REVIEW

Ten years of software sustainability at the Infrared Processing and Analysis Center

BY G. BRUCE BERRIMAN1,*, JOHN GOOD1, EWA DEELMAN2
AND ANASTASIA ALEXOV3

1Infrared Processing and Analysis Center, Caltech, Pasadena, CA, USA
2University of Southern California, Marina del Rey, CA, USA
3Astronomical Institute Anton Pannekoek, University of Amsterdam, The Netherlands

This paper presents a case study of an approach to sustainable software architecture that has been successfully applied over a period of 10 years to astronomy software services at the NASA Infrared Processing and Analysis Center (IPAC), Caltech (http://www.ipac.caltech.edu). The approach was developed in response to the need to build and maintain the NASA Infrared Science Archive (http://irsa.ipac.caltech.edu), NASA’s archive node for infrared astronomy datasets. When the archive opened for business in 1999 serving only two datasets, it was understood that the holdings would grow rapidly in size and diversity, and consequently in the number of queries and volume of data download. It was also understood that platforms and browsers would be modernized, that user interfaces would need to be replaced and that new functionality outside of the scope of the original specifications would be needed. The changes in scientific functionality over time are largely driven by the archive user community, whose interests are represented by a formal user panel. The approach has been extended to support four more major astronomy archives, which today host data from more than 40 missions and projects, to support a complete modernization of a powerful and unique legacy astronomy application for co-adding survey data, and to support deployment of MONTAGE, a powerful image mosaic engine for astronomy. The approach involves using a component-based architecture, designed from the outset to support sustainability, extensibility and portability. Although successful, the approach demands careful assessment of new and emerging technologies before adopting them, and attention to a disciplined approach to software engineering and maintenance. The paper concludes with a list of best practices for software sustainability that are based on 10 years of experience at IPAC.

Keywords: astronomy; archives; software sustainability; architecture; user communities

*Author for correspondence (gbb@ipac.caltech.edu).

One contribution of 12 to a Theme Issue ‘e-Science: novel research, new science and enduring impact’.
1. Introduction

The NASA Infrared Processing and Analysis Center (IPAC) was founded in 1983 to provide science support for the Infrared Astronomical Satellite (IRAS), which made the first all-sky imaging survey in the far-infrared. The center created IRAS science products and processing tools for the science community, and provided support for astronomers. Today, IPAC hosts data from more than 40 space-based and ground-based projects held across several archives and, since 1999, has met 35 million electronic data requests (averaging 3TB/month of data downloads). The Infrared Science Archive (IRSA) opened for business in 1999 serving data from IRAS, and now incorporates data from major projects such as the Two Micron All-Sky Survey (2MASS; http://www.ipac.caltech.edu/2mass/), and is archiving data from the active missions the Wide-field Infrared Survey Explorer (WISE; http://wise.ssl.berkeley.edu/) and the Spitzer Space Telescope (http://ssc.spitzer.caltech.edu/). The NASA Star and Exoplanet Database (NStED; http://nsted.ipac.caltech.edu) is NASA’s archive of record for data on exoplanets and their host stars. The Kepler Science Analysis System (KSAS) is an internal archive for the Kepler exoplanet search mission, and the Keck Observatory Archive (KOA; http://nexsci.caltech.edu/archives/koa/index.shtml) archives data from selected instruments at the W. K. Keck Observatory, Hawaii. The Cosmic Evolution Survey (COSMOS) is a deep multi-wavelength cosmological study along one line-of-sight that archives its datasets at IPAC (http://irsa.ipac.caltech.edu/data/COSMOS/).

The archives described above are called ‘the (IPAC) archives’, with the understanding that they have their own funding lines. The archives share a common mass storage file system, database storage and servers, database management system (DBMS), configuration management system and user support tools. They each operate dedicated Web servers hosting all of that archive’s end user applications, all built on a common software architecture. This paper describes this software architecture and its extension and evolution since it was designed in 1999 (§2); the integration into the architecture of Scanpi, an ageing yet powerful IRAS processing tool inherited from the IRAS mission (§3); and the development of a user community for the MONTAGE image mosaic engine (http://montage.ipac.caltech.edu), which has found a wide applicability in astronomy and computer science research (§4). We conclude by summarizing a set of best practices for software sustainability, based on a decade of experience at IPAC.

