Design and Implementation of the CEBAF Element Database*

Theo Larrieu, Michele Joyce, Chris Slominski

Jefferson Lab, 12000 Jefferson Ave., Newport News, VA 23606

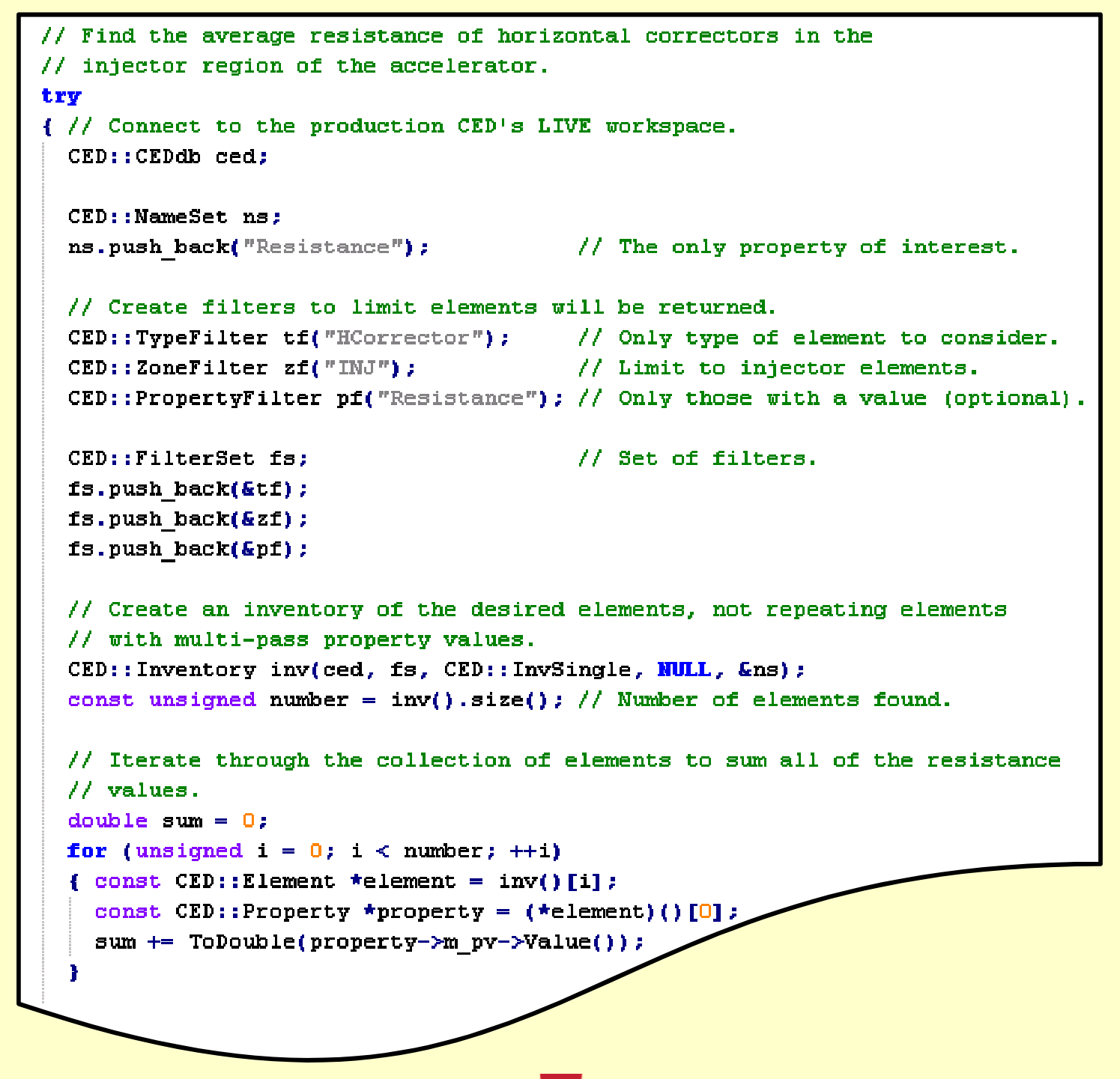
Abstract

With inauguration of the CEBAF Element Database(CED) in Fall 2010, Jefferson Lab computer scientists have taken a first step toward the eventual goal of a model-driven accelerator. Once fully populated, the database will be the primary repository of information used for everything from generating lattice decks to booting iocs to building controls screens.

