THE ALMA COMMON SOFTWARE (ACS): STATUS AND DEVELOPMENTS

G. Chiozzi, B. Jeram, H. Sommer, R. Georgieva, ESO, Garching bei Muenchen, DE
M. Sekoranja, I. Verstovsek, D. Vitas, K. Zagar, JSI and Cosylab, Ljubljana, SI
D. Fugate, NRAO, Socorro, NM, USA
R. Cirami, P. DiMarcantonio INAF-AOT, Trieste, IT

Abstract

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 [1,3].

The main purpose of ACS is to simplify the development of distributed applications by hiding the complexity of the CORBA Middleware and guiding developers to use a documented collection of proven design patterns.

ACS was presented at ICALEPCS 2001 [4] and at that time covered the basic needs for the development of Control System applications. In these two years, the core services provided by ACS have been extended and become more stable and reliable, while the coverage of the application framework has been extended to satisfy the need of high level and data flow applications.

This paper presents the status of ACS and developments over the last two years, but we will not go into technical details.

An overview of ACS technical features can be found on the paper [5] from K.Zagar at this years conference. Details on ACS architecture and design can be found in the ACS Web Page [2].

INTRODUCTION

ACS has a release cycle of 6 months, with alternating major and minor releases. At each release we add new features according to the ACS Development Plan which is based on requests from the other subsystems of the ALMA project.

The major releases also provide an occasion to cleanup and harmonize interfaces. Actually, ACS 3.0 (the release currently in its final integration) includes a major re-factoring of all ACS interfaces.

The contents of each ACS release is decided upon in a meeting at the beginning of each release cycle. All other ALMA subsystems leaders attend, allowing us to set priorities in an agile way.

A strong cooperation with the ALMA High Level Analysis group makes sure that we have a consistent architecture for ACS and the whole ALMA.

This process is very efficient and is allowing us to respond promptly to requests, converging at the same time toward a coherent and stable ACS design.

SUPPORT FOR MULTIPLE PROGRAMMING LANGUAGES

ACS 1.0, back in 2001, was concentrating mainly on Control System development, because our main "customer" was the Control System for the ALMA Test Interferometer. C++ was the chosen language for the development of control devices.

The focus of development has moved now from C++ to Java, with Python's role growing. 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. At the same time Python is very good for writing prototypes and scripts.

Now ACS provides mostly the same features in all three languages. An exception is the Component/ Property/Characteristic design pattern that is specifically designed for Control System development and is currently available only as a C++ framework.

COMPONENT CONTAINER MODEL

Using standard CORBA services, ACS implements a Component/Container model that is language and platform independent. The original MACI model described in [4] was implemented in C++ and adequate for the static structure of a control system.

ACS 3.0 contain a major re-factoring of this model, mapping the terminology and concepts of other mainstream Component/Container models from the world of business applications.

Containers can now be in C++, Java, and Python. They manage the lifecycle of Components implemented in all these languages and provide them with a very simple way to access common centralized services such as logging, alarms, error handling, configuration database, archive, object location, and at the same time, hiding most CORBA.

The Manager that administers the whole system has been re-implemented in Java for portability and maintenance reasons.

The Manager available with the previous versions was only capable of handling Components fully specified in the configuration database. This followed the (quasi-)static model of a control system, where for example, the number of Power Supplies is fixed and they are statically configured. The new Manager is also capable of handling fully dynamic components to satisfy the requirement of higher level sub-systems for software components whose number and configuration varies according to the usage of the system. An example is a component performing FFT transforms in n image pipeline: if we have many users performing pipelines, we may need more instances of this component.

Future versions of ACS will address security and scalability issues for the implementation of systems that are distributed across multiple sites and continents.

XML SERIALISATION

For the Java programming language, the Container integrates transparently the use of type-safe Java binding classes (like a complete Observing Proposal or an Observing Script) to let applications conveniently work with XML transfer objects without having to parse or serialize them [6]. This capability is very important to allow a smooth data flow from high level software down to the Control System.

"PURE JAVA" ACS

To allow "pure Java" development and deployment on any platform where a Java virtual Machine is available, a Java-ACS package provides the possibility of installing only the Java components anywhere a JVM is available. Otherwise, the complete ACS package provides Java, C++ and Python support on Linux and limited support on other selected platforms.

This capability is important to provide a light run-time and development environment for high level applications.

