Tcl/Tk guidelines for improved automated regressions - a case-study

17'th Annual Tcl/Tk Conference (Tcl'2010)
October 11 - October 15, 2010

Saurabh Khaitan
Madhur Bhatia
Tushar Gupta

Mentor Graphics®

Agenda

- Need for automated GUI regressions
- Automated GUI testing tools
- Guidelines for developers
- Testing strategies for QA engineers
- Limitations and workarounds
- Results

Need for Automated GUI regressions

- Growing complexity of GUI
- Excessive permutation and combinations of sequence of steps to test manually.
- Rule out “human error” in testing
- Reduce time-to-market
- Eliminates repeated testing efforts

Automated GUI testing tools

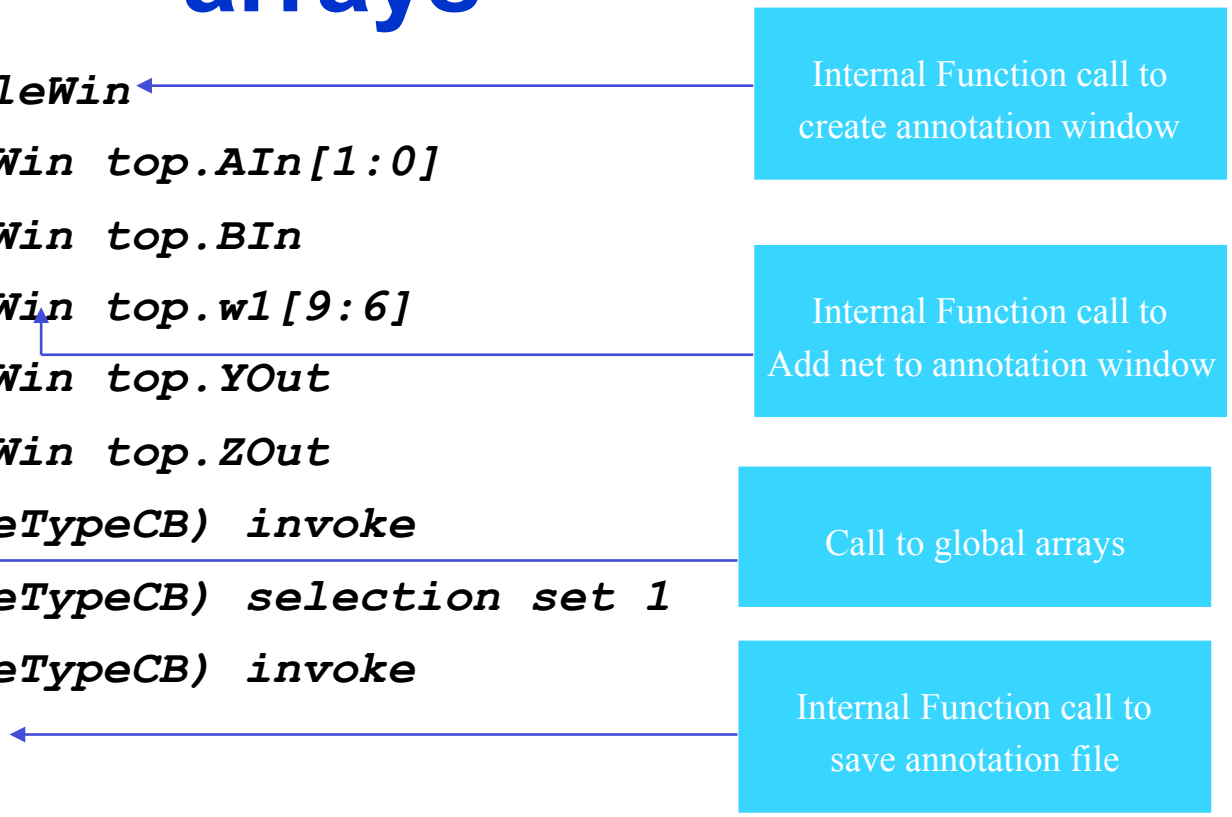
- Analog tools
  - Co-ordinate based tools
  - Tools –> T-Plan Robot, AutoIt
  - Limitations
- Object based tools
  - Works on internal objects/widgets of the GUI
  - Tools ->Squish, TKReplay

