EVLA Monitor & Control Software PDR

E2E Interfaces:

Observation Scheduling

Complaints to: Boyd Waters

John Benson, Barry Clark, Tim Cornwell, Rich Moeser, PeggyPerley , Bruce Rowen, Kevin Ryan, Bill Sahr, Ken Sowinski, Boyd Waters

Originally, Bill Sahr was going to present this topic, but it really makes sense to have an e2e person do it.

The material discussed in this presentation is the result of the work of the EVLA M&C Computing group, and the e2e Team. But I probably got some stuff wrong and left some stuff out, so complaints to Boyd Waters.

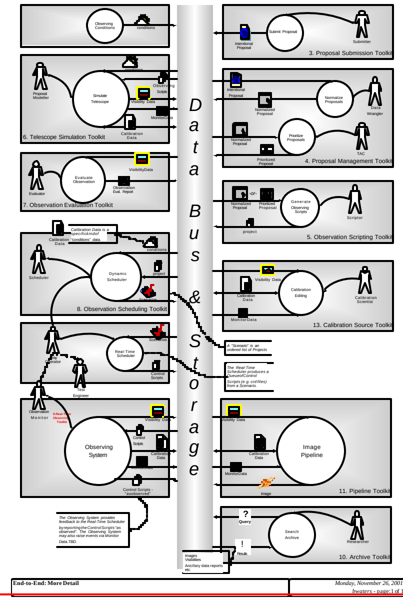
What is "e2e"?

- End-to-End Data Management
  - Capture and Preserve data…
  - …in a computer-useable format…
  - **THROUGHOUT THE PROJECT LIFECYCLE!**
- Motivation:
  - Automate where possible
  - Make Radio Interferometry POSSIBLE

May 15, 2002 EVLA Monitor & Control Software PDR Boyd Waters 2

Also, there's been lots of talk about the traditional, "roach motel" model of data archival: data checks in, but never checks out! So we'd like to provide meaningful access to the data that we (NRAO) already have.

e2e Architectural Diagrams



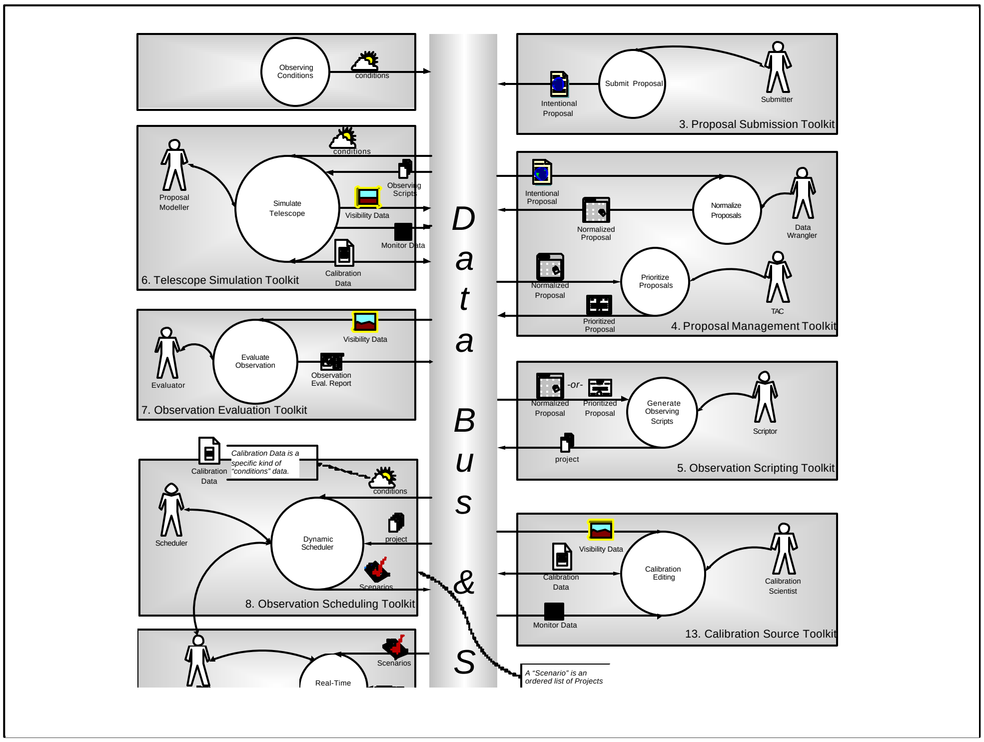
5/15/2002 Boyd Waters <bwaters@nrao.edu> Boyd Waters 3

Here is our obligatory e2e architectural diagram.

