Identifying the Successor to MEDM and what to do next

APS ICMS: APS_1438839

Pete Jemian, Richard Farnsworth, Nicholas Schwarz and others
Group leaders – Computer Systems
APS Engineering Support Division

See also:
https://wiki.aps.anl.gov/aescs/index.php/Upgrade_to_User_Interfaces
APS ICMS: APS_1438838

The charge to the committee:

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

Why replace MEDM?

- MEDM: main user interface software for initial APS operations
  - Suitable for machine operations
  - Suitable for basic beam line operations
  - Satisfied basic requirements for access to control system features
- MEDM has become obsolete
  - Maintenance cost has increased as APS computing systems have evolved
  - Support for Motif is vanishing
  - Newer GUI toolkits (such as GTK and Qt) have expanded users' expectations
  - No support for images from cameras
  - Other tools in the EPICS community have been developed but none offered significant improvements over MEDM until recently
- No immediate *need* to replace but strong *desire* to use new tool ASAP
- Need to identify and change on our own schedule (pro-active *v*. reactive)

- MEDM has always fallen short of what APS users wanted in a GUI.

How is the user involved in the process to replace MEDM?

- MEDM is Graphical User Interface software
- The process to choose the successor to MEDM must involve users
- AES has identified three potential successors
- Users are needed to participate in the trial phase
- User feedback is essential
  - Needs a wiki to be setup
  - Needs test installations at
    - APS main control room
    - At least several beam lines
- Trial phase and deployment needs a project team:
  - team should include stakeholders
  - guide tests of successors
    - develop common metrics for evaluation
    - collate results and make recommendation
    - gather training and deployment techniques
  - guide deployments

http://www.loc.gov/exhibits/treasures/trm015.html

Requirements

- Widget and feature set at least inclusive of the MEDM widget set
- Must accept or convert existing .adl files
- Must have performance comparable to MEDM
- Must run on modern computing platforms
- Must be robust
- Must be able to show 2D visualization (includes images)
- support a configuration of displayed screens
- 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

Wish List

- consistent with contemporary user interfaces
- 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
- 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
- open-source and open to collaboration

Candidates to consider replacing MEDM

- Tool based on eclipse:
- **CSS BOY** : CSS from DESY and addition of BOY from SNS/ORNL
  - 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.
  - APS has one software developer who participated in the CSS BOY developer team and has made contributions to CSS BOY of value to the APS community.

- Tools based on C++ and the Qt toolkit:
- **epicsQt** : from Australian Synchrotron
- **caQtDM** : from Paul Scherrer Institute, PSI
  - These two C++/Qt candidates 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 toolkit, 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 definition files can be interchanged between these tools)? APS investment here might be able to pull this off.

Software not considered as replacements for MEDM

- GDA, the Generic Data Acquisition from Diamond Light Source, is an open-source framework, built on Eclipse/RCP (Java), for creating customized data acquisition software for beam lines. AES-SSG investigated GDA in 2010 as the possible replacement to MEDM, working with staff from DLS, investing about 4-6 FTE months. Conclusion was that … GDA … at APS was going to be difficult to manage given available resources. GDA was also investigated by two other APS sectors: 16 and 18. Neither are using it at this time.
- Blu-Ice and the Distributed Control System 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. It is implemented at four SSRL beam lines and at the APS. An implementation of the Blu-Ice concept has been implemented by LS-CAT (sector 21) for use by their beam lines at the APS.
- GumTree is an open source scientific workbench for performing scientific experiments under a distributed network environment, developed at ANSTO. The software is built on Eclipse/RCP (Java).
- IDL is a commercial programming language and GUI toolkit used for scientific data analysis .
- LabView is a commercial measurement and control software package.
- MatLab is a commercial high-level language and interactive environment for numerical computation, visualization, and programming
- Tcl/Tk (Tool Command Language / graphical user interface ToolKit) is a very powerful but easy to learn dynamic programming language and GUI toolkit.

Comparisons against the requirements

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 against the wish list

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

Comparison against developer criteria

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
- I: requires Motif/X11 support

Performance comparisons

test system: standard issue HP Compaq 8300 Elite Convertible Minitower running RHEL6

program	version	CPU utilization (%)	Max update rate (Hz)	Loss-less rate (Hz)
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

(*) caQtDM throttles display updates at 5 Hz by default.

Max update rate: This is the maximum rate (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 (without missing update events).

Test results obtained using a 3000 PV EPICS database. 500 PV display blocks were created for each display manager. Each PV was ramped 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.

Summary Results of Pilot Implementations

More details are in the report.