Guidelines for developers

- Direct use of internal functions
- Use of global arrays
- Use of environment variables
- Use of algorithmic functions
- Text representation of graphical display

Use of Internal functions and global arrays

```
createAnnotationFileWin
addNetToAnnotationWin top.AIn[1:0]
addNetToAnnotationWin top.BIn
addNetToAnnotationWin top.w1[9:6]
addNetToAnnotationWin top.YOut
addNetToAnnotationWin top.ZOut
$annotationWin(fileTypeCB) invoke
$annotationWin(fileTypeCB) selection set 1
$annotationWin(fileTypeCB) invoke
saveAnnotationFile
```



Use of algorithmic functions and environment variables

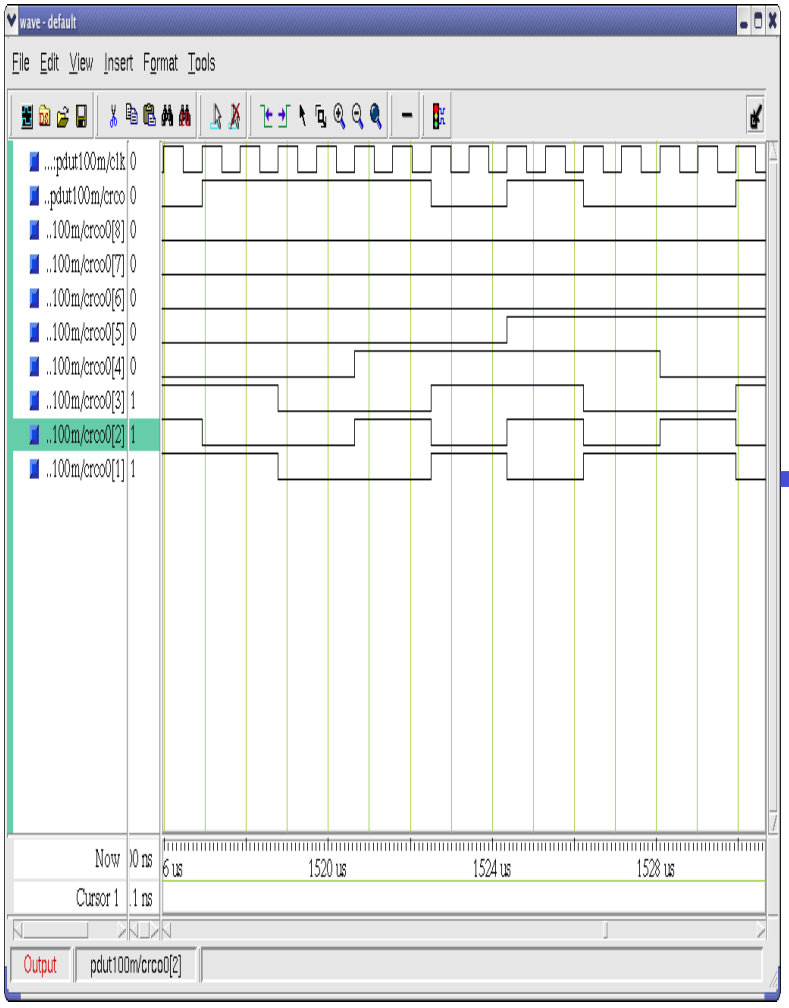
```
proc findpath {fullpath separator tree}
{
   # sanity checks
   …
   #tree search algorithm
   …
   # tree update
   …
   # code for regressions
   if {[info exists ::env(MED_REGRESSIONS)]}
   {
      if {$found}
      {
         set msg “path found”
      }
      else
      {
         set msg “path not found”
      }
      echo $msg
   }
}
```

