

HAWC CDR

February 1-2, 2001

Software

Troy Ames, Lynn Case and Bob Loewenstein



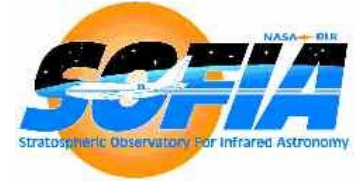
Agenda



-
- **Summary of Requirements**
 - **Design Overview**
 - **Major Changes Since PDR**
 - **Summary of Analyses Performed**
 - **Progress to Date**
 - **Test Plan**
 - **Open Issues**



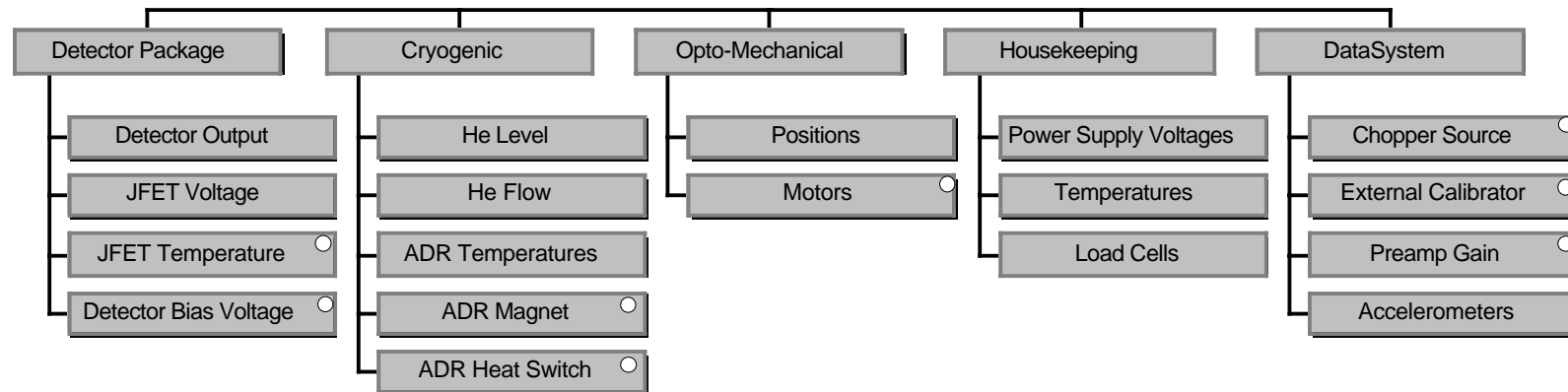
User Requirements (Level 1)



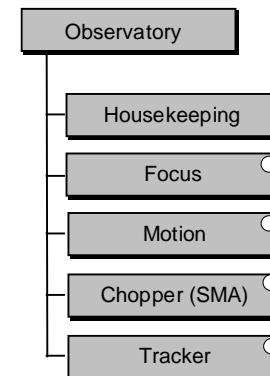
- **(1) Support full instrument operations in flight and on the ground**
- **(2) Maintainable by SSMOC staff**
- **(3) Easy to use by observer/user**
- **(4) Able to produce scientific quality data during the first year of operation**
- **(5) Support system integration and testing**



Hardware Monitor and Control



Monitor
 Control





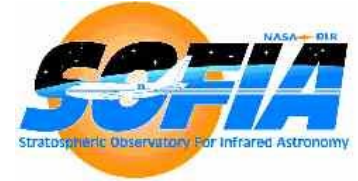
Design Concepts & Tools



- **Object-oriented design allows reuse across instrument subsystems**
 - Unified Modeling Language (UML) used to document the design in detail.
- **Java programming language used for cross-platform portability**
- **XML used to describe the interfaces between subsystems**
 - Provides a common way to describe interfaces of the instrument.
 - Instrument Markup Language (IML) used to describe command and data formats going to and from the instrument and its subsystems.
 - Command changes can be made without changing the underlying Java code.



Instrument Remote Control Software



- **HAWC Software is a specialization of the Instrument Remote Control (IRC) software.**
 - Provides a mature product to base HAWC software
 - Cost savings due to code reuse.
- **IRC is a proven product**
 - Demonstrated ability to visualize data from and control SPIREX (South Pole InfraRed EXplorer) heaters.
 - HAWC Prototype – introduced the use of XML for the Instrument Markup Language to define instrument characteristics
 - SPIRE downselect support –
 - proved IML could be used to support a second instrument
 - Java performance stress-tested
 - Achieved 15Mbps on an NT PC, although this was a data pass through with no archival of data.
 - HAWC data rate will be approximately 6Mbps, so Java should be able to achieve this data rate.

