FERMILAB-CONF-00-400-CD

Online Monitoring[1] and Module Maintenance[2] for CDF in the Upcoming Fermilab Tevatron Run II

B. Angelos , T. Arisawa, F. Hartmann, N. Ho, K. Ikado, S. Jones, K. Maeshima, R. Pordes, H. Stadie, G. Veramendi, S. White, J. Yoh

presented by

Hans Wenzel

1) http://www-b0.fnal.gov:8000/consumer/consumer.html

2) http://miscomp.fnal.gov/cdfdb/

Operated by Fermi Research Alliance, LLC under Contract No. De-AC02-07CH11359 with the United States Department of Energy.

1. Online Monitoring

Definition of a Consumer-Monitor

Consumers use event data to monitor the experiment in real-time. Consumer-Monitors are AC++ modules which is the CDF offline data-analysis-framework.

Things typically monitored by Consumer-Monitors are: detector occupancies (dead/hot channels), trigger rates and logic, luminosity, Level 3 reconstruction, physics objects, vertex positions, etc...

Specifics of each of the Consumer-Monitors are determined and written in collaboration with the experts of each subsystem.

Desired features of the Consumer-Monitoring Framework

Think about what you want to achieve, what you did or did not like about the previous system. Do you want to improve an existing system or develop something new.

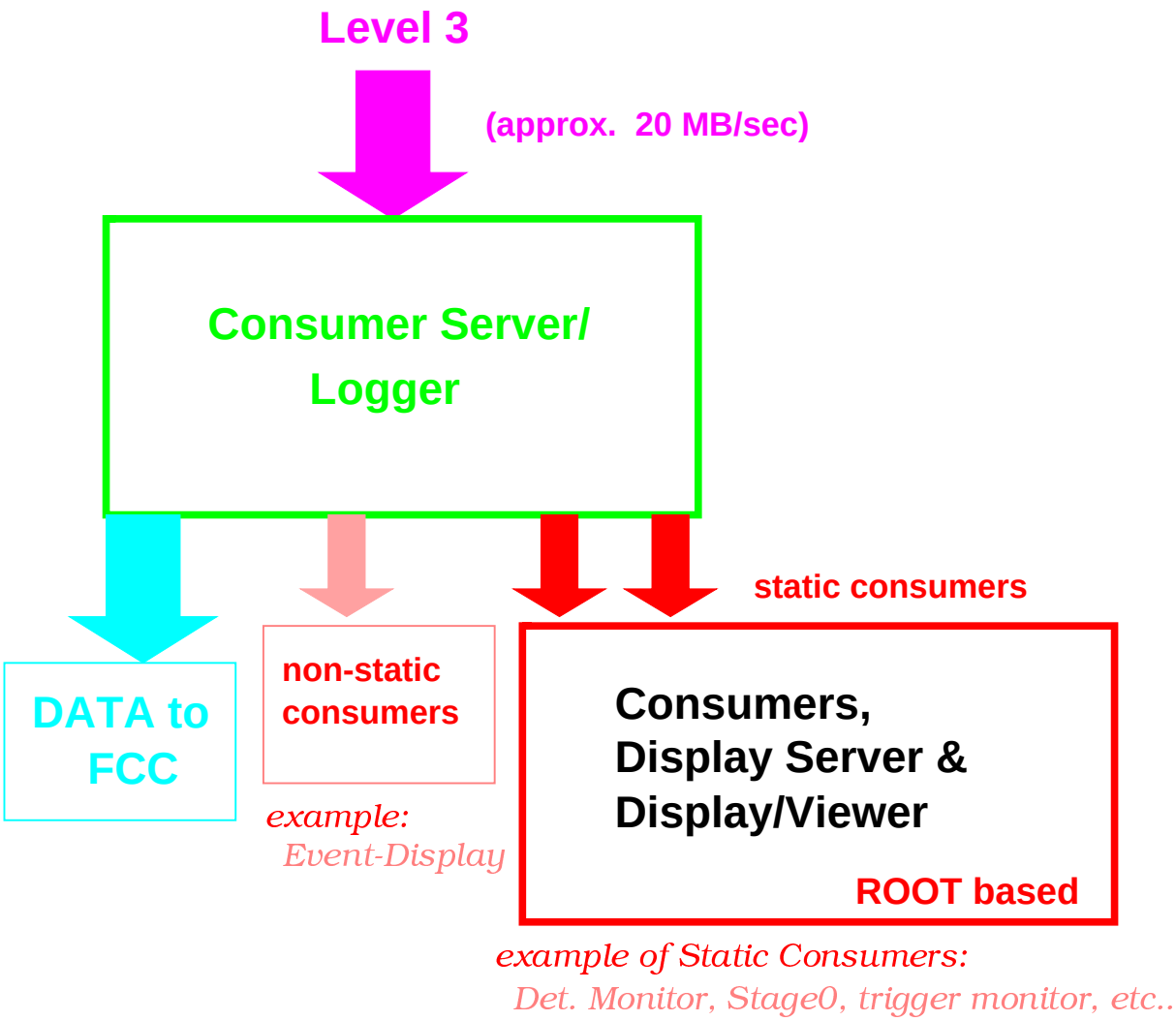
- monitor the detector without interfering with the data taking
- different consumer processes can run on different machines
- each consumer receives only the data it needs
- the monitoring and the display processes are separated

