Implementing a Visualization of an Industrial Production Cell Using Tcl/Tk[1]

Artur Brauer, Claus Lewerentz, Thomas Lindner

Forschungszentrum Informatik
Haid-und-Neu-Straße 10-14
76131 Karlsruhe Germany
email: {brauer, lewerentz, lindner}@fzi.de
Mai 1993

Abstract

In this work, an application of Tcl/Tk to the domain of process visualization is described. We developed a simulation of an industrial production cell for to evaluate and validate control software for this (reactive) system. A major requirement was to provide a simple integration of the simulation with control programs.

In the first chapter the production cell and requirements for the visualization are described. Then the way the simulation was implemented using Tcl/Tk and our experiences are reported. This should be interesting for people who want to use Tcl/Tk for building visualizations of reactive or real time systems.

1. Problem

In the context of the German "Korso" Project (Correct Software) a number of formal software construction methodologies are developed. The Forschungszentrum Informatik (FZI) proposed the case study "production cell" to evaluate and compare these approaches. At FZI we built a realistic functional model of a small industrial production cell, sized about 1 x 1 meter. A schematic diagram of the cell is shown in figure 2. The task of the case study, which is treated by several research groups in Germany, is the development of control software with various approaches, in order to compare the approaches and to show the usefulness of formal methods enforcing safety requirements in industrial applications. As the partners are distributed all over Germany it was necessary to provide them with a facility to evaluate their developments. We decided to construct a graphical software simulation of the production cell that should be easy to build and easy to connect with any control software.

We decided to use Tcl/Tk by the following reasons: Firstly, Tcl/Tk enables one to program in X on a higher level. Especially the *canvas widget* helped us to speed up the application development. In previous experiments with Tcl/Tk we observed a much faster development compared with standard X programming. The second reason was that we had to provide an extremely simple coupling mechanism between the visualization and any control software developed by the project partners. A students work [3] had shown us that this coupling is very simple using Tcl mechanisms (see section 2).

2. Solution

The production cell is equipped with about a dozen of sensors, measuring for example the angle of rotation of a robot, and about a dozen of actors, for example a motor for closing or opening a

1. This work is sponsored by the German Ministry of Research and Technology (BMFT) as part of the compound project „KORSO — Korrekte Software".

