite the ubiquity of web browsers, today's users expect high performance from web interfaces. Indeed, interfaces are often rich applications with sophisticated tools. Users are becoming increasingly computer savvy and more demanding. Rare now are interfaces that do not require users to support JavaScript, Java, Flash, or other similar application environments. This dependency, in addition to discrepancies in how different browsers interpret HTML code and cascading style sheets (CSS), makes development of a uniform, readable interface challenging at best.

Usability

More important for user satisfaction than the readability is an interface's usability [13], which is defined as "the capability of the software product to be understood, learned, used and attractive to the user…" [4]. In measuring of the quality of a user's experience, usability is divided into three areas: learnability, understandability, and operability.

Developers should consider how to deal with first-time users. Will there be on-screen prompts to assist new users? How quickly can a new user learn the interface well enough to successfully use it? How many resources should be expended to provide a differentiated experience depending on user sophistication? Beyond these questions, learnability can be addressed by interface consistency and intuitiveness. For example, navigation elements should produce the same result wherever they appear. Icons should follow common usage when possible, such as a question mark representing help.

Clearly identifiable navigation aids are key to directing the user, especially during multi-step procedures. For example, ODE guides users through its data product search screen with explicitly numbered steps and notations indicating optional and required search criteria. Miller and Remington researched web navigation and found that having clear labels is more important than having good organization [11]. Continuing along these lines, Wang and Emurian found that consistency and simplicity improve navigation [14]. As a result, trust increases, and so does usability.

The concept of usability also addresses the understandability of the interface. While use of common knowledge throughout an interface is important, developers must take care when assuming a user's knowledge set. For example, the location field of a search form may require west longitudes to be represented as negative numbers. Another system may append an "E" or "W" to denote east or west. In such cases where multiple conventions are known to be in use, clear instructions should be given. Other barriers to understandability include use of unfamiliar technical jargon and unclear or non-existent descriptions and instructions.

This latest example suggests a requirement that developers understand the expected user domain knowledge. Developers likely will need input from scientists well versed in the data. When users will have varying degrees of domain knowledge, the interface must provide guidance to the uninitiated without hindering the expert. This is true for both the knowledge domain as well as the technical expertise expected of the user. A tenured scientist with a thorough understanding of the data may be lacking in skills with web interfaces. Experiences shows that engaging such users with "how to" teaching, such as brief video tutorials or "recipes" describing how to accomplish specific tasks, can dramatically increase usability.

Despite the ability to create a readable and usable interface, operability may have significant implications on design. There are two considerations for operability: from the user's point of view and from the developer's point of view. The latter will be discussed in the section on maintenance.

Operability for the user can be measured by answering the question, how easily can a user find what he or she is looking for? The goal might be locating a data product, but could be information from a document or successful processing supported by the interface. Depending on the specific goal, the measure of success may vary considerably. For example, the same user may be expect a web page to load within seven seconds and a processing step to take less than two minutes, but be satisfied with waiting 24 hours for notification to start downloading a large amount of data.

Finally, consider the non-interactive portion of the interface relating to usability. In most cases, users will desire to download a subset of the data archive. Downloading a handful of small data products usually is simple and straightforward. When volume—either size or number of products (or both)—increases, delivery becomes more challenging. A data request may comprise numerous products along with related support documentation and files, all wrapped in a single file delivery package for easy electronic delivery. Doing so may take long enough that the user must asked to wait for further notification before accessing the data requested. In some cases, alternative delivery methods, such as shipment of physical media, may be required.

Once data are downloaded, they must be usable, but the original (archive) format may not satisfy the user's needs. As processing software and tools are created and updated over time, supported data formats will certainly change, so it is incumbent on the developer to incorporate data transformation services into the interface. Even in the case of an interface that supports data processing, the output will likely require some sort of transformation.

Requirements should provide much direction in how to approach usability concerns. Another important input is gathered from the testing and feedback phase.

