3-2007

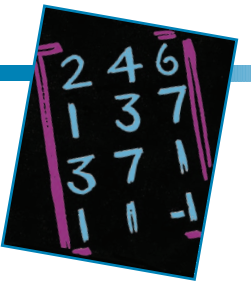
Project Hosting: Expanding the Scientific Programmer's Toolbox

George K. Thiruvathukal
Loyola University Chicago, gkt@cs.luc.edu

Recommended Citation

George K. Thiruvathukal, "Project Hosting: Expanding the Scientific Programmer's Toolbox," Computing in Science and Engineering, vol. 9, no. 2, pp. 70-75, Mar./Apr. 2007, doi:10.1109/MCSE.2007.36

Editors: George K. Thiruvathukal, gkt@cs.luc.edu
Konstantin Läufer, laufer@cs.luc.edu



PROJECT HOSTING

Expanding the Scientific Programmer's Toolbox

By George K. Thiruvathukal

Emerging technologies such as free and open source project hosting will hopefully broaden interest in scientific programming in a business context.

At the risk of sounding somewhat controversial, I'm going to make a generalization that scientific programming projects today could do a lot better when it comes to their public presence and the availability of their results. A recent article in *The Economist* cites a report that claims the "concepts, tools, and theorems [of computer science] have become integrated into the fabric of science itself" (www.economist.com/science/displaystory.cfm?story_id=5655067). Although I believe this is true for the most part, the skeptic in me wonders whether the computer science community might be getting a bit too much credit here.

True, the concepts of computer science—the academic discipline, that is—play a major role in the "new" science, but theorems and tools being the creation of computer scientists alone might be a bit of a stretch. For starters, it's debatable whether computer science theorems are entirely an invention of the field itself. Euclid's famous method for computing the greatest common divisor, for example, predates the notion of automatic calculation and computing. By all accounts, it's one of the first algorithms and theorems that are still relevant to computer science, but it wasn't the invention of a computer scientist. Likewise, tools for computational science include compilers, libraries, and development or modeling environments, but many modern software engineering tools continue to elude computational science and its practitioners. Yes, we need compilers, an actual operating system, parallel middleware (optional), and good hardware to do serious computational science, and when viewed from this perspective, our everyday tools seem to be the byproduct of computer scientists' work. But I think this traditional view is about to fade into the background in the face of larger building blocks (components) and integration into the mainstream.

Accordingly, I'm convinced that the computational scientist's toolkit must expand. Because computational science is becoming increasingly collaborative, the projects' ability to organize, collaborate, and disseminate results effectively will be the benchmark against which we'll measure future success. Simply producing models, writing code, and publishing the results in a journal (and nowhere else) won't be the model in a network-centric future, in which practitioners will place greater emphasis on being able to reproduce and integrate actual results. For this reason, the time is now for projects to embrace the concept of project-hosting tools, which the free and open source software (FOSS) community already uses extensively. This community of amateur and professional computer programmers isn't necessarily made up of computer scientists, but the people in it are behind many of the most-talked about computing applications (such as Mozilla Firefox, and Open Office), languages (Python, Perl, and Ruby), and operating systems (Linux and FreeBSD), just to name a few.

Toward the goal of expanding the computational scientist's toolbox, I'm increasingly convinced that all computational science projects should focus on using project-hosting tools to enhance their impact. Therefore, I'm working feverishly to transition virtually all my programming projects (computer and computational science) to FOSS project-hosting sites such as Google Code, which is what I cover in the rest of this article. On the whole, Google Code is designed strictly for open source software and to promote best practices for open source software development.

A New Home for Code Examples

To prepare for this article, I decided to create a new project at Google Code, which my coeditor Konstantin Läufer and I will use to distribute most code examples in upcoming installments of this department. The URL http://code.google.com/p/cise/ will be our permanent home at Google Code, and it's where you can find most—if not all—the code examples in this column.

I want to stress that several personal considerations motivated our choice of Google Code. As noted in the

 Copublished by the IEEE CS and the AIP 1521-9615/07/$25.00 © 2007 IEEE

Project-Hosting Alternatives

SourceForge (www.sourceforge.net) is one of the most established free and open source software (FOSS) project-hosting sites. It has much in common with Google Code, but it's arguably a bit more mature and established. Its robust distribution model ensures reliable download speeds and provides the capability to track the number of file downloads (which can be useful for understanding a project's impact).