I don’t expect you to follow this; the point here is that we have thought about the process of observing with NRAO instruments as data work-flow. We have identified major steps along the work flow, and are building toolkits for many of these major steps.

The next two slides are slightly larger versions of this diagram.

(You can get a (theoretically) printable version of this diagram in a larger format from the e2e web site at http://www.nrao.edu/e2e)



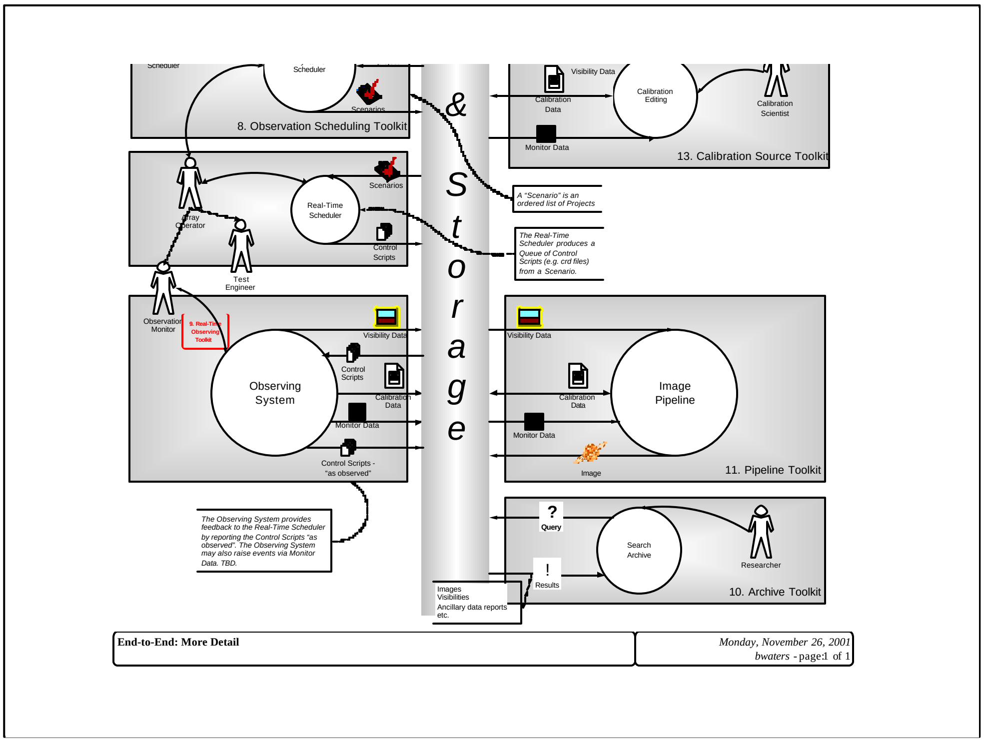
Here's the top half of the e2e Architecture Diagram.

Roughly, the right side of the diagram models interactions with hardware (e.g., the telescope, a weather station), and the left side of the diagram models interactions with people.

