AN APPLICATION FRAMEWORK AND INTELLIGENT GRAPHIC USER INTERFACE FOR MULTIPLE ACCELERATOR CODES

Barrey W. Hill, Hendy Martono, John M. Moore and James S. Gillespie*
G. H. Gillespie Associates, Inc., P. O. Box 2961, Del Mar, CA 92014, USA
* Permanent address: Intuit, Inc., 6265 Greenwich Drive, San Diego, CA 92121, USA.

Abstract

A multi-platform application framework is being developed for the implementation of a variety of particle physics simulation and analysis codes under a single graphic user interface (GUI) shell. The framework architecture supports plug-in tools such as an interactive particle trajectory module, external data interface tools and hypertext tutorials which interface with installed physics codes. A variety of physics programs are integrated into the framework by separating the computational code, which is implemented as a platform-independent Computational Module, from the I/O requirements, which are replaced by an Interface Module developed with the multi-platform GUI framework. The object-oriented framework provides a sophisticated beamline object model and a rich library of GUI components. Application-dependent components can be derived from the more abstract framework components to support specific requirements of different simulation codes. An overview of the Multi-Platform Shell for Particle Accelerator Related Codes (S.P.A.R.C.-MP) application framework is presented here with illustrations from a Windows 95/NT implementation.

1 INTRODUCTION

The Multi-Platform Shell for Particle Accelerator Related Codes (S.P.A.R.C.-MP) is an object-oriented framework which provides the underlying infrastructure for the development of software applications focused on improving the accessibility and ease of use for a variety of particle optics programs used in the accelerator community. Particle physics simulation and analysis programs are made accessible on multiple platforms as Computational Modules in an intelligent graphic user interface shell developed with the S.P.A.R.C.-MP framework. The framework provides a sophisticated beamline object model and an extensive collection of GUI components in order to support a variety of computational codes within a single user interface shell. This provides an environment for the graphical setup of beamline models and execution of multiple optics programs from a single user-friendly interface, without requiring any specific knowledge of the various input formats and syntax required by the different computational programs.

2 GRAPHIC USER INTERFACE SHELL

The S.P.A.R.C.-MP framework provides a sophisticated beamline object model and a rich library of GUI components for the development of a unique user interface shell for particle accelerator related codes. Figure 1 illustrates the main document window for the Particle Beam Optics Laboratory (PBO Lab™) [1,2,3,4] which was developed with the S.P.A.R.C.-MP framework.

Beamlines are graphically constructed in the user interface by selecting and dragging beamline Piece icons from the Palette Bar and dropping them onto the Model Space. Individual Pieces or groups of Pieces may be selected for use in other beamline construction tasks, such as dragging a copy to the Work Space, converting between hierarchical and flat representations, creating Aliases or copying between different Documents. Selections from the Work Space may be inserted into the beamline by dragging the selection to an insertion point between beamline Pieces or to the end of the beamline model. Selections of Pieces may also be grouped as Sublines, encapsulating a series of accelerator elements into a single Piece in the beamline model. Sublines may contain nested Sublines, as well as individual Pieces. Sublines can be flattened to expand encapsulated Pieces back to individual icons in the beamline. However, individual Pieces within a Subline can be accessed without flattening the Subline.

Input parameters for beamline elements are accessed by double-clicking Piece icons in the Work Space or Model Space of the Document window. Figure 1 illustrates Piece Windows for the Initial Beam and Quadrupole element. Many beamline elements provide different parameter sets that can be used to specify inputs for the element. For example, a quadrupole may be minimally defined by length and quad coefficient value, or by length, aperture and pole tip magnetic field, or by length and magnetic field gradient. Automatic calculations are made between the different parameters sets, allowing the user to select the parameters for specifying an element independent of the native inputs used by specific simulations. The user interface indicates which parameters are used as native input for the

