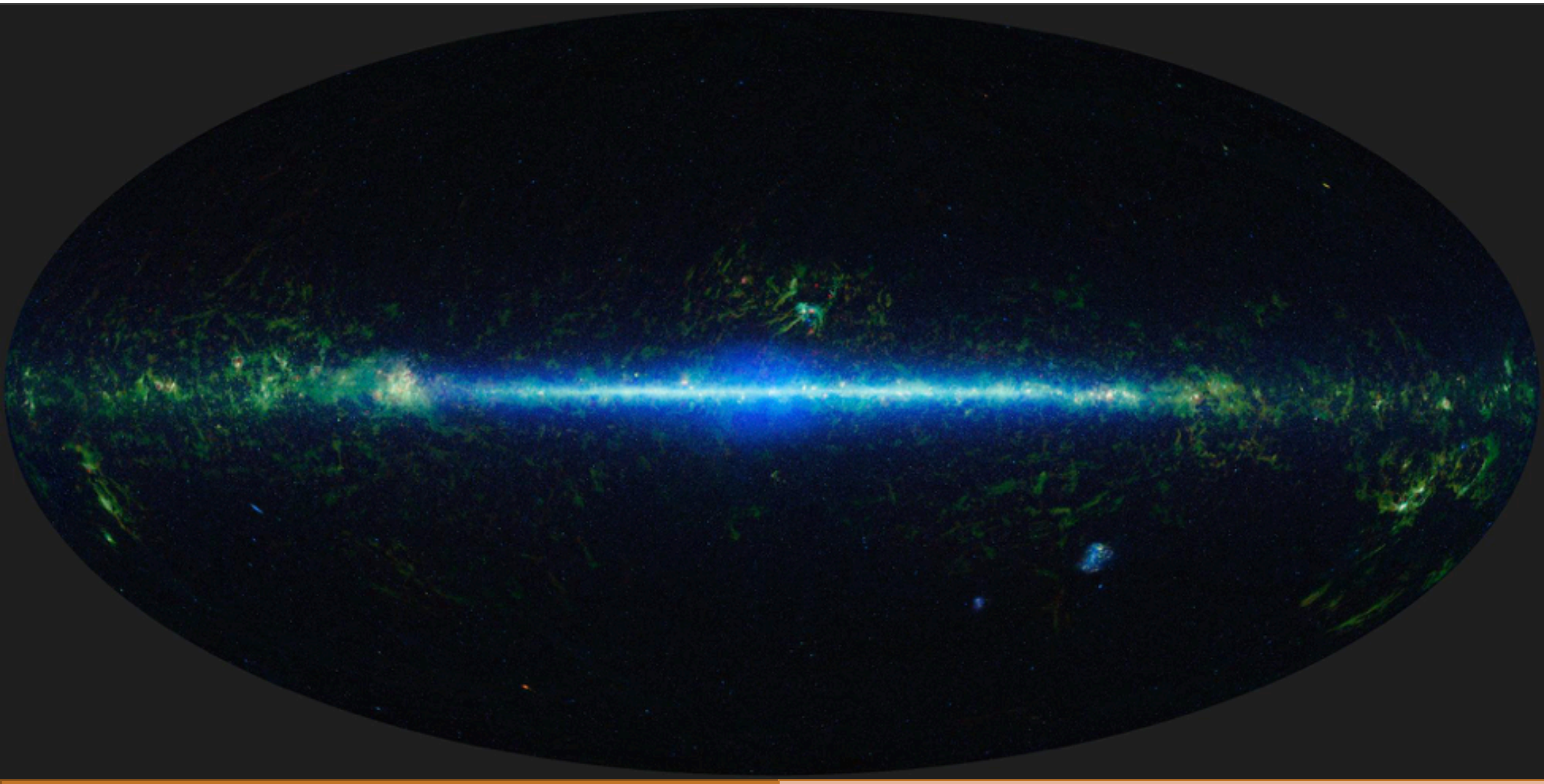


Open Code at IPAC

David A. Imel, Manager
Caltech/IPAC

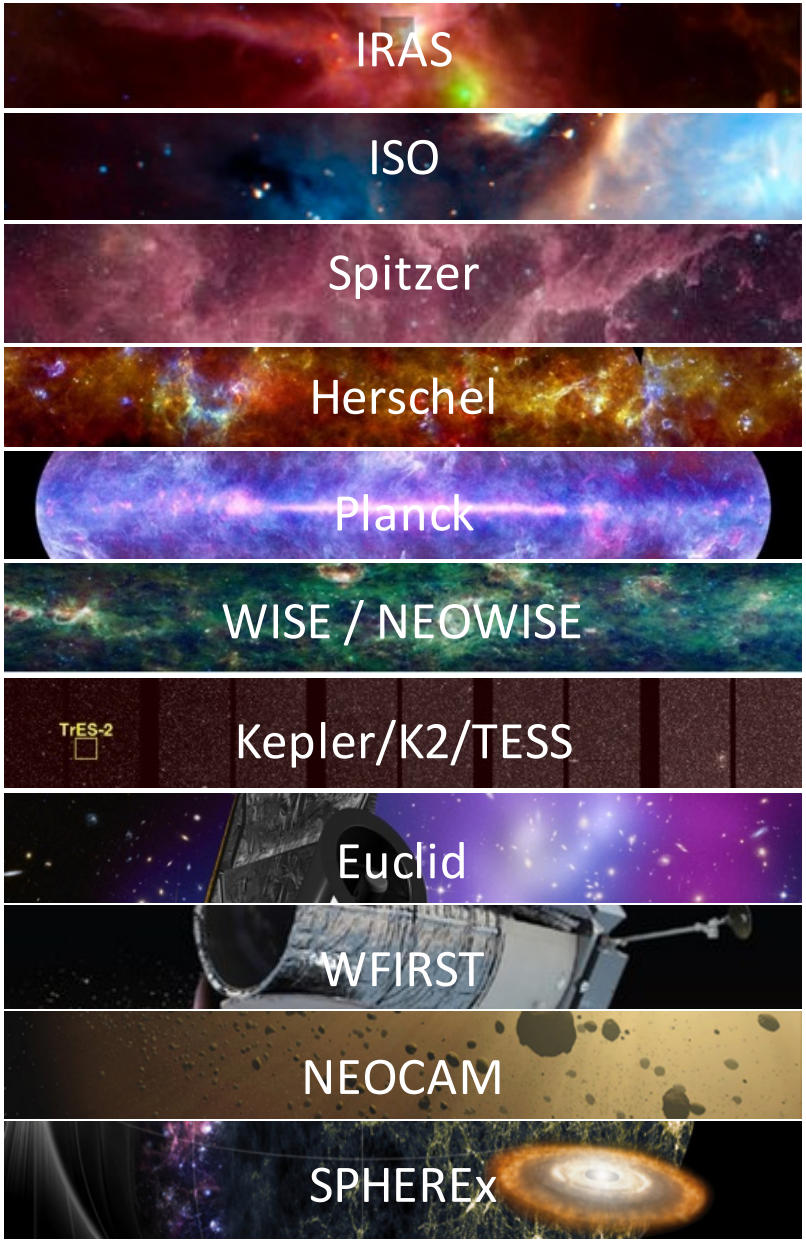
15 November 2017

1. Introduction to IPAC
2. IPAC Code
3. IPAC Code Release Process
4. Considerations for Open Code

IPAC: Caltech Astrophysics Science Center

- Science Center functions for NASA missions since 1985
- Data centers and archives for major projects such as Great Observatories and all-sky surveys, including ground-based telescopes.
- Supports NASA, NSF and privately funded projects
- Award-winning media, outreach and education support
- Vibrant research environment and staff

Cosmology and galaxy evolution	***Asteroids and the solar system***
Exoplanets	***Infrared-submillimeter astrophysics.***



IPAC Astronomy 2017

- Earth-Size planets: Spitzer finds seven Earth-size planets
