101-A0055
Infrastructure for Autonomous Instrument Reconfiguration

Michael Deck, Cleanroom Software Engineering, Inc.
Brad Hines, Jet Propulsion Laboratory
July 15, 2003

CleanSoft®

Overview

- Proposal to NRA 99-OES-08:
  - Demonstrate three distinct earth observing technologies for space instrumentation in a single integrated testbed platform
- Capabilities delivered
  - Hard realtime distributed multiprocessing systems
  - Dynamic reconfiguration
  - Autonomy

What is Dynamic Reconfiguration with Autonomy?

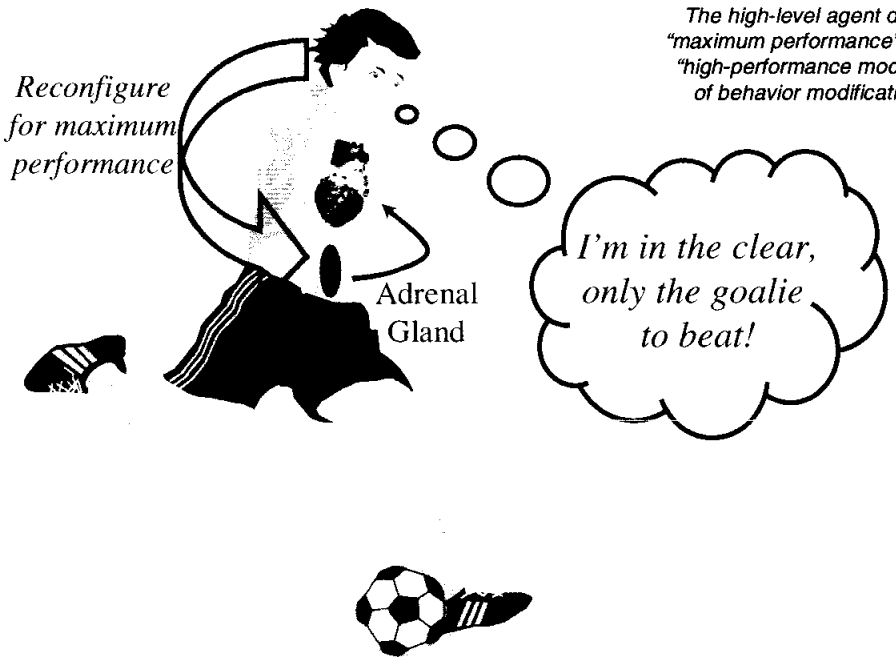
- High-Level Control of Low-Level Functions

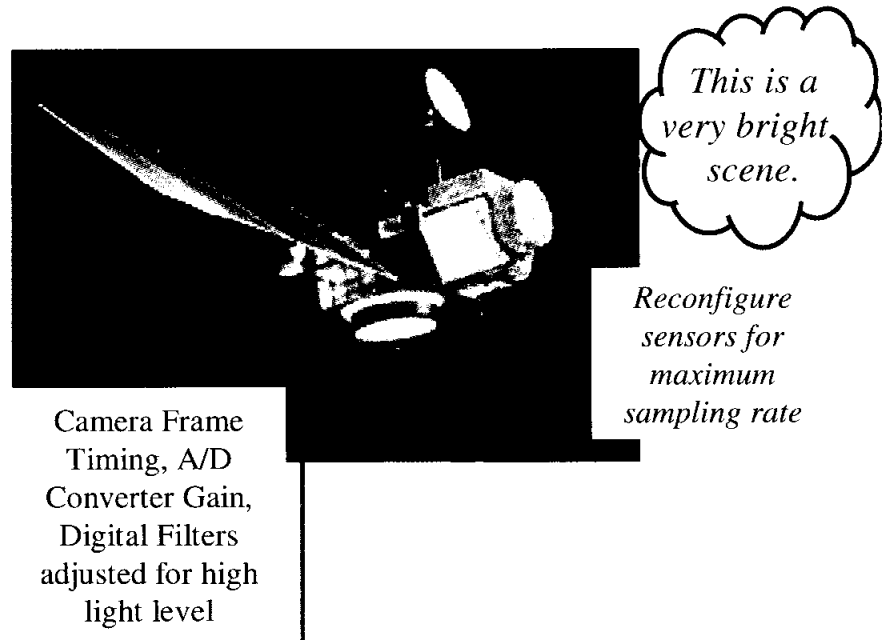
Autonomous Decision-Making

Configuration

Configurable Hard Realtime Components

The high-level agent does not understand what "maximum performance" means. It simply requests "high-performance mode", causing a complex set of behavior modifications at the lowest levels.