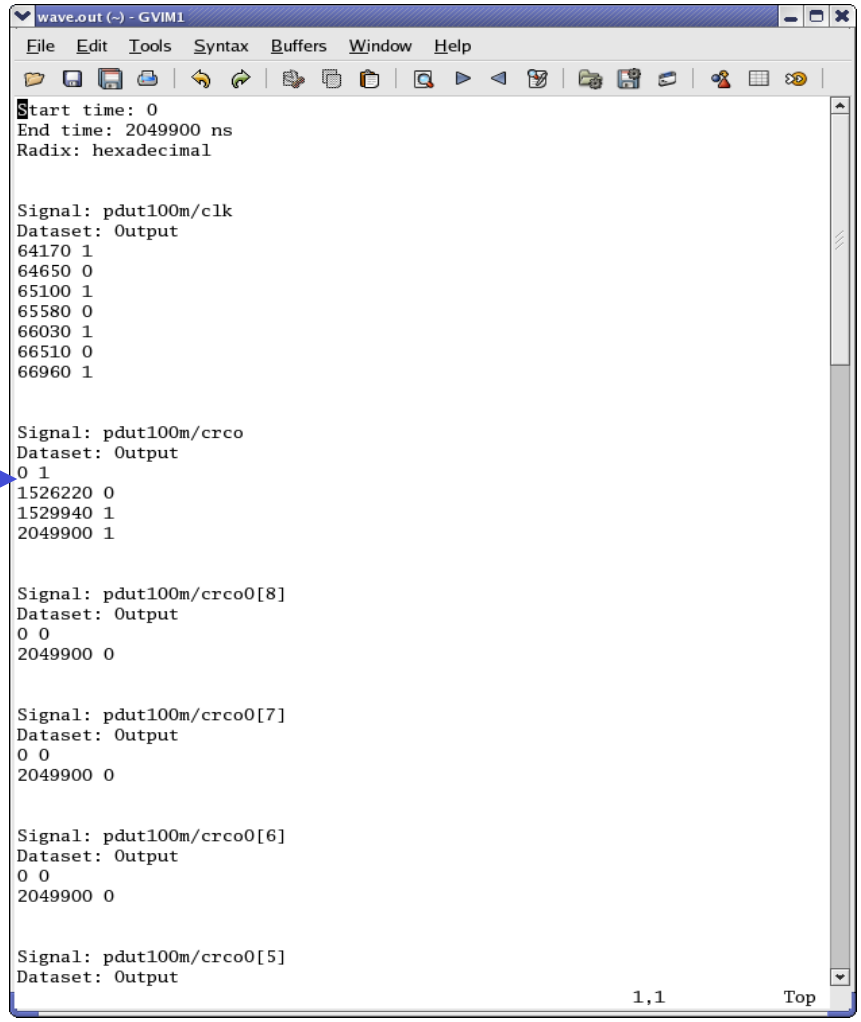
Algorithmic function call *findpath* to perform complex GUI operation

The function performs operations like tree search and tree update

This part of code is enabled under an environment variable

Text representation of Graphical display





Testing strategies for QA engineers

- Verification Points
- Synchronization points
- Use of global procedures
- Offline debug
  - Enable debugging screenshots

Verification Points

- **Checks Inserted in the code to verify the state of the GUI**
- ***test compare [property get $widget $prop_name] $expected_value***

```
# Verification Point 'newfile'

test compare [property get
[findObject ":vsim.dockbar.tbf0.standard.tb.button_0"] state] "normal"

test compare [property get [findObject ":vsim.dockbar.tbf0.standard.tb.button_0"]
    image] "__new_icon"
```

