An Alternate Paradigm for High-Level Application Development

D. Douglas

Abstract

We discuss an alternative to traditional paradigms under which high-level accelerator applications are developed. In this revised view, emphasis is placed on use of pre-existing tools, speed of development, accessibility to the user, and ease of modification, rather than traditional fiduciary parameters such as generality, robustness, security, and completeness. Applications developed under this paradigm can be easily generated, readily modified, and fully accessible to users with a minimum of training and installation or maintenance overhead. A sample application is described.

Traditional Paradigms

High-level accelerator applications have been generated under numerous organizational models, most of which are well described by a “client/vendor” relationship. In this paradigm, a “client” (typically, an intended user of the high level application) contracts with a “vendor” (typically, a software organization of some description) to provide an application that meets (nominally) clearly defined requirements. The client provides the requirements, and the vendor is more or less free to construct the application in any fashion consistent with the defined criteria.

In this paradigm, the user is insulated from the process of software generation. For this reason, vendors place considerable emphasis on generality, robustness, security, and completeness as key features of the product. Such an outlook helps ensure that all client requirements - stated, inferred, or otherwise perceived - are met. This bias also tends to fuel enthusiasm for the use of object-orient programming tools and techniques in application development. Such methods, with their implementations of polymorphism and inheritance, enable vendors to write general and robust applications with inherently secure structures that allow clients access to certain defined “public” information, but protect other information in “private” data structures available to the software developer only.

Implementation time and fault sensitivities are significantly reduced through such methods. It is for vendors, therefore, a most cost-effective paradigm

under which to develop applications. It is however not without cost to the client, who becomes fully dependent on the vendor to supply a product meeting the letter but not necessarily the spirit of the initial requirements. Moreover, the user typically cannot implement modifications and upgrades to the application. In addition, the vendor, not the user, normally imposes (or at least executes) the criteria of generality, robustness, and completeness; the resulting product can thus fall prey to the "better is the enemy of good enough" syndrome, in which development continues well beyond the users nominal requirements.

From the perspective of a user *operating* an application in an *evolving* environment, the client/vendor paradigm may therefore have certain deficiencies. The overhead of meeting changing or evolving requirements may be high, as might the cost of upgrading, modifying, or maintaining a complex application. The complexity required for applications to meet criteria of sufficient generality, robustness, completeness, and security may be high, with associated high development costs. We therefore consider an alternative paradigm, in which emphasis is not placed on generality, robustness, and security, but rather on development ease and speed, modifiability, and user accessibility. Applications developed under such a paradigm can be viewed as "just in time" products, created as "near-run-time" codes tailored to a specific purpose and treated as "disposable".

An Alternate Paradigm

As an alternative to the client/vendor paradigm, we now consider a development model that produces "user-developed, near-run-time" applications. In this model, the user is provided a *development environment*, not an application. (This environment could in fact be procured through a client/vendor relationship!) Using an appropriate development environment, the user would, at or near run time, develop an application specifically tailored for use in a single instance. In this scenario, emphasis is placed on use of pre-existing tools (analogous, but not entirely identical, to inheritance), ease and speed of development and modification, and user accessibility, rather than robustness, generality, and completeness. Given the assumed ease of development under such a paradigm, the traditional requirements are not necessary to meet – if the software has a deficiency, it is modified at run time rather than off line!

In the context of accelerator operation and control, this model could be implemented by providing the user a development environment and data access tools. With appropriate development methods, the user could capture needed data, rapidly construct tools for its analysis, and determine outcomes

that could then be applied to the accelerator. Examples of appropriate analysis environments include the TCL/TK toolkit, analysis packages such as MATLAB and MATHCAD, and comprehensive computational systems such as MATHEMATICA. Application development at or near run time then becomes a "just-in-time" activity, with the application a commodity or component of the accelerator operations product that need not be developed far in advance and stockpiled, but instead simply provided when needed through proper pre-positioning of resources.

Key to an understanding of this paradigm is a recognition that nearly all high level applications use only simple, readily captured data sets, perform processing of at most modest complexity, and produce results that are limited in scope and which do not require extensive implementation to realize within the accelerator itself. "Scope creep" in application development is not driven by algorithmic requirements but rather, first, by robustness and completeness requirements that impose on the application exception handling constraints that are difficult to meet in total generality, and second, from attempts to provide a highly polished user interface pertinent to all operational contexts. Given that the core computations of most applications are quit simple, it is reasonable to investigate paradigms that may effectively and very rapidly tailor applications to the situation at hand, rather than attempt to meet all possible combinations of events and exceptions.

