LBT Software System Definition

LBT CAN 481s001d

DRAFT

Revision 2.0 October 29, 2005

Tim Axelrod (taxelrod@as.arizona.edu)

LBT Observatory, 933 N. Cherry Ave., Tucson, AZ 85721

1. Goals of this Document

This document is intended to present the design-in-the-large of the software system that will run the LBT telescope. More detailed information is available in the LBT CVS repository in the form of UML design documents and DOxygen documents generated from source code.

2. Nomenclature

The following definitions are used within this document:

- *Subsystem* - An autonomous software entity that controls attached telescope hardware through one or more hardware interfaces.
- *Controller* - An autonomous software entity without attached telescope hardware.
- *Remote Procedure Call (RPC)* - A communication protocol that allows a client to invoke a function on a remote host. Originally defined by Sun Microsystems, RPC is now widely supported, and is available for most platforms of interest. See (Bloomer 1992) for a full discussion. The available implementations of Sun RPC suffer from problems with multithreading that make it impossible to implement a reliable multithreaded server. To circumvent this, we implemented a somewhat different, simpler, protocol that robustly supports multithreading.

- *Reflective Memory (RM)* - A memory architecture which gives the functionality of high speed shared memory on a distributed system. Each node in the system maintains a copy of the shared memory and utilizes a network interface that propagates changes made in one copy to all others. Although dedicated reflective memory boards are available (VMIC 2002), we have experienced reliability problems in testing them, and have implemented the architecture using a dedicated ethernet and specialized software.
- *Data Dictionary*

3. Relationship of Instruments to TCS

The LBT is designed around a strict separation between instruments and the rest of the telescope system, referred to below as the Telescope Control System (TCS). Instruments interact with the TCS only through an instrument interface (IIF). Through the IIF, the TCS makes available to instruments a command set that provides all required TCS services. The TCS is the servant, the instrument the master. Although we encourage instrument teams to follow our lead on issues such as GUI appearance and functionality, no software standardization is enforced beyond the IIF.

4. System Block Diagrams

The logical and physical connectivity of the LBT subsystems and controllers are shown in Figures 1 and 2. Subsystems are represented by rectangular boxes, with attached hardware shown. Controllers are represented by rounded boxes. The logical block diagram in Figure 1 shows the major communication paths between components. The physical block diagram in Figure 2 shows physical network communication links.

5. Generic Subsystem Characteristics

Even though every subsystem controls specific hardware that has unique properties, and perhaps unique interfaces, all subsystems share some common properties.

