EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
Laboratory for Particle Physics

Divisional Report **CERN/LHC 97-05 (VAC)**

Control and Operational Models for Vacuum Equipment

P.M. Strubin and N. Trofimov

Abstract

Operational models which describe the behaviour and the physical values associated with the vacuum equipment as seen by an operator have been studied for some time at CERN. Recently, they have been completed by control models, which define in a formal way the data structures required to access the physical values described in the operational model. The control models also define the operations that an application program has to send to the vacuum equipment to modify its state. Object Modelling Techniques (OMT) have been used to formalise the description of the models.
In order to test the validity of the concepts, we have made a working prototype in the LEP accelerator. This prototype is being built on top of the CERN SL-Equip equipment access package and uses the "cdev" C++ library, developed at TJNAF, for the interface to application programs. SL-Equip is used for data transmission between front-end computers and vacuum equipment. We use the "cdev" networking facilities to communicate between the workstation and the front-end computers, and the "cdev" generic server as the framework for implementing the vacuum controls software. These packages were used in order to minimise the required software investment, but also to prove that these models are hardware and software independent.

Presented at 1997 Particle Accelerator Conference, Vancouver, 12-16 May 1997

Administrative Secretariat Geneva, Switzerland
LHC Division
CERN
1211 Geneva 23
24 July 1997

Control and operational models for vacuum equipment

P.M. Strubin and N. Trofimov, CERN, CH-1211 Geneva 23, Switzerland

Abstract

Operational models which describe the behaviour and the physical values associated with the vacuum equipment as seen by an operator have been studied for some time at CERN. Recently, they have been completed by control models, which define in a formal way the data structures required to access the physical values described in the operational model. The control models also define the operations that an application program has to send to the vacuum equipment to modify its state. Object Modelling Techniques (OMT) have been used to formalise the description of the models.
In order to test the validity of the concepts, we have made a working prototype in the LEP accelerator. This prototype is being built on top of the CERN SL-Equip equipment access package and uses the "cdev" C++ library, developed at TJNAF, for the interface to application programs. SL-Equip is used for data transmission between front-end computers and vacuum equipment. We use the "cdev" networking facilities to communicate between the workstation and the front-end computers, and the "cdev" generic server as the framework for implementing the vacuum controls software. These packages were used in order to minimise the required software investment, but also to prove that these models are hardware and software independent.

1 INTRODUCTION

Recent experience, when developing a common man machine interface for vacuum controls for all accelerators at CERN [1], made evident that a significant amount of work was induced by the variety of vacuum equipment and access methods. Whereas the interface presented to the operator is identical for all accelerators, the underlying programs are quite different. The work presented here is aiming at hiding the differences between equipment to the application programmer as well as to provide him with a clear description of how to control the vacuum equipment. Starting from the operational model [2], a control model is defined, which makes use of building blocks which we call components. These components provide a uniform way of representing physical data from which the application program interface can be derived.
The application interface is built from a device server using the CDEV interface library. This server is the key to present the control model in a coherent way to the application programs.

2 MODELS

There are two concepts of a model which have to be clearly defined, the operational model and the control model.

2.1 The operational model

The operational model refers to the following:
- a narrative description of what the primary function and behaviour of the device is;
- the enumeration of the various physical values of interest which can be observed by means of the hardware and firmware installed in the control unit of the vacuum device. There is no assumption about any data type, we simply refer to variables and units applicable to the physical values;
- the state that a given vacuum device can take in normal operation, documented with the reasons which can lead to a change of state (actuation, time-out, interlock, fault, etc.);
- the commands (actuation) the vacuum device can accept to modify its state and possible restrictions to these commands;
- the settings of the various physical values of interest and the calculation parameters that the device needs to operate according to its specifications.

The operational model does not imply computer control and is therefore applicable to manual or computer controlled operation. It does not depend on a particular make of the device, although some real implementation may not fulfil all aspects of the operational model. Finally, there should be no controls related definitions or concepts in its description.

2.2 The control model

The control model refers to the following:
- define in a formal way the data structures required for the application programs to access the various physical values described in the operational models, including related information like time-stamp or validity;
- define in a formal way the data structure required for the application programs to recognise and handle abnormal operating conditions or faults;
- describe in an semi-formal, although exhaustive, way the transitions between the possible states defined in the operational models of the vacuum devices as a result of external or internal events, typically using some form of state transition diagrams;
- define in a formal way the procedures (operations) required by the application programs to send the events (actuation) required for operating the vacuum devices;

- define in a formal way the aggregation of the above mentioned data structures to emulate the modelled vacuum devices. This aggregation is referred to as configuration.