Name of the VP

Checking the state and the image of the button widget using compare calls

Synchronization points

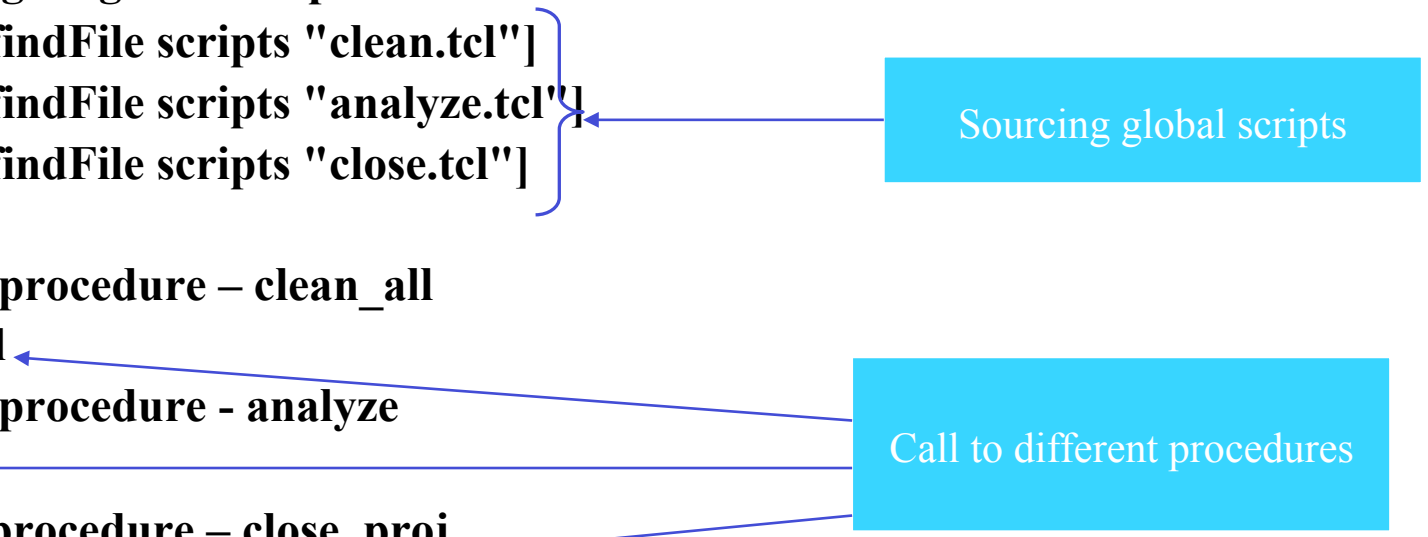
- What is SP
- Types of Synchronization
  - Time Synchronization
  - Object Synchronization
  - No Synchronization
- *waitForObjectItem $objectname $itemtext*

```
waitForObjectItem ":vsim.#mBar" "File"
invoke activateItem ":vsim.#mBar" "File"

waitForObjectItem  ":vsim.#mBar.#mBar#file" "New"
invoke activateItem ":vsim.#mBar.#mBar#file" "New"

waitForObjectItem ":vsim.#mBar.#mBar#file.#mBar#file#new" "Project..."
invoke activateItem ":vsim.#mBar.#mBar#file.#mBar#file#new" "Project..."
```

Synchronization Points inserted before every click

Use of global procedures

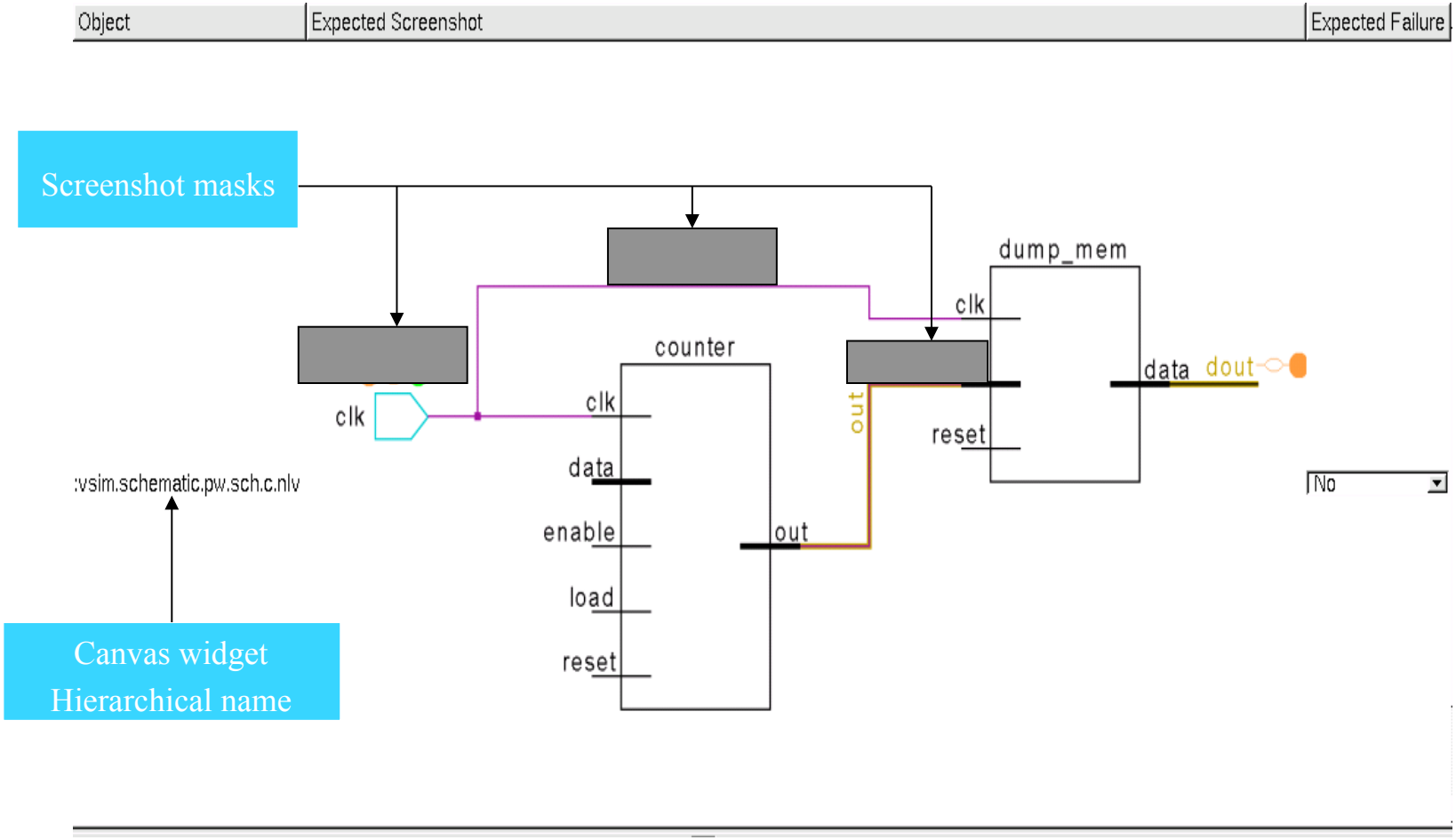
```
proc main {}
{
snooze 10

#sourcing all global scripts
source [findFile scripts "clean.tcl"]
source [findFile scripts "analyze.tcl"]
source [findFile scripts "close.tcl"]

# global procedure – clean_all
clean_all
# global procedure - analyze
analyze
#global procedure – close_proj
close_proj
}
```



