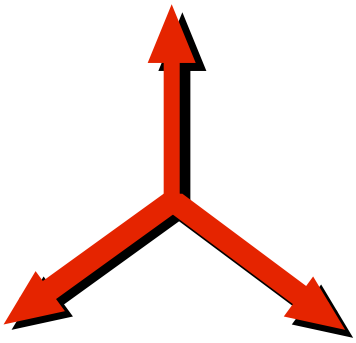
EXPANDED VERY LARGE ARRAY



ALMA Common Software (ACS) Usage in the EVLA

Kevin Ryan

July 9, 2003

NATIONAL RADIO ASTRONOMY OBSERVATORY
P.O. Box 0, Socorro, New Mexico 87801

Operated by Associated Universities, Inc.
Under Contract with the National Science Foundation

Abstract:

Two EVLA Software Engineers attended an eight-day ALMA Common Software (ACS) course for the purpose of evaluating the use of ACS in the EVLA software system.

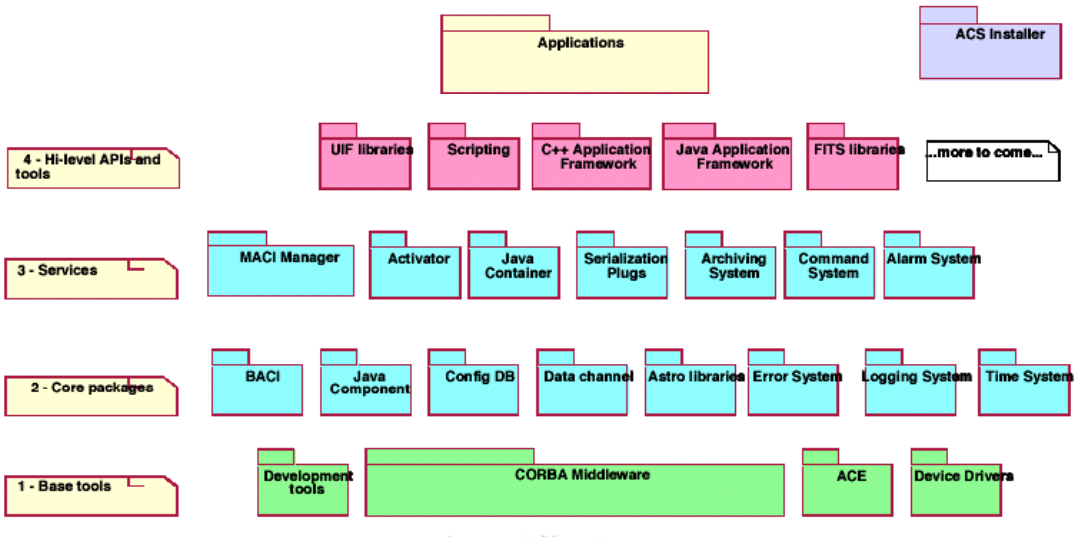
This report gives a brief overview of ACS and discusses its applicability to EVLA Monitor and Control.

1 A Brief Introduction to ACS

The following introduction is mostly taken from **"CORBA-based Common Software for the ALMA project"** G. Chiozzi, B. Gustafsson, B. Jeram, M. Plesko, M. Sekoranja, G. Tkacik, K.Zagar http://www.eso.org/~gchiozzi/AlmaAcs/OtherDocs/spie2002.pdf

ACS is a 'central object oriented framework' based on CORBA created to alleviate the complexities of ALMA Software development caused by 'the wide geographical distribution of its development teams and their diverse development cultures'.

ACS is comprised of a set of core packages and system services layered on top of CORBA.



The core layer contains the system 'components' which are similar to the EVLA software 'Device'. This is the software that represents the physical system that is to be controlled and monitored.

If the component is written in C++, it implements the Basic Access Control Interface (BACI) and instantiates into what is called the Distributed Object (DO). Each DO is further composed of *Properties* that correspond to what are called controlled points, channels or tags in Supervisory Control and Data Acquisition systems (SCADA). Each *Property* is an object too, described by *Characteristics* such as min/max values or units.