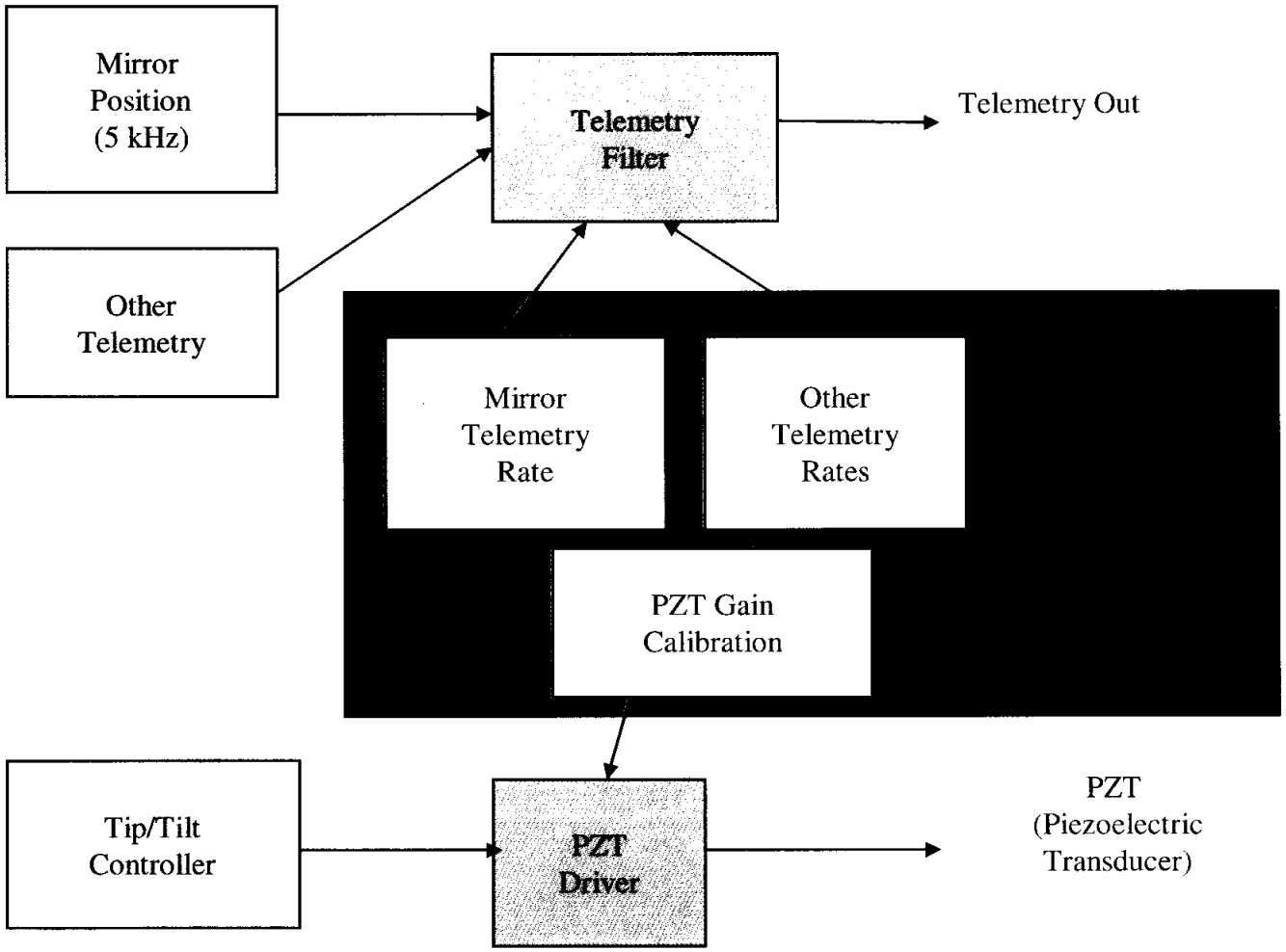




Technology Components

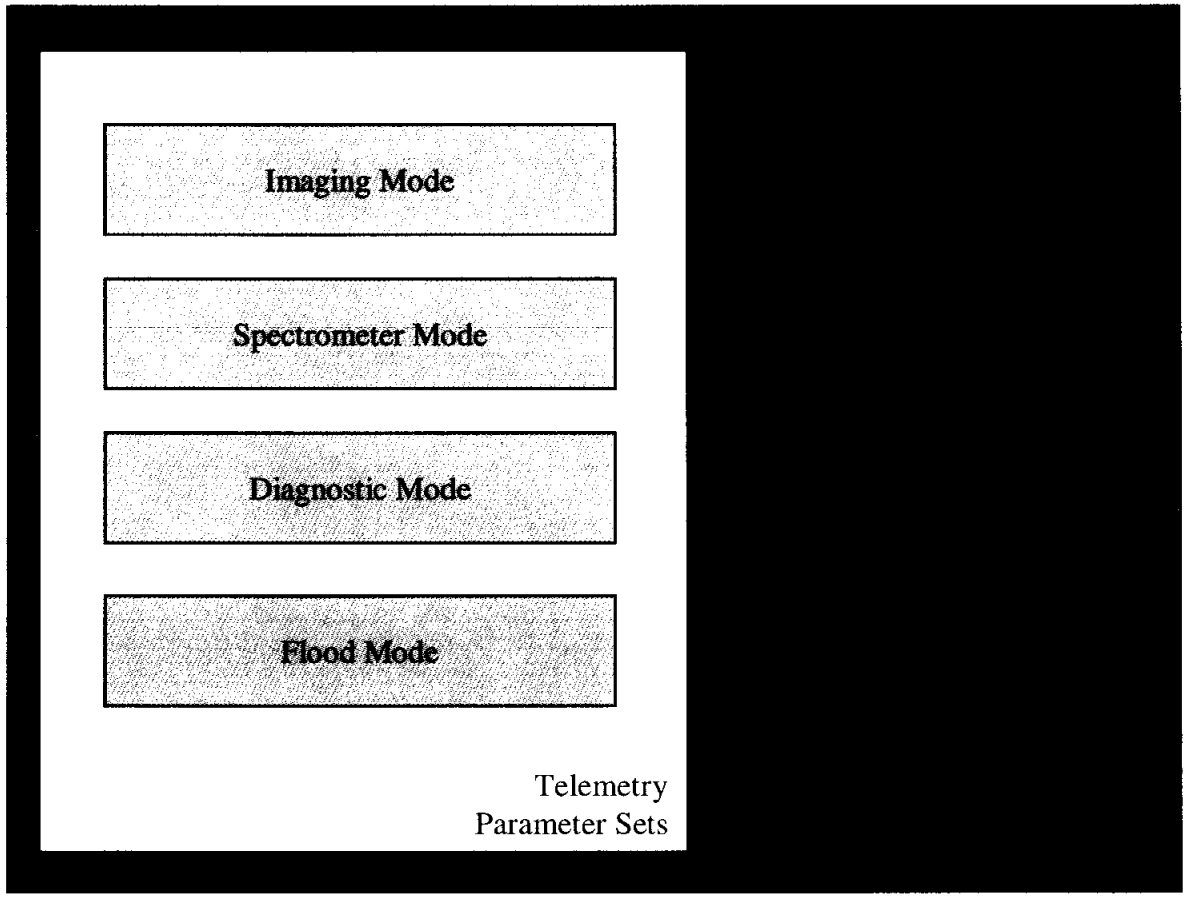
- Configuration
  - Make significant enhancements to nascent configuration technology from JPL's interferometry program
  - Significant portions of this component are new technology
- Autonomy
  - Use existing autonomy techonology – Ames/JPL Remote Agent
- Hard Realtime Performance
  - Use existing technology developed in JPL's interferometry program
- Integration
  - The *biggest* technology breakthrough is coming from the integration of these three components to achieve a combination of high performance *together with* high function.

Component Technology: Configuration



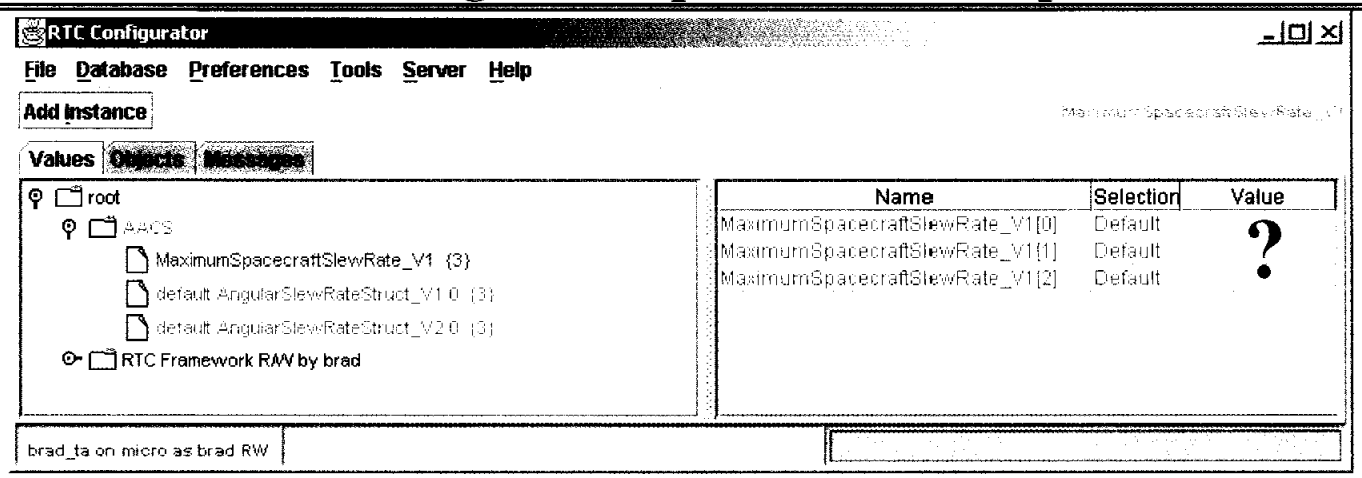
Configuration/Parameter Sets

- Parameter sets allow the assignment of a name to groups of parameters
- Parameter sets and parameter versioning (next slide) are new capabilities to be developed under AIST.

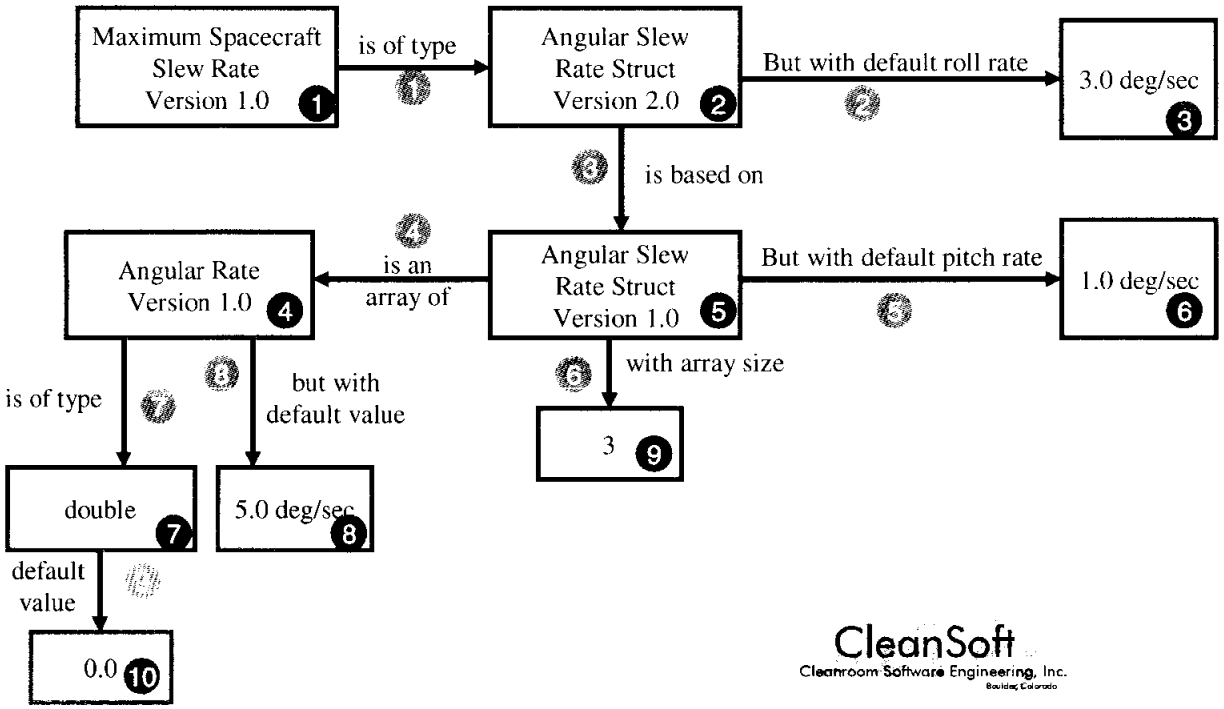


Infrastructure for Autonomous Instrument Reconfiguration

Configuration Component Technology: Modeling of Complex Relationships – AACS Example



What are the roll, pitch, and yaw rates in "Maximum Spacecraft Slew Rate"?



i	ID1	ID2	Type
1	1	2	Is of type
2	2	3	Has default value
3	2	5	Is based on
4	5	4	Is array of
5	5	6	Has default value
6	5	9	Has array size
7	4	7	Is of type
8	4	8	Has default value
9	7	10	Has default value