IMPLEMENTATION, TESTING, AND FEEDBACK

The implementation phase does not begin once the planning and design work is complete, because the latter are really never complete. Inevitably there is design modification once implementation starts. Some requirements may be refined as well. However, once implementation starts, even the most well-defined interface can suffer from the reality of implementation [10]. New technology, schedule pressures, personnel changes, and requirement "scope creep" can all contribute to less than optimal results. Fruitful interactions with data providers, archive and design experts, and end users can suddenly become stale and unidirectional. Unrealistic deadlines lead developers to abandon implementation best practices in favor of quicker results.

Too often, testing is given a low budget in terms of effort, importance, and allotted time during development. Yet proper testing can help rescue bad implementation, or keep resources from being wasted when the interface does not address requirements (or the requirements do not address the need).

Testing is an ongoing process during the design phase and as functionality is added to the interface. It should begin early in the implementation phase and occur regularly. Developers should take advantage of users with various relationships to the interface. The test group might include representatives from the data provider, archive, end user, and design team populations. Some testers will be involved earlier in the process than others.

Another testing mechanism is the stress test, which simulates operating conditions under heavy load. When a developer or other tester can observe the interface during a stress test, problems can be discovered and corrected so that "release day surprises" can be minimized or avoided. In addition, testing outside the development environment can expose coding problems, errant assumptions, or unexpected issues. For example, remote user tests of ODE revealed a latency-related problem not evident in the local setting.

Feedback from the test group can have substantial impact on the development process. Developers should have direct contact with the testers when possible. Ideally, a developer should be able to observe the test user working with the interface in order to gather feedback that otherwise might not be reported. For example, during testing of the Phoenix Analyst's Notebook, a user made an off-hand commented that he would like to download the data in a particular format that was not supported. The developer chose to incorporate the suggestion into the interface and discovered that a number of other users found the addition useful.

A user forum can be helpful for gathering feedback as well as supporting users. Depending on the user community (size, technology expertise, expectations) and available resources, this might be in the form of e-mail communications, a web-based form, a traditional online forum, or wiki.

MAINTENANCE

The most overlooked aspect in interface development may be maintenance. Because the data are part of an archive, the interface should be designed to be available for several years at least. It is not unreasonable to think that the selected development platform and server operating system may become obsolete during the life of the interface. Technologies such as virtual servers can help alleviate this problem. However, a bigger concern may be loss of functionality due to changing and evolving web browser standards.

Resources must be allocated to maintaining the interface. In some cases data collection continues after the interface is operational. (This is true for both the Analyst's Notebook and ODE.) Interfaces should be regularly tested for broken links and loss of functionality.

For long term maintenance of complex systems, developers and system administrators may wish to operate the interface architecture in a virtual environment. This takes Windhouwer and Dimitriadis' TDS model a step further by adding more stability to the underlying infrastructure. Even if the virtual environment is successful, the client portion of an interface may need extensive reworking based on user system capabilities. Eventually, there will be a time when the interface must be updated, replaced, or retired.

Of course, keeping system design documentation and implementation documentation can help with maintenance, especially when development personnel changes occur. In addition, a set of specific unit testing procedures can provide assurance that maintenance changes do not adversely affect users.

ENHANCEMENTS

The final phase in the interface development cycle is enhancement. User abilities and expectations continue to change, as does the technology supporting the interface. The interface must be considered a "living tool" that accounts for and incorporates these changes.

Enhancements can come in many different scenarios: adding data from data providers; including data sets new to the archive (either newly acquired or newly incorporated data); providing "derived" or improved versions of existing data (e.g., creating on-the-fly data products from multiple sources); and making interface enhancements based on user experiences and requests. These changes are driven by changes in the target community such as user inputs, availability or loss of other community tools, and new data sources.

An example of changes in the Geosciences Node community is the increased emphasis by NASA on lunar studies. As a result, there currently is increased interest in older lunar datasets by the planetary science community. An example of changes in community tools is best exhibited by the recent introduction of Google Mars and Google Moon, which provide an entirely new, simple way for users to search data.