All formal definitions must be independent from any programming language, but should be easily translated into a real language.

3 REQUIREMENTS OF APPLICATION PROGRAMS

All application programs must be able to control vacuum devices in an identical way, using the same configuration.
Main functionality related to vacuum equipment which should be accessible via the model is to evaluate the current state of a device, to be able to modify it and to acquire all relevant physical values.
Auxiliary information for the evaluation of the current state, like the time when the state was last observed or changed and the reason for a change must be provided to the application programs. If a request to change state fails, the application programs must be able to work out the reason of the failure.
Similarly, the application programs must be able to obtain, together with the relevant values from the physical process, the time a value was last evaluated, the limits of validity, the resolution and the units in which a physical value is represented.

4 CONTROL MODELS OF VACUUM DEVICES

A vacuum device will in general be a piece of vacuum equipment, like a gauge or a pump, associated to its corresponding power supply or controller and connected to the control system of an accelerator through an appropriated network.

4.1 Functional components

Functional components define a common way to represent the functions and data available in a vacuum device. They can be looked at as building blocks through which access to the information of specific devices can be granted. We do not describe real data access channels here.
For the purpose of vacuum equipment modelling, we define seven basic types of the functional components. Each component type represents certain device capability and holds a number of data items, or properties, that applications can observe and, in some case, modify.
For example, Continuous Input (CInput) models the analogue measurement capability. It provides read only access to the measured value of the associated physical value, as well as some related properties, such as the valid range, the units of measurement, achievable resolution, etc. A pressure returned from a gauge can typically be represented by a Continuous Input.
It is possible to describe these functional components using class hierarchy and association:

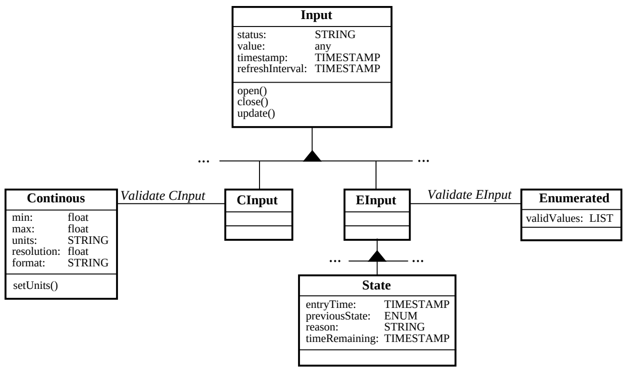

Figure: 1 Example of components used to feed data into a model

Components are defined as a result of the analysis of the requirements of application programs. They may be extended to include more information, but care must be taken not to try to adapt the functionality of the vacuum devices to the any arbitrary definition of components.

4.2 Control models

A control model is based on a set of functional components and a state model. Hence, the control models for vacuum devices are built by aggregating functional components around a generic vacuum device class. The behaviour of a controlled vacuum device is described by its state model. The generic vacuum device class is the way to provide the interface to this state model for an application.
A configuration database describes which physical values can be read, which need settings and what type of components are used to access the physical values. This is not to be confused with the database describing the topology (or grouping) of equipment.

5 INTERFACE TO APPLICATION PROGRAMS

The application programs only know a device via its control model. The exact level at which the model becomes first visible in a control system depends on the system architecture and specific design choices made in the equipment interface. It should be at the lowest possible level in order to minimise unnecessary data transfer and reformatting.
A software process (or a number of processes) maintains an image of the state and of the data generated by a vacuum device for the application programs and forwards data from the application programs to the vacuum device, using the data structures described in the control model.
The representation of the various data structures for an application within a specific implementation is derived from the control model.

6 CLASS HIERARCHY FOR VACUUM DEVICES

Analysis of the control models reveals the common functionality and capabilities between vacuum devices. This leads to the definition of a class hierarchy in which the common features are placed at the top. Specialisation of these generic classes provides the interface to more specific vacuum equipment, like gauges. Common functionality, like pressure measurement, can be added as another abstract class, from which "gauges" would inherit. Finally, further specialisation will lead to the classes representing the real equipment.

