EOS Software Systems for Satellite Laser Ranging and General Astronomical Observatory Applications

M. Pearson[1]

1. EOS Space Systems Pty. Ltd.

Contact mpearson@eos-aus.com

Abstract

EOS has developed software systems over many years to support Satellite Laser Ranging (SLR) and the delivery of general astronomical observatories to its customers. These software systems are based upon a re-useable software architecture that simplifies systems development. The design objectives of this software architecture are discussed in the context of its evolution and current deployment at the Mt. Stromlo Satellite Laser Ranging facility, located in the Australian Capital Territory.

Introduction

This paper presents a brief overview of the software EOS has developed to support Satellite Laser Ranging at Mount Stromlo, software that has been designed to support not only SLR, but a wide range of astronomical applications.

This software, known as the 'Observatory Control System', supports EOS research and development programmes. It also supports the observatory requirements of several customers, being a flexible and scalable product, which lends itself to re-use across laser ranging and astronomical observatory applications.

Requirements

The basic requirements of the Observatory Control System are:

- it should drive equipment that might be expected at an observatory;
- it should provide some sort of abstraction – an 'Observatory' abstraction – that hides the complexity of the underlying equipment and presents it in terms that end-users and operators are likely to understand;
- it should provide facilities to automate day-to-day operations and routine observatory tasks.

Challenges

With these goals in mind, EOS has developed Observatory Control Systems over many years. But there was a problem; as the complexity of observatories grew, so did the complexity of the supporting software. This led to several challenges which are encountered in all types of software:

- it was becoming monolithic with fewer, larger, more-complex components;
- these components were highly-coupled, so changes to any part of the system could unexpectedly impact seemingly un-related parts;
- these systems were becoming inflexible and difficult to change in order to meet new requirements;
- different observatory solutions were becoming increasingly problem-specific and less re-usable; they were not amenable to solving new problems.

Solution

Under these pressures EOS engaged in a complete re-design of its software, culminating in what is now the 'Observatory Control System'.

The result is that the control system is inherently unaware of its problem domain. It is called an 'Observatory Control System' but it is in fact a generic control system. Its immediate application is to SLR and astronomy, but it could drive any automated industrial facility. It is domain independence that makes the control system highly extensible, flexible and most-importantly for EOS, re-useable.

Basic Architecture

At the highest level the Observatory Control System embodies the system concept:

'a collection of components which work together in order to solve a problem'.

These components include:

- various types of hardware and software;
- usually some sort of network;
- control system and observatory-level software.

The control system software provides facilities including:

- server frameworks;
- client frameworks;
- network interfaces.

The observatory-level software provides facilities including:

- servers;
- clients;
- automation functions.

Refer to Figure 1 for an illustration of these components.

Hardware & Software

Hardware and software include such items as: telescopes, enclosures, lasers, associated software and a variety of other equipment. A common problem with such equipment is that it is often heterogeneous, with different:

- platforms, eg. PC, Mac;
- operating systems, eg. Windows NT, XP, Linux;
- interfaces, eg. serial, CANopen, USB, Bluetooth;
- protocols, eg. sockets, CORBA, COM.

A fundamental feature of the Observatory Control System is that it makes this equipment look, feel and act in a consistent manner. This is achieved by hiding the equipment behind a universal software abstraction – what is called a Device.

Network

Devices are usually accessed through an adapter card and a driver library. But there is a limit to how many devices a given computer can support; at some point a single computer will run out of capacity, or a new device will require a different operating system or computer platform. So most observatories require many computers and the Observatory Control System is network-enabled.

