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 funded and operated by Google that offers student developers stipends to write code for various open source software projects. The Tcl Community participated again this year, for the third 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.

The main entrypoint to the program for the community itself can be found on the Tcler’s Wiki[4].

2. PAST
Starting in 2007, we applied three times, and were accepted two times, with only our very first application not accepted by Google. This year was our fourth application and third participation.

Through negotiations by previous program administrators we usually got just shy of 10 slots for our projects[4], 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]. The full statistics for the past years[5] are shown in table 1 below.

<table>
<thead>
<tr>
<th></th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
<th>2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students</td>
<td>630</td>
<td>950</td>
<td>1125</td>
<td>1000</td>
</tr>
<tr>
<td>Organizations</td>
<td>102</td>
<td>&gt;130</td>
<td>175</td>
<td>150</td>
</tr>
<tr>
<td>Average</td>
<td>6.18</td>
<td>&lt;7.31</td>
<td>6.44</td>
<td>6.66</td>
</tr>
<tr>
<td>Tcl</td>
<td>-</td>
<td>-</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 1: Statistics of past four years

3. PRESENT
Matthew Burke[8], our program administrator of the past this year served in that capacity this year as well, with me as backup.

In the slot allocation game/roulette we got seven slots, a loss of two compared to previous years. We considered this to be quite higher than the average, because of Google originally dialing down the total number of slots to 950 in general. In the final statistics[5] we are only average however, as can be seen in table 2 below.

<table>
<thead>
<tr>
<th></th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students</td>
<td>1026</td>
</tr>
<tr>
<td>Organizations</td>
<td>150</td>
</tr>
<tr>
<td>Average</td>
<td>6.84</td>
</tr>
<tr>
<td>Tcl</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 2: Final statistics for 2010

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

3.1 Base JIT compiler for Tcl

This project aimed to create a base JIT compiler for the Tcl core, translating bytecodes to (x86) machine code on demand in order to improve the language’s performance.

After GSoC’s ending, the project still requires several future improvements to be able to land in the Tcl core or be usable in the wild. The first two ideas on the list are: proper register allocation and a finer grained intermediate representation (IR). Currently, the encoding of a very simplified form of the incr command requires 51 x86 opcodes, with the current IR this is all done inside a single clause of a switch statement. The idea is to transform this mid-level IR to a lower level one that is closer to a form of a generic-RISC architecture, and then develop and run instruction selection on it. Currently, only toy programs can be tested given the ad-hoc, or inexistent, register allocator, so this is another priority. This second “future” idea is already being developed. Another idea involves the application of another kind of IR, the SSA. It has been used by some known projects: LLVM[18], gcc[19], Jalapeño (or Jikes RVM)[20] and is mentioned as a way to allow some optimizations[21] to take

TABLE 1

<table>
<thead>
<tr>
<th></th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
<th>2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students</td>
<td>630</td>
<td>950</td>
<td>1125</td>
<td>1000</td>
</tr>
<tr>
<td>Organizations</td>
<td>102</td>
<td>&gt;130</td>
<td>175</td>
<td>150</td>
</tr>
<tr>
<td>Average</td>
<td>6.18</td>
<td>&lt;7.31</td>
<td>6.44</td>
<td>6.66</td>
</tr>
<tr>
<td>Tcl</td>
<td>-</td>
<td>-</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 1: Statistics of past four years

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Tcl ’2010 Oakbrook Terrace/Chicago, IL, USA
place in an efficient manner. Other important improvement is allowing the compilation of non-leaf procedures, which re-
quires a study between the interaction of the JIT compiler and the Tcl VM in order to properly manage the VM state.

3.2 MSNP2P refactoring for aMSN

Continuing the series of aMSN projects done in previous summers, this year the current MSNP2P im-
plementation was rewritten to use snit[17]. This was done to overcome the problems with the current code base, which was complicated, with code scattered over different files and mixing several different functionalities in the same procedure.

As a bit of background, MSNP2P is the protocol used for P2P data transfers, taking care of webcam video, file transfers, etc.