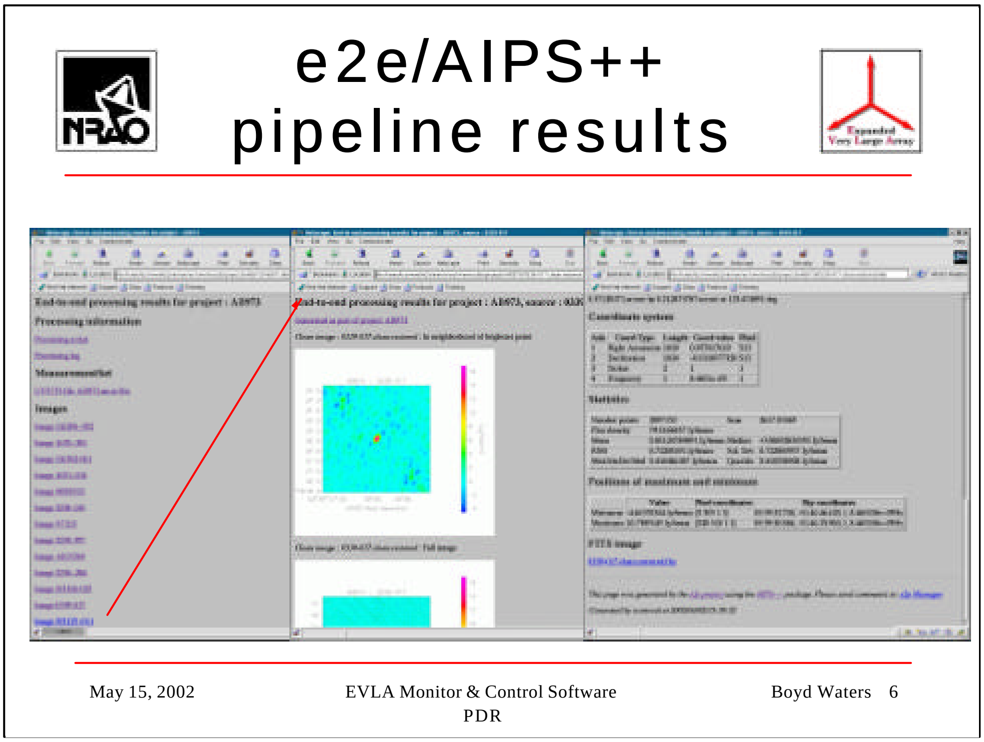
The data flow starts at the top-right of the diagram, with the Proposal Submission; Submissions turn into Approved Proposals, which are processed to obtain a Project that contains observation instructions (in a scripting language).

Sets of Projects are examined by the Dynamic Scheduler, which orders the Projects according to observing logic based on current conditions and observation constraints (e.g., good weather for high-frequency observing).

An ordered set of observing instructions – called a Scenario – is passed to a "Real-Time Scheduler" that turns the observing instructions (which are expressed in a human-readable scripting language) into more-detailed instrument instructions to be executed by the telescope. I'll talk about this more in a later slide.



Here's the bottom half of the architecture diagram, which shows the punch-line: the Image Pipeline and the Archive Query.

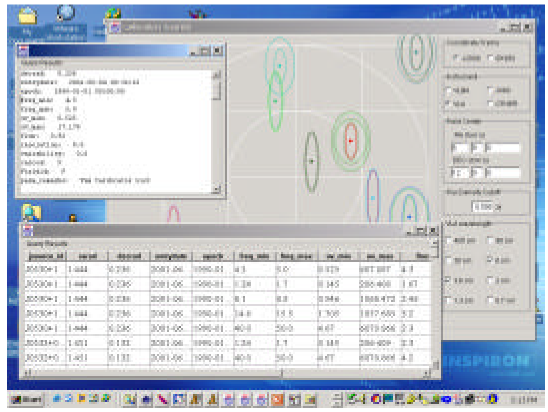


Tim Cornwell wrote this prototype Image Pipeline as a production-rule system implemented as GNU Makefiles driving AIPS++.

This is what we want: every observation from the VLA (and other NRAO instruments) will result in a set of images that have been built automatically. Instead of a Radio-Astronomy Interferometer “black belt” sitting down at an AIPS++ workstation and reducing the data, the image pipeline will try to do the same thing. Can a computer be as smart as Chris Carilli? No. But if the system is supplied with sufficient information about the observation, it can make reasonable choices about the data-reduction process to produce these default datasets.