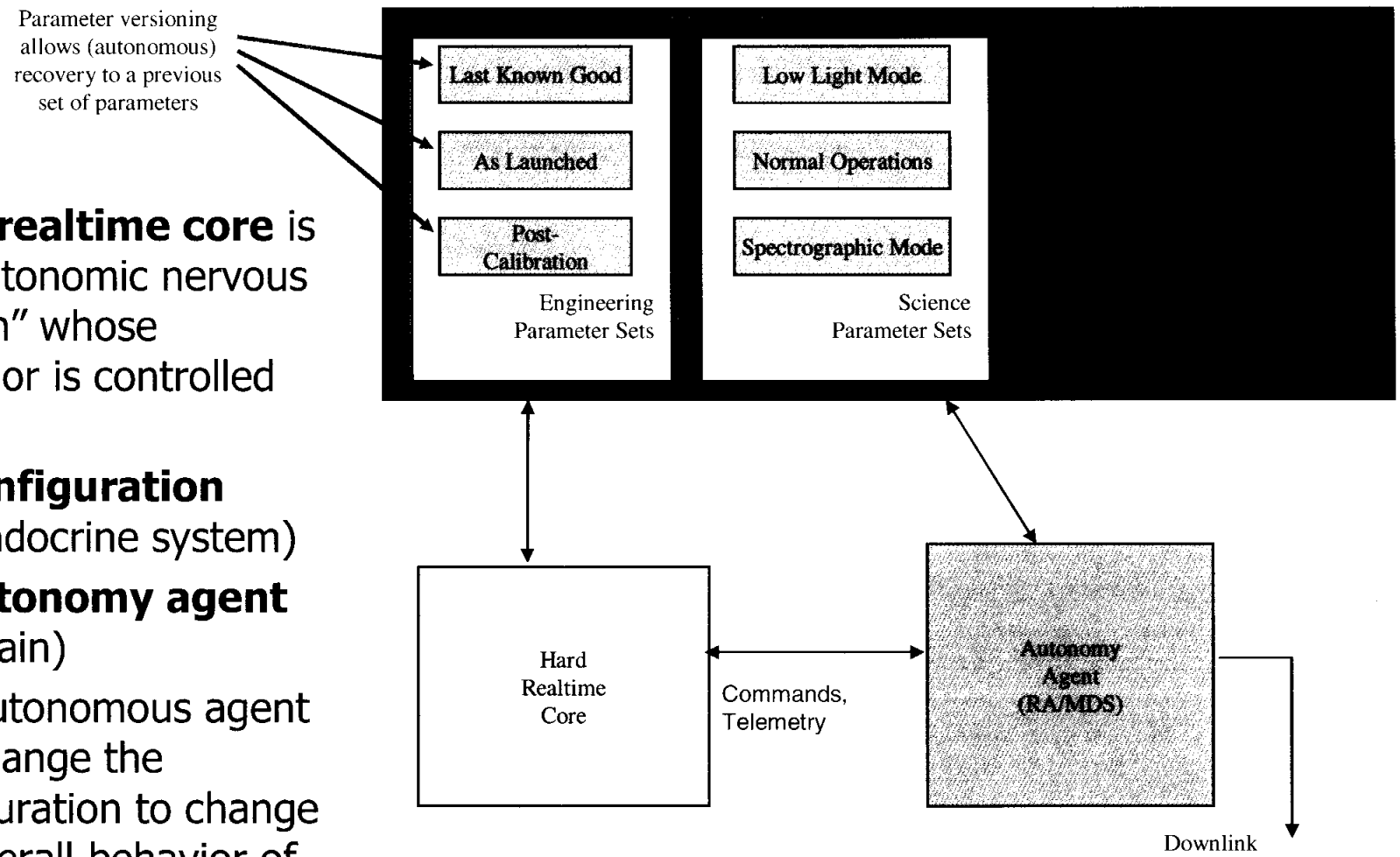
CleanSoft
Cleanroom Software Engineering, Inc.
Boulder, Colorado

Autonomy Component

- Problem: Advanced autonomous agents are traditionally multi-layered with each layer using different technology with different semantics. This increases software integration and validation costs.
- Objectives:
  - Simplify an autonomous agent architecture by using the same execution machinery (semantics and implementation) at any layer
  - Simplify the integration of autonomous agent software by using a single communication protocol between all layers
- Key Innovation:
  - IDEA virtual machine implementing a sense-plan-act loop with real-time guarantees
  - Use of a reactive model-based planner in the inner loop of execution, with each IDEA agent possibly using a different planner technology provided it is compatible with the IDEA virtual machine
  - IDEA communication protocol consisting of the interchange of task networks

Fully Integrated Capability

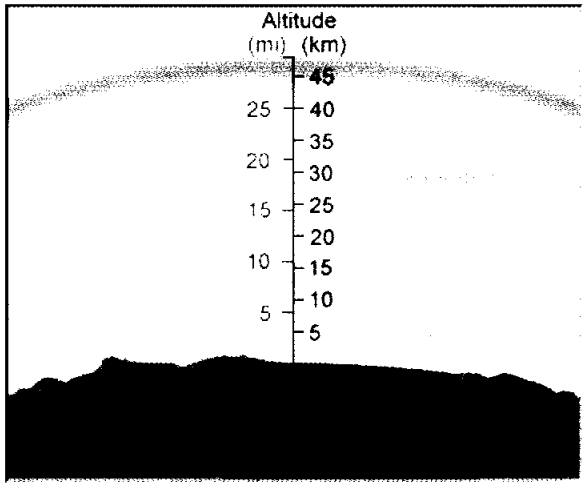
- **Hard realtime core** is an "autonomic nervous system" whose behavior is controlled by the
  - **configuration** (endocrine system)
  - **autonomy agent** (brain)
- The autonomous agent can change the configuration to change the overall behavior of the "animal".



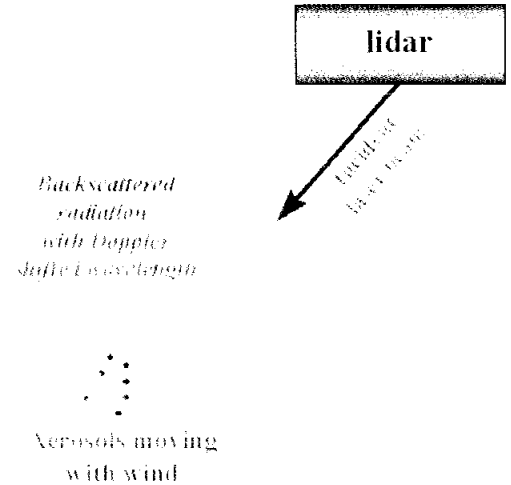
Focus Missions

- Atmospheric Chemistry:
  Solar Occultation Fourier Transform Infrared Spectrometer

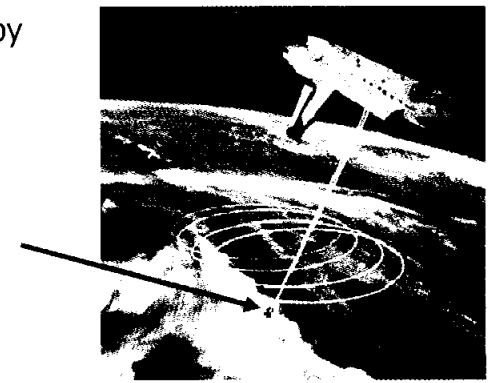
- "Targeted Goal" Activity
  - Specific Aim: Reaching into the Troposphere



