

THE VOV STORY; TCL IN EDA AND RELIGIOUS WARS Andrea Casotto



Tcl Conference, Houston, October 17 2018

ABOUT ME

- 22 years as founder @ Runtime Design Automation
 - Obsessive efficiency



- Chief Scientist for HPC @ Altair
 - By acquisition Sep 2017





MY TRAINING

- Learned programming from colleagues
 - 1985-1991 UC Berkeley CAD Group
- Ph.D. in 1991
 - Dissertation on "Automated Design Management"
 - A system called "VOV"
- In 1991, was Octtools Manager at UCB
 - Inherited >1M lines of code
 - Learned about reading / fixing other people's code
 - Developed a sense of "habitable code"



TIMELINE

- 1991 Ph.D. Dissertation
 - VOV with Tcl embedded
- 1992 Siemens AG project
 - Start of the Tk version of VovConsole
- 1995 Founding of Runtime Design Automation
 - Bootstrapped start-up
- 2017 Acquisition by Altair



THE STORY OF VOV

- Origins at UC-Berkeley
- VOV becomes Multiple Products
 - How Tcl is used in VOV
- Scripting language controversies
 - Tcl Perl Python ...



VOV is also the name of and Italian liqueur.



VOV AND TCL IN BERKELEY

- VOV Project started in late 1987
 - Goal: Automate chip design process
- TCL also started in 1987
 - https://www.tcl.tk/about/history.html



Evans Hall, in Berkeley, where Ousterhout's office was



WHY TCL?

- John Ousterhout
 - Was my professor
 - A TA course on "how to teach"
- Ousterhout brand
 - Magic, Crystal, Sprite, ...
- Elegance of language
 - All made sense
- Simple to integrate with C++
- Why do we still use it?
 - Amazing performance and capabilities



John Ousterhout



DRAWING FLOWS BY HAND

- I was TA in CS250 (prof Katz)
- Needed to teach students how to use Octtools
- Was drawing diagrams like this by hand
- Got tired
 - Needed a way to generate them automatically





WHAT IS A FLOW?

- A Job is a "Process" with inputs and outputs
 - Each output depends on all inputs
- A **Flow** is a collection of interdependent jobs



WHAT DOES VOV DO?

- Idea: Runtime Tracing
 - The tools at runtime inform VOV about their inputs and outputs
- Build a flow by simply executing tools
 - % vw cp aa bb
 % vw cp bb cc
 % vw cp cc dd
 - % vw cp cc ee

○ ○ ○							
<u>C</u> onsole <u>P</u> roject <u>S</u> laves S <u>e</u> ts	<u>V</u> iew <u>N</u> ode <u>T</u> race T <u>o</u> ols	<u>H</u> elp					
localhost on mac01ac		0 💓 X					
Sets Directories	× System:nodes +						
Predefined	OOS System:nodes						
⇒ System	♥♥♥┃♥♥♥♥₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽	1 A O 🔍					
nodes files							
jobs							
running							
⊳ User							
ephemeral	ephemeral filesToCheck jobsToCheck						
jobsToFire	Nothing to show						
jobsToWatch preempted							
recent							
zipjobs							
zippable							
No alerts	same	Ready.					
	Terminal – tcsh – 90×10						
Slaves:	1 Buckets: 0						
- ready:	1 [JDuration: 0s						
Slots:	8 SchedulerTime: 0.00s						
TotalResources:	10 Pid: 88340						
	Saved: 3m27s ago						
	Size: 9.00MB						
mac01ac demo@mac01ac BASE ~/demo >							

... AND YOU GOT A PH.D. FOR THAT?

- Most research on Flow Management at the time was exploring a "normative approach":
 - "Thou shalt only do what the Flow Management System says"

- Runtime Tracing supports expert designers
 - 1. Follow expert (record steps, trace the flow)
 - 2. Repeat when needed (retrace the flow)
- Traces became an important training tool for novices



MANY WAYS TO MAKE TOOLS TALK TO VOV

• Encapsulation:

% vw cp aa bb

Uses a Tcl script (a "capsule") to capture I/O behavior based on command line

Instrumentation:

#!/bin/csh -f
VovInput \$1
VovOutput \$2
cat \$1 | tr -d foo > \$2

• Interception:

- of OS calls like open(), unlink(), mmap()
- Uses LD_PRELOAD



FLOW DESCRIPTION LANGUAGE

- Build flows without executing tools
- Compact, simple

```
E BASE
```

```
X 2s
```