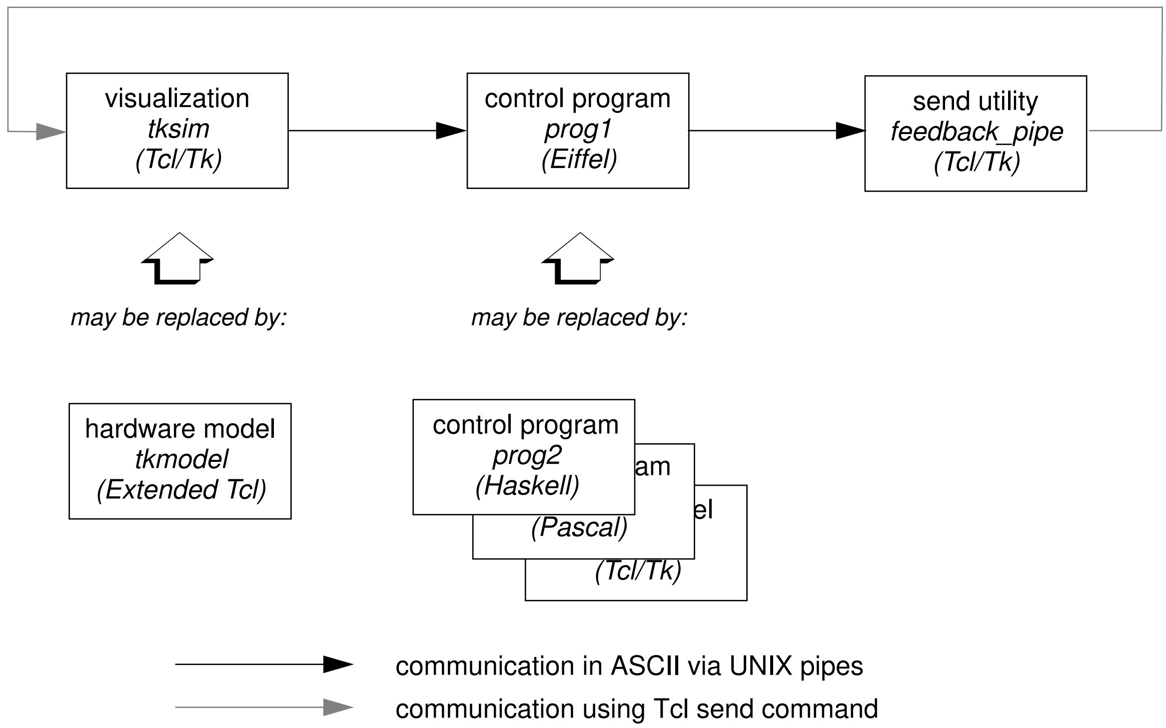

Figure 1 Communication architecture

press. For a complete understanding of the model see [4]. This understanding is not needed to catch the main points of the following presentation.

The first subsection describes the implementation of the visualization in Tcl/Tk, the second subsection reports on the coupling between the visualization and the control software.

2.1 Graphical Visualization Using Tcl/Tk

All objects making up the production cell are drawn in a canvas widget of Tk. The definition and manipulation of graphical objects is well supported by the facilities of the canvas widget. We used very simple geometric shapes such as lines and rectangles, in order to achieve good performance. Good performance is an important requirement, because the real time behavior of the production cell has to fit as well as possible to the behavior of the hardware model. As the visualization requires moving and rotating about a dozen of graphical objects, this performance is lost when using complicated graphical representation.

The updating of the graphics is performed within a main loop. There, the script looks for the status of the actor switches and moves or rotates the corresponding devices if necessary. The capabilities of Tcl/Tk for moving objects simplified the implementation. As the production cell processes metal blanks the blanks must often be moved together with other objects, e.g. the conveyor belt transporting them. This is easily achieved in Tcl/Tk by packing them to a joint object and applying the move command to the whole group. The capabilities of Tcl/Tk of dealing with what is in the foreground and what is in the background of the canvas have proven to be useful when dealing with overlapping.

The details of the implementation are described in a technical report [5].

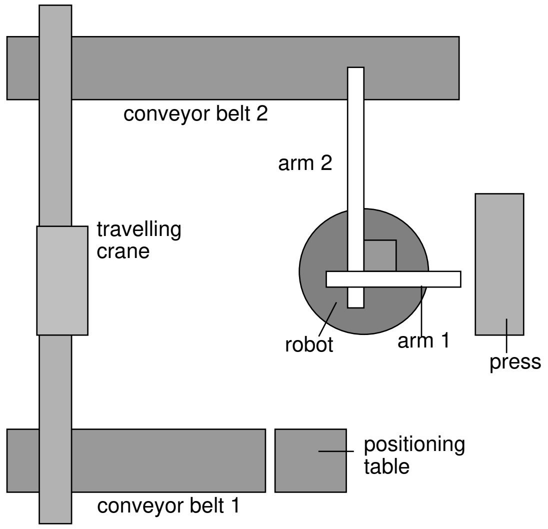

Figure 1 Schematic diagram of the production cell

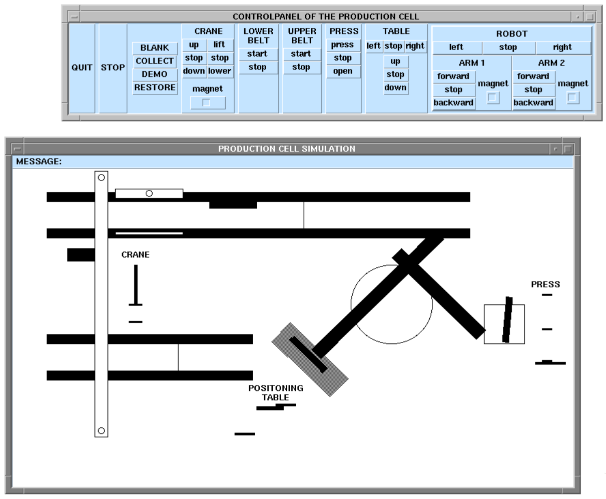

Figure 2 Screen dump of the visualization (together with panel)

2.2 Coupling the Visualization and the Control Software

We defined a simple ASCII communication protocol between the visualization and the control program. A control program issues commands provided by this protocol to *stdout,* and can thus switche the actors of the production cell on or off. With a special command also provided by the protocol definition, the control program can ask for the sensor status, which can then be read from *stdin*.

