Designing a Graphical User Interface for DELTA: Some considerations

Mauro J. Cavalcanti

Departamento de Biologia Geral, Universidade Santa Ursula,
Rua Fernando Ferrari, 75, Botafogo, 22231-040, Rio de Janeiro, RJ, Brasil
E-mail: maurobio@omega.lncc.br

Since its introduction almost two decades ago, the DELTA system has gained increasing acceptance by taxonomists all over the world, with several large databases in zoology, botany and virology already prepared plus many others under development. While the DELTA format provides an excellent way of describing taxonomic data — up to point of being adopted by the International Working Group on Taxonomic Databases for Plant Sciences (TDWG) as a standard for data encoding and exchange —the programs that comprise the DELTA package still reflect its origins in the mainframe world. Frequent criticisms have been the lack of an integrated, user-friendly interface for running the programs, the lack of an adequate data entry/editing system, and the unavailability of DELTA on computer platforms other than DOS/Windows PCs.

Over the last two years or so, I have been involved in developing user-friendly tools for DELTA — of which my DIANA shells (available for MS-DOS and Windows) were the first products — and I have dwelled deeply on the issues of user interface design, data models, search algorithms, data structures and the relevance of all this to the development of reliable biological software. Perhaps more importantly, I have also undertaken several experiments with user interfaces and data structures more adequate to the presentation and representation (sensu Diederich & Milton, 1993 — anyone seriously interested in developing GUI's for biological software would be well advised to read that paper) of biological data. Here I intend to present some of the preliminary, tentative conclusions I have reached in the course of these experiments, in the hope of obtaining some useful feedback from DELTA users and developers.

A key issue: portability

I start from the principle that a Graphical User Interface (GUI) is the most desirable kind of environment for a computer user, by making any system easier to grasp by the novice and faster to use by the more experienced user — the phenomenal success of the Microsoft Windows operating environment is an indication that this is the direction to which the software industry points.

GUI's are available on the three main computer platforms in use today — PCs, Macs, and UNIX workstations. The DELTA programs that take advantage of a GUI (ie. INTKEY and INTIMATE), however, are only available on DOS/ Windows PCs (or on other platforms under DOS emulation). So, perhaps more than offering an intelligent, editing tool (more on this below), the DELTA system should be as portable as possible, not only because of the large number of both PC and Mac users, but also to face the widespread (and increasing) availability of UNIX workstations. WWW browsers like Mosaic and Netscape offer a relevant analogy here, since they are both available on the three main platforms above mentioned.

However, developing a GUI-based program is not an easy task, even for experienced programmers — the difficulties of writing such programs are often underestimated. However, there are now software tools that not only allow simplification of the process of software development for each of the widely used GUI platforms: MS-Windows, Macintosh, UNIX/XWindows, but also make the programs portable between them.

One of these tools is VIBRANT, a high-level, multi-platform user interface development library written in C by Jonathan Kans (kans@ray.nlm.nih.gov), Information Engineering Branch, National Center for Biotechnology Information, NLM, NIH, Bethesda, USA. It is distributed as part of the NCBI Software Development Toolkit. VIBRANT acts as an intermediary between an application and the underlying windowing system toolkit. It is available for free and comes with full source codes. With it, a GUI-based application can be written that runs without modification on any of the mentioned platforms — you just have to re-compile them on the target machine, using the native C compiler.

Commercial software libraries based on the C language also exist that perform similar functions, such as Zinc and zApp, but these are usually very expensive and do not include source code.

Therefore, not only the DELTA programs themselves, but as a matter of fact, all DELTA "third-party add-ons" (like Lander's DMSWIN, Gouda's TAXASOFT, or my own DIANA software shell for DELTA) should be greatly improved if (re-)written using such tools, by benefiting from portability and, consequently, from a larger user base.

A user-friendly editing tool

Another request from many DELTA users is the availability of a user-friendly, specialized data editor, that should release users from the cumbersome and error-prone creation and maintenance of DELTA datasets using an ordinary text editor or word processor.

Early attempts to provide such a specialized DELTA editor were those of Pankhurst's DEDIT and Gouda's DDATA (distributed as part of his TAXASOFT package). However, none of these editors are GUI-based, which may become a major drawback to many users of modern operating systems and application programs. Worse, since MS-Windows and other GUI environments have highly standard interfaces, with common control objects (dialog boxes, pulldown menus, etc.), the authors of these editors will have to write entirely new applications when porting their programs to such an environment.

My specialized "DELTA Coding System" (DELCODE) is a first attempt at providing a GUI-based highly interactive environment for the editing of DELTA datasets. A DOS version is finished and a Windows version is well advanced. However, it is not a production system, but rather a teaching tool (that is the way I am using it). It is the result of one of the experiments I mentioned above, and is itself experimental. DELCODE does not intend to be a rival of the undoubtedly powerful, Windows-based, DELTA editor currently under development by the DELTA team. Its primary purpose is to test algorithms, data structures, and user interface ideas. It features pulldown menus, dialog boxes, and full mouse support, but the DOS version is obviously not a full GUI-based application and detaches at several points from the "classical" CUA/SAA (Berry, 1988) standards.