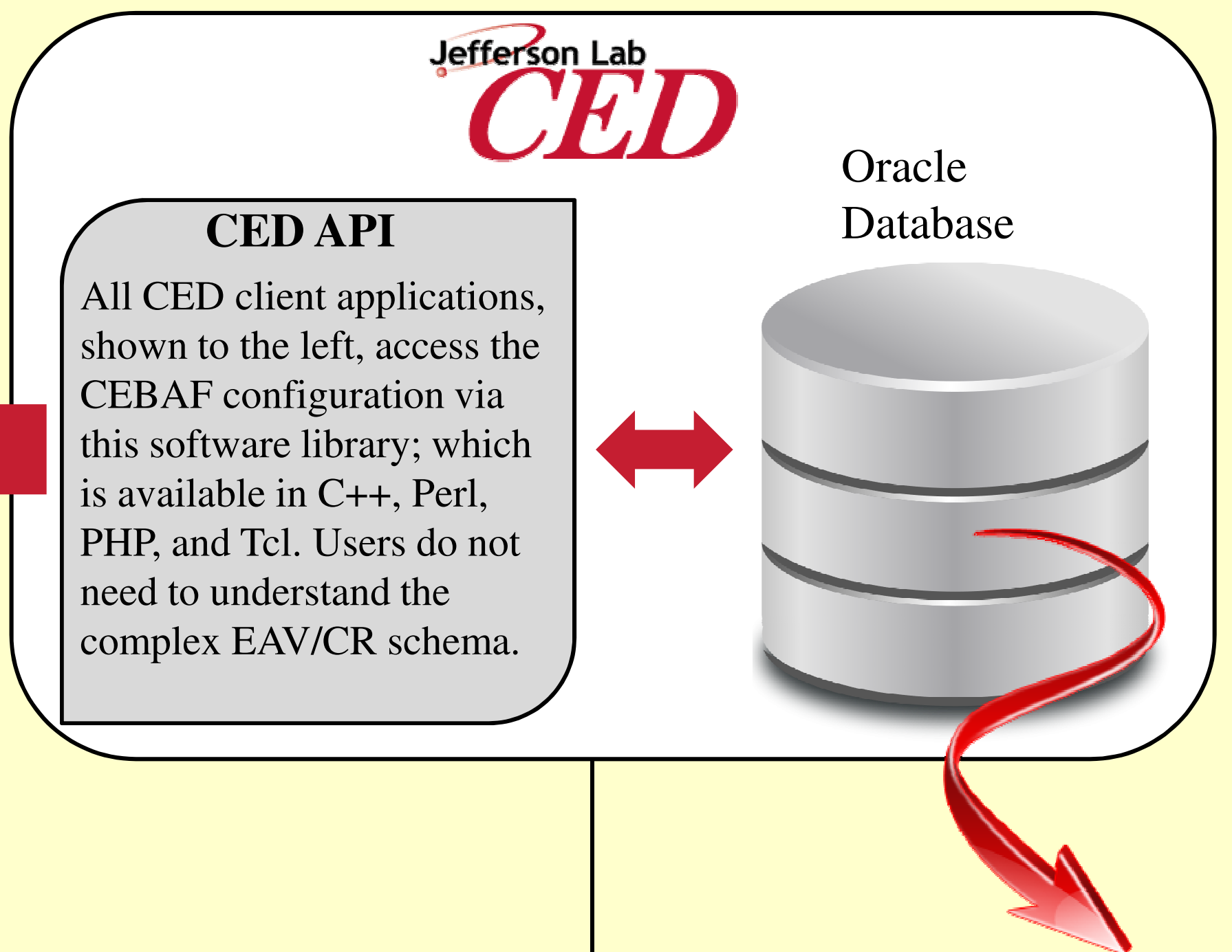
A requirement influencing the CED design is that it provide access to not only present, but also future, and eventually past, configurations of the accelerator. To accomplish this, an introspective database schema was designed that allows new elements, types, and properties to be defined on-the-fly with no changes to table structure. Used in conjunction with Oracle Workspace Manager, it allows users to query data from any time in the database history with the same tools used to query the present configuration. Users can also check-out workspaces to use as staging areas for upcoming machine configurations.

All Access to the CED is through a well-documented API that is translated automatically from original C++ into native libraries for script languages such as perl, php, and TCL making access to the CED easy and ubiquitous.

