

Experiences with Adopting the Tcl/Tk Ecosystem in a University Research Lab Setting

Gunes Koru, Yili Zhang, Urmita Banerjee, Leroy Kim, Clif Flynt, and Stephen Huntley

Health IT Lab
<http://drkoru.us/health-it-lab.html>
Department of Information Systems
University of Maryland, Baltimore County

Friday, October 19th, 2018

Outline

- ▶ Introduction
- ▶ Context
 - ▶ University
 - ▶ Lab
 - ▶ Problem Domain
- ▶ Decision Making
 - ▶ Desktop vs Web Based
 - ▶ Language and Ecosystem
- ▶ Findings and Observations
- ▶ Conclusion
- ▶ Recommendations

Introduction

- ▶ Motivation: Reporting out experiences
- ▶ Benefits: Evidence-Based Adoption of Software Engineering Tools
- ▶ Software engineers value evidence
- ▶ Nevertheless, projects are under time and budget pressure
- ▶ Finishing products and kicking them out the door often takes priority
- ▶ Still, need to take every opportunity to report out to inform future efforts
- ▶ In this case, report our experience of adopting the tools in Tcl/Tk ecosystem
 - ▶ Tcl, Tk, Fossil, and sqlite3
- ▶ This is a qualitative exploration rather than a quantitative one

Context: University

- ▶ Public Research University Setting
- ▶ UMBC now famous for defeating University of Virginia in March Madness
- ▶ One of the research universities under the University System of Maryland
 - ▶ Professors spend time with research (40-50\
 - ▶ Sponsored research is common
- ▶ Diversity is important value for the campus
 - ▶ Students from all backgrounds do not only co-exist but interact and learn from each other
- ▶ Department of Information Systems at UMBC offers PhD, Master's, and Undergraduate degrees

Context: Health Informatics and Technologies Lab at UMBC

- ▶ We conduct research to help individuals and organizations leverage informatics and IT to improve quality of healthcare, improve outcomes, and reduce costs.
- ▶ Publications in reputable and important venues or conferences
- ▶ Research highly interdisciplinary and applied; directed to solve real life problems
- ▶ Research means extending the boundaries of knowledge
- ▶ In this environment, we write a lot of code. Coding is a means to an end
 - ▶ Code for us or others to read, understand, and execute
 - ▶ Code for others to execute
- ▶ Interest in publishing and commercializing
- ▶ Currently six programmers, two professional programmers

Context: Development in Lab

- ▶ Rapid development becomes necessary to accommodate and support learning cycles
- ▶ Prototypes are often usable products deployed in the sponsoring agency
- ▶ Development resources come from the research budgets
- ▶ Master's and PhD students from various backgrounds participate in development
 - ▶ Informatics, Information

Technology, Engineering, Computer Science

- ▶ Not only computer science
- ▶ Programming knowledge from at least one programming course and one database course
- ▶ Languages: C and standard query
- ▶ OS: Linux shop. Closed network of Debian workstations and Debian and Windows servers
- ▶ Linux utilities and editors Emacs and Vim, etc.

Context: Development Domain

- ▶ Healthcare
 - ▶ Highly regulated domain
 - ▶ High sensitivities on privacy and confidentiality
- ▶ Healthcare Administration
 - ▶ Medicaid services management
 - ▶ Data analytics
 - ▶ Data quality
- ▶ Development of tools for
 - ▶ Data analysis and reporting
 - ▶ Data quality improvement

Decision Making: Desktop vs Web-Based

- ▶ Ease of use slightly favors Desktop
 - ▶ Powerful and established GUI widgets
- ▶ Maturity of the underlying platforms slightly favors desktop
 - ▶ Browsers change more often
- ▶ Performance (in terms of response time) slightly favors Desktop
- ▶ Security slightly favors Desktop
 - ▶ Web application frameworks still problematic
- ▶ Ease of Development slightly favors desktop
 - ▶ Many web frameworks require students to learn multiple languages: HTML, javascript, CSS, and a server-side language (java, python, ruby, or Tcl)

Ease of deployment slightly favors web-based

- ▶ Starkits and starpacks looked promising

Decision Making: Language

- ▶ We consider Tcl, Ruby, Python, and Java
- ▶ Java was not chosen because
 - ▶ Non-scripted nature makes rapid-development difficult
 - ▶ Language not easy to learn and use even after students take a class
- ▶ Ruby and Python not chosen
 - ▶ Their widget libraries not as easy as Tk
 - ▶ There are wrappers based on Tk
 - ▶ Generally, comes with a lot of libraries generating dependencies
- ▶ Tcl chosen because
 - ▶ Embedding C code is easy
 - ▶ Tk was made for Tcl
 - ▶ Database connection with sqlite3 seemed straightforward
 - ▶ Seemed different but easy to learn
 - ▶ Not necessarily OO
 - ▶ Mature: Still changing but not experimental
 - ▶ Can run on multiple platforms