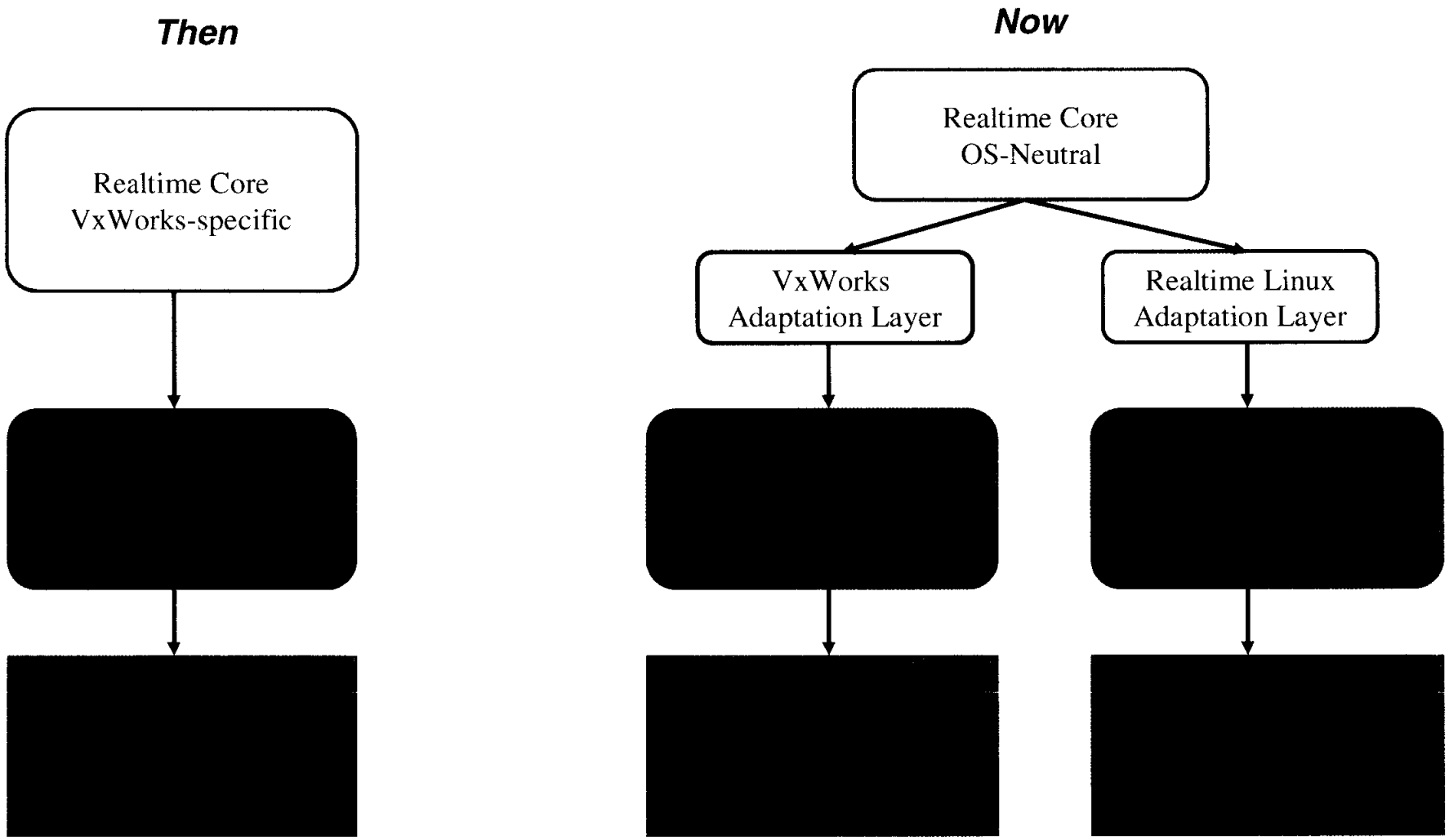
- Particulates/
  Wind Speed
  Measurement:
  Lidar



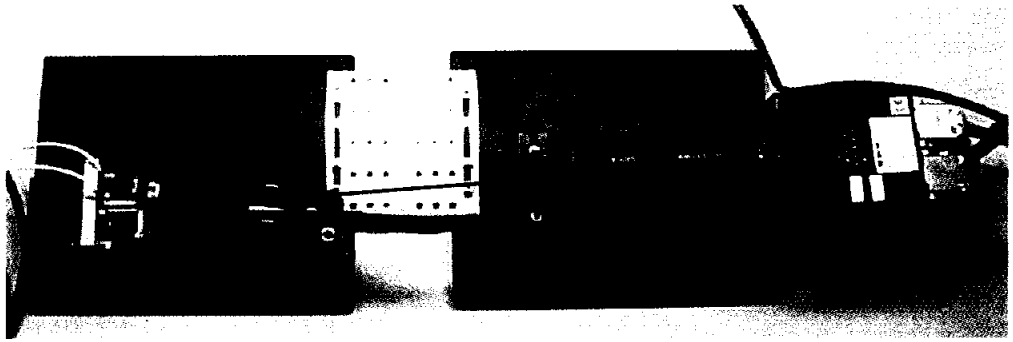
- "Shoot for the
  - Straightforward Goal: Science data quality improvement via active doppler tracking to improve SNR
  - Advanced Goal: Robust operation in the presence of cloud cover by finding holes



Low Cost of Entry Realtime System

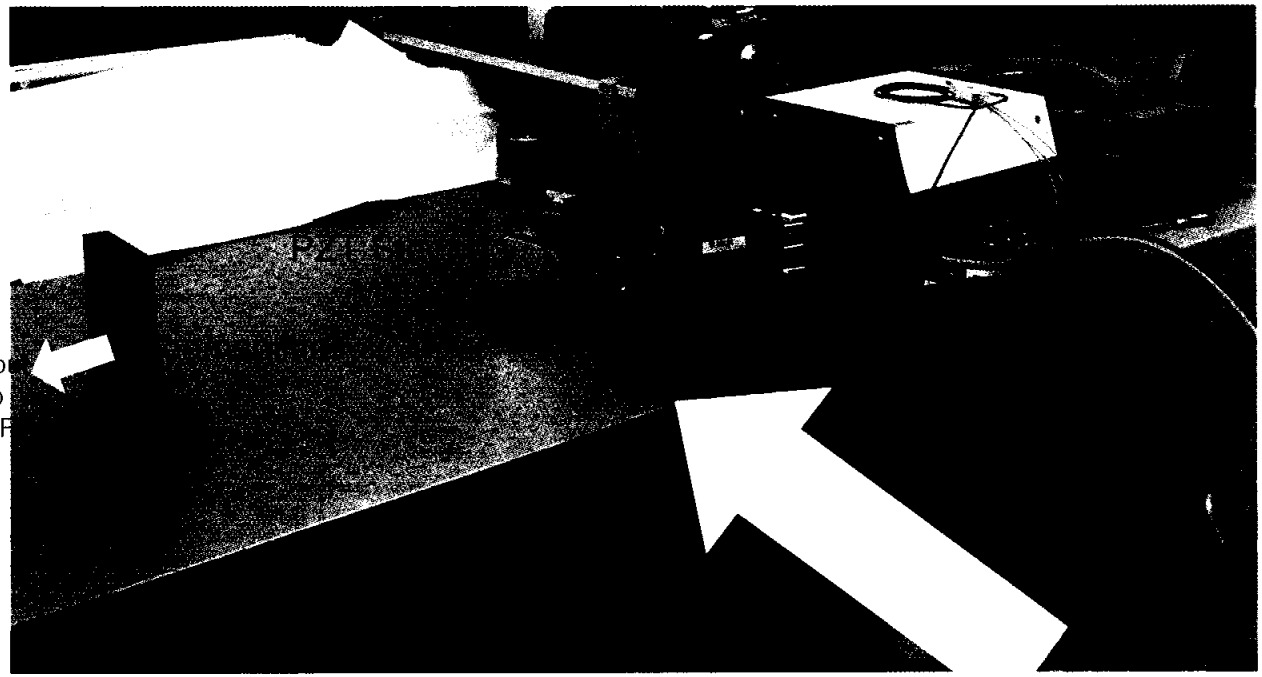


Deployment



- Deployment in a benchtop demo

- Deployment in the MkIV FTIR lab



Solar Beam
From South Wall

RealTime Core (RTCore) Provides

- A set of building blocks to develop
  - Real-time control components
  - Non-real-time support applications
- A run-time environment on which to load and run user-developed software components
  - CPU Manager CORBA application
  - Real-time control objects
    - e.g. Motion control, Data acquisition
  - Real-time services
    - To replace or enhance "built-in" services
- Middleware applications
  - Glue client applications to real-time control environment

Middleware, CORBA, and All That

- Middleware adapts interfaces between different systems
- Examples of uses for middleware
  - A Java user interface that wants to talk to a database
  - A realtime computer that wants to send data to an archive
  - A command-line application that wants to tell a realtime computer to "calibrate the camera"
- Desire to have these disparate systems speak a common language.
  - We selected CORBA for the communications interface
    - Most important feature is a language-neutral Interface Description
    - Also is machine-neutral
  - Other middleware is involved as well
    - JDBC database connectivity package for Java
    - Our own middleware applications
- Middleware helps developers to get their job done faster