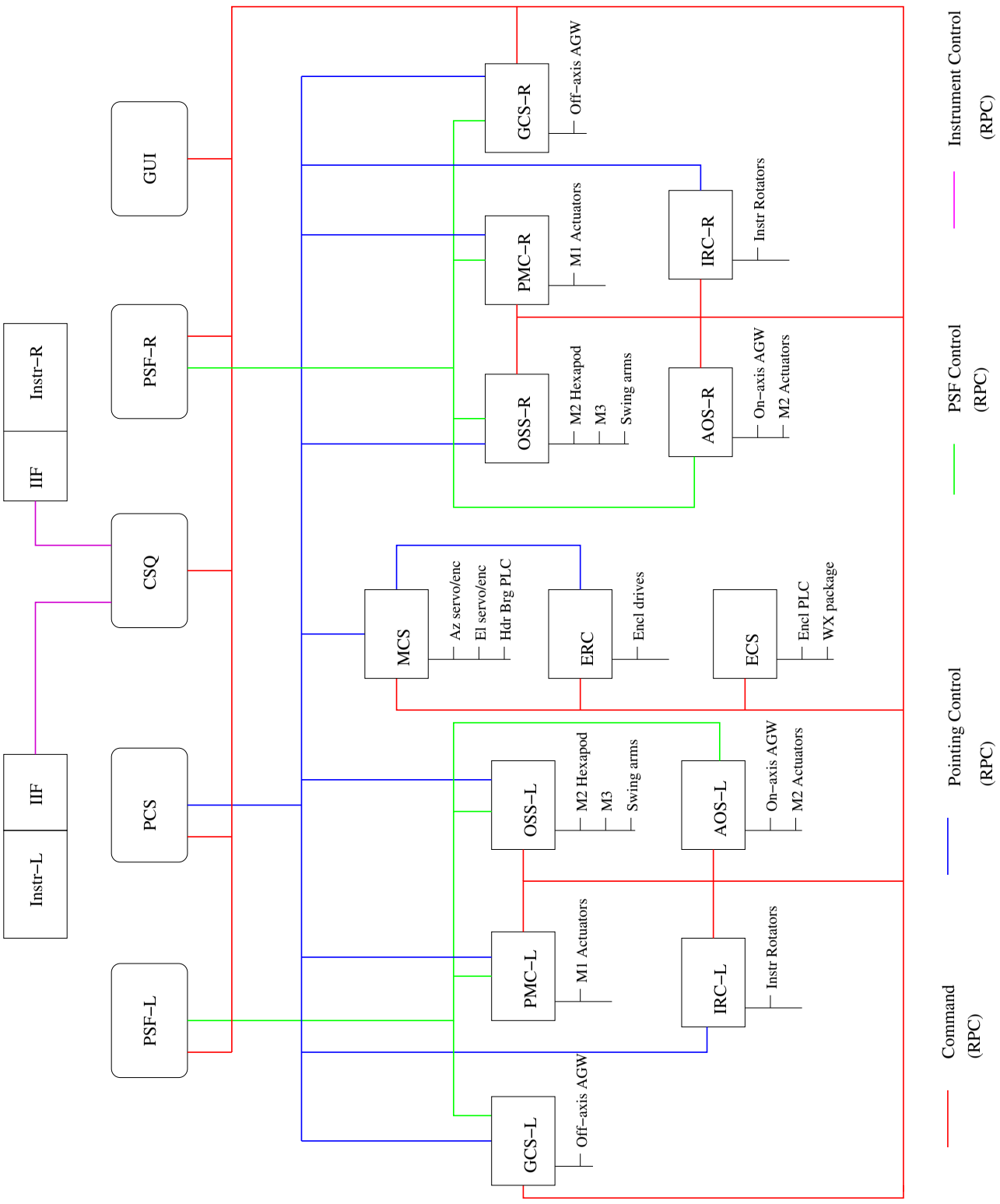

Fig. 1.— LBT Subsystems and Controllers - Logical View

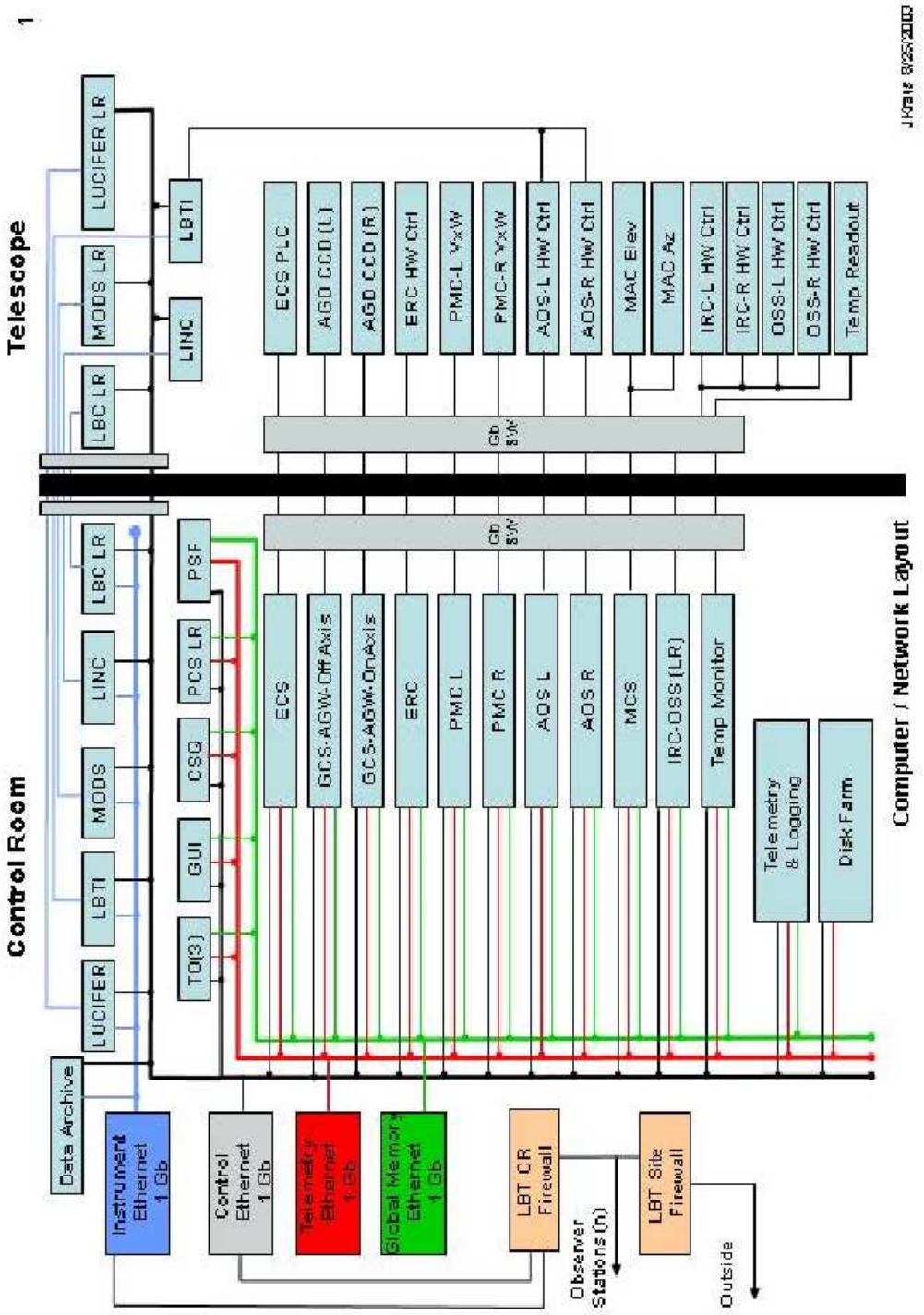

Fig. 2.— LBT Subsystems and Controllers - Physical View

5.1. Autonomous operation

During development and testing of a subsystem, it is essential that the subsystem be able to operate in the absence of other subsystems (which are also likely under development). In most cases, this can be achieved by using a minimal version of the LBT Common Software (see Section 9), the software infrastructure on which LBT subsystems are built. The minimal version offers, purely on the local subsystem machine, the following capabilities:

- management of subsystem configuration through a data dictionary and configuration files.
- logging of events to local log files
- RPC services to local or remote clients.

