Reworking CASA's build system

Jonas Larsen
CASA – build system perspective

- For Linux, Mac OS
- C++: 2.2 million physical lines of code
- Python: 0.16 mploc, Fortran: 0.10 mploc, Xml: 0.08 mploc, C: 0.015 mploc
- Many external dependencies packages (BLAS, LAPACK, FFTW, WCSLIB, ATM, RPFITS, CFITSIO, Libxml2, Xerces, Saxon, Java, Readline, Python, iPython, CCM Tools, D-Bus, X11, Qt 4, Qwt, PGPLOT, Boost)
AIPS++ build system

- Implemented in GNU make
- The quest for a correct makedefs

```bash
# WCSLIB (supplied with AIPS++); clear WCSLIBTRIG to ignore wcstrig.c.
WCSLIBROOT := $(AIPSARCH)
WCSLIBDEFS := -DSIGNBIT
WCSLIBINCD  = $(WCSLIBROOT)/include
WCSLIBLIBD  = $(WCSLIBROOT)/lib
WCSLIBLIB   := -lwcs -lpgsbox
WCSLIBTRIG  := 1
```

- Build time (incl. CASACore) is 1h 40 minutes on 4 x 2.8Hz
- Non-obvious messaging (81,000 lines sneeze.log with a background of false error messages. Was the build successful?)
AIPS++ build system

• Fragile build mechanics (addition of header files, “unresolved symbols”, generated code), incomplete builds, redundant rebuilds

• Make clean, and the problem went away

• Lack of up-to-date documentation (e.g. parallel builds)

-> Unreliable behaviour at runtime!

-> A time “tax” on (also experienced) developers
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Priority</th>
<th>Satisfied</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 Portability</td>
<td>critical</td>
<td>yes</td>
<td>(A3)</td>
</tr>
<tr>
<td>R2 Documented build procedure</td>
<td>critical</td>
<td>no</td>
<td>A5 Monitor builds</td>
</tr>
<tr>
<td>R3 Dependency packages flexible location</td>
<td>non-critical</td>
<td>yes/no(^a)</td>
<td>A8 Distribute external packages as source</td>
</tr>
<tr>
<td>R4 Verify external packages</td>
<td>critical</td>
<td>no</td>
<td>A4 CMake configuration</td>
</tr>
<tr>
<td>R5 Debug builds</td>
<td>critical</td>
<td>yes</td>
<td>(A3)</td>
</tr>
<tr>
<td>R6 Documentation build</td>
<td>non-critical</td>
<td>yes</td>
<td>(A3)</td>
</tr>
<tr>
<td>R7 Widely used technology</td>
<td>critical</td>
<td>no(^b)</td>
<td>A3 CMake build system</td>
</tr>
<tr>
<td>R8 Simple/correct builds</td>
<td>critical</td>
<td>no</td>
<td>A3 CMake build system</td>
</tr>
<tr>
<td>R9 One build method</td>
<td>non-critical</td>
<td>no</td>
<td>A1 Use SCons</td>
</tr>
<tr>
<td>R10 Work per module</td>
<td>non-critical</td>
<td>yes</td>
<td>(A1, A3)</td>
</tr>
<tr>
<td>R11 Unit test data</td>
<td>non-critical</td>
<td>yes</td>
<td>(A3)</td>
</tr>
<tr>
<td>R12 Faster builds</td>
<td>critical</td>
<td>no</td>
<td>A3 CMake build system, A11 Clean up code tree</td>
</tr>
</tbody>
</table>
### Requirements (2)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Criticality</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>R13 Out-of-source builds</td>
<td>non-critical</td>
<td>no</td>
</tr>
<tr>
<td>R14 Release structure</td>
<td>non-critical</td>
<td>no</td>
</tr>
<tr>
<td>R15 Better unit test coverage</td>
<td>non-critical</td>
<td>no</td>
</tr>
<tr>
<td>R16 Modular UT framework</td>
<td>non-critical</td>
<td>no</td>
</tr>
<tr>
<td>R17 Identify version</td>
<td>non-critical</td>
<td>no</td>
</tr>
<tr>
<td>R18 Build only good versions</td>
<td>critical</td>
<td>no</td>
</tr>
<tr>
<td>R19 Synchronize uncommitted changes</td>
<td>non-critical</td>
<td>yes</td>
</tr>
<tr>
<td>R20 Keep intermediate object files</td>
<td>non-critical</td>
<td>no</td>
</tr>
<tr>
<td>R21 External package runtime version numbers</td>
<td>non-critical</td>
<td>no</td>
</tr>
<tr>
<td>(A3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A7 Revisit release contents</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A9 Implement unit tests</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3 CMakel build system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2 Redefine build number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A6 Build server tests</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A10 Runtime version number checks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Task python binding generation

Notation:
- Build tool
- Built file

Diagram:
- `<task>.xml`
- `*.xsl`
- `sed`
- `task_<task>.py`
- Saxon
- `<task>.py`
- `<task>_cli.py`
- `<task>_pg.py`
- `tasks.py`
- `taskinfo.py`
Tool python binding generation

- conversions.xml
- casa2toolxml.xsl
- <tool>.xml
- Saxon
- casa<tool>.xml
- Saxon
- casa2idl3.xsl
- <type>.idl
- casa<tool>.idl
- CCM Tools
- CCM Tools
- casa<tool>.idl
- CCM Tools
- casa<tool>.idl
- <types>.idl
- casa<tool>.idl
- casac_python.cc
- casac_python.h
- *<types>*.cc
- *<types>*.h
- casac_python.cc
- casac_python.h
- *<tool>*.cc
- *<tool>*.h
- *<types>*.cc
- *<types>*.h
- g++
- *.o
- g++
- casac.so
- "import casac"