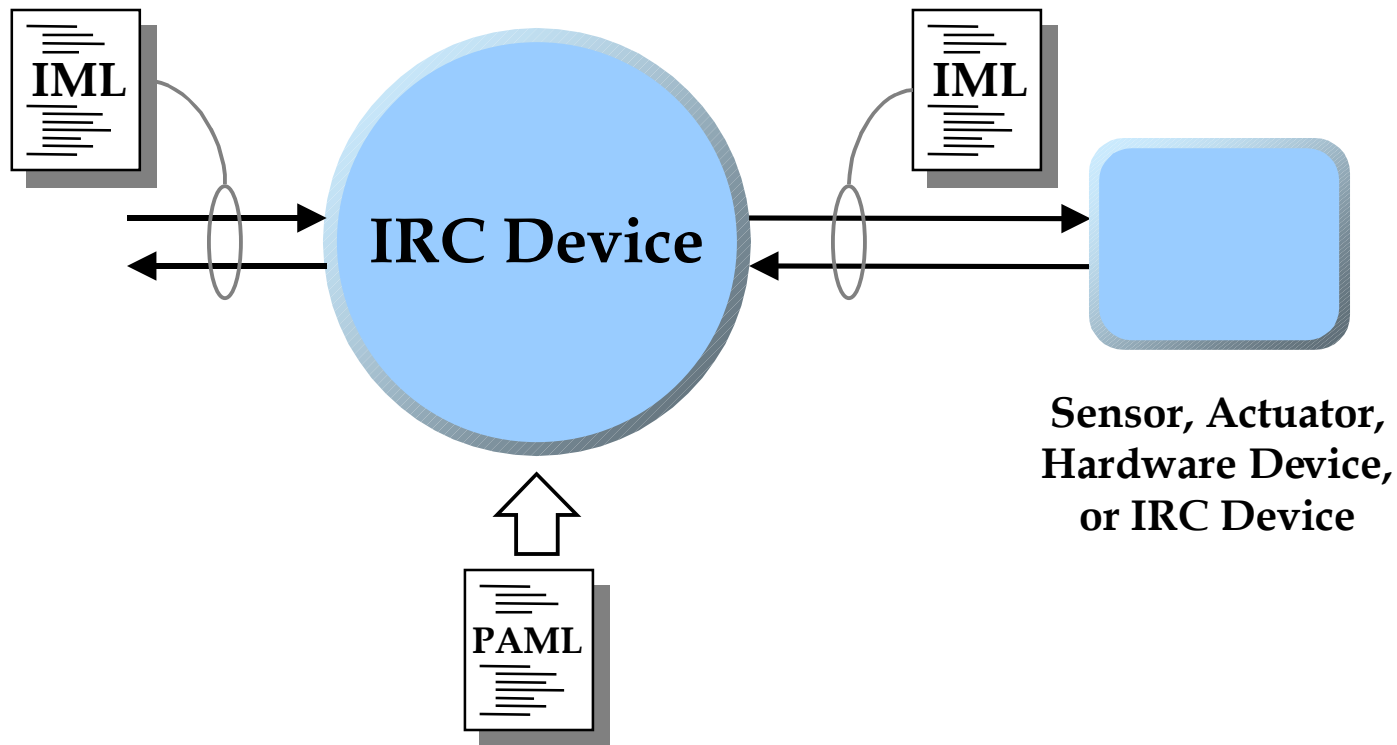


HAWC Device External Interface Configuration



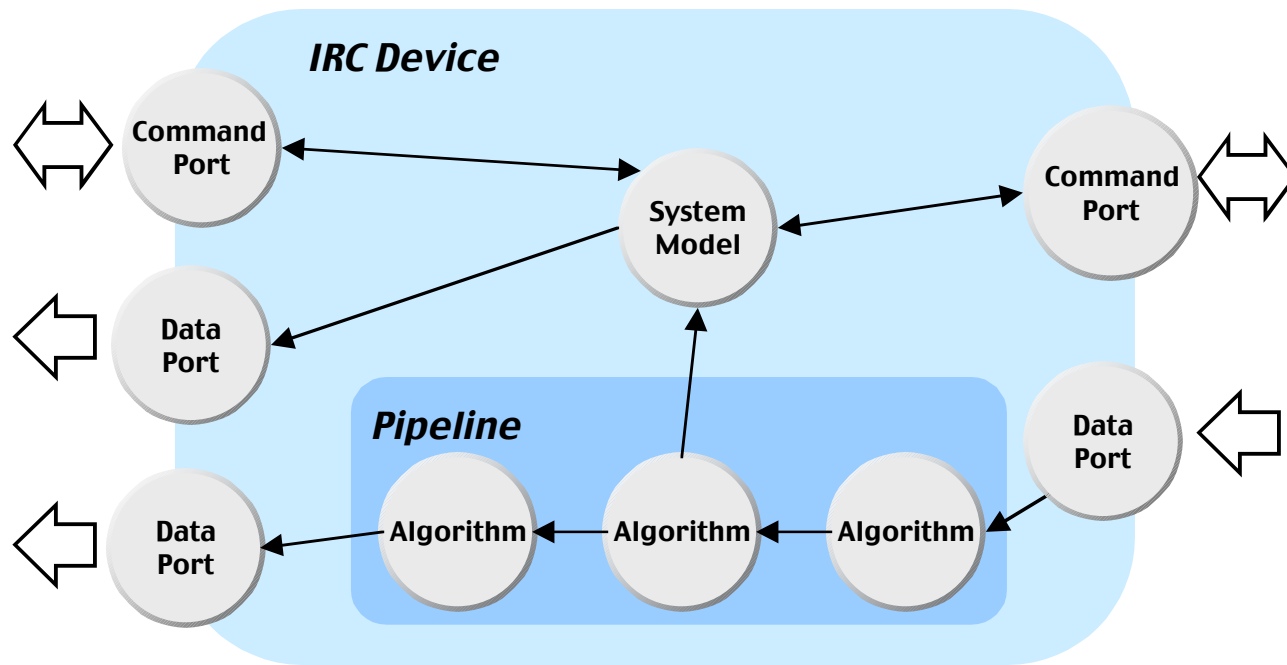