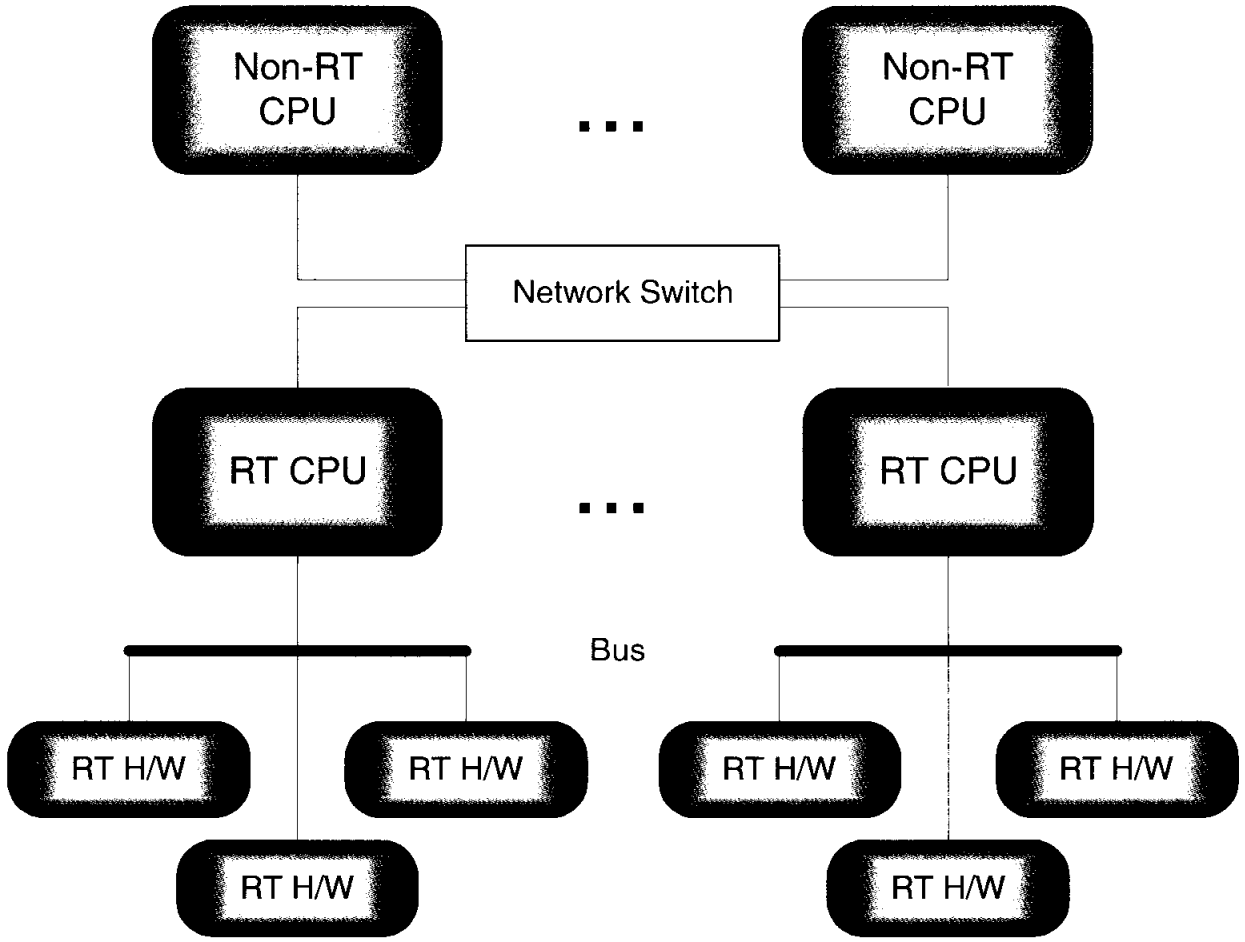
RTCore Motivation

- Commonality
  - Encapsulate common functionality into an object oriented framework
  - Reduce total code base
  - Easier to understand, maintain and deploy
- Generality
  - Not dependent on a particular target application, operating system, or computer
- Adaptability
  - Enable customers to incorporate new services and control features without having to change RTC Core
- Performance
  - Maintain real-time performance where critical
  - Make intelligent trades in performance to achieve other goals

Physical Architecture

- Real-time control hardware
  - Electronics and devices to control physical elements
  - E.g. motors, D/A converters, metrology sensors etc.
- Real-time CPUs
  - Host software to control RT control hardware
  - Connected to RT hardware by an electronic bus
    - E.g. PCI, VME, etc.
- Non Real-time CPUs
  - Host support software
    - No hard real-time requirements
    - E.g GUIs, data archivers, sequencers, other middleware servers
  - Connected to RT CPUs and each other via network

Physical Architecture cont.



Software Architecture

- Interfaces
  - Electronic
    - Memory mapped I/O
  - Internal
    - Language/platform specific APIs (e.g. C++ classes etc.)
    - Locally constrained communication between objects in a single process
  - CORBA
    - Language/platform independent
    - Distributed communication between objects on multiple processes and/or CPUs

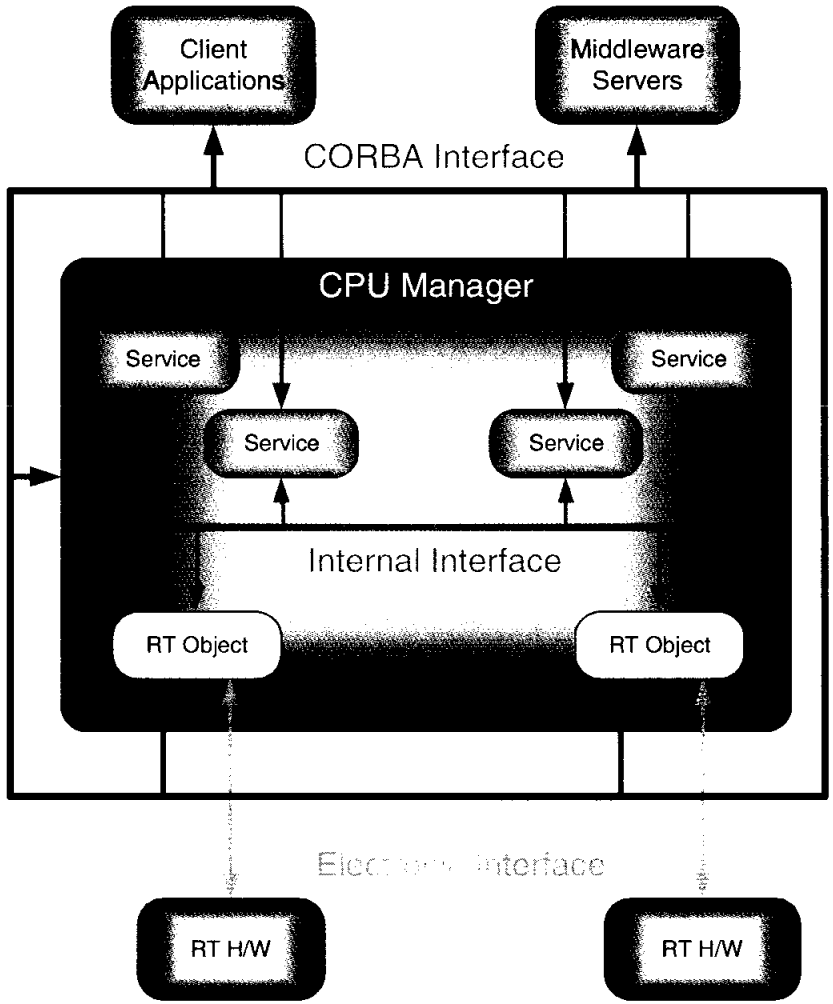
Software Architecture cont

- Real-time Components
  - CPU Manager
    - Application providing run-time environment for real-time software components
    - 1 CPU manager / process
  - RT Objects
    - Encapsulate control of a hardware component (i.e. a functional set of sensors/actuators)
  - Services
    - Provide services that can be used by any RT Object directly through internal interfaces.
    - May have a CORBA interface that allows distributed control/setup of the service by client applications

Software Architecture cont

- Non Real-time Components
  - Middleware Servers
    - Infrastructure applications
    - E.g. CORBA Name Service, Telemetry Server etc.
  - Client Applications
    - End user applications for manipulating RT Objects and Services
    - E.g. GUIs, telemetry archivers, etc.
  - Database Servers
    - In Testbed/support environment provide capability for handling large number of versioned parameters (10's of thousands)
    - In flight, could use solid-state database or simple table
    - Interfaces designed for either approach
    - Ongoing research into synchronization issues

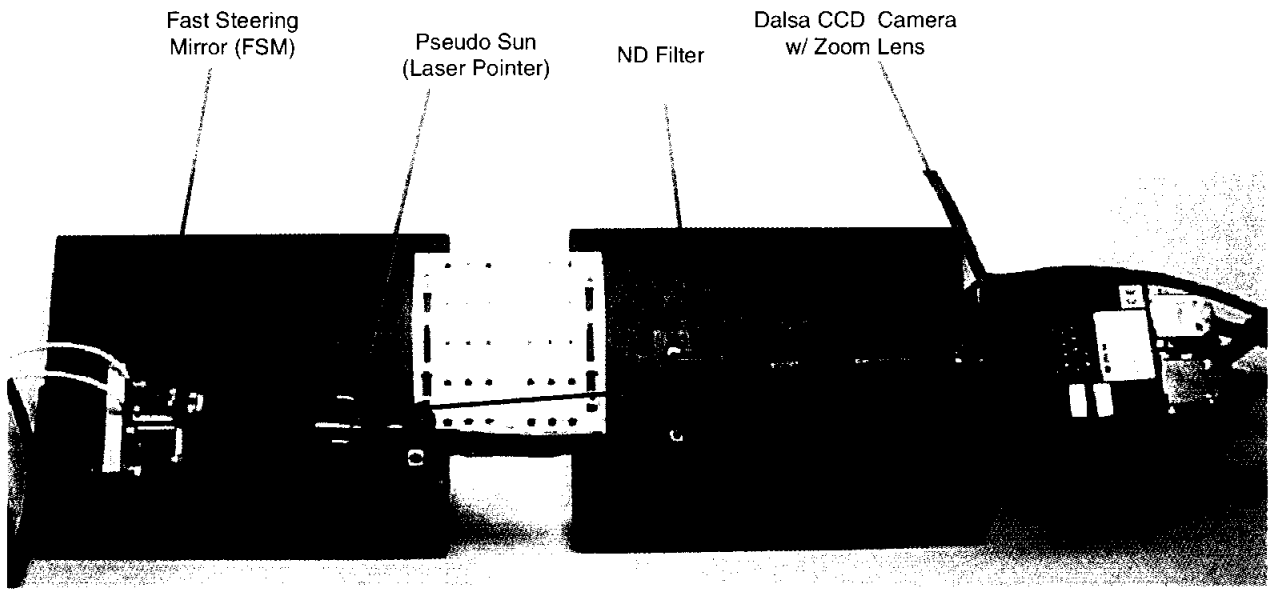
Software Architecture cont.