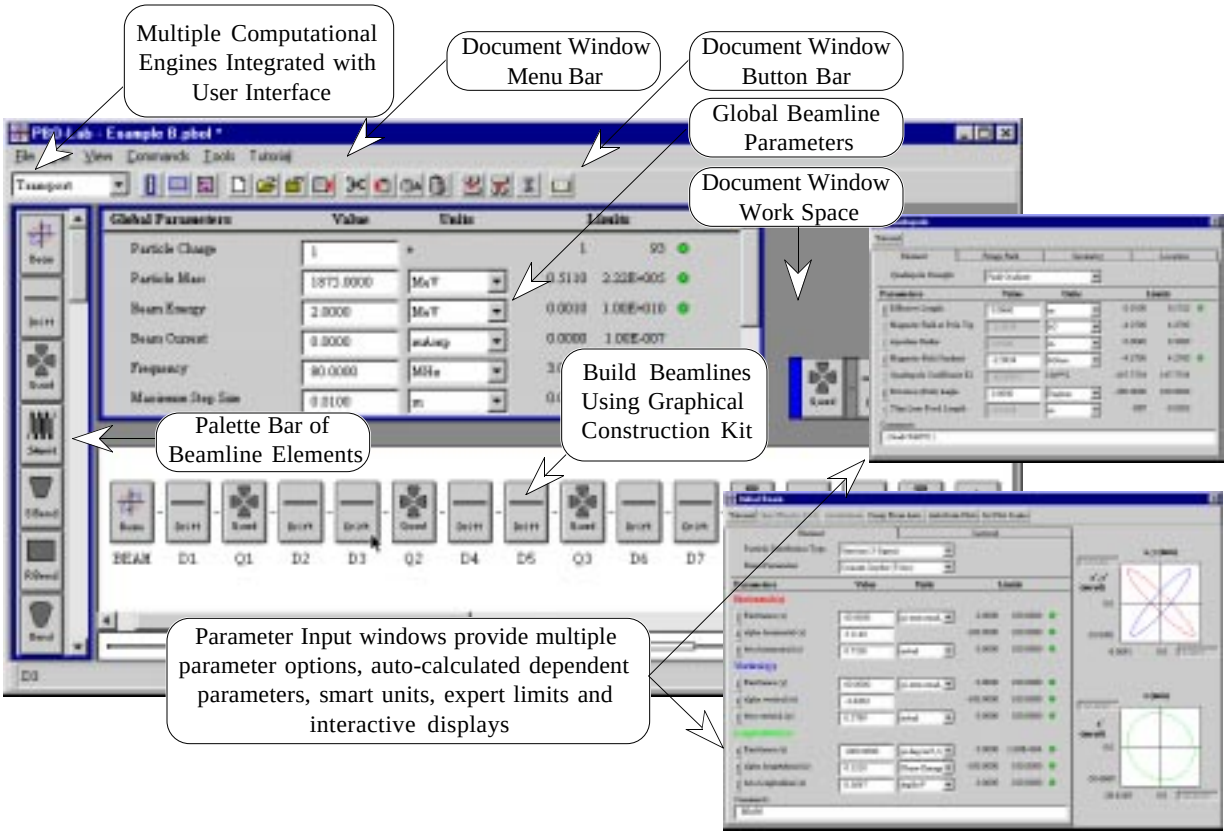

Figure 1. S.P.A.R.C.-MP based PBO Lab for Windows.

selected simulation. A variety of different units options may be selected for each parameter independent of the units required by the different simulations. Expert system type rules provide guidance for editing input parameters, and additional displays, such as effective focal lengths and phase space plots, provide useful feedback to the user.

The framework also provides data visualization components which are used to generate a variety of ellipse plots, scatter plots, line plots, bar graphs, etc. from the raw data generated by the different simulation applications. In addition, the framework architecture supports plug-in tools such as an interactive particle trajectory module, optimization algorithms, an external data interface and hypertext tutorials which interface with and enhance the functionality of installed simulations.