Limitations and Workarounds

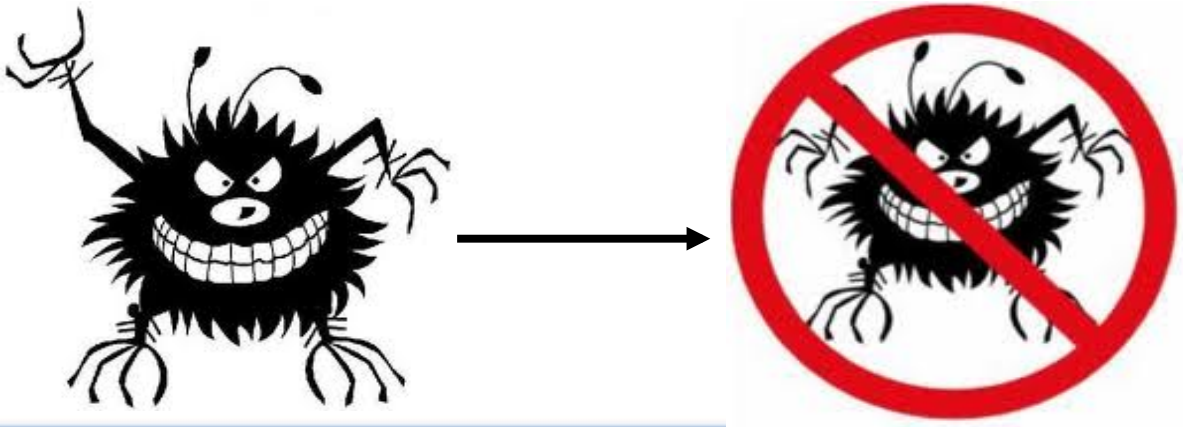
- Drag and Drop
  - Use of black box GUI testing tools
- Custom widgets
  - Test using conventional analog tools
  - Support for custom widgets from automation tool
- Canvas widgets
  - Screenshot Verification Points
  - Textual dump of graphical display

Screenshot Verification Points



Results

- Automated and stable GUI testing regressions
- Reduced testing time
- Improved test coverage
- Better product Quality – Customer reported bugs reduced significantly




Thank You
Mentor Graphics®