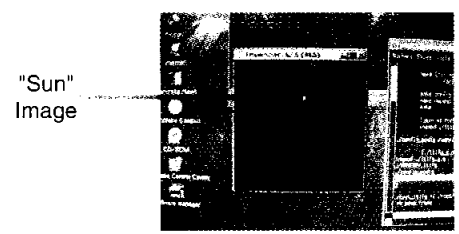


Demonstration - Simulated Sun Tracking

"Solar Gymnastics"

Simulated Sun Tracking using a Laser Spot

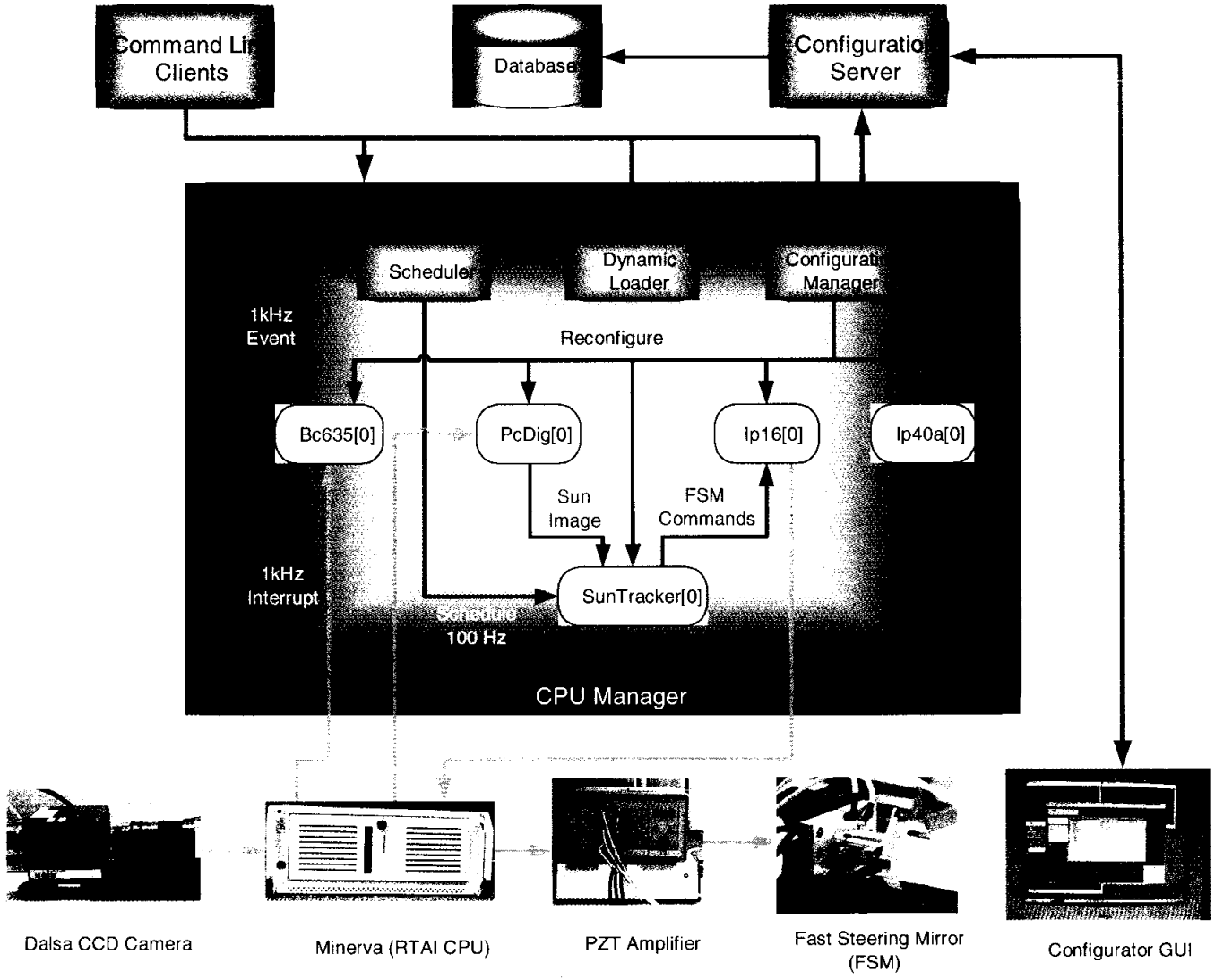




Demonstration of tip-tilt centroid tracking using a fast steering mirror

Demonstration of rapid reconfiguration

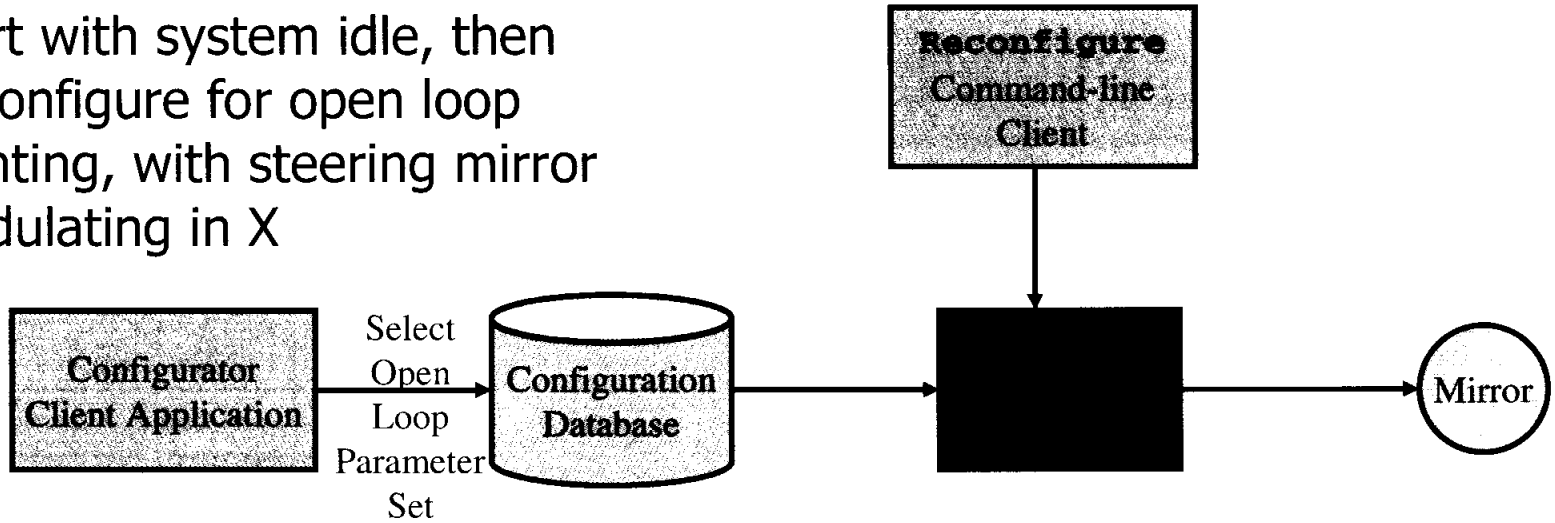
Sun Tracker Demo



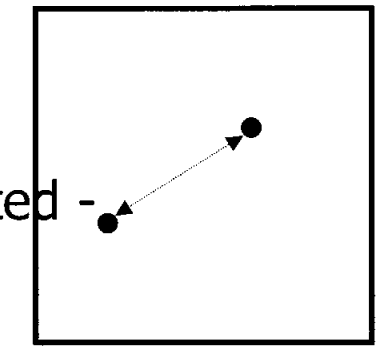
CleanSoft
Cleanroom Software Engineering, Inc.
Boulder Colorado

Sun Tracker Demonstration

- Start with system idle, then Reconfigure for open loop pointing, with steering mirror modulating in X



- We see that the mirror is uncalibrated - it points with a tilt



Infrastructure for Autonomous Instrument Reconfiguration

Configurator Client Application

Instances of Workspace rich in minerva_aistdemo on themule as deckm on aist/aistdemo

Repository Window Tools Help

Workspace Edit Tools

rich

Module Edit Help

AIST::Demo
Intrinsic_Types
RTC::Bc635
RTC::Hardware
RTC::Ip16
RTC::Ip230
RTC::Ip330
RTC::Ip40a
RTC::Ip480
RTC::IpModule
RTC::IpQuad
RTC::PcDig
RTC::Tasks
SunTrackerDemo

SunTrackerDemo.SunTracker[0]