- Gravitational Waves: IPAC astronomers, archives, and communications teams participate in synchronous EM observations.
- IPAC scientist shows how to point JWST to potential exoplanet systems.


"All the News That's Fit to Print"

The New York Times

National Edition

VOL. CLXVI ... No. 57,517 THURSDAY, FEBRUARY 23, 2017 $2.50

A rendering of newly discovered Earth-size planets orbiting a dwarf star named Trappist-1 about 40 light-years from Earth. Some of them could have surface water.

TRUMP RESCINDS OBAMA DIRECTIVE ON BATHROOM USE

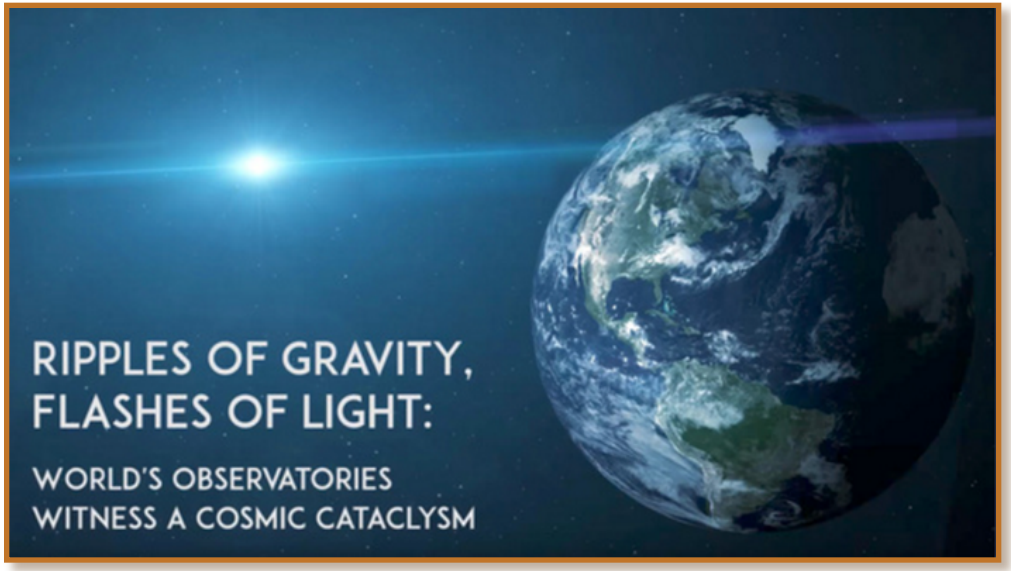
ENTERING CULTURE WARS

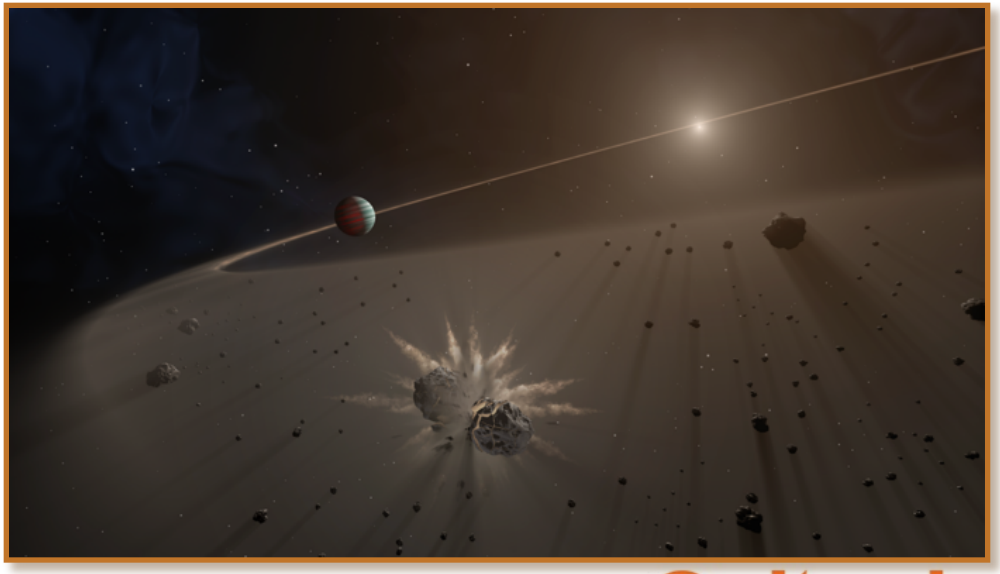
Question of Transgender Rights Splits DeVos and Sessions

Circling a Star Not Far Away, 7 Shots at Life

Uber's Culture Of Gutsiness Under Review

Migrants Hide, Fearing Capture on 'Any Corner'





Code Categories at IPAC

"Code" at IPAC covers a broad range of content.

