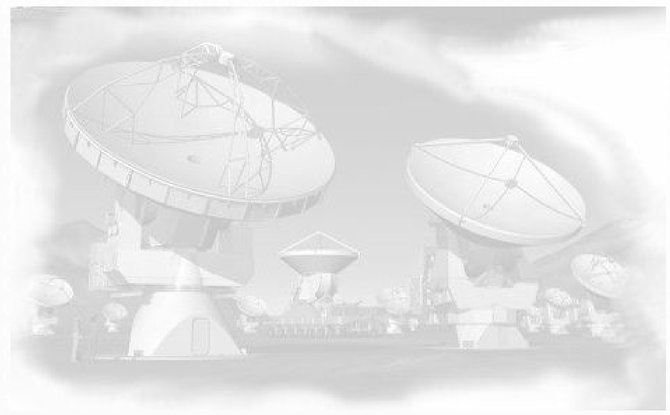
ALMA Common Software (ACS) Status and Development

G.Chiozzi, A.Caproni, B.Jeram, H.Sommer, J.Schwarz (ESO, Garching bei Muenchen), M.Sekoranja (Cosylab, Ljubljana), R.Cirami (INAF-OAT, Trieste), H.Yatagai (NAOJ, Tokyo), J.A.Avarias (NRAO, Socorro, New Mexico), A.Hoffstadt, J.Lopez (UTFSM, Valparaíso), N.Troncoso (ALMA, Santiago, Chile) A.Grimstrup (University of Calgary, Calgary, Alberta)



Abstract

ACS provides the infrastructure for the software of the Atacama Large Millimeter Array and other projects. Using CORBA middleware, ACS supports the development of component-based software, from high-level user interfaces down to the hardware device level. It hides the complexity of CORBA beneath an API that allows the application developer to focus on domain-specific programming.

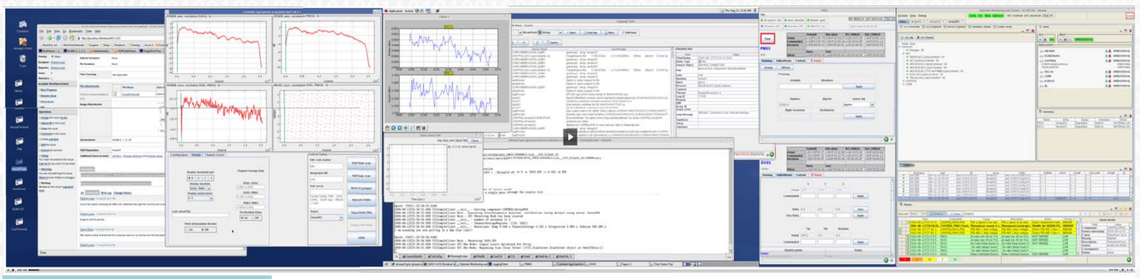
Although ACS, now at release 8, has been used operationally by the APEX radio telescope and at the ALMA Test Facility for several years, the commissioning of ALMA in Chile brings major challenges: new hardware, remote operation and, most important, up-scaling from 2 to 60+ antennas.

Work now turns to scalability and improving the tools to simplify remote debugging. To further identify potential problems, the University of Eindhoven is formally analysing ACS.

Meanwhile, new developments are underway, both to respond to newly identified needs of ALMA, and those of other projects planning to use ACS. Examples include the refactoring of the interface to the CORBA Notify Service, integration with the Data Distribution Service, generation of state machine code from abstract models and of Python binding classes from XML schema.

What is ACS?

ACS is a software infrastructure and framework for the development of distributed systems based on the Component/Container paradigm



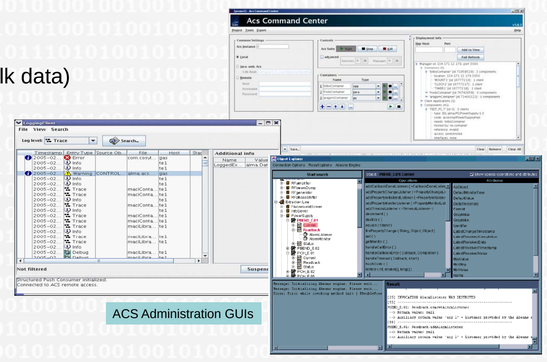
ALMA console at the OSF

The ACS framework is based on CORBA and built on top of free CORBA implementations. ACS partially wraps CORBA to hide its complexity and to make it easy to implement applications following standardized architecture and design patterns.