With ALMA, ACS will reach the desk of astronomers working with the observing preparation tools. Also, all Java GUIs for the control and administration of the system will be able to run on any machine without any specific requirements. At the same time, Java-only ACS is integrated with the entire system (an example is the capability of accessing the central logging system) so you don't really lose any functionality.

To make deployment easy, we use Java WebStart [7] technology which allows installing and updating the whole Java ACS simply by accessing a web site.

INTERFACE TO HARDWARE

As far as interfacing to hardware is concerned, we have introduced a generic abstraction layer between the Property and the hardware monitor and control points in the Component/PropertyCharacteristic model (BACI in [4]).

The DevIO abstraction [8] keeps a fully generic definition of Properties, provides to the application synchronous and asynchronous access to the value of monitor and control points, and contains features like telemetry logging and alarms.

For example, ACS high level code controlling a Power Supply is only aware of the SetPointCurrent control point and ReadbackCurrent monitor point. If the actual Power Supply is controlled via an analog I/O board, the properties are configured with a DevIO capable of reading and setting the channels of the board. If the Power Supply is replaced by a new one controlled via serial interface, only the DevIO implementation needs to be replaced.

ACS USER'S BASE

ACS is used as the common middleware for the whole ALMA software development.

The Control System of the ALMA Test Interferometer is based on ACS and is being used for the evaluation of the three American, European, and Japanese ALMA prototype antennas. The three antennas are being evaluated at the VLA site in New Mexico.

All other ALMA subsystems, from Observation Preparation to Pipeline and Calibration, are under development and their requirements are driving the development of new ACS features.

This accounts for many ACS installations at the sites of all ALMA partners in Europe, USA and Japan.

Also other projects are collaborating with ACS, already using or evaluating it, since ACS is publicly available under the LGPL license.

To be specific, the ANKA Synchrotron in Karlsruhe is in scientific production, the APEX radio-telescope in Chile is under commissioning and the 1.5m Hexapod Telescope in Chile is in an advanced implementation stage.

ESO and NRAO are evaluating ACS for the upgrade of already existing systems and for new development, as well as other external institutions.

A consortium of research institutes from the astronomical and accelerator communities along with industrial partners are evaluating the possibility of developing an ACS for embedded systems (eACS) to complement ACS with specific support for low-priced, mass-produced embedded controllers.

CONCLUSION

The development of ACS since ICALEPCS 2001 has been very fast. ALMA requirements have driven the development in the direction of Java and Python to support high level software beyond the traditional domain of Control Software. This allows developing the whole software for a scientific facility in a harmonic way.

While this high level software is being developed, we have real Control Systems running based on ACS so that its performance and reliability can be verified and improved where necessary.

The fact that ACS is publicly available and other projects have shown interest or are already using it is a big advantage. This is because we get independent, unbiased feedback, and the framework becomes naturally more general.

At the same time, we pay attention to focus as much as possible to ACS development on our "core business" (i.e., satisfying the requirements of the ALMA Project). This avoids the mistake of developing a framework that is too general and trying to satisfy everybody does not make anyone really happy.

ACKNOWLEDGEMENTS

The ACS project is managed by ESO in collaboration with JSI INAF-AOT. This work is the result of many hours of discussions, test and development inside our groups and in the various ALMA centers at NRAO, IRAM and Bochum. We thank all our colleagues in ALMA and in other projects using ACS for the important contributions to the definition and implementation of ACS.

REFERENCES

[1] ALMA Web page, http://www.mma.nrao.edu/

[2] ACS Web Page, http://www.eso.org/projects/alma/develop/acs

[3] G. Raffi, B. Glendenning, “ALMA Software Development Approach”, ICALEPCS 2003, Gyeongju, Korea, October 2003

[4] G. Chiozzi, et al. “Common Software for the ALMA project”, ICALEPCS 2001, San Jose, California, November 2001

[5] K. Zagar, et al., “ACS – Overview of Technical Features”, ICALEPCS 2003, Gyeongju, Korea, October 2003

[6] H. Sommer, et al., “Transparent XML Binding using the ACS Container/Component Framework”, ADASS 2003, Strasbourg, France, October 2003

[7] Java Web Start home page, http://java.sun.com/products/javawebstart/

[8] B. Jeram, et al., “Generic Abstraction of Hardware Control Based on the ALMA Common Software”, ADASS 2003, Strasbourg, France, October 2003