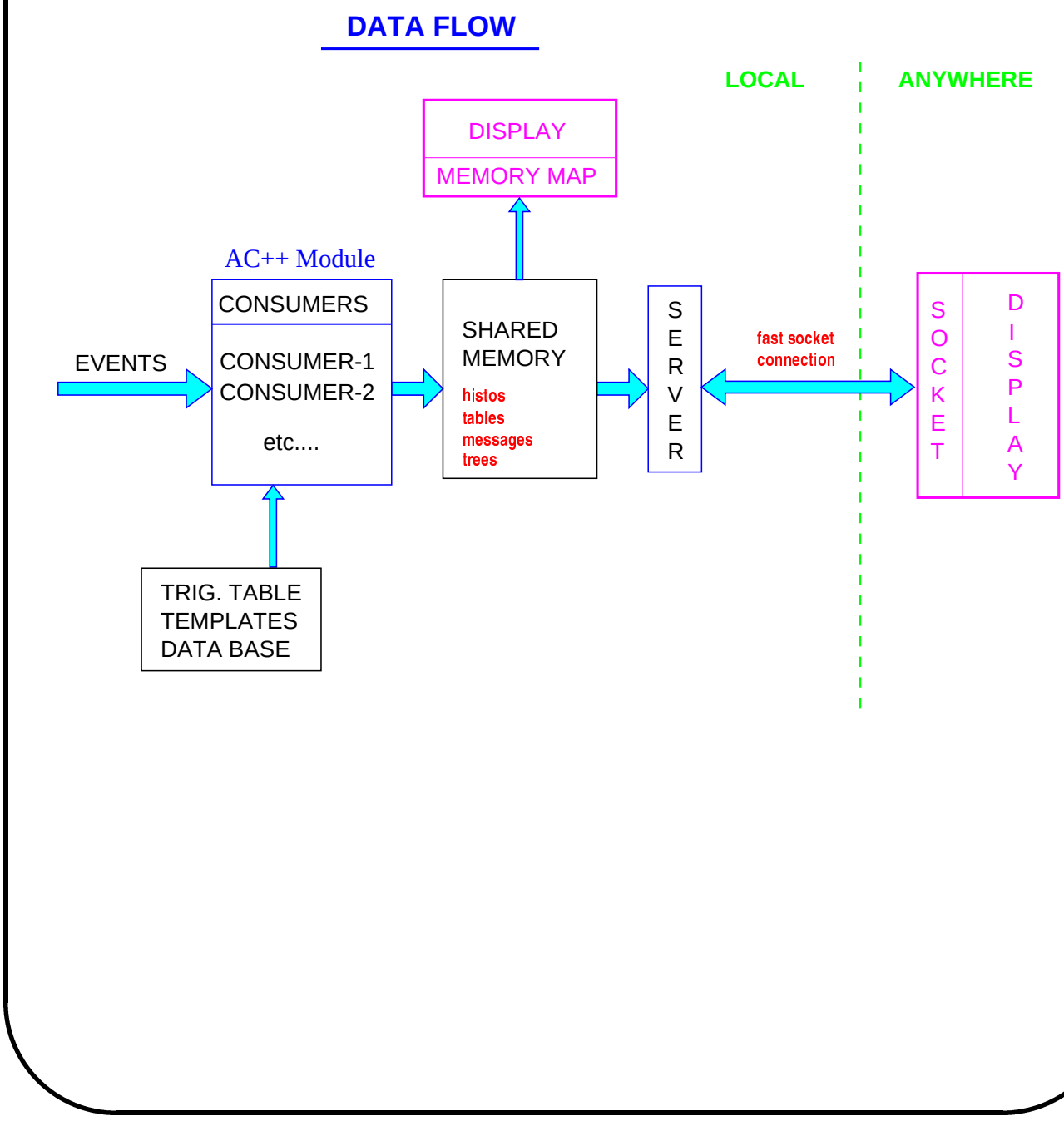
  The number of displays is only limited by network traffic and bandwidth
- different consumers can be combined

