Submitted to:
International Conference on Accelerator and Large Experimental Physics Control Systems - '97
11/3/97 - 11/9/97 - Beijing, China

BNL-64938
CONF-971171--

What Objects Do Controls Applications Need?*

J. T. Morris, A.C. Abola
Alternating Gradient Synchrotron(AGS) / Relativistic Heavy Ion Collider(RHIC) Controls
Brookhaven National Laboratory
Box 5000, Upton, NY 11973-5000

Abstract

Discussions of object-oriented controls programming often focus on the important interface to accelerator devices. Experience shows that, for most applications, only a small part of the application work involves the accelerator device interface. Much of the work in application programs is dedicated to other areas such as user interface, data management and physics calculations. This paper will consider the software objects beyond the device interface that provide the most assistance to application programmers.

1 Introduction

Object-oriented interfaces to accelerator devices are being actively explored and discussed in the controls community [1,2]. The interface to accelerator devices, however, is only one part of the work of controls applications. This paper will focus on software objects that can help programmers in other areas of application development.

In over eight years of C++ programming in the AGS Controls Group, an extensive library of classes has been developed to support the work of application programmers. This experience is used as the basis for a discussion of the objects needed for application software. The paper begins by examining several sample controls applications and categorizing the work performed in the applications. In the sections that follow, the use of software objects in each of these categories is discussed. This discussion includes a description of the objects used and a consideration of the impact of the use of these objects on the application development process. Attention is also given to areas where the need for additional objects has been recognized.

2 Controls Application Software Categories

In order to understand how best to aid application programmers, it is important to understand where time and effort is spent in application development and maintenance. Five applications were chosen as representative samples of applications that have been developed for control of the AGS accelerator and the Booster injector. These applications all make use of objects from AGS/RHIC C++ class libraries. The source code for the selected applications was reviewed and classified into the following five broad categories.

1) General User Interface
2) Mathematics/Physics Algorithms
3) Data Management/Storage
4) Data Acquisition/Control
5) Graphic Displays

The percentages listed in Table 1 represent the proportion of the lines of application code that fell into each of these categories. A count of lines of code provides an approximate measure of the effort spent in developing and maintaining an application. The total number of lines of code in each application is also listed in the table.

Some clear variations in the distribution of code can be seen in these applications. This variation depends to some extent on differences in requirements for the applications. It also depends on the extent to which these applications were able to take advantage of objects from C++ class libraries. One glaring difference can be seen in the overall size of the BoosterOrbitDisplay program, which was written at a time when many of the objects discussed in this paper were not available. Note that this program is nearly three times the size of any of the other applications.

3 User Interface Objects

A significant part of the code in any application is devoted to the user interface. Application programmers need tools that allow them to rapidly create prototype interfaces but also allow the flexibility to customize interfaces to the needs of an application. In the AGS/RHIC development environment, a UI Toolkit[3] has been developed to meet these needs.

The UI Toolkit is a a very well written and well documented collection of C++ classes layered on top

* Work performed under the auspices of the U. S. Department of Energy

Table 1. Software Distribution in Controls Applications

	Booster Orbit Display	Booster Main Magnet	Ags Gamma Jump	Ags Loss Monitor	Ags Orbit Control	Average
General User Interface	20%	37%	37%	32%	24%	30%
Math/Physics Algorithms	20%	36%	23%	10%	22%	22%
Data Management/Storage	29%	15%	10%	26%	27%	21%
Acquisition & Control	16%	4%	10%	19%	21%	14%
Graphic Displays	15%	8%	19%	13%	6%	12%
Total Lines of Code	12000	4500	3000	3700	4700	-

of OSF/Motif and the X windows system. All the the toolkit. UI Toolkit objects include pushbuttons, labels, fields for text or numeric data entry, toggle switches, scrolling text areas, selection lists, pulldown menus and many other objects. UITable objects, which are layered on top of a commercial widget for the display and editing of tabular information, have been an important addition to the basic toolkit. They are now being used heavily in AGS/RHIC controls for data display and data entry. All objects in the UI Toolkit are derived from a common UIObject class and conform to interface standards. The application programmer is therefore able to combine these objects freely in the application interface. Programmers may also derive UI objects from the objects in the toolkit to create specialized components for an application. The AGS/RHIC controls libraries include many examples of UI objects with specialized controls content. One example is the AgsPage. Built around a UITable, an AgsPage object supports device control and continuous display of device settings. It is used in almost all AGS applications.