- **caQtDM**
  - Basic tests of caQtDM were performed by a few staff from AES and XSD
  - Response was quite good, MEDM displays rendered well
  - addition of 2-D scan and image display was a strong positive
- **CSS BOY**
  - 15-ID-D USAXS: used as instrument GUI since 2011
    "*resulted in an interface that is easier to use for users than what MEDM interface was or could ever be*"
  - 2012 ANL Energy Showcase, BCDA Demo – easy to use
  - APS Control Room – *a computer was setup … but no impetus to use*
  - SPX0 Controls - easy to use and extend
  - ANL NE project – positive reception
  - ANL High Energy Physics Gammasphere – positive reception
  - SNS Control Room - *mostly use for comfort displays, webopi is great, can do things faster in EDM* (the tool they use now)
- **epicsQt**
  - No pilot implementations

Recommendations and Deployment Plans: 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

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

Thank you for your attention

Identifying the Successor to MEDM and what to do next

APS ICMS: APS_1438839

Pete Jemian, Richard Farnsworth, Nicholas Schwarz and others
Group leaders – Computer Systems
APS Engineering Support Division

See also:
https://wiki.aps.anl.gov/aescs/index.php/Upgrade_to_User_Interfaces
APS ICMS: APS_1438838

The charge to the committee:

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

Why replace MEDM?

- MEDM: main user interface software for initial APS operations
  - Suitable for machine operations
  - Suitable for basic beam line operations
  - Satisfied basic requirements for access to control system features
- MEDM has become obsolete
  - Maintenance cost has increased as APS computing systems have evolved
  - Support for Motif is vanishing
  - Newer GUI toolkits (such as GTK and Qt) have expanded users' expectations
  - No support for images from cameras
  - Other tools in the EPICS community have been developed but none offered significant improvements over MEDM until recently
- No immediate *need* to replace but strong *desire* to use new tool ASAP
- Need to identify and change on our own schedule (pro-active *v*. reactive)

- MEDM has always fallen short of what APS users wanted in a GUI.

How is the user involved in the process to replace MEDM?

- MEDM is Graphical User Interface software
- The process to choose the successor to MEDM must involve users
- AES has identified three potential successors
- Users are needed to participate in the trial phase
- User feedback is essential
  - Needs a wiki to be setup
  - Needs test installations at
    - APS main control room
    - At least several beam lines
- Trial phase and deployment needs a project team:
  - team should include stakeholders
  - guide tests of successors
    - develop common metrics for evaluation
    - collate results and make recommendation
    - gather training and deployment techniques
  - guide deployments

http://www.loc.gov/exhibits/treasures/trm015.html

Requirements for the next graphical user interface

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 interface

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		*	*	*

Use case: MEDM in the APS Main Control Room

- Relies on the host machine desktop tabs being named properly;
- Setting up of the MCR operator displays is done through the OAGapps program
- 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.
- Various sdds files containing the MEDM commands to launch in the different workspaces on the host machine various virtual desktops.
- 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 MEDM displays are located within subdirectories of the directory

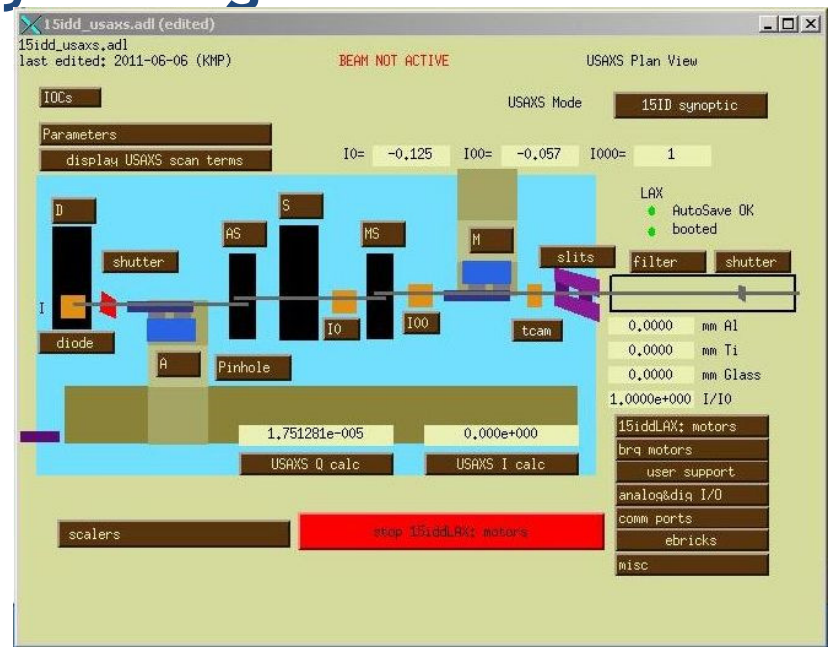
Options considered

program	UNIX	Linux	Windows	Mac OSX	files	ADL converter?	built using	maintained by
[medm]	yes	yes	yes(*)	yes(*)	.adl	not needed	C++ and X11/Motif	APS
[edm]	yes	yes			.edl	[yes]	Motif	SNS
[CSS/BOY]		yes	yes	yes	.opi	integrated	eclipse RCP	SNS
[caQtDM]		yes	yes		.ui	included	C++ and Qt	PSI
[epicsQt]		yes			.ui	not available	C++ and Qt	AuS

- Motif and X11
- C++ and Qt
- Eclipse / RCP
- web applications
- We have not considered web applications as part of this analysis.

The various tools in the EPICS community using Qt appear not to have coordinated in their development of EPICS-aware widgets.

