COMICS - an Alternative to Save/Restore?

J. Frederick Bartlett

DØ Experiment

Fermilab

Origins

- What is the meaning of the COMICS acronym?
  - I inherited it and I don't know
  - The person who previously was responsible for configuration of the DØ detector does not remember either

Justification

- If one builds yet another Save/Restore utility, some justification is required
- Why not use an existing save/restore utility?
  - The collection of all possible configuration states of a detector can be combinatorially very large
  - Efficient verification of the configuration state is essential
  - The basic configuration element is a higher-level device and not an individual process variable (PV)
    - 1 H/L device —> many PV's (records)

Justification (more)

- Why not use an existing save/restore utility?
  - Multiple processes may trigger detector reconfiguration
  - Experts require detailed control of sub-system configuration

Characteristics of COMICS

- Structure of the managed system is described as an acyclic graph (tree)
- Organized as a server with multiple clients
- Written in the Python scripting language (adherents of the Java sect may use Jpython for religious purity)
- Designed to be extensible through class inheritance

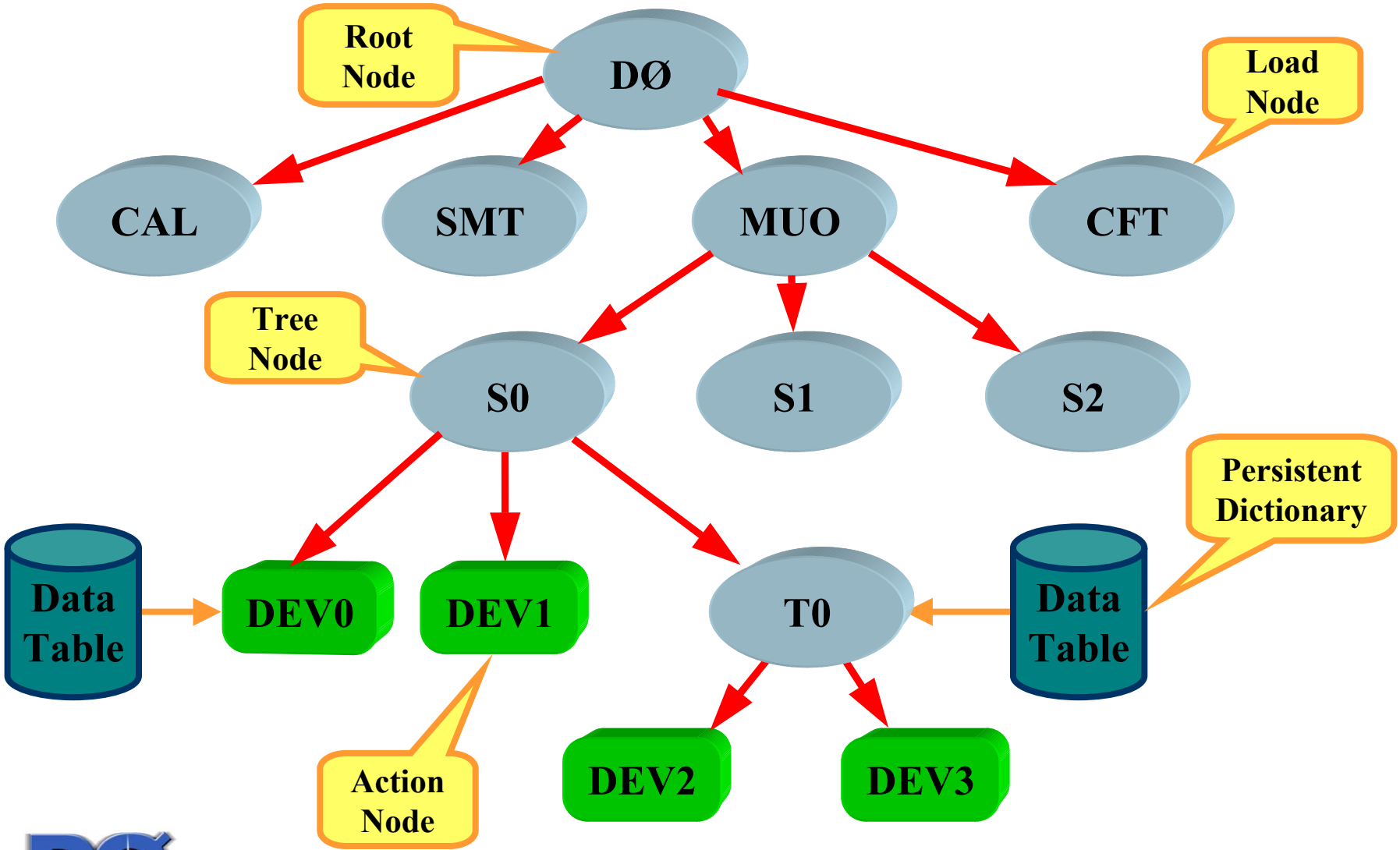
Characteristics of COMICS

- Current support is for EPICS but other control protocols may be added