Interdependencies via "#include <file>"
## A survey of build tools (1)

<table>
<thead>
<tr>
<th></th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNU make</td>
<td>Simple</td>
<td>Low level, non-portable shell</td>
</tr>
<tr>
<td>AIPS++ and autoconf</td>
<td>Reuse</td>
<td>Does not address issues with build mechanics, difficult to maintain</td>
</tr>
<tr>
<td>ACS makefiles</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boost.Build</td>
<td></td>
<td></td>
</tr>
<tr>
<td>qmake</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apache Ant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNU autotools</td>
<td>Does the job, wide-spread</td>
<td>Complex tool chain</td>
</tr>
<tr>
<td>CMake</td>
<td>Does the job</td>
<td>Mature?</td>
</tr>
<tr>
<td>SCons</td>
<td>Does the job, Python</td>
<td>Slowish, low level-ish, Python</td>
</tr>
</tbody>
</table>
A survey of build tools (2)
CMake build system (1)

CMakeLists.txt:

```cmake
# PGPLOT (FORTRAN plotting package).
# depends on X11
casa_find( PGPLOT
  VERSION 5.3.1
  INCLUDES cpgplot.h
  LIBS pgpplot cpgplot
  DEPENDS X11 )

# WCSLIB
casa_find( WCSLIB
  VERSION 4.3
  INCLUDES wcslib/wcsconfig.h
  LIBS wcs
    pgsbox
  CPP_VERSION WCSLIB_VERSION
  DEFINITIONS -DSIGNBIT
  DEPENDS PGPLOT )
```

cmake output:

```
-- Looking for WCSLIB header wcslib/wcsconfig.h
-- Looking for WCSLIB header wcslib/wcsconfig.h -- /diska/jmlarsen/gnuactive/linux_gnu/include/wcslib/wcsconfig.h
-- Checking whether WCSLIB headers compile
-- Checking whether WCSLIB headers compile -- TRUE
-- Checking that WCSLIB compile version is 4.3 or later
-- Checking that WCSLIB compile version is 4.3 or later -- found 4.3 -- ok
-- Looking for WCSLIB library wcs
-- Looking for WCSLIB library wcs -- /usr/lib/libwcs.so
-- Looking for WCSLIB library pgsbox
-- Looking for WCSLIB library pgsbox -- /usr/lib/libpgsbox.a
-- Checking whether WCSLIB links
-- Checking whether WCSLIB links -- ok
-- Checking whether WCSLIB links and runs
-- Checking whether WCSLIB links and runs -- ok
```
CMake build system (2)

CMake:
- find_library(...)
- add_executable(...)
- casa_find(...)
- casa_executable(...)

The CASA-specific infrastructure provides:
- Consistency among package finders
- make <module>, make <module>_fast
- "make install" as part of "make"
### Did anything break?

**Test vs. platform**

<table>
<thead>
<tr>
<th>Command</th>
<th>Platform</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3C129</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>accu-run</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>asdm-imp</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>asdmv1-i</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>async-se</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>b0319_re</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>b1608</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>caistat</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>clearsta</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>cyclev</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>f15s-imp</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>flagdata</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>fls3a_hi</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>gt192redux</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>gt20</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>gtgtu</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Eventually... no!
(Except OOM on 32-bit Snow Leopard)

+ GUI testing
Build documentation

Maintaining complete and verified correct build documentation, on multiple platforms

code/install/build.py:

```python
b.do("This directory must already exist, or the WCSLIB build
"mkdir -p ../*", builddir, "/bin")

b.do("mkdir -p ../*")
```

Compile WCSLIB, make sure that . is in the PATH, or the build will fail

```
.b.set_env('PATH', "$PATH:.")
```

```
.b.do("./configure --prefix=/", prefix, "/", builddir,
.b.do("Warning: Do not try parallel building (failures ha
-j 1")
.b.do("", "make install")
.b.chdir("..")
```

Compile WCSLIB, make sure that . is in the PATH, or the build will fail

```
.mkdir -p ../darwin/bin
```

```
./configure --prefix=/opt/casa/active/darwin F77=gfortran
```

```
checking build system type... i686-apple-darwin9.5.0
```

```
checking for flex... flex
```

```
checking for gcc... gcc
```

```
checking for C compiler default output file name... a.out
```

```
checking whether the C compiler works... yes
```

```
checking whether we are cross compiling... no
```

```
checking for suffix of executables...
```

```
checking for suffix of object files... o
```

```
checking whether we are using the GNU C compiler... yes
```

```
checking whether gcc accepts -g... yes
```

```
/output/casa/active_cmake/wcslib> export PATH="$PATH:
```

```
/output/casa/active_cmake/wcslib> ./configure --prefix=/opt/casa/active/darwin F77=gfortran FFLAGS=-fPIC CFLAGS=-fPIC
```

```
output
```
Does anyone else see the same?

http://www.eso.org/alma-casa-hudson/

Running on Hudson + VMware Fusion

CASA developers' meeting, May 11th, 2010
Conclusions

- Well-documented build procedure and build tool
- Configuration verification (incl. version numbers), feasible for more developers to develop on more platforms (Fedora Core, Scientific Linux, Ubuntu, ...)
- Out of source builds, modular unit test driver, better debugging support on Mac, ...
- Less developer time spent on building(?)
- Null build times dropped from 18 minutes to 20 seconds
- Hudson: The build still breaks (fact of life), but stayed broken 40 minutes - 5 hours