The major impact of the UI Toolkit on application development has been the improved ability to build customized user interfaces. Earlier tools at the AGS provided an efficient method of building simple user interfaces, but extension to more complex interfaces was very difficult. The availability of UI Objects as user interface building blocks allows the application programmer to build a user interface that is a better match to the needs of the application.

The use of UI Objects as user interface components has also assisted in efforts to present a common look and feel in controls applications. User interface guidelines for application developers are strongly recommended, but some level of conformity can be achieved just by the use of common tools. This is basic components of a user interface are included in important in an environment where many individuals are responsible for the development of application code. In the AGS/RHIC environment, there is an increased interest in the development of application software by members of the accelerator physics and operations groups. As the application development community broadens, the use of common objects becomes increasingly important.

Future plans for AGS/RHIC development are placing an increased emphasis on high level objects with specialized controls content. Generic functionality that would have in the past been packaged in a standalone application can now be packaged in a UI object and made available to any application. Since these high level objects are derived from UI Objects, they may be used as application building blocks in the same way as generic objects.

4 Mathematics/Physics Algorithms

This category includes the data reduction and analysis algorithms applied to data retrieved from the control system. It also includes the control algorithms employed to determine hardware settings for accelerator devices based on physical parameters supplied by a user. Accelerator modeling tools, which could be considered an important part of this category, are not being discussed in this paper.

This category is probably the area where objects were used least in AGS applications. Much of the code in this area is application specific. It also may be well suited to procedural, rather than object-oriented, code. Many AGS applications make use of libraries of procedural legacy code to perform fitting and other analysis techniques. There are some areas, however, where applications have benefited from the use of

DISCLAIMER

objects to assist in mathematical operations.

The DataAverager class makes it easy for applications to derive statistics from accumulated readings from accelerator instrumentation. The FunctionData class was written to support the many time dependent functions important to AGS applications. FunctionData objects represent linear, quadratic, or cubic functions. FunctionData methods are available for mathematical operations including differentiation and vectorization. The use of the FunctionData class for these operations has greatly simplified application code. It is instructive to look at the statistics for the BoosterMainMagnet application in Table 1. In order to apply some special constraints to calculations, this application performed almost all mathematical operations in direct procedural code and did not make use of the FunctionData class. This is one of the reasons that such a large portion of this application's code is devoted to mathematics.

Matrix mathematics is another area shared by multiple applications. C language matrix tools were used in AGS applications. The interface to these C tools was awkward and led to programming difficulties. Application programmers could be helped by a broader set of C++ classes to support mathematics, including matrix operations.

5 Data Management and Storage

This category includes the storage and retrieval of information using files and databases. This information may include physics parameters, display parameters, calibration data, and all other parts of an application's working context. Code to read and write information to files can be time consuming to write and prone to errors. It is important to have tools that relieve the application programmer of concern about where information is stored and how it is formatted. A number of C++ classes have been developed to assist AGS/RHIC application programmers at this task.

The FileSelector Tools have been the most effective aid to application programmers. The basic building blocks for these tools are the FileIdentifier and FileSelector classes. FileIdentifier objects are used by applications to represent files in terms of the process with which they are associated and their role in that process. FileIdentifier methods handle opening and closing of files and the storage and retrieval of standard header information. The FileIdentifier class records header information for stored files in a Sybase[4] database. FileSelector objects use this database to provide methods for the rapid retrieval of FileIdentifier objects according to specific selection criteria.

Most applications do not use FileIdentifier and FileSelector objects directly. These classes have been used as base classes for classes that specify files with specific types of content. FunctionFile objects are used for storage of the many time dependent functions used in AGS applications. The FunctionFile class is derived from the FileIdentifier class and the FunctionData class described in section 4. The ParameterFile class is similarly constructed from the FileIdentifier class and a general purpose ParameterList class. ParameterFiles are used to deal with lists of parameters associated with a process. The parameters may be integer or floating point numbers, text strings, or arrays. An application programmer using FunctionFile and ParameterFile objects does not have to write any code related to data formatting or manipulation of directory paths.

Another important class has been developed to read or write groups of files in a single operation. FunctionFamily objects represent the entire working context for an application. This working context will typically include some number of FunctionFiles and an associated ParameterFile. FunctionFamily objects provide methods for saving named instances of the application context as a single logical "file". The FunctionFamily, which is part of the FileSelector class hierarchy, also provides methods for selection and retrieval of these saved files. FunctionFamily objects also act as containers for process data within running applications.

