Event Handling in TRIUMF's Central Control System

B.Davison, S.G. Kadantsev, E. Klassen, K.S. Lee, M.M. Mouat, J.E. Richards, T.M Tateyama, P.W. Wilmshurst, P.J. Yogendran
Tri-University Meson Facility (TRIUMF)
4004 Wesbrook Mall, Vancouver BC V6T 2A3, Canada

Abstract

In TRIUMF's Central Control System (CCS), alarm handling and other types of events are dealt with in a software "scan" package. Many changes that must be monitored are not considered as alarms because there is no action associated with the various values of the control variable and thus the software package was named to reflect the action of scanning the system for defined state changes. This scan package can issue messages to different logs and take actions as determined by the user. The initial requirements, design, implementation, and user interface are described.

1 Introduction

During the upgrade of TRIUMF's central control system, it became apparent that there is a significant amount of process monitoring and alarm handling that could be controlled and operated on by a single software package. Many of the items that are monitored just report state changes or significant magnitude changes to the operators. Some items can cause beam interlocks and others change channel settings or status. Prior to the upgrade from the Nova computers, most of the scan tasks were controlled by two programs called "console scan" and "mchk". There was also a "sparking scan" program and several smaller tasks to monitor various things. It was decided to amalgamate all of these "scanning" tasks into one VMS software package which was named the "scan Utility". This package scans or monitors predetermined channels and takes one or more of many possible actions from producing a message to tripping off the beam.

2 Requirements

A significant requirement of the new system was the concept of a scan "element" which could consist of one or more channels. This provides the ability to compare one channel to another or to logically combine the comparisons of many different channels with different test conditions. This was a major philosophical change in that the operators were used to disabling a scan of a single channel whereas now they would disable a scan element which may be a combination of a number of channels. Another requirement was an interface that provided an easy, coherent way for operators to defeat any scanned element and to monitor at any time which elements (and which channels these affect) are currently off-scan. A method to specify the contents of elements, the frequency of scanning as well as the actions to take when an element condition is met was also required. As the central control system is somewhat dynamic, with new channels being added and others being changed or removed, it was important to specify the scans in a way that did not require code changes when elements were modified. Interprocess communication was needed so that other programs in the control system could communicate with the scans to temporarily disable some elements. For example, when operators are adjusting channels from a console, they do not want to see messages to this effect.

3 System design

The domain of all scan elements was divided into individual scans comprised of typically related channels monitored at the same frequency. In order to easily change the channels being monitored, it was decided to specify each scan in a separate ASCII file using a scan 'meta' language. For example, the following is a description of a scan:

```
FREQUENCY: 2.0
NUMBER OF ELEMENTS: 3
//
// TNF  warn bits
//
ELEMENT 1: ENABLED,ACTIONS: 1
TG,50,REG,RAW != PREVIOUS
DO MESSAGE 7750
//
// TNF temp check
//
ELEMENT 2: ENABLED,ACTIONS: 1
TG,54,MUX,SCALED > 200.0
DO MESSAGE 7850
//
// Machine protect on tank spill
//
ELEMENT 3: ENABLED,ACTIONS: 2
RF,360,MUX,RAW > B1,332,MUX,RAW
DO INSERT FARADAY CUP BIT=31
DO LOG opslog+mainlog "MACHINE TRIP; 2C SPILL"
```

The number of elements listed in the file and the frequency (in executions per second) is listed at the top. This is followed by each element description consisting of identifier (a number, eg. ELEMENT 3), current status (eg. ENABLED), number of functions that may be performed (eg. ACTIONS: 2), channel conditions (eg. RF,360,MUX-,RAW > B1,332,MUX,RAW) and the explicit action list (eg. DO INSERT FARADAY CUP BIT = 31 and DO LOG opslog+mainlog "MACHINE TRIP, 2C SPILL). In the TRIUMF central control system, channels are specified

by system and thumbwheel. For example, RF,360,MUX indicates the beam line 2 tank spill wall thermocouple. These files are read and compiled by the scan utility once when a scan is started. This permits both easy changes to scan lists coupled with fast operation from the compiled version stored in memory when a scan is running. Each scan is a separate process in the production environment. One scan can be stopped, modified and restarted without affecting any other scan in the system. However, for system management reasons as well as to facilitate interprocess communication, it was decided that each scan executable would run as a subprocess of a " scan control" program. The user interface and the programs that cause automatic disables (when operators are changing things via the console) communicates with a single process (scan control) which can then communicate to the individual scans as needed. Information about everything that is disabled is stored in a single place. Figure 1 illustrates the various components of the scan utility.