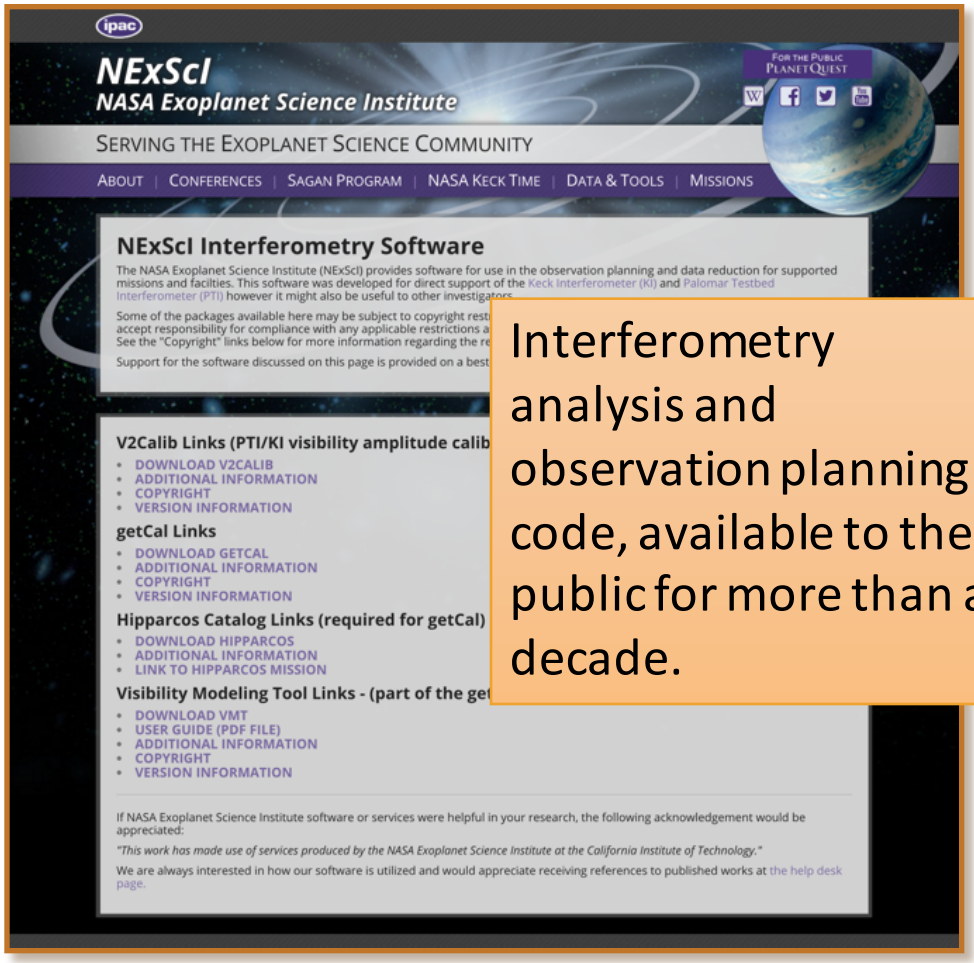
- Data processing pipeline software for NASA Missions and Ground-Observatories
  - Data transfer and ingest
  - Calibration
  - Data product generation
- Observation planning and scheduling
- Archive software
  - Database ingest, search, and access
  - Fileservice
  - Web User Interface
  - Programmatic interfaces
  - Analysis software
- Research analysis software
- Websites (including javascript, CSS); e.g. proposal submission and management, helpdesk
- Internal datacenter systems; e.g. ansible playbooks
- Configuration files for all of the above

Ansible playbook for configuring a server

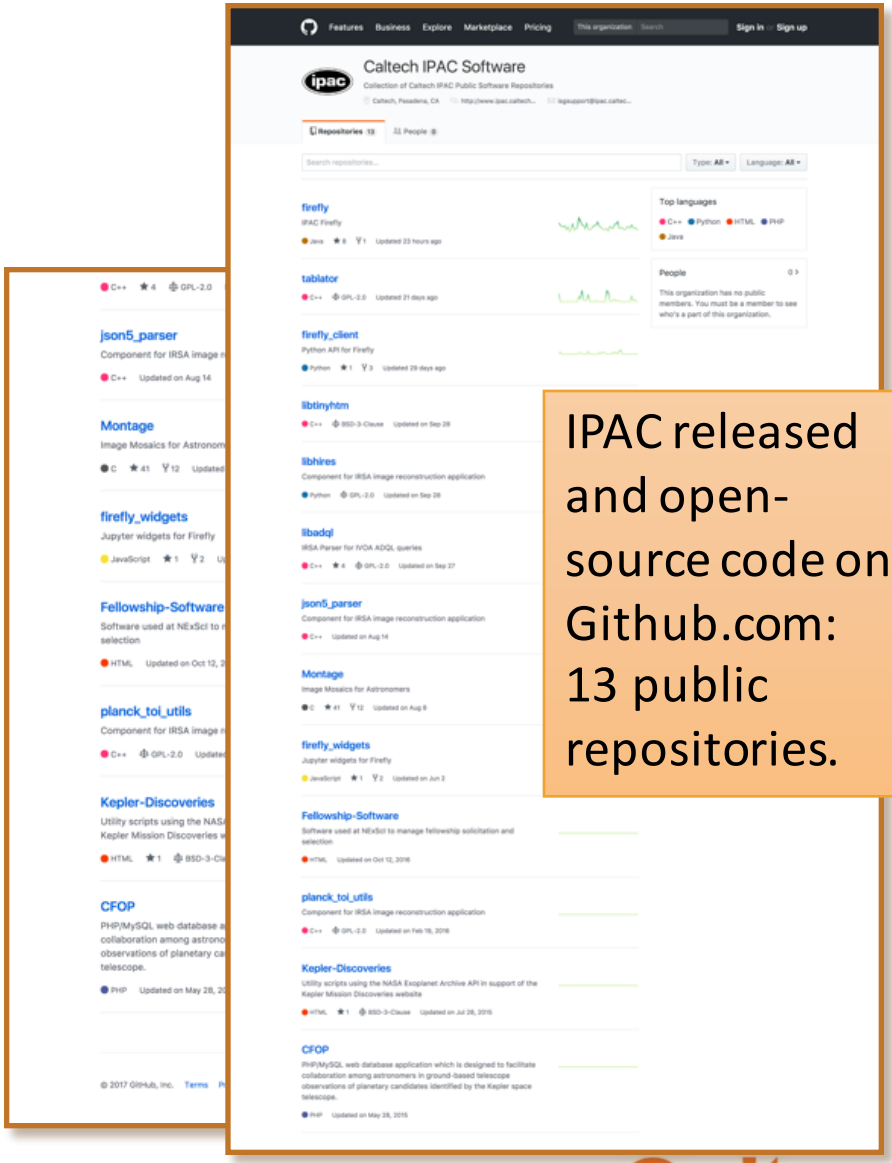
```
---

- hosts: bte
  remote_user: centos
  become_method:  sudo
  become:  true
  become_user:  root
  tasks:

  - name: "Get Jenkins repo for Yum"
    get_url:
      dest=/etc/yum.repos.d/jenkins.repo
      url=http://pkg.jenkins.io/redhat/jenkins.repo

#
# Had to disable gpg key check, because the key wasn't coming with the repo,
# and this commnad was failing all the time.  Would rather get the key with
# the repo, but use this for now
#

  - name: "install jenkins"
    yum: name=jenkins state=present disable_gpg_check=yes

  - name: "start jenkins service"
    service: name=jenkins state=started

# Tried firewalld for this, but requires separate install, and google says unstable
# These commands aren't "idempotent", of course, so this could be wasting time...

  - name: "start firewall service"
    service:  name=firewalld state=started

  - command: /usr/bin/firewall-cmd --add-port=8080/tcp --permanent --zone=public
  - command: /usr/bin/firewall-cmd --add-service=http --permanent --zone=public
  - command: /usr/bin/firewall-cmd --reload

  - command:  sudo cat /var/lib/jenkins/secrets/initialAdminPassword
    register: out
    ignore_errors: yes

  - debug: var=out.stdout_lines
```

