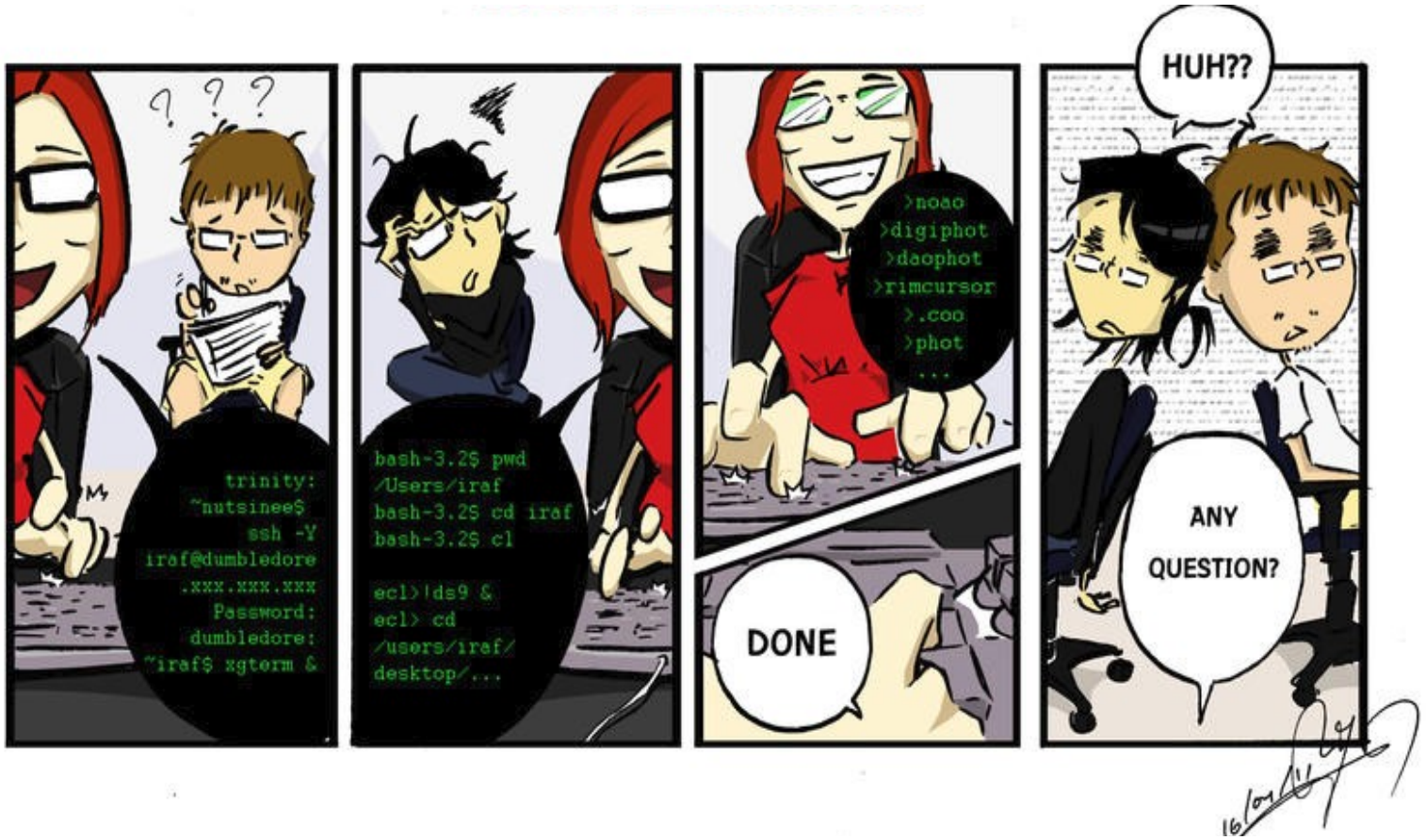
Leibniz-Institut für
Astrophysik Potsdam

Maintenance of IRAF legacy code

Ole Streicher / AG2023|Escience / 2023-09-14



How to deal with legacy code?

