The following paper was originally published in the
Proceedings of the Fourth USENIX Tcl/Tk Workshop
Monterey, CA, July 10-13, 1996

A Clinical Neurophysiology Information System based on Tcl/Tk

Martin B. Andrews and Richard C. Burgess
The Cleveland Clinic Foundation
Cleveland, OH 44195

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

A Clinical Neurophysiology Information System based on Tcl/Tk

Martin B. Andrews and Richard C. Burgess

Section of Neurological Computing
Department of Neurology
The Cleveland Clinic Foundation
9500 Euclid Ave
Cleveland, OH 44195
andrewm@ccfadm.eeg.ccf.org

Abstract

The Department of Neurology at the Cleveland Clinic has replaced a restricted patient information system with a new one based on Tcl/Tk. Tcl/Tk along with several novel extensions provides the base for creating portable client-server database applications. These extensions and some of the clinical applications are presented.

1. Introduction

The Cleveland Clinic is a large multi-specialty medical center with a high volume of both inpatient and outpatient evaluations. One of the largest departments, Neurology, is a leader in the care of patients with epilepsy and other neurological disorders. Along with physical examinations, imaging studies and laboratory tests patients are evaluated with neurophysiological tests like EEG (brain wave tests) and PSG (sleep tests). Tests are performed using computer-based acquistion and review systems developed by the Section of Neurological Computing. This section is reponsible for the information needs of the department, and provides networked databases to physicians, nurses, technologists and secretaries. These systems and software products have also been made available for use at other institutions through the Vangard subsidiary.

The Neurology department's patient information system, EBase, was integral to daily operations and research studies. It was also unreliable, difficult to use and even more difficult to program. A major overhaul of the code was enacted to remove bugs and hardware dependencies. The resulting system, EBase2, was more reliable but development was still painfully slow. Particularly hard to develop were the highly customized user interactions demanded by our users. To overcome this problem a radical redesign of EBase was initiated. Informix-4GL, the proprietary programming language used in earlier revisions, was dumped. The requirements for the new EBase3 were as follows:

- Rapid Development - The burgeoning development of new clinical tests, a desire for faster reporting of test results and a need to reduce costs drive an ever growing demand for information to be maintained by EBase. In addition, vague user requirements necessitate quick prototyping. Programmers must be able to quickly extend EBase.

- Ease of Use - Our users demand an intuitive and stream-lined user interface. These somewhat contradictory needs can be met with flexible user interface components that are customized for particular groups of users. Even further flexibility is required if other institutions are to use EBase.

- Portability - Departmental users access EBase from Unix workstations, PCs running Microsoft Windows, and Wyse 60 terminals attached to a Unix server. All of these devices must be supported. For EBase to be used at other institutions it must be portable to even more client devices, and even a variety of server platforms.

- Simple API - The section of neurocomputing also develops software to acquire and review neurology test data. A simple API for accessing the EBase database is desired for use in these systems.

- Inexpensive Licensing - The cost of development tools was an important issue because of our own budget limitations and, more importantly, the need to provide a competitively priced package to other institutions. Even tools without run-time fees

generally charge per target platform - a problem considering the number of targets to be supported.

The resulting design for EBase3 uses Tcl to help meet these goals. The core of EBase3 is an extended Tcl interpreter called Clavier. Clavier controls all access to the patient database. It enforces data integrity rules and hides implementation details of the database schema. Only Clavier sends SQL commands to the database engine. Clients send requests to Clavier via TCP/IP in the form of Tcl commands. The client itself does not require a Tcl interpreter, but the most common client is an extended Tcl interpreter called Bach. Bach is a generic tool for building GUI database applications. Applications for Bach are Tcl scripts. A version of Bach exists for X based workstations, and character terminals connected to a Unix server. A version for Windows based PCs is being developed. Clients that do not include a Tcl Interpreter use a small C library to send Tcl requests to Clavier. The components of EBase3 are depicted in figure 1. The design uses the three-tiered architecture being embraced by packages like PowerBuilder[1].

An elegant feature of the EBase3 design is that the Bach client can load all the Tcl code from Clavier. This feature simplifies maintenance by providing a single point of installation, and allows easy operation over a wide area network.

The remainder of this paper consists of three major sections. Section 2 describes the Tcl extension packages used in the Bach server and client. Section 3 presents some applications implemented using the Bach client. Section 4 concludes with observations from the EBase3 project.