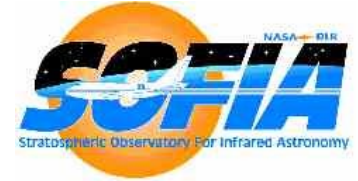
Public Interfaces

Private Interfaces





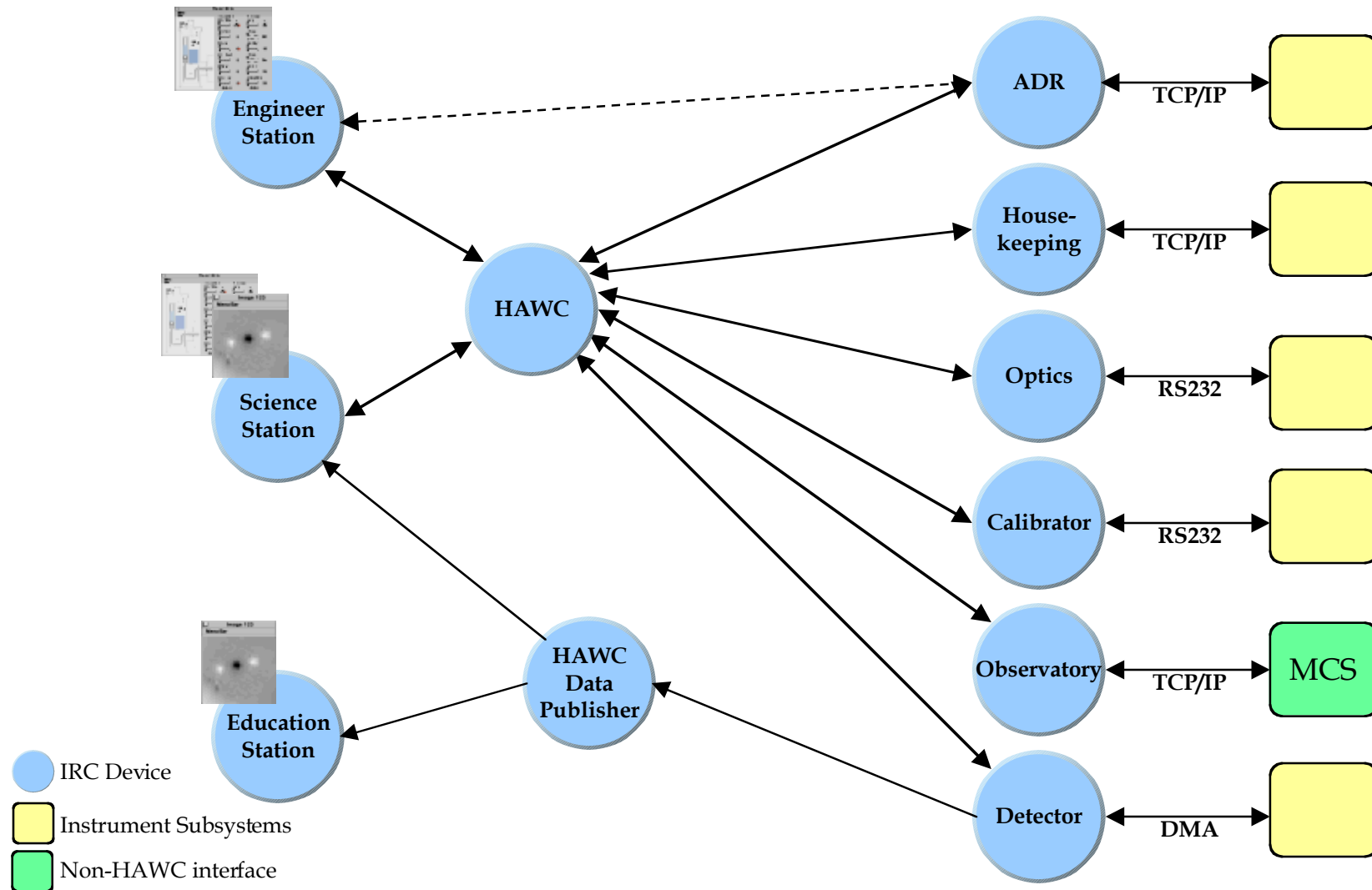
HAWC Device Internal Configuration (Example)