Desired features of the Consumer-Monitoring Framework (cont.)

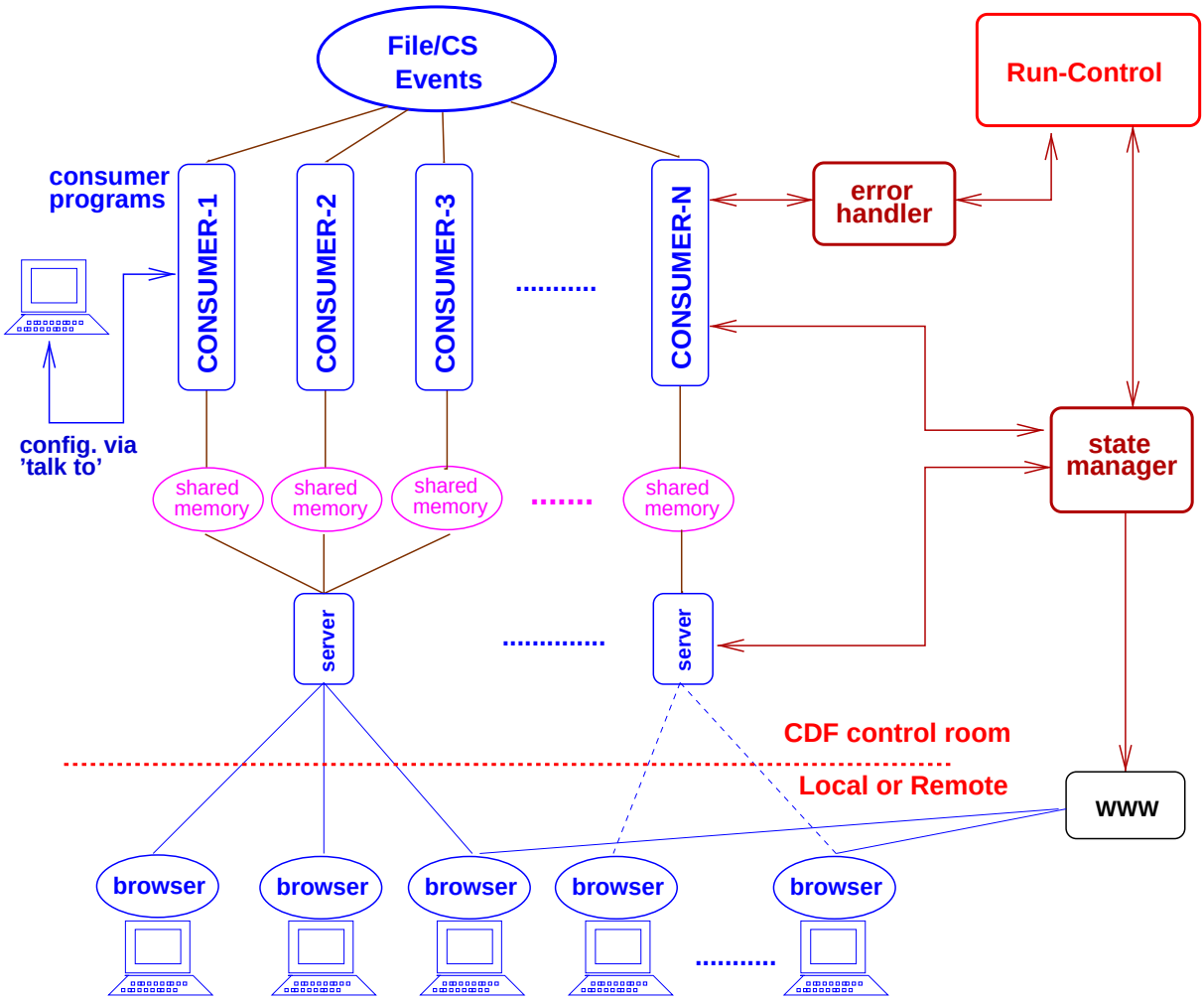
- monitoring programs are written by the experts. We just provide a framework and coordination (needs backup by management).
- common interface and maintainability
- convenient access to archived data $\rightarrow$ you can check at time of analysis. (Previous system: print out in the control room or file "somewhere")
- common Display program and error handler allows to view the results of different Consumers compare histograms and get to the bottom of a problem (hopefully). (Previous system: every monitor used different software)

