GSoCStudentGuide

1 Introduction

1.1 What is Google Summer of Code?

You've heard other students talking about it, you've seen flyers and blog posts on it and now you want to know more! So here we go...

Google Summer of Code (GSoC) is a global program that matches students up with open source, free software and technology-related organizations to write code and get paid to do it! The organizations provide mentors who act as guides through the entire process, from learning about the community to contributing code. The idea is to get students involved in and familiar with the open source community and help them to put their summer break to good use.

Accepted students gain exposure to real-world software development, and employment opportunities in areas related to their academic pursuits. Participating organizations are able to identify and bring in new developers. Best of all, more source code is created and released for the use and benefit of all; all code produced as part of the program is released under an open source license. The fact that you get to write code that people from all over the world can use - how cool is that!

This program has brought together thousands of students and mentors from over 100 countries worldwide. As of January 2017, over 565 open source projects, from areas as diverse as operating systems and community services, have participated as mentoring organizations for the program. Successful students have widely reported that their participation in GSoC made them more attractive to potential employers and that the program has helped greatly when embarking on their technical careers.

Goals of the Program

The GSoC program has several goals:

- Get more open source code written and released for the benefit of all.
- Inspire young developers to begin participating in open source development.
- Help open source projects identify and bring in new developers.
- Provide students the opportunity to do work related to their academic pursuits during the summer: "flip bits, not burgers."

- Give students more exposure to real-world software development (for example, distributed development and version control, software licensing issues, and mailing list etiquette).

A Brief History of Google Summer of Code

Google Summer of Code began in 2005 as a complex experiment with a simple goal: helping students find work related to their academic pursuits during their school holidays. In GSoC's first year, 40 projects and 400 students participated. By the conclusion of the twelfth Google Summer of Code in 2016, over 12,000 students have been accepted into the program. Best of all, most of the organizations participating over the past twelve years reported that the program helped them find new community members and active committers.

So you can imagine how GSoC has grown over the last decade plus and in the process helped students in making most of their summer time by working on exciting open source projects and also helped the open source community by finding them potential contributors.

You can take a look at the appendix if you're interested in a more extensive history of the program.

1.2 Why Should I Apply?

So you've found out about this Google Summer of Code thing that requires you to submit a lengthy proposal and write a lot of code. Why bother about it at all? Well, here are some reasons why you should start writing your proposal today!

An absolutely amazing learning experience
GSoC is a place where you don't just get to apply your your skills but also get to acquire a bunch of new ones. And the learning is not just limited to technical knowledge. GSoC introduces you to a new paradigm about building code collaboratively. Not only that, GSoC is a platform which lets you build on your current skills and hone them. There is a project for all skill levels at GSoC!

> "GSoC is one of the best professional experience a student can have during his college years. I've participated twice already, but now I'm starting to regret I didn't do this every year! Awesome! Thank you Google! "
> Evelina Vrabie, Sunlight Foundation, GSoc Student 2010

> "This is a big opportunity for all the students like us to use the things that we are learning in colleges in a practical manner. Thanks Google and all the mentoring organization for allocating your valuable time."
> Kasun (Lakpriya) Hettige, Apache Software Foundation (ASF), GSoC Student 2010

Sense of achievement
There is something very fulfilling about getting your first patch committed upstream in some application that will be used by thousands of other folks all across the globe.

> "I just want to say...it feels *awesome* when your first patch goes in. Really awesome. "
> Mike Conley, Review Board, GSoC Student 2010

Hone your developer skills.

Students who are already active developers can apply themselves to a specific project. You'll get direct feedback from your mentor.

> "At this time one of the most memorable incident happened in my life, where my mentor proposed me as a committer, and my ambition to become an Apache committer finally became true. "
> S. Suhothayan, ASF, GSoC Student 2010

Getting involved and building your network

During the 12 weeks of writing good code, you interact and share ideas with some great people. At the end of it, you've made some great friends from all over the world with whom you can discuss fun projects, get feedback on your code, and just about anything.

> "Exciting and challenging. I met kind, knowledgeable and smart people all gathered in a single place, united, working for a single goal. "
> José Luis Vergara Toloza, KDE, GSoC Student 2010

> "Summer of Code is about much more than just code. The sheer fun of integrating with the open-source community and your mentoring organization can in fact outweigh the gratification of actual coding. "
> Kamran Khan, Ubuntu, GSoC Student 2010

Your spring-board to the open source world

GSoC is a great place to get you jump started and up to speed on the basics of the open source community. This paves your way forward for making big contributions. It's a great way to make your first entrance on the open source stage and get noticed.

> "GSoC2010 was my first real foray into the Open Source World -- I had been teetering at just the edge of actual contribution for a few years, having tried out WordPress, etc. but had never actually ever submitted a patch, or released a project. "
> Kunal Bhalla, WordPress, GSoC Student 2010

For the love of code

If you enjoy coding and feel passionate about writing good quality code and building great software, GSoC is THE place for you give way to your passion. After all, what better way is there to celebrate your love for the code by participating in a summer of code.

The stipend, the fame and the t-shirt

One very tangible and obvious advantage of participating in GSoC is the stipend that you make while working with amazing people on a great project. Being a successful GSoC student is in fact a prestigious achievement. It's another way to make your resume more impressive. And of course you get a really cool t-shirt too!

1.3 Am I good enough?

Do you have *some* programming experience at the university level? Then, yes, you are good enough! No, you don't need to be a Computer Science or IT major. Students from all subject areas are successful GSoC students. Have work experience programming but spend your time studying philosophy full-time? Yes, you are good enough to be a GSoC student!

> "Although at first I was a little inexperienced in the world of mobile applications and a bit overwhelmed by the challenge, I can positively say that four months later I've gradually overcome all the obstacles. I had a great mentor, that's for sure!"
> Evelina Vrabie, Romania, Android Project - GSoC 2010

Every project has a different criteria for selecting students and subsequently different skill level requirements. If you meet the below list of general skills you are likely to find a GSoC project to which you can feel comfortable applying:

The soft skills

You find out where to go for help with technical questions
There are many available resources on the interwebs to go to for help with technical questions, knowing how to use a search engine to begin your search is very important.

You take and respond well to feedback
The community developed software model relies heavily on constructive feedback and the willingness for each contributor to take that criticism and make the code better. You are going to be getting regular feedback from your mentor -- not all of it is going to be "this is great" "you are awesome." Learning from and graciously accepting feedback is a very important trait for a successful GSoC student.

You can work independently
Since you'll be spending significant amounts of time working alone - not being afraid to face the unknown and start breaking down what may initially seem like insurmountable problems independently is important.

You know when to ask questions
Do think you all ready know everything about everything in the world of programming for open source programs? Then you probably aren't good enough for GSoC!

The technical stuff

You can install and configure software packages on your own
If you don't know how to download and install packages on your own, you'll need to figure that out, stat.

You have access to a functioning computer 40 hours a week

If your computer regularly dies, or you don't have dedicated access to a computer you'll need to figure that out before you start your project. Also, GSoC should be considered a full-time commitment. Don't expect an hour or two a day in an internet cafe to be enough!

You've got experience using the programming language and operating system of the project
Depending on the project, the skills necessary will range from beginner to expert, but you do need some experience. One of the great things about GSoC is that there is a large variety of organizations and projects to chose from. Chances are very good that you can find a project that meets you at your level. Even if you are a beginner! If the project primarily uses Linux for development and distribution, then you need to be comfortable with basic Linux usage. Sometimes, you'll see projects looking to expand onto other platforms, in which case you may be able to bring in new expertise.

Every project has additional characteristics that they look for when selecting students and projects - however, if you meet the above basic criteria - chances are good that there are GSoC projects and organizations to which you can feel comfortable applying.

Pro Tip: Don't be afraid to apply to projects where you only meet 51% of the listed requirements. Include a section on how you'll compensate for or learn the missing skills - and demonstrate during the application process that you are working on acquiring those new skills.

2 Applying

2.1 Making First Contact

When you walk into a party what do you do?

Interacting with an open source group is sort of like walking in on a party where it seems like everyone else knows each other. People are discussing topics you may be interested in, or sometimes they could be discussing topics you neither know nor care about.

If you're the type of person that would walk right up and introduce yourself at a party, then the best approach to getting started is to do what you'd do in real life. Contact the project, introduce yourself and ask questions related to your project.

However, if you're more likely to hang back and watch people for a while, spend some time observing community interactions before you jump in. You can also try to contact an organization admin for guidance, or with help introducing you to a community. If in doubt, have a look at the organization's ideas page for hints on where to go for help.

How to observe community interactions

- Join both the development and user mailing lists and spend a few days just reading the conversations.
- Read the mailing list archives.

- Join the projects IRC channel and lurk for a bit.
- Read all the available information on past GSoC projects.
- Take a stab at going through the project documentation, at least to the point where you feel like you can ask questions that are not already extensively covered in the docs.

At the end of your "listening and research" phase you'll understand how, where and when the community interacts and know the best way to ask questions.