Applications use object oriented coding to access the CED. No SQL and privy knowledge of the schema is required. The example code is C++; other language constructs are similar.



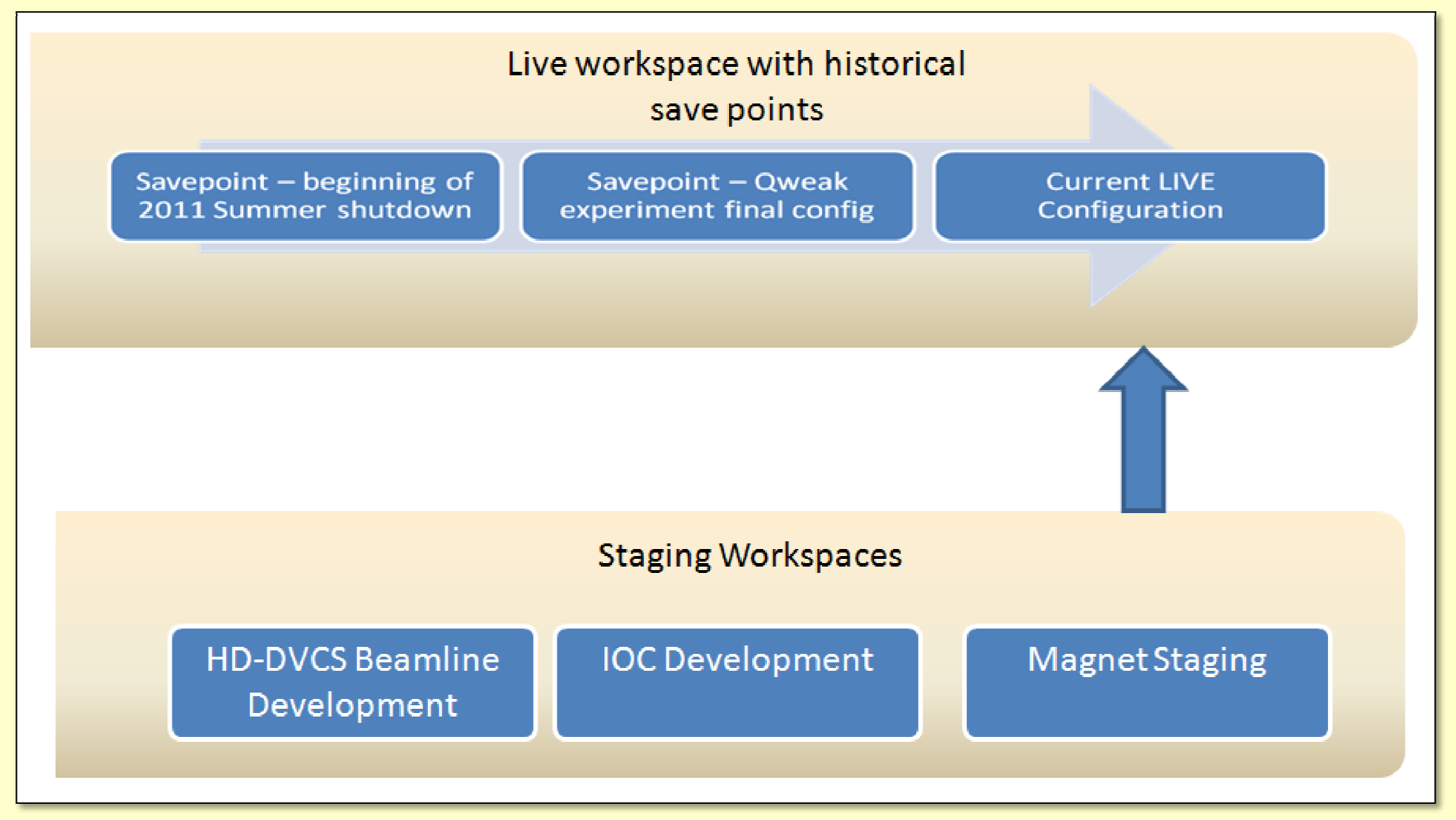
```
// Find the average resistance of horizontal correctors in the
// injector region of the accelerator.
try
{ // Connect to the production CED's LIVE workspace.
  CED::CEDdb ced;

  CED::NameSet ns;
  ns.push_back("Resistance");           // The only property of interest.

  // Create filters to limit elements will be returned.
  CED::TypeFilter tf("HCorrector");     // Only type of element to consider.
  CED::ZoneFilter zf("INJ");            // Limit to injector elements.
  CED::PropertyFilter pf("Resistance"); // Only those with a value (optional).

  CED::FilterSet fs;                    // Set of filters.
  fs.push_back(&tf);
  fs.push_back(&zf);
  fs.push_back(&pf);

  // Create an inventory of the desired elements, not repeating elements
  // with multi-pass property values.
  CED::Inventory inv(ced, fs, CED::InvSingle, NULL, &ns);
  const unsigned number = inv().size(); // Number of elements found.

  // Iterate through the collection of elements to sum all of the resistance
  // values.
  double sum = 0;
  for (unsigned i = 0; i < number; ++i)
  { const CED::Element *element = inv()[i];
    const CED::Property *property = (*element)()[0];
    sum += ToDouble(property->m_pv->Value());
  }
```