In the following section, we present a high-level application typical of those in use on many accelerators. Our bias being beam transport issues, we have considered the problem of rapid optics code development. In its simplest form, a beam-transport program is simply a tool to generate and manipulate tables of numbers; in the most general case, such codes are information systems describing accelerators. In all cases, the programs use (at least for linear modeling) simple algorithms – any computational complexity resides in user interfaces and data structures. Note, however, that the most desirable features of such programs – graphical input and output, data organization, and algorithms for data manipulation – are provided by commercially available spreadsheet programs. We have therefore constructed a machine model (for the Jefferson Lab IR FEL driver) using Microsoft Excel as the underlying computational engine. The resulting application is discussed below.

Example – An Excel-Based Optics Application

Microsoft Excel is a well-known spreadsheet with extensive analysis and application development features. In fact, "... it is important to understand that Excel is not merely a spreadsheet. Excel is also a powerful object library

that includes over one hundred advanced data analysis objects. With VBA [Visual Basic for Applications, an embedded dialect of Microsoft Visual Basic], developers can piece together Excel's objects to create power information systems... There are currently thousands of Excel-based information systems in use in corporations throughout the world [1]."

Given that information system requirements of a large commercial organization (such as a bank) can readily exceed those of a modest accelerator, we have investigated the applicability of Excel as a development tool for beam optics modeling. The following observations may be made.

- It is quite easy to generate rudimentary (linear) optics modeling capabilities within Excel.
- Using Visual Basic for Applications (VBA), it is straightforward to extend Excel to incorporate any modeling capabilities provided using typical high level languages such as FORTRAN, C, or C++.
- Excel serves as a simple, intuitive, and robust user interface for optics calculations. Little training overhead arises in the use of an Excel-based application.
- Excel, through VBA and ActiveX technology, provides network, WWW, and database connectivity. Excel based applications therefore can be made immediately available to multiple simultaneous users (for example, as shared workbooks). Robust protection, security, and configuration control features are (as one might expect, for a commercially implemented product) included.

The development cycle for such applications is quite short. The model described here was, in fact, generated in only a few hours (spread over several weeks), generally during the author's off-hours (typically while watching Monday night football!). Incorporation of features beyond basic matrix manipulations required from a few minutes (for emittance data analysis) to a few hours (to implement the CEBAF cavity matrix). The cost effectiveness of this development paradigm is thus immediately apparent.

Sample Excel 97 workbooks describing the IR FEL driver are available from the author, both via WWW distribution and through the Jefferson Lab LAN. The examples given below are drawn from a 5 December 1997 model of the initial setup of the transport from injection point to the first light dump.

Features of the Sample Application

Data Structures – The Excel-based FEL driver model was built using a standard Excel workbook. Within the workbook, individual worksheets were

used to construct data tables defining and describing a beam line. At the present time, the following six worksheets are utilized:

elements – contains a sequential list of the elements comprising the beam line, and definitions thereof.

matrices – contains a sequential list of matrix elements for individual beam line elements.

products – contains an accumulating product of the transfer matrix through the beam line.

betas – contains a table of beam envelopes, dispersion, and phase advances.

orbits – contains a table of ray-trace data, giving the image of an initial propagated ray under the transformations in the **products** worksheet.

cavitydata – contains electric field information required for computation of CEBAF cavity transfer matrices.

The application also contains (in a manner transparent to the user) a VBA module defining an Excel function evaluating CEBAF cavity matrix elements. Additional VBA forms can be used to define control buttons, sliders, and charts.