- Tree definition is by Python program segments that are executed at run time (i.e. configuration files)
- Only the action (terminal) nodes in the tree invoke the control protocols

Characteristics of COMICS

- An action is completely determined by the parameter block object
  - Passed as a string message to the server
  - Components:
    - Path name of the base node of a sub-tree
    - One or more name/value pairs
- A GUI client for experts provides detailed control of the server
  - Enable/Disable of sub-trees
  - Test mode execution

Configuration Tree Structure



COMICS Operations

- Tree construction
  - At server startup
  - Nodes are all instances of classes
    - Tree nodes define structure and the order of node traversal (recursive descent, depth first, left to right)
    - Load nodes may delay the creation of sub-trees
    - Only action nodes access the detector (accelerator, telescope, …)
  - Structure is specified in a configuration file
    - Python script (code) fragment

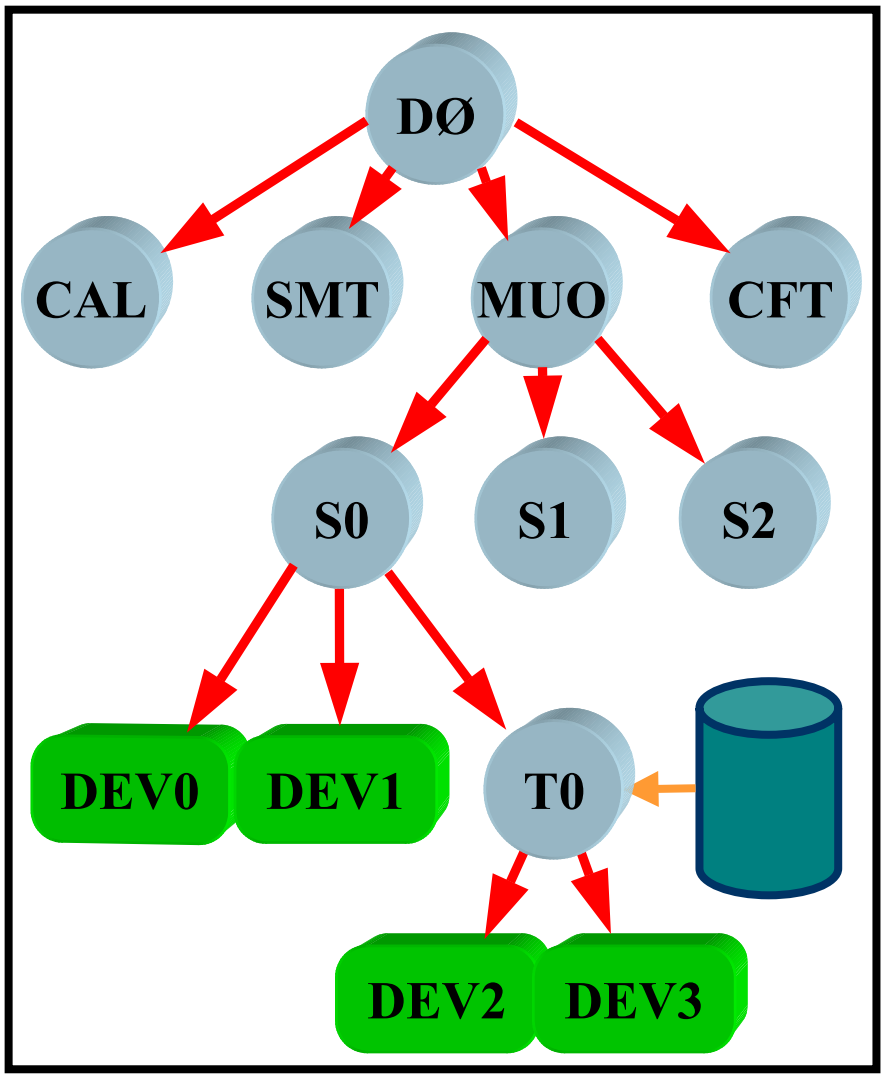
COMICS Operations

- Tree execution
  - A parameter block object is constructed from the command string
  - The execute method of the sub-tree base node is invoked
  - Each structure node invokes the execute method of its subordinate nodes
  - Each action node selects its operation based upon the contents of the parameter block object
    - An action node may do nothing