Decision Making: Ecosystem

- ▶ `sqlite3`:
 - ▶ Knew about it but took a closer look when considering `sqlite3`
 - ▶ Chosen because of its serverless and high-performance nature
 - ▶ Offered an opportunity to keep healthcare data local at the individual's desktops or on the organization server
- ▶ Fossil
 - ▶ Decision in this case was easier because I knew git did not work
 - ▶ git was difficult to understand and use for our students
 - ▶ Fossil offered web-based visualization and ticketing
 - ▶ Based on `sqlite3` it is extensible. You can see and modify all tables
- ▶ Overall, we obtained a Visual Basic + Access type of environment I had in mid 90's, but within a Unix environment

Findings and Observations

- ▶ Effective ecosystem for developing small applications rapidly
 - ▶ Student with different backgrounds learned within a week
- ▶ Professional programmers joined the projects easily and communicated over Fossil
- ▶ Students liked Tk and the availability of widgets
- ▶ Extended the widgets (Clif's rich-text editor widget)
- ▶ Students also developed ineffective and inefficient user interfaces
 - ▶ Ability to quickly deploy user interfaces need to be augmented by paying attention to timeless user interface design principles

Findings and Observations - II

- ▶ However, our systems grew over time
- ▶ Standalone to client-server switch was quickly made by redefining the db command and using comm package
 - ▶ We still left the data processing to client and stored client-specific data on the desktop
 - ▶ Server database included data shared in the organization
 - ▶ With some database adjustments (WAL), this server solution accomodated low traffic requests
 - ▶ Installation program installs and runs the program as a service in windows and linux.

Findings and Observations - III

- ▶ Students report there are lack of resources
 - ▶ They have the books, wiki, documentation
 - ▶ However, I figured they learn differently – good or bad arguable
 - ▶ They want to look at examples on stackoverflow
 - ▶ There were many instances when student got stuck for a couple of days on a technical problem
 - ▶ I found it was in the documentation

Findings and Observations - IV

- ▶ Perceived popularity problem
 - ▶ Python picked up in many domains including scientific domains
 - ▶ Easier to make a case for Python and convince others
 - ▶ Tcl needs more convincing
- ▶ We developed an autoupdate feature which regularly checks for updates and updates the main starkit for the program (presented by Zhang)
 - ▶ This feature in sdx.kit had some bugs that we had to fix, and it took time to develop
 - ▶ Now it works, it also performs database migrations (presented by Banerjee)

Findings and Observations - V

- ▶ Security is generally good because we avoid clickjacking, session stealing attacks by not developing web software
- ▶ However, often we had to use someone else's tclkit.
- ▶ Tools for generating Tclkits were complex and it required us to rely on other's code
- ▶ This was unacceptable for our clients
- ▶ Steve Huntley developed a mechanism for generating tclkits presented in this conference
- ▶ Predictive modeling capabilities going beyond regression are unavailable in Tcl
 - ▶ Python has advantages in this area; R is the best but its ecosystem is mostly GPLed

Findings and Observations - VI

- ▶ Some students preferred to batteries-ready approach of Python where you can find many libraries
- ▶ I think less reliance and dependencies to external packages is actually better
 - ▶ At least in this project, we were able to code our own solutions for things like
 - ▶ Authorization and authentication
 - ▶ Database migration
- ▶ Many external dependencies evolving at a different rate, some becoming unavailable, or breaking

the previous contracts and agreements is a huge headache in software development

- ▶ Students needed a package manager – one is available but we did not use it because it worked for Active Tcl

Conclusion

- ▶ A university lab and small business has many similarities: For example, collective code ownership
- ▶ Tcl ecosystem supports developing software in a small setting by facilitating rapid prototyping to achieve and demonstrate success
- ▶ It is appropriate for building intellectual property for commercialization because the software solutions in this ecosystem mostly use BSD license
- ▶ Fossil is hosted internally in the lab. Its features for managing source code and ticketing worked without any issue for three years

Conclusion - II

- ▶ Overall, we were able to successfully finish the prototypes with the three students in the lab within time and budget
- ▶ Demonstrated success brought additional funding which allowed us to work with Clif and Steve
- ▶ Of course, success has many different forms
- ▶ These experiences do not mean
 - ▶ Other tools or ecosystems cannot be adopted successfully
 - ▶ Or this ecosystem will lead to success each time
- ▶ Nevertheless, it reports a successful experience at least one university setting

Recommendations

- ▶ Research groups in academia and industry should consider Tcl/Tk ecosystem
- ▶ To us it looks like, promotion to increase popularity is the biggest need and growing the community is the most immediate concern
 - ▶ Creative solutions are needed since everyone is busy
 - ▶ Students mentioned, it would be good to have a
 - ▶ Better looking website
 - ▶ More organized wiki
 - ▶ More examples
 - ▶ Video tutorials highlighting the small ways in which Tcl is used
 - ▶ We should also approach this problem on the business side
 - ▶ By demonstrating success, we can bring in projects with larger budgets that hire more programmers

Recommendations - II

- ▶ For those who want to go fast, availability of libraries will be important
- ▶ Web application development framework needs to be developed and supported
 - ▶ A lot of need for rapid application development need in this area – even though we intentionally developed client-server desktop applications
- ▶ Tcl-based statistical learning packages would be extremely useful for pure Tcl applications
 - ▶ Currently, there is a need to rely on Python and R

Thanks

- ▶ We would like to say a big thank you to all those who worked on the Tcl/Tk ecosystem and made these solutions available to us
- ▶ Questions/Answers?