e2e: Calibrator Selection

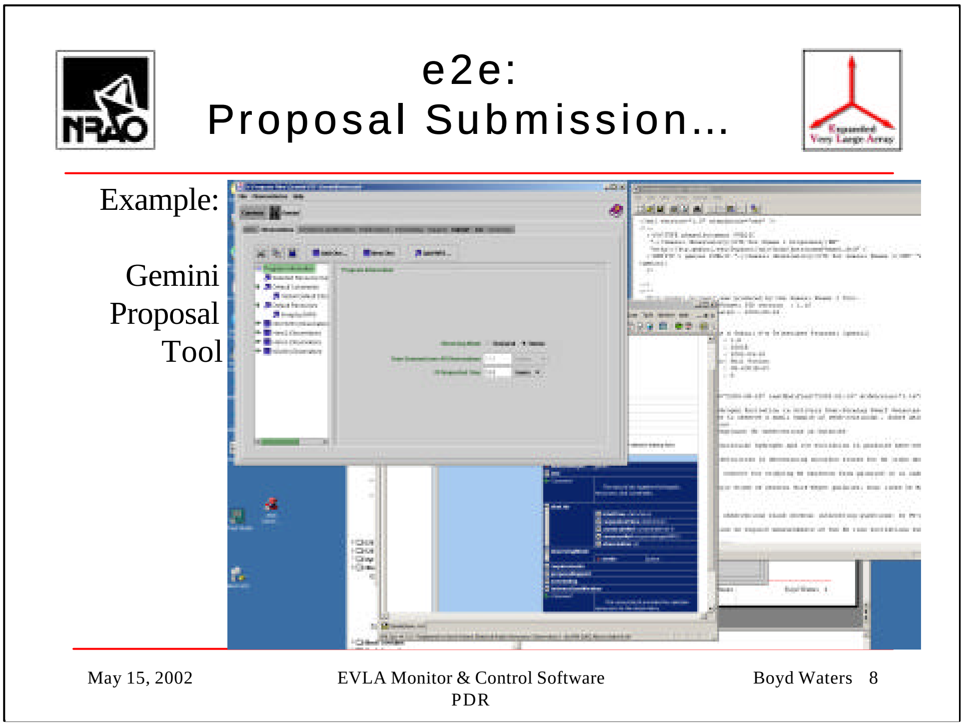
- Prototyped Java interface
- Will do user testing very soon
- Expect to deploy soon
- Make catalog available in AIPS++ for processing
- Also make available as an NVO cone search
  - more later



May 15, 2002 EVLA Monitor & Control Software PDR Boyd Waters 7

Honglin Ye wrote this Calibrator Selection Tool prototype with the help of John Benson, input from Greg.

The point here is that we are building tools all along the workflow chain, from proposal submission all the way to data reduction, to ensure that well-formed, complete data is entered into our data archive. A unified data model makes automated pipeline processing possible.