Even after all LBT subsystems are in place, it should be possible to run a subsystem in isolation using the minimal Common Software.

5.2. Heartbeat

In a distributed system such as the LBT, it is essential to have methods to monitor the health of all nodes. Nodes can fail in a variety of ways, including hardware or OS failures and application hangs. The current LBT architecture implements a simple heartbeat function provided by the "networkserver" protocoal. A server process that runs on every subsystem computer periodically exchanges information with the servers on all other subsystem computers. This software-implemented heartbeat is effective in detecting subsystem crashes, as well as some hardware and OS failures, but in general requires operator intervention to correct a problem. It would be desirable to extend the heartbeat functionality to include periodic testing of subsystem functionality as well as their mere state of existence. Additionally, we may in the future implement for some subsystems watchdog timers which cause the node to be rebooted following a hardware failure or OS hang. Watchdogs are generally not sensitive to application hangs, but have the virtue that a "hard" failure will result in the node being rebooted without operator intervention.

5.3. Subsystem Network Interfaces

5.3.1. Control Ethernet

All subsystems attach to a 1 Gbit/s ethernet whose primary purpose is to carry commands and command results.

5.3.2. Reflective Memory Ethernet

All subsystems attach to a 1 Gbit/s ethernet which is used to implement reflective memory (RM). The primary purpose of RM is to maintain a globally accessible data structure which contains the complete state information for the LBT. Because this state information is replicated across the network, it is robust to single subsystem failures. Following a subsystem failure and subsequent restart, the persistent state information can be used to allow the subsystem to rejoin the telescope system with minimal reconfiguration.

5.3.3. Telemetry Ethernet

As discussed further in Section 5.2, telemetry streams can originate from any LBT subsystem and are transported to the Telemetry and Logging System for collection. Some telemetry sources can require high bandwidth (eg MCS inner servo loops, AO loops), and cannot be accommodated on either the control or reflective memory nets. A separate 1 Gbit/s ethernet-based interface is therefore used for this function.

5.4. Subsystem API

The subsystem API has two parts. The first part is utilized to execute commands. The second part is utilized to get or set the values of keyword parameters.

5.4.1. RPC Commands

The LBT software architecture requires that subsystems execute commands at the request of other subsystems. The mechanism for remote command execution is through RPC over 1 Gbit/sec ethernet fiber. A client subsystem causes a server subsystem to take action by issuing an RPC to any of the functions that the server makes available. RPC's are

synchronous: the client blocks until the server returns a response. All subsystems are multithreaded so that they may perform other work while waiting for an RPC response. Some servers, for example the Command Seqencer, respond to an RPC request by immediately returning a command handle. The command handle allows the client to monitor status of the command and perform other work while the command is underway, without being multithreaded. This is particularly important for the Instrument Interface (IIF).

In addition to subsystem-specific commands, all subsystems support in common a minimal set of commmands that provide for orderly initialization, configuration, restart, and termination.

5.4.2. Keyword Interface and Subsystem State

Background

The Keck control system (Lupton 2000) has shown the value of keyword interfaces to hardware. All Keck hardware, including instruments, is controlled through an interface with functions

- Set keyword value
- Get keyword value
- Monitor keyword value
- Get keyword attributes

A complex Keck instrument can have several hundred keywords.

For the LBT, every subsystem will make available through a keyword interface *all* configurable values within the subsystem, as well as all status information. Unlike Keck, however, where actions are commanded (sometimes awkwardly) as well as parameters set through keywords (Kwok 2002), LBT subsystems provide RPC functions to command actions. This simplifies the use of the keyword API.

Keyword attributes

Every keyword may have associated with it a number of *attributes*. Possibilities include

- Units

- Data type
- Access permission
- Valid range of values
- Identity of owner
- Human readable descriptive text
- Events to send when value changes or goes out of range

All keyword attributes will be stored in a single LBT System Data Dictionary (SDD), discussed further in Section 8. The keyword interface is implemented using Reflective Memory, discussed further below.

System State

Every subsystem has internal and external state information. Internal state information is contained in program variables within the subsystem process address space, is directly accessed by normal memory read and write instructions, and is not externally visible.

The external state of a subsystem is the aggregation of all of its keyword values. External state is accessed through SetValue and GetValue functions, is externally visible to all subsystems through RM, and has a Data Dictionary entry for every item. A change to the value of an external state variable may be requested by any subsystem through a call to the SetValue function. The actual change is made only by the owning subsystem, after verifying that the requesting subsystem has permission to make the change, and that the new value is legal. Such value changes generally are limited to configuration parameters intended to affect subsystem function. A change in value of an external state variable can optionally cause the owning subsystem to generate an event, which will be handled by the Event System. Values can be read through the GetValue function, which utilizes the data in RM, and does not require action by the owning subsystem.