2. The development and evolution of a sustainable archive architecture

Before we describe the architecture, we outline the structure of astronomical datasets and typical archive use cases. This will provide context for understanding the architecture.

(a) Science datasets in astronomy and archive use cases

Generally, IPAC archives scientific datasets, where raw data have been corrected for instrumental and atmospheric signatures and transformed into energy units. The datasets in the archives contain images of survey regions or images centred on individual objects, catalogues of sources extracted from
images, spectra of sources and fluxes of sources organized in time sequence; each of these datasets may be ground-based or space-based. The data may be delivered and stored in self-describing American Standard Characters for Information Interchange (ASCII) tables, or in files that conform to astronomy’s Flexible Image Transport System (FITS) standard, but source catalogues are invariably stored in a relational database.

Queries to the archive are made through a Web form in a browser, or through a program interface that is embedded in a script. These queries are made on the attributes of datasets and are intended to discover data needed for research or observation planning and, in some cases, to process them for further analysis. Examples in natural language are: ‘Return all images and spectra measured by the Spitzer Space Telescope in the Taurus Dark Cloud between wavelengths of 1 µm and 100 µm’, or ‘Discover all images measured by 2MASS in the M17 nebula and generate mosaics of them at each wavelength’. The queries return tables of archived or processed data items, along with the metadata for each item and a file handle or URL that references its location. Services are provided to perform further filtering of the results and to package and deliver the data. The worldwide Virtual Observatory projects (http://www.ivoa.net) are developing a series of program interfaces that, when complete, will support common queries to datasets across internationally distributed archives. The results will be returned in a self-describing, machine-independent eXtended Markup Language (XML) structure called VOTable and reformatted into user-friendly formats such as ASCII tables.

(b) Requirements on archive architecture

While the archive inherited the IRAS datasets, the driver for the archive architecture was 2MASS, a ground-based imaging survey of the sky in three bands that began in 1997. It delivered what were at the time the largest datasets in astronomy: atlases of images of the sky, containing over 4 million images with a volume of 10 TB, and source catalogues containing records of up to 1 billion astronomical sources. Designing an archive to serve these data on a limited budget was then at the limits of the technology. At the outset, though, it was realized that the archive content would grow far more rapidly than the archive’s budget. The technical challenge facing the archive was therefore not simply to serve the 2MASS data, but to support the addition of new archive content at low cost while preparing for the inevitable evolution of Web browsers and platforms. It was also understood that user interfaces would have to be modernized, their functionality would have to be replaced and new functionality beyond the original specifications would be needed.

Recommendations for, and evaluation of, new functionality are generally made through coordination with archive user panels, staffed by members of the astronomical community. Their recommendations have proven indispensable in providing scientific direction for the archives, and examples of their advice are given later in this paper.

(c) Architectural approach and design choices

Sustainability of the archive has always been of paramount importance. The archives use a component-based architecture to enable strong re-use
and adaptation, and avoid as far as possible special, usually vendor-specific, functionality that may become unsupported and is not portable between platforms. The architecture consists of approximately 100 core components, totaling 300K lines of code written largely in American National Standards Institute (ANSI)-compliant C, each of which usually performs one generic function such as performing coordinate transformations, composing return pages and downloading data. Third-party components are used only if supported and in wide use; generally, these components support widely used standards. Perhaps the best example of this is the use of the Apache Web server, which is very widely used, maintained by the Apache Software Foundation, and keeps pace with changing platforms and the evolution of the Hyper Text Transport Protocol (HTTP) that underpins the World Wide Web. An example of components specific to astronomy are libraries for reading files that comply with the FITS, the international standard for serving astronomical images in a machine-independent fashion.