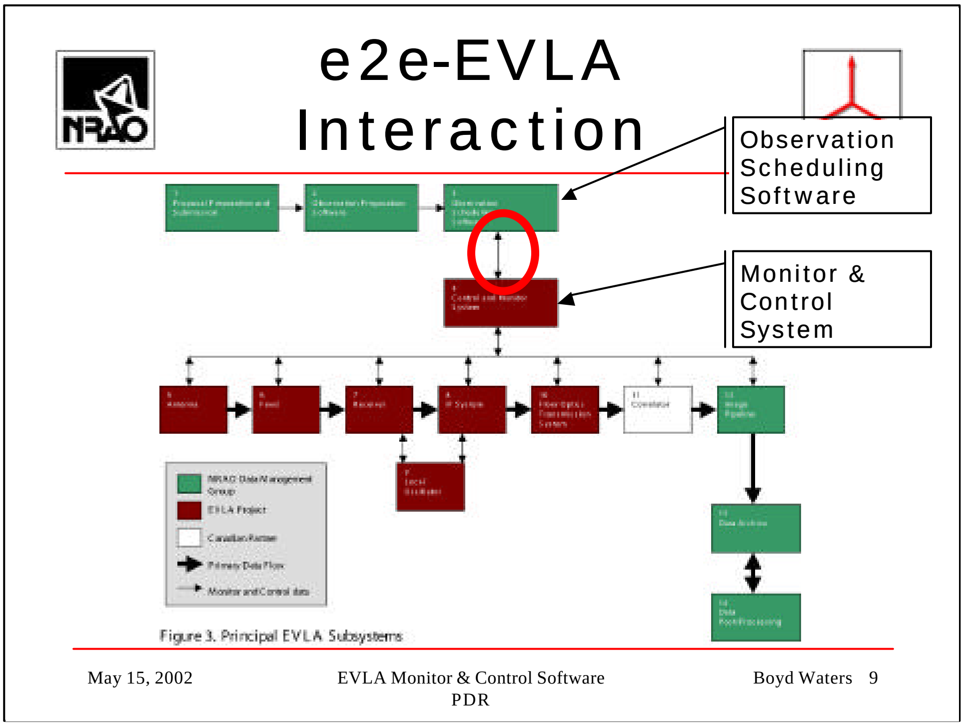


This is NOAO's Gemini Telescope "Phase I" Proposal Submission Tool. It's an example of the kind of submission tool we're thinking about.

The point here is that, with a some sort of forms-based editor, we can prompt the user for information about the observing proposal such that we obtain complete, structured information. We hope. In theory. Well, the editor will help…

In this screenshot, you can see the Gemini proposal tool as a Java application in the upper-left window. In the window on the right side of the screen, I've opened the resulting proposal in Notepad (a Windows text editor). You can see that the proposal is simply an XML file. In the bottom-middle window, I've opened this same proposal in a popular XML editor to emphasize the document structure.

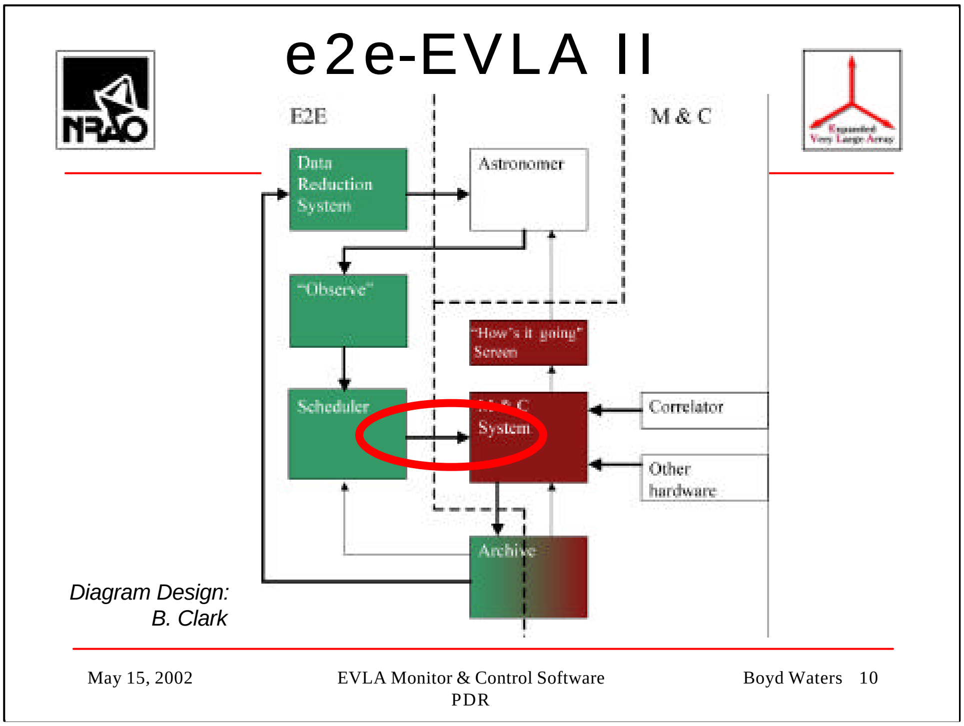
Writing the proposal as XML allows computer systems to interpret the structured data. We could use some other form like structured ASCII tables; XML is the fun thing to use at the moment. There are plenty of XML parsing tools available.



This is the data-flow diagram that was in the documentation submitted to the National Science Foundation.

