The ALMA Common Software (ACS): Status and Developments

G.Chiozzi, B.Jeram, H.Sommer, R.Georgieva (ESO)
M.Plesko, M.Sekoranja, I.Verstovsek, D.Vitas, K.Zagar (JSI and Cosylab)
D.Fugate (NRAO) R.Cirami, P.DiMarcantonio (AOT)

OVERVIEW
The ALMA Common Software (ACS) is a set of application frameworks built on top of CORBA to provide a common software infrastructure to all partners in the ALMA collaboration. The main purpose of ACS is to simplify the development of distributed applications by hiding the complexity of the CORBA Middleware and guiding the developers to use a documented collection of proven design patterns.

ACS was presented at ICALEPCS 2001 and was at that time covering the basic needs for the development of Control System applications. In these two years, the core services provided by ACS have been extended and made stable and reliable, while the coverage of the application framework has been extended to satisfy the needs of high level and data flow applications.

At the same time, the focus of development has moved from C++ to Java.

1) Real Time and Control System support
ACS was first developed to satisfy the requirements of the Control Software development, to support:
- the ALMA Test Interferometer Control Software, used for the evaluation of the 3 prototype antennas.
- the ANKA Control System
Since ACS 0.0 (that demonstrated ACS capabilities driving the Kitt Pak 12 meter telescope) we are supporting C++ Linux, Sun and VxWorks. The ANKA accelerator is running ACS on Microsoft Windows workstations.
Development of Control System devices is supported by the framework with the implementation of the Component/Property/Characteristic design pattern.
In 2004 ACS support for control system applications should receive a boost from the eACS (embedded ACS) project. A consortium from the astronomical and accelerator communities and industrial partners is studying the implementation of solutions based on ACS and Abeans for embedded platforms such as PC/104 and CEP (Custom Embedded Platform).

2) Abstract Hardware interface
Since ACS 1.1, the Component/Property/Characteristic model provides an abstract interface to the hardware with the implementation of DevIO classes. The actual interface to the hardware (for example access for IO boards, CAN bus, serial ports) is implemented as a subclass of the abstract DevIO interface.
Properties use only the abstract interface to provide read and write access to values in the hardware as well as monitoring and alarm capabilities.

3) Decoupled Publisher/Subscriber design pattern
Since ACS 1.1, a layer on top of the CORBA Notification Channel provides support for Publisher/Subscriber programming model. This is extensively used to notify ALMA subsystems of events occurring in other subsystems and to drive the flow of data. ACS 2.0 and 3.0 have provided extensions to this framework.

4) C++ versus Java
While C++ remains the language of choice for high performance and real time applications in the Control System domain, Java is considered the most suitable general purpose development language for higher level and coordination applications, also in the Control System domain.
ACS 2.0 has introduced Java Containers.

5) Administration
Deployment, system configuration and administration are supported by the ACS Component/Container model.
The original model in ACS 0.0 and 1.0 was tailored to Control System applications. Only C++ Components and Containers were supported.
ACS 2.0 extends the model based on the requirements of high level subsystems and introduces Java Components and Containers.
ACS 3.0 introduces fully dynamic Components to support pipeline and AIPS++ requirements. It also introduces support for Python Components and Containers.
Administration of Components and Containers is transparent to the implementation language. A set of Tools and GUIs allow an operator to administrate the system.

6) General services
Containers written in C++ (ACS 0.0), Java (ACS 2.0) and Python (ACS 3.0) manage the lifecycle of components implemented in these languages and provide them a very simple way to access common centralized services like logging, alarms, error handling, configuration database, archive, object location and, at the same time, hide most of CORBA. Clients written in any CORBA aware language can access these Containers and Components while the implementation of the server side in any of these languages would be.

ACS Collaboration
ACS is developed for the ALMA Project and made available under the GNU LGPL Licence.
The development is distributed among the sites of various ALMA partners and external institutes collaborating in the development.
A number of external projects are already using ACS or are evaluating the possibility of using it.
This map shows the major sites involved in ACS Development, the ACS installations and some external projects using or evaluating ACS.
The availability of ACS can trigger other collaboration projects, like eACS (embedded ACS).

7) Pure Java ACS
Requirements from the Observing Tool development team have pushed us in providing High Level multi-platform ACS support, to deploy ACS-aware applications with little configuration requirements and support directly on the PCs of astronomers.
With ACS 3.0, a "pure Java" sub-set of the ACS framework is available for easy deployment with the WebStart technology. This allows to have most high level ACS features available in a pure Java environment, where specific functionality available only in C++ of Python is not required.
This includes Abeans 3.0 and allows to deploy from the web ACS applications, GUIs and tools on any platform supporting a Java Virtual Machine.

8) XML Serialization
Since ACS 2.0, the Java Container supports transparent XML serialization of complex data entities (like a complete Observing Proposal or an Observing Script) through CORBA. Binding classes are automatically generated so that data entities are accessed through native language classes and (de-)serialized transparently on the wire. The archive is capable of handling arbitrary XML serialized data entities.
This capability (XML Serialization) is very important to allow a smooth data flow from high level software down to the Control System.

Web page: http://www.eso.org/projects/alma/develop/acs