MEDM: Motif Editor and Display Manager



example screen from 15ID USAXS

- Interactive graphical display tool for EPICS.
- Created at the APS
- Combined functions of EDD and DM *and* builds with the Motif widget toolkit
- Written using C++ and Motif and requires X11

- Motif has modern substitutes:
  - lessTif or OpenMotif
  - Neither are installed by default in the Linux used at APS (RHEL6)
- Requires a scientific graphics package (not included), either:
  - XRT Graph (licensed) – better graphics, $4k/y license
  - SciPlot (open source) – no license cost
- Currently managed by the APS Software Services Group.

Why does the APS need to move away from MEDM?

- 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
- 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
- Need to make this change on our own schedule (pro-active v. reactive)
- MEDM is out of date

MEDM: advantages and disadvantages

- + APS uses this now
- + robust and stable
- + used worldwide for many years
- - interface is dated
- - continued support costs are rising
- - MEDM falls short of what APS users wanted in a standard GUI environment
- - not easily extensible
- There is a significant advantage to having the developer “In House”
  - + APS had such a developer for many years
  - - There is no such assigned developer today
- - underlying technologies are becoming harder to install and deploy
  - Motif, as noted above
  - XRT/Graph is a licensed product ($4k/y at APS), end-of-life is anticipated within a decade

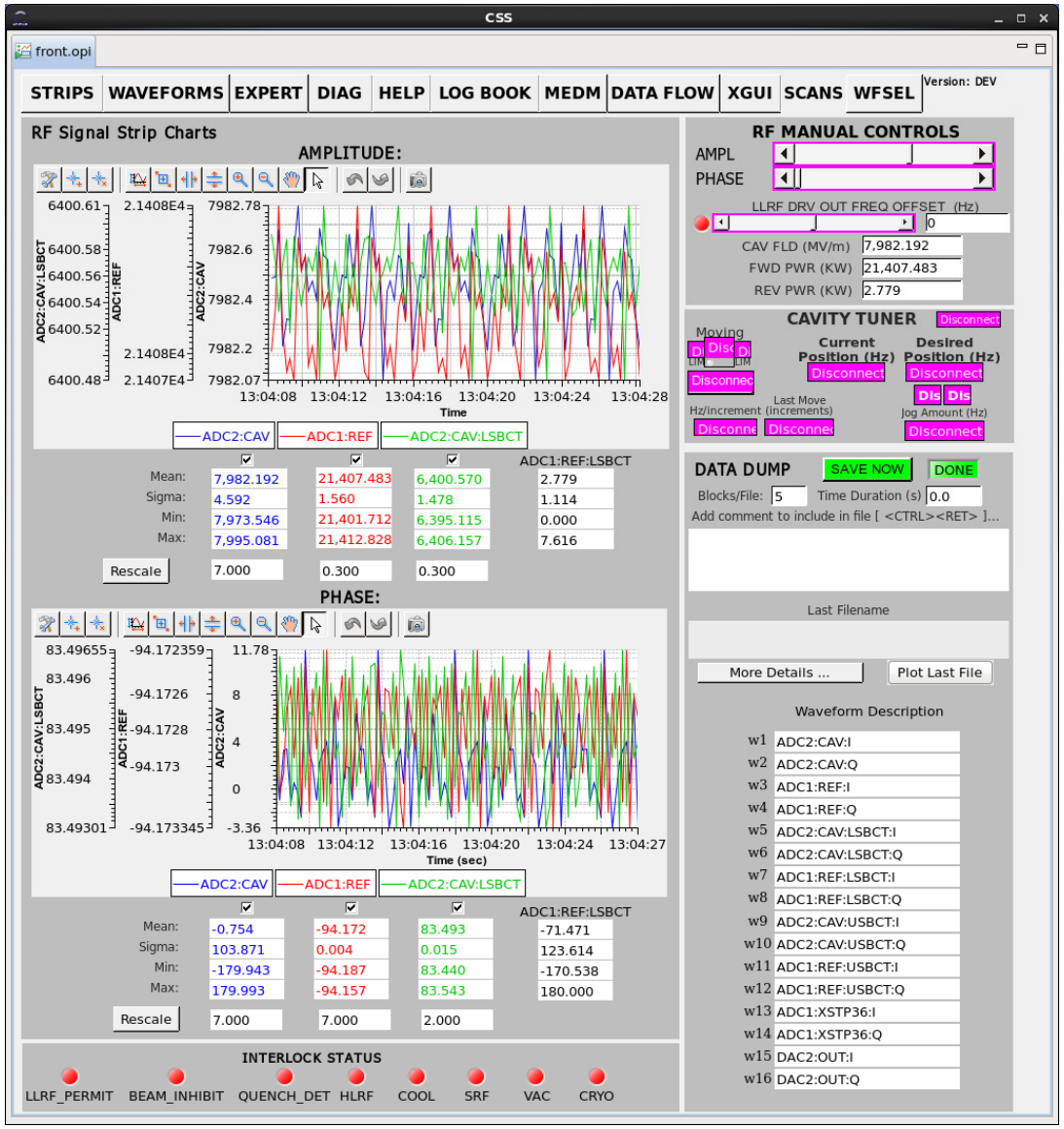
EDM: Extensible Display Manager

- Construct EPICS graphical user interface displays.
- Created as an alternative to MEDM. It is written using C++ and Motif. (Motif is required to build EDM.) It must run on an X11 server.
- **advantages and disadvantages**
  - - It has the same Motif and X11 basis as MEDM. It has the same end-of-life issues.
  - - EDM's .edl files are not compatible with MEDM's .adl files
  - + A tool exists to convert .adl files to .edl
  - - No tool exists to convert .edl to .adl because EDM has more widget types, some of which do not convert to MEDM widgets
  - - code is dependent on a *single* programmer (at ORNL)
- **the work required to make this a viable product**
  - in use now in some places at the APS as it came from vendor hardware
  - distributed as a tar.gz file
- **how this would fit into our architecture**
  - It appears that EDM is comparable in its feature set to MEDM
  - The only significant difference between EDM and MEDM is the look and feel.
  - *EDM does not offer a realistic upgrade in capabilities for the APS*