Central to the infrastructure of the S.P.A.R.C.-MP framework is the persistent beamline object model [5]. Fundamentally, the object model is the basis for describing accelerator beamlines at different levels of abstraction and providing the foundation for implementing a host of optics codes that use a variety of different beamline model descriptions. The beamline object model forms the basis for the interactive graphic functionality and the persistent beamline model description. It is also used to generate the native input required by the individual simulations supported in the framework. The beamline object model describes hierarchical, flat and mixed beamline representations. This model is especially suited to representing beamlines with a large number of repetitive components using Sublines and Aliases. Aliases can be created for any Piece or Subline in the beamline model. Aliases are implemented in the beamline object model with a persistent link to another Piece or Subline and are capable of storing deviations from the original Piece parameters without any duplication of redundant data. A hierarchical organization of the beamline, using Sublines in combination with the ability to create Aliases, allows the user to configure the beamline in any representation that suits their problem. A number of benefits can be realized with this model including more efficient problem setup, compact views of very large models and elimination of redundant data storage within highly symmetrical beamlines.

3 SIMULATION APPLICATION MODULES

Different optics programs are implemented as Application Modules in the S.P.A.R.C.-MP framework. A simulation code is selected with a pop-up menu in the Document window button bar as illustrated in Figure 1. This selection is referred to as the application context. The application context is used to provide the user with

various application-specific feedback such as the native inputs for an application. An Application Module has two primary components: an Interface Module and a Computational Module. The Interface Module is the application-specific graphic user interface and data interface code for implementing the input and output requirements of the application. The Computational Module is the computational physics code from the application (with the I/O functionality replaced by the Interface Module). The Computation Module is implemented as a dynamic link library (DLL) that is loaded at run time. Legacy FORTRAN codes, as well as the latest object-oriented codes, can be incorporated into the framework as independent simulation Application Modules. The core graphic user interface components and beamline object model are independent of the installed Simulations. However, application-dependent GUI components can be derived from the more abstract framework components to support any application-specific GUI requirements for the different simulations.

The user interface and data interface requirements for an Application Module are also implemented as independent application-specific components that leverage the functionality of the more abstract framework components. The Interface Modules generate different application-specific beamline descriptions (native input) for executing the Computational Modules, which are treated as black-box computational engines. The framework will support a variety of particle physics simulation and analysis codes such as TRANSPORT [6], TURTLE [7], TRACE 3-D [8] and MARYLIE [9]. Selected characteristics and uses for these applications are listed in Table 1. Additional particle optics applications, as well as system level and facility modeling codes, are also under development for integration with the S.P.A.R.C.-MP framework.

Table 1. Selected Simulation Application Codes.

Code	Selected Characteristics
TRANSPORT	Third-Order Matrix Code. Magnetic Optics, Fitting, Optimization, Aberation Control.
TURTLE	Third-Order Ray Tracing Code. Multi-Particle Simulator. Beam Aperture Control.
TRACE 3-D	First-Order Matrix and Space Charge. Magnetic, RF and Electrostatic Optics. Fitting, Optimization, Envelopes.
MARYLIE	Lie Algebraic Beam Transport. Linear and Non-Linear Optics. Multi-Turn Tracking, Fitting, Optimization.

4 FRAMEWORK ARCHITECTURE

The architecture that has been developed for the S.P.A.R.C.-MP framework allows modular software components to be added to, and removed from, the underlying user interface shell. This provides great flexibility for incorporating multiple optics programs as simulation Application Modules in the S.P.A.R.C.-MP framework. The development effort is greatly reduced by reusing verified software components and constructing new components based on more abstract software modules. The S.P.A.R.C.-MP framework provides an abstraction layer with an extensive collection of tested and verified components developed specifically for implementing accelerator design and analysis programs. Figure 2 illustrates the modular architecture that has been developed to support multiple Application Modules in the framework. The Interface Modules and Computational Modules are pictured in the upper right of the Figure, enclosed by a gray outline. The architecture provides the capability to determine what Application Modules are installed and to tailor the user interface for each particular application environment. Simulation Application Modules are registered with the shell so that the user interface can add application-specific menu items, tool bar buttons, tab panels, etc. where and when appropriate based on which simulations have been detected. Framework components and data interface objects can be dynamically modified to support the native requirements and capabilities of specific applications. Framework components that provide GUI features are shared between the different Application Modules. These include the main document window and beamline construction toolkit, as well as the underlying beamline object model, the palette bar of accelerator elements, piece windows with multiple parameter sets, smart units, expert limits and auto-calculated user feedback of dependent parameters. The framework also provides data visualization components that are used to plot data generated from the different simulations. Additional application-specific GUI and data interface requirements are implemented in the Interface Modules.