This rewrite of the current code base will also help with implementing MSNP2Pv2, which is the version used in the latest builds of Windows Live Messenger and supports concurrent login from different locations. Implementing MSNP2Pv2 will in turn enable full support of protocol version 18 and audio/video calls.

3.3 OpenACS Abstraction Layer

The basic idea behind this project was to rewrite the server integration of the

3.4 OpenStreetMap package and editor

A set of packages for the convenient handling OpenStreetMap data was created, both for its storage and rendering into a Tk canvas. On top of that a visual editor for OpenStreetMap data was made, using Tk for the interface and allowing the creation and editing of ways and tags.

Future work may include expansion of the editor’s modular approach into a full plugin system.

3.5 SCORM Compliant Run-Time Environment for OpenACS

This was a follow-up project to last year’s successful SCORM project, which focused on input, export, and presentation of SCORM packages. During the last project it turned out that implementing the runtime environment (RTE) is a relatively big task. As the RTE implemented only the most basic RTE-API functions it was therefore only capable of importing simple packages.

This year’s project overcame these limitations, implement-
ing a fully standard conformant RTE according to the min-

Table 3: 2010 Projects, Students, and Mentors

<table>
<thead>
<tr>
<th>Table 4: 2010 Timeline</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Organizations</th>
<th>Students</th>
<th>Coding</th>
<th>Post-Mortem</th>
</tr>
</thead>
<tbody>
<tr>
<td>March 8</td>
<td>March 18-29</td>
<td>May 24</td>
<td>August 16-20</td>
</tr>
<tr>
<td>March 12</td>
<td>March 29</td>
<td>July 12-16</td>
<td>Final evaluations.</td>
</tr>
<tr>
<td>March 13-17</td>
<td>April 9</td>
<td>August 9</td>
<td>August 23</td>
</tr>
<tr>
<td>March 18</td>
<td>April 10-21</td>
<td>August 16</td>
<td>August 30</td>
</tr>
<tr>
<td></td>
<td>April 21</td>
<td></td>
<td>October 23-24</td>
</tr>
<tr>
<td></td>
<td>April 26</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

February 8 Program announced.

March 8 Organization application window opens.

March 12 Deadline for organization applications.

March 13-17 Submission review.

March 18 Publication of accepted organizations.

March 18-29 Discussion of ideas between students and organizations.

March 29 Student application window opens.

April 9 Deadline for student applications.

April 10-21 Organizations rank and review student applications.

April 21 Ranking/scoring deadline. Mentor sign-up deadline.

April 26 Publication of accepted students.

May 24 Coding period starts.

July 12-16 Mid-term evaluations.

August 9 Soft-end of coding. Scrub code, test, document.

August 16 Hard end of coding period.

August 16-20 Final evaluations.

August 23 Final results announced.

August 30 Students can begin submitting the require code samples.

October 23-24 Mentor Summit at Google.

OpenACS’ web framework to use Colin’s Wub web server packages, extending OpenACS’ portability.
imal compliance level defined by ADL. In future subsequent projects reaching even higher compliance levels, which enable further SCORM features, could be a sensible goal. Also, incorporating newer versions of the SCORM or alternative packaging standards (e.g. SCORM 2004, IMS CC) should be considered.

3.6 Tcl state machine back-end module for XMLVM

Originally planned was the creation of a back-end for the XMLVM bytecode cross-compiler, enabling all languages producing JVM or .Net bytecodes to be translated to and executed by Tcl.

This project mutated over the summer and became a “Tcl Bytecode Assembler” instead, extending the Tcl core with a command taking a textual description of Tcl bytecodes and creating the executable internal representation.

3.7 Themed Tk on Unix

This project first and foremost updated the Qt and Gtk bindings for Themed Tk to work with the latest versions of their respective GUI toolkits. Additionally some shortcomings and general code quality issues in the previous versions of those packages were addressed.

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 2011, i.e., updating our application [6], 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 have to hide.

APPENDIX

A. REFERENCES