The architecture is inherently extensible, because new applications and archive projects inherit all existing functionality. New applications plug together existing components and new components are added as needed; these may be new generic modules or custom modules for a particular dataset or service. This is how the IRSA architecture was extended to support the COSMOS, NStED, KSAS and KOA archives [1]. Functionality developed to support a new archive is then automatically available across the board. Thus, while a module for finding calibration files in KOA is specific to it, a data packaging module for KOA is now being adopted by NStED.

Figure 1 illustrates the anatomy of an application. Each component is a module with a standard interface that communicates with other components and usually fulfils one general function. Components are plugged together and controlled by an executive library, denoted ‘SVC’ in the figure, which starts components as child services and parses return values. User interfaces are thin layers atop the architecture, and they can evolve in response to user recommendations without substantial changes in the underlying components. Development uses text editors rather than third-party integrated development environments and Web page creation tools.

Figure 1. The anatomy of a user application. The schematic shows how an application is built from stand-alone modules, called by an executive library, denoted ‘SVC’ in the figure.
The archives have sought simple scalability solutions to maintain performance as their holdings grow. An on-request image mosaic service (http://hachi.ipac.caltech.edu:8080/montage/) manages the small-scale jobs this service is intended to support through a simple Java-based system developed in-house that runs each on a single processor in a 60-node cluster [2]; see also §4. Similar remarks apply to computation of periodograms to identify the periodicities present in time-series data. Because the processing of each set of frequencies is independent of all other frequencies, it is distributed across a 128-node cluster and managed by a simple in-house executive that farms out the processing to the available nodes and assembles the results.

The archive takes advantage of distributed platforms such as the Amazon Elastic Cloud (EC2) to create new data products for the science community [3,4]. To manage the complexity of the application and of the execution environment, the processing must use tools for managing workflows. Thus, the mosaic engine, Montage, and the periodogram code, by design plug-in to the Pegasus workflow manager [5].

The terabyte (TB) scale of the database required a high-end commercial DBMS. In 1999, there was anticipation that object-oriented DBMSs would become the databases of choice for serving large astronomy catalogues. A pilot study at IPAC revealed that the traditional relational model is better optimized for handling searches on tables of sources, using sky partitioning schemes as indexes for positional searches. Thus, the archives chose Informix, then the market leader in high-end relational databases, to take advantage of its internal parallelization and partitioning of data to maximize performance. At the same time, the archives avoided performance-enhancing techniques such as stored procedures, and a proprietary Datablade technology, which embedded functions in the database kernel, because they are not portable to other DBMSs. Instead, all database interfacing is isolated to one module, called isisql, which prepares user queries for submission to the database. Further, catalogues are loaded in the database as flat tables to optimize performance by avoiding the complex joins between tables: a table of a billion records is loaded as such, but partitioned internally by the DBMS itself.

(e) Design compromises and consequences

Pilot studies of technologies are always carried out to assess their technical and cost benefits, and to assess impact were they to become unsupported. So, the archive rarely becomes an early adopter of emerging technologies. While this prevented unnecessary adoption of object-oriented databases, it has also meant that we have been slow to adopt the powerful interactive capabilities offered by technologies such as Asynchronous JavaScript and XML (AJAX).

Heavy re-use of components mandates thorough regression testing whenever one component is corrected for defects or augmented with new functionality. The archives have developed a Python-based regression test bed to automate this process, but in the past a substantial part of this testing was performed semi-manually, and this has strained archive personnel resources. Moreover, re-use requires that changes in module interfaces are rigorously managed through a configuration management system.
(f) Evolution of archive functionality

Space does not permit a full description of the evolution of the archives since it opened in 1999, and therefore this section describes four examples that best illustrate the sustainability model and compromises that were made. Deployment of each example below takes advantage of the regression tests that have been compiled over the life of the project.

(i) Migration to a new database

By 2009, the annual Informix licence costs had become prohibitively expensive, and therefore the archives have begun migration to Oracle. It was selected in part because the archives are able to take advantage of a full-featured institutional site licence and technical support plan, in part because KSAS was required by the Kepler project to use Oracle and in part because pilot studies showed that Oracle provided better performance at scale than competitors such as Postgres. Moreover, the site licence allows the archive to distribute database queries across an unlimited number of servers at no cost to it. Thus, Oracle provides far more flexibility in optimizing performance as the archive grows than is the case with Informix, where the licence costs limited the archive to three database servers, even as the database volume expanded.