CSS BOY

- Interactive 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.
- written using Java and SWT. (Note that on Linux, eclipse relies on GTK)

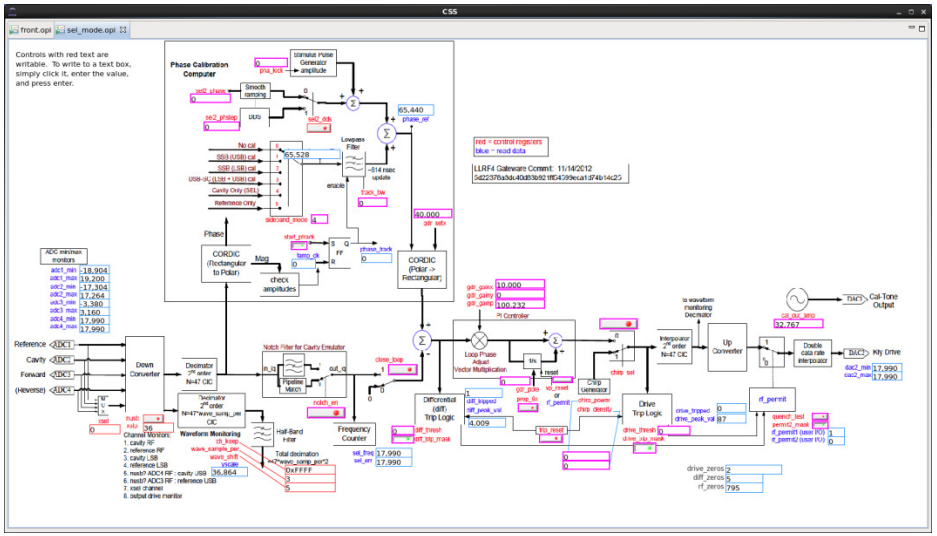
example screen from SPX0 project



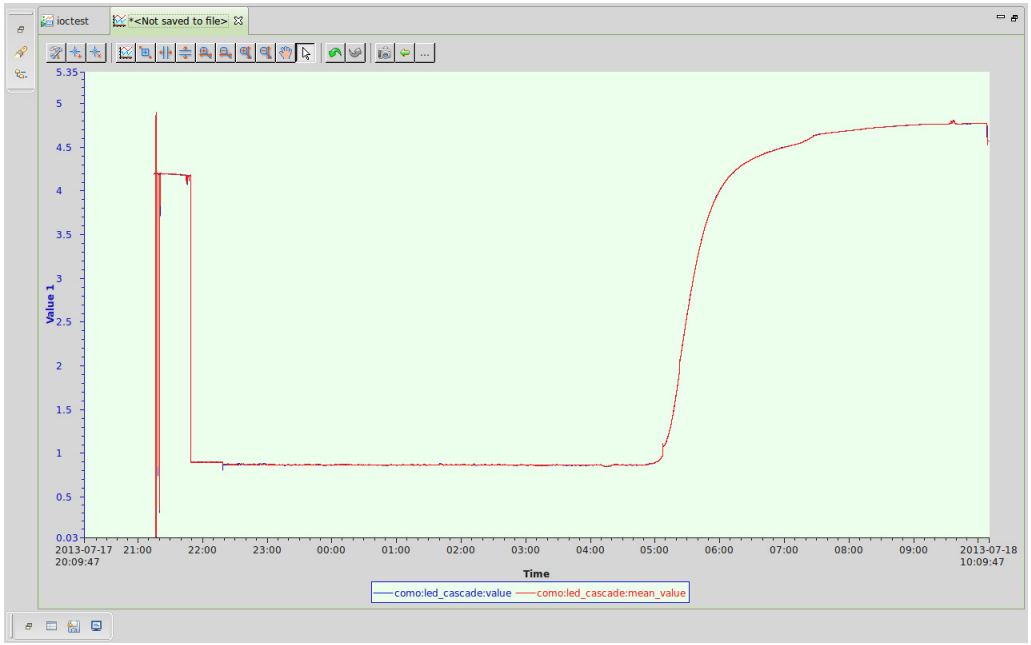
CSS BOY: advantages and disadvantages

- - different design approach to GUI controls
  - - Considering the 2-ID-D use case above
  - +/- compels a redesign of how we build such complex GUIs
- - APS has no dedicated Eclipse / RCP developers
  - - Maintenance and extension requires learning curve
  - - Hiring Eclipse/RCP programmers may be difficult
  - + John Hammonds has contributed to the CSS BOY code base
    - MEDM to BOY screen translator
    - byte display widget (shows bit values within a byte)
  - + lessened risk as long as SNS provides support
- + has screen editor built in

example screen from SPX0 project



CSS BOY (continued)



example line chart of two PVs plotted overnight

- + can be extended
- + provides access to a rich set of other tools not part of MEDM
  - Archiver
  - logging (olog) package
  - general charting tool
  - alarm handler
  - EPICS PV tree
  - support for user preferences
- + contemporary user interface, includes tabbed displays
- + code is maintained by one or two programmers at ORNL
- + runs on Mac OSX, Windows, and Linux
- + restores previous session
- + can save window arrangement as a perspective