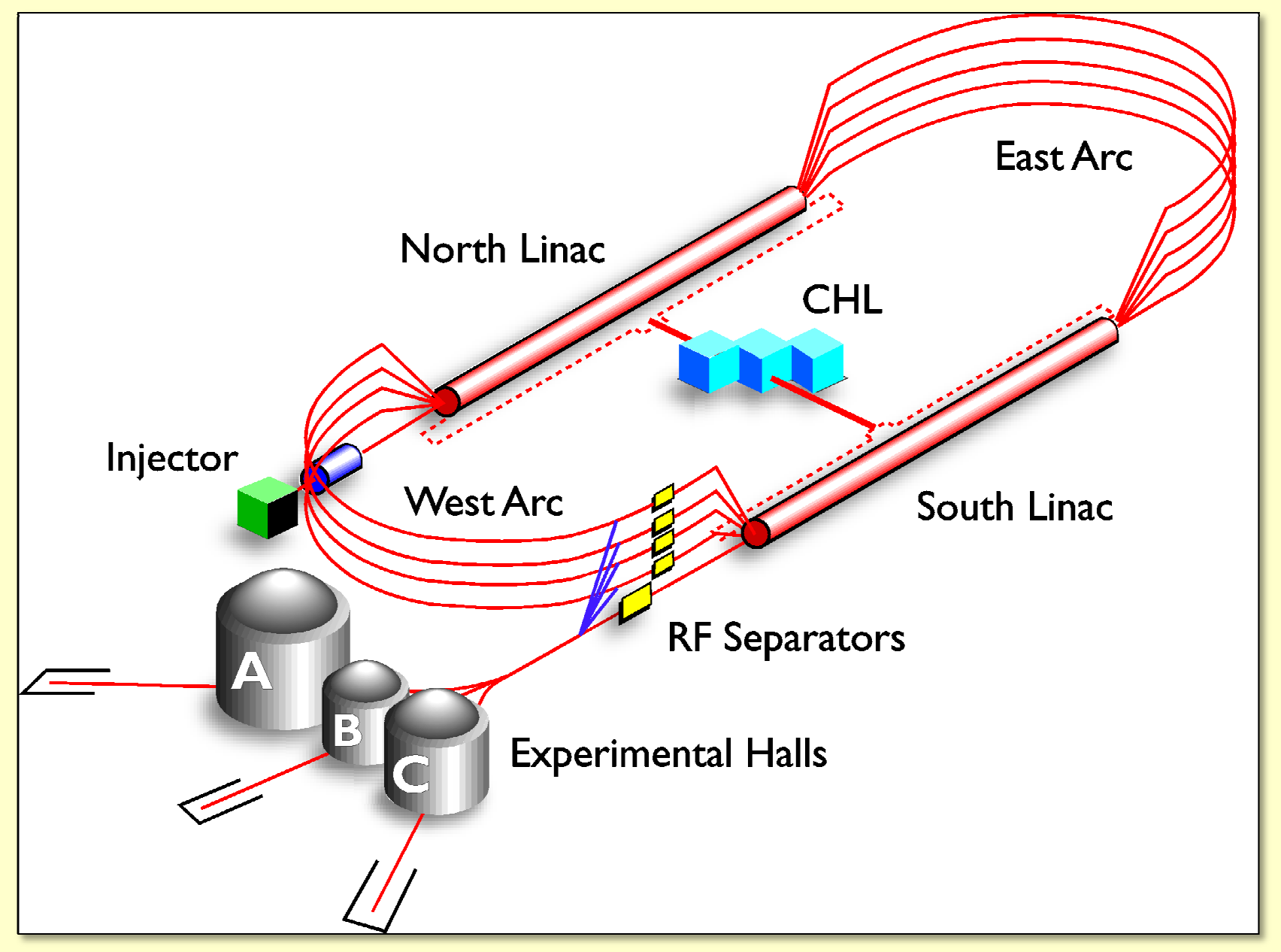
CED API

All CED client applications, shown to the left, access the CEBAF configuration via this software library; which is available in C++, Perl, PHP, and Tcl. Users do not need to understand the complex EAV/CR schema.