Code Release and Open Source

While IPAC has always had released software,
IPAC is now also doing open-source development.



Interferometry analysis and observation planning code, available to the public for more than a decade.



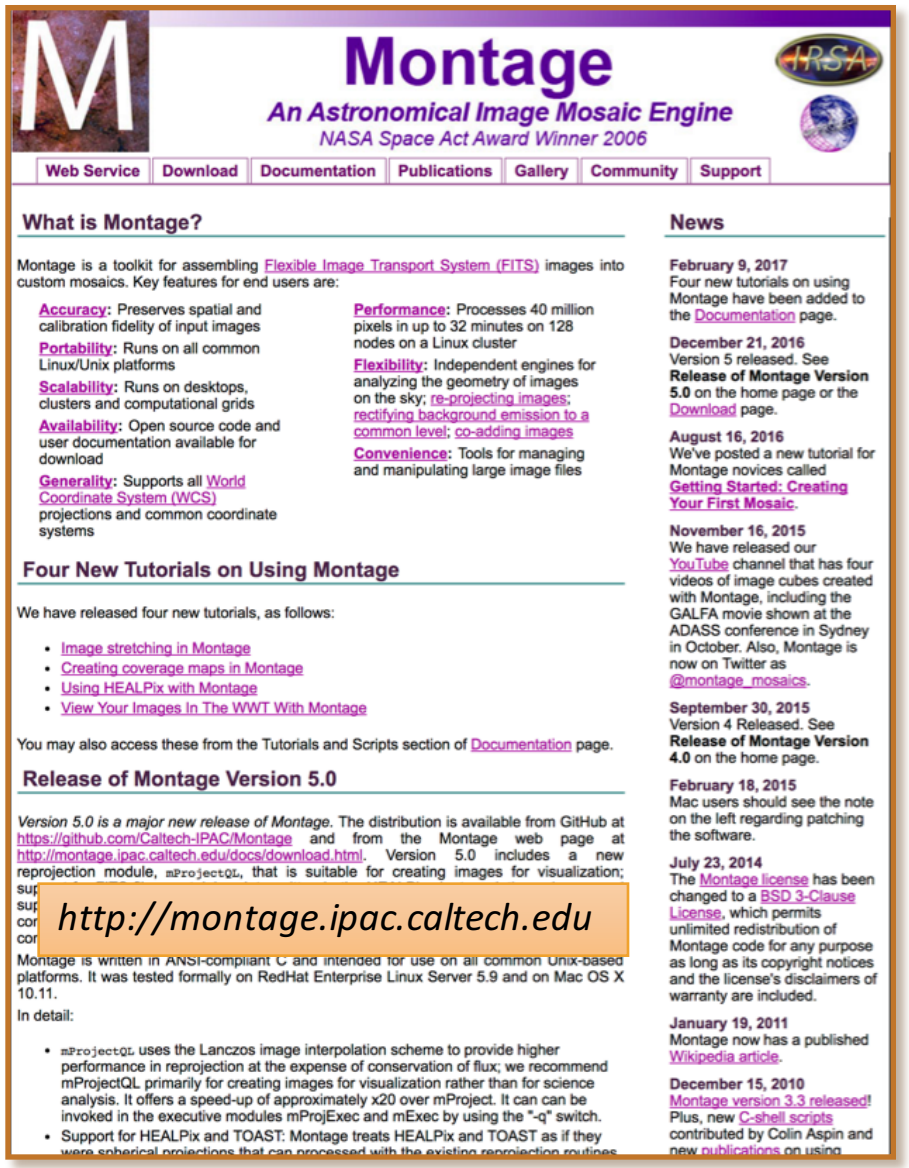
IPAC released and open-source code on Github.com: 13 public repositories.

Spitzer, Herschel observation planning code cleared for public release and shared with collaborators.

Caltech

IPAC Open Source: Montage

Montage code has been available since 2005



Montage

An Astronomical Image Mosaic Engine

NASA Space Act Award Winner 2006

Web Service | Download | Documentation | Publications | Gallery | Community | Support

What is Montage?

Montage is a toolkit for assembling Flexible Image Transport System (FITS) images into custom mosaics. Key features for end users are:

- **Accuracy**: Preserves spatial and calibration fidelity of input images
- **Portability**: Runs on all common Linux/Unix platforms
- **Scalability**: Runs on desktops, clusters and computational grids
- **Availability**: Open source code and user documentation available for download
- **Generality**: Supports all World Coordinate System (WCS) projections and common coordinate systems
- **Performance**: Processes 40 million pixels in up to 32 minutes on 128 nodes on a Linux cluster
- **Flexibility**: Independent engines for analyzing the geometry of images on the sky; re-projecting images; rectifying background emission to a common level; co-adding images
- **Convenience**: Tools for managing and manipulating large image files

Four New Tutorials on Using Montage

We have released four new tutorials, as follows:

- Image stretching in Montage
- Creating coverage maps in Montage
- Using HEALPix with Montage
- View Your Images In The WWT With Montage

You may also access these from the Tutorials and Scripts section of Documentation page.