- root
  - Bc635[0]
  - Ip16[0]
  - Ip40a[0]
  - PcDig[0]
  - Scheduler[0][0]
  - SunTracker[0]
    - Angle
    - DarkCalibrate
    - Fsm
      - A
        - BoardName
        - BoardType
        - ChannelNumber
        - ChannelType
      - B
      - C
      - PhysicalPerElectrical
      - TripodRadius
    - Mode
    - Offset
    - OpenLoopAmplitude
    - OpenLoopRate
    - Priority
    - Rate
    - SunSensor
    - TaskName
    - TrackWindowSize
    - XGain
    - YGain

Stop

Name	Selection	Value
Angle	Override	0.0
DarkCalibrate	Override	true
Fsm.A.BoardName	Override	0
Fsm.A.BoardType	Override	Ip16
Fsm.A.ChannelNumber	Default	0
Fsm.A.ChannelType	Override	D/A
Fsm.B.BoardName	Override	0
Fsm.B.BoardType	Override	Ip16
Fsm.B.ChannelNumber	Override	1
Fsm.B.ChannelType	Override	D/A
Fsm.C.BoardName	Override	0
Fsm.C.BoardType	Override	Ip16
Fsm.C.ChannelNumber	Override	2
Fsm.C.ChannelType	Override	D/A
Fsm.PhysicalPerElectrical	Override	1.0
Fsm.TripodRadius	Override	0.018
Mode	Override	2
Offset	Default	0
OpenLoopAmplitude	Override	5.0
OpenLoopRate	Override	0.5
Priority	Override	10
Rate	Override	25.0
SunSensor.BoardName	Override	0
SunSensor.BoardType	Override	PcDig
SunSensor.ChannelNumber	Default	0
SunSensor.ChannelType	Override	IMAGE
TaskName	Override	SunTrackerTask
TrackWindowSize	Override	32
XGain	Override	0.1
YGain	Override	0.1

Instances

Name	Selection	Value
SunTrackerDemo.SunTracker[0].Angle	Override	0.0
SunTrackerDemo.SunTracker[0].Mode	Override	2
SunTrackerDemo.SunTracker[0].OpenLoopAmplitude	Override	5.0
SunTrackerDemo.SunTracker[0].OpenLoopRate	Override	0.5
SunTrackerDemo.SunTracker[0].Rate	Override	25.0
SunTrackerDemo.SunTracker[0].XGain	Override	0.1
SunTrackerDemo.SunTracker[0].YGain	Override	0.1

Task Name	Status

Stop

Configurator Client Application

SunTrackerDemo.SunTracker[0]

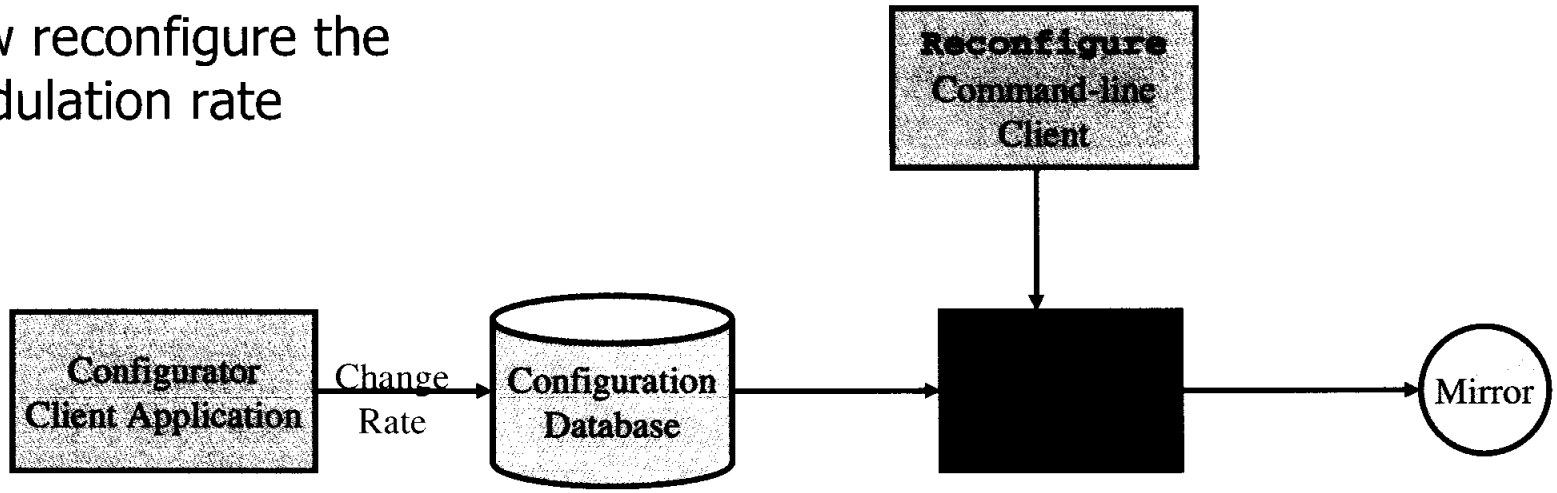
- root
  - Bc635[0]
  - Ip16[0]
  - Ip40a[0]
  - PcDig[0]
  - Scheduler[0][0]
  - SunTracker[0]
    - Angle
    - DarkCalibrate
    - Fsm
      - A
        - BoardName
        - BoardType
        - ChannelNumber
        - ChannelType
      - B
      - C
      - PhysicalPerElectrical
      - TripodRadius
    - Mode
    - Offset
    - OpenLoopAmplitude
    - OpenLoopRate
    - Priority
    - Rate
    - SunSensor
    - TaskName
    - TrackWindowSize
    - XGain
    - YGain

Stop

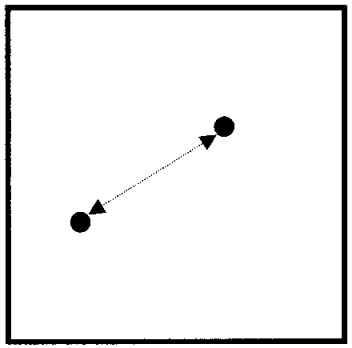
Name	Selection	Value
Angle	Override	0.0
DarkCalibrate	Override	true
Fsm.A.BoardName	Override	0
Fsm.A.BoardType	Override	Ip16
Fsm.A.ChannelNumber	Default	0
Fsm.A.ChannelType	Override	D/A
Fsm.B.BoardName	Override	0
Fsm.B.BoardType	Override	Ip16
Fsm.B.ChannelNumber	Override	1
Fsm.B.ChannelType	Override	D/A
Fsm.C.BoardName	Override	0
Fsm.C.BoardType	Override	Ip16
Fsm.C.ChannelNumber	Override	2
Fsm.C.ChannelType	Override	D/A
Fsm.PhysicalPerElectrical	Override	1.0
Fsm.TripodRadius	Override	0.018
Mode	Override	2
Offset	Default	0
OpenLoopAmplitude	Override	5.0
OpenLoopRate	Override	0.5
Priority	Override	10
Rate	Override	25.0
SunSensor.BoardName	Override	0
SunSensor.BoardType	Override	PcDig
SunSensor.ChannelNumber	Default	0
SunSensor.ChannelType	Override	IMAGE
TaskName	Override	SunTrackerTask
TrackWindowSize	Override	32
XGain	Override	0.1
YGain	Override	0.1