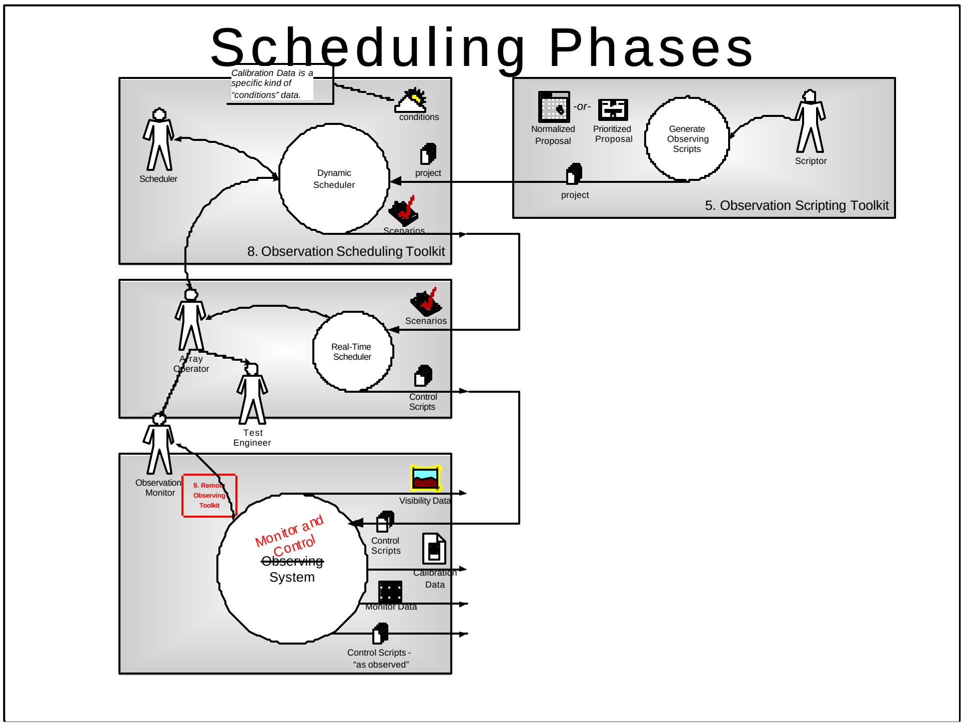
I don't expect you to read all the labels here. The point is that the green stuff is the primary responsibility of the Data Management Group (mostly the e2e Project), while the red stuff is the responsibility of the EVLA Project.

The major interaction between these teams, discussed in this presentation, is between the "Observing layer" and the "Monitor and Control" systems.



Barry Clark presented this diagram on the first day of the EVLA M&C PDR. I've added the colors consistent with the previous diagram, to emphasize the similarity between these two diagrams.

This diagram emphasizes that the piece of the "observing layer" that interacts with M&C is the Observation Scheduler.



The data flow starts at the top-right of the diagram, with the Proposal Submission; Submissions turn into Approved Proposals, which are processed to obtain a Project that contains observation instructions (in a scripting language).

Sets of Projects are examined by the Dynamic Scheduler, which orders the Projects according to observing logic based on current conditions and observation constraints (e.g., good weather for high-frequency observing).

Dynamic Scheduling

Structure the Observation…

…so that we can…

Respond to Events on a Short Time Scale

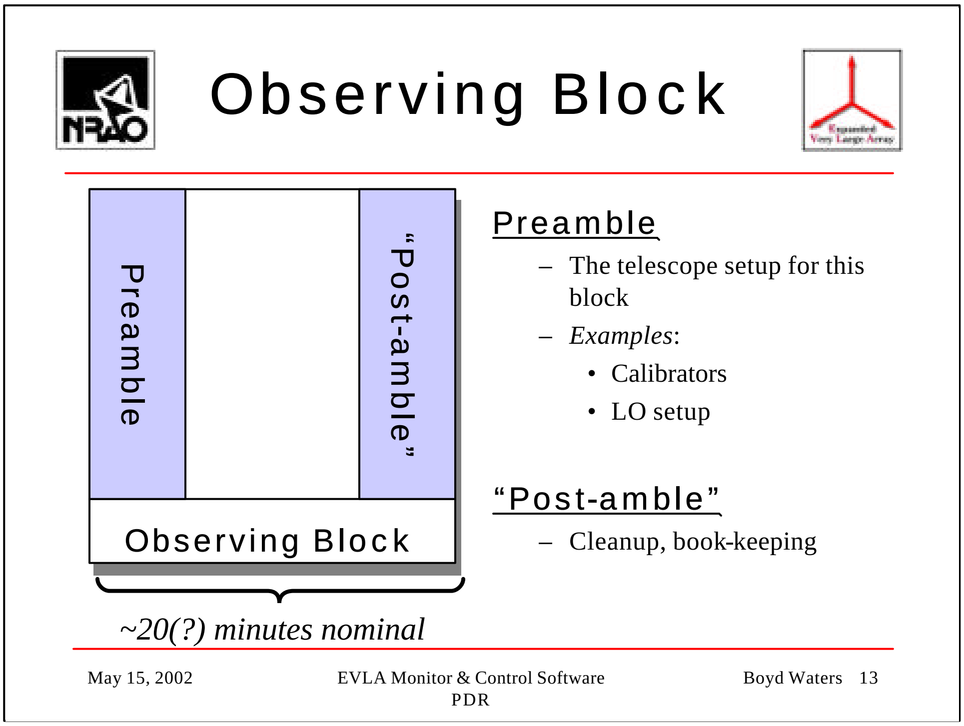
Changing conditions (e.g. weather)