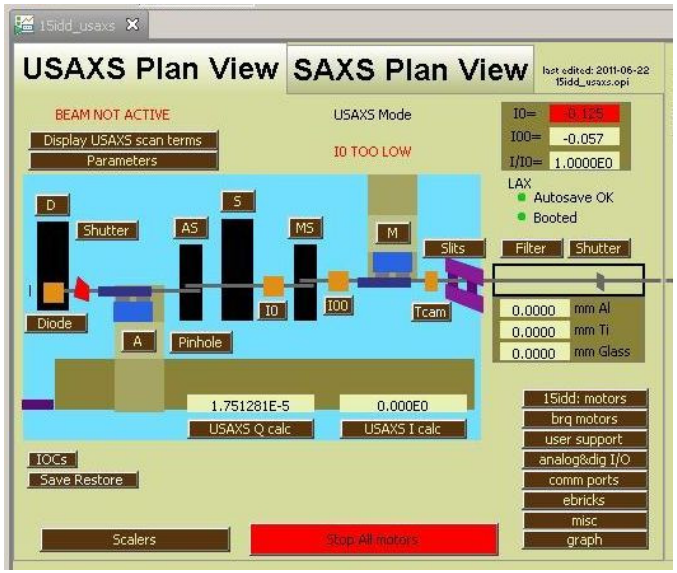
the work required to make CSS BOY a viable product

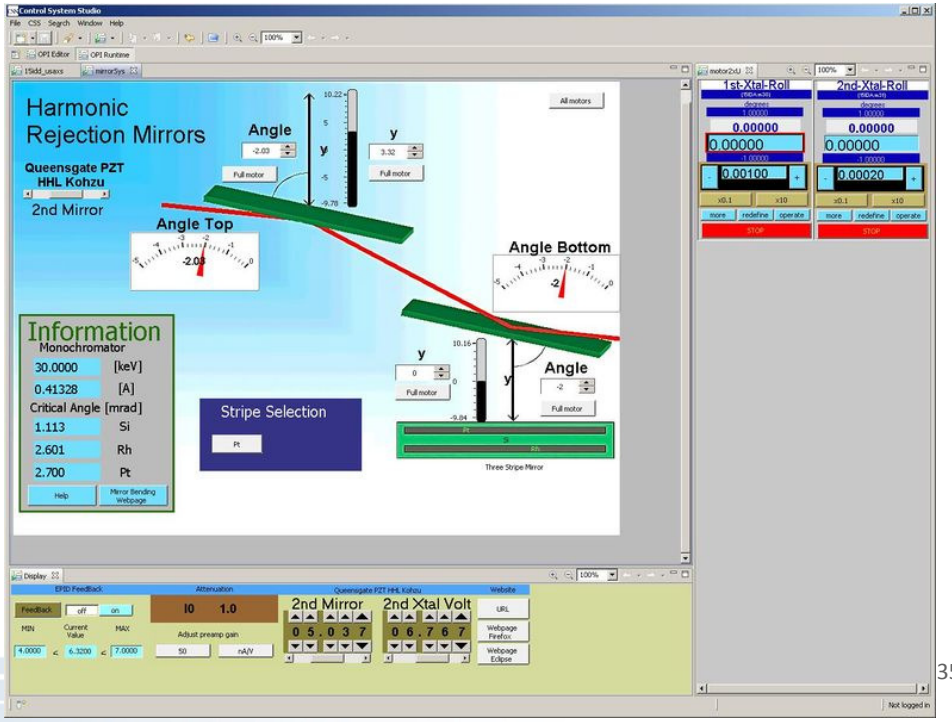
- CSS BOY is available as a product ready to be used in the EPICS community.
  - A manual for CSS is available online which includes content specific for CSS BOY. Deployment of CSS BOY at APS, on the accelerator, control room, and beamlines will take more examination.
  - A preliminary document describing the installation and configuration of CSS BOY at the APS has been prepared.
  - CSS BOY can be run from a shared server (such as /APSshare)
  - 15-ID USAXS uses it now, since 2011
- 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.
- 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.

How CSS/BOY 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, widgets actions, and a converter for .adl files.