- **T** vw cp aa bb
 - **I** aa
 - **0** bb

```
J vw cp bb cc
```



A MORE INTERESTING FLOW IN FDL

```
for { set i 0 } { $i < 30 } { incr i } {
```

```
indir -create "subdir$i" {
```

- J vw cp ../aa bb
- J vw cp bb cc
- J vw cp cc dd
- J vw cp cc ee

}

}

- Multiple directories
- Wide, parallel
- Hard to do with Makefile

DEMO: A MORE INTERESTING FLOW

- 1. Build flow with vovbuild
- 2. Retrace different targets
- 3. Change an input
- 4. Retrace whole flow



ARCHITECTURE OF VOV



INTERESTING TCL EXTENSIONS - 1

```
• shift [list_ref]
while { $args != {} } {
    set arg [shift args]
    switch -- $arg {
        ...
    }
}
```

```
• indir {OPTIONS} {script}
  indir -create subdir {
    J vw run_Simulation
  }
```



INTERESTING TCL EXTENSIONS - 2

```
• HTML gen library
  #!/bin/csh -f
  # The rest is -*- Tcl -*- \setminus
                                                                           Title
      exec vovsh -f $0 $*:q
  •••
  VOVHTML START
                                                                                • <u>item1</u>
  HTML {
                                                                                  item2
    BODY {
      H2 { OUT "Title" }
                                                                                   item3
      UL {
         set list [list item1 item2 item3]
         foreach elem $list {
            LI { HREF "/some/url/$elem" $elem }
         }
      }
    }
  }
  VOVHTML_FINISH
```

MILESTONE: 1996, VOV IS FASTER THAN MAKE!

- VOV used to compile itself
 - Multi platform
 - Linux, solaris, windows
 - Multiple configurations
 - Debuggable, optimized, profile, coverage
- Unit testing, regression, docs, packaging



COMPILING VOV WITH 104 PROCESSORS



FLOW DEBUGGING

- Identifies flow problems
 - Missing files or dependencies
 - Flow conflicts
- Stricter file checking than "Makefile'
 - Job execution window
 - Timestamp consistency
- Saves designer time and resources



INCREASED PRODUCTIVITY

- Identify problems
- Make corrections or adjustment to the flow
- Continue without having to restart
- Better resource utilization



MORE VISIBILITY

- Easy to identify job status and failing jobs
- Quickly drill down for root cause analysis
- Full command line or web interface available