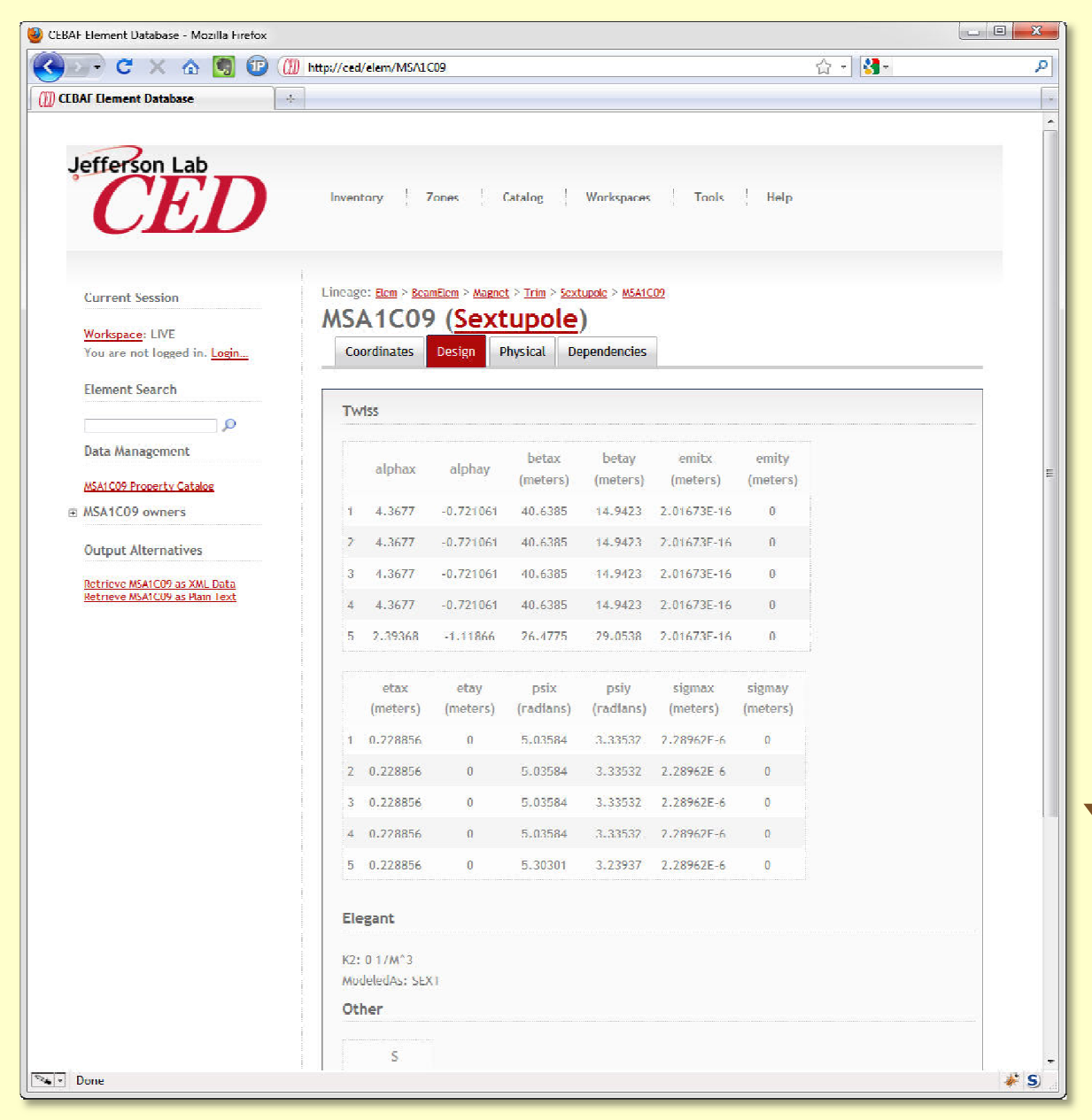
Oracle's Work Space Manager allows the CED to provide the production configuration of CEBAF, while developers construct future configurations and researchers analyze previously obtained results in the context of a past configuration.


