PRELIMINARY DESIGN OF THE AUSTRALIAN SKA PATHFINDER (ASKAP) TELESCOPE CONTROL SYSTEM

J.C. Guzman, CSIRO/ATNF, Sydney, Australia

Abstract

The Australian SKA Pathfinder (ASKAP) is a 1% Square Kilometre Array (SKA) pathfinder radio telescope, comprising of 36 12-metre diameter reflector antennas, each with a Focal Plane Array consisting of approximately 100 dual-polarised elements operating at centimetre wavelengths and yielding a wide field-of-view (FOV) on the sky of about 30 square degrees. ASKAP is currently under construction and will be located in the remote radio-quiet mid-west region of Western Australia. It is expected to be fully operational in 2013. Key requirements for the ASKAP control system include: control and monitoring of widely distributed devices, handling of a large number of monitoring points (approx. 150,000), accurate time synchronisation and remote semi-automated operations. After evaluating several software technologies we have decided to use the EPICS framework for the Telescope Operating System and the Internet Communications Engine (ICE) middleware for the high-level control message bus. This paper presents a preliminary design of the ASKAP control system as well as describing why we have chosen EPICS and ICE and how both technologies fit in the overall ASKAP software architecture.

ASKAP THE PROJECT

The future of cm and m-wave astronomy lies with the Square Kilometre Array (SKA), a telescope under development by a consortium of 19 countries. The SKA will be 50 times more sensitive than any existing radio facility. A majority of the key science for the SKA will be addressed through large-area imaging of the Universe at frequencies from 300 MHz to a few GHz. For more information about SKA, visit [1] and [2].

ASKAP is a next generation radio telescope on the strategic pathway towards the staged development of the Square Kilometre Array (SKA). ASKAP has four goals, namely:

- To carry out world-class, groundbreaking observations directly relevant to the SKA Key Science Projects
- To demonstrate and prototype the technologies for the mid-frequency SKA, including field-of-view enhancement by focal-plane phased arrays on new-technology 12-metre class parabolic reflectors
- To establish a site for radio astronomy in Western Australia where observations can be carried out free from the harmful effects of radio interference.
- To establish a user community for SKA.

ASKAP will be located in Boolardy, a remote outback mid-west region of Western Australia, approximately 400 km northeast of Geraldton and 800 km north of Perth. The location also corresponds to the Australian SKA candidate core site. This region has been identified as ideal for a new radio observatory. The population is very small and hence there is a lack of man-made radio signals that would otherwise interfere with weak astronomical signals.

ASKAP will be mainly a survey telescope carrying systematic (unattended) routine observations of the entire southern sky. However a small amount of time will be allocated for targeted observations. For more information about ASKAP project visit [3].

Major challenges in terms of software development are:

- Support for remote operations: Science operations and main control room will be located in Sydney (~4,000 km from Boolardy). Technical/maintenance operations and data processing will be located in Geraldton.
- Very large data flow: ASKAP will produce approx. 10 TB/hr of visibility data (full spectral resolution). Thus we cannot afford to store raw observed data.
- Parallel and distributed processing is vital: There is a need to run calibration and imaging pipelines in High Performance Computers (HPC).
- Limited computing staff: Approximately 60 FTE planned for mid-2006 to 2013.

ASKAP SOFTWARE ARCHITECTURE

Our Architecture Goals

The philosophy behind our system decomposition into software components took into account three priorities:

- Majority of components should be deployed in Geraldton. Only the strictly necessary functionality should be deployed in Boolardy because maintenance and support is more difficult and costly at the remote site.
- Narrow interfaces: Services provided by each component should be well defined and as simple as possible. Dependencies between components should be minimal.
- Loosely coupled: The ASKAP software system must be flexible and scalable both in terms of development, deployment and maintenance. Requirements are expected to change as more is learned about the system, the science that will be done with it, and the manner in which the system will be operated. A key to fulfilling this goal is loose coupling, where dependencies are minimised and modifications have minimal effect on the system as a whole. At a minimum, components should be loosely coupled in terms of hardware platforms, programming languages (C++, Java and Python) and even its implementation.