Free software is extensively used wherever possible, to avoid “re-inventing the wheel”. Commonly used libraries and tools are integrated in the ACS distribution to ensure that a coherent and homogeneous package is available to all developers

ACS provides the basic services needed for object oriented distributed computing using different programming languages. Among these are:

- Transparent remote object invocation
- Publisher/subscriber paradigm (low bandwidth and bulk data)
- System deployment/administration and object location based on a container/component model
- Distributed error and alarm handling
- Distributed logging
- Configuration database
- Thread management
- XML binding classes and transparent serialization
- Simulation facilities
- Standardized testing infrastructure



ACS Administration GUIs

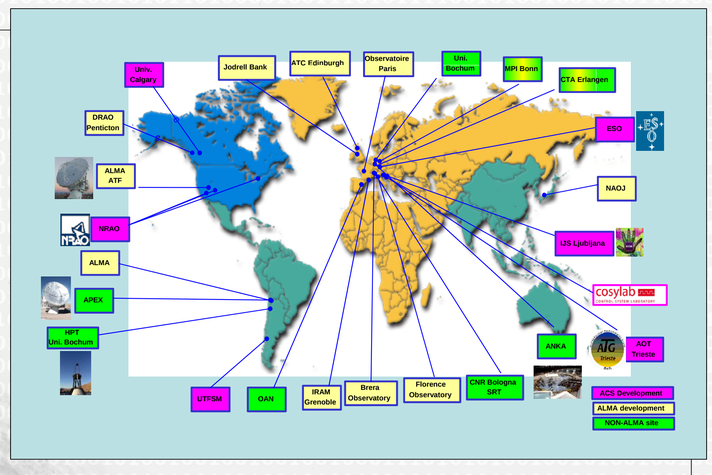
ACS’s primary platforms are Red-Hat Enterprise and Scientific Linux, but it works and is used also on other Linux variants. A Windows version is used by other projects and work is being done to provide a complete Windows support. Real time development is supported on Real Time Linux (for ALMA) and VxWorks (for the APEX project).

Development APIs are available in C++, Java and Python. Any other language with a CORBA mapping can be used, if needed. Coherent support of multiple programming languages is one of the key motivations for the implementation of ACS.

Who is using ACS?

ACS is used end-to-end in the whole ALMA project. The antennas at the ALMA Test Facility in New Mexico have been running with ACS for several years. Several antennas are now being commissioned in Chile with scientific operation foreseen to begin in 2012.

Being available under the GNU LGPL license, ACS has been adopted also by several other projects , mainly in astronomy.



Among these:

- APEX, mm/submm radio telescope in scientific operation in Chile (http://www.apex-telescope.org/)
- HPT, optical telescope in scientific operation in Chile (http://de.wikipedia.org/wiki/Hexapod-Teleskop)
- 40m OAN radio telescope in Spain under commissioning
- Sardinian Radio Telescope, under construction in Italy

Other projects are considering the use of ACS, e.g., the ESO E-ELT, the ESO SPARTA AO system, the Cherenkov Telescope Array.

There is also a strong and growing community of ACS users in Chile, with the core in the ACS Group at the UTFSM University in Valparaiso (https://csrg.inf.utfsm.cl/twiki4/bin/view/ACS/AcsUtfsmAbout) and with several projects in other universities. This group is very active in the development of new feature for ACS and makes available expertise to the organizations using ACS in Chile.

Commercial support is provided by Cosylab (www.cosylab.si), which is heavily involved in ACS development since the beginning. All in all, there are probably about 200 ACS users/installations worldwide.

Want to know more???

The ACS Web page:

- http://www.eso.org/projects/alma/develop/acs With documentation , downloads, papers….

The next ACS Workshop and Course:

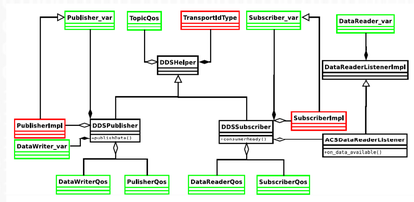
- UTFSM in Valparaiso, Chile - 09th - 13th of November 2009 (http://acsworkshop.inf.utfsm.cl/2008/

Related papers in this conference:

- Data Distribution Service as an alternative to CORBA Notify Service for the ALMA Common Software , J. A. Avarias (NRAO), H.Sommer (ESO), G.Chiozzi (ESO) - WEA006
- ALMA Software Project Management – Lessons Learned G. Raffi (ESO), B.E. Glendenning (NRAO)

New development: DDS

DDS can replace the CORBA Notify Service in the ACS Notification Channel providing better performance and more control



ACS provides support for the publisher/subscriber paradigm through the ACS Notification Channel service

The current implementation is based on the CORBA Notify Service and uses the TAO Notify Service implementation, hiding as much as possible of the CORBA complexity from the developers. The APIs are available for the three programming languages supported by ACS: C++, Java and Python.

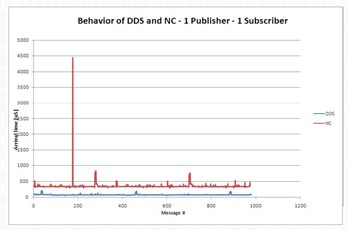
Although the present system fully satisfies all requirements of the ALMA project, the Notify Service has some limitations, being resource intensive and not scaling well with the number of subscribers.

