Tcl/GSoC 2011

Andreas Kupries ActiveState Software Inc. 409 Granville Vancouver, BC CA
andreask@ActiveState.com

ABSTRACT

As in previous years the Tcl Community took again part in Google's Summer Of Code[1], under the auspices of the Tcl Community Association[2].

1. OVERVIEW

Google's Summer Of Code[1] (short: GSoC) is a global program that offers student developers stipends to write code for various open source software projects. The Tcl Community participated again this year, for the fourth time in a row. As in previous years this participation was managed by the Tcl Community Association[2] (short: TCA) as the mentoring organization, the same organization which runs the US Tcl Conferences, like this one.

The main entrypoint to the program for the community itself can be found on the Tcler's Wiki [8].

2. PAST

Starting in 2007, we applied five times, and were accepted four times, with only our very first application not getting accepted by Google. This year was our fifth application and fourth participation.

Through negotiations by previous program administrators we usually got just shy of 10 slots for our projects[8], with our usual argument the fact that the Tcl Community Association[2] acts as an umbrella for smaller organizations with Tcl related projects. An example for this is the aMSN chat client[3]. This dropped a bit last year. The full statistics for the past years[4] are shown in table 1 below.

	2006	2007	2008	2009	2010
Students	630	950	1125	1000	1026
Organizations	102	>130	175	150	150
Average	6.18	<7.31	6.44	6.66	6.84
Tcl	-	-	9	9	7

Table 1: Statistics of past four years

3. PRESENT

Matthew Burke[6], our program administrator of the past years served as a backup this year, with me moving from backup to the main position.

In the slot allocation game/roulette we got seven slots, the same as last year, and a similar tick above the average, as can be seen in table 2 for the final statistics[4] below.

	2011
Students	1115
Organizations	175
Average	6.37
Tcl	7

Table 2: Final statistics for 2011

Our projects for this year are listed in table 3 [1] on the next page, with larger descriptions in the upcoming sections. The overall timeline we followed is shown in table 4.

3.1 W3C Widgets Compliant Content Packaging for XoWiki/OpenACS

[10] by Michael Aram
Mentor: Gustaf Neumann

Content packaging has the purpose to provide a platform and renderer-independent interchange format for a set of resources (content). Content packaging has high relevance for content exchange, i.e. for reusing and sharing content among different platforms. In the area of learning management systems content packaging has a long tradition, where a range of standards has evolved over the last years and decades (for example the SCORM or Common Cartridge, which both profile a generic packaging format). Although content packaging is getting increasingly more important as evermore learning resources become available, the e-learning community is rather small compared to the overall web development community. With the rise of rich Internet applications and mobile apps, several alternative approaches for rich content distribution have been developed outside of the e-learning communities: small, web-based applications/web-content packages commonly referred to as "widgets". One reason for the high interest in widgets is that major vendors developed user agents integrated into operating systems and web-based platforms. However, compared to the respective standards in the e-learning world, these new packaging standards are in several respects even more restricted, i.e. less powerful. On the other hand, they

[1]The same table can also be found at[9]

Student	Project	Mentor
Michael Aram	W3C Widget Packaging Standard Compliant Content Packaging Infrastructure for OpenACS	Gustaf Neumann
Krzysztof Kwasniewski	Debugging tools for NRE	Miguel Sofer
Michal Poczwardowski	Tcl Plugin for Netbeans	Arnulf Wiedemann
S. M. Saurabh	Extending and Evolving CRIMP	Kevin Kenny
Lars Hellstrom	stasher: Tcl_Obj intRep as cache also at script level	Donal Fellows
George Andreou	Create a binding to the Hwloc library	Andreas Kupries
Saurabh Kumar	Micro-benchmarking extension: access to CPU performance counters	Edward Brekelbaum

Table 3: 2011 Projects, Students, and Mentors

	January 24	Program announced.
Organizations	February 28	Organization application window opens.
	March 11	Deadline for organization applications.
	March 14-17	Submission review.
	March 18	Publication of accepted organizations.
Students	March 18-27	Discussion of ideas between students and organizations.
	March 28	Student application wind opens.
	April 8	Deadline for student applications.
	April 10-21	Organizations rank and review student applications.
	April 22	Ranking/scoring deadline. Mentor sign-up deadline.
	April 25	Publication of accepted students.
Coding		Community bonding sudents to mentors.
	May 23	Coding period starts.
	July 11-15	Mid-term evaluations.
	August 15	Soft-end of coding. Scrub code, test, document.
	August 22	Hard end of coding period.
Post-Mortem	August 22-26	Final evaluations.
	August 29	Final results announced.
	August 30	Students can begin submitting the require code samples.
	October 22-23	Mentor Summit at Google.
	October 24-28	Tcl Conference At Manassas

Table 4: 2011 Timeline

provide some aspects, which the e-learning standards have not tackled at all yet (e.g. Device APIs). In sum, the investigation of the overlaps and differences between these worlds seems to be worthwhile.