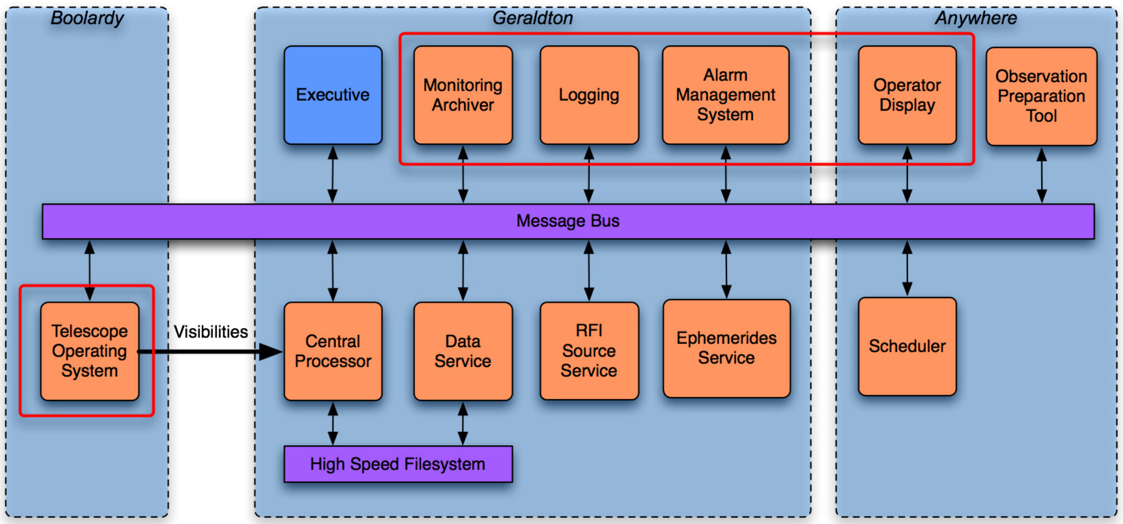

Figure 1: Top-level logical view of the ASKAP system. The red boxes group the components commonly known in other projects as the Monitoring and Control (M&C) System.

Logical View

We have decomposed the ASKAP software system into several top-level components as shown in Figure 1:

- Executive: Responsible for orchestrating an observation.
- Scheduler: Responsible for scheduling the observation for execution by the Executive component.
- Telescope Operating System (TOS): Responsible for monitor and control of the physical telescope. This component will be deployed in Boolardy.
- Central Processor (CP): Responsible for processing observations into scientific products.
- Data Service: Responsible for permanent and temporary storage of data shared by the online components.
- Monitoring Archiver: Responsible for archiving monitoring data generated in the system.
- Logging: Responsible for log messages generated in the system.
- Alarm Management System: Responsible for managing/escalating alarm conditions in the system.
- Operator Display: Responsible for presenting a User Interface for control and monitoring of the instrument by an operator.
- Ephemeris Service: Responsible for providing ephemeris.
- Radio Frequency Interference (RFI) Service: Responsible for identifying any RFI, which may impact the execution of the observation.

Top-level Message Bus

The communication between software components is done through a message bus, as shown in Figure 1. It is clear that nowadays many technologies exist that provide this type of communication infrastructure. These technologies are commonly known as **middleware**. Our requirements for the middleware are:

- Multiplatform: both client and/or server running in Linux and MacOSX.
- Language bindings for C++, Java and Python.
- Support for request/response type of communication.
- Support for publish/subscribe type of communication.
- Promote loose coupling.
- Support for fault tolerance (replication, etc.).
- Open source.
- Mature project.

Three alternatives were evaluated in early 2009: Apache Tuscany ([4]), Apache ActiveMQ ([5]) and ICE ([6]) The Apache Tuscany project looks promising and fulfils many of the requirements listed above but it was found too immature for our needs, in particular the support for C++ bindings ActiveMQ and ICE fulfil many of the requirements of our message bus. However, we have selected ICE over ActiveMQ primarily because of its Interface Definition Language (IDL). The benefits of having a strict IDL are:

- Eliminates ambiguity.
- Avoids us having to define our own IDL between software components.
- Avoids us having to build our own bindings between the programming language and the IDL.

Another reason that pushed ICE over ActiveMQ for us was that many of our interfaces appear to be best suited to an object oriented model.