We designed the visualization as a set of Tcl/Tk procedures, such that to each actor command there is a corresponding procedure. The status of the sensor values is recorded internally and, if requested for, is written to *stdout*. We added a simple Tcl-script called *feedback_pipe*. Its task is to recognize commands issued from the control program (read in by *stdin*), and to call the respective procedures of the visualization using the Tcl-built in *send* command. The very simple overall architecture of the application is depicted in figure 1.

This coupling mechanism is flexible in two ways: of course, one can easily replace *prog1* by another control program, even by a panel for manual control (which was written using Tcl/Tk, too). Note that this would not be so simple if we had used UNIX named pipes. In addition, it is also possible to replace the simulation by a simpler Tcl-script which directly sends commands and retrieves the state (via the serial port) from the hardware model. In order to achieve this, some small low-level C procedures were added to Tcl, thus resulting in a Tcl extension.

3. Conclusion

The use of Tcl/Tk allowed for building a quite complex animated graphical simulation in a rather short time. The entire effort including the learning of Tcl/Tk by one of the authors was about 120 hours. The Tcl/Tk interpreter proved to be a robust and reliable tool for this task. The smooth embedding in the standard communication mechanism of UNIX (using character streams and pipes) and the straight-forward extensibility of the Tcl interpreter by custom C functions were very useful for the integration of different parts of the system.

One major drawback of the current Tcl/Tk version in this application was the degradation of performance when relying extensively on procedures to structure the application. Additionally, powerful and suitable modularization and abstraction mechanisms for data structures as well as for functions would be very useful. We would like to have object-oriented concepts for structuring and reusing Tcl programs.

References

[1] J. K. Ousterhout, An Embeddable Command Language, Proceedings of the1990 Winter USENIX Conference.

[2] J. K. Ousterhout, An X11 Toolkit Based on the Tcl Language, Proceedings of the 1991 Winter USENIX Conference.

[3] Kai Gutenkunst, Techniques for the coupling of user interfaces and applications, Forschungszentrum Informatik, Haid-und-Neu-Straße 10-14, D-76131 Karlsruhe, 1992 (in German language).

[4] Thomas Lindner, Case Study Production Cell: Task Definition, Technical Report, Forschungszentrum Informatik, Haid-und-Neu-Straße 10-14, D-76131 Karlsruhe.

[5] Artur Brauer, Claus Lewerentz, Thomas Lindner, Implementing a visualization of an Industrial Production Cell Using Tcl/Tk, Technical Report, Forschungszentrum Informatik, Haid-und-Neu-Straße 10-14, D-76131 Karlsruhe, in preparation.

Appendix: How to Install and Use the Visualization

Prerequisite: Have a running Tcl/Tk installation.

1. Get the file *visualization.tar.Z* from the FZI ftp server *gate.fzi.de* (Internet 141.21.4.3) in directory *pub/korso/fzelle/simulation*. Use binary transfer mode.

2. Un-compress and un-tar the file. Check the completeness of the installation by comparing the contents of your current directory to the list of files in the README file.

3. Change the files *tksim, panel,* and *feedback_pipe* by entering the correct path of your *wish* interpreter in the first line.

4. Under UNIX, now just enter
   tksim | *panel* | *feedback_pipe.*
   The simulation will come up, together with a control panel for hand steering, which was also implemented using Tcl/Tk.

5. Press the *DEMO* button in order to get a demonstration of a typical processing cycle of the production cell.

Please report any errors to *lindner@fzi.de*. Also comments are welcome.