EPICS Office

M. Clausen *DESY, Hamburg, Germany*
G. Tkacik *Cosylab*

ABSTRACT

The EPICS toolkit has built up a very mature foundation for today's control systems. Over the past three years there has been a discussion in the collaboration whether the toolkit is well prepared for the challenges of the future. As a result the collaboration has decided to add more functionality to the core software running on the front-end processors called Input Output Controllers (IOC's) and to the channel access (CA) protocol to support these new functionalities.

The same adjustments to the existing controls applications will also be necessary. This has lead to the fundamental question, whether the existing applications, written mostly in X-Windows or Motif, were still state of the art, or whether a basic redesign would be more adequate. The latter has been proposed and is now in the design phase.

This paper will describe the fundamental building blocks of the new EPICS Office controls application workbench. Basic functions known from other office packages, like common look and feel and drag and drop of names and data, are fundamental to the EPICS Office functional specification. The desire to build a highly modular workbench has lead to the decision to use the Eclipse IDE and its plugin technology defined by OSGi as the fundamental building block.

This modular design will allow a very flexible integration into the existing EPICS framework as well as into the new implementations as they come along. The support for legacy applications and the integration of commercial packages like Matlab will be addressed as well.

A description of the Eclipse Rich Client Platform (RCP) as well as the organization of the collaborative software development of the EPICS Office will be described in other contributions throughout this conference.

INTRODUCTION

The EPICS collaboration has a long history. Starting with the beam telescope control system implemented on VAX-Stations using a DEC-Windows graphical user interface at Los Alamos, the first version implemented on UNIX workstations was available around 1988 named GTACS (Ground Test Accelerator Control System). The collaboration with Argonne, starting in 1989, lead to the first and most successful open source control system developed in a worldwide collaboration, named EPICS.

The design criteria for the initial implementation were reliability and performance. Computer hardware was slow compared with today's CPU boards and network speed was limited to 10Mb. Therefore, front-end processors were running the real time operating system VxWorks exclusively. The network protocol was tuned for performance and the display tools running on Sun Workstations were optimized for fast update rates. These performance design criteria could be met by accepting functional limitations in other areas. As an example, the channel access network has been optimized for throughput and response times by limiting the maximum communication packet size to the size of an Ethernet frame. This and other fundamental limitations have been discussed in the EPICS community for a long time. In November 2002 it became apparent that EPICS will need a long term strategic program to incorporate new technologies and to overcome its long term limitations.

As a result in 2005 a core group was started to design a new EPICS version. New functionalities on the front end systems and a rich functionality on the network level shall be made available by the end of 2006. These new functionalities can only be used if the applications on the workstation level are developed accordingly.

NEW APPLICATIONS

Most of the existing applications have their origin in the early days of EPICS. They have been developed for UNIX Workstations and are using either X-Window or Motif as the graphical display engine. As long as most of the EPICS control systems are running in the UNIX environment (Sun/Solaris, HP-UX, Linux, or X emulators on Windows-PCs) these kinds of applications are appropriate and have gone through a continuous process of modifications and upgrades. This collaborative development process was carried out by several laboratories. The central cvs repository at Argonne has been used to coordinate the multi-lab development effort.

The next Generation

Over the years new application developers have joined the EPICS collaboration. New ideas have come up and new requirements had to be fulfilled. Platform independent programming and object oriented programming have gained much higher priority than programming in 'C' or 'C++'. Rich graphic libraries are more important than application tuned for high performance written in native code.

Thus Java has gained momentum for new applications. A rich set of class libraries, platform independence thin client as well as rich client applications and, last but not least, enthusiastic students are the driving forces for this direction. The support for X-Window or Motif applications is getting harder and harder because the experts retire as time goes by.

The Chance

The development of a new EPICS version is not only important in order to prepare for and fulfil the requirements of future projects. It is also a chance to coordinate work on new applications. The current applications have been written by several independent programmers in different institutes. Each application has its personal look and feel. Menus and their items are placed individually. Configuration files differ between applications. The agreement that a new application suite will be necessary for the next EPICS version implies that we have a chance for a new – common – application environment.

EPICS OFFICE

The idea for a new control system application suite was presented at the regular EPICS meeting held in April 2005 at SLAC [1]. The parallelism with the development of the Microsoft Office tools – starting from individual independent tools, forming a homogeneous toolset with a common look and feel – initiated the name "EPICS Office". This name shall imply that the new suite of controls applications will incorporate a lot of the well accepted features of the existing Office packets.

Setting the Ground

Before starting to implement the applications, we needed to find a common ground whereto the train should bring us. Initial thoughts have been collected in a functional breakdown paper [2]. In a second step several available applications frameworks were analyzed and have been discussed [3], targeting a framework which will cover the following requirements.