Detailed Content of Worksheets

The **elements** worksheet lists and defines elements in the beam line workbook. Each row of the worksheet gives an element name and the parameters defining the (beam transport matrix of) the element. Parametric definitions of elements correspond to those used in TRANSPORT [2]. Typical element definitions are shown in Figure 1. In addition, the **elements** worksheet contains a beam definition (species and kinetic energy at locations where magnetic fields are to be computed, from which momentum and rigidity are evaluated), a chart of betatron functions and dispersions (See Figure 2), control knobs for various magnets (see Figure 3), and definitions for various χ^2 functions to be utilized in fitting operations (see the following “Functionality” discussion).

matrix types and input format						
DRIFT						
name	length					
QUAD						
name	length	k1				
EDGE						
name	0 (length)	rho	angle	gap	K1	K2
BEND						
name	length	angle	index			
examples						
drift1	1					
qf	0.15	5				
sb1	0	2	10	0.0254	0.3	1
b1	1.0472	30	0			
eb1	0	2	10	0.0254	0.3	1

Figure 1: **elements** worksheet element definition syntax.

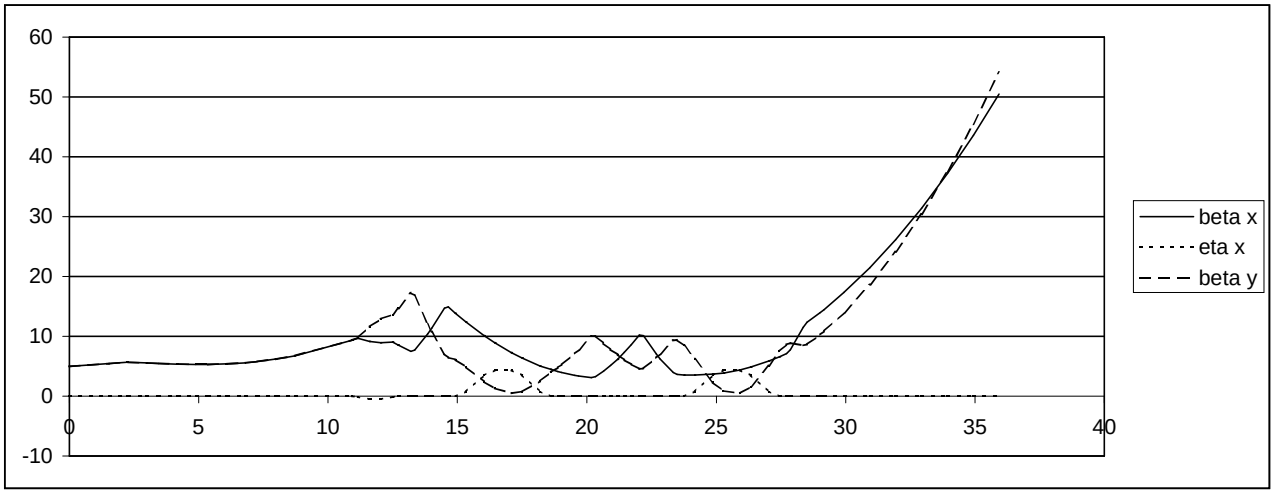

Figure 2: Sample chart presentation of propagated beam envelopes and dispersions (for IR FEL first-light transport).

		qm21	qm21	qm23	qm24	qm25	qm26
original DIMAD ans.		-2.576283	7.067754	-1.193487	8.483344	-10	3.5
		0	1.5	-2	-3	0	4
		qm11	qm12	qm13	qm14	qm15	qm16
original DIMAD ans.		-4.777481	3.658347	0.824704	-6.91892	12.24123	-9.310514
		-3.1	0	2.5	0	0	-3

Figure 3: Illustration of magnet control sliders.

The **matrices** worksheet contains a representation of the linear transfer matrix for each beam line element. Each 6 x 6 transfer matrix is stored as a 1 x 36 array on a row corresponding to that row in the **elements** worksheet defining the beam line element in question. The matrix elements are, for magnetic elements (and drifts) evaluated using standard Excel functions to evaluate the formulae given in TRANSPORT documentation [3] using values pulled from the **elements** worksheet. The elements for CEBAF cavity matrices are evaluated using a VBA port of the DIMAD [4] CEBAF cavity matrix routine, again using values pulled from the **elements** worksheet.

The **products** worksheet contains a cumulative product of the transfer map. Each row gives the matrix from the start of the beam line to the end of the beam line element on the corresponding row of the **elements** worksheet. The data is stored as a 1 x 36 array, just as on the **matrices** worksheet.