COMICS Operations

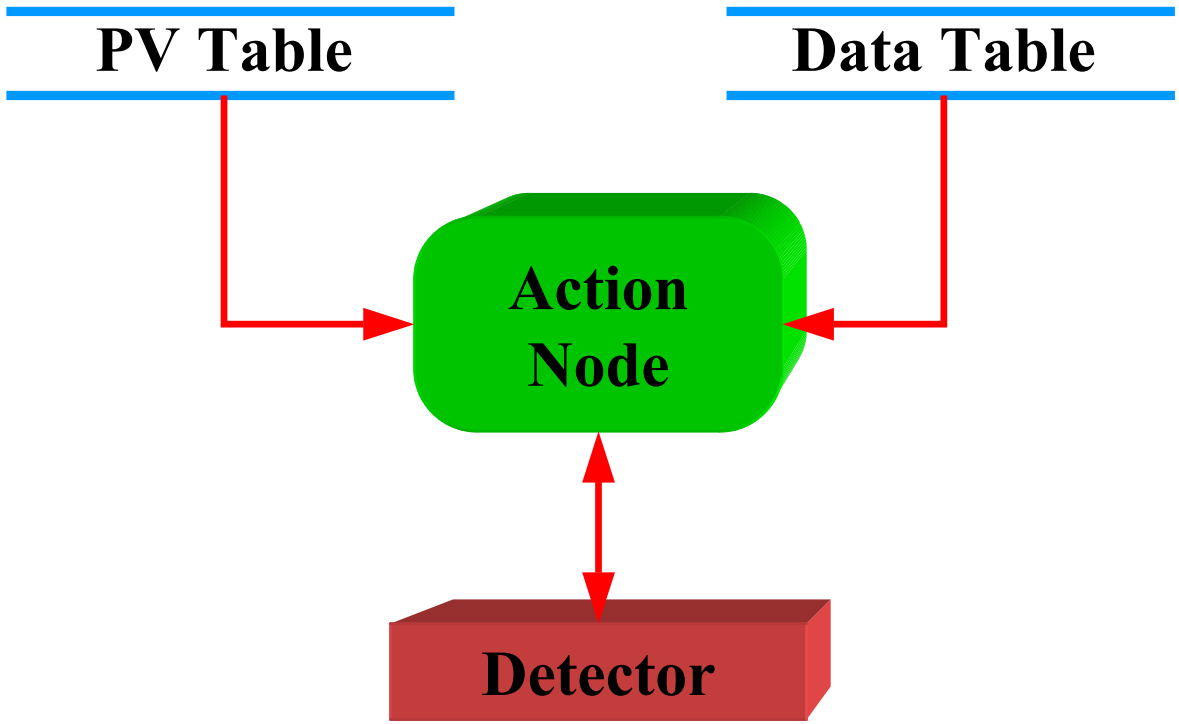
- The server writes a transaction file
  - Selectable levels of detail

Tree Configuration File




```
from ComicsNode import *
from ComicsHvc import *
from ComicsPvt import *

root = ComicsTreeNode('D0', None)
  cal = ComicsTreeNode('CAL', root)
   # … Calorimeter sub-tree
  smt = ComicsTreeNode('SMT', root)
   # … SMT sub-tree
  muo = ComicsTreeNode('MUO', root)
    s0 = ComicsTreeNode('S0', muo)
      dev0 = ComicsHvc('DEV0', s1, 'MUO', '00')
      dev1 = ComicsHvc('DEV1', s1, 'MUO', '01')
      t0 = ComicsTreeNode('T0', s1, 'Pvt:Config')
       dev2 = ComicsPvt('DEV2', t0, 'MUO', '02')
       dev3 = ComicsPvt('DEV3', t0, 'MUO', '03')
    s1 = ComicsTreeNode('S1', muo)
    s2 = ComicsTreeNode('S2', muo)
 cft = ComicsTreeNode('CFT', root)
   # … CFT sub-tree
```