Savannah (http://savannah.gnu.org) is aimed at the development, distribution, and maintenance of GNU software. Its companion site (http://savannah.nongnu.org) hosts free software projects that aren't part of the GNU project but that run on free platforms. Savannah focuses primarily on projects that follow the GPL and its variants, so you must be a GPL advocate or be clearly motivated to use it. GPL has a virtuous licensing scheme, but I don't necessarily think "one size fits all" when it comes to FOSS licensing.

BerliOS Developer (www.berlios.de) is a free service to open source developers that offers easy access to the best in Concurrent Versioning System and Subversion, mailing lists, bug tracking, message boards and forums, task management, site hosting, permanent file archiving, full backups, and total Web-based administration.

"Project-Hosting Alternatives" sidebar, plenty of options exist, but we picked Google Code for its "less is more" interface, its current (and we hope, continued) lack of intrusive advertising, and the ability for anyone to create a project without going through a lengthy approval process. Even if you don't plan to use Google Code, the ideas presented in this article apply to all major project-hosting sites.

Google Code is so new that it's still a work in progress in terms of its feature set, but it's a terrific start and is highly stable. Figure 1 shows a screenshot of our new main page. A quick perusal of the tabs gives us Project Home, Downloads, Wiki, Issues, Source, and Administer, each of which sheds a great deal of light on the typical nature of any FOSS project. In addition, the page has a summary of important items for casual visitors, such as the FOSS license used and the keywords or labels that apply to this particular project. We can also add links to the developers' home pages or any other related work.

Let's begin our tour. My aim is to give you an idea about what's going on here so that you can incorporate the virtues of project hosting into your own projects.

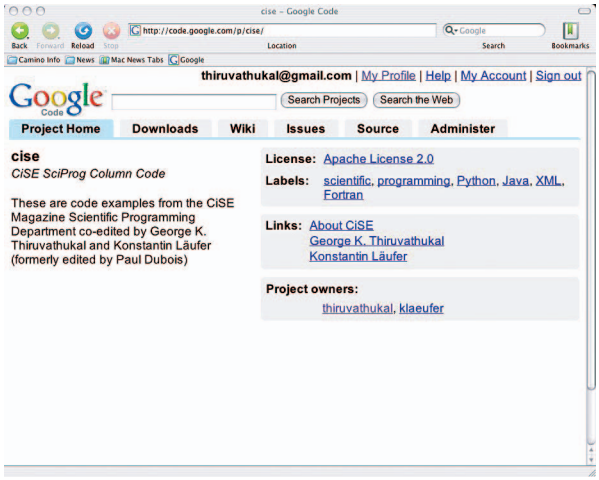


Figure 1. Screenshot of the Google Code interface. Here's the *CiSE* Scientific Programming department's new home for code samples.

File Uploads and Downloads

With a few exceptions, virtually all the famous programming projects for computational science embrace the FOSS concept for distribution. You can download practically anything, from Linpack to Numerical Python (Numeric), simply by Googling it.

A significant concern when making a file available, though, is the "fear of success": what if you're suddenly faced with a huge number of downloads? The ability to make files available for download by using someone else's pipes is a great way to address and allay this concern. Google Code makes it a breeze by providing a simple interface on its Downloads tab to upload a file for distribution. You can also include metadata to give prospective users a helpful description of the file. In fact, Google Code lets you specify lots of useful information:

- *summary*, a one-line description of the file that you're making available;
- *file*, the file content itself; and
- *labels*, which can come from a predefined collection, or you can create your own.

I've already made a download archive available for the example code we presented in our "Unit Testing Considered Useful" article from the November/December issue (vol. 8, no. 6, 2006, pp. 76–87). To make life easy for you, the code is distributed in a file called `dimensions-java-2006-12.zip`. I also assigned the labels Type-Source and OpSys-All to indicate that the download contains source code only (meaning you have to compile and run it yourself) and that it's designed to run on all platforms—assuming you know how to compile it on any given platform, of course. Figure 2 shows the file download screen.

