Implementation and Use of a Tcl/Tk API for the Accelerator Control System of the The Svedberg Laboratory

L. Thuresson, V. Ziemann
The Svedberg Laboratory, Uppsala, Sweden
E. J. Veldhuizen
Department of Radiation Sciences, Uppsala University, Uppsala, Sweden

Abstract

The installation of a Tcl/Tk [1] Application Program Interface (API) for the The Svedberg Laboratory (TSL) accelerator control system has greatly simplified the development of high level applications for steering, data acquisition and control. The Tcl scripting language has a well defined C interface that makes it easy to add user extensions in compiled code. The Tk toolkit provides a basic widget set for creating platform independent Graphical User Interfaces (GUI). Third party extension packages add high level widgets that further reduce development time for GUI based applications. We describe the implementation and discuss problems arising when maintaining a Tcl installation incorporating large numbers of extension packages. Finally, we describe some of the applications written using this framework, such as automatic beam alignment for proton therapy and a GUI based interface for operation and testing of the personnel safety Radiation Protection Interlock (RPI) system.

1 Introduction

The TSL control system controls the Gustaf-Werner-Cyclotron and CELSIUS storage ring together with about 200 m of beam lines. The cyclotron produces protons and a wide range of light and heavy ions with rigidities up to 2 Tm. The beams are used for nuclear physics, material sciences, isotope production, and clinical proton therapy. CELSIUS is mainly used for nuclear physics experiments with projectiles of up to 7 Tm (e.g. 1.3 GeV protons) interacting with a cluster jet target and, soon, a hydrogen pellet target.

The control system hardware is based on VME-computers and UNIX workstations. The VME-computers are used as interface to various front-end electronics equipment. Each VME-computer holds a live version of the subset of the control system database that corresponds to the equipment connected. The VME-computers are connected to a dedicated Ethernet network together with the workstations, which area used as operator consoles [2]. Each workstation holds a copy of the complete database but only the parts currently in use by applications are updated.

The TSL control system uses a generic device description called a *parameter* to model the hardware equipment in the control system database. The parameter has a 48-bit status field and optionally two control values and two acquisition values, integer or floating point. For example a parameter representing a magnet power supply typically includes the status field, one control value (current) and two acquisition values (current and voltage), whereas a vacuum valve parameter only has the status field. C and FORTRAN libraries are already available for writing control system applications and now we are adding Tcl.

The GUI based high-level application programs required for day-to-day operation of the TSL accelerator have traditionally been developed by the control system engineering group using the C programming language. Writing GUI based programs for the X-windows system is a complicated and time consuming task, even with the help of GUI builder tools. The installation of a Tcl/Tk API to the control system has radically changed this. High level widget extensions to the Tk tool-kit makes it possible to develop GUI based applications in a fraction of the time it used to take to do the same in C [3,4].

2 Implementation

Third party extensions to Tcl called *packages*[1] can be written in Tcl or in compiled code where Tcl's C language API provides access to the Tcl interpreter. A function written in compiled code can be compiled and linked together with the Tcl library to provide a new Tcl-shell with an extended command set. As the number of packages in a Tcl installation increases, the number of used combinations of Tcl-shells with different extensions included can rise to a point that makes maintenance a nightmare. This was observed by the developers of Tcl and as of version 7.5 packages in compiled code can be loaded on demand by the running application[2]. The mechanism used to implement this is the dynamic linking and loading of shared libraries found in many modern operating systems. Unfortunately the implementation of dynamic loading is very platform dependent [5]. Particularly it can be difficult to load packages written in C++ (which happens to be the language of choice for the control system Tcl API package) since global objects depend on constructor functions being called at program startup and the corresponding destructor functions being called at termination. Another difficulty is that the C++ run-time library is not linked into the Tcl/Tk shells from the standard distribution. The *plus patches* [6]

[1] Extensions built as *packages* follow a set of programming rules to avoid interference with other packages [1].

[2] Packages written in Tcl could be dynamically loaded before version 7.5.

is a set of patches to the Tcl/Tk distribution that enables loading of C++ packages.

2.1 The Tcl API

The Tcl API uses the object-oriented approach [1] when adding commands for accessing control system parameters. A new Tcl command is created for each parameter the application program needs to access. In the example below the *cs_par* command is used to create a new Tcl command for parameter Q_A1. The new command is given the same name as the parameter and the first argument to the command specifies the action to perform.

```
% cs_par  Q_A1
=> Q_A1
% Q_A1 c1_read
=> 35.2
```

The parameter-commands have a number of options that control return format, update method, error reporting, etc. The Tcl API uses an option-handling model where default values for options are inherited from parent to child when a parameter-command is created. The Tcl interpreter is the parent of the new command and the command is the parent of six children corresponding to actions performed on the parameter.

```
Interpreter->Parameter|->status
                      |->actuation
                      |->acquisition 1
                      |->acquisition 2
                      |->control 1
                      |->control 2
```

The default option values can be changed with *config* commands, or be overridden on the command line when executing a parameter-command. An extensive set of options gives the experienced programmer power to fine tune the behaviour of the package, yet with *sane* default values beginners get a quick start.