COMICS Action Node



Action Node for HV Device

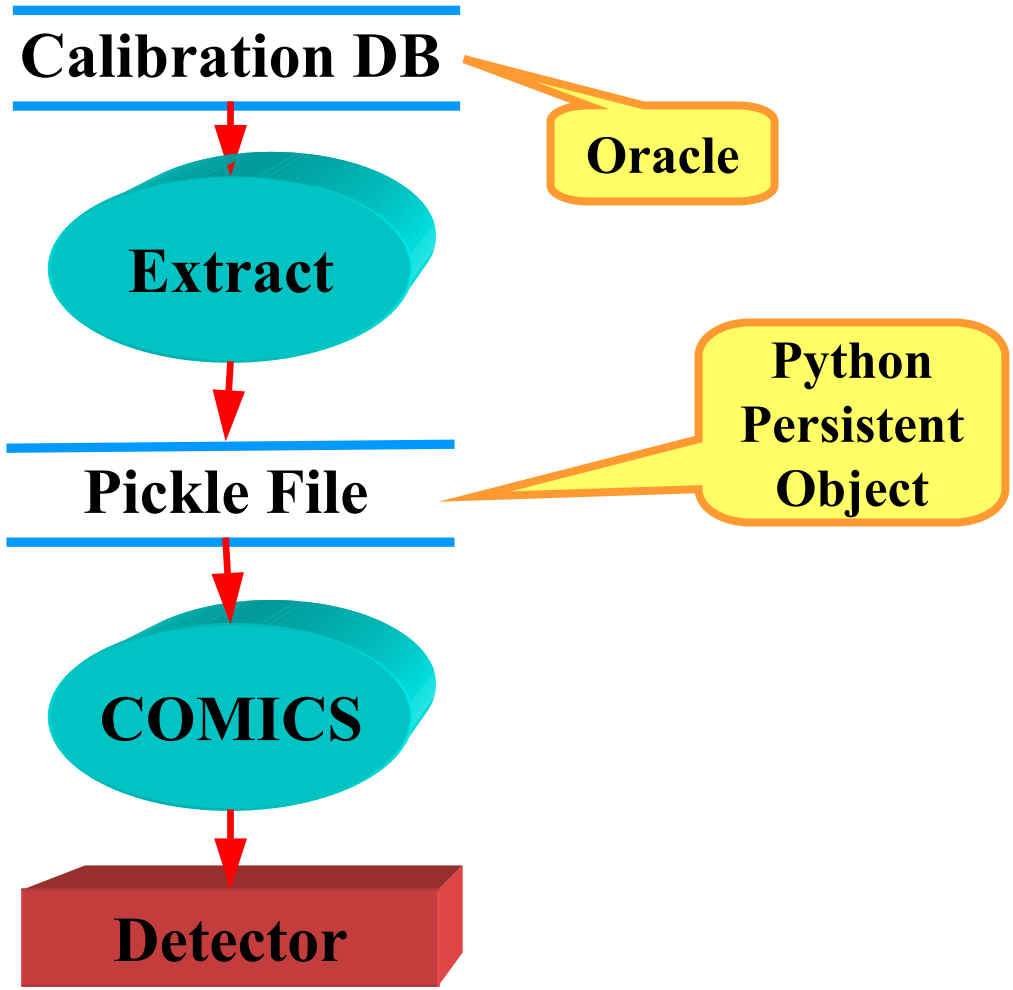
```
# Python program to load a Hv device

from ComicsNode import *

class ComicsHvc(ComicsActionNode):

  # Process variable table (PVT)
  PvTable = [
              ['/RATE', PvLoad, 1],
              ['/VTOL', PvLoad, 1],
              ['/MAXC', P vLoad,1],
              ['/CSCAL', PvLoad, 1],
              ['/STATE', PvVerify, 1]
             ]

  # Data table (DT)
  DataTable = [
               [123, 2.0, 150, 1.5, None],
               [150, 5.0, 50, 2.0, None]
              ]
```

```
def __init__(self,
             nodeName,
             parent,
             detName,
             locName):

    ComicsActionNode.__init__(self, nodeName,
        parent, detName, 'HVC',  locName)
    return

  def pvGet(self,
             pvTable):
    return ComicsHvc.PvTable

  def dtGet(self,
             runParams,
             dataTable):
    runType = runParams.paramGet('RUNTYPE')
    if (runType == 'STD'):
       return ComicsHvc.DataTable[0]
    elif (runType == 'CALIB'):
       return ComicsHvc.DataTable[1]
    else:
       return None
```

