Study of various Control and Monitoring tools

1. LOFAR

LOFAR operated through an advanced distributed Monitoring and Control system. The system is built upon a commercial SCADA system, PVSS.

LOFAR has an extensive System Health Management function to make the instrument self-diagnosing and (where possible) self-healing.

- ✔ SCADA

SCADA (**supervisory control and data acquisition**) is a type of industrial control system(ICS). Industrial control systems are computer controlled systems that monitor and control industrial processes that exist in the physical world. SCADA systems historically distinguish themselves from other ICS systems by being large scale processes that can include multiple sites, and large distances.[1]These processes include industrial, infrastructure, and facility-based processes, as described below:

➔Industrial processesinclude those of manufacturing, production, power generation,fabrication, and refining, and may run in continuous, batch, repetitive, or discrete modes.

➔Infrastructure processes may be public or private, and include water treatment and distribution, wastewater collection and treatment, oil and gas pipelines, electrical power transmission and distribution,wind farms, civil defense siren systems, and large communication systems.

➔Facility processes occur both in public facilities and private ones, including buildings, airports, ships, and space stations. They monitor and control HVAC, access, and energy consumption.

A SCADA system usually consists of the following subsystems:

- ✗ A human–machine interface or HMI is the apparatus or device which presents process data to a human operator, and through this, the human operator monitors and controls the process.
- ✗ SCADA is used as a safety tool as in lock-out tag-out
- ✗ A supervisory (computer) system, gathering (acquiring) data on the process and sending commands (control) to the process.
- ✗ Remote terminal units(RTUs) connecting to sensors in the process, converting sensor signals to digital data and sending digital data to the supervisory system.
- ✗ Programmable logic controller(PLCs) used as field devices because they are more economical, versatile, flexible, and configurable than special-purpose RTUs.
- ✗ Communication infrastructure connecting the supervisory system to the remote terminal units.
- ✗ Various process and analytical instrumentation

✔ PVSS II

PVSS II is a SCADA system. SCADA stands for Supervisory Control and Data Acquisition. PVSS will be used to connect to hardware (or software) devices, acquire the data they produce and use it for their supervision, i.e. to monitor their behavior and to initialize, configure and operate them. In order to do this PVSS provides the following main components and tools:

configure and operate them. In order to do this PVSS provides the following main components and tools:

- A run time Database

  Where the data coming from the devices is stored, and can be accessed for processing, visualization, etc. purposes.

- Archiving

  Data in the run-time database can be archived for long term storage, and retrieved later by user interfaces or other processes.

- Alarm generation and Handling

Alarms can be generated by defining conditions applying to new data arriving to PVSS. The alarms are stored in an alarm database and can be selectively displayed by an Alarm display. Alarms can also be filtered, summarized, etc.

- A Graphical Editor

  Allowing users to design and implement their own user interfaces.

- A Scripting Language

Allows users to interact with the data stored in the database, either from a user interface or from a “background” process. PVSS scripts are called **CTRL**(read control) scripts.

- A Graphical Parameterization tool

  Allowing users to:

  •Define the structure of the database

  •Define which data should be archived

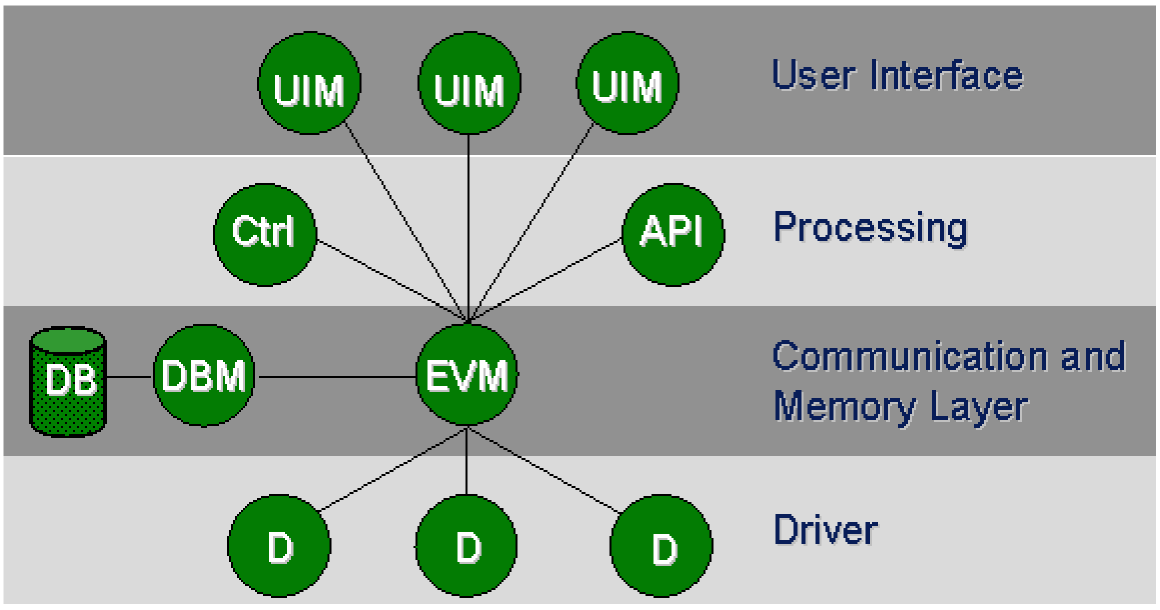
  •Define which data, if any, coming from a device should generate alarms

  •etc.

Architecture

PVSS has a highly distributed architecture. A PVSS application is composed of several processes, in PVSS nomenclature:**Managers**.

A typical PVSS application is composed of the following Managers:



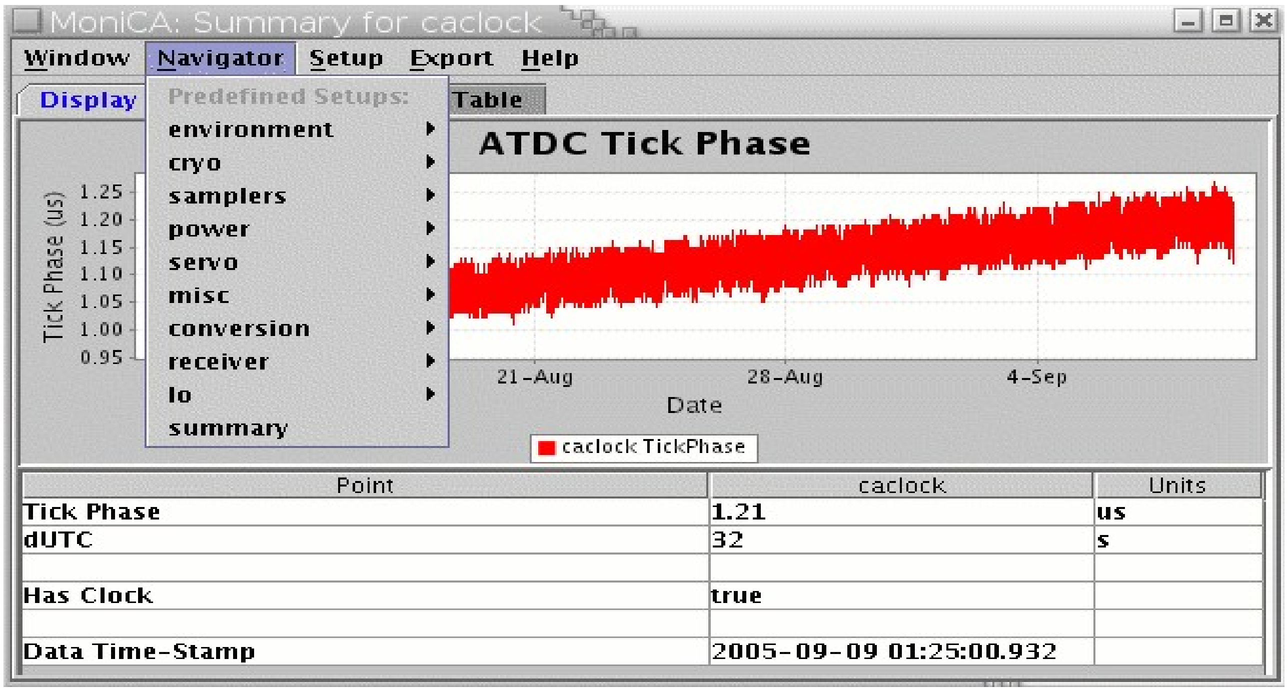
- The Event Manager (EVM) – is responsible for all communications. It receives data from any Drivers (D) and stores it in the data base.
- The Data Base Manager (DBM) – provides the interface to the (run-time) data base.
- User Interface Managers (UIM) – can get device data from the database, or send data to the database to be sent to the devices, they can also request to keep an "open" connection to the database and be informed (for example to update the screen) when new data arrives from a device.
- Ctrl Managers (Ctrl) – provide for any data processing as "background" processes, by running a scripting language. This language is like "C" with extensions.
- API Managers (API) – Allow users to write their own programs in C++ using a PVSS API (Application Programming Interface) to access the data in the database.
- Drivers (D) – Provide the interface to the devices to be controlled. These can be PVSS provided drivers like Profibus, OPC, etc. these will be described later in more detail, or user-made drivers.

