Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

The Development Process of the LUCIFER Control Software

Marcus Jütte

Astronomisches Institut, Ruhr-Universität Bochum, Germany

Kai Polsterer

Astronomisches Institut, Ruhr-Universität Bochum, Germany

Michael Lehmitz

Landessternwarte Heidelberg, Germany

Abstract. We present the design and development process of the control software for the **L**BT NIR spectroscopic **U**tility with **C**amera and **I**ntegral-**F**ield Unit for **E**xtragalactic **R**esearch (LUCIFER) which is one of the first-light instruments for the LBT on Mt. Graham, Arizona. The LBT will be equipped with two identical LUCIFER instruments for both mirrors. We give an overview of the software architecture and the current state of the software package and describe the development process by using a virtual LUCIFER instrument.

1. Introduction

The LUCIFER instrument is built by a consortium of five German institutes, Landessternwarte Heidelberg (LSW), Max Planck Institut für Astronomie in Heidelberg (MPIA), Max Planck Institut für Extraterrestrische Physik in Garching (MPE), Fachhochschule für Technik und Gestaltung in Mannheim (FHTG) and Astronomisches Institut der Ruhr-Universität Bochum. It is a full cryogenic near-infrared (NIR) spectrograph and imager (0.9 - 2.5 m, zJHK-band). It is equipped with three exchangeable cameras for imaging and spectroscopy, two optimized for seeing limited conditions and the third camera for the diffraction limited case with the LBT adaptive secondary mirror working. This leads to six observing modes: seeing and diffraction limited imaging, seeing and diffraction limited long slit spectroscopy and seeing and diffraction limited multi-object spectroscopy. For the seeing limited case there will be a field of view (FOV) of 4 x 4 $arcmin^2$ and for the diffraction limited case a FOV of 0.5 x 0.5 $arcmin^2$. For the long slit and multi object spectroscopy (MOS) up to 33 exchangeable masks over the full field of view will be available. The instrument will be equipped with a Rockwell HAWAII-2 HgCdTe-array with a pixel size of 18 microns. It is planned to build two identical instruments for both LBT mirrors having first light with LUCIFER 1 estimated in summer 2005. For more details see Seifert et al. (2002) and Hoffmann et al. (2002).

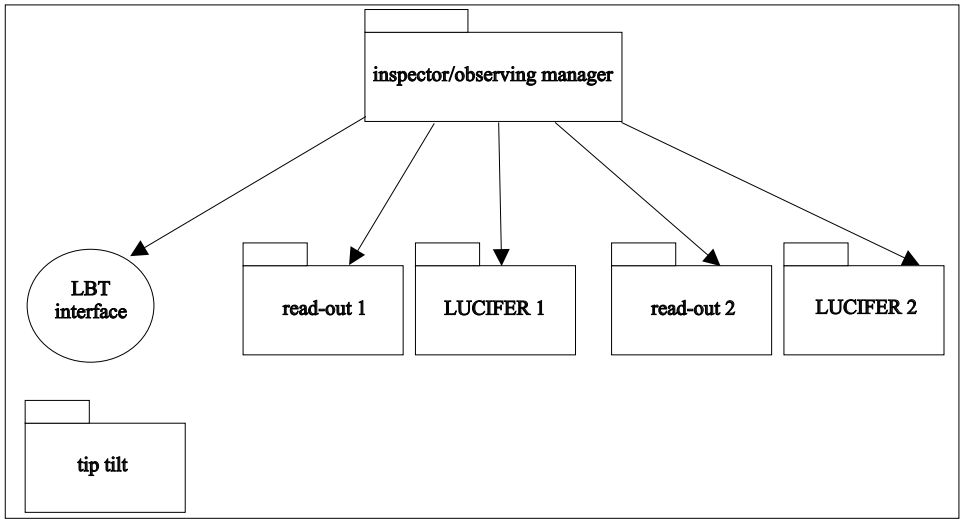

Figure 1. The overall LUCIFER software architecture.

2. Soft- and hardware architecture

For each LUCIFER instrument the soft- and hardware can be distinguished into two major parts: an instrument control and a read-out one. The instrument control software package includes individual modules for each mechanical unit like the grating, filter wheel or MOS unit. The read-out software package will work as a server and will supply the resulting image data to the overall inspector, which works as the master in this hierarchy and combines it with the instrument data to a FITS image. The read-out process is hardware independent and it is planned to adopt the C++ software for this process from the existing code of the OMEGA 2000 camera (provided by the MPIA, Heidelberg). Since the NIR detector read-out electronics, which will be built at the MPIA, requires special computer hardware, the read-out has to be controlled by at least a SUN Ultra Sparc 80 workstation with a minimum of 4 GB RAM. This machine will be faced with a data stream of approximately 40 MB/s on two PCI buses respectively. Currently two solutions are discussed: One with a powerful SUN V480 workstation with 4 SparcIII processors, 8GB RAM and 72 GB FC hard disk or the other one with two weaker SUN V 270 workstations for each LUCIFER. The motor electronics and the monitoring hardware for temperature and pressure can be accessed via serial connections. In order to reach these units from the control room a port sever is implemented in the electronic rack at the telescope site. It is a network-based serial device server for connecting 16 RS-232 directly to a TCP/IP network (Ethernet or Internet) and is delivered in a 19-inch rack.