How to begin participating in conversations

- Introduce yourself! If you are new to the community you need to let people know who you are and why you are interested in contributing to the project.
- Ask questions. You should be able to come up with a least a few legitimate questions before offering your opinion on the right way to do things.
- Be humble. Until you've engaged with community, for at least a few months, assume that those people talking about things probably are privy to community nuances you might not yet see.
- Don't be intimidated. Don't let a bad experience stop you from getting involved. Just relax and think about why you were snubbed and if there's anything that you should be careful about before participating in another conversation.

No matter what, don't wait until you apply to initiate contact! Engage with multiple communities once the participating organizations are chosen to get a feel for how different groups work.

2.2 Choosing an Organization

You now have a pretty good sense of what GSoC is all about. You've got the time available to commit to a summer project (even if it's not summer where you live). And you're pretty sure you've got what it takes to contribute to an open source organization. Now, where do you start? How do you choose a project/organization? The path to finding the right project or organization starts with knowing thyself...

> "You can do much more than what your university rank says. [The] World is full of opportunities. Google Summer of Code is one such a opportunity to find an answer for, "Who am I?" "
> Kapila Bogahapitiy, Network Time Protocol Project, GSoc Student 2009, 2010

Who Am I?

The very first requirement for a successful GSoC experience is finding a project/organization that interests you. Take a few minutes to consider the following questions. You can use the answers to these questions to search and filter through the organizations participating in GSoC.

- What open source software do you use?
- What are your professional interests?

- What are your hobbies?
- What is your skill set?
- What do you want to get out of GSoC?

Who Are They?

After the GSoC program is announced each year, a list of accepted organizations is published on the GSoC web site. Compile a list of organizations based on your answers above. The projects are also tagged with categories of programming language, platforms, topics and applications. Use the tags to filter organizations based on your skills and interests.

For each organization, take some time to learn more about what they do (i.e., Google them!). The organization's mission, it's size and range of applications may all influence your interest in working with them. Realize that through GSoC you will be joining an open source community. Ideally, you'll find an organization that you are enthusiastic to be a part of.

2.3 Finding the Right Project

Each organization will have a project Ideas Page linked to from the official Google list of accepted organizations. Browse the list of project ideas for each of the organizations for which you are interested. The ideas should give you a clear sense of the range and depth of projects being targeted and the expectations in terms of prior experience and programming skills. In addition to the list of project ideas, many organizations encourage original ideas proposed by students.

Shortlist

There's a big sea of projects out there for GSoC. Start by compiling a list of potential project ideas that catch your interest. For each idea, take some time to carefully consider what is being proposed, how its scope might be better defined, and how it might fit in with the larger picture.

Ask questions

Consider any questions you might have about the project, how it might be implemented, and what it would entail. Read the FAQs. If there's still anything unclear about the project and requirements, get your doubts cleared. Formulate your questions and suggestions regarding each idea into a clear and concise communication.

Evaluate your options

Once you've researched the shortlisted projects and got your questions answered, re-evaluate your options before writing the proposal. Also think about whether the communities you're interested in match your needs, and if you'll have fun working with those people. Is there enough documentation and help available to get you started on the project? Given all the feedback, do you think you're really excited about the project and think that you can do a good job of it?

From Project Idea to Project Plan

You've narrowed down your search of organizations and projects, you've made first contact, and you've started communicating directly with potential mentors. Now it's time for the critical process of turning a project idea into a project plan.

In most cases, your potential mentor(s) will have lots of ideas and preconceptions about each project that were not included in its original description on the Project Ideas page. Discuss and research the project idea in as much detail as possible. You may even consider preparing mock-ups (illustrations, powerpoint, or web sites) to help clarify your understanding and vision of the project. You will want to discuss the scope of the project idea, including which parts are critical versus optional for the summer timeline. This process will directly feed into your application and ideally distinguish it from all the others. Just think about it: if you've helped clarify the project idea and contributed to an actual plan of action, it makes it an easy process for mentors to evaluate your proposal and give it a high ranking!

Pro Tip: The earlier you apply, the better. Submitting your proposal early helps you get early feedback.

Don't be that person: Cut and pasting an idea from the organization page and turning that in as your project's description is a big no-no. You'll be expected to research and submit your own ideas about how to accomplish the project your way, not just state the end result.

2.4 Writing a proposal

This is a competitive program, each year Google turns down many more students than it funds. While pre-proposal activities are key to improving your chances of success, a poorly-written proposal is an easy way to fail. There is much you can do to ensure that your project proposal catches the attention of organization reviewers *in a positive way.*

The Basics

First and foremost, make sure you meet Google’s formal requirements for participation in Summer of Code. Hopefully, you have already checked this by now, but be sure to double-check before you waste time and energy on a proposal.

Inventory your time. Figure out how many hours per week are already spoken for outside of your

GSoC commitment, including time spent volunteering for other projects and activities, and counting credit-hours of University instruction. GSoC should be treated as a full-time job. If you have more than a few hours a week of extra commitments, you probably should skip GSoC; it is unlikely that you will be successful. In any case, be completely clear about outside time commitments as part of your proposal. Do not surprise an organization with your time commitments later on.

Make sure that you are able to be in regular close contact with organization mentors via the usual open source means (email, chat, etc) for the duration of the Summer. It is not necessary that you be geographically near your mentor. However, if you are not sure you will have good Internet connectivity continuously over the summer, GSoC is not for you.

This program is the Google Summer of *Code*. If you are less than fluent in the programming languages that your target organization uses, you might want to skip the work of writing an application. If you do decide to proceed, be clear about your level of ability, so that the organization can make an informed decision.

Elements of a Quality Proposal

Most organizations have their own proposal guidelines or templates. You should be extraordinarily careful to conform to these. Most organizations have many, many proposals to review. Failure to follow simple instructions is highly likely to land you at the bottom of the heap.

There are certain elements of the proposal that should apply to every organization. Proper attention to these elements will greatly improve your chances of a successful proposal.

Name and Contact Information
Putting your full name on the proposal is not enough. Provide full contact information, including email addresses, websites, IRC nick, postal address and telephone number. Also provide full contact information for a friend or relative that the organization can contact to find you in case of emergency.

Title
Your title should be short, clear and interesting. The job of the title is to convince the reviewer to read your synopsis.

Synopsis
If the format allows, start your proposal with a short summary, designed to convince the reviewer to read the rest of the proposal.

Benefits to Community
Don't forget to make your case for a benefit to the organization, not just to yourself. Why would Google and your organization be proud to sponsor this work? How would open source or society as a whole benefit? What cool things would be demonstrated?

Deliverables
Include a brief, clear work breakdown structure with milestones and deadlines. Make sure to label deliverables as optional or required. You may want your plan to start by producing some kind of white

paper, or planning the project in traditional Software Engineering style. It’s OK to include thinking time (“investigation”) in your work schedule. Deliverables should include investigation, coding and documentation.

Related Work
You should understand and communicate other people’s work that may be related to your own. Do your research, and make sure you understand how the project your are proposing fits into the target organization. Be sure to explain how the proposed work is different from similar related work.

Biographical Information
Keep your personal info brief. Be sure to communicate personal experiences and skills that might be relevant to the project. Summarize your education, work, and open source experience. List your skills and give evidence of your qualifications. Convince your organization that you can do the work. Any published work, successful open source projects and the like should definitely be mentioned.

Follow the Rules
Most organizations accept only plain text applications. Most organizations have a required application format. Many organizations have application length limits. In general, organizations *will* throw out your proposal if you fail to conform to these guidelines. If you feel you *must* have graphical or interactive content associated with your application, put just this content (not a copy of your proposal) on the web and provide an easy-to-type URL. Do not expect reviewers to follow this link.

Submit early

When you submit your proposal, the organization mentors can review it, and then ask you questions or request more detail on aspects of your proposal. You can continue to modify and improve your proposal right up until the submission deadline. If you submit a good proposal early, you are likely to stand out from the other applicants and build a relationship with the organization.

Outside the Project List

Some organizations allow students to propose work that is not on their official Ideas Page. This can be a great opportunity to get your proposal on the top of the stack. Reviewers tend to get excited about a student that goes beyond a direct response and enthusiastically proposes work that is novel and creative.

However, original proposals are also riskier; their flaws will be much more apparent. Here’s some of the ways that such proposals fail:

Projects without a mentor
Try to make sure that someone in the organization would be competent to work with you.

Projects that better belong with other Summer of Code organizations
Open source organizations try hard to avoid stepping on each other's turf. Try to find your best customer.

Projects that represent too large a scope
The time flies by quickly. If you have a large project, break it into small, coherent pieces and propose to get the first couple of them done. That way the organization can be confident that they will get at least one good piece of work out of you.

Incoherent proposals
The organization needs to see a clearly delimited, contained piece of work. If the organization can’t understand or define the work, the proposal will be thrown out.

Projects that are “inappropriate” for legal or social reasons
If your proposal is near the boundary, make sure you clear it with your target organization in advance.

Boring projects
For the mentor and the organization, half the fun is helping a student do something novel and cool. Infrastructure *per se* isn't necessarily boring, but it should be part of a luminous vision.

Stuff that’s already been done to death
Novel work should be novel. Surprise.