Specifics in astronomy

- ***very*** old code
- Examples:
  - AIPS (development started ~1978)
  - IRAF (~1981)
  - ESO-MIDAS (~1995)
  - SAOImageDS9 (~1995)

Methods

- Use unchanged in a virtual machine, docker (black box)
- **Continue (limited) maintenance of the original software ⟸**
- Refactoring; extract useful parts into a new ecosystem
- Re-implement the code in a new environment

Short history of IRAF

The Rise of IRAF

- Development started 1981
- First internal version 1984
- First public release (2.8) 1987

- First “poor astronomer” releases: Linux 1996 (2.10), MacOS X 2002 (2.12)

Deprecation

- 2nd half of 1990’s:
  - Attempt to modernize IRAF (OpenIRAF) failed
  - STScI decided to move away from IRAF: PyRAF, pyfits, …
- Since ~2000, development and support significantly slowed down; halted ~2005
- Volunteers period (Mike Fitzpatrick, Chisato Yamauchi, …)
- “limited approach” 2007-2013 (64-bit, VO compliance)
- No development since 2013 (however, recent rumors …)

Motivation

Personal motivation

- Volunteer approach; unfunded
- Debian Astro since 2014: Integration of all major astronomy software in Debian
- Several times got asked about IRAF
- IRAF appeared not suitable at that time
- No maintenance visible

Reasons

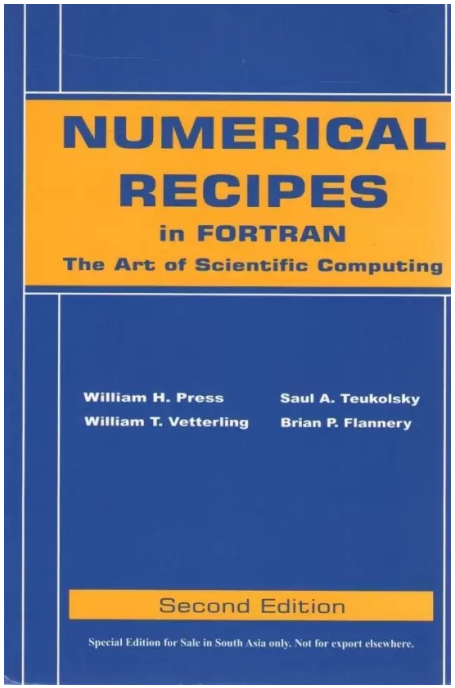
- Some users/use cases still rely on it
- IRAF is *the* reference
- Cultural heritage
- Have fun!

debian
astronomy

Legal problems

Numerical Recipes

- Very popular book in 80s/90s, with code for learning purposes
- Code often pirated these days (and up to now!)
- IRAF: probably had (paid) permission to “use” NR code
- But probably no permission to distribute as Open Source
- Need to identify and replace
- ~40 places found in IRAF
  - Replaced with free code or clean room implementation, or
  - Removed
- STSDAS has the same problem
  - May take over some of the changes
  - Others (Levenberg-Marquard, Gauss-Jordan) still needed



Technical issues I: Portability

Portability in the 80s

- Large number of incompatible platforms (“Unix like”)
- Tons of compiler and OS bugs, peculiarities, … need specific workarounds
- IRAF solution: “Virtual Operating System”, home-grown language (SPP)

Portability today

- Conform to portable standards (like POSIX)
- Adhere to portable standard solutions and languages

IRAF Portability problems

- 64 bit
  - two attempts, years of development, only non-standard memory model
  - Important external packages never ported (STSDAS)
- „mem“ common block for dynamic memory allocation
- Required “tricky” assembler code

Technical issues II: Specific language

SPP “Subset Preprocessor Language”