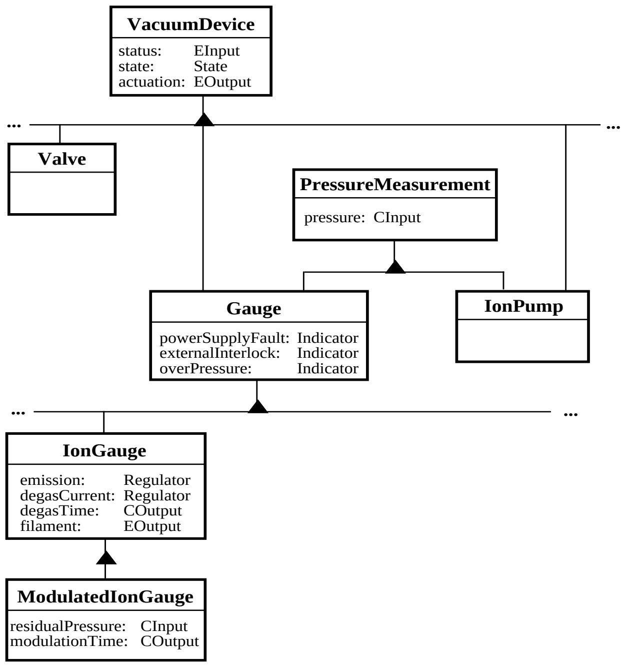

Figure: 2 Part of the Device Class Hierarchy

7 IMPLEMENTATION OF A PROTOTYPE IN LEP

The prototype implementation in the LEP control system employs the client-server architecture (Fig. 3). The VacModel servers are running on the front end computers (LynxOS on VME/PowerPC) and provide a model based view of the vacuum equipment for the applications running in the workstations (HP-UX). The VacModel server uses the standard LEP equipment access software (SL-Equip) [4] to communicate with the equipment, and the existing Oracle database as primary source of all the configuration data.
The server implementation is based on the "CDEV Generic Server Engine", a collection of C++ classes developed at TJNAF [3]. The CDEV (Control DEVice) class library aims at providing an object-oriented interface between an application and underlying accelerator control software. Basic concepts of the CDEV application programming interface are very close to our control model definition, so the mapping is rather simple and straightforward.

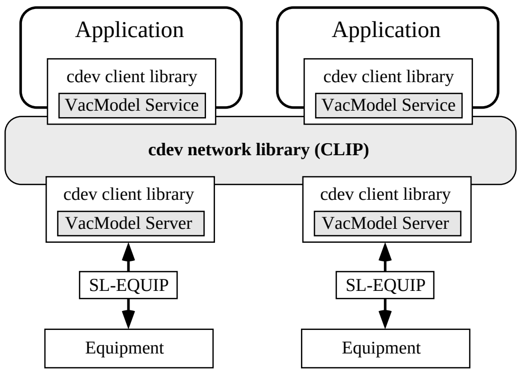

Figure: 3 Overview of implementation

The device class hierarchy (Fig 2) can easily be expressed in the CDEV Device Definition Language and the components of the control model map directly to the CDEV device attributes. Using standard CDEV messages, an application can read, write or monitor values of the components' properties. All required communication functionality is provided by the CDEV library classes. The specific VacModel software essentially binds the CDEV framework to the LEP database and equipment access environment.

8 CONCLUSIONS

Operational models have already been successfully applied in several projects, but the large variety of access methods meant that a significant amount of work was required whenever a new device was added or an application was ported to a different environment. Control models aim at minimising this effort and the first prototype implementation in LEP shows that this is quite possible.
This prototype was built using existing "general purpose" libraries, like CDEV and SL-Equip, which demonstrates that a generic description of equipment leads itself to an easy implementation on real hardware and software platforms.

9 REFERENCES

[1] 'A data driven graphical user interface for vacuum applications', L. Kopylov, M. Mikheev, N. Trofimov (IHEP Protvino), P. Strubin, M. Steffensen (CERN), ICALEPCS 1995, Chicago, USA.
[2] 'Using models to allow for object oriented programming of the vacuum control systems', R. Gavaggio, M. Steffensen and P. M. Strubin, EPAC 1994, London, UK.
[3] 'CDEV: An Object-Oriebnted Class Library for Developing Device Control Applications', J. Chen, G. Heyes, W. Akers, D. Wu and W.A. Watson (CEBAF), ICALEPS 1995, Chicago
[4] 'The Equipment Access Software for a Distributed UNIX-Based Accelerator Control System', N. Trofimov, S. Zelepoukine, E.Zharkov (IHEP Protvino), P. Charrue, C. Gareyte, H. Poirier (CERN), ICALEPS 1993, Berlin