Release of Montage Version 5.0

Version 5.0 is a major new release of Montage. The distribution is available from GitHub at https://github.com/Caltech-IPAC/Montage and from the Montage web page at http://montage.ipac.caltech.edu/docs/download.html. Version 5.0 includes a new reprojection module, `mProjectQL`, that is suitable for creating images for visualization;

http://montage.ipac.caltech.edu

Montage is written in ANSI-compliant C and intended for use on all common Unix-based platforms. It was tested formally on RedHat Enterprise Linux Server 5.9 and on Mac OS X 10.11.

In detail:

- `mProjectQL` uses the Lanczos image interpolation scheme to provide higher performance in reprojection at the expense of conservation of flux; we recommend mProjectQL primarily for creating images for visualization rather than for science analysis. It offers a speed-up of approximately x20 over mProject. It can can be invoked in the executive modules mProjExec and mExec by using the "-q" switch.
- Support for HEALPix and TOAST: Montage treats HEALPix and TOAST as if they were spherical projections that can processed with the existing reprojection routines

News

February 9, 2017
Four new tutorials on using Montage have been added to the Documentation page.

December 21, 2016
Version 5 released. See **Release of Montage Version 5.0** on the home page or the Download page.

August 16, 2016
We've posted a new tutorial for Montage novices called **Getting Started: Creating Your First Mosaic**.

November 16, 2015
We have released our YouTube channel that has four videos of image cubes created with Montage, including the GALFA movie shown at the ADASS conference in Sydney in October. Also, Montage is now on Twitter as @montage_mosaics.

September 30, 2015
Version 4 Released. See **Release of Montage Version 4.0** on the home page.

February 18, 2015
Mac users should see the note on the left regarding patching the software.

July 23, 2014
The Montage license has been changed to a BSD 3-Clause License, which permits unlimited redistribution of Montage code for any purpose as long as its copyright notices and the license's disclaimers of warranty are included.

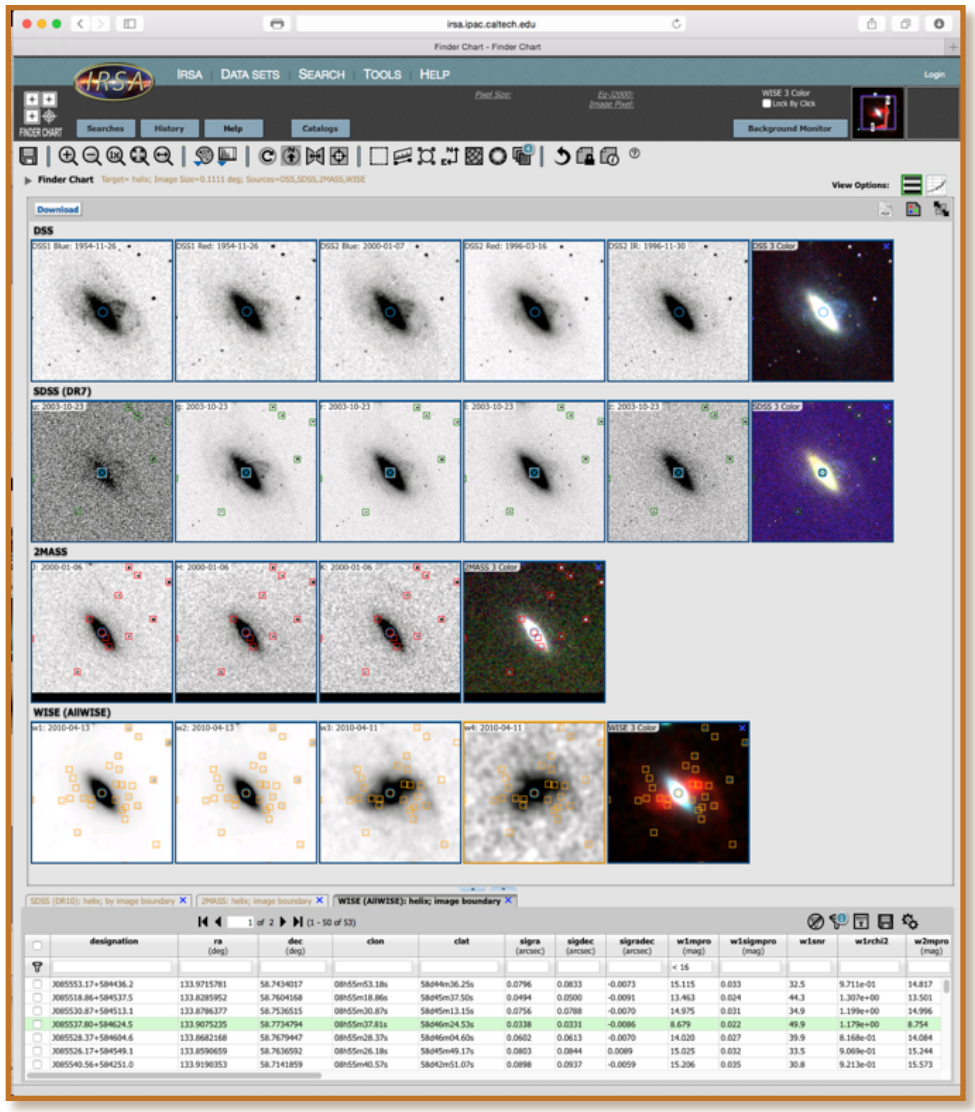
January 19, 2011
Montage now has a published Wikipedia article.