Both SetValue and GetValue functions are operable without RM, so that clients can be tested remotely over the internet. Functionality is identical, but with reduced efficiency and increased latency.

5.5. Subsystem Template

Note that, in practice, these requirements are most naturally met if every subsystem has a dedicated host computer. To avoid too great a proliferation of computers (and to save money!), we are planning to have some subsystems share computer hardware. This introduces some complications into the RM implementation, and means that remote reboot will affect multiple subsystems.

An generic subsystem block diagram is shown in Figure 3. An implementation of this block diagram as a template subsystem is now largely complete, and is available from the CVS repository.

6. LBT Subsystems

The LBT contains the following subsystems, each of which is more fully described in subsequent subsections:

- ECS - Enclosure Control System
- TEL - Telemetry System
- LSS - Logging and Event System
- MCS - Mount Control System
- OSS - Optics Support Structure Control System (L, R)
- PMC - Primary Mirror Control System (L, R)
- GCS - Guiding Control System (L, R)
- AOS - Adaptive Optics System (L, R)

A subsystem description must include its

- Role
- Attached hardware
- Communication pattern
- Keyword set

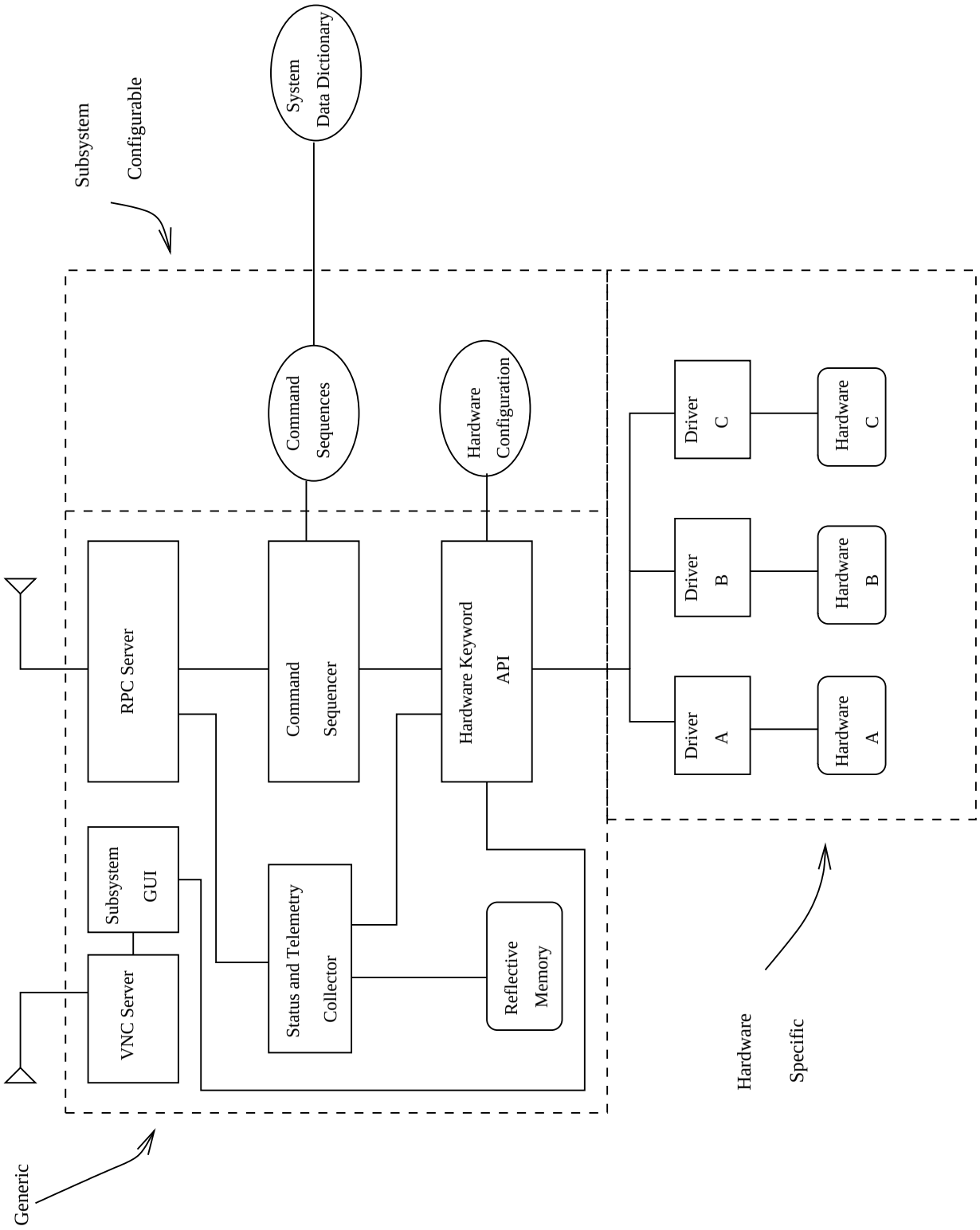

Fig. 3.— LBT Subsystem Template

- Command set

In this document, the focus is on the first three of these items. The keyword and command sets are discussed elsewhere in more detailed UML and DOxygen documents. The level of detail in the subsystem descriptions currently varies widely, reflecting the state of each subsystem design.

