Upgrade to User Interfaces

From AES-CS

date:	2013 August 12
to:	John Maclean, Associate Division Director, AES
from:	Pete Jemian, Group Leader, AES/BCDA
	Richard Farnsworth, Group Leader, AES/Controls
	Nicholas Schwarz, Group Leader, AES/SSG
	Tim Mooney, AES/SSG
subject:	Replacement of MEDM as Primary EPICS User Interface at APS
ICMS:	APS_1438838

The charge to the committee:

1. Why replace MEDM?
2. Options considered, including
    1. advantages and disadvantages of each
    2. the work required to make each a viable product
    3. how each would fit into our architecture
3. Performance comparisons
4. The results of any pilot implementations
5. Recommendations, including plans for deployment

Contents

Why replace MEDM?

The APS developed MEDM as the main user interface software (GUI) for the initial operations in the early 1990s. The default GUI provides access to the many features of the control system as well as documents those features made available to the user. MEDM has been used to satisfy an objective expressed in the development of the APS to provide an interface similar to that of an automobile dashboard. On the automobile dashboard, all common features are easily recognizable even though they may appear in different representations or locations between automobiles.

While providing many years of operations for both main control room and beam lines, the MEDM software has become obsolete, the cost of maintaining it has increased as APS computing systems have evolved, and it has always fallen short of what the users sought in a GUI. Since its creation, other tools in the EPICS community have been developed but none offered significant improvements over MEDM until the development of CSS from DESY and addition of BOY from SNS/ORNL. Appearing shortly thereafter, initiatives in C++ and Qt established other tools, including epicsQt and caQtDM.

MEDM is built using the Motif (http://www.opengroup.org/motif/) toolkit. Increasingly, this is seen as problematic for developers and the continued life of MEDM at the APS. Motif has been replaced with OpenMotif and the license does not allow it to be ported to non-open operating systems. To provide MEDM as a native-built executable for Microsoft Windows is a problem which has no solution. Since the Windows kernel is proprietary, *nobody* can legally port OpenMotif to Windows directly or via Cygwin. MEDM is currently built for Windows on all versions of Windows XP through Windows 7 (both 32 and 64 bit) by building (legally) against the Motif libraries in a proprietary product called Exceed (http://connectivity.opentext.com/products/exceed-products.aspx) (formerly Hummingbird eXceed), a commercial X-server. In order to run MEDM, an X-server (although not necessarily Exceed) must also be installed and running. This means that MEDM on Windows has an uncontrollable third party dependency.

The vanishing support for Motif illustrates why we must consider now making a change from the Motif-based Editing and Display Manager (MEDM). That the OpenMotif licensing prohibits it running natively on proprietary product operating systems such as Window contributes, as well as helps define, the requirement that the successor to MEDM must have native multi-platform support.

Looking towards the future, images from 2-D cameras will play an increasing role in the advancement of science at the APS. MEDM has no support for such images.

Additional reasons why the APS should replace MEDM are given in the MEDM section below.

Built on the Eclipse (http://eclipse.org) open-source platform from the business community (which uses Java (http://www.java.com) as its language), several facilities in the worldwide EPICS community are investing in the support of CSS BOY. The APS added some software to preserve our 15+ years investment in user-interface screens. In return for our investment in EPICS, we receive modern, community-supported software that was engineered for the right level of user. APS has one developer who participated in the CSS BOY developer team and made contributions to CSS BOY of value to the APS community.

The two C++/Qt candidates described below have communities that, at this time, are smaller than the number of facilities using CSS BOY. However, the APS has a developer team well-versed in C++ and the Qt toolset, such that code maintenance issues and support are far less risky than with CSS BOY. It is noted that amongst the C++/Qt projects and the C++/Qt developers at the APS, they all use different EPICS-aware Qt widgets. Are these projects close enough that they can agree on "widget synergy" (where the same widgets and screen definitions can be interchanged between these tools)? APS investment here might be able to ensure this synergy occurs.

At this time, there is no immediate need to replace MEDM as it continues to meet basic controls requirements at the APS. Rather, there is a strong desire expressed within the APS community to identify the successor to MEDM and begin using that successor as soon as possible. The pressure from the development of new beam lines places a sense of urgency on this decision, that they avoid development with an infrastructure perceived to be outdated.

requirements for the next graphical user interface at APS

All requirements must be met. Some may be required for initial deployment while others may be required later. We believe there is sufficient overlap in the requirements from the APS main control room and the APS beam lines that a common tool can replace MEDM at the default EPICS across the APS.

requirements for the next graphical user interface at APS

program	MCR	beam lines	support staff	developers
Widget and feature set at least inclusive of the MEDM widget set	*	*	*	*
Must have the ability to start command-line tools		*	*	
Must have the ability to show a "related display" window	*	*	*	
Must be able to bundle screen descriptions in specified subdirectories.	*	*	*	*
Must be able to override standard displays with displays customized for a beamline or experiment.		*		
Must be able to run with edit capability disabled				
Must support EPICS v3 IOCs and Channel Access			*	*
Must have a GUI editor suitable for use by a non-specialist end user	*	*	*	
Must display 1-D and strip chart plots	*	*	*	
Must be able to display a PV's alarm state	*	*	*	
Must be able to display a PV's access-security state	*	*	*	
Must accept or convert existing .adl files	*	*	*	*
Must have performance comparable to MEDM	*	*	*	
Must run on modern computing platforms	*	*	*	*
Must be robust:	*	*	*	*
Must run for weeks with undiminished performance (no memory leaks, etc.)	*	*	*	*
Must handle IOC disconnects and reconnects gracefully and promptly	*	*	*	
Must start even if PVs are not available	*	*	*	*
Must recover from temporary update overload gracefully and promptly	*	*	*	*
Must be able to show 2D visualization		*	*	
scan plots		*	*	
areaDetector images, including user interaction		*	*	
camera images, including user interaction		*	*	
support a configuration of displayed screens	*	*		
save current arrangement of open screens	*	*		
support a configuration spanning multiple windows and/or workstation workspaces		*		
restore saved configuration on restart	*	*		
Code must be part of collaborative effort in the EPICS community				*

Code must build on variety of modern platforms				*
Must be able to compose a display from one or many instances of included display fragments		*		*
Must handle EPICS' long strings		*	*	*

(*) MCR: APS main control room

feature wish list for the next graphical user interface at APS

Additional desired features that are not required in a successor to MEDM but are of sufficient value in a user interface to record them here:

feature wish list for the next graphical user interface at APS

program	MCR	beam lines	support staff	developers
consistent with contemporary user interfaces	*	*	*	
tabbed display, menus, keyboard shortcuts, tooltips, drag and drop feature		*	*	
option switch for user interface MEDM-compatibility			*	
support EPICS v4				
archiver and integration with existing archive tools (such as SDDS)	*			
access security display	*	*	*	
scripting		*	*	
easy charting		*		
any PV against another		*	*	
configurable at run time by any user		*	*	
user interaction with charts		*	*	
remote access				
native multi-platform support for smartphones and other newer technology				
screen description files should be text	*	*	*	*
be able to control the text of the subwindow title		*	*	
widget synergy - widgets reuseable in other programs, such as custom beamline-specific or experiment-specific software			*	*
Does not require a learning curve to which we are not already committed, to maintain, extend, or troubleshoot.				*
exception reporting/logging		*	*	*
Notify user about multiply defined PVs (e.g., two IOCs that both host the same PV name)		*	*	*
PV not responding		*	*	*
user input errors (local to this client and session)		*	*	
open-source and open to collaboration		*	*	*

In addition to these features above that are desired are wishes expressed by various beam lines:

- Each beam line wants ...
    - ... its local network to be wholly autonomous
    - ... to be able to fall back to MEDM during the transition period
    - ... to use the new GUI the same way they used MEDM (!)
    - ... to share the same screens at beam line workstations, allow for local modifications
    - ... to run instances of the same GUI from concurrent sessions on the same workstation and account
- Convert existing suite of MEDM screens to the new GUI and provide locally on the sector DSERV
    - synApps provides ~900 (MEDM) .adl files
        - Needs functional testing (waiting on easy deployment)
    - Accelerator provides (MEDM) ~4000 .adl files
        - Beam lines probably use ~100 of these, especially storage ring stripchart of storage ring current
- Provide developers easy access to screens defined locally for each local beam line

Use Cases

Aside from using MEDM to visualize the controls for a few screens and related displays, some parts of the APS have worked out ways to open many screens with a single command or to arrange many screens in a common way.

2-ID-D

Runs a shell script that opens about 80 MEDM screens across 9 workspaces. It is actually two scripts

```
/home/beams/USER2IDD/bin/start_epics_2idd_linux
```

sets up EPICS_DISPLAY_PATH and calls

```
/home/beams/S2IDD/bin/expmanager/setups/start_default_linux.setup.R5.6
```

which opens and places medm screens, with calls like the following:

```
wmctrl -s 0
medm -attach -x -dg +5+5 -macro "P=2idd:,S=scaler1" scaler16_full.adl &
sleep 1
medm -attach -x -dg +500+5 -macro "s=02" beamHistory_full.adl &
```

Note that

```
wmctrl -s 0
```

sends a message to the X window manager, switching to workspace 0.

APS Main Control Room

The APS Main Control Room (MCR) operator display setup relies on the host machine desktop tabs being named properly; this is a somewhat complicated process unless you know exactly how the program works.

Setting up of the MCR operator displays is done through the OAGapps program by selecting from the main menu

"Miscellaneous" and then selecting "Machine workspace setup" from the sub menu that is launched from the Miscellaneous menu. The information that the program needs is all located at /home/helios/ASDOPS /workspaceSetups for various host machines. Within the subdirectory ted are various sdds files for host machine resource and workspace setups. This directory contains the files necessary to set up the host machine workspace name tabs properly as well as setting up environmental resources.

Within the directory /home/helios/ASDOPS/workspaceSetups/wspSetups are various sdds files containing the MEDM commands to launch in the different workspaces on the host machine various virtual desktops. All of these sdds files can be viewed using the sddedit command. All of the MEDM displays are then launched in appropriate workspaces on the desktop to provide the same look and feel for the operators and give a consistent presentation. All of the MEDM displays are launched attached to the same MEDM client. This helps to provide a single per host machine Channel Access client which can connect to the EPICS Process Variables in each of the IOC crates located around the accelerator.

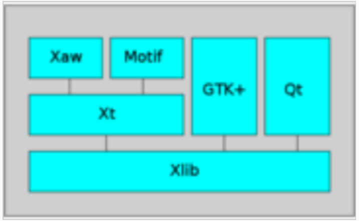
All MEDM displays are located within subdirectories of the directory on the helios file system: /usr/local/iocapps /adlsys and this path is included in the user's environment variables at login. Other required environment variable include EPICS_HOST_ARCH and EPICS_EXTENSIONS.