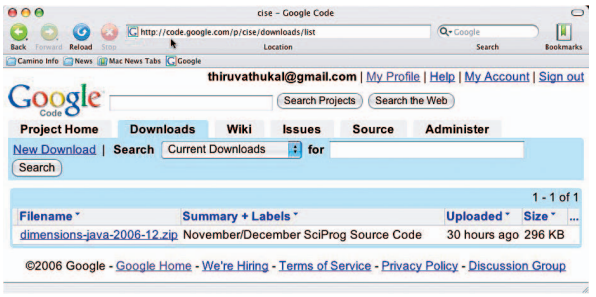


Figure 2. File uploads and downloads. This page is live, so you can go to it right now and start downloading code.

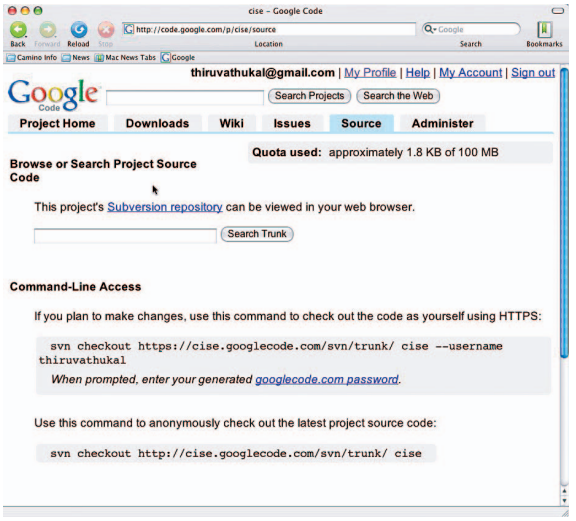


Figure 3. Source code browsing. The Subversion system has three core ideas: repository browsing, anonymous checkout, and developer checkout.

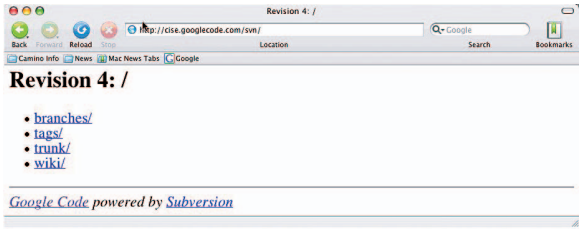


Figure 4. Repository browsing. We start with a top-level structure, which is Subversion's default layout.

Source Code Management

The major impetus for project hosting is that it allows one or more developers to manage a project's source code. Most hosting sites provide several capabilities in this regard.

Google Code uses the Subversion system; let's start with the Source tab and examine the features we can expect to find. Figure 3 highlights three core ideas:

- *Repository browsing* offers the ability to examine a project's repository entirely through a Web interface. It's a useful way to get acquainted with a project without having to download, unpack, and use an editor to view the code. Google Code, naturally, is integrated with search.
- *Anonymous checkout* gives people who aren't on the project the ability to check out all or a subset of the project's source code. This type of checkout is also called *read-only checkout*, in which you can change the local copy you checked out, but you can't commit your changes unless you're added as a developer.
- *Developer checkout* is reserved for members of the development team; they also have a read–write access option.

For clarity, let's look at each of these ideas in slightly greater detail. Figure 4 shows what happens when we click on the Subversion Repository link (you can follow along by going to our project; http://code.google.com/p/cise/, Source tab, Subversion Repository hyperlink). What we see here is a top-level repository structure, which is Subversion's default layout. It's beyond this article's scope to introduce all of Subversion's ideas, so we'll focus here instead on its *trunk*, which is the main development branch for all our example codes. Go ahead and click on it—you're now one level deeper into the repository. You can see many items here, but there's at least one subdirectory (folder) called `dimensions-java`. Click again, and you'll reach the actual source code featured in our "Unit Testing Considered Useful" article. All the `*.java` files, examples, and GUI subdirectories are the same ones we presented in that article (see Figure 5).

Repository browsing is a great way to get started, but most of you will probably want to download the code for offline study. To go this route, you need to install the Subversion client tools on your computer. Subversion works on all platforms—including Macintosh OS X, Linux, and Microsoft Windows—so read the "Subversion" sidebar for details on how to get it up and running on your favorite operating system. Once you install the command-line tools, you can perform the anonymous checkout as in Figure 6.