Examples from 15-ID





caQtDM

- **C**hannel-**A**ccess **Qt**-based **d**isplay **m**anager (caQtDM)

- Feature-based replacement of MEDM
- Created and is currently managed by Anton Mezger (Head of Accelerator Operations) of the Paul Scherrer Institut.
- It is written using C++ and Qt.
- Runs on Microsoft Windows and Linux
- 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,

caQtDM : advantages and disadvantages

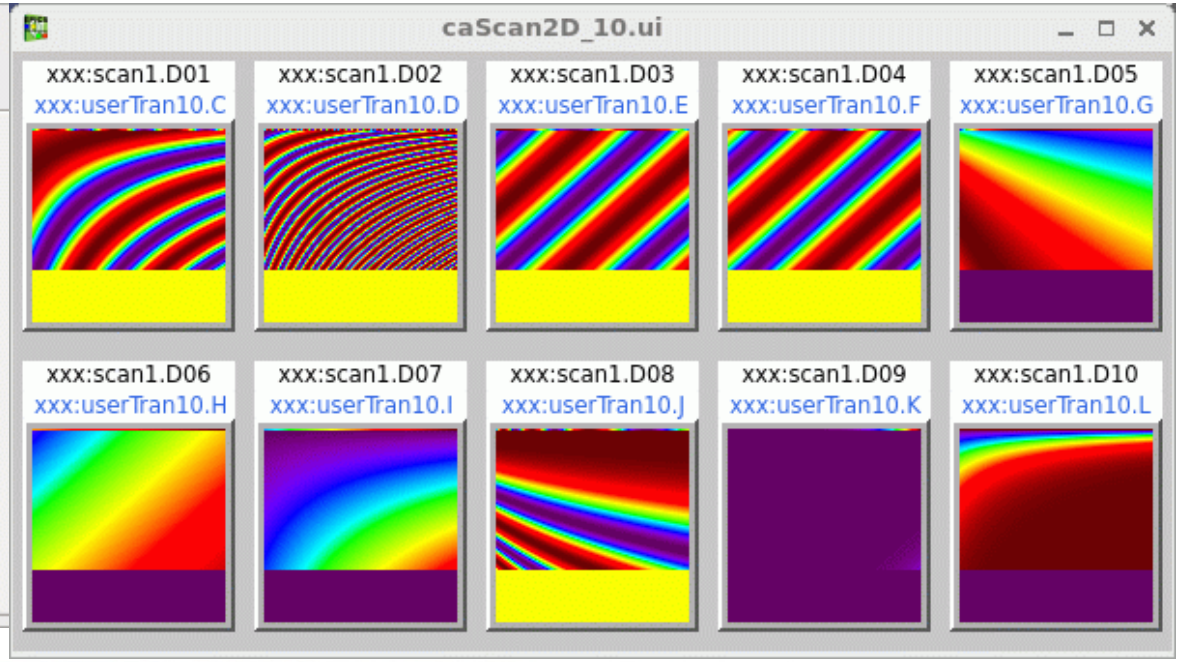
- + Nearly exact replacement of MEDM
  - + synApps' More/Less buttons work as intended
  - + Displays come up with the expected sizes
  - - Can't drag-and-drop PV names
  - - Some displays require adjustment of text-box sizes
- Extension requires learning curve (C++/Qt)
  - + SSG already uses C++ and Qt
  - learning curve limited to caQtDM code (same as any other)
- - code is dependent on a *single* programmer (at PSI)
- + caQtDM can display widgets created for epicsQt

work required to make caQtDM a viable product

- Deploy *static* build to /APSshare.
- Support save/restore placement of many displays on many workspaces
  - Placing displays is already supported. caQtDM maintains the information required to write a script that would restore the placements of a running collection of displays, but (like MEDM) it does not have code to write such a script. This might take a day of work.
- Support drag-and-drop of PV names. Dropping the current text selection onto a link-PV widget works, however.

How caQtDM 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 and image from areaDetector) has been demonstrated here, and the development process has been documented. It requires knowledge of C++/Qt, and some caQtDM-specific knowledge.
- Support for Mac OSX?



epicsQt

- Widget-based toolkit to construct EPICS graphical user interface displays.
  - Created to replace in-house screens and also to replace EDM.
  - Created and is currently managed by Andrew Rhyder and others at the Australian Synchrotron.
  - It is written using C++ and Qt.
- **advantages and disadvantages**
  - - It appears that this tool is not widely deployed
    - - either at AS or as well as at other facilities.
    - no coherent collaborative development effort
  - + has an editor
  - - reuse of epicsQt widgets in python conflicts with our existing use of PyEpics
- **the work required to make this a viable product**
  - We know very little about epicsQt beyond the performance testing reported here
- **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

Comparisons against the requirements

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 against the wish list

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

Comparison against developer criteria

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
- I: requires Motif/X11 support

Performance comparisons

test system: standard issue HP Compaq 8300 Elite Convertible Minitower running RHEL6

program	version	CPU utilization (%)	Max update rate (Hz)	Loss-less rate (Hz)
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