In general, the content packages described so far are typically deployed as a single (archive) file and hold some form of configuration / manifest file describing the content. In addition to that, the packaged content might be allowed to use some form of API provided by the run time environment. Hence, from a technical perspective, there is a range of aspects within this ”content packaging field” that could be abstracted from the different standards into a generic implementation. As a consequence, the standard specific content packages could be generated by specializations of this universal component.

The main goal of the project was to write an OpenACS package for generating W3C widgets out of learning materials that reside within OpenACS and its major content authoring tool XoWiki. Moreover, the package aims to provide means to also generate content packages adhering to other important (vendor specific) formats. As a result, the OpenACS package ”xocp” has been created in the course of the GSoC. In a nutshell, the xocp package provides an infrastructure for generating content packages that comply with various specifications, for example ”W3C Widgets”, ”Opera Widgets” or ”SCORM”. In general, ”xocp” acts as a ”backend” API to be used by other packages or by developers. In particular, ”xowiki” is considered to be the main authoring environment for end users and a focus is put on the packaging of XoWiki-based content (e.g. wiki text books).

3.2 Debugging tools for NRE

[11] by Krzysztof Kwasniewski
Mentor: Miguel Sofer

The Tcl core has become a lot harder to debug since NRE’s adoption. The problem is intrinsic to the nature and goals of the NRE: C keeps a stack of “who called me” frames, NRE does its best to replace it with a stack of “who do we call next” callbacks. But

1. Most debugging tools like gdb are designed for C.

2. Bug analysis requires understanding the path that leads to failure, not so much what would have happened after that.

Tcl on the other hand is a lispy language built onto C - a 100procedural substrate. The NRE is a mechanism that enables features like coroutines and proper tailcalls.

All's well when all's well, but when things go boom in the night the tools designed for C get lost too. Today the only way to understand what is happening is a tedious manual inspection of the NRE stack, which allows the deduction of the execution history if there were no intervening tail-calls. It is hard work that requires a lot of concentration and knowledge of the internals.

The extension to Gdb developed over the summer has the following capabilities:

1. Allows inspection of the NRE stack in a way more less similar to what Gdb offers for the C stack.
2. Parses and displays Tcl_Obj structures as strings. When a Tcl_Obj has its string representation, that one is used, however sometimes only the internal representation is present and in that case for the most common Tcl_Obj structures the string representation is created on the basis of the internal representation.
3. Displays the contents of the Tcl_HashTable structure.
4. Currently the best way of extending Gdb requires using Python and that is why my extension was developed with that language. Many Gdb users would however much rather use another languages of their choice for developing extensions to Gdb, like Tcl. For that reason I have also developed IO channels enabling a user to extend Gdb with virtually any programming language.

For more information on the capabilities and limitations of my project, please read the documentation available in the project's repository [12].

The code will be maintained until at least the end of the next edition of the GSoC program unless earlier many significant changes are introduced to the extension by someone else, in that case I may not be able to maintain the code. During the time of the maintenance I will fix all the bugs found in the extension.

I will try to commit some time for adding new functions to the extension providing I get any ideas for further development from the community - in this sense the future of this extension also depends on the Tcl community. The exact amount of time spent on the further development is hard to assess.

To sum the above up - I will try to support this extension for at least a year and providing someone will use and need it, this period may be extended.

3.3 Tcl Plugin for Netbeans

[13] by Michal Poczwardowski
Mentor: Arnulf Wiedemann

Tcl is available as a plugin for Eclipse, it would be helpfull to also have the same functions within netbeans.

The student would research how to write a plugin for netbeans and how to use available features like execution trace to drive debugging for Tcl within netbeans.

Netbeans for Tcl has been successfully brought to version 1.0, which can be downloaded from the netbeans plugin page. The student (Michal) has proposed to do further work on the Plugin, especially for Itcl which is not really functioning yet. And he will do bug fixes. Additionally he plans to add autocompletion for editing and maybe other features like debugging for itcl in javascript.

3.4 Extending and Evolving CRIMP

[14] by S. M. Saurabh
Mentor: Kevin Kenny

CRIMP, aka "C Raster Image Manipulation Package", is a package for image processing with Tcl. While it already provides the most basic algorithms it has not much of advanced or very advanced algorithms.

This project aimed at extending the package with more algorithms.

The project resulted in the addition of edge detection algorithms (Canny Sobel/Deriche), noise generators of various types, Wiener-based denoising, and the beginnings of FFT/LPT-based affine image registration. As part of the latter we got implementations of the log-polar transform (LPT) and a new image type for "complex" images, with associated operations.

Next up are the completion of the image registration functionality, and working on integrating the new pieces with the changes coming from my own work with the not-yet-released critcl v3.

3.5 stasher: Tcl_Obj intRep as cache also at script level

[15] by Lars Hellström
Mentor: Donal Fellows

The aim of this project is to make Tcl's internal mechanism of dual-ported Tcl_Obj's (which cache information about a value for fast access) available at the script level.

The stasher package has reached a point of being fully functional and stable. It is possible that further development will happen, in particular concerning making use of TclOO classes, but first more experience should be gathered of using the current interface; it may well turn out to be sufficient for what one would do in practice. The immediate focus will instead be on producing utility commands that make use of stashers to accelerate argument parsing.