The command in Figure 6 presumes you want to check out all the `cise` code that's reachable from the trunk into a local subdirectory called `"cise"`. For the most part, this is fine because our repository only has a few dozen files. However, if you want to check out only the `dimensions-java` subdirectory, you should use the following command instead (the output isn't shown):

```
svn checkout
```

```
http://cise.googlecode.com/svn/trunk/
dimensions-java dimensions-java
```

Subversion is designed to be as general as possible, so you can check out virtually anything to anywhere. For our collective sanity, however, let's keep the repository structure simple. You can assume for the most part that all of this department's future articles will introduce a new subdirectory that's directly reachable from the trunk. I plan to cover Subversion in more detail in a future installment, so this isn't the last you'll hear of it.

Wiki

Many of the tools and features we'll discuss in this and the remaining sections are integrated into Google Code and other project-hosting solutions, which is a huge benefit for serious collaborative development.

In its basic form, a wiki page is nothing more than a page name with structured content. Figure 7 shows how to create a wiki page called WelcomeToSciProg.

Wiki pages are traditionally named via CamelCase (also known as WikiWords), wherein at least two words are concatenated to form a wiki name. Google Code requires the use of CamelCase, but some Wiki systems, such as the famous MediaWiki that powers Wikipedia, allow pages to be created with any name, including spaces. To refer to another wiki page within a page, you simply use the CamelCase name within the page text; it turns into a link automatically. Ultimately, the text in wiki pages is both easier to edit (see Figure 7) and read (see Figure 8) than it is in other markup approaches such as HTML.

It might not be readily apparent, but wikis are the preferred technology for organizing most FOSS projects' business and technical ideas. It isn't uncommon for developers to write pages describing a proposed project's core components while simultaneously writing the code. Wiki pages are also commonly used as a project's initial documentation—sometimes before the developer has even produced a user manual or other appropriate documentation (such as technical papers).

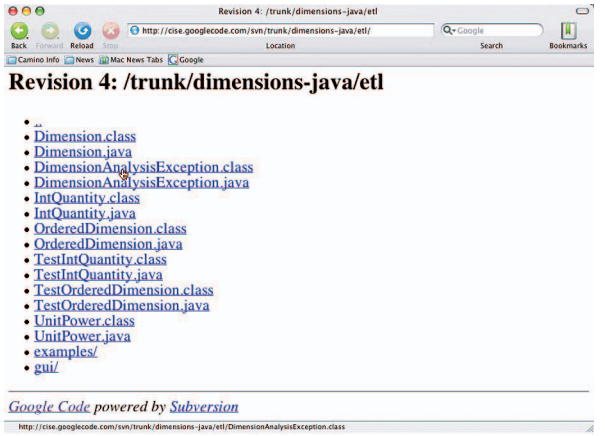


Figure 5. Source code. If you click on any of the `*.java` files, the actual source code from the November/December article "Unit Testing Considered Useful" will appear in the browser window.

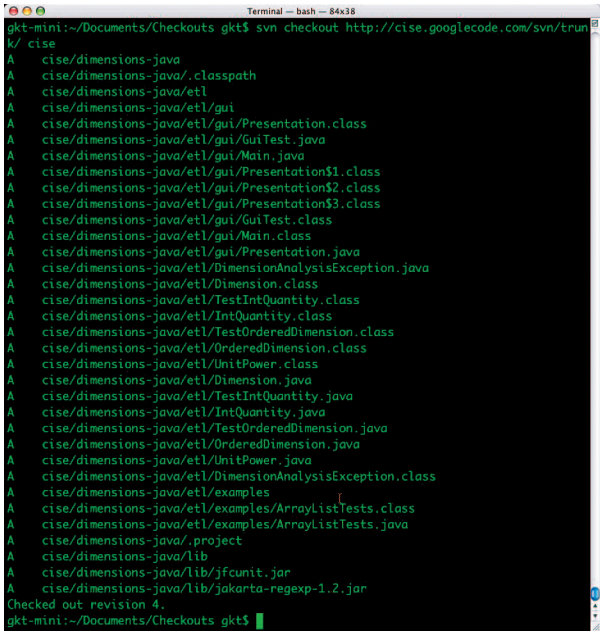


Figure 6. Anonymous checkout. Once you install the command-line tools, you can use the Subversion command-line client to download code for offline study.

