Sourdough Saloon in Beatty, CA
What Software Does an Observatory need?

- Timekeeping
- Track a source – ephemeris; coordinate transformation; telescope mount model
- Data acquisition – take raw data; calibration; other processing
- Control electronics and embedded devices
- Access telemetry
- Monitor error conditions – fault system, alarm
- User interface – programs, scripts, CLI or GUI?, help files
- Networking; communication between multiple computers
- Activity log
- Weather station – refraction correction; opacity; system temperature
- Archive the data – database management
- For the astronomers – Observational planning and data reduction
- Software engineering – revision control; regression testing; external libraries; bug-tracking; Object-oriented? Top-down? Spaghetti?
What Software Does an Radio Interferometer need?

All of the preceding plus:

• Correlator and downconverter – reconfigure bands; field programmable gate arrays (FPGAs)
• Delay and lobe rotation calculations
• IF/LO control
• Receiver tuning/optimization
• Holography
• Collision avoidance for compact arrays
• Linelength – remove extra phase (delay) due to changing cable lengths
• Atmospheric delay correction? – water vapor radiometry
How CARMA Does It – Timekeeping

• Master clock
  − Microprocessor controlled clock that distributes precision time.
  − 1 pulse-per-second signals from two GPS clocks/10 MHz rubidium oscillator
  − Computers are locked to this via Network Time Protocol

• Astronomical time
  − “frame count” – number of ½ second frames since Jan 1, 2000
  − Conversion between UTC, LST, MJD, standard unix data formats
  − Earth rotation corrections – UT1-UTC
  − Leap seconds
How CARMA Does It – Antenna Control

• All antennas appear identical to the control system despite their differences
  - Define a “common antenna” that is the union of capabilities/features
  - Those without a particular feature do nothing when commanded to change it

• Source tracking
  - Ephemeris code built on top of NOVAS package, compute RA, Dec for the current epoch (which is constantly changing)
  - Quadratic interpolation of position with 8 minute update rate
    • gives 1 arcsecond accuracy (fast-moving comets)
    • reduces computational load
    • for delays, need 20 second update. Use two threads.
  - Mount model to correct antenna AZ/EL (encoder, collimation, sag, axis misalignment, etc) applied in antenna computers (different for every antenna)
How CARMA Does It – Device Control

• **Controller Area Network**
  - Industry standard bus protocol for connecting electronic devices (CAN nodes)
  - Long range (ethernet)
  - Most CARMA low-level devices are CAN nodes
  - CAN messages are 29 bit packets containing destination address, message, and various ID and status bits.
  - CARMA has high-level, generalized wrappers to make it easy to command and query any CAN node

• **General Purpose Bus Interface (IEEE-488)**
  - Parallel interface designed by Hewlett-Packard in the 1960s (HPIB); standardized in 1975
  - Short range only (a few meters)
  - Linux libraries available
  - Frequency synthesizer; frequency counter; linelength system

• Also some serial devices (Weather station, phase monitor)
How CARMA Does It – Monitor Point Definition

• Every monitor point in each subsystem is described in ASCII file
  - **Monitor Point Markup Language** – an XML format with tags for subsystem, name, type, units, description (show example here)
  - Easy to add new monitor points (too easy?)
  - Generic monitor point templates and access code pre-defined
  - Specific subsystem access code is *generated* at compile time from templates and MPML files
  - MPML also defines monitor points for archive database
The Weather Station keeps track of the temperature, pressure, wind speed and direction, as well as humidity level. It also publishes a few derived quantities, such as water vapor density and column.

**MonitorPoint**

**timestamp** type="absTime" persistent="true" update="10">
  <shortName>Timestamp</shortName>
  <longName>Packet completion time</longName>
  <description>
    Time that packet data was completed/written
  </description>
  <units>utc</units>
</MonitorPoint>

**MonitorPoint**

**windSpeed** type="float" persistent="true" update="10">
  <shortName>Wind Speed</shortName>
  <units>mph</units>
  <description>
    Measured Wind speed, in miles per hour (mph)
  </description>
  <precision>1</precision>
  <errHi>30</errHi>
  <warnHi>20</warnHi>
  <warnLo>-0.01</warnLo>
  <errLo>-0.01</errLo>
</MonitorPoint>

**MonitorPoint**

**peakWindSpeed** type="float" persistent="true" update="10">
  <shortName>Peak Wind Speed</shortName>
  <units>mph</units>
  <description>
    Peak Wind speed, over the last (roughly) 5 minute sample, in miles per hour (mph)
  </description>
  <precision>1</precision>
  <errHi>30</errHi>
  <warnHi>20</warnHi>
  <warnLo>-0.01</warnLo>
  <errLo>-0.01</errLo>
</MonitorPoint>
How CARMA Does It – Monitor data flow

Frames are collected at each subsystem computer and transferred on the half-second to the Array Control Computer.

The gray arrows signify access using the hierarchical MonitorPoint API.

The black arrows show inter-computer data transport using the CORBA Notification Service.
How CARMA Does It – Interprocess Communication

- **Common Object Request Broker Architecture**
  - Open standard for interprocess communication across networks
  - Separates interface from implementation
  - All objects appear to be local
  - Can be bound to many languages (C/C++, Java, Python, etc)

![Diagram of Interprocess Communication](image)
How CARMA Does It – Interprocess Communication

- **Implementation Repository**
  - Stores information about CORBA server applications
    - is it starting, running, stopped?
    - what parameters was it started with?
    - has it stopped and been restarted?
  - Can start/stop any registered CORBA server application
  - Makes connection between CORBA applications transparent
    - clients don't know and care where servers are
    - if a server is down and client makes a call, the IMR will restart the server. The client never notices.

- **NOTE: Problem this morning with Tim's track the IMR itself failing!**
  - Solution: exit sac window then at unix prompt: carmaSystem restart
    then restart sac with “sci1”
How CARMA Does It – Science Data Pipeline

A = archived at NCSA
How CARMA Does It – Project Data Flow
How CARMA Does It – User Interface

• Command line interface
• Subarray Control functionality implemented in C++
• Python bindings to all subsystem objects via CORBA
  - Can invoke commands directly on objects
  - Easy-to-remember (hopefully!) wrappers for common tasks
• Displays
  - RTD invocation starts up individual C++ server on acc
  - Server sends all information about layout and content to Java client
  - Periodic updates (as fast as 2 Hz) with minimal size messages (as originally designed for use over modems)
  - See Shukla, Scott, & Weaver 1998 ADASS paper.
Includes C++, Python, Java, test code, config files, documentation
How CARMA Does It – Software Engineering

- Object-oriented – C++, Java, Python
- Code style guidelines
  - Because we have several programmers
  - Not only want code to look as similar as possible, but to use best practices for implementation (avoid gotchas)
- Concurrent Versioning System (CVS)
  - Keep track of changes in every part of code
  - Eliminates conflicting edits by multiple programmers
  - Helps when trying to find bugs
- Bugzilla – bug tracking software
  - Web interface (report, assign, accept, resolve)
- Unit tests – software that tests your software
- Tinderbox – automated build and test
Automated build web page