Options considered

To transition from MEDM to the next generation tool, we considered EPICS open-source software tools that have been discussed on the EPICS tech-talk email discussion list or have been demonstrated as part of EPICS meetings. There are four main categories of such tools:

- Motif and X11
- C++ and Qt
- Eclipse / RCP
- web applications

To understand how these tools compare in perspective of their underlying graphics toolkits, see this graphic from Wikipedia (http://en.wikipedia.org /wiki/File:X-client-libraries.svg) that shows how (in X11 environments such as Linux) Motif, Qt, and GTK (underlies eclipse on Linux) exist as widget toolkits in the X11 software stack.



Motif, Qt, and GTK+ exist as widget toolkits in the X11 software stack

We have not considered web applications as part of this analysis. In the following table, we show the tools we believe to fit these basic criteria for consideration as successors to MEDM. For the purposes of baseline comparisons, we also include similar assessments of MEDM and its contemporary, EDM. The assessment of each of these tools appears later in this section.

GUIs for EPICS at APS

program	UNIX	Linux	Windows	Mac OSX	files	ADL converter?	built using	maintained by
[medm (http://www.aps.anl.gov /epics/extensions /medm)]	yes	yes	yes(*)	yes(*)	.adl	not needed	C++ and X11/Motif	APS

[edm (http://ics-web.sns.ornl.gov/edm/)]	yes	yes			.edl	[yes (http://ics-web4.sns.ornl.gov/adl2edl/log/show.php?mon=Apr&theDay=12&year=2004)]	Motif	SNS
[caQtDM (http://epics.web.psi.ch/software/caqtdm/)]		yes	yes		.ui	included	C++ and Qt	PSI
[CSS BOY (http://sourceforge.net/apps/trac/cs-studio/wiki/BOY)]		yes	yes	yes	.opi	integrated	eclipse RCP	SNS
[epicsQt (http://sourceforge.net/projects/epicsqt/)]		yes			.ui	not available	C++ and Qt	AuS

(*) Requires additional X11 support package to be added to OS

Archive of an email discussion thread on the topic of "EPICS developers interested in the QT Initiative" may be found online:

- http://www.aps.anl.gov/epics/qti-talk/index.php

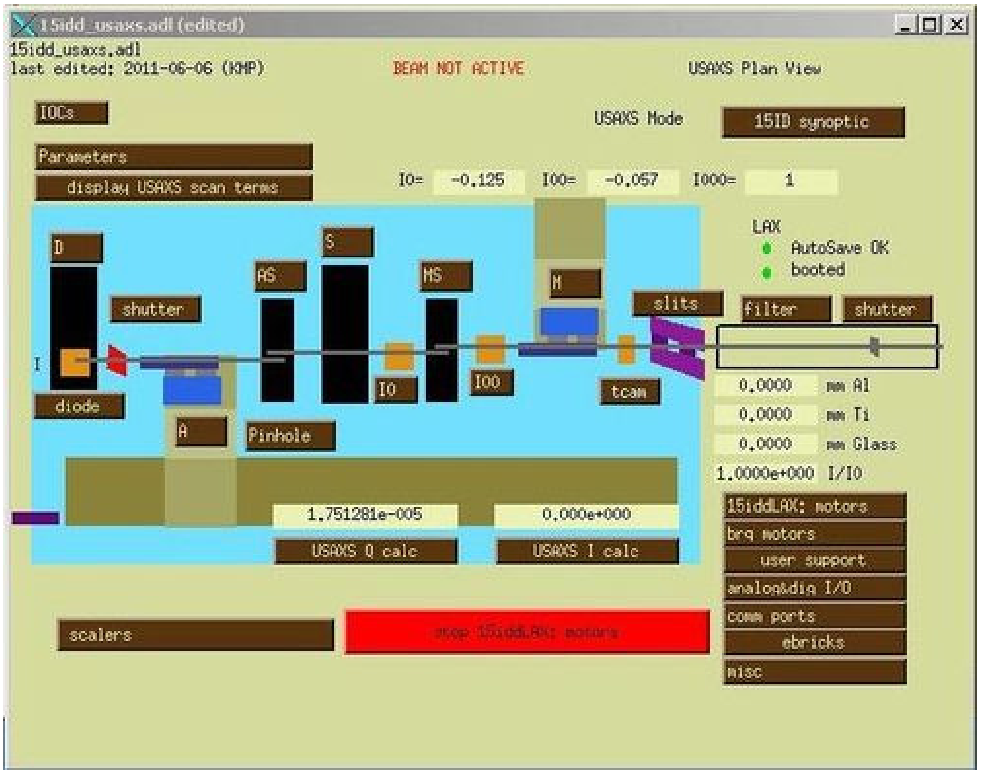
The various tools in the EPICS community using Qt appear not to have coordinated in their development of EPICS-aware widgets. As such, the widgets are not necessarily interchangeable. This may be a temporary situation, correctable through collaboration. The path is not clear but the value of such collaboration is evident.

MEDM

- http://www.aps.anl.gov/epics/extensions/medm

The Motif Editor and Display Manager (MEDM) software is an interactive graphical display tool to construct EPICS graphical user interface displays. It was created for the APS to combine the functions of EDD and DM and to use the Motif widget toolkit. It was created at the APS and is currently managed by the APS Software Services Group. It is written using C++ and Motif. (Motif, lessTif, or OpenMotif may be used to build MEDM.) It also needs a graphics package which is not included: either SciPlot (open source) or XRT Graph (licensed). It must run on an X11 server.

- MEDM example screen from the USAXS instrument at beam line 15-ID



reasons why the APS needs to move away from MEDM

- There is no one remaining at APS or in the EPICS community that knows the MEDM code base well-enough to add new features
- Higher risk that MEDM will stop working with any OS upgrade or patch
  - OS upgrades and patches must be accepted and thus present a risk to continued use of MEDM
  - Need to identify and change on our own schedule (pro-active v. reactive)
- It lacks the functionality of modern graphical user interfaces
- It has an antiquated look and feel
- Lacks features expected of a modern control system
  - scripting
  - flexible charting (any PV against any PV, historical data)
  - remote access
  - native multi-platform support for smartphones and other newer technology
- Depends on X11 and Motif, both vanishing technologies

- MEDM is out of date

advantages and disadvantages

1. + APS uses this now
2. + robust and stable
3. + used worldwide for many years
4. - interface is dated
5. - continued support costs are rising
6. - MEDM falls short of what APS users wanted in a standard GUI environment
7. - not easily extensible
8. There is a significant advantage to having the developer “In House”

   1. + APS had such a developer for many years
   2. - There is no such assigned developer today

9. - underlying technologies are becoming harder to install and deploy
   1. Motif, as noted above
   2. XRT/Graph (http://www.bci-bluestone.com/products/xrt/) is a licensed product ($4k/y at APS), end-of-life is anticipated within a decade

the work required to make this a viable product

1. APS uses this now
2. Training of a new Motif/X11 developer to support MEDM will be significant

how this would fit into our architecture

1. APS uses this now
2. provides a GUI for process variables from EPICS IOCs
3. Our architecture is changing: e.g., MEDM does not support images from EPICS Area Detector

EDM

- http://ics-web.sns.ornl.gov/edm/

The Extensible Display Manager (EDM) software is an interactive graphical display tool to construct EPICS graphical user interface displays. It was created as an alternative to MEDM. It was created and is currently managed by John Sinclair at the Spallation Neutron Source (SNS, ORNL). It is written using C++ and Motif. (Motif is required to build EDM. OpenMotif has been used on Linux platforms successfully. LessTif is not currently recommended.) It must run on an X11 server.

advantages and disadvantages

1. - It has the same Motif and X11 basis as MEDM. It has the same end-of-life issues.
2. - EDM's .edl files are not compatible with MEDM's .adl files
3. + A tool exists to convert .adl files to .edl
4. - No tool exists to convert .edl to .adl because EDM has more widget types, some of which do not convert to MEDM widgets
5. - code is dependent on a *single* programmer (at ORNL)

the work required to make this a viable product

1. in use now in some places at the APS as it came from vendor hardware
2. distributed as a tar.gz file

how this would fit into our architecture

1. It appears that EDM is comparable in its feature set to MEDM and does not offer a realistic upgrade in capabilities for the APS.

The only significant difference between EDM and MEDM is the look and feel.

caQtDM

- http://epics.web.psi.ch/software/caqtdm/

The Channel-Access Qt-based display manager (caQtDM) is a feature-based replacement of MEDM written to run on the Microsoft Windows and Linux operating systems. It was created and is currently managed by Anton Mezger (Head of Accelerator Operations) of the Paul Scherrer Institut. It is written using C++ and Qt.

It appears that no intention has been made to support the Macintosh OSX operating system. It may be possible, or even trivial, to build caQtDM for the Mac, which is similar to Linux, but that effort has not yet been expended.

advantages and disadvantages

1. + Nearly exact replacement of MEDM
    1. + synApps' More/Less buttons work as intended
    2. + Displays come up with the expected sizes
    3. - Can't drag-and-drop PV names
    4. - Some displays require adjustment of text-box sizes
2. Extension requires learning curve (C++/Qt)
    1. + SSG already uses C++ and Qt
    2. learning curve limited to caQtDM code (same as any other)
3. - code is dependent on a *single* programmer (at PSI)
4. + epicsQt widgets can be used in caQtDM displays.
5. + contemporary user interface, includes tabbed displays
6. + can be extended
7. + support for EPICSv4 (PV Access) protocol

work required to produce a viable product

1. A test build has been deployed to /APSshare, and has run at 2idd. epicsQt widgets should be added, and this installation should be built against a version of epics base on /APSshare.
2. Save/restore placement of many displays on many workspaces
    1. Placing displays has already been demonstrated.
    2. caQtDM maintains the information required to write a script that would restore the placements of a running collection of displays, but it does not have code to write such a script. This might take a day of work.
3. Support drag-and-drop of PV names. (Dropping the current text selection onto a link-PV widget works, however.)
4. Work should be done to investigate the feasibility and usefulness of sharing C++/Qt code (e.g., widgets) among Qt-based EPICS tools. For example, it might be practical for experiment-control software that runs on several beamlines to handle beamline-specific devices or techniques by including beamline-specific caQtDM .ui displays. Similarly, beamline-control user interfaces might be enriched by using widgets or code fragments from experiment-control software. APS investment here might be able to ensure this synergy occurs.
5. For each beamline or beam station: translate local custom adl displays; edit a script to launch caQtDM with the appropriate display path. This took about five minutes for 2idd.
6. Translate medm displays for older versions of synApps. (synApps 5.6 and 5.7 have caQtDM files already.)
7. Write better docs on the caQtDM widgets, and an abbreviated caQtDM-specific doc on using Qt Designer.
8. Improve the caQtDM build. Right now, it's basically a recipe.