December 15, 2010
Montage version 3.3 released! Plus, new C-shell scripts contributed by Colin Aspin and new publications on using

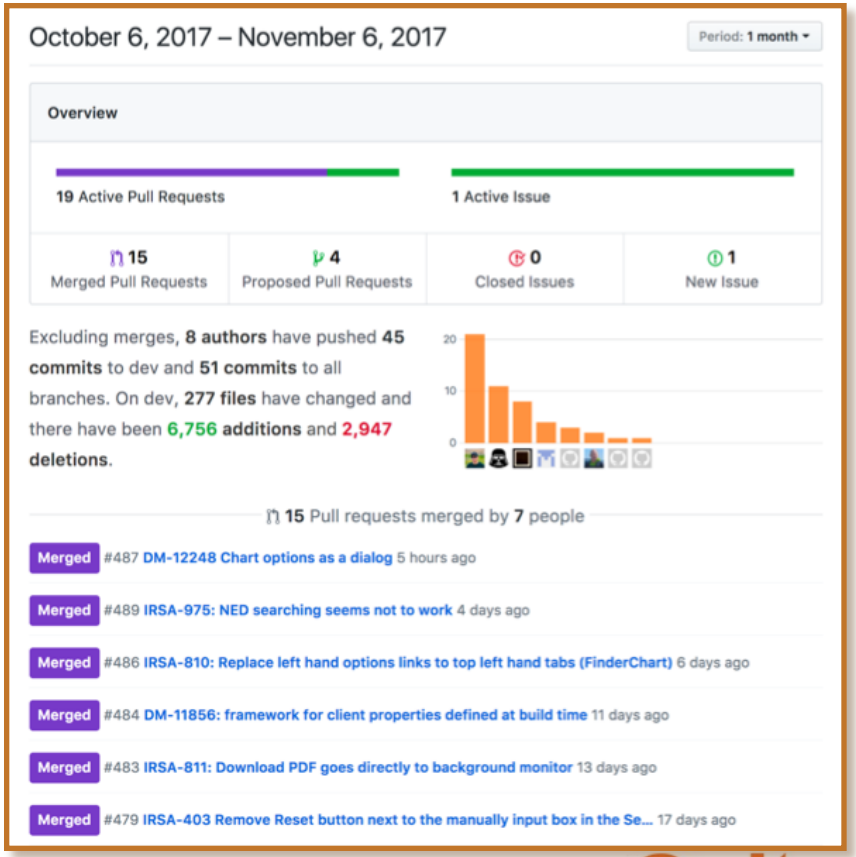
This image mosaic centered on the dark cloud Barnard 92 preserves the positional accuracy and intensities of a total of 92 input images, and is an example of the types of images astronomers use in their research. Image credit: ***Montage****.*

IPAC Open Source: Firefly

IPAC's primary User Interface Code is now open source software.



Firefly is used by both the Infrared Science Archive (IRSA) to display and analyze 2MASS, WISE, Spitzer, Planck, Herschel, and many other datasets, as well as by the Large Synoptic Survey Telescope (LSST) Science Platform. Firefly is about ~200K lines of code.



Caltech

IPAC Process for Code Release

All Software created at Caltech is Caltech Intellectual Property

- IPAC is committed to sharing code under the right conditions.
- IPAC must follow Caltech policy on Intellectual Property
- Process when request is received (internal or external) for releasing code:
  1. Evaluate support required and resources available to support the release: both initial and ongoing.
  2. Obtain approval from Caltech Office of Technology Transfer
  3. **Conduct Code Release Review**
- Publish code (typically via Github.com) after successful review.

Code Release Review Criterion: Security

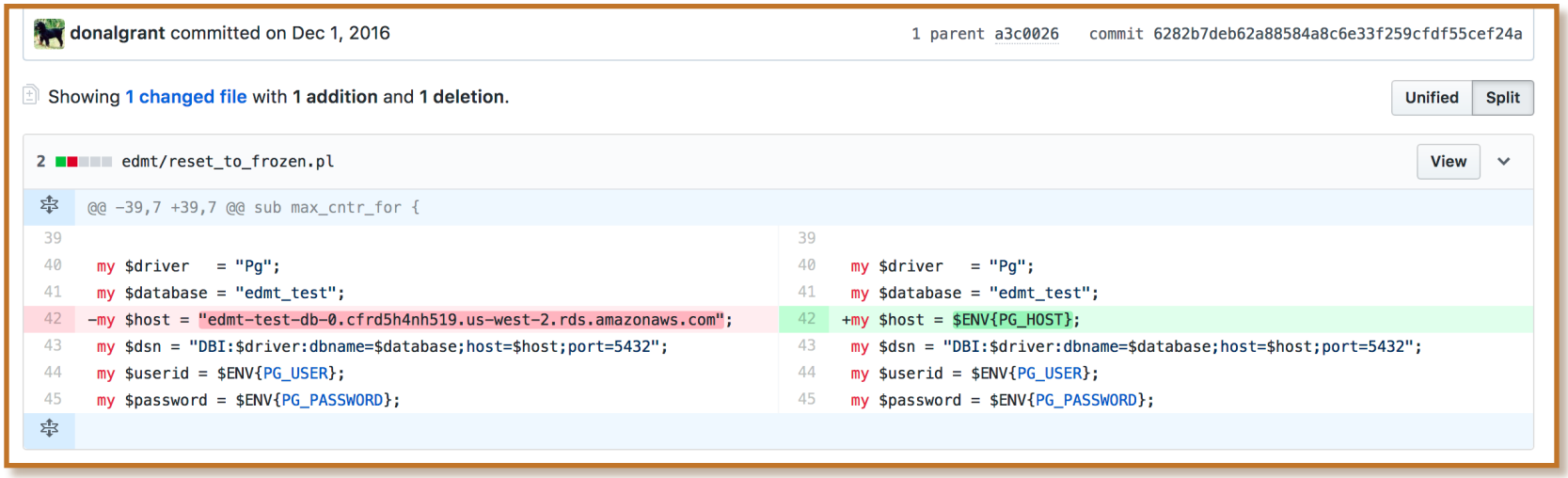
Beware git revision history and comments!

- Does the code expose internal network or server configurations?

  Example from pre-released software:

```
// example: http://bacchus.ipac.caltech.edu:5000/wise/pass1/i3om_cdd/{coadd_id}/  {band}/{product}
```

  - comment with instructions reveals internal server and filesystems
- Does the code reveal interfaces in a way that makes the system easier to hack?
- Are all inputs taint-checked?
- Passwords or other sensitive info in code or revision history?



If a password or some other sensitive piece of information was once part of the code, it will be there forever in the git revision history, unless careful measures are taken to excise all references to those revisions. Just scanning the current code will be insufficient to detect this issue.

Caltech

Code Release Review Criterion: Legal

Beware of mixed license types in embedded software packages

1. Are the Caltech Office of Technology Transfer requirements met? (Approval, license files / copyright statements in each source file.)
2. Are the constraints of any licenses of embedded modules met?

Software licenses and rights granted in context of the copyright according to Mark Webbink.[1] Expanded by freeware and sublicensing.