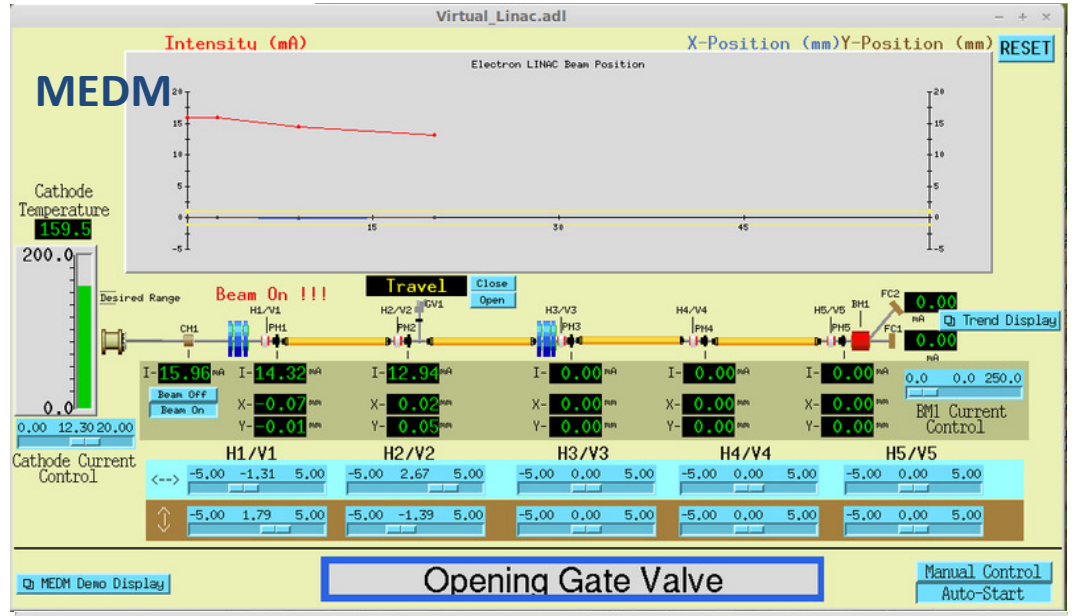
(*) caQtDM throttles display updates at 5 Hz by default.

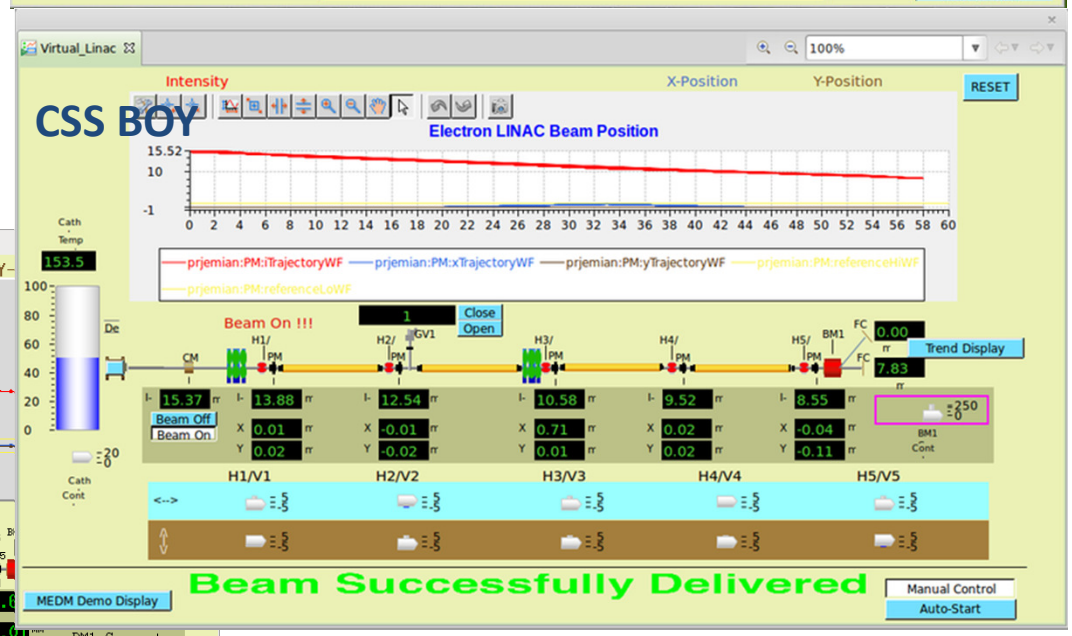
Max update rate: This is the maximum rate (Hz) at which the display manager rendered changes to PVs. Beyond this, the performance decreases as the display manager consumes resources simply keeping up. These are for a for a numerical display only.
Loss-less Update: This the maximum rate (Hz) at which the display manager rendered changes to PVs without any data loss (without missing update events).