2. Tcl Extensions

Clavier and the Bach client include many Tcl extensions to simplify the creation of an information system. Both Clavier and Bach include the object-oriented programming extension OTcl [2]. The PC and workstation versions of

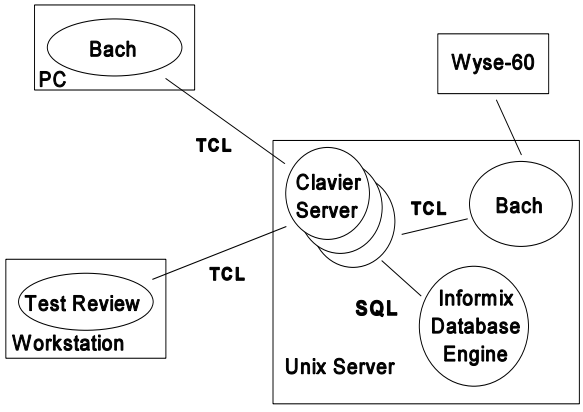

Figure 1. EBase3 Components.

Bach include Tk as the basis for their user interface. The character terminal version of Bach includes CTk, a curses based replacement for Tk that is discussed in the next section. All Bach versions include the table geometry manager from BLT that simplifies the layout of forms, and Stephen Uhler's HTML library for rendering formatted text. In addition to these outside-developed extensions, several new extensions have been developed at the Cleveland Clinic. These in-house extensions are presented in the following subsections.

2.1 CTk

Curses Tk, or CTk, is a replacement for Tk that displays to character terminals. The user interface is navigated using keyboard bindings mostly compatible with Tk (some differences occur because most terminals do not support the range of modifier keys used in Tk). While not as pleasant to use as Tk, CTk provides an important fall back when Tk cannot be used. Besides the obvious case when only a character terminal is available, CTk has also proven useful for remote access. A screen snapshot of the EBase3 schedule application running under CTk is shown in figure

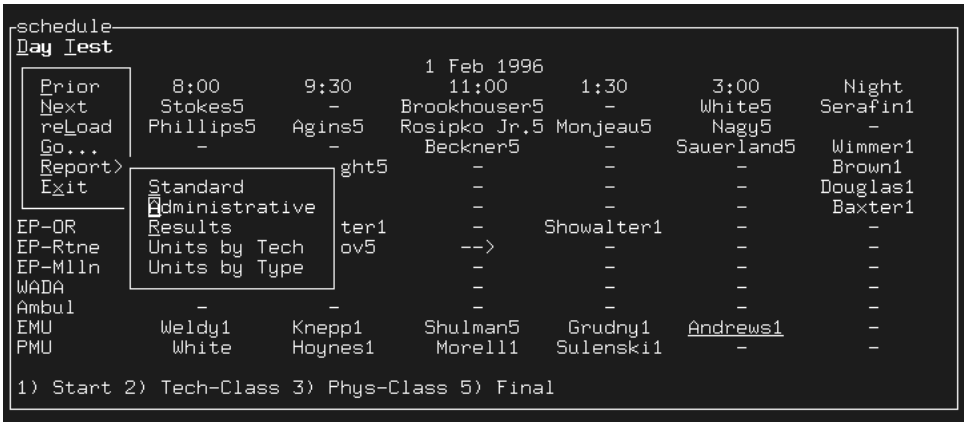

Figure 2. EBase3 Schedule application using CTk.

```
informix connect myConn "mydatabase"
myConn prepare mySelect {
  select id, first_name, last_name from employees
  where last_name matches ?
  order by last_name, first_name
}
mySelect execute A*
while {[mySelect into vals]} {
  puts "$vals(id) - $vals(last_name), $vals(first_name)"
}
```

Figure 3. Tcl script using the Rdb command "informix". It prints all employees whose last name begins with "A".

2. Like Tk, CTk is written in C except for event bindings, which are defined in Tcl.

CTk supports the majority of Tk commands, but the differences require existing Tk applications to be modified to run under CTk. It easy to write applications that will work with both Tk and CTk if this goal is considered from the outset. For an application to be portable to Tk and CTk, the most important requirements are resolution independence and support for small screens. These attributes are useful even for applications solely intended for Tk.

2.2 Rdb

Rdb provides a Tcl interface to relational databases. Currently, Rdb only provides an interface to Informix. Rdb is loosely based on DBI [3], the universal database interface being developed for perl, so it is intended to interface with a variety of database engines. An example Tcl script using Rdb is shown in figure 3. Rdb includes the flexible error handling scheme from DBI that supports recovery from failed database commands. It also supports access to BLOBs. Rdb is implemented entirely in C.