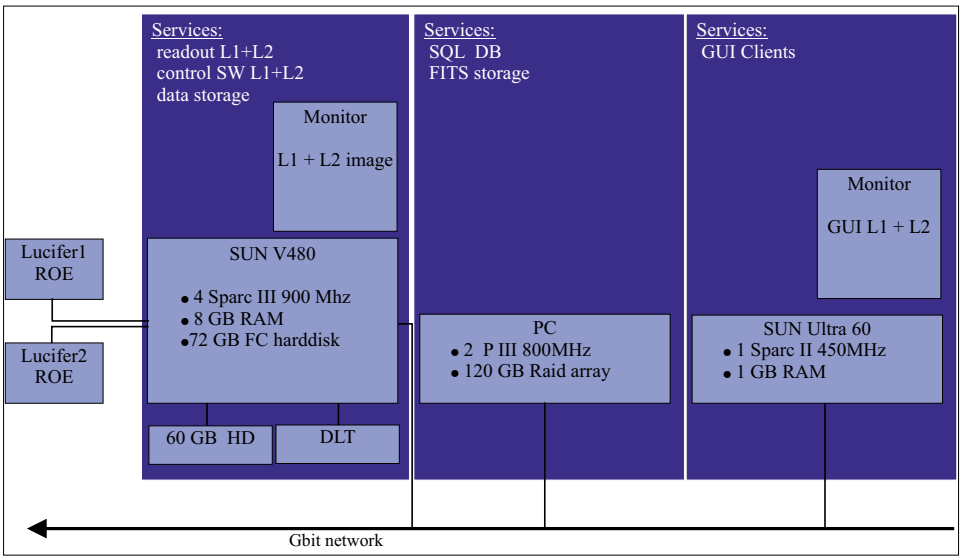

Figure 2. Hardware architecture and services.

3. Software development process

Since the server provides RJ45 connectors an additional fibre-to-RJ45 adapter is necessary to connect to the local LBT network. The port server is identified by its own IP address and running this server in raw mode allows direct connection to each serial device, which is represented by a specific socket address. Due to the electronics design every serial port is responsible for sending commands to eight motors inside LUCIFER and receives the appropriated answers.

Due to the complexity of the instrument (it is a cryogenic instrument where no encoder are available) and the fact that the whole software package has to be easy maintained after commissioning an object-oriented software solution with extensive UML (Unified Modelling Language) design is desirable. Therefore we choose Java as the appropriate OO language. It is platform independent and easy to maintain. For the documentation the powerful Javadoc style is used. The UML design has been prepared with the Together Control Center, which even provides reverse diagrams for already existing source codes. For the coding process we use the open source tool editor Eclipse, which supports Java syntax and CVS and much more. The derived classes are proofed with the help of JUnit tests.

The development process can be describes in four major steps:

1. Analysis of the problem
2. Design of classes
3. Development of a rapid prototype
4. Re-Design

In order to test the communication concept and to get experiences with the control electronics a prototype software has been developed. It is able to control the motors by sending appropriate firmware commands and can measure temperature and pressure from the monitoring devices.

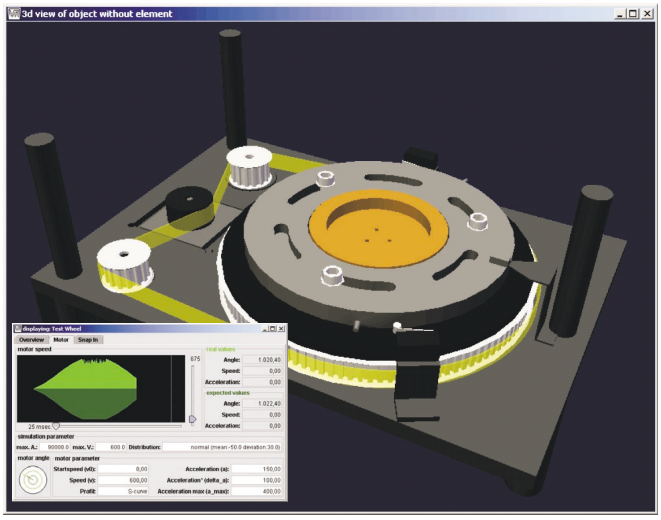

Figure 3. The virtual LUCIFER instrument.

4. Software tests

The most important part during the software development process is to ensure that the build software complies with the specification made previously. Because of the complexity of the instrument a test of the logical behaviour of the software is necessary in advance of the completion of the instrument. For this purpose a virtual instrument has been developed, simulating the behaviour of all implemented units and motors. It consists of a TCP/IP interface so that the control software communicates the same way as the real instrument. To provide a tool which can be used as a testing environment three independent software packages had to be implemented:

- A package simulating the mechanical movement inside the instrument. This was done by mixing an event orientated approach with a mathematical one to ensure temporal consistency of real and model time.
- A package emulating the communication protocol of the used MPIA control electronics. This packages was realized by writing a command parser which controls the simulation and generates responses.
- A package for visualisation of the current instrument status. This package was written using Java3D and enables the user to see virtually the moving of the simulated instrument.

Acknowledgments. This work is supported by the German Federal Ministry for Education and Research (BMBF) under Ids: 05 AL2V01/8, 05 AL2PCA/5, 05 AL9EE1/7.

References

Hoffmann, R., Thatte, N.A., Tomono, D. 2002, SPIE Proc. Vol. 4841, 2026

Seifert, W., Mandel, H., Appenzeller, I., et al. 2002, SPIE Proc. Vol. 4841, 2002