Consumer Server, Consumers and Display Server




DATA FLOW
LOCAL
ANYWHERE
DISPLAY
MEMORY MAP
AC++ Module
CONSUMERS
EVENTS
CONSUMER-1
CONSUMER-2
etc....
SHARED
MEMORY
histos
tables
messages
trees
S
E
R
V
E
R
fast socket
connection
SOCKET
DISPLAY
TRIG. TABLE
TEMPLATES
DATA BASE

Components of the Consumer-Monitoring Framework



- Consumers (monitor modules)
- Server
- Display
- Error Handler
- State Manager

I. The Consumers

- analyse and monitor the event data and check the status of the experiment
- store the results in shared memory (TMapFile)
- can consist of different monitors
- are written by the experts
- we try to make the start as easy and painless as possible!

Our help for the monitor writers:

- a script that automatically generates all necessary files:
  - Makefile
  - monitor header and source
  - module to integrate the monitor(s) into the CDF offline framework
  - build job to link the required modules (e.g. the interface which allows to connect to the DAQ/Consumer Server) together
  - tcl script to run a test job
- well documented example monitors
- WWW pages with information about the project and its programs: http://kcdf1.fnal.gov/ wenzel/consumer_new

Physical Design of the Consumer Package

- Consumer
  - Consumer
    - BaseMonitor.hh
    - CQIEMonitor.hh
    - DisplayServer.hh
    - FKbnkInputModule.hh
    - HistoDisplay.hh
    - PQIEMonitor.hh
    - YMonModule.hh
    - link_Consumer.mk
  - dict
    - DisplayServer_linkdef.hh
    - GNUmakefile
    - HistoDisplay_linkdef.hh
    - TRates_Elem_linkdef.hh
  - src
    - BaseMonitor.cc
    - CQIEMonitor.cc
    - DisplayServer.cc
    - FKbnkInputModule.cc
    - GNUmakefile
    - HistoDisplay.cc
    - PQIEMonitor.cc
    - TRates_Elem.cc
    - YMonModule.cc
  - Template
    - GNUmakefile
    - Template.cc
    - Template.tcl
    - TemplateModule.cc
    - TemplateModule.hh
    - XXXXMonitor.cc
    - XXXXMonitor.hh
  - YMon
    - GNUmakefile
    - YMon.cc
    - YMon.tcl
  - Executables
    - DisplayServerMain.cc
    - GNUmakefile
    - HistoDisplayMain.cc
  - Test
    - GNUmakefile
    - Producer.cc
  - cr_mon.csh

How to create a consumer:

source cr_mon.csh MUMon CMUO

Result:

- MUMon
  - GNUmakefile
  - MUMon.cc
  - MUMon.tcl
  - MUMonModule.cc
  - MUMonModule.hh
  - CMUOMonitor.cc
  - CMUOMonitor.hh