2.3 Bach

The Bach extension is a library of database utilities us ed in both Clavier and the Bach client. It defines network commands and OTcl classes for domains, fields, forms and reports. Bach is implemented in a mixture of C and Tcl.

The Bach network commands provide a simple method for sending Tcl commands via TCP/IP. In addition, the Bach library includes a short wrapper for the Tcl source command that allows the Bach client to auto-load Tcl files from Clavier.

Domains are OTcl objects that provide easy access to a database. A domain represents a set of member values, and defines a list of attributes for its members. Each attributes itself has an associated domain. For example, date is a domain and one of its attributes is year. The domain of the year attribute is integer. In addition to algorithmic domains like date and integer, Bach includes the SqlTable class for domains representing the rows of a relational database table. Bach automatically defines parameterized types like character(20), the set of all character strings with length 20 or less, by patching the unknown Tcl command. Domains also specify default fields, forms, and reports for depicting members of the domain.

Fields are widget classes that can take on any value from their assigned domain. All fields provide methods for storing, retrieving, and validating their value. Forms are mega-widget classes containing one or more fields. A simple form layout can be generated automatically, or an intricate layout can be specified explicitly.

The Report class provides the basis for generating HTML formatted reports. Using HTML provides a natural method for browsing records in the database. HTML is used for all formatted text in EBase3, removing the need for separate code for printing or viewing a record. To facilitate printed reports, non-standard mark-up tags and attributes are used to control features like page breaks and page orientation. A custom HTML formatter interprets these non-standard notations and outputs the report in PostScript, HP-PCL, or ASCII format. The HTML report formatter is an independent program implemented in C.

3. Applications

EBase applications are Tcl scripts executed by the Bach client. These applications share a common library of Tcl code that defines domains specific to EBase, and their associated user interface components. The EBase library currently consists of 8000 lines of Tcl code, over half of which is for the complex domain representing neurology tests. Neurology tests have thirty attributes including

complex items like findings and classification that contain many attributes themselves. The actual format of these complex attributes is determined by the study type of the test. The neurology test form is depicted in Figure 4.

The Patient application, shown in Figure 5, provides easy access to all EBase records associated with a single patient. Users are presented with the patient demographics and a concise medical history. Selecting a history entry opens a hypertext browser. At any point, a form can be opened to update or add new patient records. The schedule application, shown in Figure 6, presents neurology tests by the date they were performed. In addition to scheduling or rescheduling tests, the user can enter the hypertext browser by selecting one of the entries. In addition, the schedule application provides a collection of administrative reports. Both of these applications are short Tcl scripts: the patient script contains 221 lines, the schedule script 376 lines.

Figure 4. Form for editing Neurology Tests.

4. Conclusion

EBase3 has succeeded in creating a portable information system that is easy to use and extend. It is in production use in an environment that schedules 800 tests per month and averages a dozen active users. The new system has been greeted enthusiastically by a wide range of users including secretaries, technologists, physicians, and staff in training.

The tools created for EBase3 have also shown promise for use in other information systems. Clavier and the Bach client were used to create a modest application to generate project status reports. This application manages six database tables, and includes five forms and one report. It required only 550 lines of Tcl code.

I initially considered Tcl for use in EBase3 not because of any attribute of Tcl itself, but because of the Tk Toolkit. While the Tk toolkit has succeeded in providing a rich user interface for EBase3, Tcl itself has proven well suited to