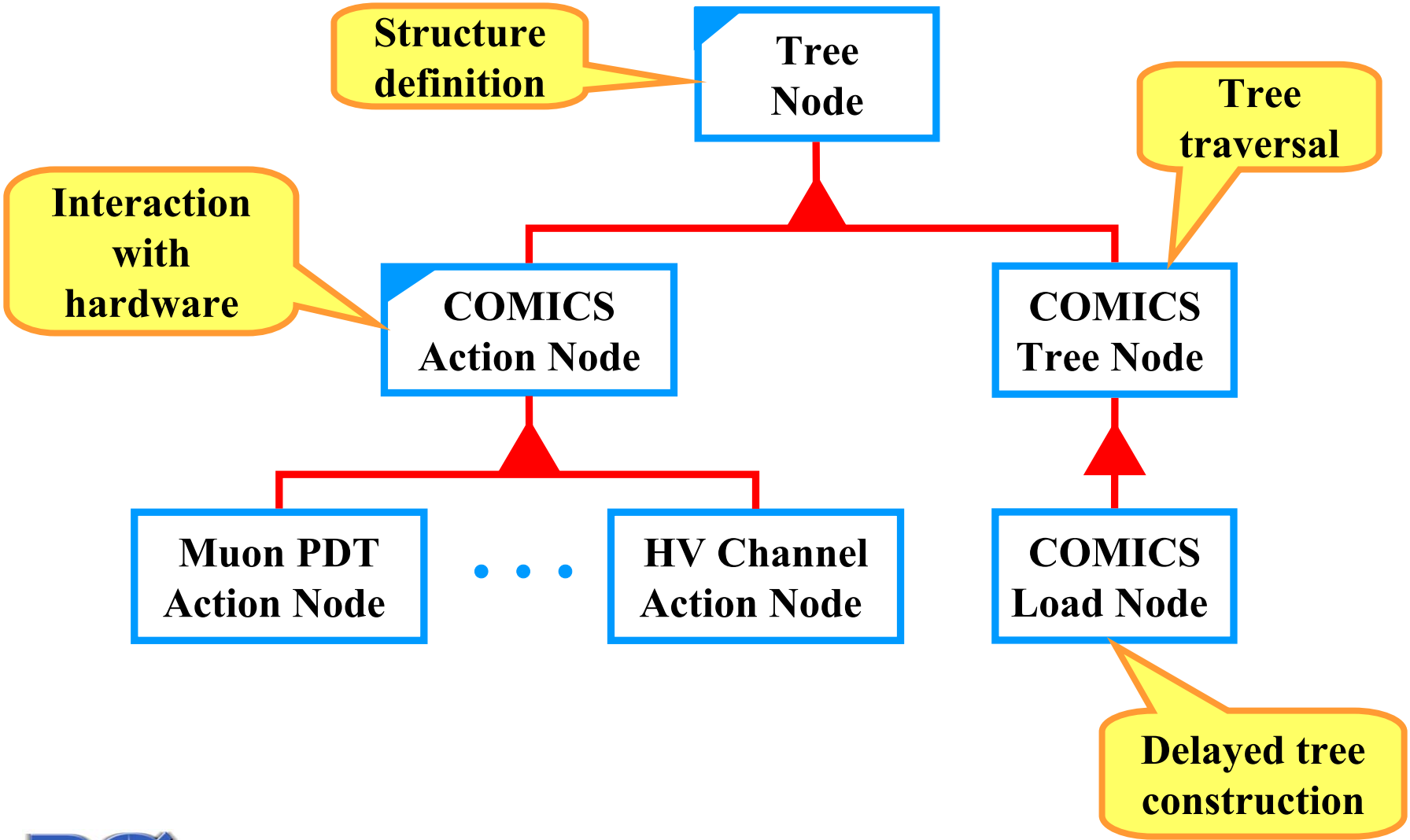
Sources for the Data Table

- Action node class variables
  - Assigned at coding time
  - Common to all instances of the class
- Action node instantiation variables
  - Assigned at tree construction time
  - Unique to instance of class
- Input file
  - Assigned at calibration time
- Parameter block
  - Name/Value pairs
  - Assigned at execution time