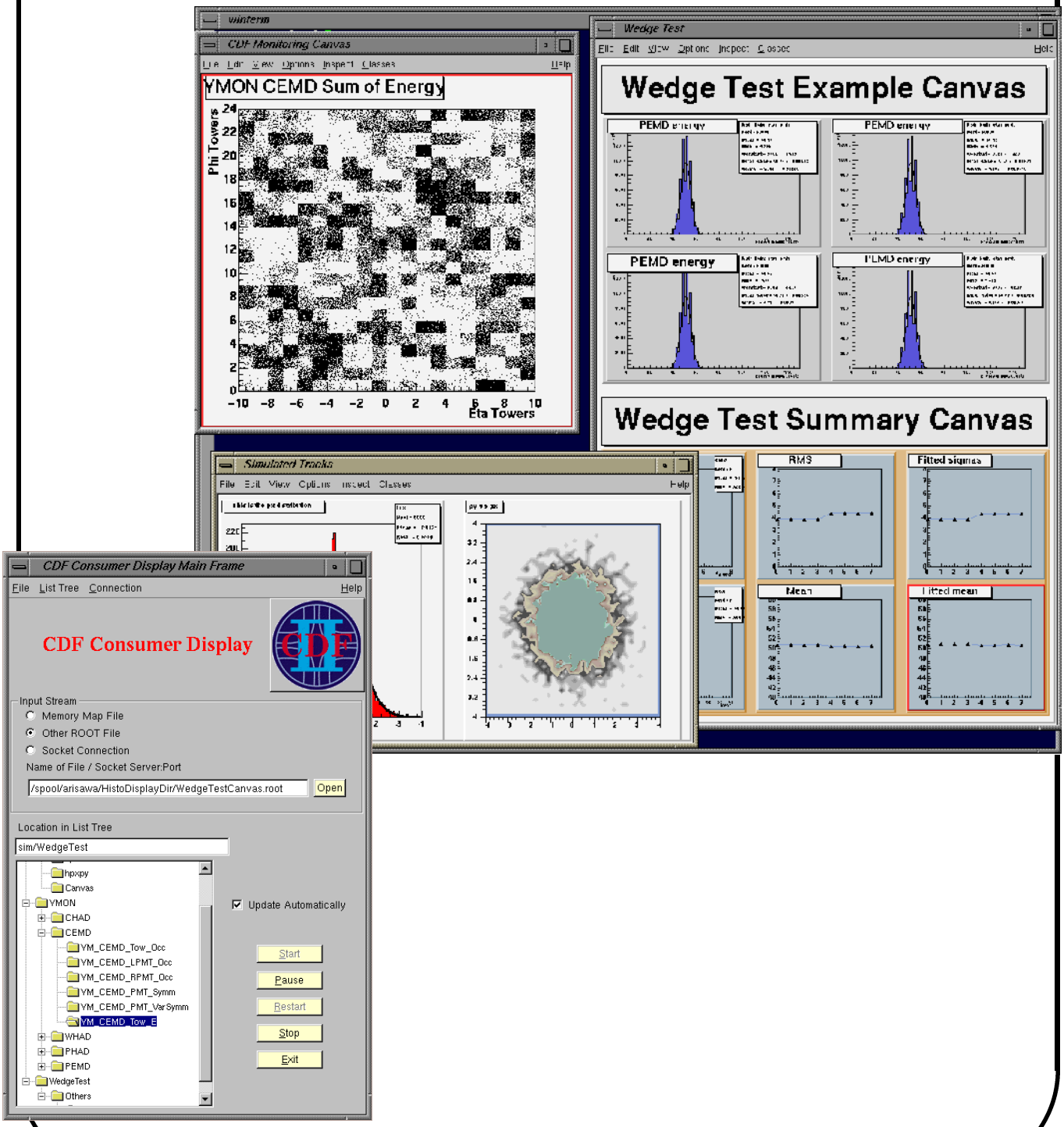
II. The Server

- takes the ROOT objects out of shared memory and sends them to the clients

  TMapFile, TSocket

- displays the status of each monitor job on a WWW page

III. The Display



IV. The Error Handler

We are evaluating the ZOOM ErrorLogger:

(part of the Run II framework)

(http://www.fnal.gov/docs/working-groups/fpcltf/fpcltf.html#ErrorLogger)

V. The State Manager

- shows and controls the status of each consumer and server
- reports errors, communicates with Run_Control
- parts are implemented in the server and startup scripts

Current Status

- all components of the framework have been tested
- we successfully received data from the Consumer Server
- we are currently developing the full scale software programs to be used in Run II
- various Consumer-Monitor programs are being developed using the provided templates and framework
- we intend to use the programs during the ongoing commissioning of the CDF II detector.
- documentation and tutorials

Problems and wishlist

ROOT is a great tool. Would be nice to have good Error handling, graphical GUI builder....

Much better tools (e.g. debugger) are really needed for efficient C++.

