Using Tcl/Tk in a Legacy Application

Franco Violi - Tech Mgr
Metodo Srl - Modena - Italy
fvioli@metodo.net
http://www.metodo.net
http://www.lapam.mo.it

Metodo Srl

- IT technology leader for a pool of companies, 560 employes
- Our application is used on 60 servers, from 60 to 2 users each
- The application has been also installed in 300 companies, growing each month
- From Payroll System to Order Processing and Stock Management

Legacy Applications

- Old, but they works!!!
- RPG or COBOL language
- Character based user interface
- Huge maintenance needed, because the problems change in the time

Our Application

- Written in Cobol
- A clear and well defined API User Interface
- Character based
- Heavy use of batch Database processing
  - The 60 servers collect data in the main database each month
  - The size of the main database is about 20 Gbytes

Our Application numbers

- about 3.100 Cobol Programs
- about 5.000 Screen Forms
- about 850 Database Tables
- about 1.570.000 cobol statements
- … and we use abstractions !!!!

The problem of evolving

- A Cobol Programmer is a Cobol Programmer
- The IT leader must evolve without killing the Cobol Programmer, because
  - He knows the application
  - He knows the problems
  - He knows the company

Rewrite or Evolve ?

Who will follow You ?

What will happen during rewriting ?

What about the intermediate stages in the real world ?

The Cobol Engine

- The compiler generates intermediate code (bytecode)
- The bytecode is interpreted by an executable (runtime)
- You can customize the runtime
  - Adding C code
    int my_c_sub(int argc , char **argv)
  - Making it accessible by Cobol with the standard syntax

The Tcl/Tk Engine

- Easy to be embedded in other languages, the Cobol Interpreter
- Easy to be extended with new commands, the cobolwakeup command
- A window opened to a new world
  - i.e. Cobol is not able to poll for an E-Mail and read the attachements
  - … and too many other things that makes Cobol not fully usable today

Cobol and the Event Loop

- A single C variable that defines a status
  - Tcl wants to run
  - Cobol wants to run
- A single DoOneEvent loop
  - Run until Tcl give control to Cobol
  - Update the event queue before returning control to Cobol

Cobol and Tcl/Tk
Initialize the interpreter

```
CALL "TCL" USING
           FUNC-INIT
           FUNC-STATUS
           TCL-INTERPRETER
           [ EXTENSIONS-MASK ]
```

Cobol and Tcl/Tk
Destroy the interpreter

```
CALL "TCL" USING
           FUNC-EXIT
           FUNC-STATUS
           TCL-INTERPRETER
```

Cobol and Tcl/Tk
Evaluate a script

```
CALL "TCL" USING
           FUNC-EVAL
           FUNC-STATUS
           TCL-INTERPRETER
           TCL-SCRIPT
           TCL-RESULT
```

Cobol and Tcl/Tk
Entering the Event Loop

```
CALL "TCL" USING
           FUNC-CONVERSE
           FUNC-STATUS
           TCL-INTERPRETER
           TCL-RESULT
```

Cobol and Tcl/Tk
Updating the event queue

```
CALL "TCL" USING
           FUNC-REFRESH
           FUNC-STATUS
           TCL-INTERPRETER
```

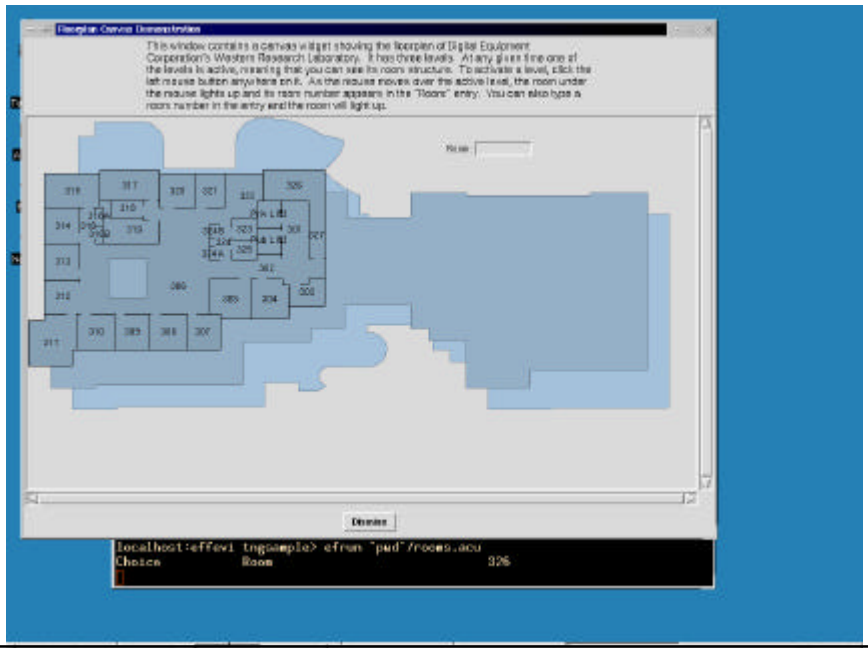
Cobol and Tcl/Tk
Connecting variables

```
CALL "TCL" USING
           FUNC-DEFINE
           FUNC-STATUS
           TCL-INTERPRETER
           TCL-NAMES
           COBOL-BUFFERS
```