Console Project Slaves	Sets View Node Trace	Tgols Help				
c1	c2	c4 c3 c5 BKG 23 💟 🖗 🖏				
Sets Directories	× gui:client:1:000004					
Predefined	OOG 45 Quiction	nt.mac20.local.37509.downcone.node_1.000004533				
System nodes	000 000	9 C P P P P A S X X I III 1926				
files	Console Project Slaves	Sets View Node Trace Tools Help				
jobs	ct	e2 e4 e3 e5 BiG 44 👷 🖉 🙀				
running						
User ephemeral	Sets Directories	× System jobs × CDT integration unit USB +				
filesToCheck						
jobsToCheck	nodes					
jobsToFire	files	clever.cdt_pl_cdt_sc_cdt_ro.cdt_cl_cdt_to.clever.cdtcdt_ex.cdtcdtcdtcdt_sc.cdt_sc.cdt_ro.cdt_cl_cdt_to.cdt_ex.cdtcdt_ex.cdt_ex.				
jobsToKill jobsToWatch	jobs running	edt_ cdt_ cdt_ clever cdt_k.cdt_sc.cdt_ro.cdt_cl.cdt_to.clever.cdt_ cdt_sc.cdt_ cdt_ cdt_ cdt_ cdt_ cdt_sc.cdt_ro.cdt_cl.cdt_to.				
preempted	D User	clever cdt_cdt_ex_cdt_cdt_cdt_cdt_cdt_sc_cdt_sc_cdt_sc_cdt_sc_cdt_sc_cdt_sc_dever_cdt_cdt_ex_cdt_cdt_cdt_cdt_cdt_sc_				
recent	ephemeral	cat_ro_cat_el_cat_to_clever_catcat_ex_cat cat cat cat_elever_cat_pl_cat_sc_cat_ro_cat_el_cat_to_clever_cat cat_ex_cat cat cat cat				
zipjobs	filesToCheck	clever.cdt_pl_cdt_sc_cdt_ro_cdt_ct_cdt_to_clever.cdtcdt_ex_cdtcdtcdtcdt_sc_cdt_ro_cdt_ct_cdt_to_clever.cdtcdt_ex_				
zippable CDT	jobsToCheck jobsToFire	eat_ cat_ cat_ cat_s				
	jobsToKill	clever, cd_ cdt_e, cdt_ cdt_ cdt_ cdt_ cdt_ cdt_o cdt_o cdt_o cdt_o cdt_o cdte, cdt_ cdt_ cdt_ cdt_ cdt_ cdt_ cdt_ cdt_				
propagatefp	jobsToWatch	cd_o_cd_c_cdt_o_clever_cdt_ex_cdtcdt_ex_cdtcdtcdtclever_clever_clever_cdt_ex_cdtcdtcdtclever_clever_clever_cdt_ex_cdt				
⇒ step	preempted	eat_ catver_clever_clever_cat_ro_cat_o_catcatcatcatcatcatclever_cat_ro_cat_cat_o_cat_o_clever_cat_ro_cat_o				
drc erc	recent zipjobs	cdl_ cdwer. clever. cd.w. cdl_ cdl_ cdl_ cdl_ cleverclevercdl_w. cdl_ cdl_ clever.cdl_w. cdl_ cdl_ cdl_ cdl_ cdl_ cdl_ cdl_ cdl_				
lvs	zippable	cat_to clever cat_ex cat_ cat_ cat_ clever c				
place		cat_sc_cat_ro_cat_ol_cat_to_clever.catcatcatcatcatcat_sc_cat_sc_cat_ol_cat_ol_cat_ol_cat_ol_cat_ex_catcat_ex_catcat				
route	- integration	edt clever_cdt_pi_cdt_sc_cdt_ro_cdt_ci_cdt_sclever_cdtcdt_ex_cdtcdtcdt_cdt_pi_cdt_sc_cdt_ro_cdt_ci_cdt_lo_clever_cdt				
sta synth	propagatefp 					
⇒ unit	drc	000003607 PAILED vw cdt_synth squ 🐓 🚱 🚱 💓 🗐 😌 🚱 🛷 🖡				
ARM	erc					
LEG	lvs	Job Info Execution & Impact Annotations & Properties Why O:Stdout O:Stdour O:Log +				
USB	place	Flags autoforget autoflow doprofile preemptable unsafe scheduled suspended systemiob nojournal barrierinvalid 🔲 😒				
adder	sta					
alu	synth	Host mac20 User kent.staff Fair Share Group /time/users.kent				
cgu chip	⊅ unit	Job Name Job Class Job Proj				
cpu		JOD Mane JOD Class JOD FIG				
dlu	⊅ step	Environment EDADEMO				
emu	Þ unit	Directory S/HOMEL/EDADEMO/work/integration/gigalite/units/ggu/synthesis				
fpu fsu	⊳ tmpgui	Directory \${HOME}/EDADEMO/work/integration/gigalite/units/squ/synthesis				
mem12x128		vw cdt_synth squ				
mem32x512		Command				
mmu		Contrains				
mmu_x mmu_y						
No alerts		Resources License:lic synth				
		Submittime Wed Feb 25 14:28:47 PS In Queue 4m18s Auto Kill 0s				
		Start Time Wed. Feb. 25 14:33:05 PS Exit code 1 Duration 55s				
	No alerts					
		End time lised. Eeb. 25. 14-24-00. REFUS Legal code la Expected Sta				
		PROF S CHANGEDIO-NEW OUTPUT: (FILE).stdout_cdtsy_30472ba2 NEW OUTPUT: (FIL				
		PROF S WHY= Not S PROF S SOLUTION=RAM/20 CORES/1 SLOTS/1 PERCENT/1				

DESIGN ACCELERATION

- Dependency Awareness
 - Jobs only start when all prior dependencies are met
 - Efficient parallel execution