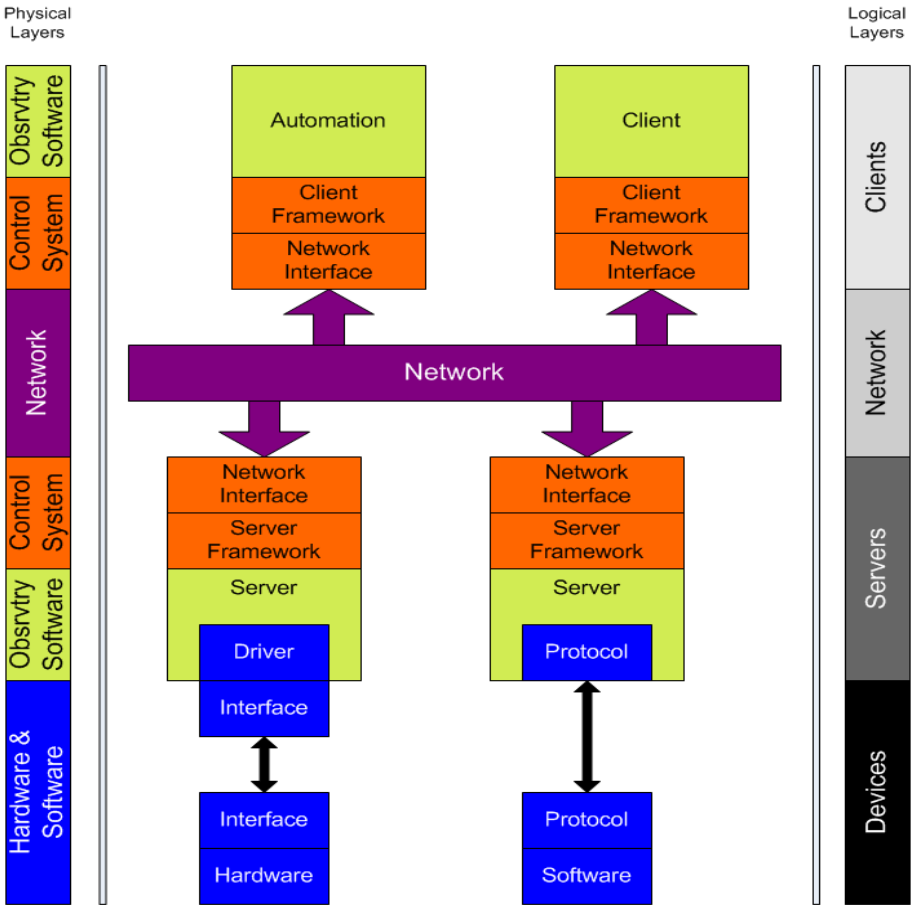

Figure 1. *Basic Software Architecture*

Control System

The Observatory Control System is a Client / Server software architecture. This is a network computing model based on the following concepts; that:

- clients connect to servers;
- clients issue commands to servers;
- servers respond to client commands.

Within the Observatory Control System, device server applications wrap devices and make them available over the network.

Client applications connect to these device servers over the network, or even over the Internet, and drive devices via commands to the relevant device server.

At the heart of the Observatory Control System are several software frameworks; these are code libraries which embody the most-re-useable, but complex and technical aspects of the control system.

These frameworks encourage re-use, and are used by EOS to extend its systems. These frameworks are also available to customers, who can extend their observatories over the longer term, independent of EOS.

Server Framework

Server applications, also known as device servers, are built using a *Server Framework*.

Servers directly manage observatory devices. A device server may manage one or many devices, depending on our requirements.

Servers may be active and/or passive; that is, they may manage devices autonomously and robotically, or in response to user commands.

Servers may participate in a hierarchy, where a parent server may depend on several child servers. Child servers provide services to their parent, which may perform some aggregate function; the parent may itself have a parent, to which it provides services, and so on.

This cooperative, 'building block' approach facilitates a separation of concerns; resulting in simple components from which complex systems can be built.

Client Framework

Client applications are built using a *Client Framework*.

Client applications are the focus of general observatory operators and users. This is where users interface with the observatory. So client applications will usually perform several functions:

- sending commands to / receiving replies from servers;
- displaying server state and responding to server state changes.

The display of server state is by a mechanism called Subscribe / Publish. The subscribing client asks for server state to be delivered at specified intervals, upon which the data is repeatedly published, arriving at the client without the need to keep asking. This asynchronous approach is much more efficient than simply polling for data. It should be noted that efficient network communications are important given the distributed, network-focused nature of the Observatory Control System.

Network Interfaces

In addition to the abstraction which hides the complexities of devices, the network provides its own abstractions – these hide the additional complexity of communicating with devices over the network. The result is that whatever the nature of a device or its location on the network, it can be accessed in a simple manner which is consistent for all devices.

All parts of the Observatory Control System employ the same, universal abstractions to facilitate end-to-end communication between client applications and the device servers which host the observatory devices.

Observatory Software

Observatory software includes client and server applications and automation applications that meet general observatory requirements and specific customer requirements. Domain and problem-specific control system functions are implemented at this level.

Like all parts of the Observatory Control System, this software is built upon a common set of software Frameworks.

It is at this level that the Observatory Control System can be customised, even by customers, with support from EOS if necessary.

Servers

Observatory-level servers will typically drive the hardware and software devices specific to any given observatory.

Clients

Client applications will provide an interface that hides the complexity of the underlying equipment, and present it in terms that end-users and operators are likely to understand.

Automation Functions

Observatory software provides system automation functions at many levels, including:

- device management – automatic management of device behaviour and state by device servers;
- scripted tasks – scripting of automation functions;
- task scheduling – scheduling of robotic tasks to be executed continuously, at scheduled intervals or in response to system events;
- closed loop control – automatic execution of system functions in response to changes in system state.

Case Study – Mount Stromlo

The Mount Stromlo facility contains two ranging systems, for satellite ranging and space debris ranging. These systems have common, but mostly different requirements, and have shared and dedicated components. But both systems were built sharing a single instance of the Observatory Control System.

This integration presented no significant problems or difficulties. Furthermore, no problems are foreseen concerning extensive capability upgrades over the next year. So the Observatory Control System provides technical certainty in terms of EOS' ability to extend and enhance its observatory systems.

Conclusion

The Observatory Control System supports EOS' demanding technical and business requirements. It has evolved over many years and continues to do so. The next evolution may well be a network of stations, where each station is a cooperating instance of the Observatory Control System. This will enable highly coordinated, world-wide observation and ranging programmes.

Currently EOS provides the Observatory Control System with its telescopes and enclosures. There is nothing inherently necessary about EOS' equipment, however; so long as basic requirements are met, the Observatory Control System can operate with any vendor's equipment. So in future EOS may offer the Observatory Control System as a stand-alone product, independent of its telescopes and enclosures.

Finally, the underlying control system is domain-independent. There are no astronomical or observatory-specific concepts embedded in the fundamental control system. So it is plausible that it could be used to drive a range of automated facilities.