Even given this list, there’s plenty of room for cool work. Given the opportunity, you should seriously consider taking advantage and writing a proposal that differentiates you.

General Notes

While there is an official limit of five submitted proposals, you are encouraged to submit more than one high-quality proposal. If you have several strong possibilities for the same organization, consider submitting several proposals. Organizations will figure out which one they like best. But avoid sending many medium-quality proposals and concentrate on fewer high-quality proposals.

Most organizations are risk averse. It is better for everyone if your project is under-scoped and sure to complete; as opposed to a large-ish project which may not get done. Under-scoping is an annoyance. Incomplete is a disaster.

Integrate and leverage existing open source code in your project. Only propose to write something yourself if you cannot get it any other way.

The “Pencils Down” deadline for your project to be complete is usually sometime in mid-August. This will come sooner than you think.

2.5 Being Turned Down

You've done your homework, found an exciting project, and you've written the best proposal you could. And you didn't get into GSoC.

Don't despair! The beauty of engaging in the GSoC process is that you're learning about groups of people that extend beyond just GSoC. Making contact with potential mentors and a software

community sets the stage for future opportunities for participating in community-developed open source software projects.

What to do now?

First, d**on't take it personally.** Just like when you apply for a job, there are reasons why you might not get in, some that have nothing to do with you. Mentors may not be available, the organization may not have enough space for your project or it may just not be the right time for your proposal.

So, where to go from here? There are several strategies you can consider to go forward positively.

Ask for feedback on your proposal
Just like in job interviews, gathering information about why your proposal wasn't accepted is a great thing to do to improve your next application. Some example questions to politely ask if your proposal is turned down include:

- Was there a mismatch with my skills and the project requirements? If so, what skill areas can I work on to be better qualified next year?
- Did I engage enough with the community during the application process?
- Do you have any suggestions on how to improve my pre-application communication?
- Was my project plan clear? Do you have any suggestions on how I can better communicate my ideas via the project plan next year?

Following up and getting more information about what you might be able to do differently next time is a great pathway to success.

Approach an organization about doing the project anyway
For those students with the drive to forge ahead without GSoC financial support, you may find that a community really is interested in your project anyway. Don't be afraid to approach your community, GSoC org admin or mentors you communicated with about future contribution.

Perhaps you can work on a smaller portion of your idea over a longer period of time on your own, or find another project better suited.

Some successful students have even found business sponsorship of their work later on.

> "This story is about how after about one year and half I became a Linux Kernel Maintainer [of] the Bluetooth Subsystem. On the 2009's GSoC, I worked in important new feature in the kernel side of the Bluetooth stack. The work was too big and I wasn't able to finish it, then after some months a big company hired the company I worked with to have me finish that work. [Now] all work is merged in the Linux tree."
> Gustavo Padovan, BlueZ, GSoC Studnet 2009, 2010

Stay connected
If you've already invested time and energy getting to know a community, stay involved! Subscribe to relevant mailing lists, participate in IRC or fix small bugs that you have time for.

There are also many non-code ways you can contribute to software. If you're interested in documentation, graphic design, release testing, public relations or marketing, most projects welcome contributions in these areas. Taking on small non-code projects can be a great way to stay connected and build a reputation in a community.

Try a new organization
Maybe that project wasn't the right fit! Part of selecting a project is selecting a community that works with you. (Also see the chapter on "What does community mean")

> "I had a look and approached many organizations, and finally decided to hook up with Sakai Foundation. The mentors and the people I interacted with from the community really encouraged me and I felt very comfortable with their coding practices [and] the programming languages."
> Ashish Mittal, Sakai Foundation, GSoC Student 2010

Keep trying
Just keep trying. The next proposal just might be accepted...

> "*Never* give up. It took me 3 years and 12 proposals to finally get into the program. If none of your proposals gets accepted, sit back and relax. You have a whole year ahead to improve your role with the open-source community by writing more code."
> Kamran Khan, Ubuntu, GSoC Student 2010

3 Community Bonding Period

3.1 How GSoC Works

Google Summer of Code moves in phases after you are accepted. The first phase is the Community Bonding Period in which you get to know your community and get familiar with their code base and work style. The next phase is the initial phase of coding (Phase 1) which is evaluated with Phase 1 evaluation a month into the Google Summer of Code term. The second phase (Phase 2) is evaluated two months into the program coding period. The final phase is your time to complete your project. There will be a final evaluation at the end of the term; you will also need to submit a sample of the code you produced.

Evaluations
Evaluations are not as daunting as they may sound. This is an opportunity for you to evaluate your mentor and your mentor to evaluate you, not a quiz on your coding abilities. Some mentors even choose to review their evaluations afterward with the student in order to integrate the feedback into the coding process for the rest of the term. Be honest about your experiences with the program. This helps GSoC improve in future years. Remember, you must complete your evaluation of your mentor at each phase or you will receive an automatic fail.

Payments

Payment is in three pieces. The first payment is sent shortly after you pass the Phase 1 evaluation. The second payment is sent after you have passed the Phase 2 evaluation; the final payment is sent once you pass your final evaluation. All of these payments take a few business days to become active on your card or in your bank account if you choose direct deposit.

T-shirt and Certificate
You get a t-shirt, notebook, pen, sticker and a digital certificate of completion at the end of the program if you successfully pass GSoC, unless you are in a Restricted Country. If you are in a Restricted Country you will receive a digital certificate of completion. The program administrators again have to ship each individual package to your home, so please be patient. Mail systems are particularly hard to navigate in some parts of the world and can take almost a month to reach some locations.

The rest of your experience with the program will be determined by your interactions with the community within your mentoring organization. Most students consider the interactions with their mentor and the rest of the open source community they're involved with to be the most important part of the Google Summer of Code experience.

Participant Roles

There are four roles in the Google Summer of Code program:

Student
This is you! A student participant in GSoC is typically a college or university student; the only academic requirement is that the accepted applicants be enrolled in an accredited academic institution. Students must be at least 18 years of age to participate. Students come from a variety of academic backgrounds, and though most students are enrolled in a Computer Science program there is no requirement that they be studying CS. Past students have come from disciplines as varied as Ecology, Medicine, and Music.

Organization Administrator
Org admins are the "cat herders" for GSoC open source projects. Some org admins also mentor students during GSoC. Org admins are the final authority about matters such as which student projects will be accepted and who will mentor whom. If you're having difficulties communicating with your mentor or making progress, an org admin can help.

Mentor
Mentors are people from the community who volunteer to work with a student. Mentors provide guidance such as pointers to useful documentation, code reviews, setting milestones for the student, etc. In addition to providing students with feedback and pointers, a mentor acts as an ambassador to help student contributors integrate into their project's community.

Program Administrator
Program administrators are employees of Google's Open Source Programs Office who run the program. These folks do a variety of tasks: select the participating open source projects each year, create and

analyze the program evaluations, administer the program mailing lists, ensure that participants are paid, and send out the all-important program t-shirt. Program administrators do not select which student proposals are accepted into Google Summer of Code.

Afterward

It is a primary goal of Google Summer of Code that the student participants stick around long after the program has ended and continue contributing to their project communities. Great mentors continue working with their students and encourage them to do so. In the end, mentors and students take a well-deserved break before the GSoC cycle starts again.

3.2 What does community mean?

The word "community" gets thrown around a lot in open source. So what does it mean?

People involved with open source community have differing political, social, economic and ethnic backgrounds. They range from elementary school students to the elderly. They work across timezones, languages, and cultures.

Some developers may work on only a small part of the project codebase, while others know every line from memory. Communities also include people who don't work on the source code directly, such as graphic designers or testers.

When talking about community, it helps to focus in on political and social structure. The project participants you talk to may or may not formally represent the project. Assume, unless otherwise stated, that people represent and speak only for themselves when you interact with them. However, identifying the leaders and the project structure can help you to better interact with all members of the project team.

Community leadership structure

Much of the authority in a project rests with those people who control the community code. At least, it starts there. Actual organization varies widely, but there are some common structures that may help you find your bearings.

Benevolent dictator
Many projects start with a single developer. As time goes on, this person, often called a "Benevolent Dictator" maintains control of most or all commits to the core code, taking patches or branch pushes as they see fit. As more contributors are attracted to the project, the Benevolent Dictator may not do so much dictating, but is often guide for new folks. When there's conflict, the Benevolent Dictator usually makes the final call.

Many committers, rough consensus
Some projects allow almost anyone to change their codebase. These groups are likely to give commit

access to GSoC students as part of a project. When deciding which features to work on or what code is committed to the master code repo, decisions may be made by the group or left to a few trusted individuals. When there's conflict, majority approval may be requested before further action is taken.

Decisions are often made based on who has working code, rather than on a theoretical basis. Groups may not formally document how they make decisions. In this case, a little time spent with the group can help you learn the culture and how to work together. If you end up working with a group like this, your mentor will help you acclimate!

Formal organization
As groups grow in size, some kind of organization becomes necessary. The type of formal organization varies widely across social, cultural and political boundaries. Some ways of organizing include:

- Electing a steering committee by a vote of developers.
- Forming a non-profit organization or foundation.
- Creating a for-profit business.
- Joining an existing non-profit organization.
- Formalizing the existing developer relationships by naming a group of committers the "core" decision makers.