9. Edit selected displays that didn't translate optimally.
10. Build for Mac OSX. Qt supports Mac OSX as a first-tier architecture, so there should not be a problem, but a build has not been demonstrated.

how it would fit into our architecture

- Closest replacement of MEDM of all the possibilities listed here
- Development of a new widget type (display of 2D data acquired by the sscan record) has been demonstrated here, and the development process has been documented. It requires knowledge of C++/Qt, and some caQtDM-specific knowledge.
- Potential synergy with SSG experiment-control user-interface software, which also is written in C++/Qt.

CSS BOY

- http://sourceforge.net/apps/trac/cs-studio/wiki/BOY

The Control System Studio - Best OPI Yet (CSS BOY) software is an interactive graphical display tool to construct EPICS graphical user interface displays within an interactive development environment (eclipse). The BOY component of CSS BOY has been developed and is currently managed by Kay Kasemir and others at the SNS. The CSS basis was written by Matthias Clausen and others at DESY. It is written using Java and SWT. (Note that on Linux, eclipse relies on GTK rather than Motif for its widget set.)

advantages and disadvantages

1. - different design approach to GUI controls
    1. - Considering the 2-ID-D use case above
    2. +/- compels a redesign of how we build such complex GUIs
2. - APS has no dedicated Eclipse / RCP developers
    1. - Maintenance and extension requires learning curve
    2. - Hiring Eclipse/RCP programmers may be difficult
    3. + APS has a developer now who has contributed to the CSS BOY code base on behalf of the APS
        1. MEDM to BOY screen translator
        2. byte display widget (shows bit values within a byte)
    4. + lessened risk as long as SNS provides support
3. + has screen editor built in
4. + can be extended
5. + provides access to a rich set of other tools (http://cs-studio.sourceforge.net/docbook/ch02.html) not part of MEDM
    1. Archiver
    2. logging (olog) package
    3. general charting tool
    4. alarm handler
    5. EPICS PV tree
    6. support for user preferences
6. + contemporary user interface, includes tabbed displays
7. + code is maintained by one or two programmers at ORNL
8. + runs on Mac OSX, Windows, and Linux
9. + restores previous session
10. + can save window arrangement as a perspective

11. + can run arbitrary scripts/commands
12. + very good integration with python
13. + support for EPICSv4 (PV Access) protocol
14. + can provide screens through a (webopi) WWW interface (http://sourceforge.net/apps/trac/cs-studio/wiki/webopi)

the work required to make this a viable product

CSS BOY is available as a product ready to be used in the EPICS community. A manual for CSS (http://cs-studio.sourceforge.net/docbook/) is available online which includes content specific for CSS BOY (http://cs-studio.sourceforge.net/docbook/ch15.html) . Deployment of CSS BOY at APS, on the accelerator, control room, and beamlines will take more examination. (A preliminary document describing the installation and configuration of CSS BOY at the APS has been prepared (http://subversion.xray.aps.anl.gov/admin_bcdaext/cssboy_deployment/) .) For example, supporting the use case of 2-ID where a session is started with 80 screens across 6 Linux workspaces is not clear how to implement in CSS BOY at this time. It may already be possible to do this with existing support but this investigation requires some effort.

Deployment of CSS BOY is initially quite easy but to fit it into existing workflows at APS, in the main control room and at the beam lines, is difficult to determine without experiences from more installations. CSS BOY provides additional toolkit components over MEDM, such as tabbed displays. These additional toolkit components, and the way in which CSS BOY provides screens, may compel some of our control screen suites to be redesigned. The work for this cannot be estimated in a general sense. It must be evaluated for each separate workgroup.

We have tested minimally that CSS BOY can be installed on a shared server (such as /APSshare) and then run by users from the shared server.

Training of APS staff and users will be required before CSS BOY becomes a viable product at APS. The way of using this software is different from the other candidates on this page. The assessment of difficulty with learning how to deploy and use CSS BOY will vary across the community.

In 2012, as preparation for larger testing and deployment of CSS BOY, all extant MEDM screens from both synApps (beam line control) and the *adlsys* (accelerator information) were converted into .opi files for use at beam lines. The MEDM screens were autoconverted using the adl2boy translator and then inspected. Some screens were found to require individual further adjustment to render properly as certain design elements in MEDM were difficult to translate automatically. Further testing of these screens will require a larger deployment. Additionally, since the translation last year, some MEDM screens have been revised. Those changes, which can be enumerated through the version control system, must be passed into the .opi files for CSS BOY.

Probably the single largest user interface challenge to use CSS BOY as the successor to MEDM is the eclipse workspace (http://eclipse.dzone.com/articles/eclipse-workspace-tips) . In the eclipse model underpinning CSS BOY, the *workspace* is defined once *for each account*. This presents a challenge to *run instances of the same GUI from concurrent sessions on the same workstation and account* as stated above in the feature wish list. To use CSS BOY at APS beam lines, it is necessary to establish a clear method to run instances of the same GUI from concurrent sessions on the same workstation and account. The eclipse community is very large and several solutions or work-arounds to the multiple workspace question have been discussed in various web forums. We must do the online research, test for fit in our community, and then document and implement a solution that is acceptable.

BCDA has begun assembling instructions for deploying CSS BOY at APS beam lines. This work in progress is available online: http://subversion.xray.aps.anl.gov/admin_bcdaext/cssboy_deployment/

Additional work will be necessary to address certain widgets (or their translation from MEDM screen files) such as slider, scale (a.k.a progress or thermometer), and charts.

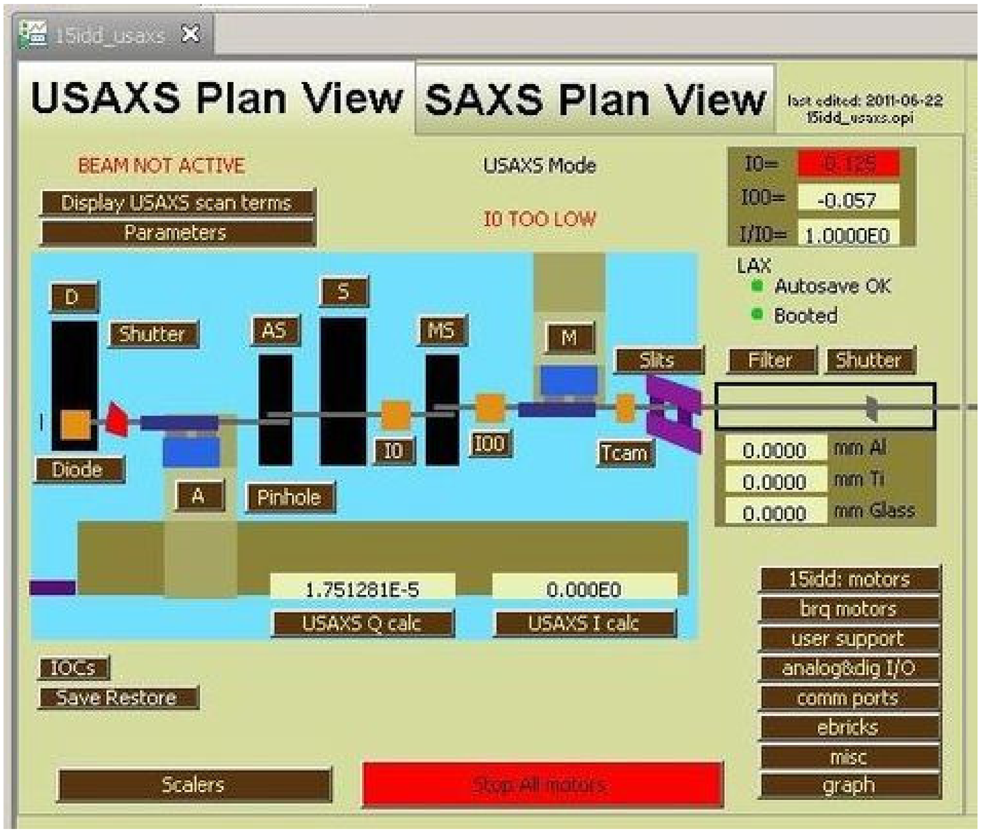
how this would fit into our architecture

Built on the Eclipse open-source platform from the business community (which uses Java as its language), several facilities in the worldwide EPICS community are investing in the support of CSS BOY. The APS added software to preserve our 15+ years investment in user-interface screens. In return for our investment in EPICS, we receive modern, community-supported software that was engineered for the right level of user.

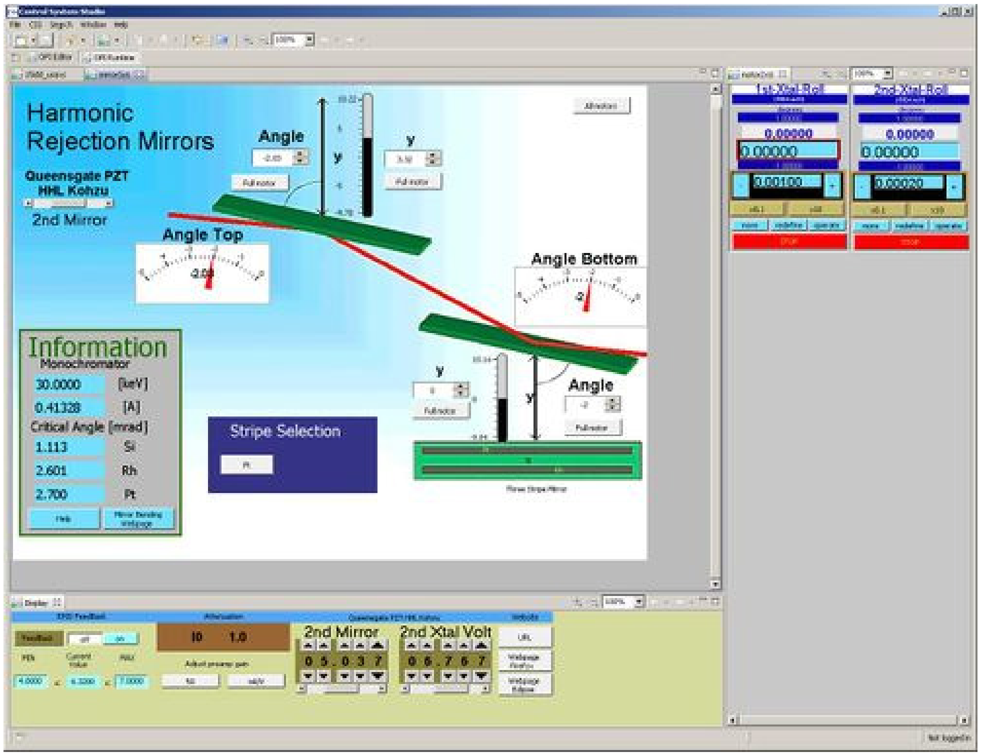
Note: ITER, for example, has been contributing widgets. The APS has contributed to CSS BOY in features, widget actions, and a converter for .adl files.