Rights granted	Public domain	Non-protective FOSS license (e.g. BSD license)	Protective FOSS license (e.g. GPL)	Freeware/Shareware/ Freemium	Proprietary license	Trade secret
Copyright retained	No	Yes	Yes	Yes	Yes	Yes
Right to perform	Yes	Yes	Yes	Yes	Yes	No
Right to display	Yes	Yes	Yes	Yes	Yes	No
Right to copy	Yes	Yes	Yes	Often	No	No
Right to modify	Yes	Yes	Yes	No	No	No
Right to distribute	Yes	Yes, under same license	Yes, under same license	Often	No	No
Right to sublicense	Yes	Yes	No	No	No	No
Example software	SQLite, ImageJ	Apache web server, ToyBox	Linux kernel, GIMP	Irfanview, Winamp	Windows, Half-Life 2	Server-side World of Warcraft

Table illustrating different software license types, from Wikipedia

Open Source Licenses

- IPAC typically releases software under the BSD-3 license:
  - Allows redistribution
  - Retains copyright
  - Limits liability
- Another popular license is the GPL "Copyleft" license:
  - More restrictive than BSD-3
  - Requires that all derivative work retain the same license as the original.
  - Guarantees perpetual open access to derivative work.
- Public domain:
  - May not be an adequate approach for private institutions (liability) and sponsored organizations (citations).

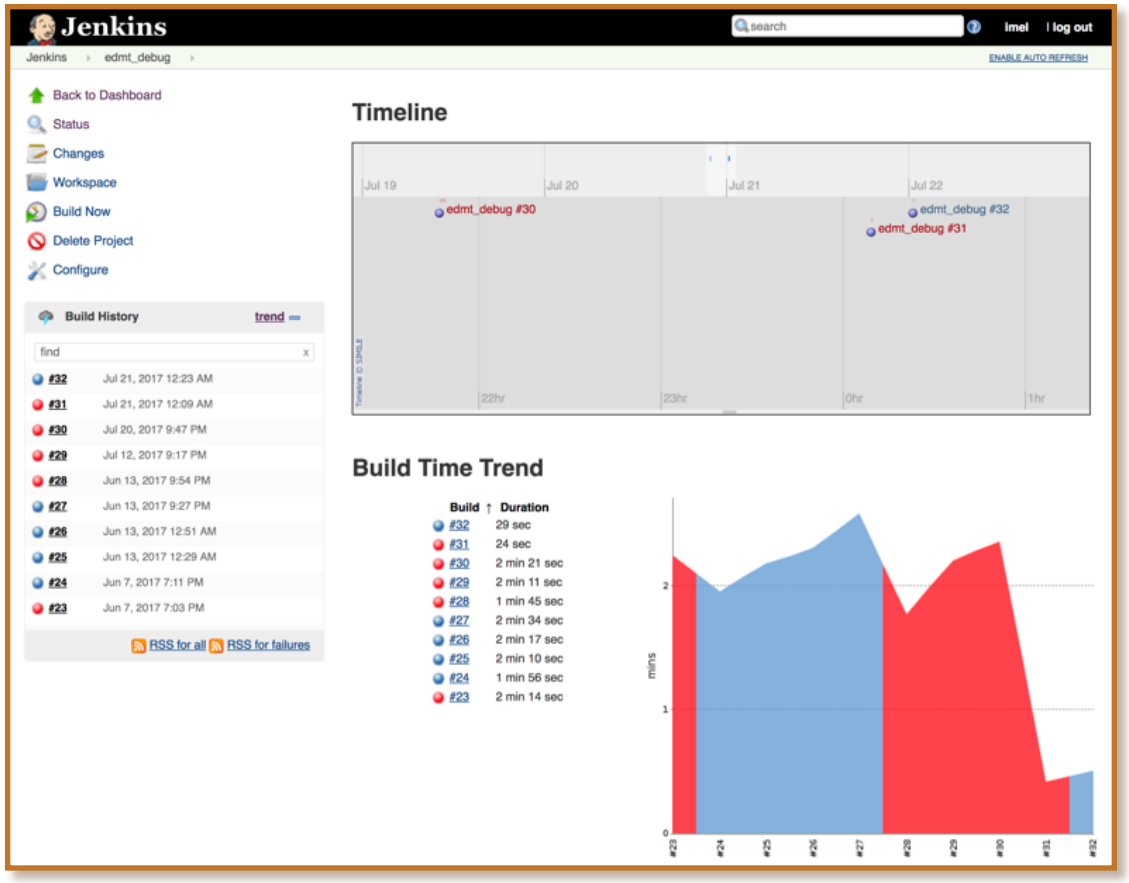
```
Copyright © <2014>, California Institute of Technology, developed for the
Spitzer Science Center and the NASA/IPAC Infrared Science Archive (IRSA) with
support from NASA, and for Large Synoptic Survey Telescope (LSST) with
support from NSF.

All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

  * Redistributions of source code must retain the above copyright notice,
    this list of conditions and the following disclaimer.
  * Redistributions in binary form must reproduce the above copyright notice,
    this list of conditions and the following disclaimer in the documentation
    and/or other materials provided with the distribution.
  * Neither the name of the California Institute of Technology (Caltech) nor
    the names of its contributors may be used to endorse or promote products
    derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

BSD-3 License

Code Release Review Criterion: Functionality and Builds

Does it work out of the box?

1. If the system is not already in operations, what testing has been done to validate that the system works as advertised?
2. Demonstrate a successful build using a fresh clone from Github.com repository into a system where the code has not previously been used.



Some IPAC software systems use automated continuous integration, where a build is automatically executed either periodically or whenever a change is pushed to the software repository.

Code Release Review Criterion: Style and Professionalism

Will this code embarrass anyone?

1. Any inappropriate comments in the code?
2. Does the code have commented-out components that serve no purpose?
3. Typos?
4. Any other stylistic issues that are so significant as to be embarrassing to the organization or the sponsor?