The framework will:

- Ease new application developments
- Provide support for common look and feel, drag & drop, copy & paste, file operations, undo, browsing, common data visualizations, forward and backward navigation
- Follow the plug-in design pattern which will allow component based developments
- Provide long term perspective in the open source community
- Be a well accepted tool kit, supported by many companies

As a result of the evaluation [3] the Eclipse framework was proposed and was well accepted as the development platform for EPICS-Office applications.

ECLIPSE

After the principal decision was made to go ahead with Eclipse, several announcements by software companies encouraged us to continue along this path. Companies like Wind River (development environment), Macromedia (development platform for rich client applications), Oracle (JSF tooling project), Compuware (developer version of Optimal J) have committed themselves to supply application based on the Eclipse workbench. The Eclipse Foundation has 101 members. Recently Nokia and IONA joined as Strategic Board Members. There are 68 add-in providers and more than 900 Eclipse plug-ins. The number of committers to Eclipse projects has increased from 220 to 470 over the last year.

Running anywhere and SWT/ AWT

A discussion about Eclipse always has to deal with the question whether the compile once/ run anywhere paradigm must be fulfilled or not. The 'Eclipse story' is a story of the two companies Sun and IBM and their approach for the optimal IDE. While Sun pushes for AWT/Swing, IBM prefers SWT/ JFace. Since Eclipse initially was, and still is, a product from IBM, it comes with SWT and thus has the limitation that the graphic package uses the native graphic system of the host machine. The compile once run anywhere paradigm is broken. The code only runs on machines where a JNI interface library to the native graphic library exists. The question whether this really limits the developed application to run on all platforms currently used in the EPICS community is still open. The future will tell us whether we will ever use a platform, and where a Java runtime environment exists (at least a JVM must be available for the platform) but no SWT/JNI library will be available. At least for now this does not apply.

Supported Operating Systems
Windows
Linux (x86/GTK 2)
Linux (x86_64/GTK 2)
Linux (PPC/GTK 2)
Linux (ia64/GTK 2)
Linux (x86/Motif)
Solaris 8 (SPARC/GTK 2)
Solaris 8 (SPARC/Motif)
AIX (PPC/Motif)
HP-UX (HP9000/Motif)
Mac OSX (Mac/Carbon)

Table 1: Platforms for Eclipse SDK

Eclipse IDE

Eclipse has been developed as an Integrated Development Environment (IDE). As Eclipse was written in Java, the first supported language was of course Java itself. Nowadays Eclipse supports several other languages. One of these is also 'C' / 'C++'. This creates the opportunity that all members of a development team can work with the same tool. Basic functionalities are the same. Besides the same look and feel, the common access methods to cvs repositories must be mentioned as an important feature.

In addition to the free support for several languages, various tools supporting software development are available free of charge. The UML toolkit omondo and the database development toolkit Hibernate are just two examples of a growing suite of tools developed for Eclipse.

One of the main reasons for the fast growing Eclipse community is the basic idea of Eclipse – namely the idea that Eclipse itself just provides a basic frame which accepts and supports plug-ins written for Eclipse. Plug-ins can depend on other plug-ins. This way the whole Eclipse workbench is a bouquet of interacting plug-ins.

Update Mechanism

One built in feature of Eclipse is the web based update site. Simply by selecting an update site by http address, the developer can browse the content and select and install plug-ins. Besides this update feature, Eclipse itself has a built-in wizard which helps to create update sites and to deploy projects on the web.

Eclipse RCP

The development of the latest version of Eclipse has improved the features for using Eclipse as a rich client platform (RCP). Developing Java applications in Eclipse has become easier. Another built in wizard helps to setup your application environment. Eclipse RCP applications can be started within the same Eclipse workbench, they can also be started as individual new Eclipse instances. This leaves a lot of freedom to the developer.

As an example for the RCP support in Eclipse, the GEF framework should be mentioned. The Graphical Editing Framework (GEF) allows developers to create a rich graphical editor from an existing application model. GEF consists of two plug-ins. The org.eclipse.draw2d plug-in provides a layout and rendering toolkit for displaying graphics. The developer can then take advantage of the many common operations provided in GEF and/or extend them for the specific domain. GEF employs an MVC (model-view-controller) architecture which enables simple changes to be applied to the model from the view. In this way GEF also helps to implement thin client applications which run independent of the Eclipse framework.

One prominent example for an Eclipse application in the science field is the gumtree project [4]. It could be called a predecessor of the EPICS-Office project. Even though the two projects have been started complete independently from each other, they both came to the conclusion that the Eclipse environment would fit best for a collaborative software development in the science field.

COLLABORATIVE DEVELOPMENTS

EPICS-Office is an initiative from within the EPICS collaboration. Like any other development in the EPICS collaboration it is an effort driven by several laboratories around the world. A web site has been set up in order to coordinate the development between the international partners [5]. The development environment and the collaboration are described in a paper presented during this conference [6].