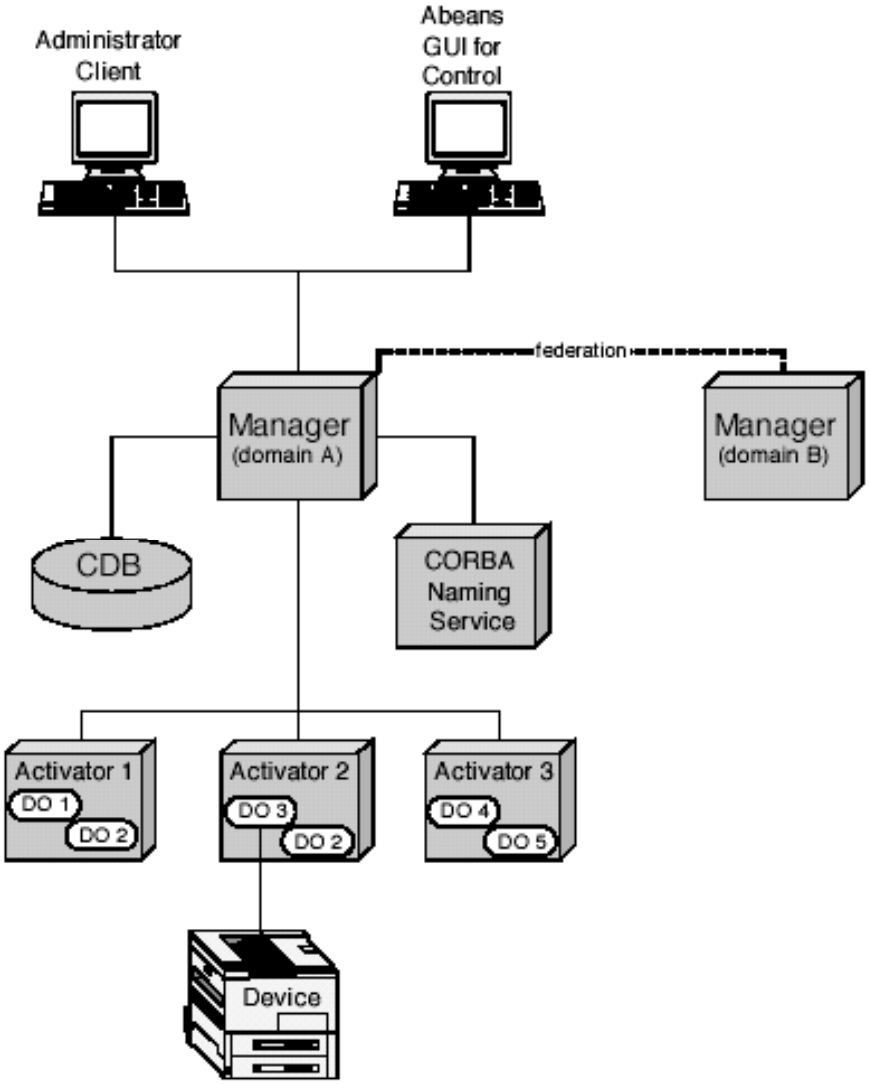
If the component is written in Java, it is simply called a Java Component. ACS developers still consider Java as a slow, non-real-time, high-level language not suited for control applications. As such, Java Components do not "enforce the *Distributed Object, Property and Characteristic* paradigm, which is particularly suited for control system components".

Component properties (like EVLA ControlPoints and MonitorPoints) are defined in the Configuration Database (CDB) that is also part of the Core packages layer.

Components reside in 'containers' that are a part of the Services layer. This is similar to the EVLA DeviceServer (or sometimes called MIB Object) class. In the EVLA, by definition, there is exactly one DeviceServer per physical processor, but in ACS more than one container may exist in a processor.

If a container is written in C++ it is called an Activator; if it is written in Java it is called a Java Container.

The 'glue' logic for most of the packages and services is contained in the Management and Access Control Interface (MACI). MACI contains a Manager that is set up at one central location and is widely known across the entire system. The Manager is acquainted with all of the Containers and Components in the system as well as other resources such as the CDB.