Issue Tracking

As a software project evolves into a product that other people can use—especially a large community of other people—it becomes necessary to provide that community with a structured way of capturing issues (or bugs). Virtually all project-hosting sites provide this capability.

Figure 9 provides a glimpse of how Google Code handles issue tracking. Because our current project doesn't have many issues, I've captured the output from a significant project called the Google Web Toolkit.

As you can see, this project has several issues, some of which are actual bugs (defects) and others that are requests for new features (enhancement). The development team prioritizes these issues and assigns a disposition to them (such as accepted, rejected, and so on).

Creation of a new issue is straightforward, as Figure 10

SUBVERSION

The home of "all things Subversion" is http://subversion.tigris.org. Among all the things it offers is the official Subversion code itself.

svnX (www.lachoseinteractive.net/en/community/subversion/) is a Cocoa GUI aimed at providing a more natural integration of Subversion with Macintosh's OS X.

Tortoise SVN (http://tortoisesvn.tigris.org) is a Subversion client for Windows that works as a Windows "shell" extension. It literally lets you connect to a subversion repository and perform all the Subversion commands through the Windows Explorer interface by bringing up the context menu and doing what you expect it to. It's a great way to keep a permanent connection to our examples and simply update your checked-out copy as new examples appear—all from the comfort and safety of your own desktop. Subclipse (http://subclipse.tigris.org) is an Eclipse plug-in that lets you do everything in the Eclipse development environment.

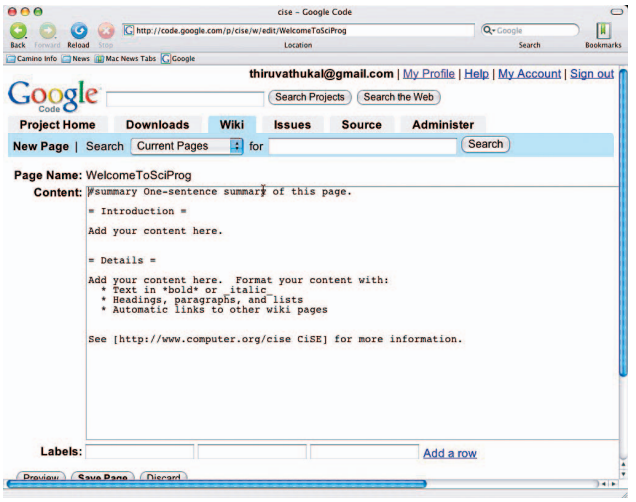


Figure 7. Wiki page creation. This is how we'd create a page called WelcomeToSciProg.

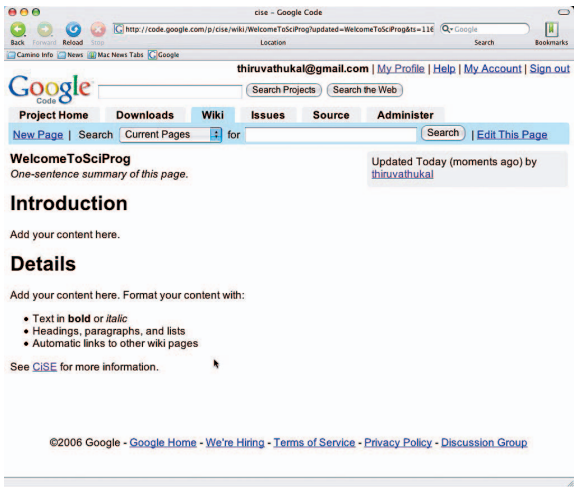


Figure 8. Wiki page rendering. From the creation stage in Figure 7, we get a final rendered page.

shows. Here, the user simply enters a concise summary and description of the issue, and, if it's a bug, how to reproduce the problem. The most important thing in reporting an issue is to give it some sort of classification so that the development team can best figure out what to do with it.

In my opinion, issue tracking is a critical component in any quest to organize a project—it's not just for bug reporting anymore. Issue tracking gives a project a way to balance what users want with what developers view as project priorities. Email alone can't address this need, primarily because it lacks structure.

Administration

No project-hosting solution is complete without the ability to actually manage the project. I'm not talking about the general notion of project management but that it must have facilities for administration (and customization).