- Specific derivative (dialect?) of Ratfor “Rational Fortran”
- Preprocessed into FORTRAN 66
- Inherits limitations of old FORTRAN (“syntactic sugar”)
  - Common blocks, no structures, … → cause for 64-bit issues
  - But: dynamic memory management
- Complicated bootstrap process
- SPP Preprocessor is itself written in Ratfor
  - Deprecated since 80s
  - Fortunately some (very responsive!) maintainance
- FORTRAN code converted to C with f2c
  - Another outdated program
  - Still somehow maintained (netlib)
  - Patched version in IRAF

```
task alpha, beta, epsio=eps

procedure alpha()

int npix, clgeti ("npixi")
real lcut, clgetr()
char file[SZ_FNAME]

begin
     npix = clgeti ("npix")
     lcut = clgetr ("lower_cutoff")
     call clgstr ("input_file", file, …
     .
     .
```

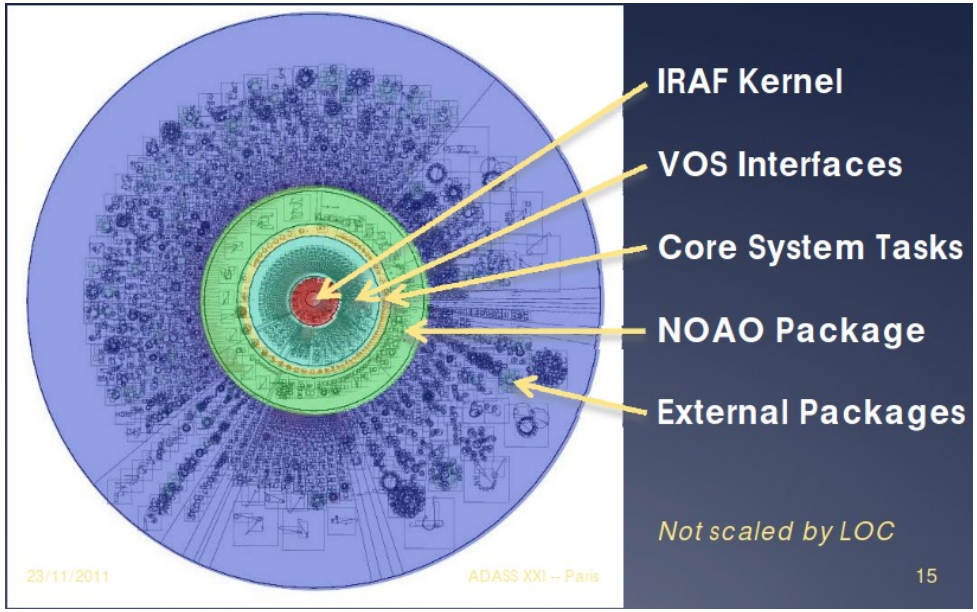
Technical issues III: OS dependencies

IRAFs Virtual Operating System

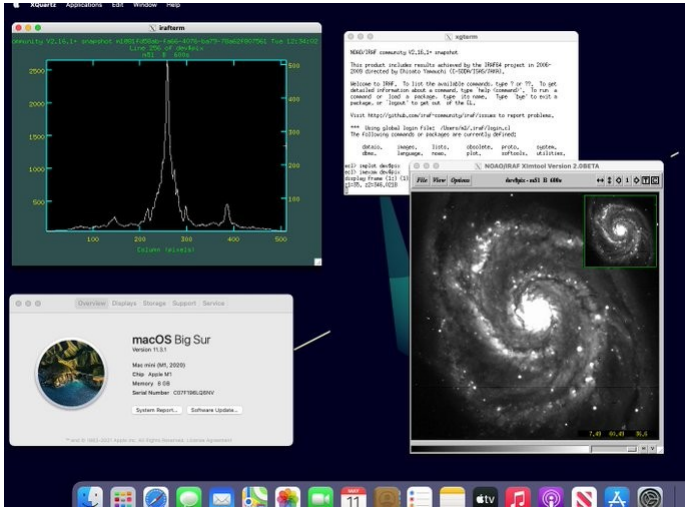
- Unix-style SPP/Fortran API, shields direct OS access (fio, imio, ...)
- OS specific code (mainly) in `$iraf/unix/os/`

OS specific code, kernel

- Evolved since ~40 years
  - #ifdef
  - Workarounds
  - Ad-hoc implementations and fixes
- Needed a cleanup
- Improve robustness
  - C prototypes
  - Display (remove) compiler warnings