6.1. ECS - Enclosure Control System

6.1.1. Role

The ECS is responsible for the control of all aspects of the telescope enclosure, both rotating and non-rotating. These functions are fully described in (M3 2000). The ECS provides access to all control primitives for the enclosure hardware. Additionally, the ECS executes complex functions (eg opening both shutter doors) by issuing a sequence of lower level commands to the attached hardware.

The ECS maintains a status structure which contains all readable state information from the enclosure. This status information will be provided to RPC clients, and also maintained in reflective memory (RM) for low overhead use by other LBT subsystems.

The ECS controls the LBT weather monitoring system, and makes information available from it to RPC clients, and through RM.

6.1.2. Attached Hardware

All control of enclosure hardware is carried out through the Enclosure PLC (EPLC). The EPLC is a Rockwell RSLogix 5500, connected to the ECS by 10 Mb/s ethernet, and to enclosure hardware by a combination of ControlNet and hardwired lines. The following classes of hardware are controlled by the EPLC:

- Air compressors
- Chillers
- Instrument cooling
- Dome cooling

- Mirror ventilation system
- Dampers
- Vent doors
- Exhaust fans
- Vane axial fans
- Shutter doors
- Snow melting system
- Lighting

Additionally, there is a TBD interface to a weather monitoring system to be placed on the enclosure roof (EWX).

6.2. LSS - Logging and Event System

6.2.1. Role

It is essential that all significant events that take place within the LBT TCS are recorded in an event log. Rather than have each subsystem maintain a separate event log, a global logging service is provided. This ensures that all events, regardless of their source, are recorded in a uniform way and are properly timetagged so that event ordering is never in question.

Additionally, subsystems and gui's may register with the LSS to be notified when particular classes of events occur.

If the TLS is unavailable (for example during subsystem test in isolation from LBT), TLS functionality will devolve to the use of local log files.

6.3. TEL - Telemetry System

6.3.1. Role

The function of the TEL is to accept and archive telemetry streams from any subsystem that wishes to provide them. A telemetry stream is a sequence of timetagged values for some

subsystem variable. Although variables may have simple data types (eg an int or a float), in general they will be structures of arbitrary complexity. All LBT variables will be defined in a system-wide data dictionary (SDD) that provides the data structure, units, and a human-readable definition (see Section 8). Telemetry streams may have high bandwidth, eg for servo inner loop variables. For this reason, streams are transported over a dedicated 1 Gbit/sec ethernet rather than over the telescope command ethernet.

The telemetry archive stores the telemetry data in HDF files (NCSA 2003), and maintains a telemetry catalog in a MySQL relational database. This design both allows flexible searching for streams of interest and achieves efficient access and storage.

Streams may be defined from configuration files at system initialization time, and this facility will be used to ensure that some basic telemetry streams are always present. Other ad-hoc streams can be defined or removed as the need arises.

6.3.2. Attached Hardware

- High performance disk array for telemetry archiving

6.4. MCS - Mount Control System

6.4.1. Role

The MCS controls a wide variety of telescope hardware associated with telescope safety, balance, main axis motion, enclosure rotation, and instrument rotator motion.

The design of the MCS is described in (McKenna 2002) and (Ashby 2003).

6.4.2. Attached Hardware

- Elevation and azimuth motors and encoders are interfaced to DSP boards which are linked by fiber to the MCS host.
- The hydrostatic bearings are controlled through an attached PLC.
- Many telescope auxiliary functions (eg hydrostatic balance system) are controlled through an attached PLC.
- Enclosure rotation is controlled through an attached PLC.

- Instrument rotators and associated cable chains.

6.4.3. Communication Pattern

Aside from ensuring the safe movement of the telescope, the main role of MCS is to follow trajectories for the main axes and instrument rotators that are calculated by the PCS. Trajectory information will be streamed from PCS to MCS through RM, at an update rate of 20 - 50 Hz. Commands to change operating modes and the like will be delivered via RPC, as with other subsystems.

6.5. OSS-L,R - Optics Support Structure Control System

6.5.1. Role

The OSS is responsible for positioning the secondary hexapod, tertiary mirror, swing arms, and mirror covers.

6.5.2. Attached Hardware

- M2 Hexapod DSP controller (fiber link)
- M3 DSP controller (fiber link)
- Mirror cover via PLC
- Swing arms via PLC

6.5.3. Communication Pattern

As with the MCS, the OSS is responsible for following M2 hexapod trajectories generated by the PCS. Trajectory delivery will be through RM. Other operations are commanded by RPC.

6.6. PMC-L,R - Primary Mirror Control System

6.6.1. Role

The primary mirror control system is responsible for the position, shape, and temperature distribution of the primary mirror. The overall approach is presented in (Hill 1995).