INTERFACES

While the existing EPICS applications have been written to exactly match the requirements for the EPICS control system, we want to take the chance to start the EPICS-Office with a more general ansatz. The Control System Office is the idea behind initiative: Writing applications which could also be used in conjunction with other control systems. To fulfill this more generic ansatz we can widen the scope of the interfaces being used in the Eclipse environment. The collection of requirements for the basic interfaces has been started. The specification will be written with the aim of providing a 100% compatible API to the new EPICS-V4 interfaces, but also to enable compatibility with most of the major existing control systems like TINE at DESY or TANGO at ESRF, and of course the previous EPICS version V3.

Needless to say, all of the interfaces will be written as plugins for the Eclipse framework. This way it will be possible to easily extend the spectrum of supported control systems on demand.

Control System API (Data Access Layer)

The most important interface for a control system application is the data access layer. To give this interface the necessary focus, the design for this interface has been subcontracted to Cosylab by DESY. A detailed description of the work can be found in the paper presented during this conference [7].

Name Services

One of the missing functionalities of the existing EPICS implementation is the absence of a name service for the records/ devices throughout the system. This feature will be part of the next EPICS version und thus must be also foreseen as an interface for new EPICS-Office applications. Regardless what the implementation in the control system(s) will look like, the implementation in the Java(Eclipse) framework should make use of the state of the art naming techniques. For the Java environment the focus – for now – is on the Java Naming Services. For the new EPICS version the new naming service will probably be based on LDAP. Since there are already several implementations of the JNI to LDAP mapping, this is a reasonable approach.

Archive Interface

Nowadays every control system has its own way to store archived data. It's obvious that this also results in several individual archive interfaces to these data. An archive application interface (AAPI) has been developed at DESY. This interface allows not only access to different data sources, but also provides a network layer for client/ server implementations. In addition to this bottom up approach from DESY (for existing X-Window applications), another approach for a top/ down design is on the way and will be presented during this conference [8]. The implementation of this archive viewer is based on the model view controller (MVC) approach and is implemented in Eclipse. The API to the archive data is implemented as a pluggable interface in the Eclipse manner. This interface will also be used for the EPICS-Office archive applications.

ACTUAL STATUS

After the initial kick off in spring 2005 several activities have been launched. In conjunction with the web page [5] a mailing list has been established [9] and the necessary collaborative tools are in operation. Several developers have shown an interest in workin collaboratively on the implementation of EPICS-Office applications based on the Eclipse framework. The fundaments data access API is subcontracted. A beta version of this API is due before end of 2005. Experts from DESY and local companies in Hamburg are working on the specification of the Eclipse framework for the EPICS-Office environment. The first feasibility studies for drag and drop implementations show promising results.

CONCLUSIONS

The development of the new EPICS version V4 will give the EPICS collaboration a chance to develop a homogeneous control system environment from the front-end (IOC) level to the operator interface layer. The performance of processors and networks will allow new techniques on the front-end and the application layer. The demands to run EPICS applications on UNIX workstations as well as on MS-Windows workstations is calling for more flexibility. X-Window applications cannot fulfil these requirements anymore.

Java as the programming language and Eclipse as the development and the (rich) client application framework has been well accepted in the EPICS collaboration. The activities for a new initiative have been launched and already show results. The aim of presenting the first beta releases of new applications based on this ansatz in 2006 is ambitious, but still possible. Ideally we will see a new EPICS version with a new EPICS-Office application suite at the same time.

REFERENCES

[1] M. Clausen, "EPICS workbench", EPICS meeting 2005 at SLAC (http://www-ssrl.slac.stanford.edu/lcls/epics/agenda.php)

[2] Control System Office Whitepaper, http://users.cosylab.com/~kzagar/cso/SPE-Control_System_Office_Whitepaper.html

[3] Control System Office Framework comparison, http://users.cosylab.com/~kzagar/cso/SPE-CSO_Framework_Evaluation.html

[4] Gumtree: An Eclipse based graphical user interface, http://gumtree.sourceforge.net/wiki/index.php/Main_Page
[5] EPICS-Office web site, http://epics-office.desy.de
[6] M. Clausen, M. Möller, „Collaborative Software Development“, ICALEPCS’2005, Geneva Switzerland
[7] G.. Tka_ik, M. Ple_ko, M. Clausen, “Transplanting the Success of Eclipse to Control Systems”, ICALEPCS’2005, Geneva, Switzerland
[8] S. Chevtsov, “EPICS Archive/ Viewer”, ICALEPCS’2005, Geneva, Switzerland
[9] Mailing list for EPICS-Office coordination: epics-office@desy.de