The **betas** worksheet contains a table of propagated beam envelope functions, dispersions, and phase advances. Each worksheet row gives the image at the end of the beam line element (on the corresponding row of the **elements** worksheet) of beam envelopes propagated (using the matrix on the corresponding line of the **products** worksheet) from initial conditions specified at the start of the beam line. The data in this worksheet are displayed in the chart presented on the **elements** worksheet (Figure 1). The worksheet can be readily modified to present beam spot sizes and evaluate χ^2 functions for fitting of beam-line parameters and/or emittance data reduction (see the "Functionality" section, below).

The **orbits** worksheet contains a table of ray-trace data, giving the image of an initial propagated ray (with local steering) under the transformations in the **products** worksheet. This is presented in tabular format and graphically (see Figure 4). The worksheet data present the initial condition being traced, and the image (after each element on the corresponding line of the **elements** worksheet) of that ray under the matrix on the corresponding line of the **products** worksheet. This worksheet contains a table of ray-trace data comprising the images of an initial ray (with intermediate steering) under the transformations in the **products** worksheet. BPM data can be presented using this worksheet and is readily incorporated into the graphical presentation (again, see Figure 4). The intermediate steering is provided by entries for impulsive (thin lens) kicks; these can be used to model error effects and, together with easily defined χ^2 functions,

evaluate steering needed to provide orbit correction (see "Functionality", below).

The final worksheet, **cavitydata**, contains electric field information required for computation of CEBAF cavity transfer matrices. It consists of two 1 x 1401 element arrays: cavitydata!a1:a1401 is the longitudinal electric field, and cavitydata!b1:b1401 its derivative, every 0.5 mm through the 0.7 m length of a cavity. The data are derived from a cosine series expansion of SUPERFISH data and are passed to an Excel routine evaluating cavity transfer matrix elements.

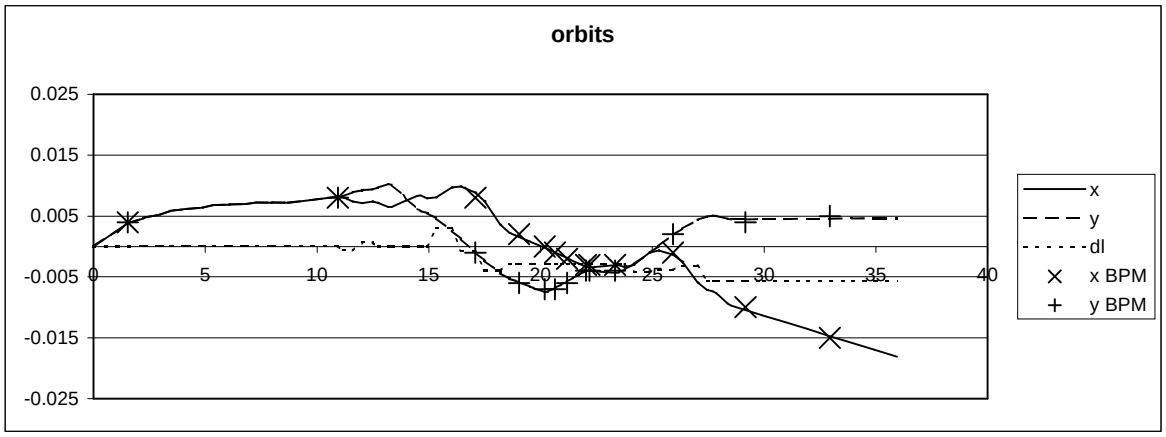

Figure 4: Orbit presentation by the chart in the **orbits** worksheet.

Functionality – Excel provides tools allowing numerous data analysis and reduction operations to be quickly programmed. One such tool is "solver", a nonlinear optimization package with which any cell of a workbook can be fit to a desired target value by varying any set of cells. The user can define a fit function in the target cell, and activate solver to minimize the parametric fit. In this fashion, a number of optics-related optimization problems can be readily addressed. The sample application has been used in this manner in the following contexts:

Fitting of matrix, beam, and lattice properties – using solver and defining appropriate χ^2 functions, matrix, beam and lattice properties can be fit by varying beam-line element parameters. Such optimization processes have been used to prepare IR FEL driver transport tunings (perform "matching") for various wiggler operating configurations [5]. Beam envelopes, dispersions, momentum compactions, phase advances, and other transport parameters at any location along the beam line may in this manner be optimized as a function of any set of available beam line element parameters. By extending the model to include beam line layout (see "Upgrades" section, below), the geometry

of a transport system can be coupled to, and optimized simultaneously with, beam performance parameters.