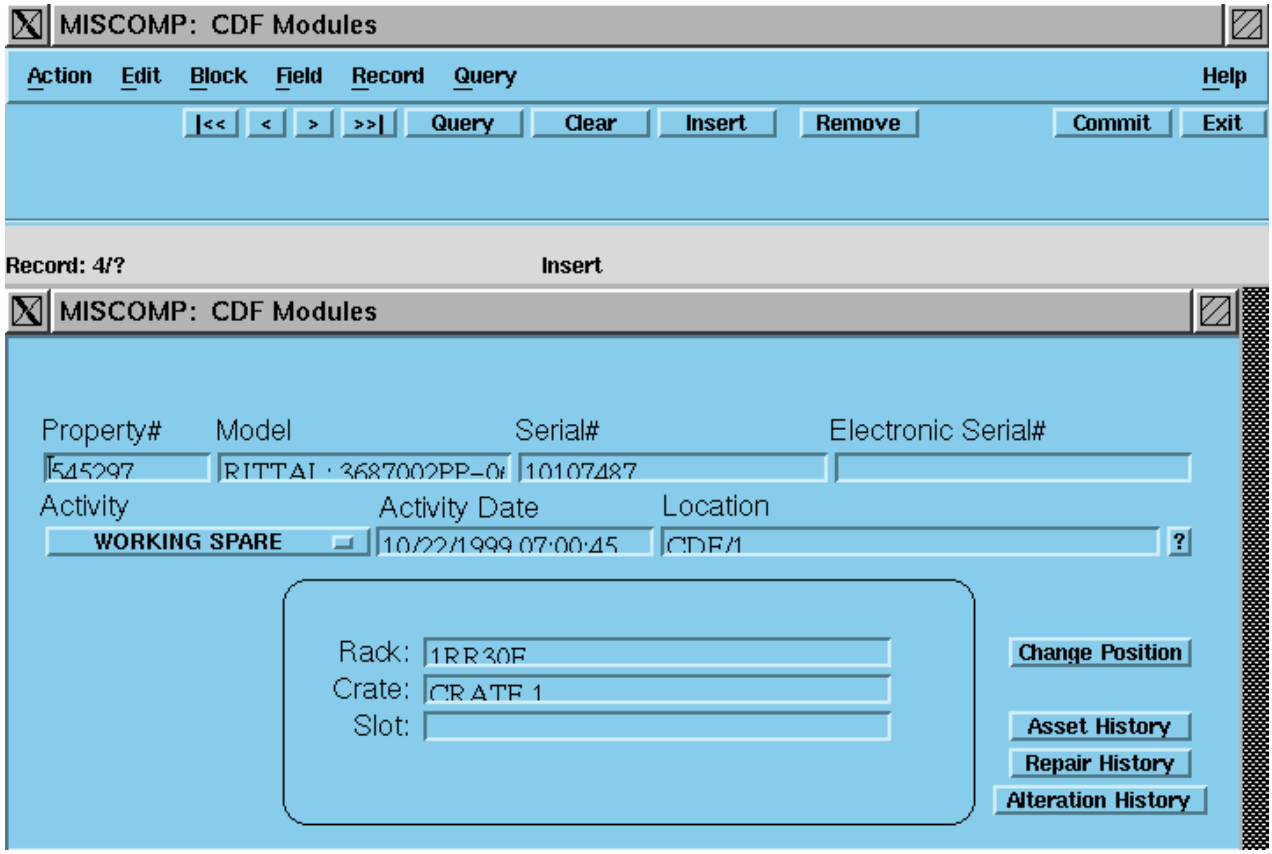
We encountered some problems using TMapFile: Some of them were fixed by the ROOT team but some remain.

- how to get the size and limit the size of the memory region?
- the memory address to map to is not communicated or checked $\rightarrow$ segmentation faults and system crashes.
- no directory structure and inefficient use of memory.
- no selective update of objects in memory.