Fortunately, Google Code provides many screens for administration:

- *Project Metadata* describes the project's public information, such as what you see when you visit http://code.google.com/p/cise/. This is by far the most important administrative screen when it comes to a project's public face.
- *Project Members* is used to add or remove members from a project and establish their roles within it. Most project-hosting sites have two roles, administrators and developers.
- *Downloads, Wikis, and Issue Tracking Labels* are the configuration options for the labels assigned when developers or users create something in a project. Although the label concept is somewhat endemic to Google applications, the capability is present in most other project-hosting sites.

Google Code also lets you create an incredible amount of project metadata:

- *summary*, a one-line explanation of your project;
- *description*, additional details for those who might not understand your exceedingly terse summary;

- *license*, the FOSS licensing scheme for distributing your code (you can choose from several, depending on your needs, but you shouldn't use Google Code or any FOSS hosting solution if you don't embrace at least one FOSS licensing scheme);
- *links*, the ability to connect to other related pages, such as the developer's home page, blog, groups, or other related work (there's no limit); and
- *notifications*, one of the most important options on the page because it handles communication between developers.

It's important for any development team with more than one person on it to use notification, especially for code changes. When anything happens to our repository, for example, I have email sent to an alias, `cise-svn@etl.luc.edu`, which results in a message to me and Konstantin at our regular email addresses.

Project Members is a fairly straightforward administrative screen. Basically, it has two boxes for you to indicate project administrators and developers. You must have a Google account ending in @gmail.com to own a Google Code project, but the same is true of all project-hosting sites (all developers must register for an account). An interesting question is how you become a project administrator—rather simply, the answer is to create a project. When you do, you become the administrator by default, but you can add others as administrators as well. Once you're anointed as a developer or administrator, you can do whatever you want with the actual project data, so these roles shouldn't be doled out lightly. It's beyond this article's scope to talk at length about FOSS culture, but many FOSS projects don't allow someone to become a developer without demonstrating initial competence in the software being developed.

The screens for Wiki, Issues, and Downloads are interesting but mostly trivial. With these screens, you can define or change sets of labels—for example, when entering an issue, you have only four types of "types" available (Type-Defect, Type-Enhancement, Type-Task, and Type-Other) and for "priorities" (Priority-Critical, Priority-High, Priority-Medium, Priority-Low). This granularity is fine for most projects, but in some environments, you might want more or fewer options. Google Code and most other project-hosting sites give you the ability to customize the labels used in any particular situation.

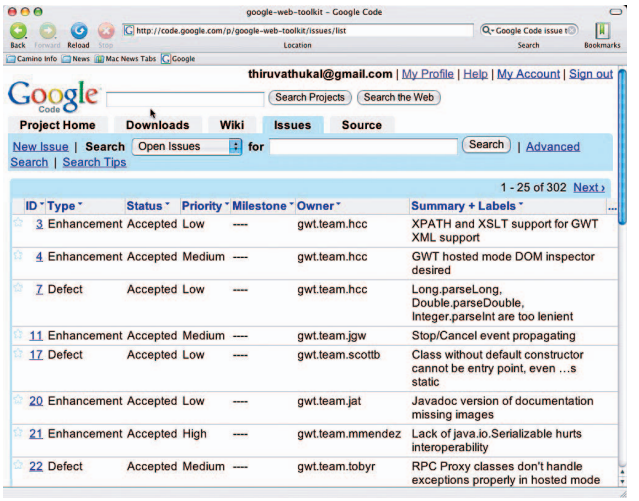


Figure 9. List of issues. Our current project doesn't have many issues, so this list comes from the Google Web Toolkit project.

Figure 10. Creating a new issue. The user enters a summary and description of the issue, and, if it's a bug, how to reproduce it.

Although a certain amount of collaboration can happen in a proprietary or closed setting, I'm convinced that the notion of sharing as defined in FOSS communities will help science—especially computational science—become even better than it is today. I hope this article helps you get started, if not for your own projects, then to encourage others to do so!

George K. Thiruvathukal is a professor of computer science at Loyola University Chicago. His research interests include programming languages, operating systems, distributed systems, architecture and design, computing history, and enhancing science and computing education with emerging technologies. Thiruvathukal has a PhD from the Illinois Institute of Technology. Contact him at gkt@cs.luc.edu or http://people.cs.luc.edu/gkt/.