When a distributed processor comes online, it starts its Activator (or Java Container) that, in turn, establishes a connection via CORBA to the Manager. The Manager checks to see which DO's (or Java Components) are hosted by the Activator and orders them brought to life.

Once started, operators may access the DO's via GUI's that first connect to the Manager and request the DO(s) that it wishes to manipulate.

The Manager terminates the DO's lifecycle when it decides that the DO is not being used by anyone and that its presence in the system is 'not crucial'.

2 ACS in the EVLA

2.1 Physical Fit

ACS is big. It will not physically fit into the EVLA at the embedded processor level including the MIB, the CMIB or the CMP processors. It is physically possible for ACS to be used in the AMCS at the Antenna Device and in the CMCS at the Virtual Correlator Interface layers and higher.

2.2 Philosophical Fit

In many respects ACS is the antithesis of the EVLA design philosophy. EVLA is designed to be very modular with loose coupling between modules. The philosophy presented at the ACS course was that loose coupling was undesirable; therefore ACS was designed to be a tightly coupled system. This is seen in the case of the MACI Manager who is responsible not only for each component's access but also even controls their lifecycle.

The EVLA was designed so that access to all components of the system could be direct and symmetrical. This means that every device, no matter what layer it resides in, can be directly accessed by the same means as every other device (thus giving a module its independence from the rest of the system).

ACS is dependent on the network being available for the system to work and has at least two single points of failure (the MACI Manager and the CDB). This would break the EVLA's requirement that the system be operable in at least a default 'safe' mode if the network is down. There are no single points of failure in the EVLA (yet).

Every device in the EVLA will be operable in or out of the system independently. It appears that it would be difficult for a technician to directly connect to a device and operate it on the test bench if the device were under ACS control.

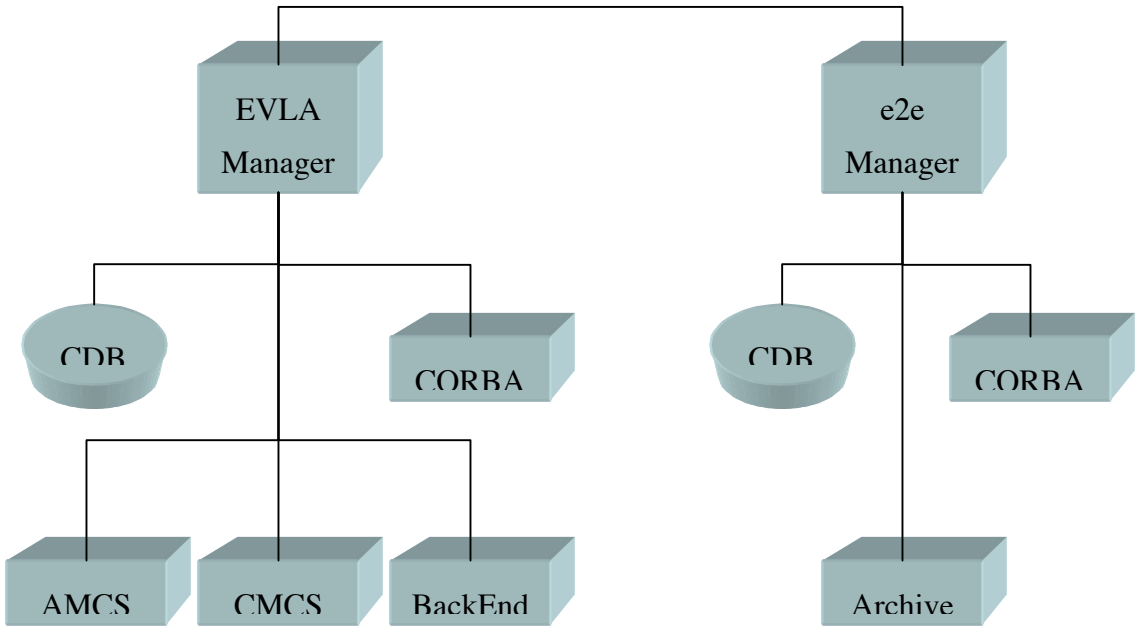
It is opined that the philosophy of the Object Oriented system is lost in the ACS. In the EVLA, a software Device represents each physical component of the system. Because of the modular nature of EVLA each Device's primary 'focus' is the physical device that it represents. It is felt that, in ACS, a device is an ACS component first. During the course it was felt that a larger portion of the functionality of a component was dedicated to ACS than to the part of the actual system that it was supposed to represent.