2. Module Maintenance

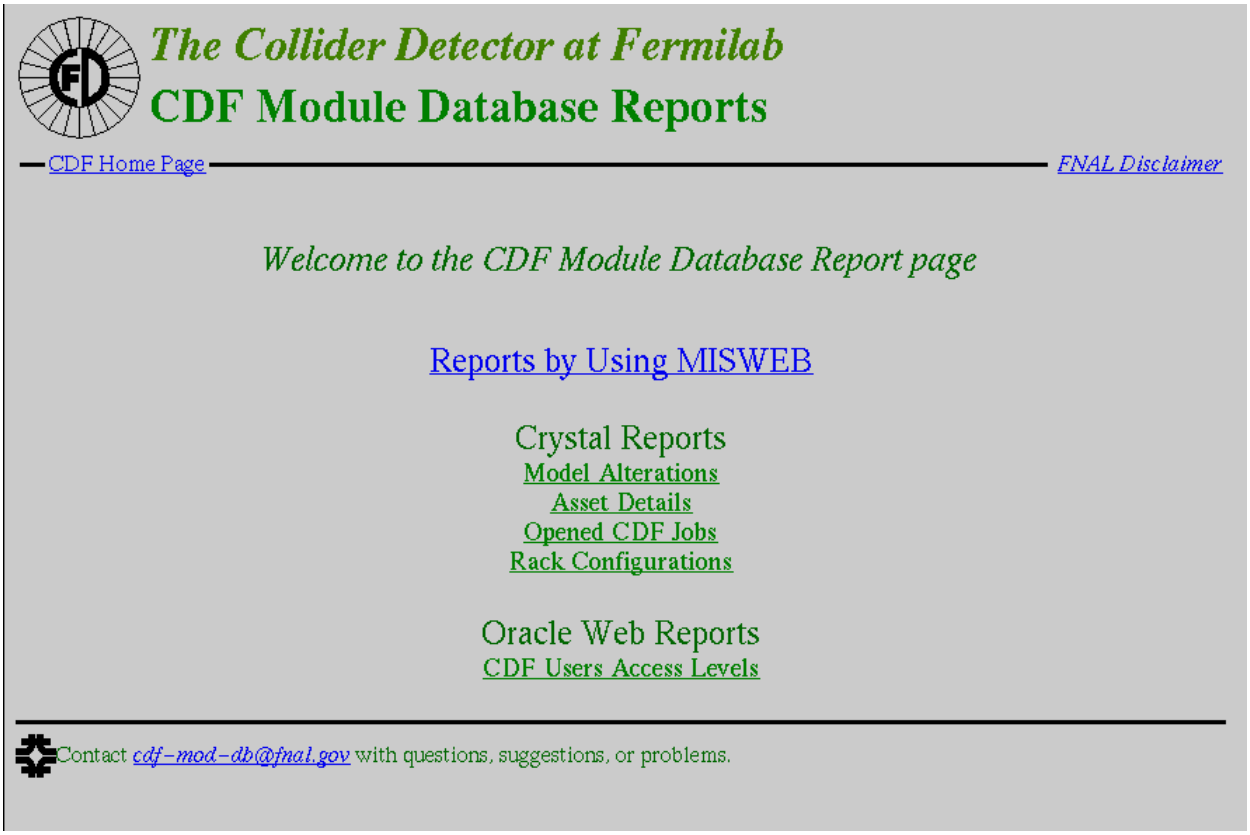
The CDF Module database is based on ORACLE. It is an extension of the main Computing Division equipment database, MISCOMP-EQUIPDB. It keeps track of:

- exact location, specific location within the CDF readout system.
- serial number
- repair history
- prom versions
- engineering change orders


MISCOMP: CDF Modules
Action Edit Block Field Record Query Help
|<< < > >>| Query Clear Insert Remove Commit Exit
Record: 4/?
Insert
MISCOMP: CDF Modules
Property#
Model
Serial#
Electronic Serial#
Activity
Activity Date
Location
WORKING SPARE
Rack:
Crate:
Slot:
Change Position
Asset History
Repair History
Alteration History

Various querying tools API's are available:

- Oracle reports
- cron jobs
- WWW: MISWEP a flexible PERL CGI script
- commercial tools: Crystal Reports
- CDF silicon project: custom java interface



Priorities

We anticipate that once the run has started, applications will make use of the various oracle API's to integrate information about modules into experiment applications and reports. (Getting at the source of a problem, optimising maintenance...)

The success of such a database depends on its acceptance by the people "in the field". The efforts now are concentrating on tuning the application to deliver the most intuitive and easy to use update and reporting interfaces.

Acknowledgements

We thank all the people who are contributing to this work. Special thanks to the ROOT team, the CDF online group, and the Fermilab Computing Division. We also thank the community of ROOT users who via the roottalk mailing list provide answers and solutions to many problems. It's an invaluable help to find the code of other successful application on the web. The on-line monitoring e.g. benefited fom the LHCb -test-beam software.