The modular architecture reduces complexity through abstraction and lends itself to the multiple platform approach adopted for the S.P.A.R.C.-MP framework. Figure 2 illustrates the multi-platform and multi-application aspects of the framework architecture as OS dependent and OS independent layers. GUI code is OS (operating system) dependent. OS independent code is ANSI compatible and independent of the platform and operating system because it contains no references to the native OS libraries. The OS dependent layer is subdivided further into application specific and application independent layers. The multi-platform capability of the S.P.A.R.C.-MP framework is derived from the use of different libraries, each of which encapsulates a specific

operating system. These libraries use a common application programming interface (API) for multiple operating systems. The libraries, which are a collection of basic functions utilizing OS native routines, provide a means of interacting with the platform-specific Operating System, while maintaining the native "look and feel" of the graphic interfaces associated with the target platform. With the use of these libraries, applications built with the S.P.A.R.C.-MP framework are made platform-independent, since the framework code calls the API

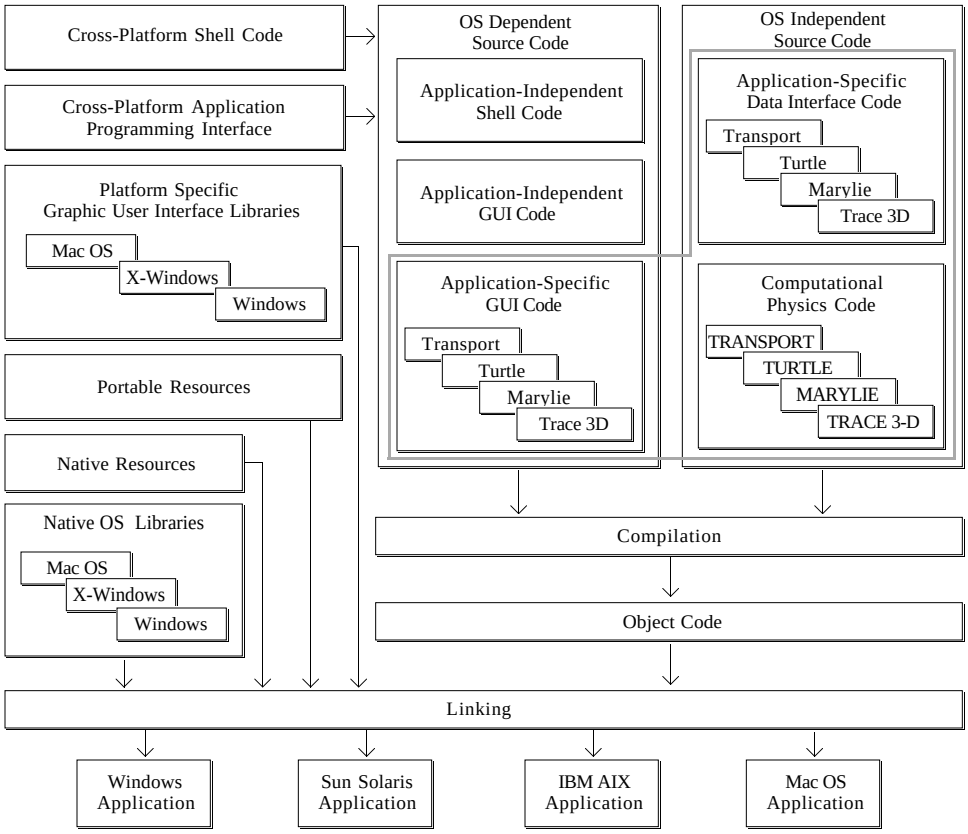

Figure 2. S.P.A.R.C.-MP Multi-Platform Framework Architecture.

routines, not platform specific routines, directly. This approach provides an abstraction layer from the different operating systems and yields a single version of the application source code for all platforms as opposed to developing and maintaining separate source codes for each platform. A platform-specific executable application is created by linking the application code with the appropriate library for the intended platform. Using this approach, the software may be easily ported to a new operating system by utilizing an interface library for that system. The API libraries are implemented as a layer on top of the native toolkits for each of the target platforms. This ensures a native look-and-feel, low overhead and still provides direct access to native toolkits if required. This multi-platform approach has been successfully implemented on personal computers running Microsoft Windows™ 95, 98 and NT, Apple's Mac OS™ for the PowerPC, as well as IBM and Sun work stations running AIX[a] or Solaris™.