3 Applications

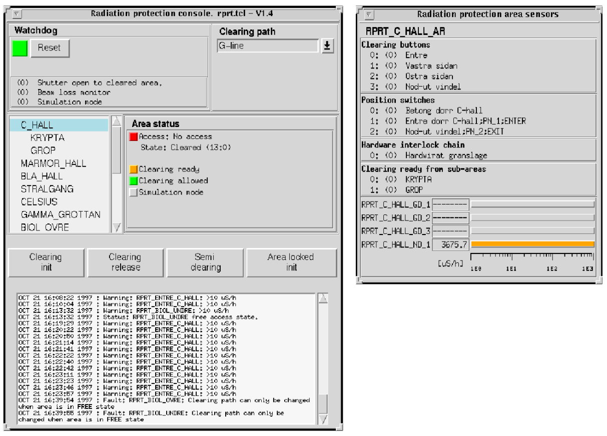
Although Tcl is great for rapid development of small applications it's support for structuring the code and data is poor. This becomes more and more noticeable as the programs grow larger. The Tix mega-widget[3] package [7] has been used at TSL in the development of larger applications. The Tix-package comes with a simple Object Oriented Programming (OOP) framework, the *Tix Intrinsic*. The Tix Intrinsic is not a general purpose OOP system such as [Incr Tcl] [8]. It is mainly designed for writing widgets, but it has been used successfully to encapsulate the control system functionality directly into the widgets used for interaction with the operator. [Incr Tcl] has not yet been used at TSL mainly because it requires modification of the Tcl core and thus can not be dynamically loaded[4].

[3] Widgets made out of other widgets.

[4] Tcl 8.0, released Aug 97, have built-in support for name-spaces. This will allow new versions of object based systems like [Incr Tcl] to be loaded dynamically.

3.1 Radiation protection interlock system

One system where Tix is used is the RPI system. This system *supervises* the laboratory localities to ensure personnel safety during accelerator operation. The RPI divides localities into *areas* that can be sealed-off before the beam is allowed to enter. The number and types of RPI sensors/actors varies in each area, but the control system API allows the application program to query the control system database about each area configuration. The Tcl/Tk user interface uses this information to build an area control panel at run time. The Tk packer geometry manager is ideal for this situation, since it automatically adjusts size and position of the control panel widgets when the configuration changes. The Tix-package has been used to construct a hierarchy of high level widgets that match the configuration of sensor/actor groups in an area. The RPI system includes some 50 radiation detectors for measuring the dose rate in different areas of the laboratory. The BLT-package [9] is used to generate time plots of the dose rate measurements.



```
(0)  Shutter open to cleared area.
(0)  Beam loss monitor
(0)  Simulation mode
```

```
Access: No access
 State: Cleared (13:0)

Clearing ready
Clearing allowed
Simulation mode
```

```
OCT 21 16:08:22 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:10:04 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:13:32 1997 : Warning: RPRT_BIOL_UNDRE: >10 uS/h
OCT 21 16:13:32 1997 : Status: RPRT_BIOL_UNDRE free access state.
OCT 21 16:19:29 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:20:22 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:20:50 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:21:14 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:21:41 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:22:22 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:22:40 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:22:42 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:23:11 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:23:23 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:23:46 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:23:57 1997 : Warning: RPRT_ENTRE_C_HALL: >10 uS/h
OCT 21 16:39:54 1997 : Fault: RPRT_BIOL_OVRE: Clearing path can only be changed
when area is in FREE state
OCT 21 16:39:55 1997 : Fault: RPRT_BIOL_UNDRE: Clearing path can only be
changed when area is in FREE state
```

```
Clearing buttons
  0: (0)  Entre
  1: (0)  Vastra sidan
  2: (0)  Ostra sidan
  3: (0)  Nod-ut vindel
Position switches
  0: (0)  Betong dorr C-hall
  1: (0)  Entre dorr C-hall;PN_1;ENTER
  2: (0)  Nod-ut vindel;PN_2;EXIT
Hardware interlock chain
  0: (0)  Hardwirat granslage
Clearing ready from sub-areas
  0: (0)  KRYPTA
  1: (0)  GROP
RPRT_C_HALL_GD_1 --------
RPRT_C_HALL_GD_2 --------
RPRT_C_HALL_GD_3 --------
RPRT_C_HALL_ND_1   3675.7
```

Fig 1. RPI system console and area sensor display.

