

Mentoring in Free Software Projects

[a review of 6 years of GSoC and SoK in KDE and what we learned so far]

Lydia Pintscher
lydia@kde.org

ABSTRACT

This paper provides a review of KDE's Google Summer of Code (GSoC) experience from 2005 to 2010 and Season of KDE (SoK) in 2006, 2008, 2009 and 2010. It also goes into lessons learned from those mentoring experiences that can be applied to other projects taking part in GSoC as well as taking on their own mentoring programs.

Keywords

Google Summer of Code, Season of KDE, mentoring, Open Source, Free Software, community building

1. INTRODUCTION

KDE is a community creating a free and open desktop environment, applications for it and services around it. It relies on contributors with various backgrounds. Many work on it in their spare time, others are being paid by companies to work on KDE software. Contributors are working on code, promotion, documentation, support, artwork and much more. To be able to grow and stay healthy the community needs to get new people into the project all the time. Those need to be recruited, made familiar with the project's work flow, shown what areas they can help out in depending on their skill set and be guided into becoming an integral part of the team. There are two distinct ways of doing this - dedicated mentoring programs and mentoring-as-you-go. We will concentrate on the former here and talk about what did and what did not work in KDE's mentoring programs.

2. KDE AND GSOC - THE STORY SO FAR

2.1 What is GSoC and SoK?

GSoC is a program run by Google. The aim is to give students a summer job working on Free Software projects to give them real-world software development experience and get them in touch with the Free Software community and at the same time get a lot of code written for the projects taking part. It was started in 2005 and KDE has taken part

as one of the biggest mentoring organisations each year so far.

Each year at the start of the program projects apply to become mentoring organisations and a few of them get selected. They collect project ideas and then students can write a proposal and apply for one of those ideas or propose their own. The projects review the applications and select the students they want to work with for the summer and match them up with mentors from their current contributors. Google then allocates a number of slots to each organisation. The accepted students then get to know their community better during a community bonding period. After that the actual coding starts. Students are expected to work on their project full-time. In the middle and at the end of the coding period the mentor and student are asked to evaluate each other. Depending on the outcome of the evaluation by the mentor the student gets paid or not. The midterm evaluation determines if the student can continue or not. Each student and mentor also receive a t-shirt ("coding for cotton") and the organisation gets a certain amount of money for each accepted student.

KDE gets a lot more excellent applications than there are slots to fill each year (about 3 good applications for 1 slot). As it would be a shame to let a lot of very motivated students go away, SoK was started where students can work on their GSoC project with a mentor by their side but without being paid. They still get a t-shirt and a certificate. The scope and timeline of their project can be adjusted as they can't be expected to work on it full-time. They are however asked to at least keep the timeline close to that of GSoC.

2.2 What is Mentoring-As-You-Go?

Mentoring-as-you-go is the opposite to a dedicated mentoring program like GSoC and SoK where the mentee has a mentor and task assigned. Here the community as a whole takes on the job of the mentor and the mentee is generally more self-driven. It is usually not formalized and lacks the pressure to complete a task. KDE facilitates this by offering junior jobs for example.

2.3 Dedicated Mentoring vs. Mentoring-As-You-Go

Dedicated mentoring programs have a lot of advantages. They make someone responsible for the mentee, they give concrete goals, timelines and rewards and they often have a defined start and end which helps create buzz and get peo-

ple's attention. Knowing that one was accepted from a large pool of applicants is a boost for one's self-esteem and gets people to want to justify those expectations. Those are at the same time their disadvantages though. If the student finished the project he might consider his work done and leave. The buzz that is created might attract people who are not up to task at all and waste time. Knowing that one was accepted for a huge task can make people fail because of performance anxiety. And last but not least the reward is seen as problematic as it shifts the motivation to contribute to a Free Software project especially in the case of money being the reward.

2.4 GSoC by Numbers

The following statistics only include successful students as data for the others was unfortunately not available.