5 SUMMARY

The S.P.A.R.C.-MP framework is a multi-platform object-oriented application development environment specifically designed for creating applications which integrate a variety of particle optics and accelerator design codes under a single graphic user interface shell. The framework provides a sophisticated beamline object model and an extensive collection of GUI components, supporting a variety of particle physics simulation and analysis codes such as TRANSPORT, TURTLE, TRACE 3-D and MARYLIE. The Operating Systems supported by the S.P.A.R.C.-MP application framework and graphic user interface include Windows, Mac OS, Solaris and AIX. Platform-specific GUI components are implemented for multiple platforms with a single source platform-independent project. The framework multi-platform GUI components are developed by utilizing a single application programming interface (API) instead of the native software development kits (SDK) for each different platform. The API is implemented for a variety

of computer platforms as platform-specific libraries that interface with the native SDK for each platform. An executable application is built from the single source project by compiling and linking the code on the target platform with the appropriate API. Physics applications are integrated into the framework by separating the computational physics code, which is implemented as a platform independent Computational Module, from the I/O requirements, which are replaced with an Application Module developed with the multi-platform GUI framework components. The S.P.A.R.C.-MP framework is significant not only for its multi-platform capabilities but also for its open and expandable development environment. This environment supports the implementation of multiple optics codes under a single graphic user interface, improving the accessibility and ease of use for a variety of programs used in the accelerator community.

6 ACKNOWLEDGEMENTS

Portions of this work has been supported by the U. S. Department of Energy under SBIR grant number DE-FG03-95ER81975.

7 REFERENCES

[1] G. H. Gillespie, B. W. Hill, N. A. Brown, H. Martono and D. C. Carey, "The Particle Beam Optics Interactive Computer Laboratory," AIP Conference Proceedings 391, 264-269 (1996).

[2] G. H. Gillespie, B. W. Hill, H. Martono, J. M. Moore, N. A. Brown, M. C. Lampel and R. C. Babcock, "The Particle Beam Optics Interactive Computer Laboratory for Personal Computers and Workstations," to be published in the proceedings of the 1997 Particle Accelerator Conference, 3 pages.

[3] N. A. Brown, G. H. Gillespie, B. W. Hill, M. C. Lampel, H. Martono, and J. M. Moore, "The Particle Beam Optics Laboratory (PBO Lab™): A New Education and Training Aid," to be published in the proceedings of the Sixth European Particle Accelerator Conference, 3 pages.

[4] The PBO-Lab software is available from AccelSoft Inc., San Diego, California, U.S.A., www.ghga.com/accelsoft.

[5] B. W. Hill, H. Martono and J. S. Gillespie, "An Object Model for Beamline Descriptions," AIP Conference Proceedings 391, 361-365 (1996).

[6] D. C. Carey, K. L. Brown and F. Rothacker, "Third-Order TRANSPORT - A Computer Program for Designing Charged Particle Beam Transport Systems," Stanford Linear Accelerator Center Report No. SLAC-R-95-462, 295 pages (1995).

[7] D. C. Carey, "TURTLE (Trace Unlimited Rays Through Lumped Elements) A Computer Program for Simulating Charged Particle Beam Transport Systems," Fermi National Accelerator Laboratory Report No. NAL-64, 45 pages (1978).

[8] K. Crandall and D. Rusthoi, "TRACE 3-D Documentation," third edition, Los Alamos National Laboratory Rep. LA-UR-97-886, 106 pages (1997).

[9] A. J. Dragt, D. R. Douglas, J. van Zeijts, F. Neri, C. T. Mottershead, R. D. Ryne, E. Forest, L M. Healy and P. Schutt, "MARYLIE 3.0 User's Manual: A Program for Charged Particle Beam Transport Based on Lie Algebra Methods," Draft, 543 pages (1998).