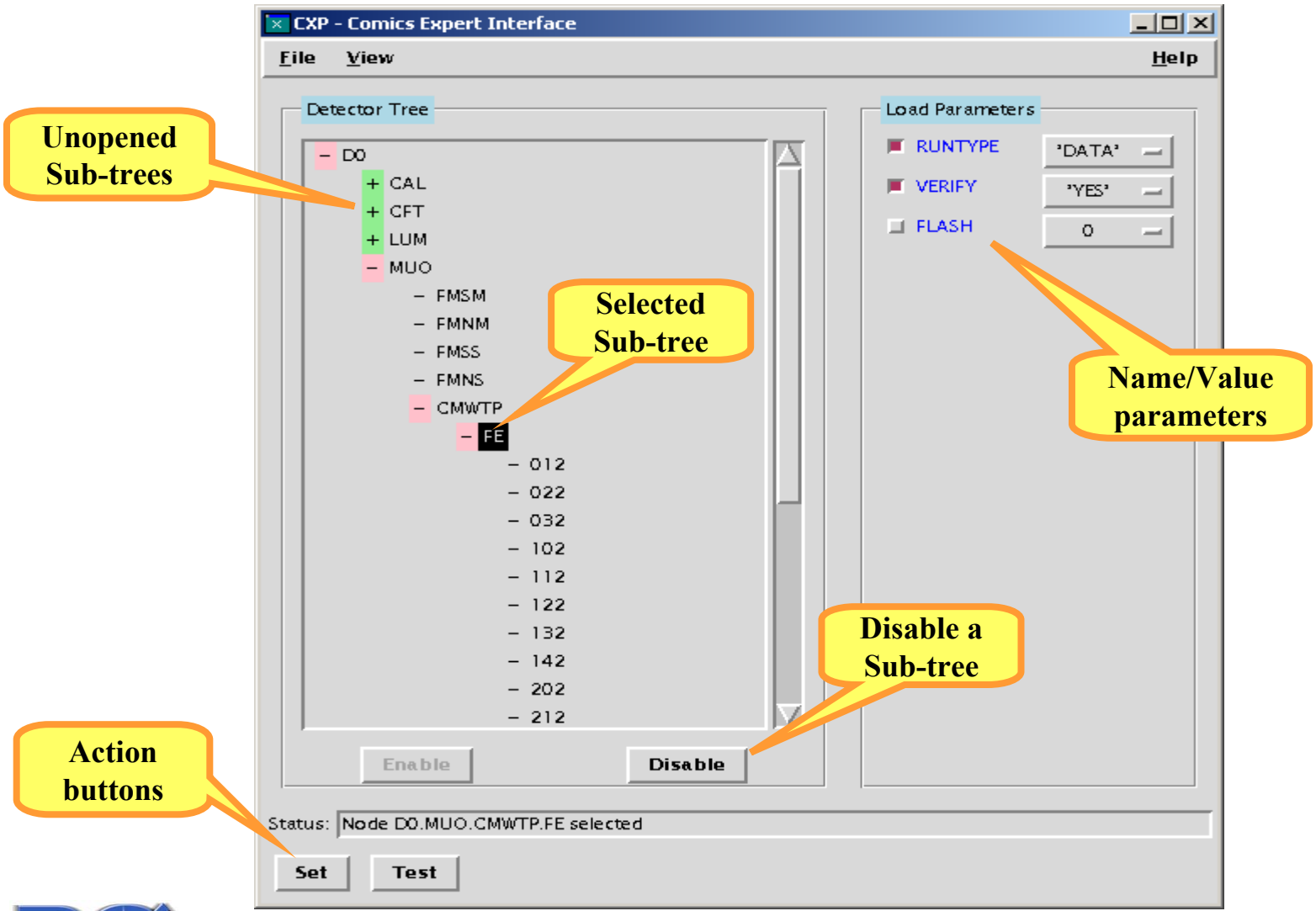
ORACLE Database Extraction



Class Inheritance Diagram



Expert Interface (GUI)



DØ Detector Configuration