- Maximizes resource utilization
 - Software licenses
 - Computer utilization

c1	c2	c4	c3	c5		BIG	23 💽 🔞
						a di di	j 20 🔤 🗳
Sets Directories		× System:jobs × gui:client:.	de:000002376 +				
✓ System	OO Sys						
⇒ System nodes	000 0	008 PPI	P 🎤 🗚 🐮 💱 🗲 🗙 🛙	I 🔛			547
files							
jobs	cleverc.	.uIt_im cdt_im s	y cdt_im it_im cdt_i	m it cdt sv sv	. cdt sv. sv	dcdt rou	.cdt_sysy.
running			,				· · · · · · · · · · · · · · · · · · ·
⊳ User							
ephemeral	cdt_ex[cd	t_ext <mark>> cd</mark> cdt_drc		everc cleverc	c cleverc Ic	levercrcdt_	extcleverc.
filesToCheck							
jobsToCheck	odt odt or		as as as add and all	adt also hadda	a a a a a a a a dt		ata la adt p
jobsToFire	cdt cdt cd	it_ercadddddle	cd cd cdcdt_ext pl	. <mark>cdt_clo</mark> . Idddo	ddddcle cdt	_PI _PI	sta i cdt_p
jobsToKill		The second second		**************************************			29-18-9-9-9-
jobsToWatch		cdt_ch	cdt_sc cdt_sta rc st	cdt_ro <mark>d.cdt_</mark> d	chi d cdt_s	sc <mark>. c cd c</mark>	dt_sc sc so
preempted							
recent							
zipjobs		cleverc cleverc	cdt_clo cdt_clo cc	dt_to <mark>rc</mark> pddo	d cleverc <mark>)</mark>	¢lo¢dt_	_cloþþ¢lo.
zippable							
		cdt_t_cdt_extrd	o ro cdt_ro i cdl	cdt_drcslever	c ro	it_ro cdt_r	oro
Integration							
😓 mary							
propagatefp	C	dt_sc cc cdt_erc	sc sccdt_tecdt_sc	. <mark>sc</mark> dt_erc i	ocdt_te	o t_to	to
▷ step							
⇔ unit		dt cloc cdt drc cdt			dro yt	drc drc cdt	drc rc :dro
ARM LEG		ar_arda car_ardaar	_clo;lorc cdt_clo				
PCI	_	A REAL PROPERTY	BILLINGER	N /N DA 700 AUGIZ-		458915 1 427	LANK ANY ASIA
USB		cdt cdt_erc rs	cdt_ro) ; i cdt_ro	ro erc cdt	lvs cdt_chir	c rc i i	ta : sta rc s
adder							
alu							
cgu		cdt_tooto_	cdt_too cdt_cd	t_to	cdt_chi		
chip							
cpu		c cdt_drc cleverdc	cdt_drcrcleverc	dt_ext: t	cdt_	to	
dlu					out_		
emu							
fpu		cd cdt_erc erc er	cdt_chi erc i cdt c	dt_erc	c cdt	_drc .	
fsu							
mem12x128			cdt_chi		1 odt	_erc	
mem32x512			out_on		Cut	_0.0	
mmu							
mmu_x			cdt_to				
mmu_y							
qqu			de edt dre				
ring			cc cdt_drc				
squ							
⊳ tmpgui			cdt_erc				

SCALABILITY

- Small memory footprint
 - 1M jobs \rightarrow 1 to 2GB RAM
- Simple to complex flows
 - Scales from tens to millions of jobs
- Built-in high-performance scheduler
 - for faster throughput



OTHER CAPABILITIES

- Change Propagation Control
 - Stop insignificant changes (patented)
- Auto compression of large files
 - Based on flow info
- License / RAM / Cores requirements
- Built-in HTTP server
- Super extensible
 - Thanks to Tcl



MOST CAPABILITIES DEVELOPED FIRST IN TCL

- Examples:
 - Preemption of jobs
 - Distribution of licenses across geographies
 - Reconciliation of license usage
 - AWS monitoring, bursting to cloud
- After extensive testing and learning over many years ...
 - Some capabilities are converted into C++
 - Tcl implementation is "executable specs" for the C++ implementation



IMPACT OF VOV IN EDA

- Most cell phones have a chip designed with VOV
- Largest flows: ~2 million jobs in a single flow (library validation flow)
- Typical chip design flow: 2k to 20k jobs



GROWING FAMILY OF PRODUCTS

- In the beginning, there was FlowTracer
 - New name for Original VOV
 - Generic flows
- ... begat NetworkComputer
 - Just the scheduler part
 - Specialized for many independent jobs
- ... begat LicenseMonitor, LicenseAllocator, WorkloadAccelerator
 - License management layer in NC
 - License distribution across multiple sites
 - Hierarchical scheduling



MANY PRODUCTS: SAME CODE BASIS, DIFFERENT TCL "DRESSES"