Up to date, all DELTA editors implement what we might call a "form fill-in" interface (see Diederich and Milton, 1993), with which the user enters data, usually from the keyboard, in specific fields on pre-built forms. This system is also used by the Windows versions of some DELTA programs (namely, INTKEY4 and INTIMATE), to get information from the user. However, as one of the DIANA users has quite rightly pointed out, character-by-taxon data matrices are the logical way to manage taxonomic data (Coddington, pers. comm.). A matrix is convenient, already known to most systematists, and is the accepted "metaphor" for comparative data on taxa. Visually, a user can chase homologies across taxa, or taxa through morphological space. So, a GUI-based DELTA data entry/editing system should be able to provide the user with some sort of matrix-oriented editor.

One such a matrix editor for taxonomic data already exists and might provide a relevant paradigm for further development of a user-friendly, GUI-based, biological data entry and editing system. That is the editor of Maddison & Maddison MacClade's interactive character analysis program, available only for the Macintosh computer. A good summary description of this editor (and of MacClade itself) — for those who do not have access to a Mac or do not have a copy of MacClade — can be found in Maddison and Maddison (1989). Indeed, MacClade's editor behaves somewhat as an interactive "character designer", which helps the user "to think about the biology behind the data" (Maddison and Maddison, op. cit.). This is a very innovative and promising approach that should be pursued further in the development of a GUI-based DELTA editor.

Another of my experiments with GUI software for DELTA was the development of a matrix-oriented editor for DELTA. A crude prototype was implemented in Visual BASIC (that provides a rather limited spreadsheet-like custom control), but it has not yet been developed further. Parts of it have been incorporated into the code of the LORIS browser program for DELTA databases, currently under development.

Reusable components

In recent years, the demands placed on software for management of taxonomic data have increased dramatically in response to an ever-increasing concern that such data are the key to biodiversity conservation and sustainable use of biological resources. As a result, the very few software-developers engaged in the field of biodiversity information management (including the members of the DELTA development team) are being highly pressed to satisfy all user needs in less time and with more efficiency. As with other fields of human activity, biodiversity data management is also faced with a "software crisis" (Cox, 1986; Gibbs, 1994), and perhaps software engineering should also be enlisted as a "crisis discipline", along with conservation biology and cancer biology (Soulé, 1985).

Such a crisis in the development of biological software means that not only are better and portable software development tools (as the VIBRANT interface library above mentioned, or the more expensive commercial ones) needed, but also that a biodiversity information management package far more programmable by the end-user should be developed. The access to preprogrammed subroutines would allow systematists with limited programming experience to develop their own specialized programs for biodiversity data management (eg., in her/his own national language). In the future, it might be more fruitful for DELTA developers to put their efforts in this general direction, rather than trying to anticipate and meet (with scarce resources) the precise needs of the next generation of computer taxonomists.

Along this line of thinking, I have developed DELTA Library, a general-purpose library of routines for reading text files in DELTA format, designed to work with any language that supports Windows DLLs (Dynamic Link Libraries). They allow a programmer using any language that supports DLL calling, such as Borland Pascal/Turbo Pascal for Windows, C/C++, and Visual BASIC, to write an application that can access DELTA datasets without effort, with just one line of code for each routine's call. I have already been able to sucessfully integrate them with LORIS, an experimental "DELTA database browser" I am currently writing in Visual BASIC for Windows, and these routines are also being used as the back end of the Windows version of my DELCODE editor for DELTA.

Anyway, such libraries of pre-programmed routines would be only the first step towards actual "off-the-shelf" reusable software components — best exemplified today by Visual BASIC's custom controls (VBX's) — that may be used to build real applications in less time and with less effort (Udell, 1994). These and many other developments of the current software industry are likely to yield most rewarding results in the design and implementation of reliable software to successfuly face the data management issues posed by the the sheer diversity of the biota they are intended to help preserve.

References

Berry, R.E. (1988). Common User Access - a consistent and usable human-computer interface for the SAA environments. *IBM Systems Journal* 27: 281-300.

Cox, B.J. (1986). *Object-Oriented Programming - An Evolutionary Approach*. Addison- Wesley Publishing Company.

Diederich, J. and J. Milton. (1993). Expert workstations: a tools-based approach. In: *Advances in Computer Methods for Systematic Biology: Artificial Intelligence, Databases, Computer Vision*, R. Fortuner (Ed.). The Johns Hopkins University Press, Baltimore, Maryland, pp. 103-124.

Gibbs, W.W. (1994). Software's chronic crisis. *Scientific American* 271: 72-81.

Maddison, W.P. and D.R. Maddison. (1989). Interactive analysis of phylogeny and character evolution using the computer program MacClade. *Folia Primatologica* 53: 190-202.

Soulé, M.E. (1985). What is conservation biology? *Bioscience* 35: 727-734.

Udell, J. (1994). Componentware. *Byte* 19: 46-56.