The ‘hello-world.cob’ example

- Demonstrates how to connect cobol variables to tcl variables
- How to include tcl scripts into the Cobol Program
- It interacts with tcl’s event loop and tk widgets
- Demonstrates how to update the event queue

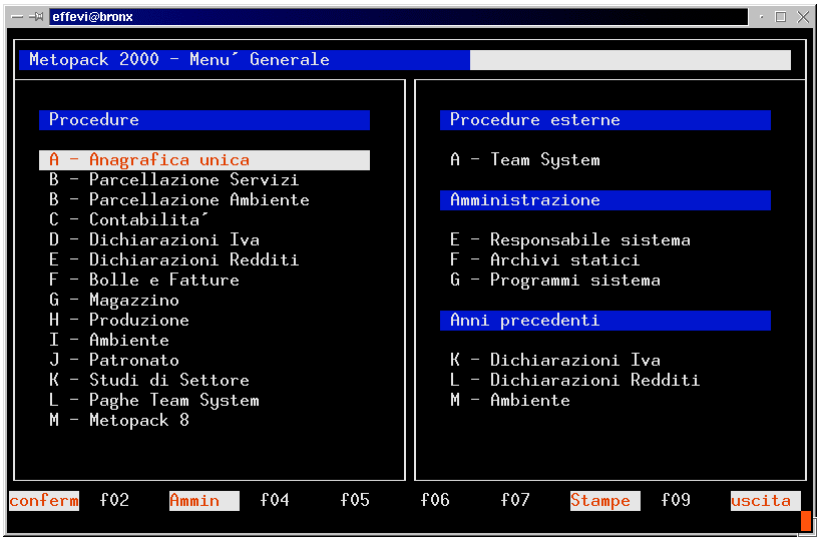
The rooms.cob screen shot



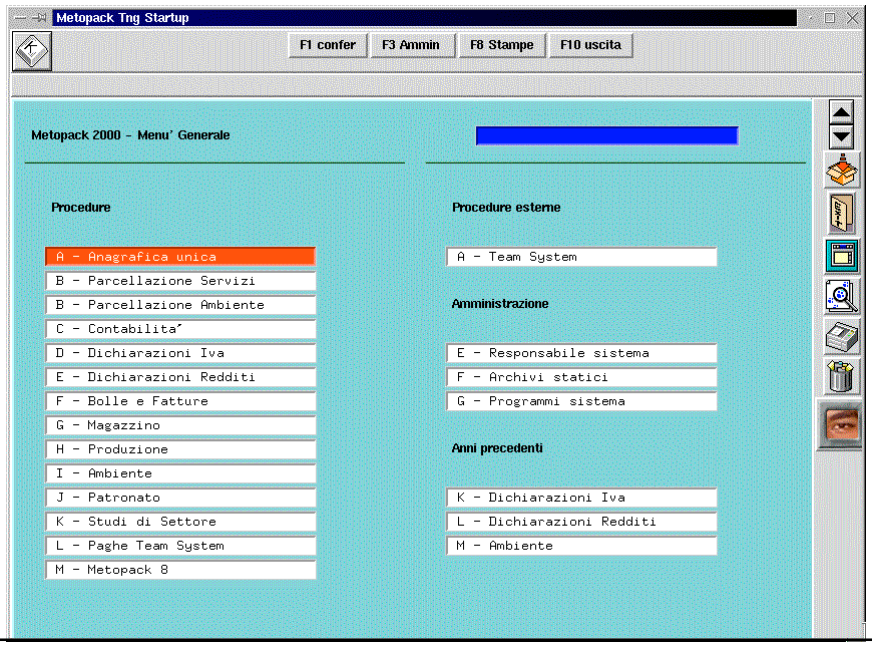
The ‘rooms.cob’ example /usr/lib/tk8.0/demos/floor.tcl

```
7a8,14
> wm protocol . WM_DELETE_WINDOW {
>       cobolwakeup Exit {} {}
> }
1306,1308c1315,1316
< button $w.buttons.dismiss -text Dismiss -command "destroy $w"
< button $w.buttons.code -text "See Code" -command "showCode $w"
< pack $w.buttons.dismiss $w.buttons.code -side left -expand 1
---
> button $w.buttons.dismiss -text Dismiss -command "cobolwakeup Exit {} {}"
> pack $w.buttons.dismiss -side left -expand 1
1358a1367,1370
> proc cobolRoom {c} {
>     cobolwakeup Choice Room [newRoom $c]
> }
1364a1377
> $c bind room <Button-1> "cobolRoom $c”
```

A screen shot before Tcl/Tk



A screen shot after Tcl/Tk



What's now in mind

- Why a tcl script must be always evaluated on the server ?
- Modern Cobols can be launched by inetd. Is it usefull ?
- … all what you can imagine in a tcl/tk 'window'