A **PVSS System** is an application containing one data base manager and one event manager and any number of drivers, user interfaces, etc.

PVSS Managers can run on Windows or Linux and they can all run in the same machine or be distributed across different machines (including mixed Windows and Linux environments). When the managers run distributed across different machines this is called a **PVSS Scattered System**.

"MoniCA" Monitoring Software ---> ATNF

MoniCA is a Java-based graphical application for viewing real-time and archival monitor data from ATNF facilities. It also gives facility for remote observation using Java Web Start Technology.



Point	caclock	Units
Tick Phase	1.21	us
dUTC	32	s
Has Clock	true	
Data Time-Stamp	2005-09-09 01:25:00.932	

15 M NCRA Telescope

15 M NCRA Telescope has a Control and Monitoring software developed by PSPL technology Ltd.

Silent Features :

- Context based web based interface using Java spring framework as core system services.
- Modular and allow easy hardware/software upgrade.
- Message passing/Service Oriented architecture.
- Scalable and configurable (via XML-DTD specification).
- Automatic state-restoration,exception handling using batch/scripts.
- Alarm notification-Audio, Visual and email.
- End to end software for Radio telescope.

Technologies used :

CMS-WEB : Tomcat, Java, Blaze DS,Flex, XSLT and Java native Access.

Core System services : Spring Framework

State Machine : FSM – Physhun, Jboss Drools

Messaging Layer : Active MQ, **Communication Layer** : TCP/IP Socket based.

Valida tor : Castor, Drools

Database : Hibernate **Build Tool** : Maven (Build entire application)

Graphics plotting package

1. PGPLOT :

The PGPLOT Graphics Subroutine Library is a Fortran- or C-callable, device-independent graphics package for making simple scientific graphs. It is intended for making graphical images of publication quality with minimum effort on the part of the user. For most applications, the program can be device-independent, and the output can be directed to the appropriate device at run time.

The PGPLOT library consists of two major parts: a device-independent part and a set of device-dependent ``device handler'' subroutines for output on various terminals, image displays, dot-matrix printers, laser printers, and pen plotters. Common file formats supported include Post Script and GIF.

PGPLOT itself is written mostly in standard Fortran-77, with a few non-standard, system-dependent subroutines. PGPLOT subroutines can be called directly from a Fortran-77 or Fortran-90 program. A C binding library (cpgplot) and header file (cpgplot.h) are provided that allow PGPLOT to be called from a C or C++ program; the binding library handles conversion between C and Fortran argument-passing conventions.

2. SuperMongo (SM)

SM is an interactive plotting package for drawing graphs. It does have some capability to handle image data, but mostly works with vectors. The main features of the package are that one can generate a nice looking plot with a minimum number of simple commands, that one can view the plot on the screen and then with a very simple set of commands send the same plot to a hard copy device, that one can build and save ones own plot subroutines to be invoked with a single user-defined command, that the program keeps a history of ones plot commands, which can be edited and defined as a plot subroutine, to be reused, and that one can define the data to be plotted from within the program, or read it from a simple file.