These groups can have documented processes for assuming leadership roles, and may put more effort into non-code activity. Generally speaking, formal organizations have governing documents, and a clearly documented leadership group that changes from time to time.

Community is people

Apart from project leaders, open source projects tend to be full of independent, capable people who like to learn. Long-term collaboration tends to turn strangers into friends. Part of what people enjoy about open source development is that they have the opportunity to choose their colleagues. Developers often participate for fun, relaxation and friendship.

When you're talking to a software community, realize that you're really just talking to people, albeit some of the most interesting people who create software. Enjoy the experience and be a good neighbor, and you'll almost always fit right in.

3.3 Open Source Culture

When you encounter an open source group for the first time, it may be a bewildering experience. Whether posting to a mailing list for the first time, blogging about the project you're taking on or hanging out on an IRC channel - the way people interact, and what they expect from each other is pretty different than in classroom or with friends and family.

Openness and Sharing

Open source communication can vary a lot. A core value held in common is that sharing code is good. Regardless of license, language or indentation style, open source developers create, share and modify source code together.

Working on code together means a lot of things: transparency, directness and cooperation are words that are often mentioned by developers when describing the process. It can involve bug reports, code refactoring, implementing new features, documentation, project management and advocacy.

Amazingly, the ways in which people actually share code are as varied as the individuals involved. Even if you have previous experience with other open source projects, keep in mind that you still need to take the time to learn how the new open source project works, and acquaint yourself with their particular brand of sharing.

One aspect of "open culture" is that people are informal. People address each other by first name. They tend to speak directly to one another, regardless of social status or formal title. Disagreements about code, whether as profound as which algorithm is most appropriate, or as seemingly mundane as how many spaces are used for indentation, are aired in public. This process is very intimidating to newcomers, who might be concerned about having their words immortalized on the Internet, and worse, saying something wrong or embarrassing. The only way to get over this fear is to practice and participate publicly.

Although "open culture" is generally informal, it is important to remember that you still need to mind your manners when participating in conversations.

Remote Communication

Many projects involve individuals who are working not only in different cities, countries and continents, but collaborating across major cultural and language differences. And rather than having procedures or policies on how to interact with one another handed down from HR departments or other authorities, communication practices evolve between individuals over time.

Because there are few rules around working together on open source projects, there is a lot of freedom to share directly with one another. This freedom is at the core of what attracts many people to open source software. Multi-cultural projects offer an incredible opportunity to share knowledge between individuals directly. With sharing, however, comes a responsibility to inform new participants about expectations for communication and ways to solve misunderstandings to the benefit of all.

Be mindful of cultural assumptions about race, gender, sexual orientation and disability. People often make assumptions, have stereotypes or are biased in ways that no one can control. The best practice is to assume that people don't mean any harm, and when they're told respectfully that they've offended or hurt someone, that they'll stop whatever it is that they are doing that is harmful.

All that tolerant rhetoric aside, it is never productive for an open source project to allow individuals

who consistently try to cause harm to others to do so. If you are causing undue problems don't be surprised if you are asked to discontinue your participation regardless of your contributions. It's often better for an open source project to ask someone to leave, than to allow them to harm others and in turn, cause other productive members of your team to depart.

Abbreviations and Slang

People come up with abbreviations and slang that are meaningful inside the group, but not necessarily to outsiders. Ask questions when you don't understand a term, a joke or some arcane bit of project lore.

Here are a few useful resources for teasing out meaning from initialisms, acronyms and abbreviations:

- Urban Dictionary (http://www.urbandictionary.com)
- Webster's Online Dictionary (http://www.websters-online-dictionary.org/)
- Acronym Finder (http://www.acronymfinder.com/)
- A to Z word finder (http://www.a2zwordfinder.com/)

Volunteerism and Gift Economies

Donated time are the life blood of open source projects. Many individuals contribute their time and energy without any expectation of compensation or even a simple "thank you" in return.

Contributions to projects are often self-directed, with developers having a personal itch to scratch in the form of a new feature, by correcting a bug that they've encountered or by implementing something from a TODO list. Projects operating in this way may seem chaotic if you are only familiar with top-down management - where teachers, professors or bosses tell you what to do and when. Of course, some projects do have clearly defined project management, with milestones and tasks meted out carefully.

Because many (or in some cases all) contributors are volunteering, methods of coercion available to businesses are not available. The best way to collaborate is to behave in a way that encourages others, and ultimately, makes people want to contribute. Some easy ways to encourage volunteerism include:

- Saying thank you
- Giving compliments when they are deserved, regularly and in public
- Publicizing cool hacks and features as they are implemented, in blog posts and on the mailing lists
- Promptly committing useful code
- Responding promptly to requests for information
- Clearly defining ways to contribute to your project (TODO lists are great!)

Consider treating every patch like it is a gift. Being grateful is good for both the giver and the receiver, and invigorates the cycle of virtuous giving.

Overall, your goal is to help create and maintain an atmosphere around contribution that is enjoyable.

What that means will vary significantly depending on the project, but certainly the above points apply to any project.

3.4 Communication Best Practices

One of the best things about GSoC is the group of individuals from diverse cultures and different open source organizations that participate each year. This also means that you cannot make any assumptions about communication. Specifically:

- Always read the Program Rules, FAQ and manual first! One of the best ways to destroy your reputation early is to repeatedly ask questions on IRC or the mailing list that are covered in a FAQ or other types of documentation.
- Don't expect instant answers. Remember you are working with volunteers.
- Avoid humor when communicating with the entire GSoC community. It does not translate well to large groups and is likely to be misconstrued by someone.
- Remember that other peoples' time is just as valuable as your own.

How to Use a Mailing List

Every community uses mailing lists differently. It is extremely important that you spend some time understanding how to participate in a way that is acceptable to the community. Read the mailing list archives, observe for awhile first and be sure to inquire about and read any codes of conduct or guidelines before you post.

Some specific guidelines that should apply across communities:

- Follow the mailing list posting style. Do people top or bottom post? Follow the norm.
- Keep your posts on-topic. Avoid tangents and posting "I know this is off-topic but..." type posts. If you have any doubt, email the moderator *before* you post.
- Don't cross-post to multiple lists. Communicating well on mailing lists means knowing where and when to post items, if you have any doubts regarding what is appropriate, ask the moderator.
- Never send community members unsolicited personal messages. Especially, if your content is along the lines of "the way you talk about code really turns me on." Inappropriate!
- If you are offended by a response or post, walk-away. Do not respond when you are angry.
- Avoid posting to the mailing list if you are significantly under the influence of anything that

makes you behave out of your norm. Just like drunk-dialing is bad -- so is drunk emailing to your open source community mailing list.
- Avoid profanity.
- Don't post chain letters, marketing messages or other types of non-topical spam.
- Always check the archives before re-posting your message. Lots of mail clients don't send you duplicates of your mailing lists posts. Check the archives before you re-send!
- Always read the entire thread before replying. Seriously, read every message in a thread first!
- Don't use the mailing list as your own personal Google. Take the time to research the question first. Check the archives, search the project documentation first, Google it yourself.
- Don't begin or get involved in religious or political arguments.
- Never proselytize on the mailing list.
- Do your best to always assume the best of the poster. We all have bad days. Sometimes, a non-native speaker may appear to have an offensive "tone" in the post where no offense is meant.
- Avoid unintentional "tone" in your postings. If you have any doubt read your message out loud or have someone else read your message before you post.

Like a bullhorn

When you post to the GSoC student list you are essentially using a bullhorn to broadcast your words to 7,000 people across time zones and international boundaries. Use your bullhorn wisely, or it might be ripped from your hands by an unruly and angry mob, or by a responsible moderator.

How to Use IRC

Many projects use IRC for real-time conversations. Often these IRC conversations appear very casual. It is always best to assume formality if you have any doubts about who may be listening or participating in the conversation.

The Fedora Project has a great FAQ on how to use IRC. Read it!

http://fedoraproject.org/wiki/Fedora_Channel_FAQ

Some additional guidelines:

- Include your real name in your IRC signature information.
- Never private-message (pm) someone that you don't know without first asking if it is OK in the public channel.
- Read the topic before asking questions.
- Stay on-topic for the channel.
- Before asking questions in #gsoc - read the FAQ!

3.5 How to Get a Head Start

The last thing you want is for the coding period to start and then realize that you don't have all the tools you need installed and configured to actually begin work. Don't let this happen to you! The community bonding period is the perfect time to get these things sorted out.

Get Your Development Environment in Order

Each project has a unique set of tools and packages required to work with developers. These often include:

- Compiler
- Language interpreter
- Text editor
- Version control system
- Modules and libraries needed by the software
- Database, mail or web server
- Real-time communication tool (e.g. irc client, instant messaging)

Some organizations require testing on multiple operating systems and/or platforms. Make sure you know what is expected of you as early as possible. Read the available development documentation and contact your mentor to figure out exactly what tools you need to succeed. Also learn the the bug-reporting process that your organization uses and also understand the the project's release management strategy.