2.2.1 At the Subsystem (AMCS, CMCS, BackEnd) Layer

Though it is physically possible to represent some components of the AMCS and CMCS in ACS, it is recommended that this not be done, mainly because it would cause a loss of symmetry in the manner in which the components are accessed. An observe script or operator GUI would access an ACS component via MACI but would be required to access other components differently. This implies that the user (GUI or other application) would have to be aware of where the device resides in the system (under ACS control or not).

2.2.2 At the Top (System) Layer

The first feasible EVLA layer that could come under ACS control would then be the EVLA System layer (the layer that represents the whole system of antennas, correlator, correlator back-end and whatever else makes the EVLA).



This would have the advantage that e2e and the rest of the outside world would see the EVLA via ACS. The EVLA would consist of only three ACS components, the AMCS, CMCS and Correlator Backend.

It is not understood however what this would gain for us. e2e could operate the three components via the MACI but it is not really e2e's function to *operate* the EVLA. This would blur the line between real-time and offline systems.

The ACS philsophy is remote object control; a higher-level process manipulates lower-layer objects 'directly' via CORBA object brokering. The loose coupling philosophy of EVLA mandates that higher level processes do not directly control lower level entities but only specifies what they want them to do – leaving the 'doing' to the lower-level entity itself.

For example, e2e would send EVLA an observe script consisting of one or more 'scans' or configuration blocks that define what state the system should configure itself to at specific times in the future. The scripts are parsed at each layer, refined for and sent to each component at the next lower layer until reaching the final destination where it is executed at the specified time. No direct control (remote method invocation) is done between layers in the EVLA. Having e2e operate say the AMCS as a remote object would confuse this approach and might cause the need for subsequent remote object brokering at lower levels.

2.3 Some Good Ideas

2.3.1 Configuration Database

The ACS course did bring to light some things that could be adopted for use in the EVLA; the foremost being the CDB (configuration database). Currently, the EVLA design stipulates individual XML files to contain the ControPoint (CP) and MonitorPoint (MP) data. It makes better sense to adopt the ACS approach that places all of this information into a single database. This has several advantages:

- All of the system configuration information is maintained in one place.
- Access to the data will be afforded by the many standard interfaces available to the database.
- Diagramming tools that access the database could be more easily used to present up-to-date diagrams of the system based on the configuration data.
- Archived data can be related to the state of the system at the time of their archiving.

At least one policy issue will have to be addressed if a common database is used. We will have to resist the temptation to rely on the database as a source of information for the state of the system. The state of the system, including which CP's and MP's are in use and the values of their parameters should be obtained directly from the system itself.

The system should also not rely on the database for its initialization and operation. Each MIB, for example, should have its portion of the database resident in its non-volatile memory.

3 Summary.

ACS represents a completely different philosophy than that of the EVLA. ACS was designed as a tightly coupled system where all components - whether they are low-level hardware controllers or high-level GUI's - operate through a single managerial process that is knowledgeable of everything in the system. EVLA was designed to be a modular system with each module designed to be operable independent of the rest of the system.

ACS is built around remote object control where a higher layer process controls a lower level process by manipulating its objects as if they were local (via object brokering). EVLA is designed to eliminate direct control between layers (mainly to satisfy the requirement that no real-time operations take place over the network). Instead lower layer components in the EVLA are simply told what state they should be in at a specified time and it is left to the sub-component to configure itself at that time.

ACS is a very large system requiring system resources that precludes its use in any of the EVLA's embedded processors. This means that any use of ACS would be at the higher layers or even just at the interface to e2e. Since it is not e2e's function to directly control the EVLA, it makes no sense to incorporate ACS at the higher levels.

It is therefore recommended that we do not use ACS on any part of the EVLA including its interface with e2e.