The archive design means that migration, scheduled for completion in spring 2011, involves only two main thrusts: a major upgrade of isisql to support Oracle, and migration of the database contents to Oracle. When developed in 1999, isisql made database calls by embedding SQL statements in-line, an approach that meant isisql contained Informix-specific calls. The upgrade is replacing embedded calls with Open Database Connectivity (ODBC), which provides a translation layer to replace the Informix-specific calls in isisql with standard calls that are DBMS neutral.

The feasibility of using ODBC was demonstrated in a pilot study aimed at understanding the technical problems involved in the migration and in informing the migration schedule. It showed that Oracle’s proprietary SQL*Loader application was much slower than loading via a direct, parallelized insert-as-select mechanism at the multi-TB scale that the database must support. Oracle has a larger set of reserved keywords than Informix, and it supports fewer data types than Informix, and especially the many time formats used in astronomy. These formats will be written as character strings to preserve the fidelity of times delivered as datasets, at the expense of disallowing mathematical operations on them in the DBMS. The archives are currently implementing these changes.

(ii) Migration of the Web servers to LINUX platforms

The archive aims to replace hardware every 3–4 years to take advantage of improved machine performance and reduce the risk of failure in ageing machines. Fast, cheap LINUX servers were chosen to replace ageing Sun servers. The software running on Sun servers was written in ANSI C, avoided the use of shared memory and was compiled with the gcc compiler, but the migration nevertheless required makefile and code changes. LINUX, for example, requires that specific libraries and
macros are linked at compile time, returns different information from system calls than does SOLARIS, and is much less forgiving of uninitialized variables and weak casting of variables.

The archives recognized the need for greater attention to these details during development. Code reviews now explicitly examine the issues of typecasting and initialization, and developers routinely build code on several platforms to ensure portability. The availability of virtual machines has made this latter task much easier than in the past.

(iii) Indexing of image datasets

Image datasets are becoming increasingly larger and complex—the Spitzer Space Telescope alone will deliver over 100 million images—and queries on these datasets place increasing load on database servers and produce ever more complex result sets. Astronomers on the user panel were unanimous that acceleration of searches on large image datasets were more important than interactive technologies to explore complex result sets. Consequently, the archive developed an R-tree indexing system that stored the indices in memory-mapped files, deployed across a cluster of machines for parallel access by Representational State Transfer (REST) based Web services. Its performance scales as \( \log(N) \), and has seek times of 2s for the Spitzer Space Telescope images, a performance gain on up to \( \times 1000 \) over a full scan of the data. The scheme was developed as an IPAC contribution to the National Virtual Observatory project.

(iv) Visualization tools

When the archive opened in 1999, it used two inherited command-driven programs, SKYVIEW and AGRA, to create Joint Photographic Experts Group (JPEG) images for display of datasets in a browser and overlay information such as image footprints on them. Their limited support for image projections and for colour rendering led to their replacement by a Java-based plotter driven by an XML definition of the plot. While this supported the growing complexity of plots, it suffered from poor memory management. A third generation visualizer now under development completely decouples the client and server processing. On the server side, it will use the same XML definition of the plot, but will use the full FITS library to support all projections used in astronomy, and will replace the Java-based code with a C-based JPEG image creation module developed for MONTAGE (§4) to create the JPEG image. On the client side, it will use AJAX to render the image in the browser. This complete decoupling of the graphical user interface and image generation allows both to evolve independently in future.

3. A case study in software modernization: the scan processing interactive (Scanpi) tool

(a) What is Scanpi and why modernize it?