screen examples

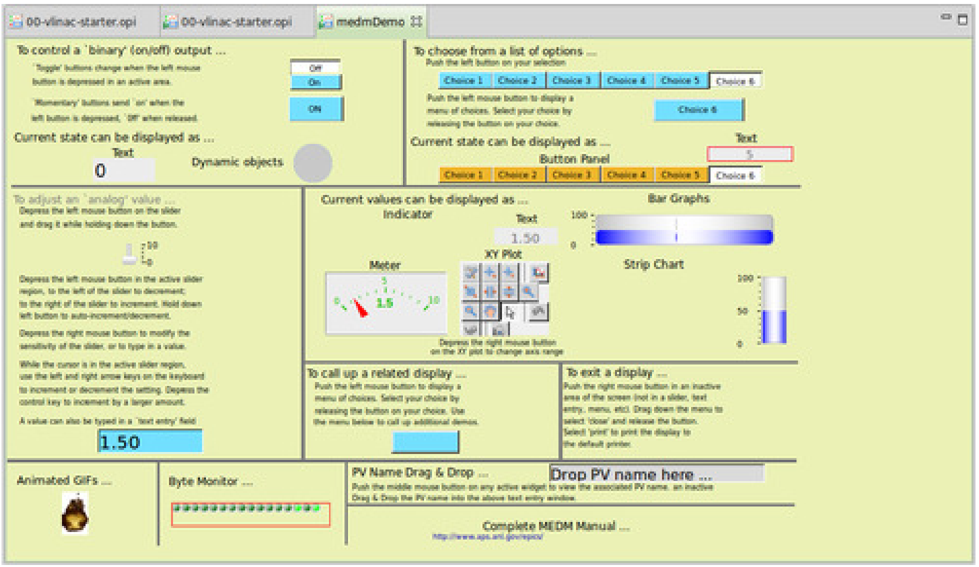
- CSS BOY example screen, converted from MEDM screen



- screen created by users in CSS BOY with added .jpg background file, also shows standard motor widget screens



- medmDemo screen from the Virtual Linac project as shown in CSS BOY, converted from MEDM. (No additional adjustments have been made to this .opi file after automatic translation by the adl2boy converter. Not all MEDM widgets have been represented properly in CSS BOY.)


To control a `binary' (on/off) output ...

`Toggle' buttons change when the left mouse button is depressed in an active area.

`Momentary' buttons send `on' when the left button is depressed, `Off' when released.

Current state can be displayed as ...

To choose from a list of options ...

Push the left button on your selection

Push the left mouse button to display a menu of choices. Select your choice by releasing the button on your choice.

Current state can be displayed as ...

To adjust an `analog' value ...

Depress the left mouse button on the slider and drag it while holding down the button.

Depress the left mouse button in the active slider region, to the left of the slider to decrement; to the right of the slider to increment. Hold down left button to auto-increment/decrement.

Depress the right mouse button to modify the sensitivity of the slider, or to type in a value.

While the cursor is in the active slider region, use the left and right arrow keys on the keyboard to increment or decrement the setting. Depress the control key to increment by a larger amount.

A value can also be typed in a `text entry' field

Current values can be displayed as ...

Depress the right mouse button on the XY plot to change axis range

To call up a related display ...

Push the left mouse button to display a menu of choices. Select your choice by releasing the button on your choice. Use the menu below to call up additional demos.

To exit a display ...

Push the right mouse button in an inactive area of the screen (not in a slider, text entry, menu, etc). Drag down the menu to select `close' and release the button. Select `print' to print the display to the default printer.

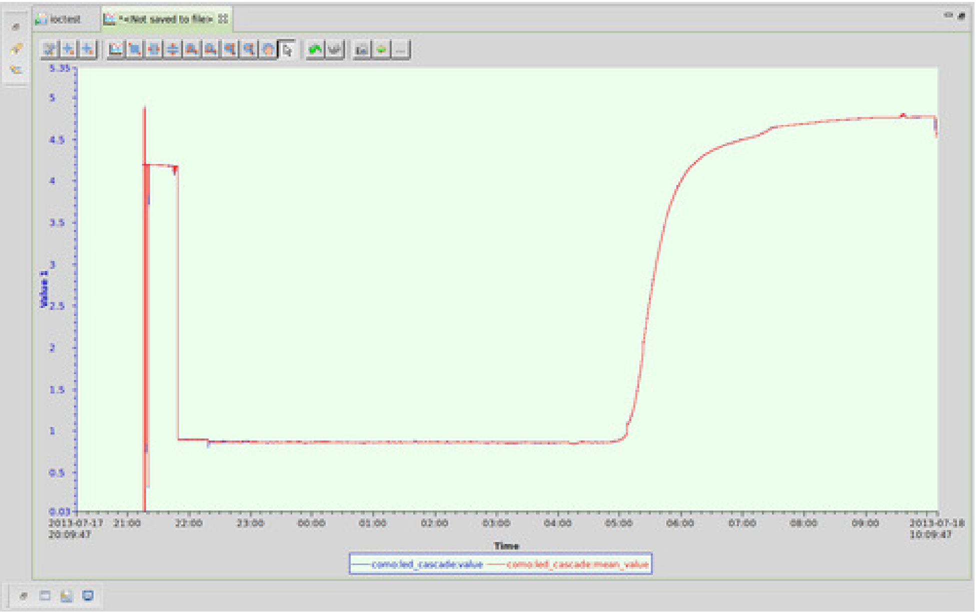
PV Name Drag & Drop ...

Push the middle mouse button on any active widget to view the associated PV name. an inactive Drag & Drop the PV name into the above text entry window.

Complete MEDM Manual ...

http://www.aps.anl.gov/epics/

- CSS BOY example line chart of analog voltage signal from a photocell



epicsQt

- http://sourceforge.net/projects/epicsqt/

The epicsQt software is a widget based toolkit to construct EPICS graphical user interface displays. It was created to replace in-house screens and also to replace EDM. It was created and is currently managed by Andrew Rhyder and others at the Australian Synchrotron. It is written using C++ and Qt.

advantages and disadvantages

1. - It appears that this tool is not widely deployed
    1. - either at AS or as well as at other facilities.
    2. no coherent collaborative development effort
2. + has an editor
3. - reuse of epicsQt widgets in python conflicts with our existing use of PyEpics

the work required to make this a viable product

1. Provide test candidates for MCR and beam lines
    1. translate screens from MEDM
    2. revise screen elements that defied an automated translation
2. Provide instructions for using
3. Only with feedback from a testing phase can the scope of additional work be determined to make this tool a viable product at the APS
4. training
5. Work should be done so that the widgets from other Qt-based EPICS tools can be used by this tool. APS investment here might be able to ensure this synergy occurs.

how this would fit into our architecture

- screens are created in an IDE and stored as text files (as .ui files)
- Most screens do not require programming, they can be built with the Qt Designer tool.

- new widgets types require programming knowledge

---

table of comparisons (feature sheet)

Comparison of GUIs for EPICS at APS

User Criterion	MEDM	EDM	CSS BOY	caQtDM	epicsQt
Has MEDM feature set	🟩	🟩	🟩	🟩	🟥
Can convert .adl	🟩	🟩	🟩	🟩	🟥
Has adequate performance	🟩	🟩	🟩	🟩	🟩
Runs on modern platforms	LW!	LW!	LMW	LW	LW
Is robust	🟩	🟩	🟩	?	?
Strip charts	🟩	🟥	🟩	🟩	🟩
2D scan plots	🟥	🟥	🟥	🟩	🟥
2D areaDetector images	🟥	🟥	🟩	🟩	?
Camera images	🟥	🟥	🟩	🟩	🟩
save display config	🟥	?	🟩		
restore display config	🟩	?	🟩	🟩	

- L: Linux
- M: Macintosh OSX
- W: Microsoft Windows
- !: requires Motif/X11 support

Comparison of GUIs for EPICS at APS

Developer Criterion	MEDM	EDM	CSS BOY	caQtDM	epicsQt
Collaborative development	🟥	🟩	🟩	🟩	🟩
Builds on modern platforms	LW!	LW!	LMW	LW	LW
Composable display	🟩	?	🟩	🟩	?
EPICS long strings	🟩	?	🟩	🟩	?

- L: Linux
- M: Macintosh OSX
- W: Microsoft Windows
- !: requires Motif/X11 support

Comparison of GUIs for EPICS at APS

Wish list	MEDM	EDM	CSS BOY	caQtDM	epicsQt
Easy screen creation	🟩	🟩	🟩	🟩	🟩
Contemporary feel	🟥		🟩	🟩	🟩

Tabbed display	🟥	🟥	🟩	🟩	🟩
warn multiple PV	🟩			🟥	
EPICS V4 support	🟥	🟥	🟩	🟩	
Archiver integration	🟥	🟥	🟩	🟥	🟥
Alarm display	🟩		🟩	🟩	
Access security display	🟩		🟩	🟩	
Scripting	🟥		🟩	🟨 1	🟨 1
Remote access	🟩				
Smartphone, etc.	🟥	🟥	🟩	🟥	🟥
text display file	🟩	🟩	🟩	🟩	🟩
titles	🟥	?		🟥	?
widget synergy	🟥	🟥		🟩	🟩
learning curve	🟥	🟥	🟥	🟩	🟩

- 1: Supported by Qt, but not demonstrated in EPICS widgets

Options not considered

- Blu-Ice and the Distributed Control System (http://smb.slac.stanford.edu/research/developments/blu-ice/) were developed to provide unified control over the disparate hardware resources available at a macromolecular crystallography beam line. The role of Blu-Ice is to provide scientific users and beam line support staff with intuitive graphical tools for collecting diffraction data and configuring beam lines for experiments. According to this source (http://smb.slac.stanford.edu/research/staff_publications/blue%20ice%20and%20distributed%20control.pdf) , Blu-Ice is implemented at four SSRL beam lines. An implementation of the Blu-Ice concept has been implemented by LS-CAT (sector 21) for use by their beamlines at the APS. Blu-Ice was not examined for this report.