Targets of Opportunity (e.g. GRBs)

…

➔Block-Based Scheduling

May 15, 2002 | EVLA Monitor & Control Software PDR | Boyd Waters 12



20 minutes is a typical calibration duty cycle. But the blocks can vary in size.

The internal structure used by the observing block is To Be Decided.

Observing Block Constraints

- “Run this block until the calibrations converge”
- “I can’t run unless the previous block has run successfully”
- “I MUST run at 16:42:30 GST on 30 May 2002”
  - (fixed scheduling is dynamic scheduling with time-domain constraints)

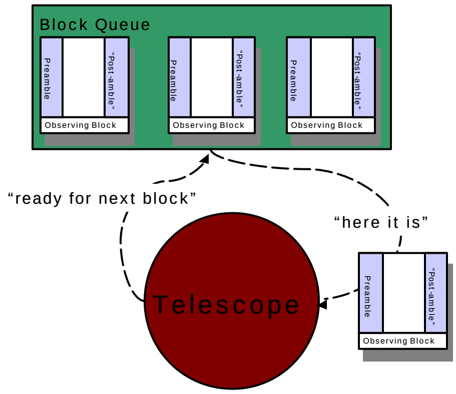
May 15, 2002 EVLA Monitor & Control Software PDR Boyd Waters 14

These are EXAMPLES, not REQUIREMENTS.

Requirements here have yet to be gathered, but we can collect such and use the requirements to get a better idea of what the observing blocks will need to express.

Block-Based Scheduler

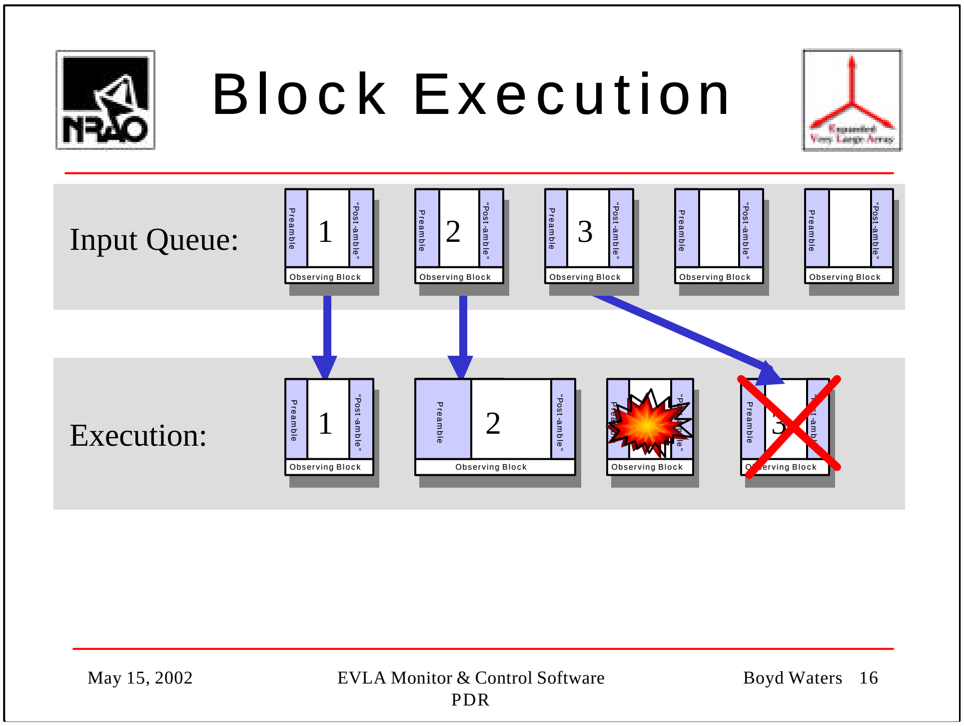
Telescope sees ONE BLOCK AT A TIME:

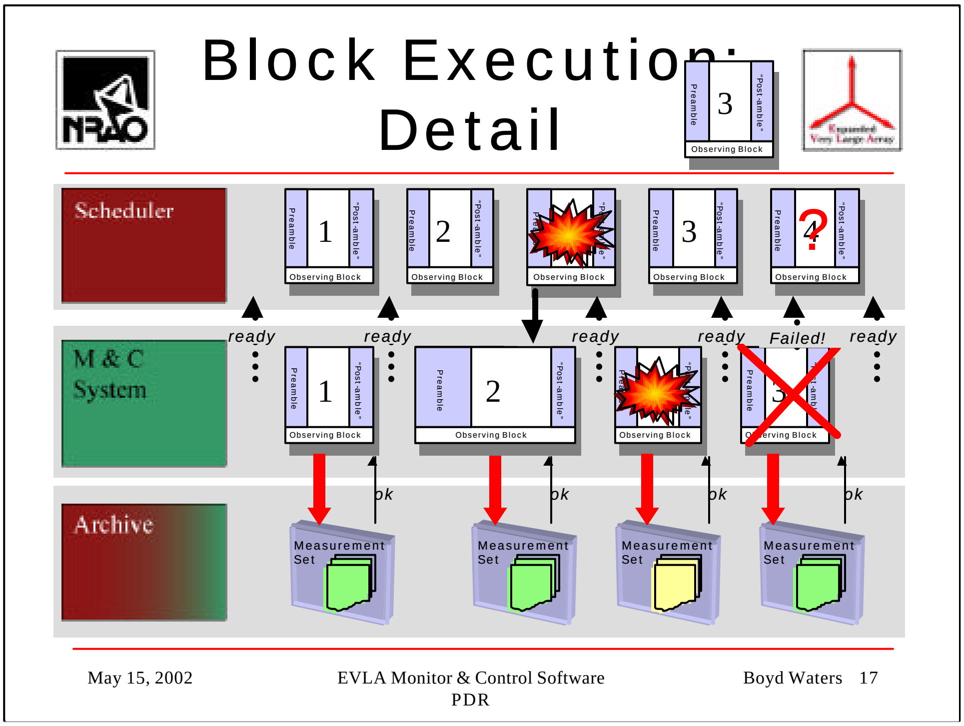


Implications:

- Simplifies the telescope state data
- Telescope reports block execution status back to the block queue
- All "observing logic" is maintained by the Block Queue

May 15, 2002 | EVLA Monitor & Control Software PDR | Boyd Waters 15


Block Execution
Input Queue:
Preamble
1
"Post-amble"
Observing Block
2
3
Execution:
May 15, 2002
EVLA Monitor & Control Software PDR
Boyd Waters 16


Block Execution:
Detail
Scheduler
M & C
System
Archive
Preamble
Post-amble
Observing Block
1
2
3
4
?
ready
Failed!
ok
Measurement
Set
May 15, 2002
EVLA Monitor & Control Software
PDR
Boyd Waters 17

Block Templates

- Debug blocks
- Template blocks
- “Default” blocks

Next Steps

- Gather and Codify Requirements
  - Observing Block constraints
- M&C ⇔Observing System Interaction
  - Formal Model
- Observing System Scripting
  - How instrument commands and observing constraints are expressed