Technical issues IV: User interface



IRAFs user interface

- Text I/O: terminal
- Pixel graphics (image): ximtool, SAOImageDS9
- Vector graphics: xgterm

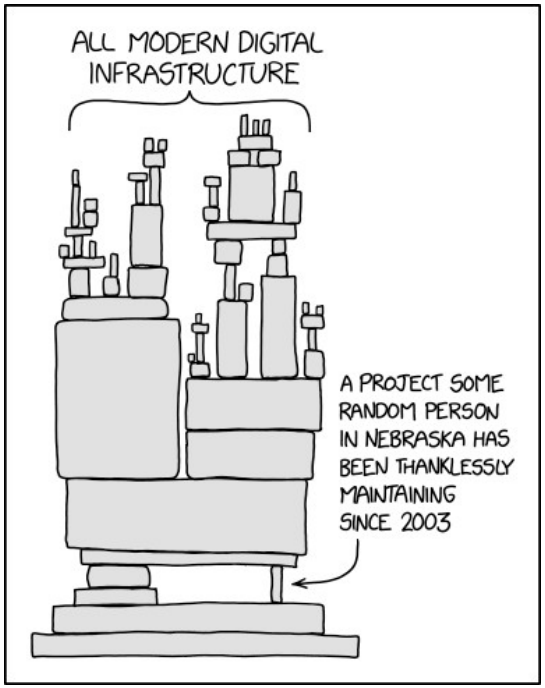
X11iraf

- Xgterm (text+vector graphics), ximtool (pixel graphics)
- Tightly bound to X11
- X11 is deprecated
  - MacOS doesn’t install X11 by default anymore
  - Linux migrates to Wayland
- Xgterm needs a complete rewrite from scratch (take over parts of PyRAF?)
- Help needed

Technical issues V: Dependencies

IRAF dependencies

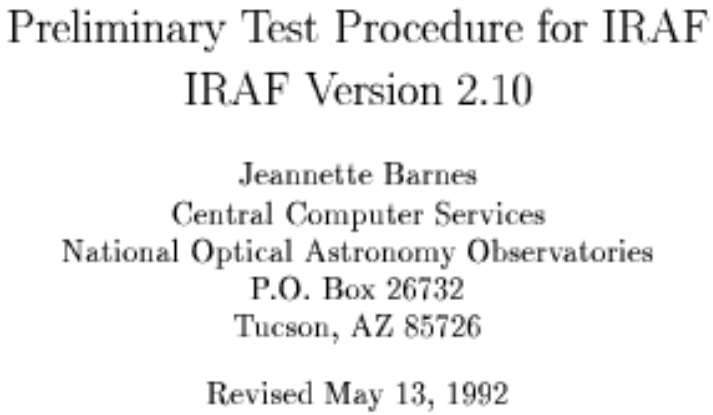
- *Fortunately, old software has only a limited number of external dependencies*
- C compiler
- F2c (included, patched) or Fortran compiler with ILP64 memory model
- Lex, yacc (be aware of evolution and incompatible changes)
- Ratfor
- Cfitsio (included)
- Curl, openssl, expat (VOTable support)
- Ncurses
- Readline or libedit
- Tcl, several X11 libs (xaw, xaw3d, xmu, xpm) for x11iraf



Technical issues VI: Tests

Original IRAF tests

- Tests were done before release (manually?)
- Site installation test document available
- (Almost) no (documented) formal tests



Preliminary Test Procedure for IRAF
IRAF Version 2.10

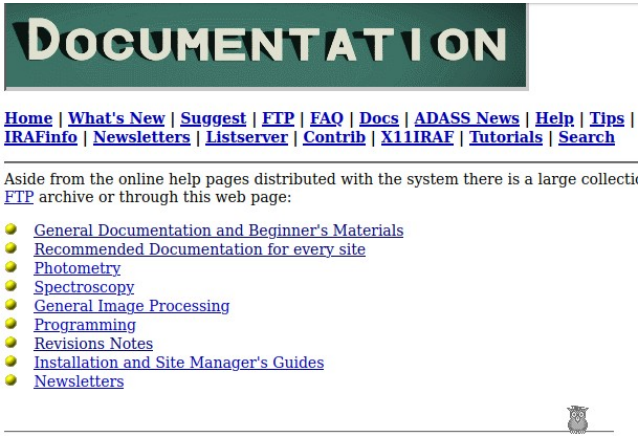
Jeannette Barnes
Central Computer Services
National Optical Astronomy Observatories
P.O. Box 26732
Tucson, AZ 85726