- GDA (http://www.opengda.org/) , the Generic Data Acquisition from Diamond Light Source, is an open-source framework for creating customized data acquisition software for beam lines at science facilities such as neutron and X-ray sources. The software is built on Eclipse/RCP (Java), free and released under the GPLv3. AES-SSG (John Hammonds) investigated GDA in 2010 as the possible replacement to MEDM, working with staff from DLS. APS invested about 4-6 FTE months. We demonstrated area detector data coming into GDA. Summary is that separation of Diamond's configuration of GDA for a deployment at APS was *going to be difficult to manage given available resources.* One of the difficulties was the expectation that GDA would be deployed on a system where both server and client would run the same operating system. Unnecessary complexity in the server had grown due to historical reasons in the project. GDA was also investigated by two other APS sectors: 16 and 18. Neither are using it at this time.

- GumTree (http://gumtree.codehaus.org) is an open source scientific workbench for performing scientific experiments under a distributed network environment, developed at ANSTO by Tony Lam. The software is built on Eclipse/RCP (Java). It was not examined for this report.

- IDL (http://www.exelisvis.com/ProductsServices/IDL.aspx) (Interactive Data Language) is a programming language and GUI toolkit used for scientific data analysis with a strength in image visualization and analysis. It was not examined for this report.

- LabView (http://www.ni.com/labview/) is a commercial measurement and control software package. An EPICS binding for LabView was developed and is maintained at APS sector 20 and is used now for user experiment controls. It was not examined for this report.

- MatLab (http://www.mathworks.com/products/matlab/) is a commercial high-level language and interactive environment for numerical computation, visualization, and programming. It was not examined for this report.

- Tcl/Tk (http://www.tcl.tk) (Tool Command Language / graphical user interface ToolKit) is a very powerful but easy to learn dynamic programming language and GUI toolkit. An EPICS Channel Access binding (*et_wish*) was created by Bob Daly and is most recently maintained by the APS AOP Group in the ASD division as *oagtclsh*. Its 2-D graphing capability has been provided by the BLT (http://wiki.tcl.tk/199) package, the latest version of which was released in 2002. Performance of the BLT charting widget for high EPICS PV update rates has been seen as problematic in some charting tools at the APS. Despite the availability of *et_wish* since the early stages of APS beam line development, it was only used at a small number of beam lines. It was not examined for this report.

Performance comparisons

Performance tests were performed with Red Hat Enterprise Linux version 6 (RHEL6) and also Windows 7. The published results are for RHEL6. The workstation was standard hardware issue at the time of this report:

```
HP Compaq 8300 Elite Convertible Minitower
  Delivered with:
  Processor: Intel® 3.4GHz i7-3770 Quad Core 8 MB cache
  Chipset: Intel® Q77 Express Chipset
  Memory: 8GB (2x4GB) DDR3-1600 (32GB Max)
  Hard Drive: 1TB SATA Hard Drive (7,200rpm)
  Graphics: NVIDIA Quadro 600 Graphics with 1GB (WS093AT)
  DVD Writer SATA SuperMulti LightScribe
  Audio: Integrated Realtek® ALC221 High Defination Audio
  Network Connection: Integrated Intel(R)82579LM Gigabit
  (Specs from  http://www8.hp.com/us/en/products/desktops/product-detail.html?oid=5232845#!tab=features )
```

Performance of GUIs for EPICS at APS

program	**version**	**% CPU utilization**	**Max update rate**	**Loss-less Rate**
medm	3.1.7	12.83	65000	45000
EDM	1.12.40	10.95	35100	20000
caQtDM	2.8.0	13.22	8400	5000
CSS BOY	3.1.4	14.88	22000	15000
epicsQt	2.4.18	13.12	11100	5000

- Note that caQtDM throttles display updates at 5 Hz by default.

Definitions of Results:

Max update rate: This is the maximum update rate in Hz at which the display manager rendered changes to PVs. Beyond this, the performance decreases as the display manager consumes resources simply keeping up. These are for a for a numerical display only.

Loss-less Update: This the maximum rate (Hz) at which the display manager rendered changes to PVs without any data loss.

Test Method: The results was obtained by using a 3000 PV EPICS database. Each PV was updating at 0.1 Hz. Displays in blocks of 500 were created for each display manager. Each PV was provided a ramp function which cycles from 0-99 incrementing by 1 at 10 Hz. The count was then displayed. PVs were reused on displays when required. An independent video camera operating at 30 fps was used to determine the performance of the display manager. The number of displayed PVs were increased until failures were observed. The number of displayed PVs and the frequency of the updates was used, then, to calculate the max update rate.

```
#PVs * (%updates * 10)
```

To obtain the rate in Hz, the total number of characters on the screen at the failure point was multiplied by 10.

The results of any pilot implementations

This section reports feedback from various installations of the software tools discussed above. They subsections are sorted in alphabetical order by tool. At the time of writing, we have no substantive feedback from the epicsQt community.

Feedback from users and BCDA developers on caQtDM

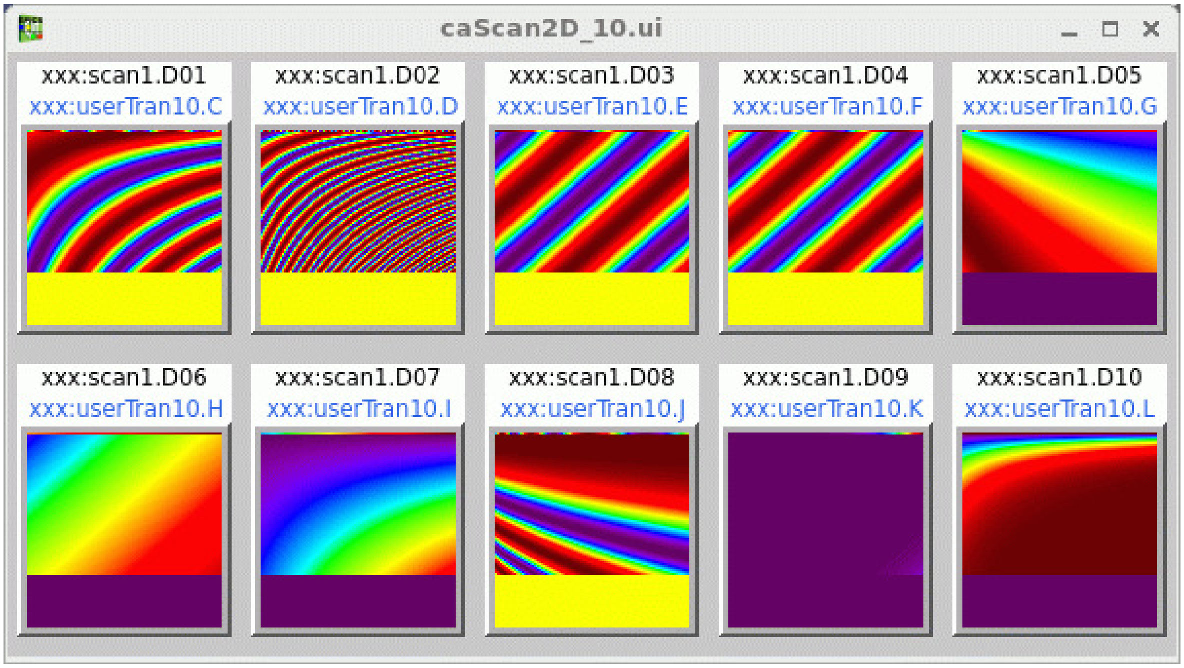
Chris Roehrig: *I played with it a bit and I think it is very nice. I do like the 2D data display, especially the way you can pick and choose individual detectors to appear in larger windows. I showed Stefan [Vogt] and he too thought it was good. One thing I did notice is that it is harder to see at a glance which choice button is selected, such as the Go/Pause buttons on scan records or the Enable/Disable buttons on the motor screen. I think that is a minor detail.*

Kurt Goetze: *I've been playing around with the live 2D and haven't been able to break it yet. It looks really good so far. This is the first live 2D data I've seen since scanSee. It would be great to not have to worry about keeping scanSee going anymore, and this looks to be a step in that direction.*

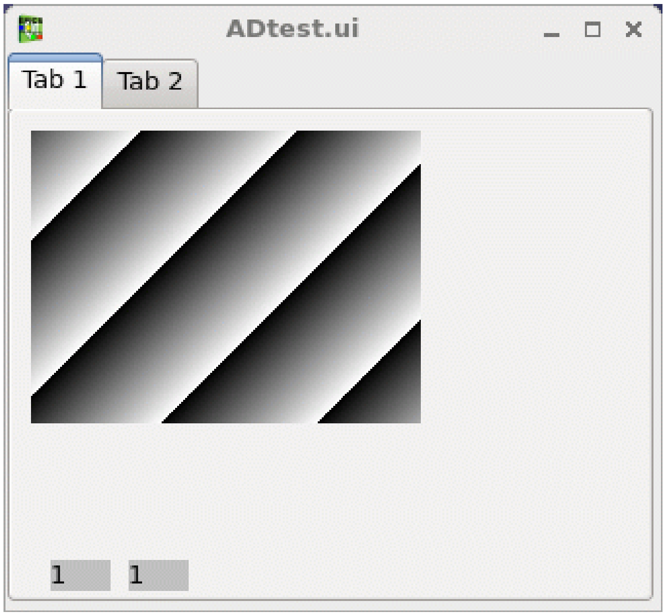
Tim Mooney: *Evaluated ability to place multiple windows at specified locations and workspaces, as is currently done at 2idd. caQtDM does this in the same way, and with the same syntax, as MEDM. For example:*

```
  caQtDM xxx.ui&
  sleep 1
  caQtDM -attach -macro "P=xxx:,S=scan1" -dg +500+300 scan.ui
  sleep 1
  caQtDM -attach -macro "P=xxx:,S=scan2" -dg +100+300 scan.ui
  ...
```

Tim Mooney: *I've never programmed in Qt before, but it was pretty easy to cobble together a 2D scan-data display widget, by converting the caCamera widget to catch live raster data from the ioc, and to back fill with stored data supplied by Dohn Arms' mda file reader (which is written in C).*



Tim Mooney: *caQtDM can display areaDetector images, using the caCamera widget, and local "soft" PVs, which can be programmed to map areaDetector image-description PVs (e.g., bytes per pixel) to caCamera image-description requirements.*