Scanpi is an interactive tool, developed in 1983 and written in FORTRAN 66, for averaging IRAS calibrated survey scans. A unique tool for studying extended, confused or faint sources, it offers sensitivity gains of up to a factor of five
over the IRAS mission science products. Between 2002 and 2007, Scanpi received 25,000 requests annually and 5–15 papers per year cited it. Its complex layering of programs added over the years had, after a quarter of a century, rendered the ageing code difficult to maintain and keep apace with platform changes. Moreover, Scanpi was tightly integrated with an internal, compressed copy of the IRAS scan data. In 2007, the data decompression algorithm, unsupported for over 15 years, failed during migration to SOLARIS 2.9, and Scanpi became orphaned on SOLARIS 2.8. Finally, the Scanpi developer decided to retire.

The IRSA user panel and members of the Spitzer and Herschel Science Centers recommended modernization of Scanpi over decommissioning it, because it would prove invaluable in planning and interpretation of deep observations with the Spitzer Space Telescope and Herschel Space Observatory.

**(b) Requirements for modernizing Scanpi**

Consultations with the user panel, archive users and members of the Spitzer and Herschel communities revealed that the scan processing algorithms were adequate, but recommended visibility into the results processing steps to guide quality assurance of the results, and to improve efficiency by supporting the input of lists of sources.

**(c) Design and implementation**

Scanpi was redesigned and rewritten from the ground up as a workflow application integrated into the archive architecture [6]. None of the original code was preserved. Figure 2 shows the design. Each module in the top panel of the figure is specific to Scanpi and represents one step in processing the scans. Through a simple Web form, users select processing parameters, choose whether to visualize and download the output of each workflow step and enter their sources, individually or as a list.

Each component in the new design made substantial re-use of components already developed to support the archives, and these are named in the oval panel at the bottom of figure 2. We describe two of them, and the reasons for using them, in the rest of this paragraph. Scanpi supports the uploading of a list of sources, so it calls a table processing library that uploads and parses a source list and interprets all common astronomy coordinate systems and data formats. Implementing the new requirements substantially increases the processing time, and therefore Scanpi calls a background processing library that prevents browser time-outs and passes process control to the server. Finally, the scan archive was decompressed into ASCII files.

The use of a workflow design and integration into the archive allowed Scanpi to meet its requirements simply and efficiently. The design reduced the code base to 21 K lines of code from 102K lines of code, and the service was implemented successfully in C (for performance) on a dedicated Red Hat LINUX server with 1.25 FTE of labour; originally, Scanpi had required 1.25 FTE over five years purely for maintenance. Since modernization, Scanpi has largely been maintenance free: it has required less than 0.1 FTE maintenance over the three years since deployment.
Figure 2. The design of Scanpi as a workflow application. The top panel in the figure shows functionality specific to Scanpi that was rewritten from the ground up. The arrows denote the flow of data between components. The oval at the bottom of the figure lists some of the existing, re-usable archive components that were called by the new functionality to reduce code size and promote efficient development. The double arrow connecting the box and the oval denotes the calls made by new functionality to existing components.

(d) Deployment and usage

Scanpi (http://scanpiops.ipac.caltech.edu:9000/applications/Scanpi/index.html) has performed flawlessly since it was deployed in operations in early 2008, and retains its value to the community as an observation planning and analysis tool. Serjeant & Hatziminaoglou [7] used it to estimate whether the Herschel Space Observatory will be able to measure the host galaxies of quasars, and Matthews et al. [8] used it as part of a Herschel Key Programme to measure the energy distributions of debris discs in many types of stars.

4. A case study in building a user community: the MONTAGE image mosaic engine

(a) Drivers for developing MONTAGE

MONTAGE was built in response to a community need, articulated through user panel recommendations and through conversations with astronomers, to build mosaics of sets of input images to study regions larger than the fields-of-view of modern cameras, and to detect very faint sources by combining sets of images. These use cases implicitly assume that the mosaics are science-grade and preserve the calibration and positional fidelity of the input images. Yet the input images are invariably made on a variety of instruments and telescopes whose pixels all
Figure 3. The processing flow in the MONTAGE mosaic engine, illustrated for a mosaic made of three images.

sample the sky differently. MONTAGE, therefore, aggregates astronomical images format into mosaics having user-specified parameters of image projection, pixel size, coordinates and image size [9].