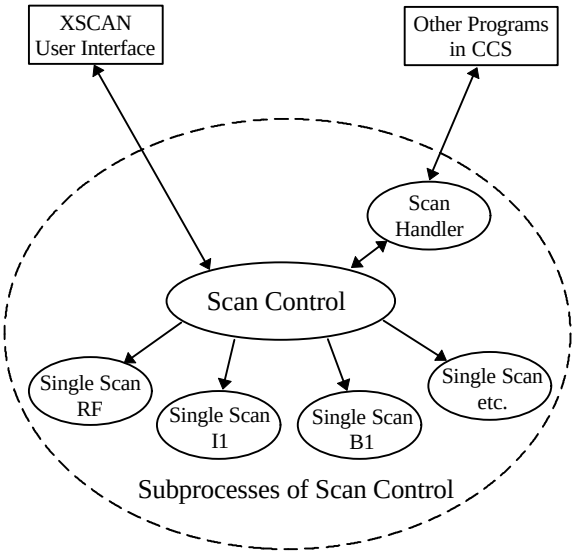

Figure 1

The scan handler process shown in figure 1 is used to translate requests that come from other CCS programs into the format that the scan control program understands. XSCAN is an X windows application that is the primary user interface.

4 Implementation

Our current computing environment consists of two clusters, each of four nodes. Both clusters are mixed containing one Alpha/VMS machine and three VAX/VMS machines. The two clusters allow us to use one cluster as a production cluster to run the cyclotron and the other as a development cluster to test new software.

4.1 DEC messageQ

One of the first implementation decisions was to choose DECmessageQ (DMQ) as the mechanism for interprocess communication. DMQ is also used in the control system to transfer and organize all of the messages produced by the control system. DMQ is message oriented middleware manufactured by DEC and allows programs on any of our 8 nodes (arranged in two clusters) to seamlessly communicate. DMQ can be set up with several independent message queueing buses. Currently we have two buses which allows us to run DMQ linked applications in a test mode on the development cluster without the message traffic from the development bus interfering with the message traffic on the production bus.

4..2 C++

In order to use memory efficiently and save on image activation times, much of our commonly used code, such as device access, is in installed shareable images. We continued this with the DMQ calls and code. The scan utility was implemented using C++ but C++ without all of its objected oriented features. We found that using VMS shareable images precluded using all of the features of C++.

4.3 X/Motif

The user interface was implemented using X Windows-/Motif, which is the standard for the TRIUMF CCS. It provides a reliable, common look as well as enough flexibility to provide the operator with a visible indication of what is defeated and provide a way to enable and disable scan elements. The user interface is described in section 5 below.

5 User interface

The user interface is an X/Motif application which runs detached. The main interface window lists the scans that are active in a scrolling window on the left side and the scans that are inactive in a scrolling window on the right hand side. An operator can activate a whole scan by selecting a scan from the right hand window and then pressing an activate button. There are a variety of buttons below the scrolling windows that allow operators to view the status of each scan and determine which elements are active in a selected scan. There is a command entry window at the bottom that allows operators to enter commands for enabling and disabling various scan elements. There is also one special display activated by the Faraday Cup Push Button which provides a display for all of the elements (spread between many scans) which cause a machine trip (by inserting the faraday cup). There is no limit on the number of user interfaces that can run at a given time. When a command is entered or a request for off-scan information is received, the user interface sends a message via DMQ to scan control (which may be running on a different computer) and it performs the required action or replies with the requested information.

6 Results and conclusions

The scan utility was commissioned in December of 1996 and has been running since. All of the individual scans (single scans) along with the scan control and scan handler processes run on one computer. The user interface runs on any CCS computer desired by the user. The computer running the scans uses approximately 40% of its cpu time to scan 3161 elements. Many of the scans run once every 6 seconds, several of the scans (ones that cause interlocks) run 5 times per second and then some individual scans run at a variety of different frequencies.

One of the strengths of the system is the speed with which new scans can be described and implemented. Sometimes, a particular experiment or operation will require something to be monitored at a specified frequency for a short time period of a day or a week. We are able to quickly describe the scan using the scan meta language, send a message to the scan control process to spawn another subprocess to run this scan without changing or interfering with any of the other scans. One of the limitations of this system is the speed with which DMQ can deliver messages from one program to another. In a test at TRIUMF, it was found that the DMQ can handle about 1000 messages per second. However this includes all of the message traffic on the system. We have found that under normal operating conditions, we have no problems with this limitation. However, during an unusual event, typically a power outage, we find that nearly every scanned item as well as many other programs all report problems to the operators. This causes a large flurry of message traffic and depending on how fast messages pending for programs are emptied from the DMQ queues, it is possible for some of the messages to the operators to be lost.

The scan utility has been running for nearly a year with very little down time and generally good acceptance by the operators. It exceeds the older Nova version of the software both in the complexity of the scans that can be described and in the ease with which scans can be added and changed.

References

[1] Pam Browns documentation on the Nova version of this software (mchk and console scan)
[2] DEC DECmessageQ product documentation