Sun Tracker Demonstration

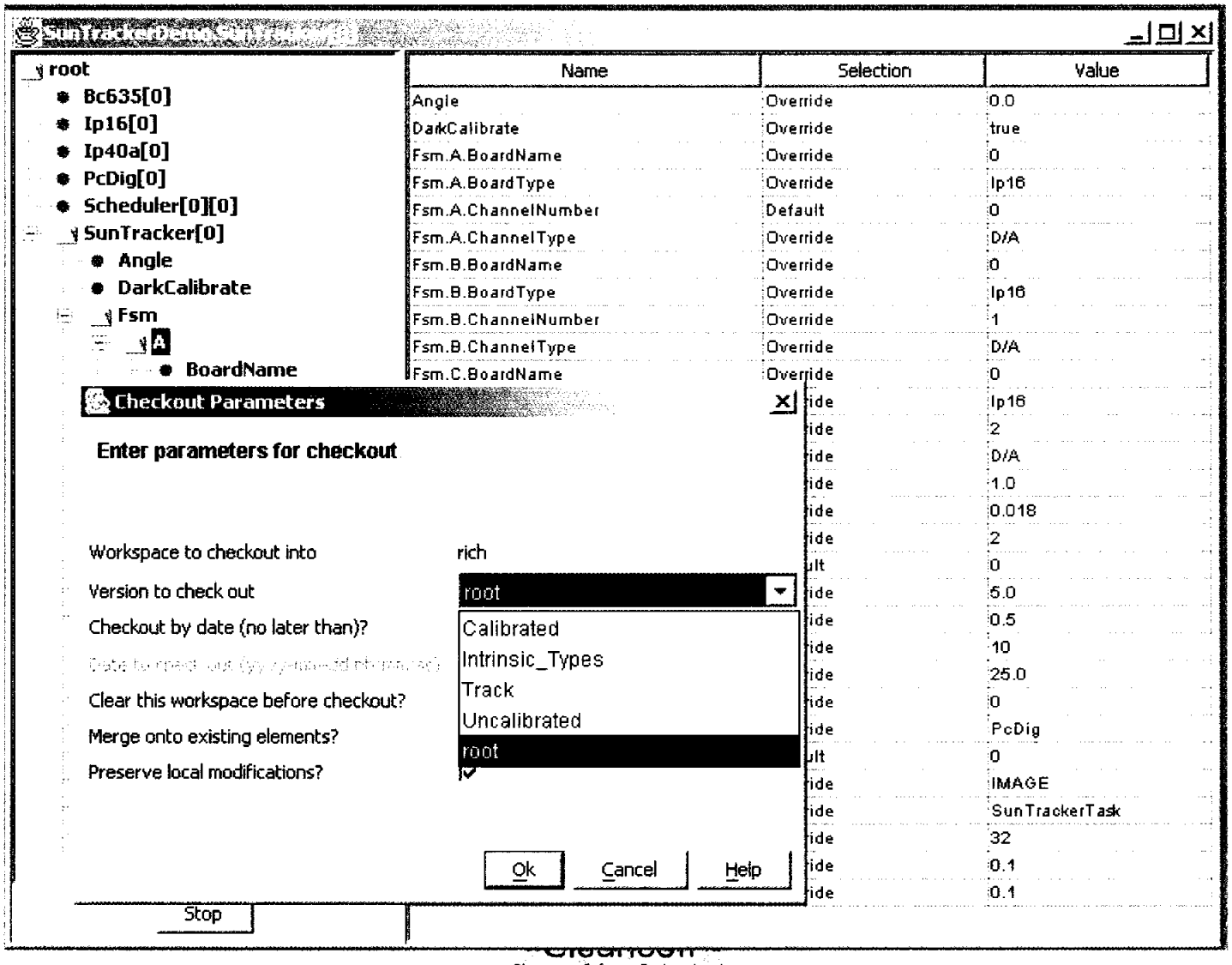
- Now reconfigure the modulation rate



- The spot moves faster

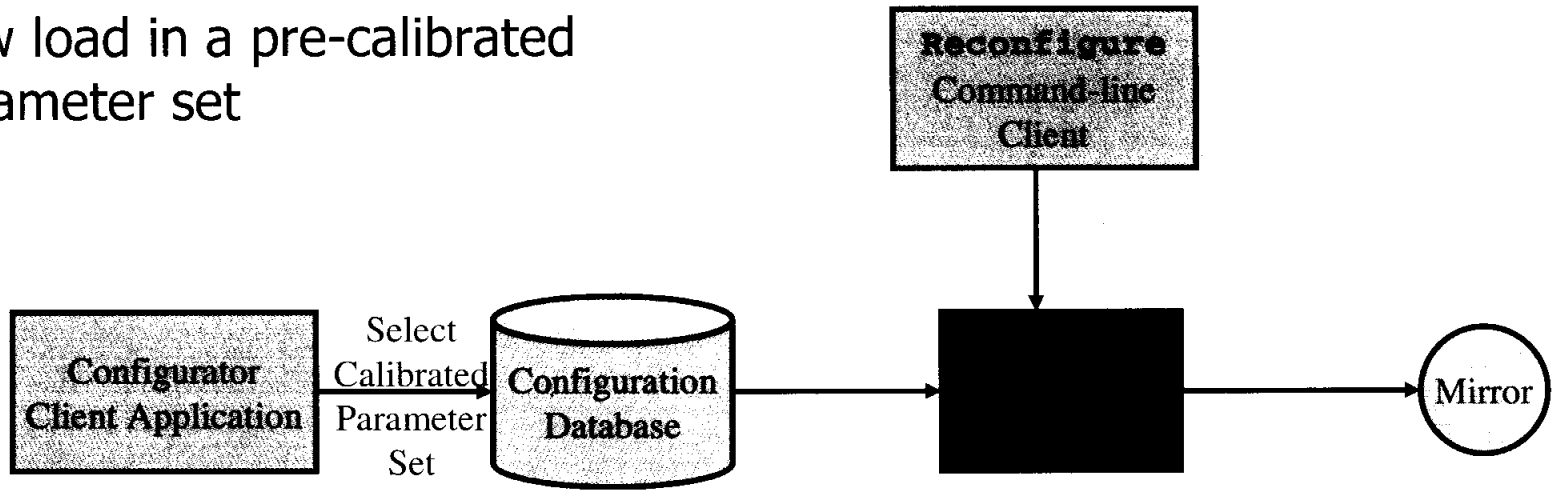


Loading a Named Parameter Set

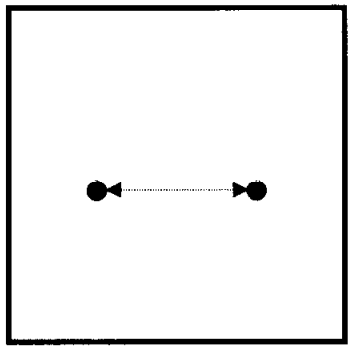


Sun Tracker Demonstration

- Now load in a pre-calibrated parameter set

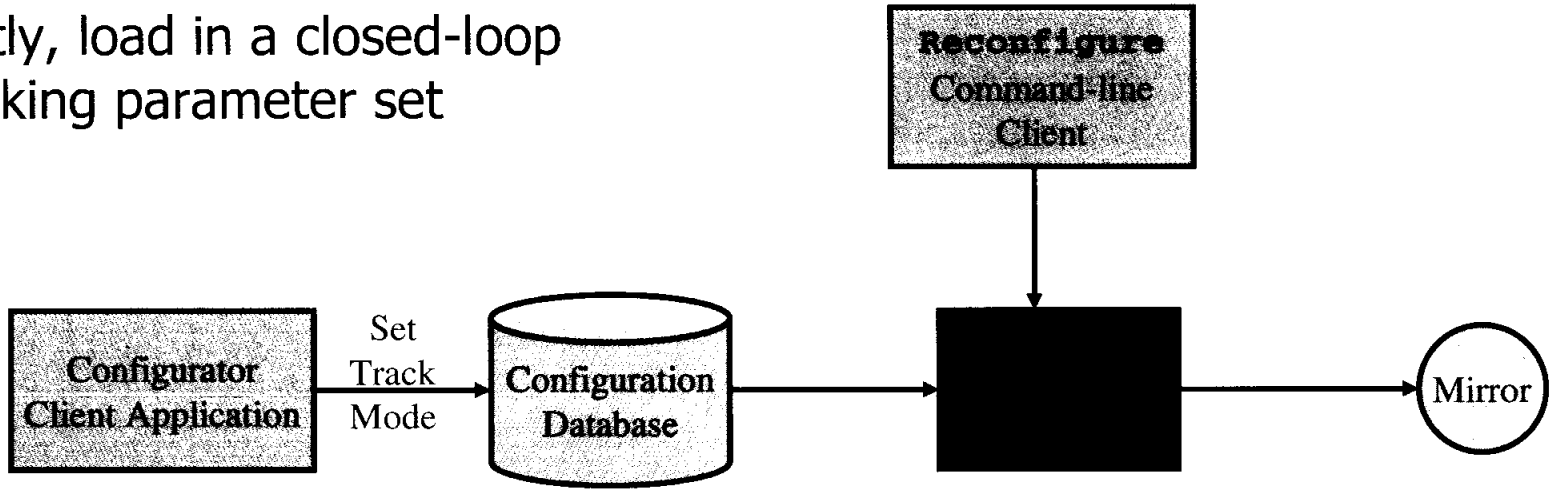


- Now the mirror pointing is correct.

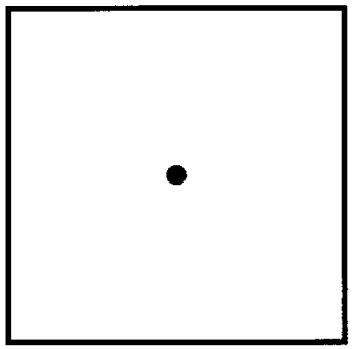


Sun Tracker Demonstration

- Lastly, load in a closed-loop tracking parameter set



- The spot is actively tracked to the center of the camera.



The Investigation Team

- Principal Investigator:
  - Braden E. Hines, Jet Propulsion Laboratory
- Co-Investigators:
  - Thomas Lockhart, Jet Propulsion Laboratory
  - Michael Deck, Cleanroom Software Engineering, Inc.
  - Dr. Shirley Becker, Florida Institute of Technology
  - Dr. Geoffrey Toon, Jet Propulsion Laboratory
  - Dr. Nicola Muscettola, NASA Ames Research Center
  - Dr. Robert Menzies, Jet Propulsion Laboratory
- Additional Researchers:
  - Jean-Francois Blavier, Jet Propulsion Laboratory
  - Christian Plaunt, NASA Ames Research Center
  - Richard Johnson, Jet Propulsion Laboratory