ALS CONTROL SYSTEM UPGRADE IN C#*

H. Nishimura, M. J. Beaudrow, W. Byrne, C. Ikami, G. J. Portmann, C. Timossi and M. E. Urashka
LBNL, Berkeley, CA 94720, U.S.A.

Abstract

We are just completing the demonstration phase of the high-level control system upgrade of the Advanced Light Source (ALS) injector. The goal of this upgrade was to modernize the control room operator interface software and hardware. To upgrade the software, we are consolidating and replacing exiting programs with new applications programmed with C#, Matlab and some EPICS EDM. To upgrade the hardware we replaced console computers with higher performance PCs running Windows Vista. The accelerator area of upgrade is from the electron gun, through the LINAC, to the booster injection point. The upgraded system is now in the testing phase. At the same time, we have started the new phase of upgrading the remaining sections by using the newer technologies.

THE CURRENT STATUS

The ALS Injector Control System

The ALS[1] control system[2] has been gradually migrating to EPICS[3] for over a decade since its commissioning in 1993. All the accelerator upgrades adopted EPICS whenever possible. On the other hand, the injection control system is still using the original control system. Due to the increasing age of this system and the launch of the top-off injection project[4], an upgrade for the injector has became more urgent. A small team, known as **HLC** (High Level Controls), with limited resources and a tight time schedule, was formed to address this problem.

The team addressed two primary issues. First, the underlying Application Programming Interface (API) of the existing software, which was developed for the original system, needed to be modified to use channel access. Second, over the long history of the ALS, so many different applications had been developed that performed similar functions, that it became confusing for operations to operate both the injector and the machine in general; these applications had to be re-written and consolidated.

Choice of the .NET Framework

The **HLC** chose the .NET Framework[5] and the C# programming language[6] for re-writing the software since we already had significant experience in Windows development including using Windows with EPICS Channel Access and .NET (SCA.NET[7]). Similarly, we chose to use the latest version of Windows (Vista) since we were already using Windows (2000) on the existing consoles.

*Work supported by the U.S. Department of Energy under Contract No. DE-AC02-05CH11231

Software Design

The main design goals were to make the software *component-based* and *data-driven*. Screens with live data are assembled from Windows Forms custom controls, which in turn are built with the SCA.NET component. The data for configuring these components, such as the process variable names, is read from XML files at run time. These XML files are created from data from a relational database using ADO.NET 2.0[8] and LINQ to XML[9].

Although the .NET runtime on the Windows Vista is version 3.5, for this demonstration we chose to stay at the 2.0 level but to utilize some 3.0/3.5 features such as *LINQ to XML* and the *Windows Communication Foundation* (WCF)[10] keeping us compatible with the Unix/Linux implementation of .NET called MONO[11].

Progress

The goal of the initial phase was to operate the first half of the injector section entirely from the newly developed software running on the new consoles[12]. In addition to .NET, several important applications were developed with Matlab[13] and EPICS EDM[14].

After a year's effort, we are in position to commission the new system: the software and some hardware is in place and some operator tuning of the injector has been tested. The final step in this phase will be to employ the new system in day-to-day operation. The biggest difficulty to this commissioning effort has been the successful operation of top-off restricting our access to the injector for testing to a few hours of physics shifts which occur at most one every week. More detail will be reported in our companion paper[15].

THE NEW PHASE OF HLC

The Plan

While completing the initial phase, we have started the next phase of development: to replace the controls for the rest of the accelerator, the storage ring and booster. During the new phase, we will make a change to our development strategy by switching from WinForm to the Windows Presentation Foundation (WPF)[16].

WPF allows a separation of application logic and the description of its graphical properties. A .NET language, like C#, is used to create the application logic but the graphical configuration and simple behavior are described in an XML-based language called XAML (the Extensible Application Markup Language)[17] which describes the configuration of user-interface (UI) components and the interactions between their properties.