Live workspace with historical save points
Savepoint – beginning of 2011 Summer shutdown
Savepoint – Qweak experiment final config
Current LIVE Configuration
Staging Workspaces
HD-DVCS Beamline Development
IOC Development
Magnet Staging

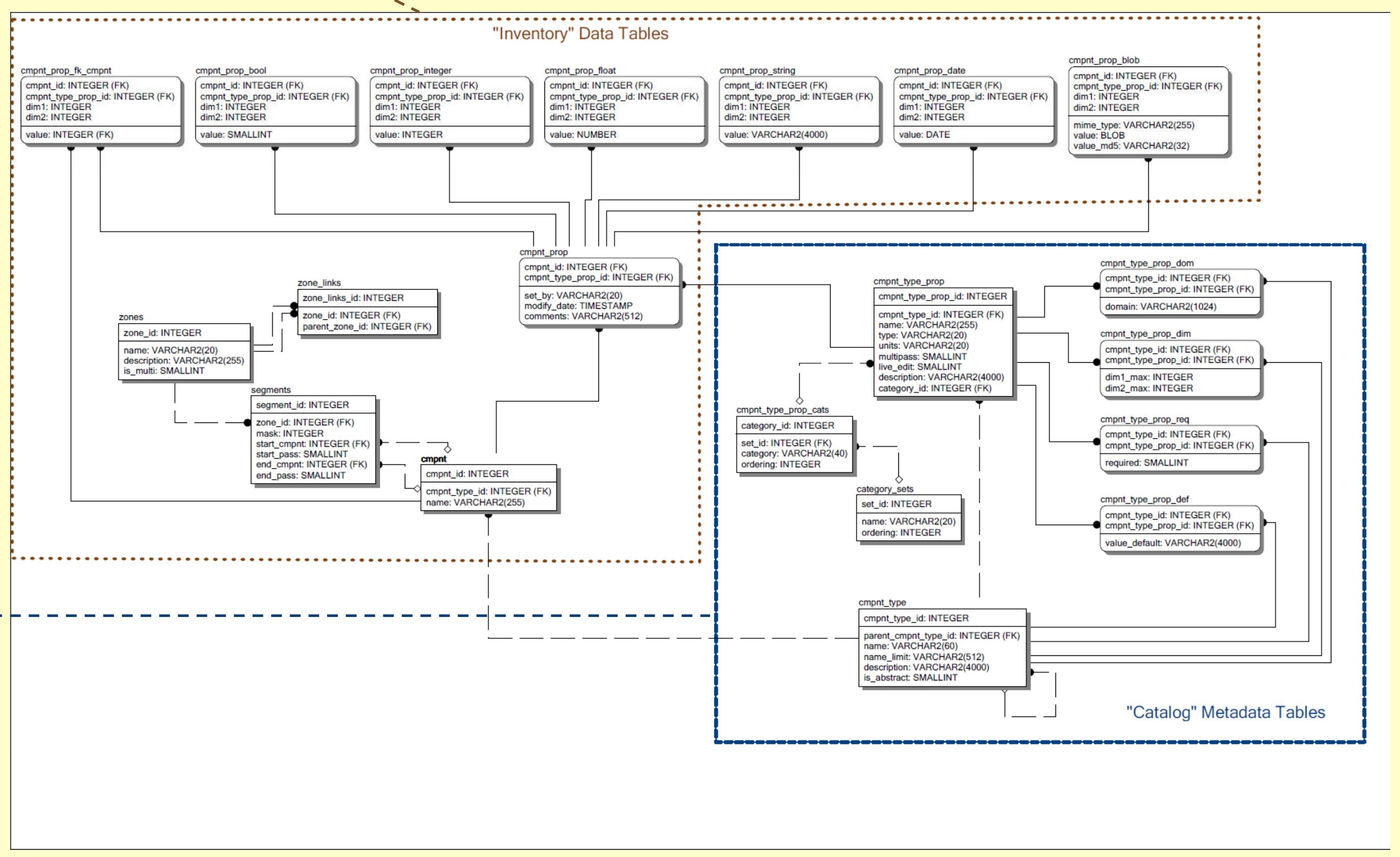
The CED will be used to setup, control, and diagnose the CEBAF accelerator. Software applications accessing the CED will include traditional high level physics applications as well as drivers in embedded hardware controllers.