- This is an example internal configuration of an IRC device.
- Commands and responses typically flow through one port while the data flows through separate ports.
- An internal pipeline might provide data through one data port while the System Model might route interim data through a second data port.

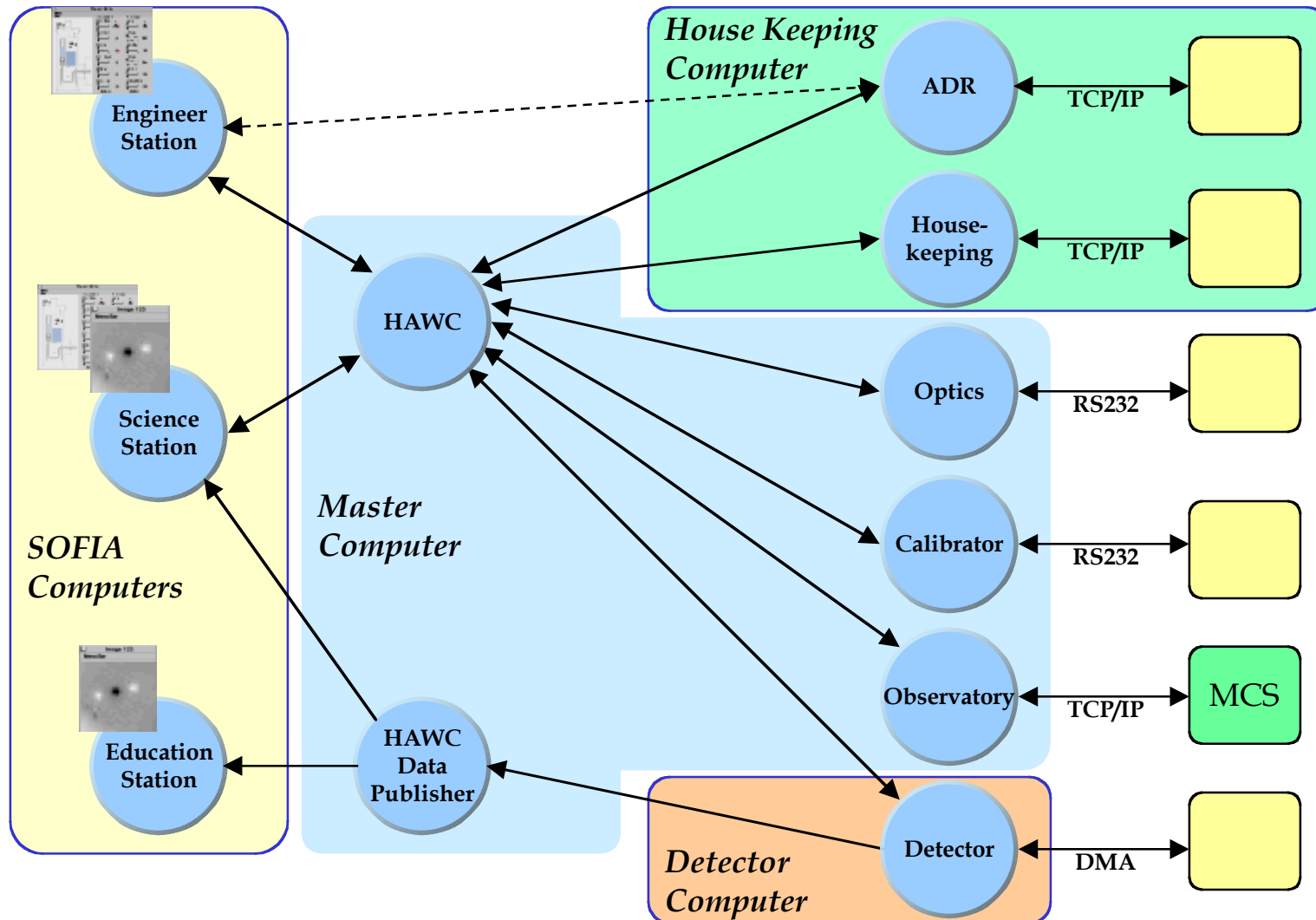


HAWC Device Design

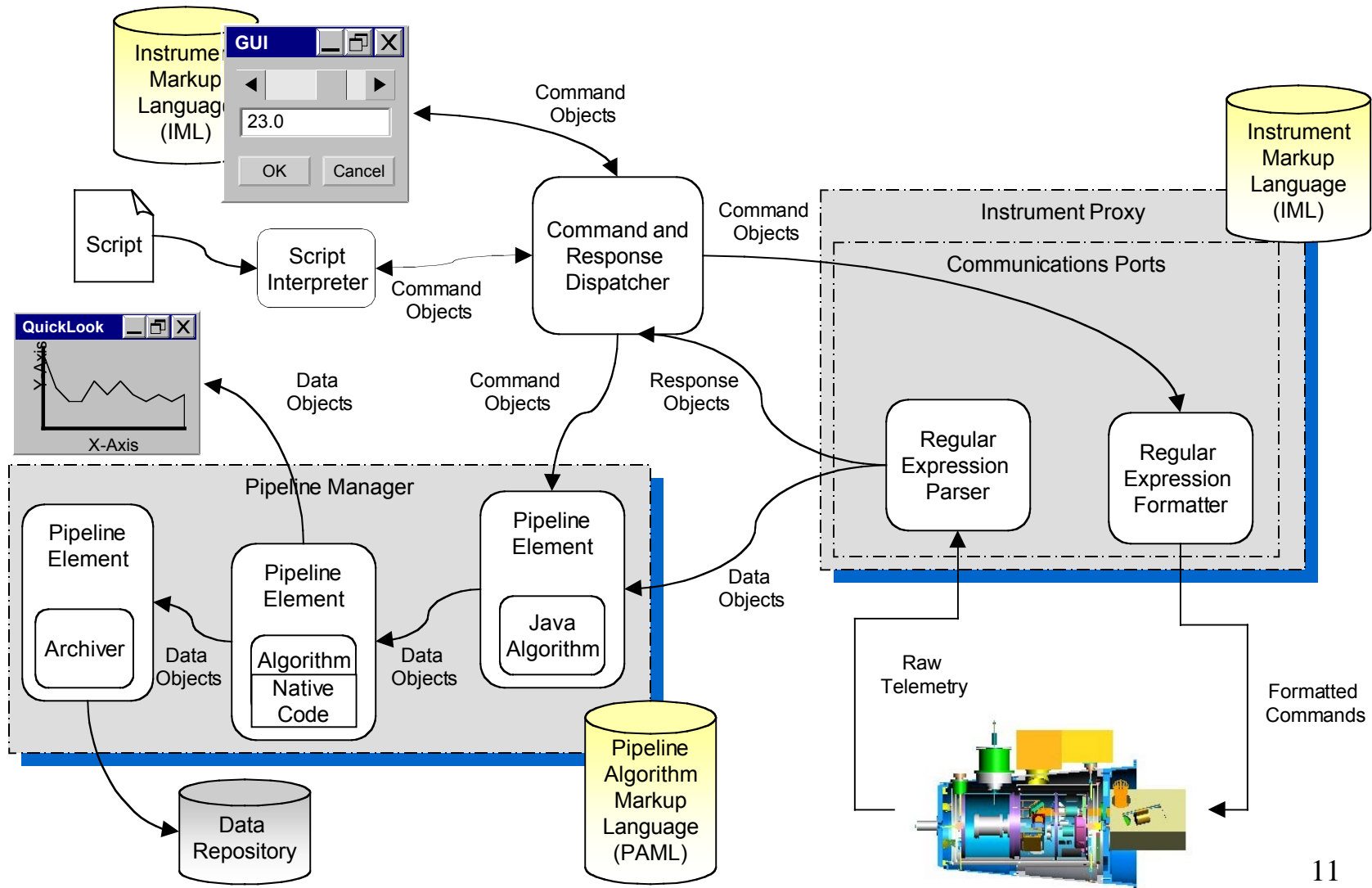




HAWC Devices Mapped to Computers

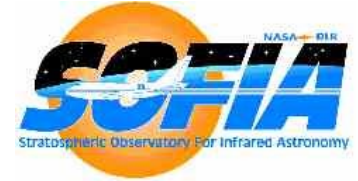


Detailed Overview of HAWC Device Internals- Example





Design Components



- **Descriptors package**
 - Hide the XML / IML details from the rest of the application.
 - Provide convenience functions to traverse the IML structures and retrieve information easily.
- **Command package**
 - Contains classes for dispatching and routing commands and their responses.
- **Formatting package**
 - Contains classes for taking data and command objects and formatting them for delivery to an instrument or another IRC device according to the rules specified in the IML.
- **Parsing package**
 - Contains classes for receiving data and command objects and parsing them according to the formatting rules specified in the IML.