6.6.2. Attached Hardware

One VxWorks box per cell is responsible for controlling all PMC hardware:

- Mirror cell hardpoints (6)
- Mirror cell actuators (160)
- Air supply valves
- Voltage monitors

6.6.3. Communication Pattern

As with the MCS, the PMC is responsible for following M1 pointing trajectories generated by the PCS. Trajectory delivery will be through RM. Other operations are commanded by RPC.

6.7. AOS-L,R - Adaptive Optics System

6.7.1. Role

The AO system is described in (Esposito 2002). It's goal is to maintain a diffraction limited PSF for the science object. To do this, it must close the loop from wavefront sensor to mirror actuators at approximately 1 KHz.

6.7.2. Attached Hardware

The AO system contains the following hardware, all interconnected with a dedicated fiber optic network. The wavefront

- Motorized optics (on-axis AGW unit)
- Wavefront detector (on-axis AGW unit)
- Slope calculator
- Wavefront reconstructor
- Deformable secondary mirror

6.8. GCS-L,R - Guiding Control System

6.8.1. Role

All LBT instruments will normally make use of guiding and wavefront sensing during science exposures. The way in which these capabilities are provided depends on the instrument in use:

- LBC has internal guider and wavefront sensors.
- Lucifer, PEPSI, LBTI, and LINC/NIRVANA make use of the LBT AGW units.
- MODS uses a unit functionally similar to the LBT AGW, but differently packaged, and with different motor controllers.

The LBT AGW and MODS AGW will be on the LBT side of the LBT/Instrument interface, and will be directly controlled by the TCS. The LBC handles its guiding and wavefront sensing entirely within the instrument, and provides guiding and wavefront correction information to the TCS over the Instrument Interface (IIF) (Axelrod 2003)

Guiding of the LBT is performed by the Pointing Control System (PCS), based on information delivered from the active guiders (in general separate for L and R sides of the telescope). Centroid information is to be delivered to the PCS in a standard focalplane coordinate system (SFP) which is instrument independent. A natural choice for this system is polar coordinates in an ideal tangent plane, where theta is measured with respect to the

rotator reference axis (ie, the coordinate system moves with the rotator). The units of both r and theta will be in radians.

It will be the responsibility of the guider to transform between detector coordinates and SFP coordinates, correcting for flexure of the guide probe, guider unit temperature, optical distortion within the guider, etc.

Similarly, optimization of the imaging performance of the LBT is performed by the PSF Control System (PSF), based on information from the active wavefront sensors. This information is to be delivered in a standard, instrument independent form. Zernike polynomial coefficients are a possible choice, but a final decision still needs to be made. As with SFP coordinates, the reference axis should be tied to the rotator.

The guiding task naturally partitions into four layers:

- 1. Control of guider hardware, and readout of guider CCD.
- 2. Identification and centroiding of guide star.
- 3. Transformation of guide star centroid from detector coordinates to standard FP coordinates. This includes compensating for guide probe flexure, optical distortion, etc.
- 4. Action by the telescope to correct tracking errors.

The first three layers are dependent on the details of the guider hardware, while the fourth layer is explicitly designed to be independent of the guider hardware, and is performed by the PCS.

In the case of the LBT and MODS AGW units, layers 2 and 3 are part of a common GCS software package, run by the GCS subsystem. Layer 3 must be aware of the details of guider system geometry, flexure, etc, which differ between the LBT and MODS. These details will be encapsulated in a configuration file, or similar external data structure. Layer 1, which includes low level motor control, is also contained in the GCS subsystem. It will differ significantly between the LBT and MODS units, and will likely be implemented as a dynamically loadable library, or similar.

In the case of the LBC, layers 1 through 3 reside wholly within the LBC instrument system.

From the perspective of the TCS, the key interface is between layers 3 and 4. In addition to employing SFP coordinates, the interface has the following characteristics:

- The guide star centroid information should be transmitted "raw", without any time domain filtering.
- An error estimate for the centroid position should be included. Both the position and error are in standard FP coordinates.
- The measurement should be time tagged, using a time source synchronized to the observatory NTP net.
- For the LBC, the rotator angle must be included with the guide star centroid information.

6.8.2. Attached Hardware

The GCS controls the off-axis AGW unit (Storm 2002):

- Guider camera
- Wavefront sensor
- Filter wheel
- Stage motion

7. LBT Controllers

As noted above, *subsystems* are directly tied to hardware, and the management of this hardware is their main role. Higher level functionality, that involves cooperation of multiple subsystems, is provided by *controllers*. The LBT controllers are briefly described in this section.

7.1. CSQ - Command Sequencer

Commands from attached instruments to the LBT are delivered to the Command Sequencer. Typically, such an external command will require a number of coordinated actions to be taken by multiple subsystems and controllers. The role of the CSQ is to decompose such incoming commands into sequences of more primitive commands to be executed by the involved subsystems and controllers.

Although the CSQ is the sole entry path into the LBT for attached instruments, its services may also be called upon by the GUI, or by other LBT subsystems and controllers.

7.2. GUI - Graphical User Interface

The GUI are being built using Qt Designer. A draft GUI standards document is available in (De La Pena 2003).