The RPI system also has a simulation mode to simplify testing. In simulation mode the operator can select which sensor signals should use the actual hardware sensor or take its value from a button or slider. The simulation mode also makes it possible to run the RPI system on an off-line computer with no connection to the actual sensor/actors while testing new versions of the program.

3.2 Correlation plot facility

The availability of all control system parameters together with a graphical user interface allows writing a simple application that changes one parameter and monitors various other simultaneously to determine their dependence. The application is enhanced with a graphical display of the data taken. This code is modelled after a similar system in the SLC control system at SLAC [10].

3.3 Heavy ion irradiation facility

Tcl/Tk has been used to control a recently constructed heavy ion irradiation facility for material science. The facility is capable of homogeneously irradiate a 40 x 40 mm sample and varying the dose rate over 6 orders of magnitude. Most of the hardware equipment (such as an electrostatic deflector at the ion source, steering magnets, shutters, beam intensity monitors and timing electronics) was already present and connected to the TSL control system because they were used in other experiments. Thus a major portion of the task was writing software to coordinate the equipment actions during the irradiation process. The only time-critical section was the scanning of the beam over the surface of the sample. This was written in C. All other functions, like calculation of the scan speed and chopper frequency, calibration of the beam and putting shutters in the right position was done in Tcl, which was much faster to program.

A new version of the control system for the irradiation facility is in development. It will have the time-critical section hardwired and use Tcl/Tk for all other functions. The first prototype of the Tcl/Tk program was written within two days. The new version will allow us to graphically pre-program a complete irradiation and monitor it's progress when running.

The final application is intended for researchers that have to perform the irradiation but do not have any knowledge about accelerators and thus it requires a graphical user interface that is foolproof and user friendly.

3.4 Front-end to other applications

Applications written in C or FORTRAN that sometimes have arcane command line options were provided with a Tcl/Tk front-end. This greatly increased the acceptance of new control features by our operator community and reduced errors in running the applications.

In one case a slow steering system that monitors and corrects the position of the proton beam for proton therapy treatment received a Tcl face-lift. The needed Tcl code was written within a day and has made expert intervention when running the system obsolete.

In another case a program that executes other programs synchronised with injections into the CELSIUS storage ring, which typically happen every five minutes, received a Tcl front-end. It is now routinely used to deliver beam to experiments that parasitically utilise it when CELSIUS does not need beam for injections.

In a third case applications that provide simplified interfaces to complex instruments connected to the control system via GPIB such as digital oscilloscopes or network analysers are written. These applications allow setting up and using these devices by non-experts. The applications are enhanced with simple data analysis and display routines written in Tcl using the BLT-package. These codes allow fast on-line interpretation of the accumulated data. Consequently, data taking and analysis during accelerator physics experiments have become faster and more reliable.

In another group of applications we use Tcl/Tk applications to provide simple interfaces to automate repetitive tasks that are typically done by shell scripts.

4 Conclusions

The advent of Tcl has increased the amount of useful applications that can not only be used by their respective authors dramatically. The availability of all control system parameters, GPIB devices, advanced graphics, and applications written in other languages such as C, FORTRAN, or shell scripts under the Tcl/Tk roof of a graphical user interface provides a perfect framework for generating small applications that help our operators running the cyclotron and CELSIUS more stable and reliable. The ease at which GUI based application can be built with Tcl/Tk has lead to an increased interest from the operations/physicist groups to take part in program development for the control system.

References

[1] J. K. Ousterhout, Tcl and the Tk Toolkit, Addison-Wesly Professional Computing Series 1994.

[2] K Gajewski, L Thuresson, O Johansson, Upgrading of the control system for the accelerators at TSL, ICALEPCS 1991.

[3] B. Welch, Practical Programming in Tcl and Tk second edition, Prentice Hall 1997.

[4] Tcl/Tk is available from http://sunscript.sun.com/TclTkCore/

[5] K. B. Kenny, Dynamic Loading for Tcl: (What became of it ?), http://ce-toolkit.crd.ge.com/papers/gecrd/mtl/mdip/tcl94/tcl94.ps

[6] Plus patches are available from http://www.worldaccess.nl/~nijtmans/plus.html

[7] I. K Lam, Designing Mega Widgets in the Tix Library, Tcl/Tk Workshop 1995, Tix is available fromhttp://www.xpi.com/tix/

[8] [Incr Tcl] is available from http://www.tcltk.com/itcl/

[9] BLT is available from http://www.tcltk.com/blt/

[10] L. Hendrickson, N. Phinney, L. Sanches-Chopitea, Correlation Plot Facility in the SLC control system, PAC 1991.