15ID-D USAXS uses CSS BOY

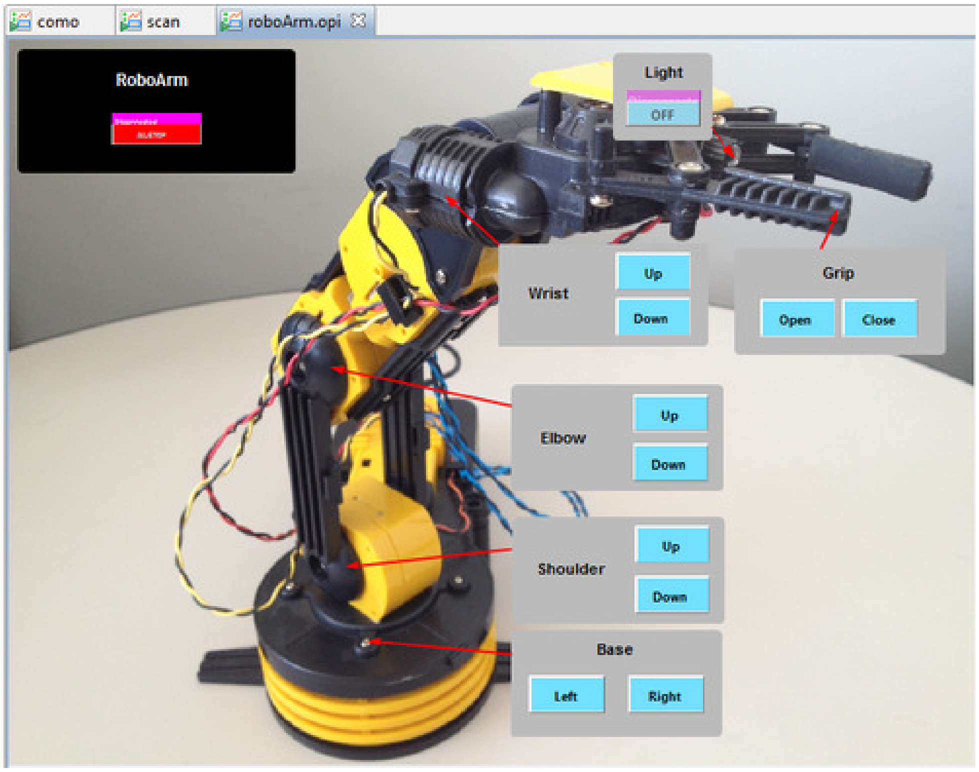
USAXS instrument (http://usaxs.xray.aps.anl.gov/) moved to its new location at 15ID beamline in summer 2010. Major revision of the GUI control system was needed as part of the move. Therefore, the USAXS instrument decided to port to CSS BOY while the rest of the beamline stayed with medm. USAXS hired two students (for about 6 months combined) in the following year and they, with less than 50% effort, converted most of the USAXS related medm screens into CSS BOY and created many new. Over time the APS support groups and the instrument staff built the system in which the instrument specific GUI, mostly CSS BOY, is seamlessly integrated with existing beamline medm screens. This integration enables the beamline staff to convert at appropriate pace into CSS BOY. Synchronization of the code across three architectures in use at the beamline - Linux, Windows 7, and Mac OSX - is provided by use of a subversion server. After updating all of the computers to sufficient CPU, graphics, and memory, the CSS BOY performance and stability is as high as medm or higher.

User and beamline staff experience with CSS BOY is very good. Based on staff feedback, the (at least perceived) flexibility and GUI capabilities of CSS BOY are higher than medm. It is easier to program for the beamline staff due to the more advanced programming environment provided by eclipse. It is relatively easy for beamline staff (or summer student) to design complex GUI screens and sometimes javascript which react to various beamline conditions and settings. The GUI currently also integrates video camera signal, web pages and graphics, and other remote content. Also important is, that staff can quickly put together simple, ad-hoc, screens needed for specific short user experiments with little effort. Current effort of the beamline is to integrate new generation python beamline operations with the CSS BOY GUI to enable seamless "push-button" user operations from a graphical interface.

In general, the conversion from medm to CSS BOY has been a success. While it was not always easy due to this specific beamline being the first at the APS to use CSS BOY, it resulted in an interface that is easier to use for users than what MEDM the interface ptovided or could ever be.

2012 ANL Energy Showcase Demo used CSS BOY

CSS BOY was used as the GUI at the 2012 ANL Energy Showcase to demonstrate an EPICS-based robotic arm to the general public. The control screen was prepared by an undergraduate intern and provides access to basic motions of the various motors and LED on the robot arm.



ANL NE project uses CSS BOY

The ANL Nuclear Energy division implemented a CSS BOY GUI for a reactor controls project recently with positive reception. Additional feedback from that project was not available in time for this report.

APS Control Room CSS BOY instance

The external left rear workstation, not part of the "operators ring" has CSS BOY installed on It. A number of MEDM adl files were converted.

There has been no impetus for the operator to use this machine.

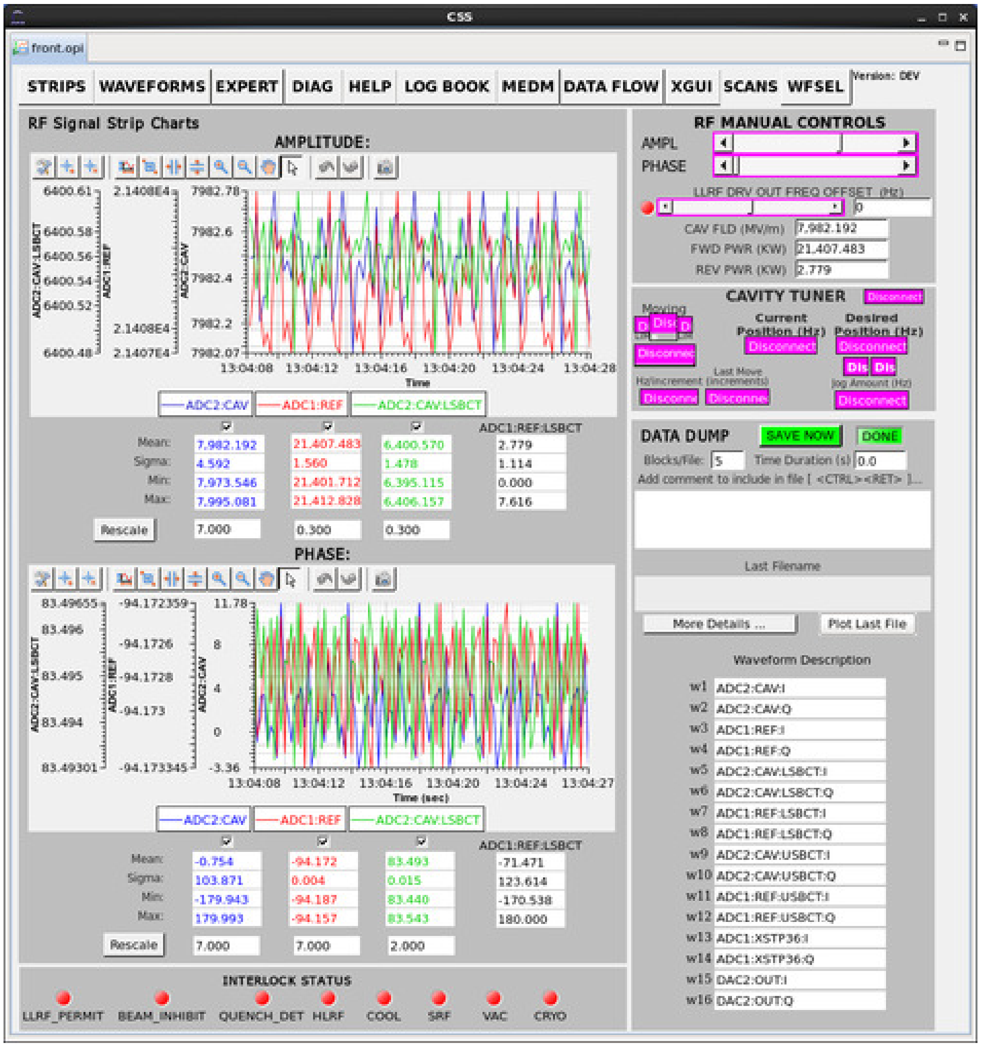
High Energy Physics Implementation use of CSS BOY

Tim Madden implemented CSS BOY for a high-energy physics group (Gammasphere) at ANL with positive reception. Additional feedback from that project was not available in time for this report.

SPX0 Controls use CSS BOY

CSS BOY is extensively used for SPX0 Controls. (*SPX0*: test installation for the Short Pulse X-ray project as part of the APS Upgrade) At the moment only cavity control (LLRF) OPIs have been implemented. Future work will involve addition of OPIs for timing controls, diagnostics, etc.

SPX0 Feedback: In general we find CSS BOY rather easy to use and extend. It comes with a rich set of monitoring and charting tools, and provides very good integration with python.



AMPLITUDE:

	☑ ADC2:CAV	☑ ADC1:REF	☑ ADC2:CAV:LSBCT	ADC1:REF:LSBCT
Mean:	7,982.192	21,407.483	6,400.570	2.779
Sigma:	4.592	1.560	1.478	1.114
Min:	7,973.546	21,401.712	6,395.115	0.000
Max:	7,995.081	21,412.828	6,406.157	7.616
Rescale	7.000	0.300	0.300	

PHASE:

	☑ ADC2:CAV	☑ ADC1:REF	☑ ADC2:CAV:LSBCT	ADC1:REF:LSBCT
Mean:	-0.754	-94.172	83.493	-71.471
Sigma:	103.871	0.004	0.015	123.614
Min:	-179.943	-94.187	83.440	-170.538
Max:	179.993	-94.157	83.543	180.000
Rescale	7.000	7.000	2.000	