LicenseMonitor™	LicenseAllocator™	U U WorkloadXelerator™	WorkloadXelerator Meta™	HERO™	FlowTracer™
 Software license monitoring & management Optimizes license mix Reduces license cost 	 Multi-site license allocator Real-time license adjustments Visibility into license allocations & usage 	 High-performance job scheduler >5x faster than competition Full featured: FairShare, preemption, reservations, etc. 	 High- performance hierarchical scheduler ~6-10x increased throughput Scale-out strategy 	 High-performance scheduler for hardware emulation End-to-end solution: compilation, emulator selection, & regression 	 Platform for developing, executing design flows Reduces design risk & cost Accelerates time to market

 \bigtriangleup

CURRENT COMPLEXITY OF VOV

- C, C++ code: ~360k lines
- Tcl/Tk:
 - .tcl 230k
 - .mod 22k
 - .cgi 62k
- Regression Testing:
 - *.tcl 45k lines
 - *.csh 50k lines

V	
Terminal - more - 102×39	
Terminal – more	
SetInfo(size)	
set lastSize 0 source [] } {	
set lastSize set countrorgotten 0 set countrorgotten 0 } { = = = = = = = = = = = = = = = = = =	
<pre>set counter > 0 } { test for (size) > 0 } { while {SetInfo(size) > 0 } { test forget -elements \$setId =max 1000} } while {SetInfo(kize) = coupt elements; \$errmsg"</pre>	
<pre>set lastSize \$sector set countForgotten 0 while {\$setInfo(size) > 0 } { (catch {vtk.set_forget -elements \$setId -max 1000} errmsg] } { vovError "cannot forget elements: \$errmsg" vovError "cannot forget elements: \$errmsg"</pre>	
vovError break	
<pre>} if ([catch {vtk_set_get \$setId setInfo} errmsg] } { f { [catch {vtk_set_get \$setId setInfo} errmsg] } { # set is gone before we start. Get out. We do not care. # set is gone before we start.</pre>	
if { [catch {vtk_set_get we start. Get Out. No and hefore we start.	
# Set 18 goud = *	
} set deltaSize [expr {lastSize - \$setInfo(size)] set lastication = 0 } {	
<pre>set deltaSize [expr statestor if { \$deltaSize == 0 } { if { \$deltaSize == 0 } { forget \$setInfo(size) nodes from set \$setId"</pre>	
set deltaSize (exp. vizitation) if { sdeltaSize == 0 } { vovMessage "Cannot forget \$setInfo(size) nodes from set \$setId"	
break	
}	
incr countPorgotten \$deltaSize	
set lastSize \$setInfo(size)	
} return \$countForgotten	
et usageMessage "	
DESCRIPTION:	
Forget objects from the project's vovserver.	
The objects are typically nodes (files or jobs) and are are passed by World or by file name.	
The objects may also be users, sets, or selected by option.	
<pre>vovforget \{options\] \{objectId\]</pre>	
: [objectid/]	
skcs: vovicinget \[opicceld\]	
(ons) /(objection)	
forget cojects aror do chy file name are phased by Yord and by file name The objects any also be users, sets, or selected the objects any support of the set of the selected the objects any support of the set of the set of the set of the set of the set of the set of the set of the set o	

LICENSEMONITOR IN ONE SLIDE

- Real time license management
 - Identify underutilized licenses
 - Usage by project, business unit, user, ...
 - License expirations
- Greater operational efficiency
 - Decisions based on real time and historical reports
 - Greater management and designer visibility



NETWORKCOMPUTER: FASTEST JOB SCHEDULER*

- Event driven scheduler
- Small memory footprint
- Full-featured
- Speed record:
 - 10,000 sleep 0 jobs in 8s

* Vs. Slurm, SGE, LSF, ...



TCL API IN VOV

- Total of 410 "vtk" (Vov Tool Kit) procedures
- Example: For Reservation
 - vtk_reservation_create <type> <what> <q> <start> <end>
 - vtk_reservation_get <id> <array>
 - vtk_reservation_update <id> <fieldname> <value>
 - vtk_reservation_delete <id>
- Attempts to use SWIG to port to multiple languages
 - have not yielded results

SCRIPTING LANGUAGE CONTROVERSIES



IS TCL POPULAR?