Revised May 13, 1992

Automated tests (new)

- Simple, „doctest“ style (based on MarkDown)
- Problem: code size of IRAF (~615,000 LOC in 2.8)
- `testproc.ps` (non-interactive part), beginner’s guide
- Tests of software environment (SPP, CL, mkpkg, generic)
- Few tests of machine specifics (`zsvjmp`, date/time)
- Tests for changes and fixed bugs
- Few random tests on some packages (lists, images, nttools, …)

Documentation



IRAF documentation

- Old software often badly documented
- **This is not the case for IRAF!**
  - Well-structured, enforced documentation for >1000 tasks
  - Lots of supplementary documentation: CL+SPP reference, programming standards, site managers guide
  - Inline documentation of function API
- Original documentation formats:
  - nroff, Lroff (homegrown, simple but powerful markup language)
  - LaTeX
  - Now published on https://iraf.readthedocs.org
- Contributed documentation:
  - zillions tutorials, 3rd party documentation
  - Many astrophysical standard procedures covered
- Complexity still a problem

Development workflow

IRAF versioning

- Originally distributed by architecture (up to ~10 archs per version)
  - Same version not identical between architectures
  - Patch versions; only the last one was preserved
- Code evolution often unclear
- Incomplete, sometimes unspecific revision logs
- Distributed as source/binary tarballs or patch sets

Github

- IRAF history hard to squeeze into git workflow (or any RCS)
- independent branches:
  - „main“: started with 2.16.1
  - „history“: versions along „CXOS/SOS4/LNUX/PCIX“ architectures
  - „iraf64“, „stsci-ureka“: 3rd-party branches

Credits

IRAF authors and contributors

- IRAF didn’t acknowledge individual authors
- Created a list of contributors
  - Compiled from revision notes, IRAF and NOAO newsletters
  - Very general
  - Probably incomplete ← *Help needed!*
- Contributors since 2017: collected with Github
- Total: 41 contributors identified
- Specifically to mention
  - Doug Tody (Inventor and development lead for many years)
  - Mike Fitzpatrick (Maintenance over >30 years)