- RPC and VNC based - can display anywhere that has network access, including remote sites. RPC is used for obtaining status information (if RM is not available), and for sending commands to the CSQ. Display uses VNC, so that display is practical even over low bandwidth lines (eg from Steward or Arcetri), and to devices like handheld computers.
- Will use RM for speed if attached (configurable)
- Each controller and subsystem will have a dedicated GUI panel.
- Additionally, a central telescope system display provides at-a-glance assessment of telescope status, and directs any corrective action required from the TO through the use of checklists.

7.3. PCS - Pointing Control System

The Pointing Control System has the task of coordinating all subsystems that play a role in telescope pointing. "Telescope pointing" in this context has the most general possible meaning. Any action that affects the (x,y) coordinates of an astronomical object on either instrument focal plane is by definition part of telescope pointing. It is here that the binocular aspects of the LBT are handled. The PCS brings together the pointing demands for the left and right telescopes, and attempts to optimally translate those demands into command streams to LBT subsystems. It is unique within the LBT system in this regard. With the exception of the PCS, controllers and subsystems handle left and right sides independently (or are responsible for non-duplicated hardware, eg ECS and MCS).

Four subsystems accept commands from both the PCS and the PSF: OSS-L and -R, and PMC-L- and -R. There is in fact, however, a clean separation between the actions invoked by the PCS and the PSF. The PCS commands only "lateral" degrees of freedom that affect telescope pointing, as defined above. These comprise tip-tilt of both primary and secondary

mirrors, and relative lateral movement between primary and secondary. The PSF, on the other hand, handles all other degrees of freedom, in particular relative longitudinal motion (focus), and the forces that affect primary mirror figure.

All geometric transformations are based on Pat Wallace's virtual telescope concept, and build on the experience with Gemini pointing and guiding (Wallace 1997b).

7.4. PSF - PSF Optimizer

The Point Spread Function Optimizer has the task of coordinating the M2 hexapod motions, the M1 hardpoint motions, and the M1 actuator forces so that the delivered PSF at the instrument is optimized.

8. Instrument Interface Library

Instruments access LBT functions by issuing RPC calls to the Command Sequencer (CSQ). As sketched in (Axelrod 2003), the client (instrument) part of the RPC code will be incorporated into an Instrument Interface Library.

9. LBT Common Software

The structure of subsystems and controllers described above depends on several infrastructure capabilities provided by a Common Software layer.

- Time Service. A GPS synchronized time server will be present on the LBT ethernet, and will offer time services to any client via ntp. Particularly since the system design is intended to keep the ethernet load light, ntp should easily provide time accurate to 1 msec.
- System Data Dictionary (SDD). All externally visible subsystem variables will have their attributes defined in a single dictionary. The technology for implementing this dictionary is based on XML. The dictionary will be be populated mostly automatically, working from source code. Otherwise, it is inevitable that the dictionary will not remain up to date as the code changes.
- Event System Subsystems can both generate events and receive events generated elsewhere in the system. Events are generated by errors or abnormal conditions, but can