Practice

Once you get your development environment setup start practicing! This includes getting familiar with the coding standards, codebase, and testing and documentation policies of the open source project community. Do a few practice commits and work on understanding how source control works within your project. Brush up on any new skills and start asking questions.

Do Some Background Research

Look through the projects bug database and read through the user list to understand your end users. Peruse the mailing list archives and go through the project's existing documentation.

Start Interacting

Take advantage of the community bonding period to connect with your mentor, and other students in the program. Set up a blog, get involved on relevant forums and mailing lists and in general, start interacting with the development community. Make sure you have what you need to succeed, and if you don't, ask your mentor for help.

Start Working with Your Fellow Students

GSoC is not only about working with your mentor. There's this amazing group of outstanding and motivated students too.

Student mailing list
Google has a private mailing list for the GSoC students. Go ahead and introduce yourself on the student mailing list. Talk about your project ideas, get feedback on your proposal. The mailing list also has GSoC participants from earlier lists. Use the student mailing list as an additional resource. Your questions can be technical or non-technical. Of course, remember not to be a bullhorn and be mindful of the mailing list etiquette.

They're just like me!
There are many students who have faced or are facing the problems like you. Don't be scared to ask your questions. This is more true for the students who have been accepted to the same organizations as yours. You can ask them about how they got their dev setup working. Help each other out on the irc and mailing lists. Don't be afraid to ask and don't be afraid to answer!

Make friends around the world
GSoC is a great opportunity that helps people and communities collaborate across boundaries. Use this opportunity to learn more about diverse technologies and cultures and be respectful of the cultural differences.

Meetup and discussion groups
You can always find students who are excited by the same ideas as yours. Use your GSoC contacts to organize meetups and discussion groups. You can meet up with people who are. It's always good to put a face on the names that you've been friends with. Help each other out with coding problems.

Organize Student Chapters
You can even consider starting a local student chapter for your community if you can find enough interested people. It's a great way of socializing, making and keeping new friends and also spreading word about your community, open source and GSoC.

Review Your Project Plan

Do you have a good project schedule? Have you informed your mentor of any planned absences? Make any project adjustments you may now recognize as necessary based upon getting your dev environment setup and your new understanding of how the project works.

4 Let the Coding Begin

4.1 Working With Your Mentor

Once your project is accepted you are assigned a primary mentor and potentially secondary mentors as

well. Consistent access to mentors is one of the most valuable parts of the GSoC program. Your mentor is going to work with you throughout the GSoC program to help you be successful, but you also need to make sure you are contributing to and helping to manage your relationship with your mentor.

Community Bonding Period

The community bonding period is when you work out further details of your project plan, schedule regular upcoming meetings with your mentor, get your development environment set up and start to engage with the project's open source community. This is the time to work with your mentor on setting expectations for your interactions and how your progress is measured during the GSoC program. Hopefully, you have already participated in many discussions with your mentor, clarifying the project and expectations during the application period, but now is the time to finalize your plans.

Staying on Track

Ideally, you have already worked with your mentor to layout a clear schedule for regular meetings, reports, code check-ins and any planned time off. If you haven't, *do this now!*

Your mentors are busy people too. They are juggling many tasks and GSoC is one more added to their load. If you rely solely on your mentor to keep track of and enforce the schedule of work, your project is almost guaranteed to fail. Be proactive in keeping the schedule you've agreed to and proposing modifications before deadlines pass.

Be Respectful of Your Mentor's Time

Your mentor is volunteering her time to help you with your GSoC project. Regularly missing scheduled meetings without prior notice and failing to come prepared to meetings is likely to cause your mentor to not want to continue helping you with your project. Remember your mentor is your most valuable resource to help you complete your project successfully, treat them accordingly.

Using Your Mentor

During the coding period, you should continue to make full use of your mentor. As you implement each phase of your project plan, ask your mentor for feedback on the code as you go. Avoid working for two weeks on a chunk of code that is headed in the wrong direction.

Never be afraid to ask questions, but remember you are going to be more successful in receiving a good answer if you try to be specific and do a bit of background investigation first. Take a look at logs and try to think through the error or problem before asking for help. Equally important is to not wait until you are significantly behind in your project to ask for help.

Asking For and Receiving Feedback

Hearing criticism can be hard. It can be even harder to not respond defensively. Asking for and

receiving constructive criticism about your project from your mentor, the larger open source project community and your peers is an invaluable exercise to help you write the best code possible.

How do I ask for feedback?
During the community bonding period you and your mentor should have established a schedule for regular communication about your project progress. But what if you want additional feedback or maybe you aren't getting a very detailed response from your mentor. How do you go about asking for additional critique? A general "how am I doing" is going to probably get you a general "you are doing fine" response back. Instead, do your best to ask specific questions about specific areas.

My mentor is wrong! My project is going great! What do I do?
First, take a deep breathe. OK, now take another. Chances are your mentor is not wrong and you need to listen to their advice. This is a great opportunity to start asking questions. Try not to be defensive but rather, "I really don't understand what you mean, would you please explain?". Remember that the open culture surrounding most open source projects encourages and thrives on discourse. Don't view the feedback as negative criticism, but rather an opportunity to explore how to do something different and make your project better. Do your best to listen, learn and incorporate feedback into how you work.

Dealing with Problems

One of the best things you can do when working to solve real or perceived problems with your mentor relationship is to stop and remember that your mentor(s) is a volunteer and has non-GSoC responsibilities.

Where did my mentor go?
So you haven't heard from your mentor in a while, what do you do? First, did your mentor miss a scheduled meeting or did he just not respond immediately on IRC or to an email?

If you didn't get an immediate response outside of scheduled meeting times either on IRC or email, be patient.

If your mentor missed a meeting, then a gentle email along the lines of "did I mix up our meeting time" to your mentor is a great idea. Chances are you'll hear back shortly and things will continue on as planned. If you don't hear back, then try contacting your secondary mentor and if there is still no response, then contact your org admin.

Why is my mentor so mean?
Remember that everyone has different communication styles and you might both be communicating in a second language. Providing project critique is part of a mentor's job. That being said, if you feel like your mentor is being unduly harsh and it is effecting your project success the best approach is to talk with your mentor about his communication style in a non-confrontational manner. If you still feel like your mentor is being unduly harsh -- after your conversation -- talk to your org admin.

Why is my mentor so friendly?
Again, this could be a disconnect based on different communication techniques within different

cultures. Make sure you let your mentor know what about their style is making you uncomfortable. If things don't change for the better, contact your org admin.

4.2 Time management for students

Well, 12 weeks isn't a lot of time to write all the code and have all the fun that you're supposed to have during GSoC. Efficient time management goes a long way in helping you make the most of your GSoC experience. Even though Google puts out a timeline for making sure that things do happen on time, you and your mentor should come up with a custom time line that fits in well with your project requirements and also with your plans for the summer.

The following are some suggestions that can help you to manage your time well:

Community Bonding Period
There's already a complete section in the manual on this. (It is *that* important.) Use this time to get familiar with your community, setup your development environment and get an early start on the code development. This is time where you're getting to know your mentor and other students. Chat with them on IRC, ask questions and share ideas on projects and get ready to begin the fun ride ahead.

Be upfront about your time commitment
GSoC is a job and needs an investment of about 30 hours a week. If you know in advance that you'll need to take a couple of days of vacation/break during the Summer of Code, you should communicate that with your mentor beforehand and plan to make up for lost days.

Regular meetings with your mentor
You should absolutely, absolutely make sure you're interacting often with your mentor through mutually agreed upon communication channels. It is extremely important that you and your mentor are on the same page in terms of the project status and plans. Your mentor is your biggest resource and you should make sure that you capitalize on that well.

Have mini-goals for each week
It always helps to set short-term goals that you can discuss with your mentor. Breaking down your project in smaller tasks helps in many ways:

- It gets you started!! You have a smaller, well-defined task to work on which is easier to manage.
- Mini-goals help create a road map to get to the final output you're trying to produce.
- Smaller goals are less daunting, and completing a small goal gives you the confidence to tackle the next one.

Regular code reviews
Regular code reviews are extremely important for staying on track. Don't be afraid to ask for them as frequently as you need them. It is much better to ask for a code review after 10 or 20 lines instead of hundreds of lines. If your mentor tells you that you are going about something the wrong way, you

don't want to find out *after* writing a ton of useless code.

Maintain a log to keep track of your progress
Keeping a log of your progress is a great way to monitor progress both for yourself, your mentor and anyone else. It also turns out to be valuable in cases when you need to go reconsider a decision and figure out why you made that choice in the first place. Blogging is a good way to achieve this, since you may get good advice in the way of comments from people that you normally wouldn't communicate with, and it gives you a bit of limelight as well.

Handling the time-zone differences
It's very likely that your mentor is in a different time zone than you are. Be sure that you take this into account while you're preparing your plan. Time zones should be included for any meeting times, and UTC is often the best time zone to use, since it is not affected by Daylight Savings Time.

Be prepared for the unexpected
Things usually don't go the way you plan them. Make sure you have room for the unplanned changes. It's always best to keep aside some buffer time that you can use in case you do digress from your original plan and save yourself from considerable pain and panic attacks.