The MER Analyst's Notebook provides the opportunity to observe enhancements made to a web interface. Figure 5 shows a comparison of the same user view for a given data product from 2004 and 2009. Continued enhancements of the interface have lead to improved readability, usability, and functionality. Readability improvements include grouping related information and added white space. Usability changes include adding more navigation options, reducing the size of the title, and removing unnecessary text. Supporting information has been added, such as direct access to product documentation and in situ product maps.

Figure 5. Comparison of the same MER Analyst's Notebook user views (sol 84 product summary) from 2004 (left) and 2009 (right).

CONCLUSION

Developing web interfaces to archive data is a balancing act in which the requirements must be considered in terms of resources (time, money, development expertise). Unlike many interface projects, science data archive interfaces must be intentionally planned for longevity because the archive itself is meant to exist for the long term.

Developing interfaces to science data archives requires expertise from four knowledge domains: science data, data archival, user interface design, and information architecture. The phases of development are defining the requirements, determining data sources, developing the information architecture, choosing a development environment, designing and then implementing the interface, testing and feedback, and maintenance. Often a phase of deciding o enhancements will follow, and the development cycle begins again.

Creating an interface is not a trivial task. Even with the best designs, a number of issues can affect the actual flow of development (which in reality is not strictly linear), and ultimately, its success. Involving all system stakeholders early in the process, and maintaining open communications throughout the development process is the foundation for developing a successful interface

This work is supported under NASA Grant NNG05GB73G, "Planetary Data System Geosciences Node".

REFERENCES

[1] - J. Chavez, J: Spitzer Science Archive Interface. Proc. SPIE, v 6270 (2006)

[2] - F. Chéreau: VirGO: A Visual Browser for the ESO Science Archive Facility. Astronomical Data Analysis Software and Systems (ADASS) XVII, 394, p. 221 (2008)

[3] - F. Herzberg: One more time: how do you motivate employees? Harvard Business Review, 46 iss 1, pp 53-62 (1968)

[4] - ISO 9126-1: Software engineering—Product quality—Part 1: Quality Model, 2000 (2001)

[5] - ISO 42010:2007: Recommended Practice for Architectural Description of Software-intensive Systems (2007)

[6] - K.A. Walsh, C.M. Pancake, D.J. Wright, S. Haerer, F.J. Hanus: "Humane" Interfaces to Improve the Usability of Data Clearinghouses, Proceedings of GIScience2002, pp. 333-345 (2002)

[7] - N.J. Lightner: What users want in e-commerce design: effects of age, education and income. Ergonomics, 46:1, pp. 153-168 (2003)

[8] - G. Lindgaard, G. Fernandes, C. Dudek, J. Brown: Attention web designers: You have 50 milliseconds to make a good first impression! Behaviour & Information Technology, 25, 2: 115-126 (2006)

[9] - R. Malan, D. Bredemeyer: Functional Requirements and Use Cases. Bredemeyer Consulting white paper (1999)

[10] - V. Merlyn: Managing the Organizational Change That Comes with CASE. SIM Network, the society of information management newsletter (1991)

[11] - C.S. Miller and R.W. Remington: Modeling Information Navigation: Implications for Information Architecture. Human-Computer Interaction, 13(3), 225-271 (2004)

[12] - E.M. Rogers: “Elements of Diffusion” from Diffusion of Innovations, 2003, pp 1-37, Free Press (2003)

[13] - M. Thüring, S. Mahike: Usability, aesthetics and emotions in human-technology interaction. International Journal of Psychology, 42(4): 253-264 (2007)

[14] - Y.D. Wang, H.H. Emurian, H.H: An overview of online trust: Concepts, elements, and implications. Computers in Human Behavior 21: 105-125 (2005)

[15] - M. Windhouwer, A. Dimitriadis: Sustainable operability: keeping complex resources alive. Proceedings of the LREC 2008 Workshop Sustainability of Language Resources and Tools for Natural language Processing, (2008)

[16] - P. Zhang, et al.: A Two Factor Theory for Web Site Design. Proceedings of the 33rd Hawaii International Conference on System Science, IEEE Computer Society Press (2000)