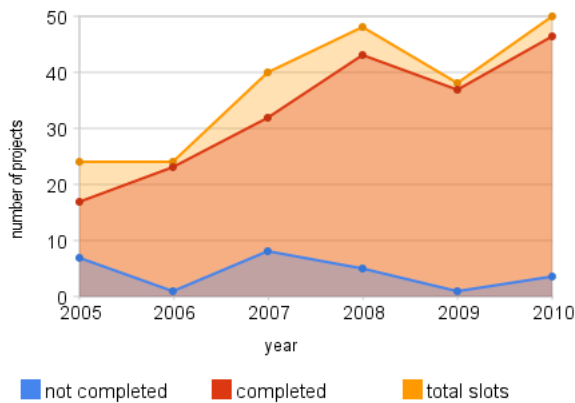


Figure 1: projects/year

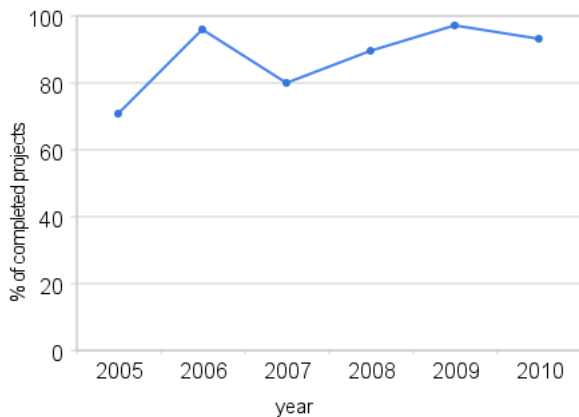


Figure 2: percentage of completed projects/year

Figures 1 and 2 show the number of students that worked with KDE and how many of their projects were finished successfully. A dip can be seen in figure 2 in 2007 which is likely caused by the relatively big increase in slots allocated.

Figures 3 and 4 show how many of the students who finished their projects are still committing code to KDE's source code repository X months after their GSoC ended. (Their last commit was taken as a reference for determining activity.)

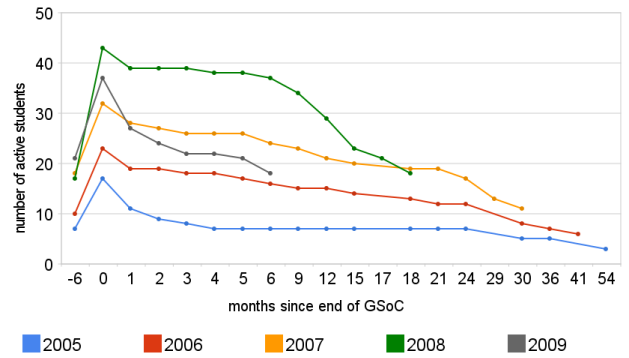


Figure 3: number of active students after X months

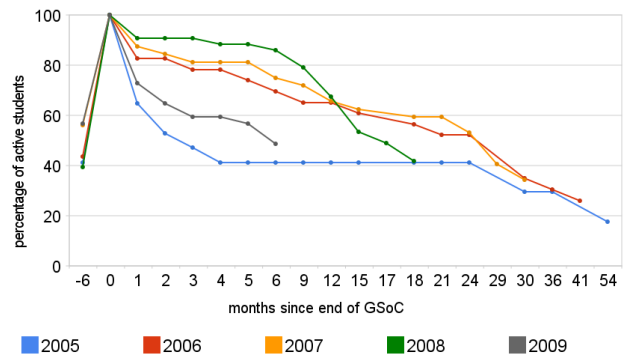


Figure 4: percentage of active students after X months

It can be seen that quite a few of them give up in the first two to three months. Who is still active after that is very likely to stay with the project for some time - some of them many years.