Of course, do remember that GSoC is not just all coding and working. It's about having fun too! If you manage your time well, you will have a great Summer of Code, meet interesting people and have lots of fun.

4.3 Evaluations

Evaluations occur three times (after the first, second and third months of coding). They are your opportunity to evaluate your mentor and your mentoring organization's performance. This is also your mentor's opportunity to evaluate you.

The pass or fail decision from an evaluation should not come as a surprise. You and your mentor will already be communicating, and you should be discussing the quality of your code, your participation in the community and your progress on your project to that point. If you aren't getting this feedback, ask for it.

How Evaluations Work

Evaluations are a survey that both you and your mentor fill in during the evaluation period. Google publishes the evaluation questions for both students and mentors in advance.

Evaluations are administered via the webapp you used to submit your project proposal at the start of the term. Evaluations will take about 20-25 minutes to complete. The questions focus on the scope of your project, the quality of your interactions with your mentor and your community. You, your mentor and org admins will receive an email after the evaluation window closes that tells you whether you've passed.

Your evaluations are only visible to your mentoring org admins and program administrators from Google. In some cases, Google's program administrators may need to share some details of evaluations with you and your mentor. This may happen, for example, when evaluations indicate a payment should not be made.

If you or your mentor do not submit an evaluation, you fail and will be removed from the program. Similarly, failing an evaluation means you are immediately removed from the GSoC program.

Final Evaluations and Work Product Submission

At the end of the Google Summer of Code term you are required to provide a link to the work you created during your current GSoC participation, in addition to your evaluation. Essentially the target of the link should contain a short description of what work was done, what code got merged, what code didn't get merged, and what's left to do.

Depending on what portions of code are included in the scope of your project, you may need to submit diffs between code that you've written and others have written, or even include an entire branch of the code base. Discuss this with your mentor and your community and use your best judgement.

Payments - Be Patient!

Successful completion of your evaluations triggers the payments to your account or your payment card. This takes a few business days to process. Please be patient.

5 APPENDICES

5.1 Strategies for getting your code committed

A key goal of GSoC is to produce useful code that is integrated into the code base of your community. Every organization has different standards for code submissions, but there are some general rules you can follow that will help make your code easier to integrate:

- Follow the documentation guidelines. Some organizations have a policy that code must have some documentation before it can be committed.
- Make sure the tests pass! You have tests, right?
- Include lots of useful comments in your code. Comments make people happy.
- Tests are sensitive to the versions of software, operating system and libraries you are using. It is a good idea to run your tests on a few of the most commonly used versions of libraries or prerequisites, if possible.
- Document exactly what kind of environment you wrote the code in, and whether certain things may be dependent on your operating system or platform. For example: "This patch was tested

on Linux x86 and OS X Intel, but may have some issues with FreeBSD"

- Read through recent commits to the repository that you would like your code to be in. You will often learn about something relevant to your code, such as a macro that makes your life easier or a special trick for avoiding bugs.
- Commit often. You can always maintain a local branch of the code to which you keep committing regularly. This keeps you from losing work; it also gives you something concrete to show your mentor and get feedback on.
- Take the feedback on your code positively. Appreciate the comments and suggestions you receive and use them for your next commit. It certainly helps you write better code and grow as a developer.

There's something really satisfying and fulfilling about getting your code accepted and seeing it used by many others. Getting your code submitted is your first step towards achieving that accomplishment! More importantly, it helps you in honing your skills as a great developer and contributor.

Pro Tip: You can also ask other students to review your code for you and give you feedback. Use other students in your community and the GSoC program as a resource.

5.2 Staying Engaged with Your Community

A glorious summer has ended. GSoC is now over—but only officially. Your obligation to the program has been met, but your opportunities still abound. Make the "summer" last by staying involved in your community. Yes, it's now *your* community. Make the most of it. We have a couple of suggestions that can help you make your experience last beyond the summer.

Maintaining Your Project

You worked on a wonderful project during the summer. You can't abandon it just because the summer ended! After all it's your *baby*. One way to make sure that your project keeps on growing is by maintaining it well, adding new features to it. There might be a few loose ends that you weren't able to tie up during the coding period. If there's anything in (or not in) your code that's keeping you awake at night, you can always work on that after the coding period ends.

> "Please do the communities a favour by not abandoning your projects after the GSoC results are announced, or the boogeyman will get you.[!!!] "
> Lalith Suresh P., Network Simulator 3, GSoC Student 2010

New Projects

Do you have other amazing ideas for projects? Why wait until next GSoC to get started on them. You should start working on them right away. You know the folks in your community well now, and you know where to shout for help. Just get started on your wonderful new project idea; you can even get it accepted as a GSoC project next year!

Retreats and Hackfests

A lot of organizations host opportunities to meet face-to-face throughout the year. Attend if there's one happening close by. They're great fun! In fact, you can help organize such activities or even start a local chapter of your community in your area if you have enough people around.

Spread the word

You had a great time this summer. It's your turn now to tell your friends about it. GSoC is also about growing the open source community. Share your GSoC success story with others and inspire them too!

Present your work

You can present your project at some of the open source conferences. The open source community is very much interested in finding out about the cool things that people have been working on. Conferences are a great way of doing that. If possible, you should attend a conference and use the opportunity of presenting your project to other people. It gets you great feedback on your project and helps you find out about other cool things happening in the open source world.

Next Summer

There are so many ways that you can help your organization and the open source community for the next Google Summer of Code. You can submit project ideas, apply again as a student or even become a mentor for a project. You can even contribute by just idling in the IRC channels, helping out the new students who come looking for advice.

Stay connected

Everyone likes to hear back from friends and people they've worked with. Be sure to stay in touch with your mentors, other students and the many great friends that you've made during GSoC.

> "GSoC may be over, but I'm really hoping my contribution to the Congress application will go beyond that. I'll be sure to stay in touch with Eric and the Sunlight Foundation. "
> Evelina Vrabie, SunLight Foundation, GSoC Student 2010

6 APPENDICES

6.1 About This Manual

This manual was written during a Book Sprint sponsored by Google and facilitated by FLOSS Manuals. The manual was written in two days but the maintenance of the manual is an ongoing process to which you may wish to contribute. A second sprint occurred in 2010 to update the manual (adding

the Org Admin section) and to write the Students Guide ("Flipping Bits not Burgers").

An update of the manual was done in January 2017 and what was quite impressive is how few changes had to be made as all of the advice the original authors of this manual still stand today.

Since the manual may be updated at any time, you may wish to periodically check here for updated versions:
http://www.flossmanuals.net

You can also find the electronic book (for use with Android ebook readers software, Sony Reader etc) here:
http://objavi.flossmanuals.net/books/

How to Contribute to This Manual

If you would like to contribute then follow these steps:

1. Register on the front page (http://booki.flossmanuals.net/) by clicking on **Register** in the sidebar.
2. Under **Create Account** add your desired details
3. Click **Create account**.
4. Booki will generate the account and display a Your account has been created message before redirecting you to your **Profile Page**. Your user page is also accessed through **My Profile** on the menu.
5. Contribute by visiting http://booki.flossmanuals.net/GSoCStudentGuide/_edit/. Press 'edit' and start work on a chapter.
6. **Mailing List**
   For discussing all things about FLOSS Manuals join our mailing list:
   http://lists.flossmanuals.net/listinfo.cgi/discuss-flossmanuals.net

For more information on using FLOSS Manuals you may also wish to read our manual:
http://en.flossmanuals.net/FLOSSManuals

About the Authors

This manual exists as a dynamic document on flossmanuals.net, and over time will have an ever-increasing pool of authors and contributors.

Except were noted the following individuals were part of both the 2009 and 2010 GSoC Book Sprint. We thank them for their tireless efforts to create this first-of-its-kind volume.