EPICS Display Manager in WPF

By creating custom controls for EPICS Channel Access, which we call EPICS WPF Components, other UI components can interact with EPICS data by defining the relations in XAML. This means that EPICS client programs can be created in XAML by using an XAML editor such as Expression Blend 2[18]. The process of creating WPF applications in XAML is similar to that of authoring web pages in an interactive visual designer. We have also been using EPICS WPF Components to create an application that we call *EPICS Display Manager in WPF* which can be use to create EPICS clients with simple graphics.

An example will illustrate our use of EPICS in XAML. We use *dm* as the XML name space to reference our components. *DmAnalogChannelEdit* is one of the EPICS components that exports a public property *Channel* that holds the EPICS Channel name.

```
<dm:DmAnalogChannelEdit
        Channel="SR13C___HCM1___AC00"/>
```

When a program runs, this component reads and displays the channel value at 1 Hz. We can type in a new value and set it to the device. These functions are embedded in C#.

To add a Scroll Bar, a standard WPF UI component, and link it to the EPICS component. First, give the EPICS component a name *HCM1* so that the scrollbar can refer it.

```
<dm:DmAnalogChannelEdit
   x:Name="HCM1" Title="SR13 HCM1"
   Channel="SR13C___HCM1___AC00"/>
```

Then, we create a scrollbar and bind its value to that of the EPICS component.

```
<ScrollBar Orientation="Horizontal"
   Maximum="10" Minimum="-10"
   Value="{Binding ElementName=HCM1,
            Path=Value, Mode=TwoWay}" />
```

These two components are now synchronized by the property Mode=*TwoWay* so that the scrollbar's knob follows the channel value, and vice versa. This is a feature of WPF called Data Binding[19].

As the above example shows, we can use EPICS WPF Components with other components, and make use of various features of WPF. We do not need to create any special parser or scripting. In addition, we can add C# routines whenever needed which makes this Display Manager highly versatile.

Conclusion

After a year of working with C# and .NET on Windows Vista, we have demonstrated that these tools can be used successfully for modernizing the high level software controls for the ALS injector. It remains to be seen how a graphics editor for WPF like Blend can be used by operations staff and others to produce useful accelerator applications with minimal C# coding.

AKNOWLEDGEMENTS

The authors thank A. Biocca, P. Denes and D. Robin for their support. We appreciate patient cooperation of machine operators for using new programs and giving us constructive comments.

REFERENCES

[1] ALS CDR, LBL PUB-5172 Rev. LBL,1986
A. Jackson, IEEE 93PAC, 93CH3279-7,1432, 1993.
[2] S. Magyary, IEEE PAC93, 93CH3279-7,1811,1993.
S. Magyary et al, NIM A 293, 36-43, 1990
[3] L.R. Dalesio, et al., ICALEPCS '93, Berlin, 1993.
http://www.aps.anl.gov/epics/
[4] C. Steier, et al., PAC 2007, Albuquerque, 1197, 2007
[5] http://www.microsoft.com/net/
[6] http://msdn.microsoft.com/en-us/vcsharp/aa336809.aspx
[7] H. Nishimura and C. Timossi, PCaPAC 2005
H. Nishimura and C. Timossi, PCaPAC 2006, 37
C. Timossi and H. Nishimura, PCaPAC 2006, 56
C. Timossi and H. Nishimura, PCaPAC 2008, 24
[8] http://msdn.microsoft.com/en-us/library/aa286484.aspx
[9] http://msdn.microsoft.com/en-us/library/bb387098.aspx
[10] http://msdn.microsoft.com/en-us/library/ms735119.aspx
[11] http://www.mono-project.com
[12] H. Nishimura et al., PCaPAC 2008, Ljubljana, 122
[13] G. Portmann, PCaPAC 2005, Hayama, LBNL Pub-925
[14] J. Sinclair, http://ics-web.sns.ornl.gov /kasemir/train_2006/ 1_4_EdmTraining.pdf
[15] G. Portmann et al., this proceedings.
[16] http://msdn.microsoft.com/en-us/netframework/aa663321.aspx
[17] http://msdn.microsoft.com/en-us/library/ ms752059.aspx#xaml_files
[18] http://www.microsoft.com/Expression/
[19] http://msdn.microsoft.com/en-us/library/ms750612.aspx