LLRF DRV OUT FREQ OFFSET (Hz) 0

CAV FLD (MV/m) 7,982.192

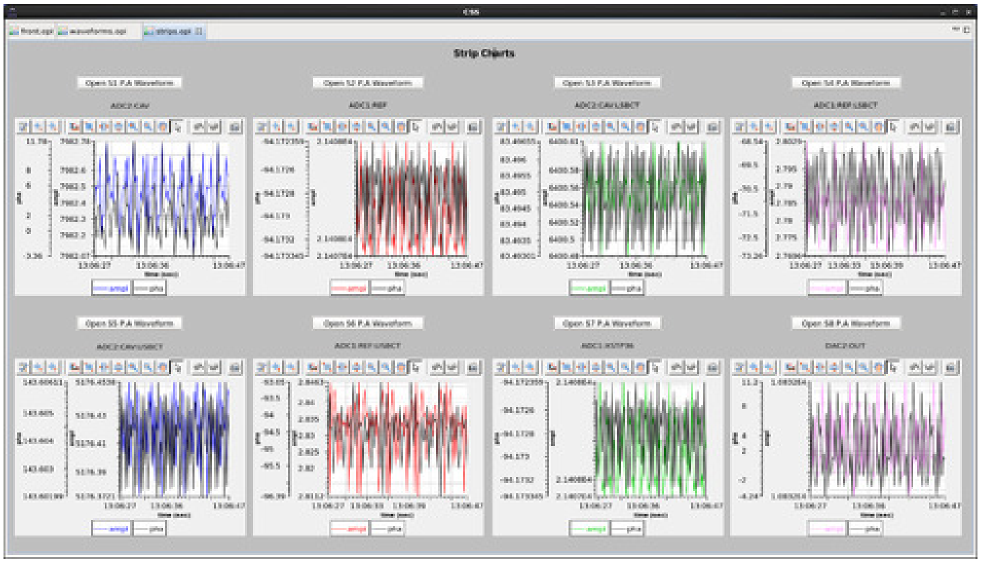
FWD PWR (KW) 21,407.483

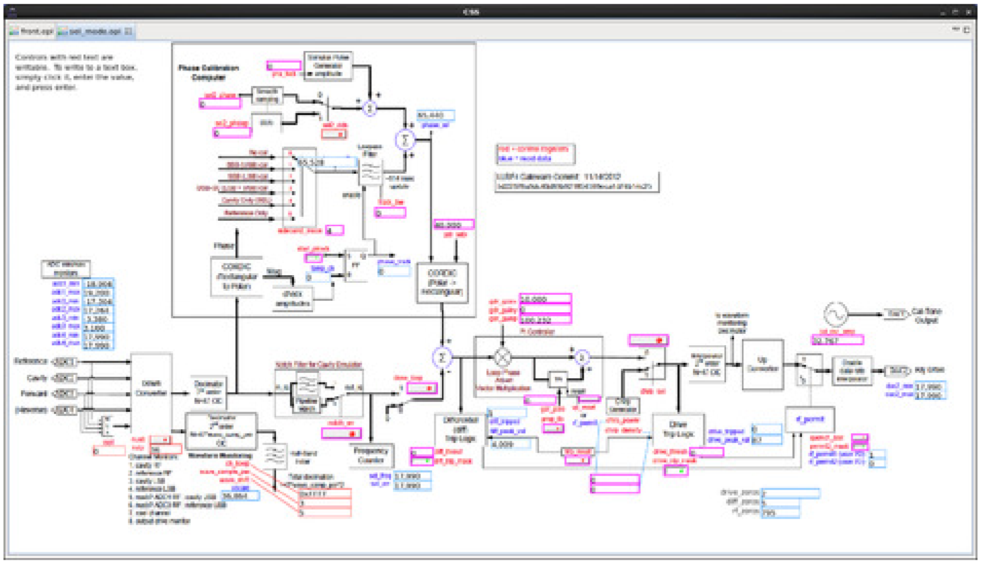
REV PWR (KW) 2.779

Blocks/File: 5 Time Duration (s) 0.0

Add comment to include in file [<CTRL><RET>]...

w1	ADC2:CAV:I
w2	ADC2:CAV:Q
w3	ADC1:REF:I
w4	ADC1:REF:Q
w5	ADC2:CAV:LSBCT:I
w6	ADC2:CAV:LSBCT:Q
w7	ADC1:REF:LSBCT:I
w8	ADC1:REF:LSBCT:Q
w9	ADC2:CAV:USBCT:I
w10	ADC2:CAV:USBCT:Q
w11	ADC1:REF:USBCT:I
w12	ADC1:REF:USBCT:Q
w13	ADC1:XSTP36:I
w14	ADC1:XSTP36:Q
w15	DAC2:OUT:I
w16	DAC2:OUT:Q




CSS
front.opi
cal_mode.opi
Phase Calibration Computer
CORDIC (Rectangular to Polar)
CORDIC (Polar -> rectangular)
Reference
Cavity
Forward
Down Converter
Notch Filter for Cavity Emulator
Frequency Counter
Up Converter

Controls with red text are writable. To write to a text box, simply click it, enter the value, and press enter.

SNS Control Room comments on CSS BOY

from *Charles Peters*:

I have not had a lot of experience building CSS BOY screens.

How we use BOY:

At SNS in the control room we mostly use BOY for comfort displays. There are a few screens that do control hardware, but in general it is used just to view. It does a great job with that. The drag and drop features to add plots or webpages or images is very easy. That interaction is great.

What I love about CSS BOY:

The BOY webopi app is great. I can create a webpage with all of the EPICS PVs that I want and be able to view from home on my computer or on my iPhone. I can watch the beam real-time from my iPhone and know what's wrong sometimes before the operators are able to update the e-log. For me it is the most useful thing about BOY so far.

CSS as an archiver is great, but of course I use it all of the time so I am quite familiar. I just haven't spent enough time with BOY.

Issues with BOY:

With BOY there are java scripts, which takes some time to learn the syntax. The great thing about EDM is someone has probably already done whatever you want to do so you can copy and build quickly. With BOY (at least here at SNS) there are things we want to do, but don't want to spend the time to relearn how to do something when we can do it faster with EDM. It also seems to respond quite slow. Much slower than EDM, but again this could be programming inexperience.

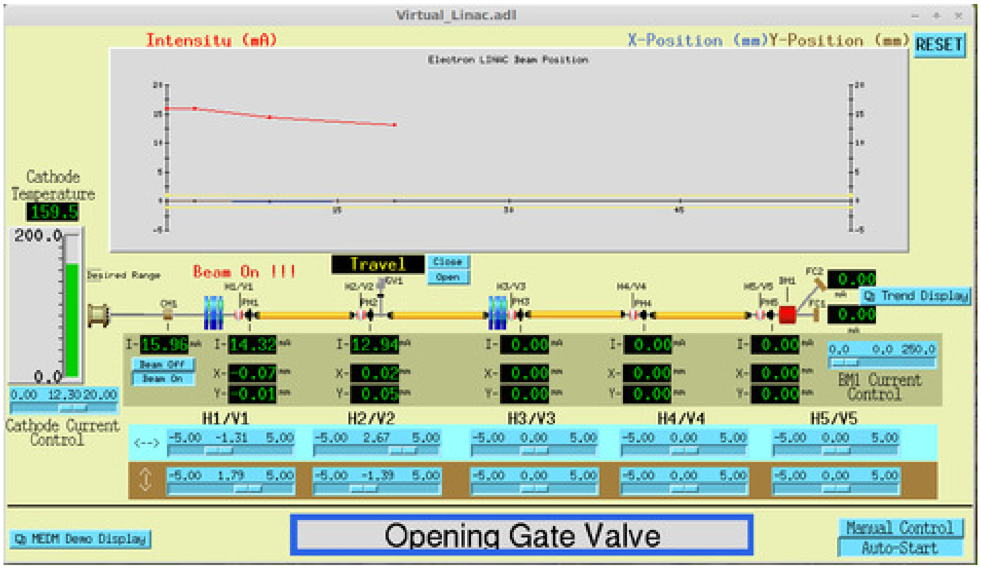
The other issue for me is that pages open in tabs. In the control room we have 6 monitors at each station. We open different EDM pages and spread them out all over. With CSS BOY everything is done with tabs. I like to monitor many different systems and want to see them all at once. I don't want to click through each tab, and miss something else on another page.

Virtual Linac

The Virtual Linac is software created by APS for use as a training platform for EPICS software developers and users. Prebuilt versions (http://www.aps.anl.gov/epics/download/examples/index.php) , ready for immediate execution, are released by the APS. Please note that the screen shots shown below for the various tools were recorded at different levels of progress in the auto-tune feature of the Virtual Linac software.

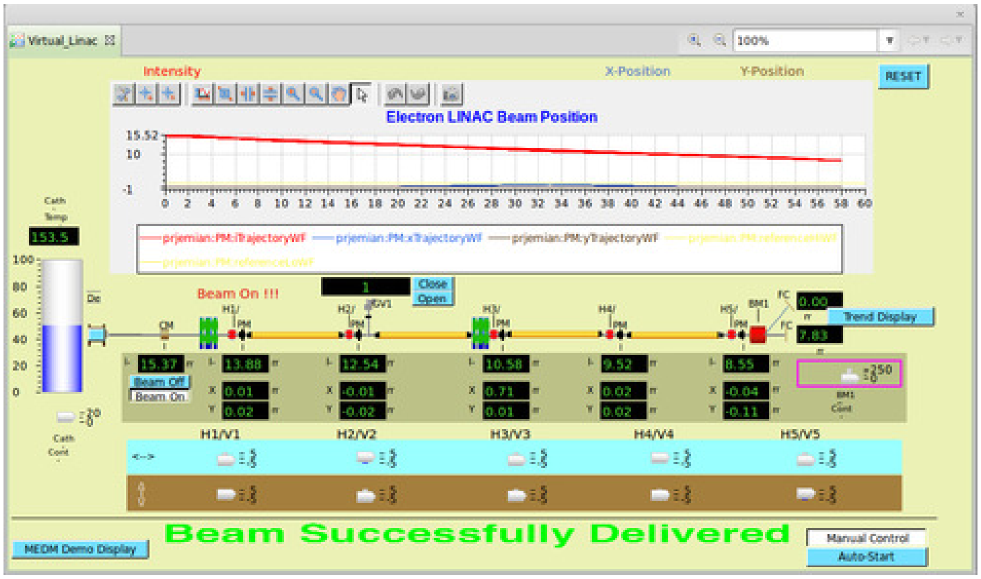
Virtual Linac with MEDM

The released versions come with MEDM installed and ready to execute. This screen shot from an MEDM session was captured as the auto-tune demo was progressing:



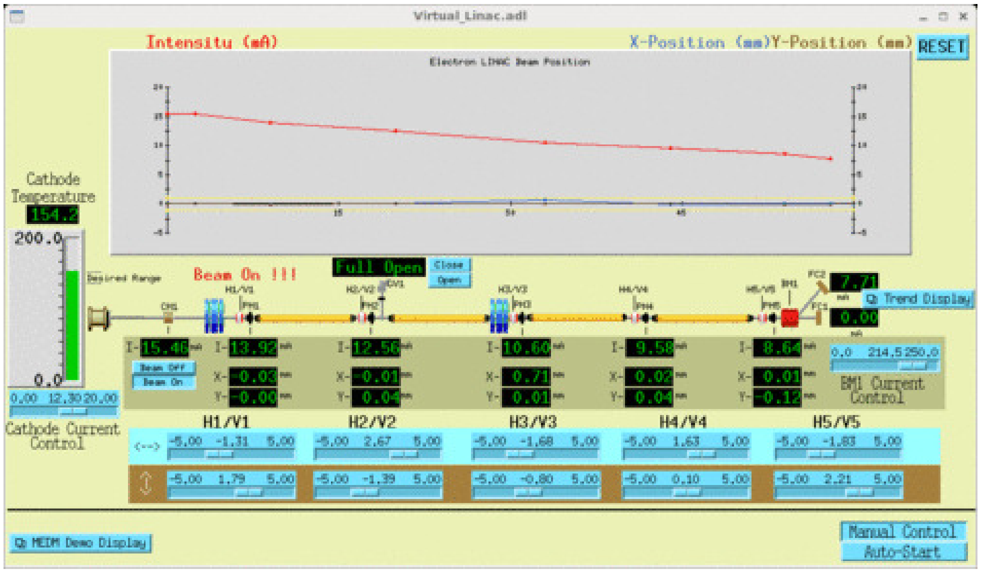
Virtual Linac with CSS BOY

The EPICSlive.zip package provides a full Ubuntu operating system, an EPICS database and fires up CSS BOY on suitable hardware (most modern PC's). Unzip and burn the images to a bootable DVD or extract to a datastick and then boot to the new medium to start the Virtual Linac and CSS BOY interface. This screen shot from a CSS BOY session (the screen definition file was autoconverted from the MEDM screen file using the adl2boy translator in CSS BOY) was captured as the auto-tune demo was complete:



Virtual Linac with caQtDM

This screen shot from a caQtDM session was captured as the auto-tune demo was complete. The screen was converted from the MEDM screen using the adl2ui translator included with caQtDM.