3.6 Create a binding to the Hwloc library

[16] by George Andreou
Mentor: Andreas Kupries

This is a larger idea spun out of the idea for extending CRIMP, notably the ticket proposing to enhance the package's performance through parallelization and/or threading. As a foundation for that we need some introspection into the machine Tcl runs on, i.e. number of processors, cores per processor, threads per core, etc.

The HWloc library, aka "Portable Hardware Locality" provides all this information, and more. As such it is natural to create a Tcl binding for it to lift the information it provides up to the level of scripts.

The project was successful, creating a draft binding to the most important pieces of hwloc's functionality, i.e. creating, reading, and writing of topologies, navigation, plus CPU and memory binding of threads and processes.

Currently still left is the writing of proper documentation and test suite before a 0.1 release can be made. When that is done people working with the Thread package, thread pool implementations and the like should start working with the binding, to determine if the API as is is useful to them, like for sizing a thread pool to the available resources or multiple thereof, pinning threads of the pool to processing units, etc.

3.7 Micro-benchmarking extension

[17] by Saurabh Kumar
Mentor: Edward Brekelbaum

The goal of this project is to design and implement a Tcl extension with commands to interact with the CPUs harware counters. Initially the goal is to code an extension that works under Linux using its infrastructure for performance counters.

If time permits, we will research the possibility of porting the extension to Windows and/or OSX. This will entail finding out about interfaces analogous to the Linux Performance Counters, (possibly) redesigning parts of the extension's C-code so that it can be configured to work with the three different APIs, and coding a portable extension.

The project was a success and the support for the Linux and Windows operating systems is in working condition. For Linux, the "Performance Application Programming Interface" library [18] has been used to gain access to the CPU counters and the results are very precise and accurate for the latest Linux kernels. The support for Microsoft Windows is based on the "Performance Inspector" [19] and is not as accurate as the Linux version. The typical events which can be traced using the extension include various cache events, number of different type of CPU instructions, total number of CPU cycles etc.

Currently we do not see an obvious way to implement the utility for Mac OS and we will try to find a way to go about it in future. We will also try to make the output data for Windows OS more precise.

4. FUTURE

For a mentoring organization Google's Summer Of Code[1] is pretty much a year-round operation. Simply look back at the timeline (Table 4 on the previous page).

Next up in this cycle is starting the preparations for 2012, i.e., updating our application[7], restarting the collection of project ideas, and reaching out to prospective students and mentors in general. The last point is one of the more important things to do, not only for us as a mentoring organization, but for the Tcl community at large too, to make a general effort of spreading awareness of Tcl and its community as a viable (and fun) scripting language which doesn't has to hide.

APPENDIX

A. REFERENCES

[1] Google, GSoC.
`http://socghop.appspot.com/`

[2] Tcl Community Association.
`http://www.tclcommunityassociation.org/`

[3] aMSN.
`http://www.amsn-project.net/`,
`http://wiki.tcl.tk/12783`

[4] GSoC Statistics. `http://code.google.com/p/google-summer-of-code/wiki/ProgramStatistics`

[5] Andreas Kupries.
`http://wiki.tcl.tk/26`

[6] Matthew Burke.
`http://www.seas.gwu.edu/~mmburke/`

[7] Tcl Org Application 2011.
`http://wiki.tcl.tk/27864`

[8] Google Summer Of Code on Tcl'ers Wiki.
`http://wiki.tcl.tk/25801`

[9] GSoC 2011 Executed Projects.
`http://wiki.tcl.tk/28291`

[10] W3C WPS Compliant Packaging Infrastructure for OpenACS, Michael Aram.
`http://wiki.tcl.tk/28188` Idea,
`http://wiki.tcl.tk/28538` Execution

[11] Debugging tools for NRE, Krzysztof Kwasniewski.
`http://wiki.tcl.tk/28091` Idea,
`http://wiki.tcl.tk/28334` Execution

[12] NRE Source Repository
`http://chiselapp.com/user/krzykwas/repository/nredebug1/index`

[13] Tcl Plugin for Netbeans, Michal Poczwardowski.
`http://wiki.tcl.tk/28110` Idea,
`http://wiki.tcl.tk/28292` Execution

[14] Extending and Evolving CRIMP, S. M. Saurabh.
`http://wiki.tcl.tk/27866` Idea,
`http://wiki.tcl.tk/26953` Execution

[15] stasher: Tcl_Obj intRep as cache also at script level, Lars Hellstrom.
`http://wiki.tcl.tk/28288` Idea, Execution

[16] Create a binding to the Hwloc library, George Andreou.
`http://wiki.tcl.tk/28167` Idea,
`http://code.google.com/p/tcl-hwloc` Execution

[17] Micro-benchmarking extension: access to CPU performance counters, Saurabh Kumar.
`http://wiki.tcl.tk/28157` Idea,
`http://code.google.com/p/tcl` Execution

[18] Performance Application Programming Interface
`http://icl.cs.utk.edu/papi`

[19] Performance Inspector
`http://perfinsp.sourceforge.net`