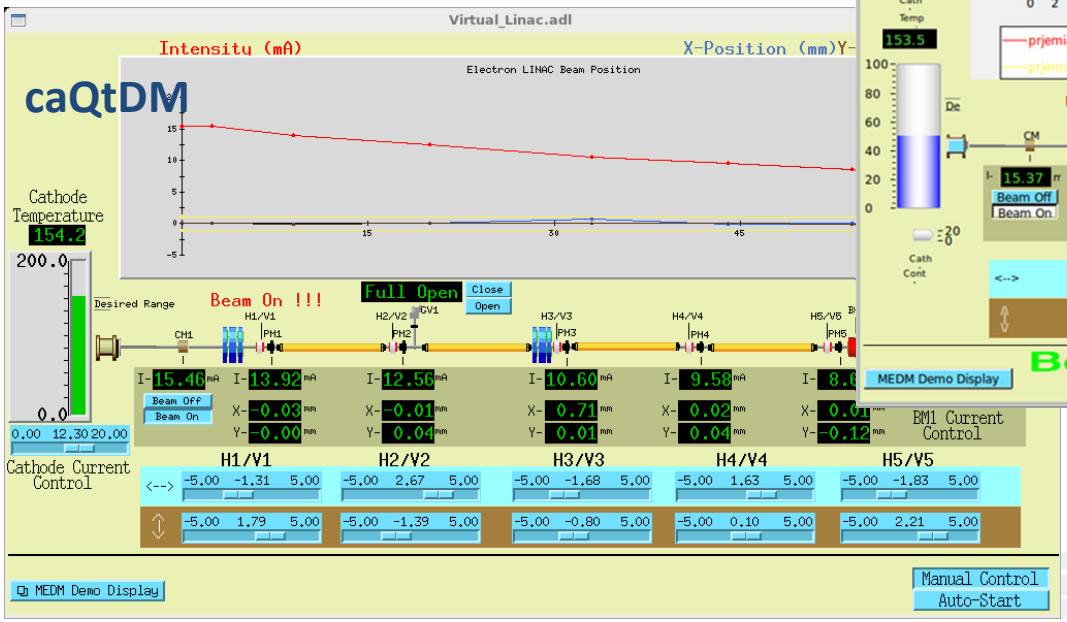
Test results obtained using a 3000 PV EPICS database. 500 PV display blocks were created for each display manager. Each PV was ramped from 0-99 incrementing by 1 at 10 Hz. The count was then displayed. PVs were reused on displays when required. An independent video camera operating at 30 fps was used to determine the performance of the display manager. The number of displayed PVs were increased until failures were observed.

Translation Example using the Virtual Linac

- The same MEDM screen rendered in different tools
- No additional adjustments applied after automated translation
- Note that auto-tune was in different stage of completion for each







Good displays all, but CSS BOY screen needs more work

Devices at APS that use EDM

- Oxford (for example) has provided instruments with EDM interfaces
- 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."*
- 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."*

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

caQtDM: Feedback from users and developers

- 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:*

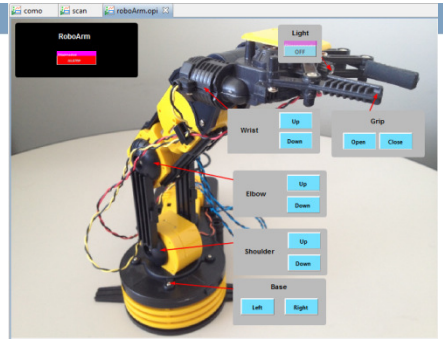
```
caQtDM xxx.ui&
sleep 1
caQtDM -attach -macro "P=xxx:,S=scan1" -dg +500+300 scan.ui
sleep 1
caQtDM -attach -macro "P=xxx:,S=scan2" -dg +100+300 scan.ui
...
```

- 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.*

CSS BOY: at 15-ID-D USAXS

- In 2010, a major revision of the GUI control system was needed. The USAXS instrument decided to port to CSS BOY while the rest 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 (perceived) flexibility and GUI capabilities of CSS BOY are higher then 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**. … it resulted in an interface that is easier to use for users than what MEDM interface was or could ever be.

CSS BOY: other experience at ANL



- **2012 ANL Energy Showcase, BCDA Demo used CSS BOY**
  A control screen was prepared by an undergraduate intern and provided access to basic motions of the various motors and LED on a robotic arm.
- **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.
- **SPX0 Controls use CSS BOY**
  CSS BOY is extensively used for SPX0 Controls. 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."*
- **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.
- **ANL High Energy Physics Implementation use of CSS BOY**
  CSS BOY was provided as the GUI for a Gammasphere with positive reception.

CSS BOY: comments from SNS Control Room
from Charles Peters

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.*

CSS BOY: more comments from SNS Control Room
from Charles Peters

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.*

Recommendations and Deployment Plans: 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

Recommendations, including plans for deployment -1

- 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. 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. We should choose only one of these candidates as the successor to MEDM. Prepare similar demo suites in CSS/BOY, caQtDM, 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

Recommendations, including plans for deployment - 2

- We recommend following this generalized deployment plan.
- user comparison and evaluation
- community feedback from the comparisons (setup a community wiki, page for each candidate)
- task force charged with recommending final selection for common facility support
- implement required features
- plan for implementation of wish list features
- establish training schedule
- establish deployment schedule
- execute deployment schedule
- 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 AND Beamlines

Plans for the MCR to Compare the Candidate MEDM Replacements

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

Plans for APS Beam Lines to Compare the Candidate MEDM Replacements

- identify a small number (2-5) of beam lines for comparison tests, XSD can suggest
- provide a suite of each beam line's MEDM screens in all candidates (CaQtDM, CSS/BOY, and EpicsQt)
- provide each candidate (if possible, ready for execution directly from /APSshare)
- automate the screen conversion process (may already be done for some candidates)
- include area detector or 2-D images where appropriate
- include at least one complex MEDM setup (such as 2-ID-D, described above), there is other complexity
  - 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

Thank you for your attention