```
 */
public Position( double        lon,
                 double        lat,
                 CoordinateSys coordSystem)
                                  throws IllegalArgumentException {

       // the following line is about as clear as mud, what it is doing
       // is passing an epoch of 1950 when the coordinate system is
       // b1950, an epoch of 2000 when J2000 and epoch to 2000 in
       // all other cases.  In all other cases the epoch is meaningless
       // so it is just a placeholder
    this(lon,lat,null,coordSystem,
      coordSystem.equals(CoordinateSys.EQ_B1950) ? EPOCH1950 : EPOCH2000);
}
```

Code Release Review Criterion: Documentation

Superfluous content can be as bad as missing documentation.

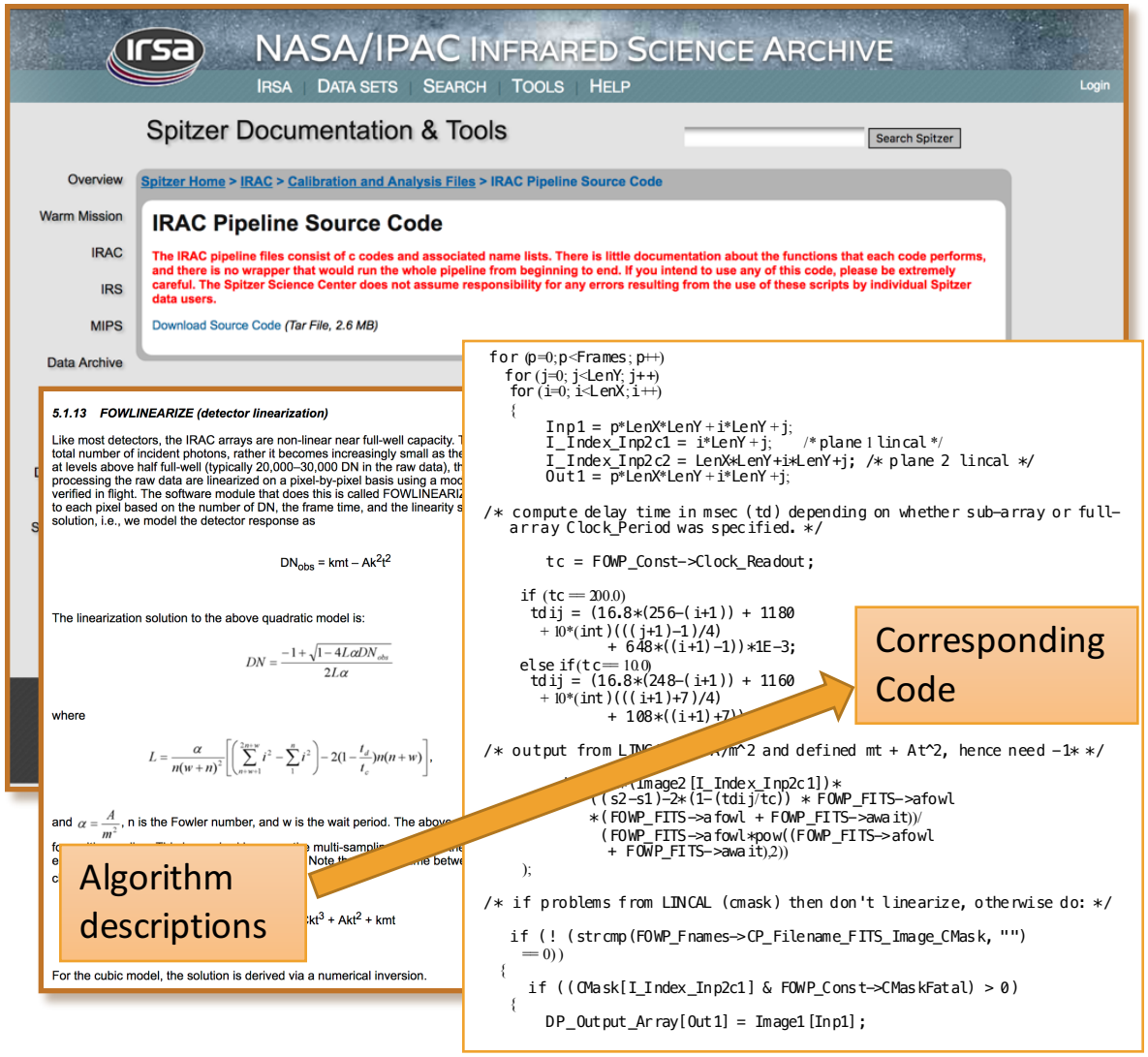
1. Is the associated documentation current, accurate, complete, organized, and professional?
2. Is there any documentation or material related to the system which does not need to be included and should be removed? Some examples we've encountered:
   - Word documents with revision histories still embedded.
   - Early design documents which no longer accurately reflect the system.

Even when complete documentation exists, it can take time to make sure that each item is appropriate for public release. In effect, the documentation itself must be regression tested for the software system.

Case Study: The Spitzer Pipeline

Spitzer IRAC pipeline source code was released.

http://irsa.ipac.caltech.edu/data/SPITZER/docs/irac/calibrationfiles/pipeline/



- 1998 design decisions—driven by requirements and costs—meant that pipeline code is not buildable outside of Spitzer environment (specific connections to Informix database, Spitzer filesystems)
- Recognized as an issue after launch but no resources for complete infrastructure change to make exportable modules for all pipelines
- Algorithms for all instrument pipeline modules are documented in the Instrument Handbooks
- Mosaicking and Source extraction pipelines, and multiple other tools, released as user-buildable code
- IRAC pipeline source code released with warnings (2638 files, 490k lines of code)

Considerations with Wider Code Release

Enforcing public code release could impact the cost of NASA missions.

Capability:

- Public release can provide volunteer external review and validation of algorithms
- Open source development may lead to external bug catches and algorithm improvements.
- In some cases, projects may require applications where no open-code alternative to a commercial package exists, and the cost of developing a new open-code solution is prohibitive.

Intellectual Property:

- Typically the implementing institution owns rights to the code. Caltech Office of Technology Transfer has been very helpful with code release.
- Can probably be addressed by contract with NASA.

Security:

- ITAR and export-compliance restrictions. Source code related to covered technologies is explicitly covered.
- Exposing source code can increase the risk of a successful hack attempt. It may also provide broader feedback on security vulnerability.
- Some config files or parameter files probably cannot be released because of security concerns.

Licensing:

- Will need a guiding principal on software license. For example, BSD-3.
- Software components with more restrictive licenses cannot be released under less restrictive licenses.
- Some restrictions on use (misuse) probably needed.

Cost:

- Most projects are not costed with public code-release as part of the estimate:
  - additional review activities.
  - development and documentation costs may be higher for publicly-released code.
  - additional support is required to respond to queries about released source code. This cannot be avoided without damaging project and organization reputations.
- Code may be used for cases not covered by original requirements and testing. Release notes must give bounds on support.
- Need to find a support-model for released code after project termination. What happens when code no longer builds on any currently-supported compiler version?