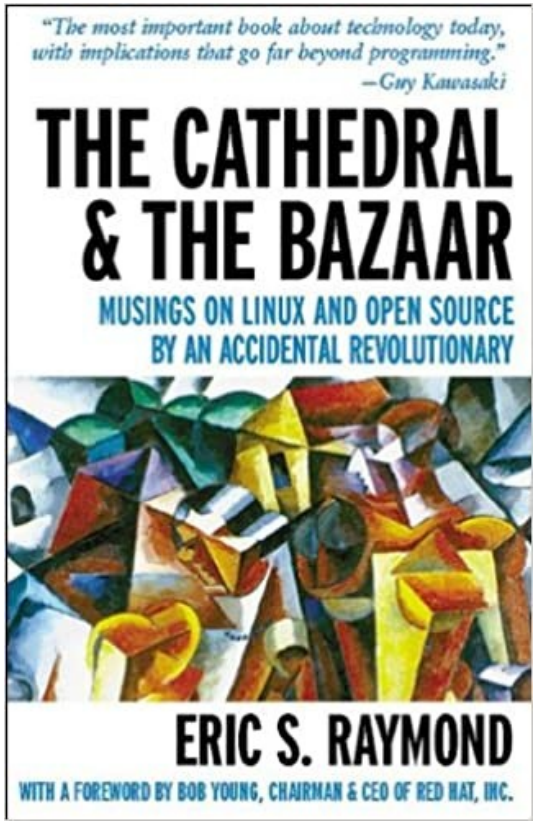
Credits II

- Ed Anderson
- Jeanette Barnes
- David Bell
- Chris Biemesderfer
- Todd Boroson
- Matt Cheselka
- Mike Cobb
- Dennis Crabtree
- Lindsey Davis
- Michele De La Peña
- *Christian Dersch*
- Elwood Downey
- Jonathan Eisenhamer
- **Mike Fitzpatrick**
- Pedro Gigoux
- Rick Hill
- George Jacoby
- Suzanne Jacoby
- Dyer Lytle
- Eric Mandel
- Phil Massey
- Tom McGlynn
- Drew Phillips
- Fred Romelfanger
- Steve Rooke
- Jim Rose
- Skip Schaller
- Rob Seaman
- Peter Shames
- Richard Shaw
- Peter Stetson
- Cliff Stoll
- *Ole Streicher*
- **Doug Tody**
- Jay Travisano
- Frank Valdes
- *Josef Wang*
- Phillip Warner
- Richard Wolff
- *Chisato Yamauchi*
- Nelson Zarate

Social issues I

The Cathedral: IRAF

- Carefully selected development team
- Road map defined by NOAO
- Only final versions are published
- No general knowledge about IRAF internals
- Almost no external contributions
  (or contributions went in unacknowledged)



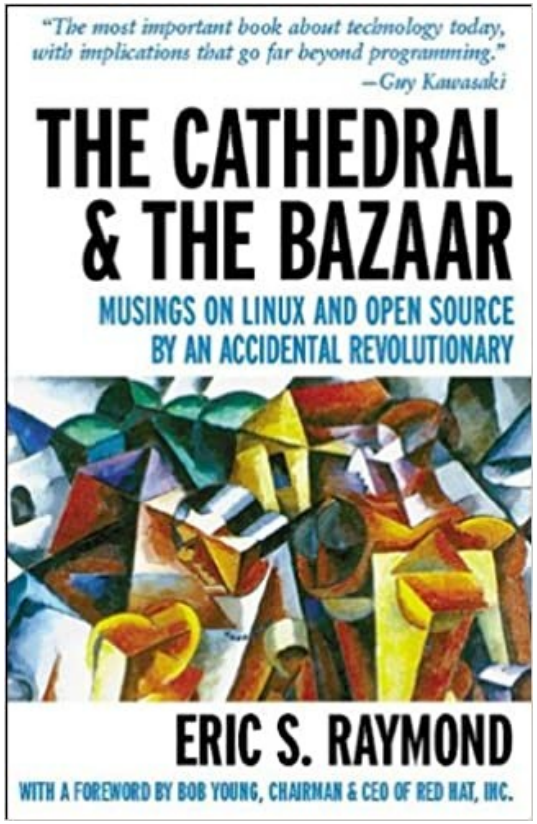
Todays standard: Bazaar/Open Development

- Development as an open, transparent process
- Community developed road map
- Encourage users to contribute code, documentation, …
- Large base of contributors
- Example: Astropy (>450 co-authors in core package)

Social issues II

Future: Open Development of IRAF

- No dedicated institutional support
- No contributions from original authors anymore
- Loosely coupled interest in maintenance
- But: de facto no real community (just a single person)
  - Too late?
  - Need to keep internal knowledge and experience
  - „Bus factor“
  - How to spread?



IRAF Universe

PyRAF

- Developed by STScI from ~1997 as path-maker to Python
- Abandoned 2018
- Converted to Python 3 now

STSDAS

- Major IRAF package since early days
- Infected by NR code
- Not 64-bit ready
- No plans to maintain it (volunteers?)

CTIO, ADCCDROM, DEITAB, MSCRED, SPTABLE, NFEXTERN, VO, …

- Lots of extension packages
- Unclear which are required

Thank you!

IRAF related links

- Homepage: https://iraf-community.github.io
- Git repository: https://github.com/iraf-community/iraf
- Documentation: https://iraf.readthedocs.org
- Discussion forums:
  - https://github.com/iraf-community/discussions
  - https://iraf.net

Titlepage comic strip was taken from https://www.deviantart.com/nk-c