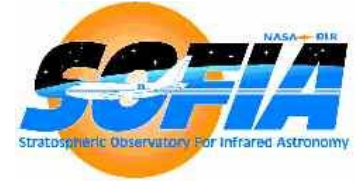
Design Components (cont.)



-
- **Pipeline package**
 - Manages the data analysis pipeline – the place where data manipulation algorithms are constructed and executed.
 - **Algorithm package**
 - Contains classes that implement the data manipulation algorithms.
 - Transforms the data which is then provided to the visualizations and/or archiving pieces of the system.
 - **GUI package**
 - Contains classes for graphical user interfaces
 - **Visualization package**
 - Contains classes for visualizing data objects produced by the data analysis pipeline.



Graphical User Interface



- **Used to control and monitor the instrument**
- **Data visualizations**
 - User can easily see the data and analyze results of commands that are issued or algorithm properties that are changed.
- **Consistent “property” editing throughout the application**
 - Allows the user to learn how to change command arguments, algorithm or visualization properties once, then reuse that skill throughout the interface.



Graphical User Interface - Example



Instrument Remote Control

File Edit View Tools Visualizations Help

C S M T 0 90 180 270 + - [Color Buttons]

Observing Modes Scans

Visualizations Pipeline

Secondary Mirror Setup

Property	Value	Units
Chop amplitude	0	arcsec
Chop angle	20	degrees
Chop mode	external	ft
Chop frequency	4	Hz
Offset	0	arcsec

Apply Changes Revert Changes

Integration Setup

Property	Value	Units
Chop cycles per frame	4	
Integration time per nod	10	sec
Number of nod pairs	20	
Start Position	00 00 00.00, 11 11 11.11	RA, DAC

Nods

Complete: 4
Remaining: 55

Stop Run Pause

S & N Dither Mosaic

MyPlotAdapter2d

File Edit Tools

ADC 0 0, DAC 0 0 vs. TIME

AnotherPlotAdapter2d

File Edit Tools

ADC 0 0, DAC 0 0 vs. TIME

Start Time: Tue Jun 13 11:25:35.598160 EDT 2000



Science User Interface Example



Instrument Remote Control

File Edit View Tools Visualizations

Status Observing Instr Telescope

Instrument

- Cryogenics ●
- Temperatures ●
- Cold Shield ● 4 K
- Detector ■ 0
- Computers ●
- Housekeeping ●
- Detector ●
- Master ●

Telescope

- Temperatures ●
- Cavity ● 30 K
- Primary ● 27 K

Aircraft

-
-
-
-
-
-
-
-
-

Observation

Nods
Complete: 4
Remaining: 55

Secondary Mirror

Property	Value	Units
Chop amplitude		arcse
Chop angle		degrees
Chop mode		ft
Chop frequency		Hz
Offset		arcse