Devices at APS that use EDM

Oxford (for example) has provided instruments with EDM interfaces: 26-ID has a monchromator. FMB-Berlin has provided mirror systems for 29ID using EDM as the GUI.

Comments about EDM from one of the BCDA staff: *"EDM is much more difficult to compile than MEDM. The installation is full of configuration files with hard-coded paths. It is understood that this comes from the developers giving it more functionality, but it makes things more complicated.*

"From the little I did do with EDM, which was trying to fix a screen, it was not obvious how to edit anything, unlike MEDM which is fairly easy to start. Coming from MEDM, it's not obvious with EDM how things are supposed to work."

Comments about EDM from one of the beam line scientists: *"As far as I know, only the Oxford monochromators and cooling system (in-use cryo DCM and non-commissioned DMM) use EDM, as installed by Oxford. The DCM is addressed through MEDM interfaces from the beamline controls stations, so, for practical purposes, EDM is hidden."*

summary: There is very little experience using EDM at APS beam lines.

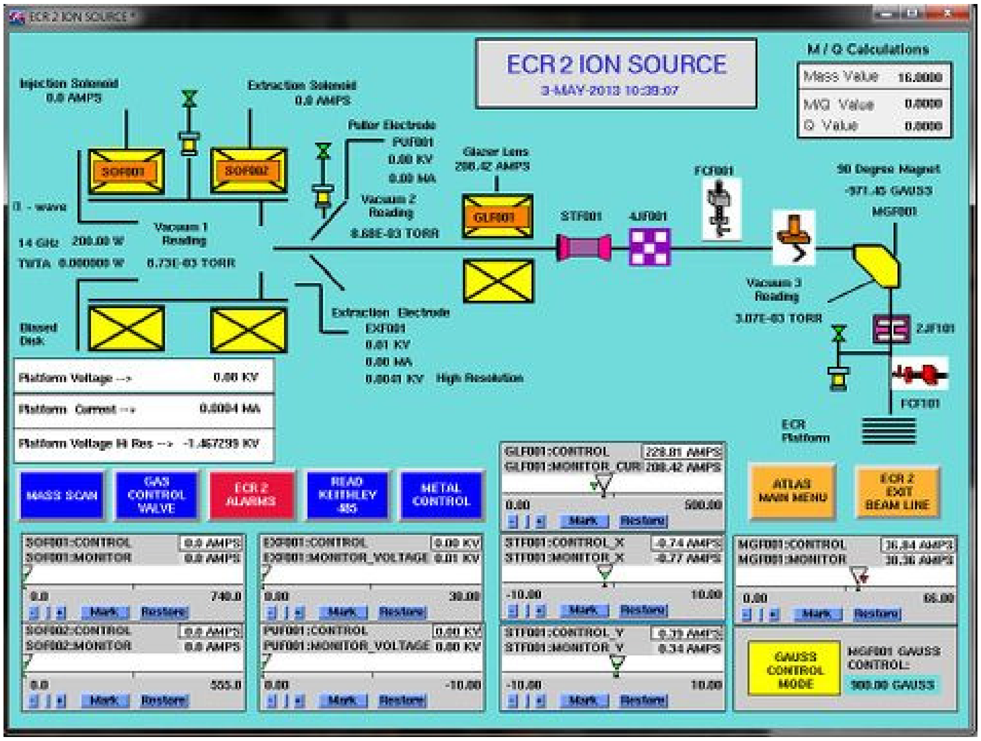
Comments about EDM from one of the beam line scientists: *"The EDM screens that we use have been provided by a manufacturer using the PV names and IP addresses that we provided. These were essentially part of a black-box controls system. When the manufacturers have configured everything properly the use of EDM has been seamless. However, if there is an error in PV names or IP addresses then it has been a bit of a struggle to get things up and running.*

"As far as use, the only real difference I've noticed are the obvious ones:

- *"you need a button to activate an operation in the GUI*
- *"you can continue to type into a field even if the mouse has moved out of said field."*

ATLAS - not EPICS

Example control screen from the commercial VDraw (http://www.vista-control.com/vsystem.htm) package (not EPICS) for an ECR Ion source at the ATLAS (http://www.phy.anl.gov/atlas/) (Argonne Tandem Linac Accelerator System). VDraw has a dependency on the Motif toolkit.



M / Q Calculations

Mass Value	16.0000
M/Q Value	0.0000
Q Value	0.0000

Platform Voltage --> 0.00 KV

Platform Current --> 0.0004 MA

Platform Voltage Hi Res --> -1.467299 KV

GLF001:CONTROL 228.81 AMPS
GLF001:MONITOR_CUR 208.42 AMPS
0.00 500.00

SOF001:CONTROL 0.0 AMPS
SOF001:MONITOR 0.0 AMPS
0.0 740.0

SOF002:CONTROL 0.0 AMPS
SOF002:MONITOR 0.0 AMPS
0.0 555.0

EXF001:CONTROL 0.00 KV
EXF001:MONITOR_VOLTAGE 0.01 KV
0.00 30.00

PUF001:CONTROL 0.00 KV
PUF001:MONITOR_VOLTAGE 0.00 KV
0.00 -10.00

STF001:CONTROL_X -0.74 AMPS
STF001:MONITOR_X -0.77 AMPS
-10.00 10.00

STF001:CONTROL_Y 0.39 AMPS
STF001:MONITOR_Y 0.34 AMPS
-10.00 10.00

MGF001:CONTROL 38.84 AMPS
MGF001:MONITOR 38.36 AMPS
0.00 66.00

MGF001 GAUSS CONTROL:
900.00 GAUSS

Recommendations, including plans for deployment

Overview

- MEDM is GUI software. The process to choose its must involve users.
- We should choose only one of these candidates as the successor to MEDM.
- No clear choice between these three candidates at this time. There are strong concerns to weigh between the various choices that optimize between the user experience and the ability of APS to provide substantial support as the needs of our facility evolve. We recommend a testing phase using consistent metrics and user feedback.
- Once the successor is chosen, ...
  - ... we will need a training program for the new tool across the APS
  - ... we will need a deployment schedule and assistance with the transition
  - It is recommended that we establish an end-of-support date for MEDM

Specific Recommendations and Observations

- It is still too early to decide between the two viable C++/Qt candidates (caQtDM and epicsQt) and CSS BOY as the common successor to MEDM.
- It is probably reasonable to consider epicsQt and caQtDM as a single entity, given that epicsQt widgets work in caQtDM.
- There are strong concerns to be weighed between the various choices that optimize between the user experience and the ability of APS to provide substantial support as the needs of our facility evolve.
- We must have useful feedback and strong support from users and management to make a firm decision.
- It is not likely to have a unanimous opinion about which tool should replace MEDM.
- We should choose only one of these candidates as the successor to MEDM.
- Prepare similar demo suites in caQtDM, CSS BOY, and epicsQt for user testing and evaluation
  - There is sufficient overlap in the requirements and wish list to choose only one
  - Economies of scale derive from choosing a single GUI
- We must realize that it is unlikely we will get (or even tolerate) the same product lifetime from the new GUI as we have from MEDM. This is due more to the rapid pace of facility development and computing environment than to the integrity of any decision at this time.
- We need user testing to better evaluate what additional components need to be developed to make any of these a viable product at the APS.
- Needs a reserved resource allocation, comparable to a project, to realize a complete transition from MEDM to its successor.
- We may need to redesign our more complex screen arrangements to suit the new tool (such as convert multiple workspaces into a tabbed display)
- We must become experts at troubleshooting and installation of the successor.
- Will need to become knowledgeable about how to get more support.
- We need a way to record issues management and bug reporting
- We recommend following this generalized deployment plan.

1. user comparison and evaluation
2. community feedback from the comparisons (setup a community wiki, page for each candidate)
3. task force charged with final selection for common facility support
4. implement required features
5. plan for implementation of wish list features

6. establish training schedule
7. establish deployment schedule
8. execute deployment schedule
9. establish an end-of-life plan for MEDM support at APS

- During the comparison phase, it is expected that some redesign of screens will occur. There is potential for inconsistencies in screen design to creep in which could spoil the effort to compare between the candidates. We recommend that MEDM be treated as the defining source during the comparison phase and that new versions of the screens be (re)created from the MEDM sources should such redesign occur.

Plans for the MCR to Compare the Candidate MEDM Replacements

1. setup a reserved space with test machines for each candidate (caQtDM, CSS BOY, and epicsQt) installed
2. provide a suite of the same test screens in all candidates.
3. automate the screen conversion process (may already be done for some candidates)

Plans for APS Beam Lines to Compare the Candidate MEDM Replacements

1. identify a small number (2-5) of beam lines for comparison tests, XSD can suggest
2. provide a suite of each beam line's MEDM screens in all candidates (caQtDM, CSS BOY, and epicsQt)
3. provide each candidate (if possible, ready for execution directly from /APSshare)
4. automate the screen conversion process (may already be done for some candidates)
5. include area detector or 2-D images where appropriate
6. include at least one complex MEDM setup (such as 2-ID-D, described above), there is other complexity
    1. attempt to represent Blu-ICE or equivalent (perhaps 11-BM?) at least once

What's Next?

- Testing is needed of three candidates in MCR and at a few beam lines
- Establish wiki to collect user feedback and provide documentation
- Evaluation Project Team is recommended
    - Choose metrics for evaluation
    - Establish test infrastructure
    - Evaluate, collate, and report tests
    - Recommend one of the candidates
- Train staff to use
- Deploy in MCR and beam lines
    - Deployment Project Team is needed to assist with transition at installation
- Management support is needed to ensure the success of this process

Retrieved from "https://wiki.aps.anl.gov/aescs/index.php/Upgrade_to_User_Interfaces"

- This page was last modified on 14 August 2013, at 13:58.