Emittance data reduction – use of solver, together with appropriately defined χ^2 functions on the **betas** worksheet allows reduction of spot size measurement data to evaluate beam emittance and envelope function values. The resulting phase space information can be used as the basis for matching computations of the type described above.

Error analysis and orbit correction – use of solver, together with appropriately defined χ^2 functions on the **orbits** worksheet, allows analysis of beam line alignment and powering errors. By including beam position data in the fit function and allowing solver to vary angular impulses at the location of correction magnets, orbit correction solutions can be generated.

Excel also includes network connectivity. At the present time, rudimentary networking has been demonstrated on the Web using Excel's WWW access features. Ongoing work (see the discussion of "Upgrades", below) will explore use of shared workbooks for multiple simultaneous users, implementing ActiveX technology for network and WWW connectivity, and potential tie-in to EPICS data transfer standards.

Upgrades - Several upgrades to the above Excel model can, at least in principle, be implemented. Some of the more obvious ones follow:

Improved model accuracy – The model described above is linear. The accuracy can be improved by increasing the **matrices** worksheet array size from 1 x 36 to 1 x (6 x 27) (for second order matrices) or to 1 x 209 (for third order modeling using Lie transforms). Each of these improvements will require some recoding, for example, of the algorithms on the **products** and **orbits** worksheets.

Beam line layouts – As the **elements** worksheet provides complete element definitions, a **layouts** worksheet could be easily included. This would provide a table of coordinates locating entry or exit points of the design orbit through various beam-line elements.

GUI – A graphical user interface can be readily developed using VBA forms and tools provided with Microsoft Office 97. This, in fact, is the topic of much of Reference [1]. As noted above (see Figure 3), knobs of various types can be easily added to various worksheets; different

widgets could be added to charts to represent beam-line elements or give schematic beam-line layouts.

Quantum excitation formulae – the **betas** worksheet can be readily extended to evaluate quantum excitation formulae such as those given by Sands [6].

Download file generation – a prototypical download file generator is under development. This application comprises a worksheet in a beam-line workbook; magnet excitation values or set points are evaluated through links to beam line operating conditions described by the workbook model. Updates to the model are then automatically propagated to updates in the download files. At present, the most efficient means of data transfer to EPICS is under study. A simple method is to save the worksheet as a tab-delimited table and use it as a .snap file. More elegant solutions are discussed below.

Communications – The above discussion alludes to our desire to establish links between this application and other applications in use in or around accelerators. Various methods to achieve this goal are available using existing Excel tools. These include:

- communicating with ODBC compliant databases
- use of ActiveX technology to access objects in applications external to Excel, and
- using Web queries to communicate with applications across the Internet.

Determination of the optimal approach is ongoing.

Conclusions

We have presented an alternative to traditional paradigms under which accelerator-related high-level applications are developed. An example application developed under the alternative “user-developed, near run-time” paradigm was presented and discussed. This application is in use for IR FEL driver commissioning.

References

[1] Wells, E. and S. Harshbarger, “Microsoft Excel 97 Developers Handbook,” p. xxiii, Microsoft Press, Redmond, WA (1997).

[2] Brown, K.L., F. Rothacker, D.C. Carey, and Ch. Iselin, “TRANSPORT A Computer Program for Designing Charged Particle Beam Transport Systems”, SLAC Report SLAC-91, Rev. 2, UC-28 (I/A), May 1977.

[3] *ibid*.

[4] Servranckx, R.V., K.L. Brown, L. Schachinger, and D. Douglas, “Users Guide to the Program DIMAD”, SLAC Report SLAC-285, UC-28 (A), May 1985.

[5] Douglas, D., “Matching Solutions for Various Wiggler Operating Configurations”, JLAB-TN-97-040, 27 October 1997.

[6] Sands, M., “The Physics of Electron Storage Rings An Introduction”, SLAC Report SLAC-121, UC-28 (ACC), November 1970; Sands, M., “Summary of Radiation – Related Equations”, SLAC Report RLA-27, July 1972; Sands, M., “Emittance Growth From Radiation Fluctuations”, SLAC Report SLAC/AP-47, December 1985 (AP).