Apply Changes Revert Changes

Integration

Property	Value	Units
Chop cycles per frame		
Integration time per nod		sec
Number of nod pairs		
Start Position	00 00 00.00, 1	RA, DAC

Nods
Complete: 4
Remaining: 55

Image

Pointing

File Edit Image Options Apertures Help

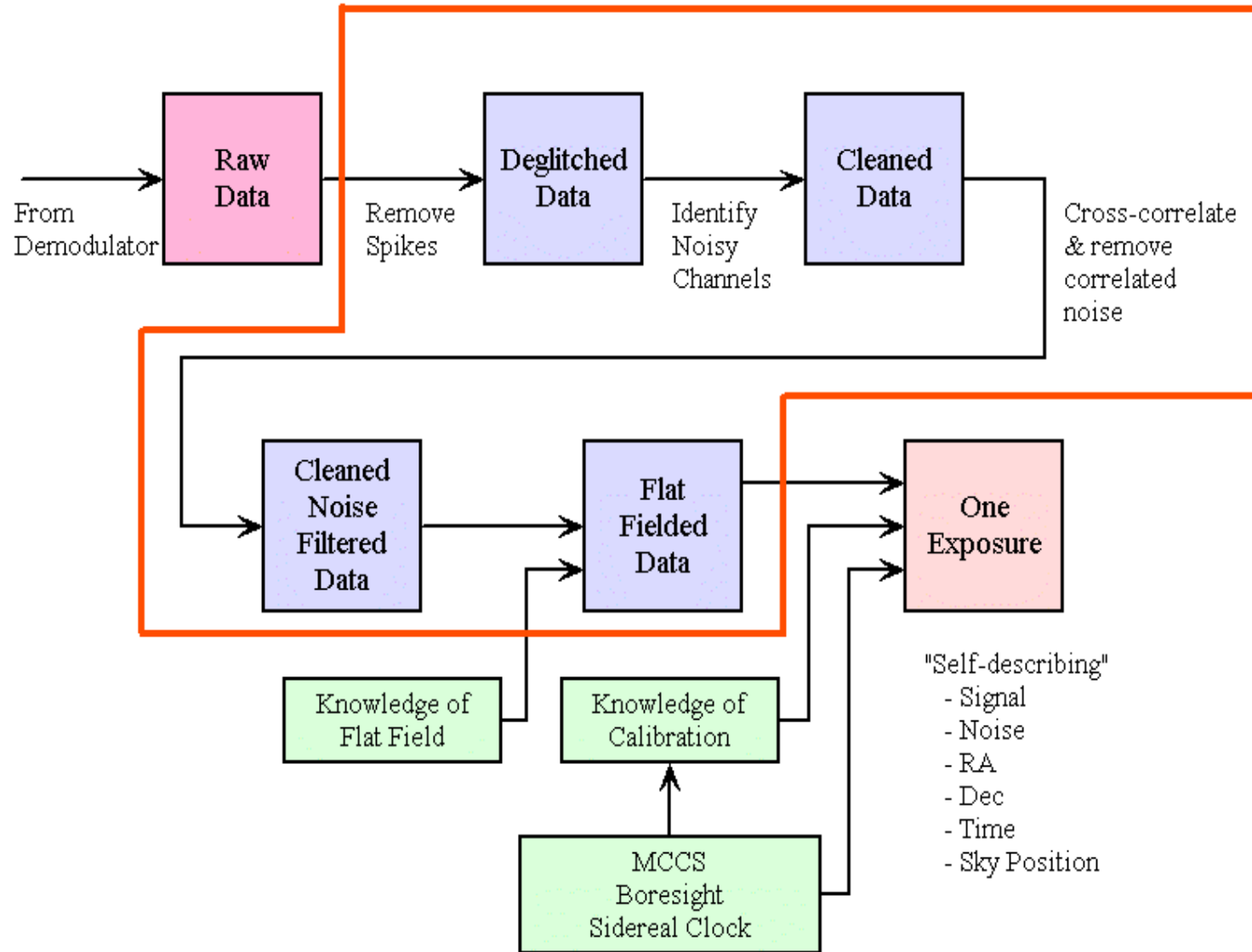
Mosaic Image

File Edit Image Options Apertures Help

Help



Data Pipeline Conceptual Model





Summary of Analyses Performed



- **Evaluated the ability to provide algorithms in IDL (Interactive Data Language) procedures.**
 - Built a prototype which uses JNI (Java Native Interface) to call C programs.
 - Proves that IDL algorithms can be used.
 - Also provides a framework for algorithms in other languages such as C.
 - More analysis of performance of IDL needs to be completed
- **Evaluated the use of Jini to perform device lookups.**
 - Jini, an add-on Java library, supports delivery of services over a network through dynamic discovery mechanisms.
 - Determined that this will make the system more extensible and easier to maintain in the future, but is not a requirement for the near future.