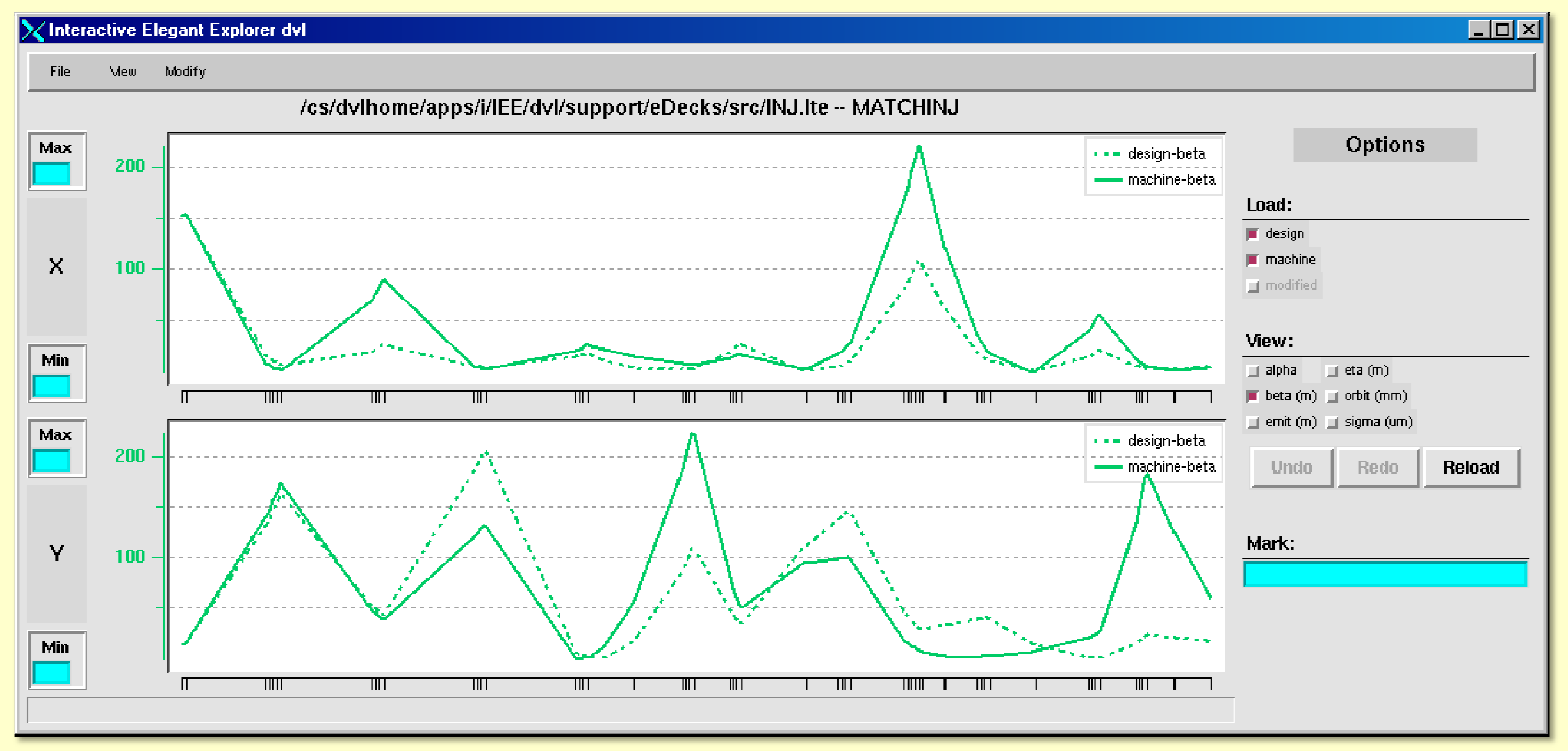
The CED serves as a centralized reference for CEBAF configurations. Scientists, engineers, and technicians browse the properties of installed equipment. Their viewport into the inventory of CEBAF devices is their web browser.



