PROCEEDINGS OF SCIENCE

EVIDENCE - A control system for laboratory applications and small-scale experiments

Oliver Grimm*
ETH Zürich, Institute for Particle Physics, Schafmattstrasse 20, 8093 Zürich, Switzerland
E-mail: oliver.grimm@phys.ethz.ch

EVIDENCE is a Linux-based control system suitable for small-scale applications. It uses the CERN-developed Distributed Information Management system as communication layer and the freely available Qt and Qwt libraries for visualization.

The 2011 Europhysics Conference on High Energy Physics, EPS-HEP 2011,
July 21-27, 2011
Grenoble, Rhône-Alpes, France

*Speaker.

PoS(EPS-HEP2011)398

1. Introduction

Experiments integrating several hardware and software components require, except for the simplest cases, a control system for operation. Such a system typically comprises several individual programs that require configuration information, produce data, need to exchange information, visualize data of other programs, and monitor system health. Comprehensive control system frameworks exist, for example EPICS or DOOCS,[1] but for a small scale application, these systems are often too complicated and need professional support for their installation and maintenance.

An alternative is the EVIDENCE control system, originally developed within the FACT project (First G-APD Cherenkov Telescope [1]). Care was taken to keep it flexible to become applicable also for other projects. It allows the integration of software with a minimum of extra coding, uses the CERN-developed DIM system[2] as communication layer and the freely available Qt and Qwt libraries[3] for visualization.

EVIDENCE consists of a small C++ class and a few server programs that form the control system backbone. It supports central message logging and the distribution of standard warning and error conditions. Configuration information and data storage are centralized. It imposes only little definite structure on the programmer and is thus easy to integrate into existing programs.

2. EVIDENCE functionality

The basic functionality of EVIDENCE is contained in a small number of server programs. The *configuration server* supplies configuration information to other servers from a single, central text file.[4] Clients are automatically informed of changes to this file. The *data collector* subscribes to all published DIM services (unless excluded through a configuration setting) and logs updates into a data file. An *alarm monitor* surveils the system health by checking if all required servers are up and if any warning or errors messages were published. It generates a master alarm and can send emails to inform of an error condition. To allow quick online access to recent data, without reading the files written by the data collector, the *history server* keeps the most recent updates to DIM services in memory and can dispatch these histories via a DIM call, for example to a graphical user interface.

The data exchange between the programs uses the CERN DIM system: a central name server keeps track of published services, clients can subscribe to services and will exchange information directly with the servers using TCP/IP.

A common C++ class unifies the basic functionality and can be used for application-specific servers. This class automatically starts the DIM server, provides a standardized message service with severity encoding (`INFO`, `WARN`, `ERROR`, `FATAL`), a method for the request of configuration information, and a method for translating DIM service safely into text.

To ease program debugging and fault detection, this class also provides standardized DIM exit and error handlers and installs handlers to react to some standard termination signals. It also catches

[1] `http://www.aps.anl.gov/epics`, `http://tesla.desy.de/doocs`

[2] Distributed Information Management, see `http://dim.web.cern.ch/dim/` and [2] for details.

[3] `http://qt.nokia.com/`, `http://qwt.sourceforge.net/`

[4] The format follows the .INI structure.

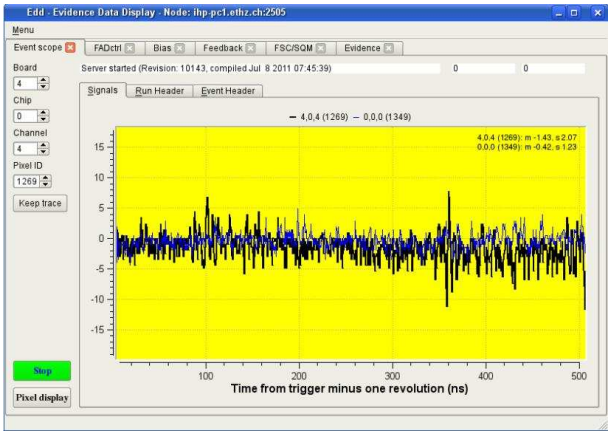
un-handled C++ exceptions and publishes a suitable message with FATAL severity for automatic logging of the problem by the data collector. The class furthermore allows browsing of the data returned by the history server.

The control system has been developed under Linux.[5]

Neither EVIDENCE nor DIM contain specific functions for access control. These can, however, be implemented easily using standard Linux tools like `IPtables`.

3. Graphical user interface

The graphical user interface (GUI) is based on the Qt and Qwt toolkits which are freely available. Widget classes for displaying DIM service information and to send DIM commands are derived from Qt/Qwt standard widgets. A single subscription to a DIM service per GUI instance, even if displayed multiple times, limits the server load. Intuitive drag and drop features, plot navigation and export functions allow to monitor and cross-correlate different DIM services. GUI building follows standard Qt programming procedures.



The GUI displays an alarm window (and may emit an audio alarm) in case the master alarm, generated by the alarm server, increases in level.

The graphical user interface is easily portable. No operating-system specifics are used, only standard C++ code, the Qt/Qwt capabilities and DIM. Qt and Qwt are designed for cross-platform developments, DIM is available for Unix/Linux and Windows.

The figure illustrates the GUI with a digitizer signal display as used in the FACT project. The tabs on the top access the various subsystems. Clicking on a numerical DIM service displays a history plot for this service. For a text service, the recent text messages will be displayed.

Acknowledgments

The help of Clara Gaspar (CERN) was indispensable for understanding the DIM system and making good use of it.

References

[1] H. Anderhub et. al, Nuclear Inst. and Methods A 628, 107 (2011)

[2] C. Gaspar, M. Dönszelmann, Ph. Charpentier, Computer Physics Communications 140, 102-9 (2001)

[5] Tests have been performed on OpenSuse, Scientific Linux and Ubuntu. A detailed description of the EVIDENCE system is available from the author.