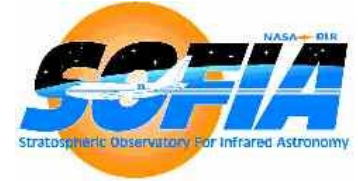
Prototype Testing and Risk Mitigation



- **Risk – Because of performance concerns of IDL, it is possible we may have to translate all algorithms into Java. This is a minor schedule risk.**
 - Mitigation: Prototype built using IDL. Simple IDL algorithm was implemented and run using existing prototype software. Need to test the performance more closely.
 - Conclusion: Barring severe performance problems, some algorithms can be written in IDL rather than translating everything into Java.
- **Risk – Originally, the maturity of Java was a risk. Although we feel that risk is minimized, there still is a risk that it will not be sufficiently platform independent.**
 - Mitigation: Continue to monitor Java progress on the Mac.
 - Conclusion: It would be less risky to go with a Solaris and NT architecture while the Mac support for Java matures.



Test Process



- **Software test criteria will be provided to University of Chicago (UC) for review prior to testing.**
- **Initial software tests will first be performed at the development facility using instrument simulators.**
 - Early testing with simulators provides the ability to catch software errors internal to IRC itself.
- **Early integration with subsystems necessary as soon as possible.**
 - Successfully tested with the HAWC Housekeeping subsystem in November 2000.
 - Issues identified and retesting will occur when fixes are provided.
 - Successfully tested with the MCS periodically from Sept. on
 - Ready to test with the optics subsystem in December.
 - Weekly telecons with UC, GSFC, and IRC development staff to discuss subsystem and software development and testing status.
 - UC reviews test results either in person or through test reports.



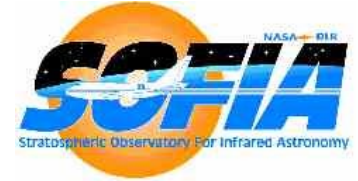
Test Plan



- **Tests have been divided into several categories:**
 - Installation Testing - This testing will verify that the software can be installed according to instructions without errors.
 - Configuration Testing - This testing will verify that the software will work with multiple operating systems (Windows, Solaris, Linux, MacOS).
 - Error Conditions Testing – This testing will verify that the software “behaves gracefully” when an error condition is introduced.
 - Performance Testing - This testing will verify that the software performance is within parameters defined by the instrument requirements.
 - Recovery Testing - This testing will verify that the software can recover from catastrophic failure and still function normally.



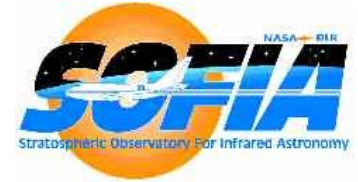
Test Plan (cont.)



-
- **Test Categories (cont.)**
 - Reliability Testing - This testing will verify that the software will continue to function normally over days of continuous use.
 - Stress Testing - This testing will verify that the software can function normally even under an extreme load (beyond the performance requirements).
 - Functionality Testing – This testing will verify that the functional requirements of the software work as designed.
 - **Usability Testing - This testing will verify that the user interface is intelligently designed and makes sense.**
 - This will be treated as separate from the software test plan as it will be performed by usability experts in conjunction with users.



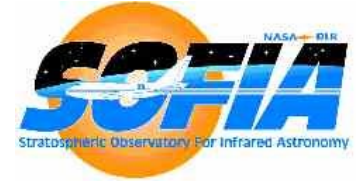
HAWC Software Schedule



Test / Function	Status	Due Date
IRC – Opto Integration Test	In Progress	March 30, 2001
IRC – Housekeeping Integration Test	In Progress	March 30, 2001
IRC – ADR Integration Test	Not Started	March 30, 2001
IRC – Detector Integration Test		
Using Simulator	Not Started	April 30, 2001
Using Prototype Instrument	Not Started	June 30, 2001
Using Complete Instrument	Not Started	det. availability +1 mo.



Open Issues



-
- **ICDs between IRC software and subsystem software not final**
 - **GUI decisions need to be finalized**
 - Some specific visualizations need to be designed more fully.