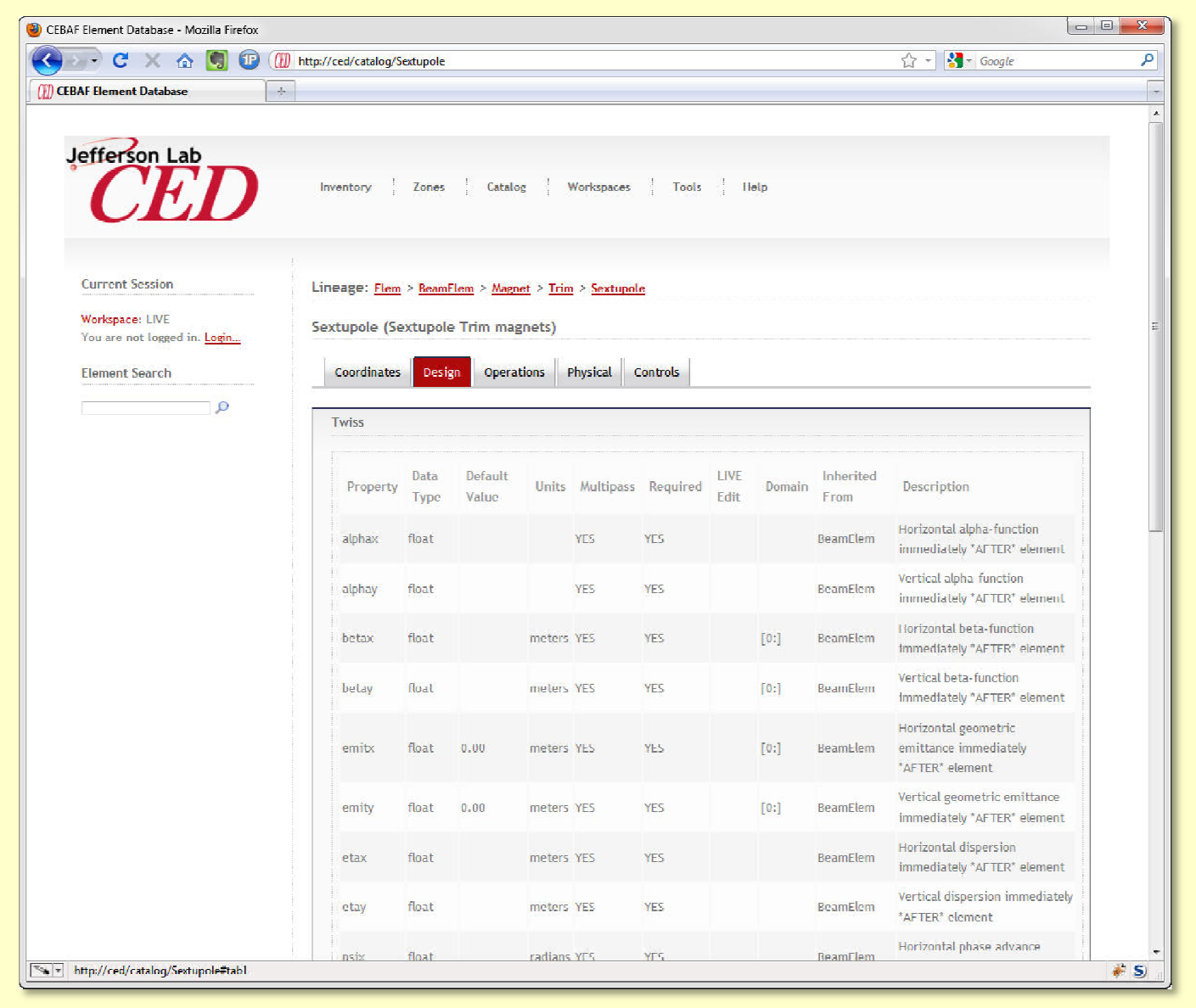
The Entity-attribute-value with classes and relationships (EAV/CR) schema offers a static schema in a volatile data structure environment. This type of schema means today's CED tools work with past CEBAF configurations and minimal staff support of the database and API.



The Interactive Elegant Explorer uses the CED to create a focusing lattice from a CEBAF configuration for the purpose of modeling accelerator beam transport.


Interactive Elegant Explorer dvl
/cs/dvlhome/apps/i/IEE/dvl/support/eDecks/src/INJ.lte – MATCHINJ
Options
Load:
View:
Mark:
Reload

Hardware system custodians and designers, in conjunction with CED administrators, define the various hardware element types and their properties via the web interface.



*Authored by Jefferson Science Associates, LLC under U.S. DOE Contract No. DE-AC05-06OR23177

Also see https://cebaf.jlab.org/CED/