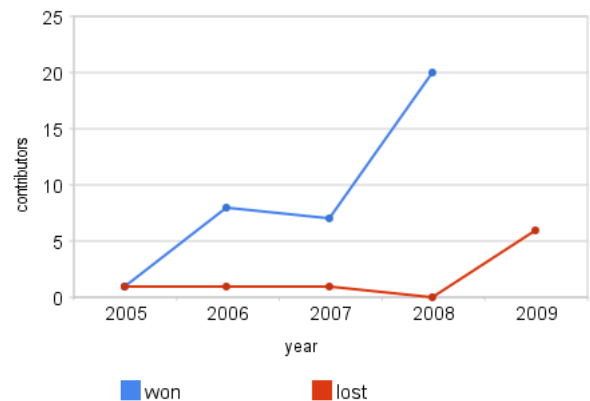


Figure 5: lost and won longterm contributors

Figures 5 and 6 show the number of longterm contributors KDE has won and lost due to GSoC. A longterm contributor counts as lost due to GSoC when he contributed to KDE for at least six months before the program started and less than three months after the program ended. A longterm contrib-

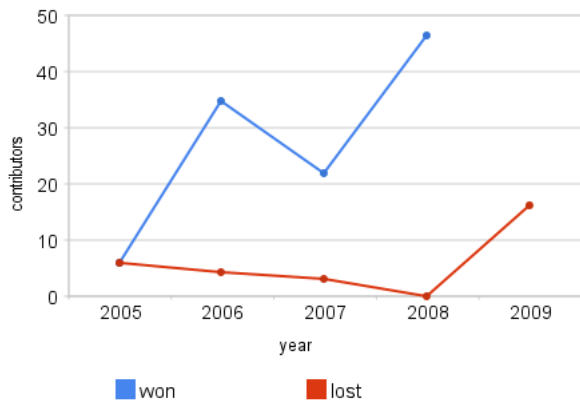


Figure 6: lost and won longterm contributors relative to the number of projects

utor counts as won when he contributed to KDE for less than three months before the program started and continues to contribute for at least six months after the program ended. It can be seen that we have reduced the number of people we lose constantly until 2008 and then had a spike in 2009. Also noticeable is the dip in the percentage of won longterm contributors in 2007 - again probably due to the big increase in the number of total slots. In 2008 nearly 50% of the students continued to contribute for at least 6 months. The most important part however here is probably that each year a lot more (or as many) new contributors were won than lost. Also some of the students that were lost for KDE might not actually be lost for Free Software. Students have changed projects inside KDE so it is reasonable to assume that at least some of them moved on to other projects.

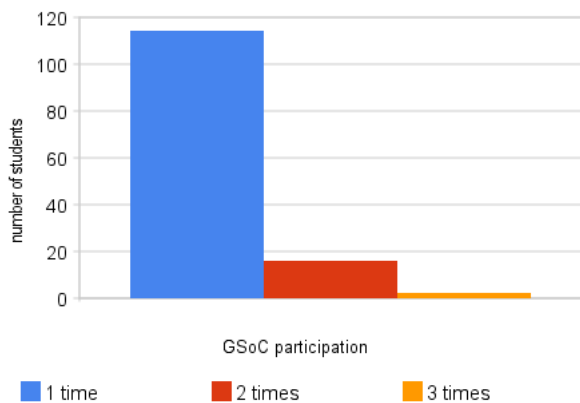


Figure 7: the number of times a student participated (2005-2009)

As shown in figures 7 and 8 most of the students took part as a student only once while mentors took part several times. However quite a few of the students returned as mentors in the next years. Most of the mentors had only one student in a given year (figure 10).

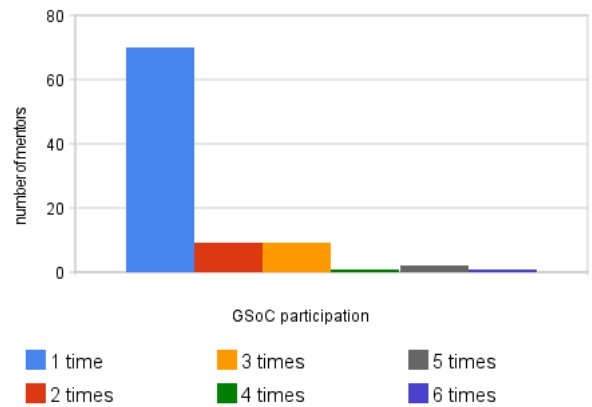


Figure 8: the number of times a mentor participated = number of students mentored (2006-2009)