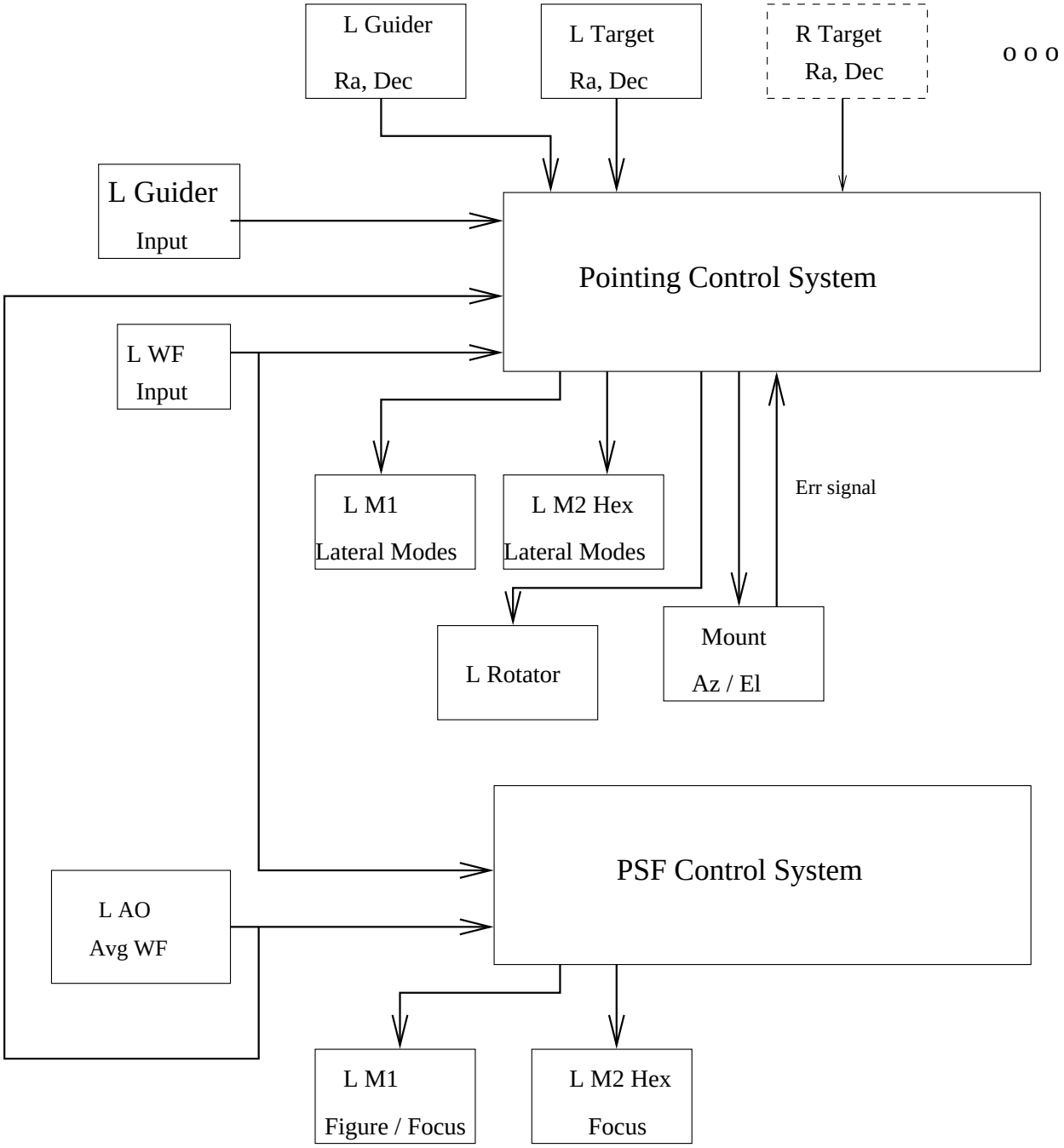

Fig. 4.— LBT Pointing and PSF Control

also be generated when an external parameter's value is changed by SetValue. Subsystems receive only events to which they have subscribed, and on reception may take whatever action is appropriate. All events are logged by the TLS to a central event log. This will initially consist of XML files, but is likely to evolve to a full database.

- Telemetry Streams Subsystems may register telemetry streams, sets of time sampled variables, which are delivered over the telemetry network to the TLS for archiving.
- Reflective Memory As discussed extensively above, reflective memory underlies several key software concepts. In particular it makes the entire telescope external state visible to all subsystems with an effective rate of about 100 Hz.

10. Implementation Guidelines

- Language and OS. Where possible, system components will be implemented using Linux and C++. It is recognized that some subsystems may use other technologies, such as VxWorks (AGW units) or Windows (GUI). As long as RPC is available, all technologies should be able to work together.
- Document Generation. As with any software project, creating and maintaining up-to-date documentation will be a challenge. We will need to select, and consistently use, an automated document generator such as DOxygen.
- CVS Use. All code and documentation will be maintained under CVS, with remote access enabled so that all partners can make use of the repository. Given the wide geographical distribution of the partners, the repository architecture needs to be carefully considered to optimize its usefulness.
- Bug Tracker. We have adopted RT (http://xxx) as the tool for tracking software bugs and issues. Experience with other software projects shows that this is a major contributor to producing, and maintaining, a reliable system.
- Multithreading. As noted above, all system components are likely to be multithreaded. Multithreading will be implemented using POSIX threads where available.

11. Unresolved Issues

- High speed chopping interface for LBTI.
- Network security plan, particularly given desire for remote use.

REFERENCES

Ashby, D, "LBT Project Main Axis Motion Control", LBT CAN 483s101a (2003)

Axelrod, T. Schmelmer, T., and Wagner, R.M., "Command Set at the Instrument-Telescope Interface for the LBT", LBT CAN 481g010a (2003)

Bloomer, J., "Power Programming with RPC", O'Reilly and Associates, Inc (1992)

Butenhof, D.R., "Programming with POSIX(R) Threads" Addison-Wesley(1997)

De La Pena, M., "LBT Telescope Control System GUI Guidelines", LBT CAN 481s010a (2003)

Esposito, S., et al, "First Light Adaptive Optics System for Large Binocular Telescope", SPIE Conference on Astronomical Telescopes and Instrumentation (2002)

Hill, J., "Mirror Support System for Large Honeycomb Mirrors", University of Arizona UA-95-02(1995)

Kwok, S.H. and Cohen, R.W., "Keywords Revisited", Proceedings SPIE Astronomical Telescopes and Instrumentation, 2002

Lupton, W., "Keck Telescope Control System", ADASS IX, ASP Conf. Series, V216, 261(2000)

McKenna, D, "Large Binocular Telescope Mount Control System version 2.0", LBT CAN 480xxx (2002)

NCSA, http://hdf.ncsa.uiuc.edu/doc.html

M3 Engineering, "LBT Enclosure Control Description", LBT-CAN-309m001b(2000)

Storm, J., "LBT AGW Software Interface Specification", LBT-CAN-680a030a(1999)

Storm, J., "LBT AGW Technical Specifications", LBT-CAN-680a020d(2002)

VMIC, http://www.vmic.com/products/reflectivememory/index.html

Wallace, P.T., "Gemini Pointing Algorithms", Gemini TCS Note (1997)

Wallace, P.T., "Autoguiding and M2 Tip/Tilt: Astrometric Procedures", Gemini TCS Note (1997)