THE TELESCOPE OPERATING SYSTEM (TOS)

TOS Responsibilities

The main functions of the TOS are:

- Provides a narrow interface to other components mainly involved in observations.
- Coordinates the operation of many distributed and heterogeneous hardware subsystems. All hardware subsystems are located in Boolardy and include: Antenna drives, receivers, local oscillators, digitisers, beamformers, correlator, event generators, time and signal distribution and environmental sensors (weather, lightning, power generator, humidity, room temperature, etc.).
- Coordinates the correct time synchronisation between hardware subsystems.
- Captures telescope monitoring data and generates the visibility's meta-data for data processing by CP component. Publishing of meta-data is synchronised with integration cycle.
- Handles safety operations with the instrument such as automatic wind stowing, network failures, etc.

Evaluation of Control Software Frameworks

In 2008 we evaluated four control software frameworks ([7]) to be used in the implementation of the TOS and its subsystems: PVSS-II ([8]), EPICS ([9]), Alma Common Software (ACS) ([10]) and TANGO ([11]). All the software frameworks reviewed offer many of the technical aspects required by ASKAP and in general any monitoring and control system. The conclusion of the evaluation was that building the control software infrastructure (communication middleware, real-time database, alarm system, logging, etc.) from scratch in-house is risky, not only in terms of timeframes and cost but also in terms of maintainability and reliability. Using existing software frameworks simplifies development and maintenance of complex control system. Many astronomical observatories and high-energy facilities have adopted commercial or non-commercial software frameworks for building their control systems.

The result of the evaluation also selected EPICS as the framework to implement the ASKAP TOS. The main reasons are:

- It is free, open source and with a very active community; and experience within Australia (Australian Synchrotron).
- Both clients and servers can run in many platforms; not only in VxWorks.
- Proven technology: EPICS have been used for more than 15 years.
- Proven scalability. There are several facilities using EPICS supporting larger number of I/O points.
- All the client needs to know is the PV name. No messing around with fixed addresses.
- Lots of software tools available on the web.
- Real-time database design appeals also to non-programmers.
- Presents a unified interface to high-level control (easier integration).
- Provide common software for hardware subsystems developers.

The evaluation report also presented some limitations of EPICS:

- It is mainly used for soft real-time control system (suggested maximum control loop rate via Channel Access is 20 Hz). This is not an issue for us since hard real-time is done via hardware.
- Limited data types in Channel Access protocol. For ASKAP there is no requirement to have a single software framework for the entire data flow system. Complex data types are needed at the top-level where we will use a different communication middleware (ICE).
- Very "narrow" client/server API with no support of request/response-type of communication. This is not an issue for us since the operation of the control system is mainly asynchronous.

CURRENT STATUS

We have developed an EPICS-based system for the Parkes Testbed prototype. This is a 12-m single-antenna located in Parkes Observatory and used mainly for studying the performance of focal plane Phase Array Feeds (PAF). Currently the antenna is equipped with a 40-element PAF. EPICS software is used in all the low-level control: antenna drives, local oscillators, environmental monitoring, digitiser and beamformer. Python is used to implement the observation scripts and to interface to EPICS I/O controllers (IOCs) via third-party Python bindings.

In terms of the overall ASKAP software, the Preliminary Design Review was passed successfully early 2009 and we expect to have a Critical Design Review in March 2010.

The first ASKAP antenna will be installed on site in late 2009. A six-antenna interferometer called Boolardy Engineering Test Array (BETA) will start early science commissioning in 2011. Full ASKAP will be operational in 2013.

REFERENCES

[1] Australian and NZ SKA: http://www.ska.gov.au
[2] SKA: http://www.skatelescope.org
[3] ASKAP: http://www.atnf.csiro.au/projects/askap
[4] Apache Tuscany project: http://tuscany.apache.org
[5] Apache ActiveMQ: http://tuscany.apache.org
[6] ICE: http://www.zeroc.com/ice.html
[7] J.C.Guzman, "Evaluation of Software Frameworks for the ASKAP Monitoring and Control System", ASKAP-SW-0002, CSIRO, 2008.
[8] PVSS-II: http://www.pvss.com
[9] EPICS: http://www.aps.anl.gov/epics
[10] ACC: http://www.eso.org/~almamgr/AlmaAcs
[11] TANGO: http://www.tango-controls.org