Most AGS controls applications have benefited greatly from the use of FunctionFamily objects. In the AgsGammaJump application, the FunctionFamily class was used as a base class for a specialized class representing the data for the AgsGammaJump process. This further reduced the amount of code necessary for data storage in the application.

One major benefit of the use of FunctionFamily, FunctionFile and ParameterFile objects has been the guarantee of common data formats. Function and parameter data can be shared between applications. Generic AGS applications for data archiving or function diagnostics can access files stored by any application using these objects.

Two of the sample applications, BoosterOrbitDisplay and AgsLossMonitor, do not deal with FunctionData and were therefore unable to take advantage of the

FunctionFamily class. These applications generally need to save setup files and save snapshots of acquired data. Future plans call for the definition of data format standards for these applications. The SDDS[5] format is being considered. New objects will likely be defined to handle these needs.

Improvements are also planned in other areas of the AGS/RHIC controls libraries. New attention is being focused on tools for data logging. A DataLog class has been written to allow applications to log parameters in files in standard formats. Attention is also being given to object-oriented tools for retrieval of information from Sybase relational databases. Some database access tools have already been written but improvements are expected as database access becomes more important for AGS/RHIC applications.

6 Data Acquisition/Control

This category, the interface to accelerator devices, will not be addressed in detail here. It is worth noting that one reason that only a modest amount of application code is devoted to data acquisition and control in the sample applications is the availability of a simple object-oriented interface to AGS devices. Applications use AccelDevice objects to get or set information from individual devices. A DataCollector object is used to collect data, synchronously or asynchronously, from a grouping of any devices in the control system. An effort is now underway to provide a CDEV[2] service layer that can communicate using both the AGS device interface and the adoIf[1] interface to RHIC devices.

7 Graphic Display Objects

The requirements for graphic display objects are much the same as that for other user interface components. Applications programmers need simple methods to build simple displays, but they also need the flexibility to construct more complex displays when necessary.

The sample applications used graphics tools at two different levels. The QuickGraphics (QG) package is a C language library of basic graphing tools developed in the AGS control system. FunctionDisplayer and FunctionEditor objects were defined for the graphic display of the FunctionData objects that were described in Section 4. The interface to these objects was much simpler than the QG interface. A UIPlot object now is available to fulfill the basic graphing requirements for AGS/RHIC applications.

The UIPlot class, which is layered on top of the widgets in the commercial XRT/graph[6] package, has several advantages over the earlier tools. It includes improved tools for interactive management of a graph. The fact that it is part of the UIObject hierarchy gives the application programmer the freedom to build UIPlots into the user interface as required.

Though some AGS applications do generate 3d displays, generalized 3d graphics tools have not been a part of AGS class libraries. The XRT/3d[6] graphics package is expected to be the basis for new 3d graphics objects in AGS/RHIC applications. Objects that support additional graphics options, such as bar charts, could also prove useful to the application programmer.

8 Conclusion

The development of controls applications requires effort in a number of different categories. The need for software objects has been identified in all of these categories. In AGS applications, the use of C++ class libraries that address many of these needs has simplified the task of the application and enhanced application capabilities. The use of common objects has provided additional benefits by encouraging compliance with standards. The advantage of this compliance with standards has been particularly recognized in the areas of user interface and data storage. Areas have been noted where more benefits could be realized with the availability of additional object-oriented tools. Some of these areas will be addressed by future work in AGS/RHIC controls. The authors encourage continued discussion in the controls community of ways to address all categories of application needs.

9 Acknowledgments

The authors would like to acknowledge the work of the AGS controls staff in the development of C++ class libraries and the applications that use them. We would particularly like to acknowledge Ted D'Ottavio for the creation of the UI Toolkit library and Stewart Mandell for his work on Function classes.

10 References

1. L.T. Hoff and J.F. Skelly, "Accelerator Devices as Persistent Software Objects", Nucl. Instr. and Meth. in Phys. Res. A 352(1994), 185-188
2. J. Chen et al, "CDEV: An Object-Oriented Class Library for Developing Device Control Applications", Proc. ICALEPCS 95, Chicago 1995
3. T. D'Ottavio "UI Toolkit Overview", AGS tech note #471
4. Sybase Copyright,1997 Sybase, Inc.
5. M. Borland, "A Self Describing File Protocol for Simulation Integration and Shared Postprocessors", Proc. Particle Accelerator Conf, Dallas, Texas 1995
6. XRT/graph,XRT/3d Copyright, 1997 KL Grp., Inc.