Alexander Pico
Alex works at the Gladstone Institutes in San Francisco as a Bioinformatics Software Engineer. He holds a PhD in Molecular Neurobiology and Biophysics and has over 10 years of bioinformatics experience in data mining, analysis, visualization and integration. Over the past 5 years he has led the

GenMAPP pathway analysis project, managing software development and coordinating research projects involving wet lab biologists and senior programmers. Alex is a member of the Cytoscape core development team, a creator of SNPLogic.org, and a founder of WikiPathways.org. He has also administered GenMAPP's participation in the Google Summer of Code program for the last 4 consecutive years.
http://nextnucleus.org

Bart Massey
Bart Massey graduated Reed College in 1987 and spent two years as a software engineer at Tektronix, Inc. He received his MSc in Computer Science from University of Oregon in 1992 and his PhD in 1999 for work with the Computational Intelligence Research Laboratory there. Since then, Bart has taught at Portland State University, where he is currently an Associate Professor, and for the Oregon Master of Software Engineering program. Bart is Secretary of the X.Org Foundation Board; his current research interests include open tech, software engineering, desktop interfaces and state space search.
http://www.cs.pdx.edu/~bart

Jonathan "Duke" Leto
Jonathan is an open source hacker who currently focuses on the Parrot Virtual Machine and Rakudo, Perl 6 on Parrot, as well as being the maintainer of many CPAN modules, many with a focus on scientific computing and cryptography. He first was a mentor for Math::GSL in GSoC 2008 and then became organization administrator for The Perl Foundation's involvement in GSoC 2009 as well as being the mentor for Math::Primality. Jonathan received a masters in mathematics from University of Central Florida and has published several papers in the field of differential equations. Currently he works in the Bioinformatics sector and consults for Git migrations and training. He enjoys discovering wheels within wheels.
http://leto.net

Jennifer Redman
Jennifer is the founder and CEO of Buunabet, a technical consulting company that helps businesses and organizations integrate open source software into their computing infrastructure. Jennifer is currently the Associate Systers-Keeper for Systers, the oldest International online community of technical women. One of the founders of the Systers open source project, she participated in GSoC 2009 and 2010 as an org admin and mentor. Previous careers include canvassing for Greenpeace and as a staff member on a national (and successful) presidential campaign. She also likes to travel and reads a lot of books.
http://buunabet.com

Selena Deckelmann (2010)
Selena is a major contributor to the PostgreSQL project. She founded Open Source Bridge conference, a conference for open source citizens. She helps run PDXPUG, a PostgreSQL Users Group, and is part of Code n Splode, a programming group whose goal is to get more women involved in open source. In her spare time, she likes to mix drinks for her local user groups, and fetch eggs from her chickens.
http://chesnok.com/daily

Carol Smith
Carol works for the Open Source Programs Office at Google administering the Google Summer of Code program. She has worked at Google for 5 years in a variety of positions. She has a degree in photojournalism from California State University, Northridge and is an avid cyclist.
http://www.fossygirl.com

Malveeka Tewari (2010)
Malveeka is currently a graduate student in the CS dept at UC, San Diego. She participated in GSoC 2009 as a student for Systers and again as a mentor for Systers in GSoC 2010. She is also a member of Systers and WIC (Women in Computing) group at UC, San Diego and has helped in organizing events in her school for engaging and encouraging women in computing. Her favorite past times including hiking, cooking and reading books. She is a big fan of P.G. Wodehouse and Agatha Christie.
http://cseweb.ucsd.edu/~mtewari/

2009 Particpants

Olly Betts
Olly is the lead developer of the Xapian search engine library. He's been working on Xapian for 10 years, and makes a living as a freelance developer and consultant on Xapian-related projects. In GSoC, he's represented SWIG as a mentor in 2008, and a mentor and co-admin in 2009. Olly is originally from the UK where he studied mathematics and then computer science at Cambridge University, but now lives near Wellington in New Zealand. He once broke a toe falling off a cliff in Majorca.
http://survex.com/~olly/

Leslie Hawthorn
Leslie held various roles at Google before joining the Open Source Programs Office in March 2006. Her first project after joining the team was spinning up Google Summer of Code 2006 and she has managed the program ever since. She also conceived, launched and managed the Google Highly Open Participation Contest, an initiative inspired by GSoC that helps pre-university students get involved with all aspects of open source development. Mentoring in open source communities is one of her personal passions, along with humanitarian uses for open source software. She loves to cook, read and can occasionally be found pining for illuminated manuscripts. She also likes to think of herself as a superb filker.
http://www.hawthornlandings.org

Facilitation (2009 & 2010)

The Book Sprints were facilitated by :

Adam Hyde
Adam is the founder of FLOSS Manuals and project manager for Booki. FLOSS Manuals is a community of 1500 (at the time of writing) volunteers that create quality free documentation about free

software. FLOSS Manuals is pioneering the Book Sprint methodology that enables the development of well written manuals on free software in 2-5 days. Adam has facilitated over 15 Book Sprints on Free Software including Inkscape, OLPC, Sugar, CiviCRM, Firefox, Introduction to the Command Line, Digital Foundations (conversion to free software examples), Ogg Theora, How to Bypass Internet Censorship, Open Translation Tools, PureData, Video Subtitling and now the Google Summer of Code Mentors Guide. Adam is also the Project Manager for the development of 'Booki' - the free software Collaborative Authoring Platform (see below).
adam@flossmanuals.net

The Platform

This book was written using Booki and output to templated HTML, book formatted PDF, and epub by the same platform. Please consider helping us develop this wonderful collaborative authoring and book production software. It is GPL and more info available at http://booki-dev.flossmanuals.net. You can also see booki at http://www.booki.cc.

6.2 History of GSoC

Google Summer of Code began in 2005 as a complex experiment with a simple goal: helping students find work related to their academic pursuits during their school holidays. Larry Page, one of Google's co-founders, was pondering the age-old problem of scholastic backsliding: students work hard and learn a great deal during the academic year, but without the right employment opportunities or other pursuits outside of school, technical skills atrophy rather than get honed and expanded. Larry wanted Google to help solve this problem.

The obvious solutions failed on geographical grounds. If a student wasn't in the optimal location, obtaining a useful internship could be difficult if not impossible. Finances were also a problem; many internships are low paying or entirely unpaid, making it difficult for students to take the right job while still paying the bills. Finally, even if a job was available in a technical field, it would not necessarily introduce a student to the broad set of skills required to do software development well. For example, creating a website for a local non-profit requires technical skill and would no doubt be personally gratifying, but wouldn't necessarily require using an IDE, checking into source control, or creating tests.

The perfect answer came from encouraging students to participate in open source projects. Open source development occurs online, both solving the geography problem and giving students the chance to work in a globally distributed team. Working on an open source project provides exposure to the entire software development process and tool chain. Students could enjoy the added benefit of having a body of reference work available to future employers or committees. Even better, students would get the chance to work on a code base under active development rather than a lab project or other single use assignment.

A cash stipend from Google allowed students to focus on their development work rather than getting a job unrelated to their academic pursuits. The final part of the puzzle was finding projects who were excited to find new contributors and to provide helpers to get these new folks up to speed both technically and socially. The first year 40 projects participated; 400 students began the experiment.

In 2016, the twelfth Google Summer of Code wrapped up with more than 85% of the 1206 student participants in the program successfully completing their projects. Best of all, most of the organizations participating over the past eleven years reported that the program helped them find new community members and active committers.

You can find more information about each year of Google Summer of Code on the program statistics page on the History tab of the GSoC website: https://developers.google.com/open-source/gsoc/resources/stats

6.3 Additional Resources

We've collected our favorite and most useful resources specific to GSoC here.

General Resources

You want to take a look at the Program Rules and Frequently Asked Questions each year to make sure you have a good idea of the rules for the program. There's a wealth of information included in the Program Rules and FAQs (and this manual), even for experienced participants. You can always find the latest information on the site: https://g.co/gsoc

Additionally, these resources are quite helpful:

Program IRC Channel
Several knowledgeable folks (mentors, org admins, former students, current students and Google Program Admins) hang out in #gsoc on Freenode and would be happy to give you a pointer in the right direction. And many organizations have their own IRC channels that you can hang out in.

Blog Posts
You can find material related to GSoC on the Google Open Source Blog at https://opensource.googleblog.com. Your project may have a blog or newsletter where GSoC information was published in the past, as well.

Knowledge Base
If you're looking for advice for mentors or students, program promotional materials, presentations about GSoC, etc., start with the program site https://g.co/gsoc, particularly the Resources section.

List of Organizations
You can see a list of the mentoring organizations that participated in previous years of the program by going to the program site and looking at the Archive site: https://developers.google.com/open-source/gsoc/past-summers

Mailing Lists

There are two program mailing lists.

Announcement Only List
For announcements from Google's program administrators only. Used infrequently.
http://groups.google.com/group/google-summer-of-code-announce

Program Discussion List
Open subscription list for the program. General talk about the program, light traffic except during the launch phase of the program each year. It is always excellent for you to stop by and encourage a newbie, though, so please don't totally ignore this list.
http://groups.google.com/group/google-summer-of-code-discuss

Books

Producing Open Source
Written by Karl Fogel. Excellent guide to Open Source development. Its available free online.
http://producingoss.com/

Google Summer of Code Mentors Guide
http://en.flossmanuals.net/GSoCMentoringGuide

Teaching Open Source
by Karsten Wade
http://teachingopensource.org

Associated Projects

Teaching Open Source
http://teachingopensource.org
irc : freenode #teachingopensource

What to Do When the Unexpected Happens?

1. Contact your Mentor. He or she can help you figure out what to do next or contact Google for more help.
2. Contact the Org Admin.
3. Talk to Google's Program Administrators. They have plenty of experience with all the corner cases and strange issues that can arise during GSoC. Email gsoc-support@google.com for help.

6.3 Proposal Example 1

"Database Abstractions" By Kanika Vats, Systers, 2009

Abstract

Systers use GNU mailing list manager Mailman2 which currently uses Python pickle files to store its data. Systers moderators have customized it to make use of PostgreSQL database. They make use of raw SQL statements and python db-api which makes the code :

- Dependent on the existing database
- Reduces the efficiency and maintainability

The project idea aims at making the code :

Independent of the database by making the use of python classes and objects to interact with the database rather than direct SQL statements.