(b) The design of MONTAGE

MONTAGE was designed as a toolkit that contains the components for creating mosaics. Figure 3 shows how these components operate as a workflow application to produce a mosaic, as follows:

— analyse the geometry of a set of images to identify those images that will be used in the mosaic,
— reproject the input images on the sky to the required output projection,
— rectify the background radiation in the images to a common level by minimizing the differences in brightness levels between the images, and
— co-add the reprojected, rectified images to form the mosaic.

Reprojection computes which fraction of an input pixel’s energy should be redistributed to an output pixel by computing their areas of overlap on the sky. This is computed exactly with classical spherical trigonometry techniques, and guarantees that the mosaics are science grade, but at the expense of performance. A much faster plane-to-plane reprojection algorithm is provided for the common case of small-scale images projected in the tangent plane of the celestial sphere.

The toolkit is written in ANSI-compliant C for performance and portability. It is available for download for non-commercial users, and runs on desktop, cluster or supercomputer environments running common UNIX-based operating systems such as LINUX, SOLARIS, Mac OS X and AIX. A Web-based help desk is available to support users, and full documentation is available on-line, including the specification of the applications programming interface (API).

(c) Applications of MONTAGE and growth of a user community

The MONTAGE modules were designed from the outset to integrate into the archive architecture, and are routinely used to support archive applications. One simple example, described in §2, is that the module for creating a JPEG image

1By analogy with terrestrial cartography, a projection is a mathematical relationship between the pixel coordinates in the image and the pixels on the three-dimensional surface of the sky. As in cartography, there are many projections in use in astronomy.
is being re-used as part of the next-generation image visualization tools. But beyond that, MONTAGE has been downloaded over 4000 times by astronomers and computer scientists. It has found a wide applicability in the fields of scientific research, product generation and developing data-driven compute infrastructure. This section analyses how this user community has developed.

MONTAGE is easy to build on desktops with a simple make command. All components and libraries, including utilities for managing mosaics and third-party libraries, are bundled in the distribution, along with a simple build test to determine that the software has built correctly. Astronomers have made extensive use of it in their research. Recently, Stock & Barlow [10] used it to search for ejecta from WR-stars by creating mosaics of images from the AAO/UKST Southern Hα Survey (SHS).

Because all the components are run from the command line, astronomers have developed custom scripts to build mosaics to particular specifications, and they are beginning to share them with the community through the MONTAGE project Web pages. These scripts are intended by the authors to be run on users’ desktops, but there is no reason why they could not be used to underpin Web-based interfaces and data access portals. Dr Inseok Song shared his bash script for computing three-colour mosaics of Digitized Sky Survey (DSS2) images (http://montage.ipac.caltech.edu/docs/pleiades_tutorial.html). Dr Thomas Robitaille developed a Python API to MONTAGE, which supports calls to individual MONTAGE modules and calls to functions that create mosaics. The API takes care of setting up the directories and cleaning up afterwards, so users only need to specify a directory containing the input files to mosaic and the output filename. Dr Colin Aspin has contributed his scripts for computing three-colour mosaics from the 2MASS, Sloan Digital Sky Survey (SDSS) and DSS image datasets. These scripts often take advantage of modern program-friendly interfaces to access data needed for input to the mosaic engine.

The toolkit design offers flexibility to end-users, who have incorporated modules into processing environments to create science products. Recently, Aguirre et al. [11] used MONTAGE to reproject images to create an atlas of images for the Bolocam Galactic Plane Survey (BGPS), a millimetre-wave continuum survey of the Galactic plane. Anderson et al. [12] used it to resample 350 μm maps of data from the Herschel Space Telescope in their study of the physical properties of the dust of the HII region RCW 120. The Arecibo Legacy Fast ALFA survey (ALFALFA) [13] has used MONTAGE to create a large wide-field mosaic of neutral hydrogen emission at 21 cm.