The Data Distribution Service (DDS, http://www.omgwiki.org/dds) is an OMG open international middleware standard (as well as CORBA) directly addressing publish-subscribe communications for real-time and embedded systems. With respect to CORBA Notify, it offers better performance and features decentralized message processing, scalable peer-to-peer communication, and a wide set of QoS policies.

For ACS we have implemented a prototype of the same ACS Notification Channel APIs using OpenDDS and used it for evaluation in the E-ELT Demonstrator . We have also been evaluating RTI DDS

It is possible to switch transparently between the two implementations, so that a good comparison of performance and reliability is possible.

This implementation seems very well suited for intensive data-centric applications where the publisher/subscriber model allows keeping much better de-coupling between the various components of an application with respect to a more traditional client/server architecture.



More details on DDS for ACS are presented in another paper in this conference:

- Data Distribution Service as an alternative to CORBA Notify Service for the ALMA Common Software , - WEA006

New development: Model Driven Development

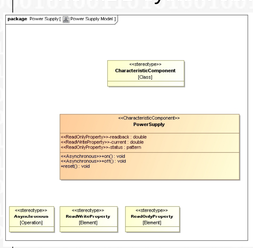
Modern control systems involve multiple subsystems and devices. During the lifetime of the project, the design is often subject to changes and keeping the models and the implementation aligned is a time consuming activity often neglected.

Moreover, many of the steps needed to move from design to implementation involve a lot of repetitive code copying activities, in particular when using a comprehensive application framework like ACS.

A better approach is to use model-driven development techniques, where the code is generated automatically by a generator, based on a model. This directly reduces the number of errors per LOC and enforces a single coding style for a large portion of the code base.

Such an approach allows accommodating model changes more easily and it improves productivity, since developers can focus on business logic instead of on implementation details . ACS is very well suited for this approach.

The MDD system currently under development allows:

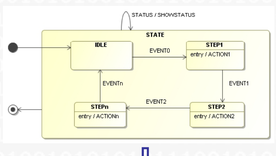


1. Code generation starting from a UML model:a) Generate the IDL (Interface Definition) files. This implies creating a full implementation of the UML model as an IDL file so that it may be compiled by IDL compilers.
   b) Generate (now only in Java) fully functional base class implementation of the applications’ components starting from a class diagram.
   c) Generate the corresponding Configuration Database and deployment information
   d) Generate a basic test suite
2. Integrate design patterns into the code during the generation process.

An advanced prototype is available and has been used to generate simple systems, making it much easier to develop ACS applications

New development: Finite State Machines

… generate FSM from model



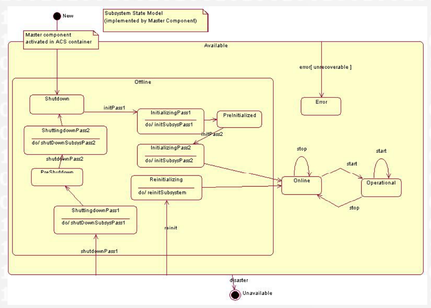
⇩

Executable

Control applications are in principle very naturally mapped into state machines.

Clearly the direct control of physical devices needs to be modeled using finite states machines, but also the high level coordination between the subsystems of a telescope or the sequencing of observations would be very conveniently described using state machines.

Finally, in the last few years some generic ways of specifying and implementing finite state machines have become available.



In recent ACS developments we have adopted the approach of modeling state machines with a UML tool and generating from that the skeleton of a complete application where only the specific code for actions and transitions need to be implemented.

The actual state machine is generated from UML into a general purpose state machine engine, using the OpenArchitectureWare Framework, so that this stage of generation can be used in different application frameworks (like ACS and VLT CCS).

An additional generation step produces all application code needed for the specific application framework adopted by the system.

This strategy allows generating the application for different frameworks from the same model, allowing better reuse of the model.

New development: ACS Daemons

Until ACS 7.0, all ACS services and Containers running on the different nodes of a distributed control system were started by the main startup node by means of remote secure shell sessions. The intelligence on the deployment of the system was therefore centralized on this main node.

Starting with ACS 7.0 we are introducing a new strategy for startup, deployment and system’s management based on daemons deployed on the distributed nodes with the responsibility for monitoring and deploying services and containers.

The central ACS manager delegates to the daemons the task of starting and stopping containers.

The daemons can monitor resources on the nodes, check the life status of all entities under their control, collect statistics on CPU, disk, memory and other resources much better than what the manager and the other ACS administration tool were before able to do using plain SSH sessions.

This approach has significantly improved the reliability of the system, its resilience to problems and the possibility of monitoring its status and diagnosing problems.

New features are being added to the daemons in the upcoming releases, based on the feedback from the commissioning team at the ALMA operation site.