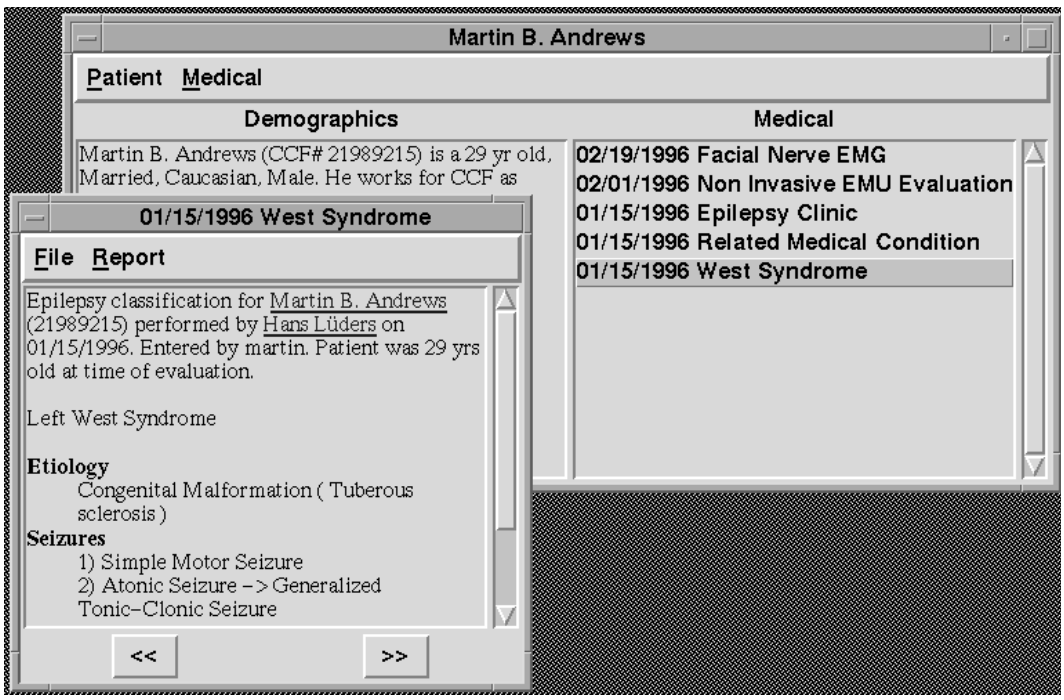

Figure 5. EBase3 patient application.

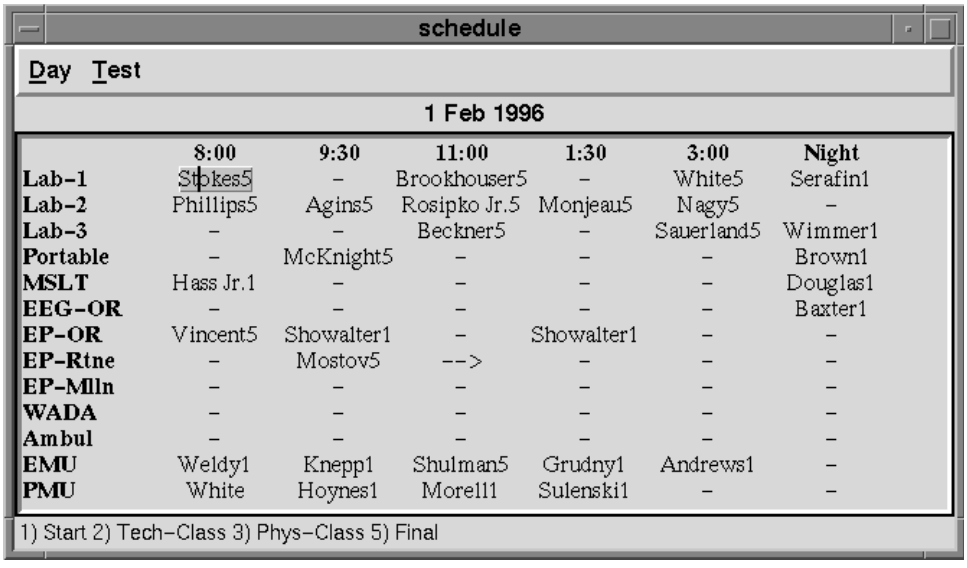

Figure 6. EBase 3 schedule application using Tk.

departmental information systems, where rapid development is usually more important than blazing performance and most of the work consists of string manipulation and interacting with the user. The main problems discovered with Tcl are listed below.

- Performance - While performance was not the main concern in EBase3, there were times where Tcl was too slow even for high level functions. Performance problems have been overcome by recoding commands in C, but this is not an ideal solution. In addition to the greater development effort required by C, C code must be built and installed on all of the client platforms. Tcl code only needs to be installed on the Clavier server. The eagerly awaited on-the-fly Tcl compiler will hopefully solve this problem. Tight integration with Java is another possible solution.

- Development Tools - GUI development tools for Tcl would further the goal of rapid application development, and greatly assist in bringing new programmers up to speed. The SpecTcl project [4] at Sun Labs addresses the need for a GUI builder. A source browser (supporting object-oriented programming) and a debugger are also needed. Production grade tools are a must.

References

[1] Powersoft, *Building Distributed PowerBuilder Applications,* Powersoft, Concord, MA, 1995.

[2] David Wetherall and Christopher J. Lindblad, *Extending Tcl for Dynamic Object-Oriented Programming,* 1995 Tcl/Tk Conference Proceedings.

[3] Tim Bunce, *Perl 5 Database Interface (DBI) API Specification,* Draft Version 0.6, 1994.

[4] Stephen Uhler, *A Graphical User Interface Builder for Tk,* 1995 Tcl/Tk Conference Proceedings.