- This will be achieved with the help of an ORM (Object Relational Mapper). Storm will be our choice of ORM.
- Also, Systers aim at bringing this feature upstream and incorporating this feature in the yet to be released version - Mailman3 (which will switch to use a database) - so that the open source world can benefit themselves with the addition of this feature.

Thus, mapping existing schemas of the Systers database to an object oriented paradigm and determination and incorporation of necessary modifications in the database needs to be done so that it fits cleanly and nicely into Mailman3's Architecture.

Proposal Timeline

Before April 20:

- To familiarize myself completely with Mailman2's functionality and architecture.
- Study of the customized files of Systers Mailman available in the Launchpad Baazar version control.
- To familiarize myself with Storm(ORM that we will be using)

April 20 – May 23 (Before the official coding time):

- To do self coding with Storm to improve my further understanding and ease of use with this ORM and database(PostgreSQL)
- During this period I will remain in constant touch with my mentor and the Mailman community. I will remain active on IRC and Mailling lists to discuss and finalize on the modifications (if any) that needs to be on existing schemas and design of new schemas (if needed to fit cleanly with Mailman3's Architecture)

- Thus with the help of my mentor I will become absolutely clear about my future goals,the final database implementations that need to be done as well as the approach that I will follow to map the schemas to the Object Oriented Paradigm.

May 23 – June 18 (Official coding period starts):

- Define all the required Relations(Tables) in my local database using STORM.
- Define all the corresponding Python Classes and Objects that will store,modify and retrieve data in database.
- Define all the interactions that Systers perform with their database (virtualize or stimulate all interactions) in STORM that will deal with my local database.

This will help in testing of the proper working of the entire basic code changes that we will later on incorporate in Systers Source code.

June 18 – July 5:

- Bringing about the decided changes in the Relational Schemas of Systers database.
- Replacing parts of the above code in their respective places in the Systers source code. (This should not take much time as most of the functionality has been implemented in the previous step).
- Testing the overall working of each and every module of the modified source code with the help of Python Test Suites.

JULY 6th MID TERM EVALUATION

July 6 – July 15:

- Making further changes in the code to improve the Functionality, Exception handling, Bug Removal.

July 15 – July 25:

- To be in constant touch with the Mailman3's developers and to let them know about our progress.
- Most of the time will be consumed for rigorous testing and bug fixes.

July 25 – July 31:

- For Documentation

A Buffer of two weeks has been kept for any unpredictable delay.

6.4 Proposal Example 2

"Reactome-Wikipathways Round-trip Format Converter" by Leontius Adhika Pradhana, GenMAPP, 2010

Problem description

Reactome is a "free, online, open-source, curated pathway database encompassing many areas of human biology". Each pathway in Reactome is manually curated -- peer-reviewed and cross-referenced with other database -- and thus has great reliability. Another pathway database website WikiPathways, by contrast, lives on the "wiki spirit" allowing anyone to edit and annotate pathways in the website. This makes WikiPathways an ideal venue for staging new pathways to be included in the official Reactome database, as well as a place for the community to review and make changes to pathways which may end up as an official amendment in Reactome.

However, the two websites use markedly different data structure to store their pathways: WikiPathways uses GenMAPP Pathway Markup Language, a vector graphics format similar to SVG; Reactome internally stores the pathways in a proprietary semantic database schema. The formats differ not only in their presentation but also in their focus of data stored, making information exchange difficult.

Recent development of Reactome introduced a new proprietary graphical XML format akin to GPML. This XML format adheres to SBGN specification which semantically defines symbols representing biological systems. This project will provide the means to convert to and from GPML and the new Reactome XML format.

Implementation plan

The project consists of three components:

GPML to Reactome XML layout converter
Unlike the Reactome XML format, GPML mainly describes the graphical representation of pathways and does not contain semantics of the reactions. To produce Reactome XML, therefore, the converter must employ certain heuristics to infer semantic relations from graphical representation and eliminate ambiguities. The heuristics will follow SBGN as close as possible while still retaining compatibility on other formatting conventions.

Reactome XML layout to GPML converter
The Reactome XML layout contains further pathways data that are not viewable in GPML. Therefore, the resulting GPML after conversion will contain additional comments containing the Reactome data or at least their identifiers, so that when a back-conversion (from the GPML to Reactome XML) occurs, data will be preserved.

During the conversion, SBGN semantics will be employed to provide unambiguous back-conversion to Reactome XML later when necessary. Some additional shapes might need to be implemented in

GPML, or alternatively comments can be written to differentiate SBGN symbols that do not have corresponding graphical representation in GPML.

During the development of this converter a schema for Reactome XML will also be made so that converted test files can be easily validated.

Automatic update mechanism between WikiPathways and Reactome
A separate script will be made that periodically pulls updates from WikiPathways and convert it to Reactome XML layout. The script can be set to automatically update the pathways in Reactome if correct credentials are provided. This will mainly be done for pathways that are already tagged to be high quality.

The script will also pull updates from Reactome and push new pathways to WikiPathways. Only Reactome pathways that have XML layout will be pushed to WikiPathways.

Deliverables

- an XML schema to validate the new Reactome XML format;
- a GPML to Reactome XML layout converter and Reactome XML layout to GPML converter, which will be available both as command line tool and a library that can be integrated with WikiPathways infrastructure;
- a system using the above converter, integrated to WikiPathways, that will periodically check for updates on both WikiPathways and Reactome and update the websites accordingly;
- proper documentation and tests for the above-mentioned components.

Timeline

This week-by-week timeline provides a rough guideline of how the project will be done.

3 -- 16 May

Familiarize with the code and the community, the version control system, the documentation and test system used, and the new Reactome version.

17 -- 30 May

Write the Reactome XML layout schema and the command line Reactome XML to GPML converter, keeping in mind that the internals are to be used subsequently as a library.

31 May -- 6 June

Test and document existing code more thoroughly.

7 -- 20 June

Determine algorithms used to convert GPML graphical representations to Reactome XML. Then, write

the command line GPML to Reactome converter, keeping in mind that the internals are to be used subsequently as a library.

21 -- 27 June

Test and document the GPML to Reactome XML converter and the heuristic algorithm more thoroughly.

28 June -- 11 July

Ensure that round-trip conversion works flawlessly (i.e. no data is lost when converting GPML to Reactome XML to GPML again, and vice versa). Also test and document round-trip conversions.

12 -- 25 July

Integrate the converters to WikiPathways. A system that periodically check for updates on both WikiPathways and Reactome and update the websites accordingly is written.

26 July -- 1 August

Test and document the periodic push/pull mechanism more thoroughly.

2 -- 16 August

Further refine tests and documentation for the whole project.

6.5 Glossary

+1

The shortest way in the geek world to say "I agree with this" or "This is a great idea". It is often used when others have already fleshed out the details and a consensus of how many agree/disagree with the sentiment. It is worth noting to your students if your project uses this as a voting signal so they do not accidentally comment on issues when, as newbies, they should be observing rather than commenting/voting.

-1

The opposite of +1. Often accompanied by an explanation why, if you are lucky.

Committer

An individual who has special rights in an open source project. While the scope of this term varies by project, the general idea is that this individual is able to check in source code to the project's main repository.

Community Bonding Period

The period of time between when accepted students are announced for a particular year of GSoC and the time these students are expected to start coding. This time is an excellent one to introduce students to the community, get them on the right mailing lists, etc. See the "Mind the Gap" section for more details.

DVCS

Distributed version control system. A version control system that does not require talking to a centralized server.

FLOSS

Free/Libre Open Source Software. Likely the most inclusive acronym to describe the software produced for GSoC.

GSoC

Google Summer of Code

IDE

Integrated Development Environment

IM

Instant Messenger

IRC

Internet Relay Chat

JFDI

Just Fabulously Do It. Use your imagination. Ask for forgiveness, not for permission. :)

Lurk

To spend some time watching. Often used in reference to a mailing list where you will read the posts but not make any posts yourself or an IRC channel where you watch how people interact but don't say anything.

Mentor

Someone who helps a student with their project proposal. See the What is GSoC section for more details.

Organization

An open source, free software or technology-related project that mentors students for Google Summer of Code. Also known as a mentoring organization.

Organization Admin (Org Admin)

Cat herders for each open source project participating in the program. Often abbreviated to org admin. See the What is GSoC section for more details.

Program Administrator

Google employees who run the program. See the What is GSoC section for more details.

RTFM

Read **T**he **F**LOSS **M**anual ;)

Secondary Mentor

A person who helps out a student's assigned mentor. At time of writing this manual, the GSoC online system only allows one mentor to be officially assigned to a student proposal, as one person must be responsible for submitting evaluations, etc. However, it is quite common to have multiple mentors for one student.

SMOP

Simple Matter of Programming

Summer

Not so much a season as a state of being. While the program is run during the Northern Hemisphere's Spring and Summer, the "Summer" in Google Summer of Code is actually a play on the "Summer of Love.

TDD

Test Driven Development

Use Case

A use case describes what a user can do with a particular software system.

UTC

Coordinated Universal Time.

Waterfall Model

A sequential software development process that usually doesn't work.