An on-request mosaic service [2] (http://hachi.ipac.caltech.edu:8080/montage/) returns single-colour mosaics from three major image datasets: 2MASS (http://www.ipac.caltech.edu/2mass/), DSS (http://archive.stsci.edu/dss/) and SDSS (http://www.sdss.org). At the request of end-users, future plans for the service include extension of the service to return three-colour mosaics, offer access to more image collections and support interactivity of the image through the use of the AJAX technology.

The MONTAGE workflow in figure 3 is naturally data parallel, apart from the computation of the sky background model, which requires as input the differences in background emission between all the images. The distribution includes modules for installation of MONTAGE on computational grids and so it can be run as a parallel application through workflow management tools such as Pegasus.
and through the message-passing interface (MPI). MONTAGE has consequently appealed to the computer science community who have used it at scale as an exemplar application on grid, cluster and cloud platforms in developing the algorithms and infrastructure of the next generation of data-aware computing and cyber-infrastructure, including:

- task scheduling in distributed environments (performance focused) [14,15],
- designing job schedulers for the grid [16,17],
- designing fault tolerance techniques for job schedulers [18],
- exploring issues of data provenance in scientific workflows [19,20],
- exploring the cost and performance of scientific applications running on clouds [4,21,22],
- developing high-performance workflow restructuring techniques [23],
- developing application performance frameworks [24], and
- developing workflow orchestration techniques [25].

(d) MONTAGE and Web 2.0

A recent survey [26] showed that scientists in general have not yet adopted the so-called Web 2.0 as a tool for collaboration and sustaining communities. To date, MONTAGE has not made extensive use of Web 2.0 to sustain its user community, primarily because the development was completed before the ubiquity of interactive media. The potential of Web 2.0 became clear in a recent discussion thread on mosaics on the astrobetter.org forum, dedicated to information sharing among astronomers, which led to a reorganization of the MONTAGE website to make information on tutorials and scripts visible on the front page. MONTAGE plans to release a public blog or forum for users to discuss issues and solutions; it will be more valuable in this regard than the user help desk, which is seen by the community as a place to report bugs.

Katz et al. [27] have described how Web 2.0 can encourage MONTAGE users to collaborate and share information. One approach would be collaborative building of wide-area atlases of image mosaics. Requests for images not yet in the atlas would trigger a call to MONTAGE to generate the page. Procter et al. [26] then took this idea one step further by identifying a use case where members of a consortium performing multi-wavelength investigations of, for example, star-formation regions or clusters of galaxies can generate multi-wavelength atlases of the survey region as a basis for verifying newly discovered sources and analysing their spectral energy distributions.

5. Best practices for software sustainability

In conclusion, our experience over the past decade leads us to recommend the following as best practices for software sustainability:

- design for sustainability, extensibility, re-use and portability from the outset,
- use modular- or component-based designs,
- be careful about new technologies; do a cost–benefit analysis before adopting them, and understand the risk if they become unsupported,
— build a user community that encourages users to contribute to sustainability; take advantage of Web 2.0 in this endeavour,
— listen to your users, and have a formal user-advisory group,
— use rigorous software engineering practices to ensure well-organized and well-documented code; and control and manage interfaces, and
— develop when possible in an open software, open data mode, where source code, a set of input data and tests are freely available.

This paper showed how these techniques have sustained the data archives developed at IPAC. No one can fully predict new advances in software and hardware platforms but we hope that our approach will provide sustainability in the future.

G.B.B. is supported by the NASA Exoplanet Science Institute at the Infrared Processing and Analysis Center, operated by the California Institute of Technology in coordination with the Jet Propulsion Laboratory (JPL). MONTAGE was funded by the National Aeronautics and Space Administration’s Earth Science Technology Office, Computation Technologies Project, under Cooperative Agreement Number NCC5-626 between NASA and the California Institute of Technology. MONTAGE is maintained by the NASA/IPAC Infrared Science Archive. The modernization of Scanpi was partially funded by the Spitzer Science Center. E.D. is supported by the National Science Foundation under grant no. OCI-0722019.

References