- Tcl not in list of most popular languages
 - Python is popular
 - Perl less and less

• Yet still Tcl is a de-facto standard in EDA

- <u>https://www.cadence.com/content/cadence-</u> www/global/en_US/home/training/all-courses/82158.html
- And also used in mechanical CAD



CUSTOMERS' REACTIONS TO TCL/TK

- About Tcl
 - "yuk!"
 - "Why don't you have a Perl interface?"
 - "Why don't you have a Python interface?"
 - "Why don't you have a REST API?"
- About Tk
 - "It looks like it is from 1980!"

onsole <u>P</u> roject <u>S</u> laves S <u>e</u> ts		
mac01ac	tasks on mac01ac	143
ts Directories	× Predefined:stuff to do ×gui:client::System:jobs +	
Predefined System	OOS Sterning Guixelient.mac01ac.lan:1879:downcone:System.jobs	
All	OOOIOO⊗⊖I₽₽₽₩IXX÷₽ЩⅡI%IIA	53,7
CDROM		
Class		
FlowTracer unk		
WebSet		
irc		
est		
unit_testing vendors		
	E E	



CONTROVERSY HAS TASTE OF "RELIGION"



TCL OVER PERL: HIGH-LIGHTS FROM HTTPS://WIKI.TCL.TK/1330

- Tcl is simpler. Those without a C/Unix background generally find Tcl syntax far easier to learn and retain.
- Tcl is smaller.

• Tcl is easier to extend, embed, and customize.

- Tcl source code traditionally is a model of lucidity. Perl source code traditionally is dense in magic.
- Tcl/Tk is far more portable than Perl/Tk, and generally more current.

<u>TCP networking</u> is more succinct and less intimidating.

- Tcl's exec, open and socket are gems of accessible and portable functionality, in comparison to the analogous Perl offerings.
- Tcl's unified channel API makes life much easier, particularly on Windows.
- As of spring 2001, Tcl's Unicode [1] capabilities are considerably more mature.
- As of spring 2001, Tcl's threading savvy (read "Tcl and threads") is considerably more mature.
- Subjective stuff: some people find Tcl a better fit to their own sensibilities.
- You can read your own code 6 months after you've forgotten how the program worked.
- (file)event, trace and friends often solve requirements for functionality better than threads.
- Tcl is way ahead of Perl in VFS capabilitities; fuse provides an example of the potential consequences.
- As "Tcl's string handling has been written by paranoiacs", to quote DKF, Tcl is immune to many "format string vulnerabilities".
- Yerlp: this page has been here for years, without adequate emphasis on "The uniqueness of safe interps".
- Deployment of Tcl does not need a full installation. It can be delivered as starkit or starpack



IN THE END, CUSTOMER IS ALWAYS RIGHT

- Best implementation
 - A kludge!

 Then demand subsided around 2008 ... 2010



TCL VS PYTHON

- "What is the best programming language to learn first?"
 - Python is ranked 1st while Tcl is ranked 29th. ...
 - Source: <u>https://www.slant.co/versus/110/5079/~python_vs_tcl</u>
- We still have not done a Python port

REST API: THE NEW DEMAND

- Apparently, cannot enter the Fin-Tech market without it
- Not really a standard
- We are doing it!
- Also we have Python wrapper for REST API

Tcl equivalent, w/o error checking
vtk_node_get 029462592 nodeInfo
puts \$nodeInfo(status)

API to retrieve status of a job	JSON reply
http://HOST:PORT/api/v1/jobs/029462592/status	<pre>{ "startrow":1, "endrow":1, "query":"SELECT status FROM jobs WHERE idint==29462592", "errormsg":"", "columns": [{"col":0,"field":"status"}], "rows":[["VALID"]] }</pre>

MOST IRRITATING THING IN TCL

```
# This example inspired by https://news.ycombinator.com/item?id=4921066
proc ciao {} {
    # A comment with }
    puts "Ciao"
}
ciao
% tclsh ./badtcl.tcl
Ciao
invalid command name "}"
    while executing
"}"
    (file "./badtcl.tcl" line 4)
```



OPEN ISSUES

- Lots of dead Tcl code
 - No code coverage tool
 - Some coverage at "procedure level"

- Tcl Interpreter initialization dominates runtime
 - "nc run hostname" 95% of time is spent initializing Tcl

In 2017, ALTAIR acquires Runtime Design Automation



SUMMARY

- Tcl choice in the beginning was opportunistic
 - Largely a UC Berkeley connection
- Tcl has been instrumental in the success of Runtime Design Automation
 - Several users of Tcl also in Altair
- VOV development continues at Altair
 - Tcl still actively used to increment functionality