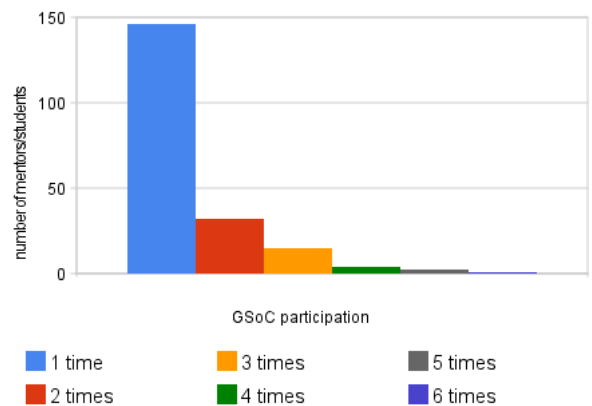


Figure 9: the number of times a person participated as either student or mentor (2005-2009, not including mentors of 2005)

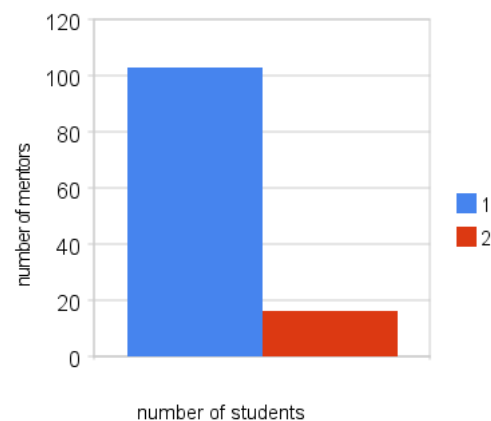


Figure 10: the number of students a mentor had in one year (2006-2009)

3. LESSONS LEARNED SO FAR

3.1 Intimidation and Encouragement

GSoC is extremely exciting opportunity for a lot of students. At the same time it is also an absolutely intimidating idea for a lot of them for various reasons. These reasons include: Working with big names like Google, the organisation they are with, their mentor who might well be one of the famous people in that organisation and of course the other people in the community. Working with the founder of the project you have been adoring for years is not an opportunity you get often. On the other hand the publicity created by those big names and the large number of organisations involved attracts a lot more people than would usually be reachable for a single project.

A lot of students think they are not good enough for the task while being absolutely capable of completing the tasks they could work on. Some of them apply despite having doubts if they are really good enough, some only apply after a lot of encouragement from their potential mentors but a lot do not even apply. Communities are losing out on a lot of potential contributors - not only for GSoC but also in day-to-day activities. They need to be given a way out. This could for example be something like SoK where it is ok for the student to work on a smaller task and he doesn't have to compete with other students to hand in the best application and promise the best outcome he might not realistically be able to deliver. The other option is regular advertising of small tasks that people can take on according to what they think they can manage.

Students who doubt their abilities to the point of never actually having really tried to program on any large program can gain tremendously in a GSoC-like setting. It can give them the much needed pressure to pull through with their project and gain confidence in their abilities. They just need to be encouraged to take that first big step.

Encouragement matters a lot.

3.2 The Application Period

It is important to have a variety of ideas in terms of difficulty, topic and how likely they are to attract a lot of applications. There are usually a few rock star ideas that get a large share of all applications which might lead to not being able to accept excellent students simply because they applied for the wrong idea. In this case it is useful to have other ideas they can be lead to.

The most important thing during the application period is probably to give feedback early and in a timely manner. It is the only way to make sure the final application is complete, realistic and what both student and organisation want.

Students will email mentors and admins privately. They should however be encouraged to go public right away even with drafts of their application to make sure they get (friendly!) feedback from the community.

Students know that they are being evaluated against a pool of other talented students. Some of them will promise much more in their proposal than they will be able to deliver later. It needs careful judging if the student is able to deliver what he promises or if it would be better to reduce the scope of the proposal. Some mentoring organisations require students to

write a patch or do other tasks during the application period to do this. However if applications are reviewed thoroughly and the pool of applications is large enough it might not be needed for judging the student's applications. It will however reduce the number of applications the organisation receives significantly.

Close to the deadline there will always be a rush of last-minute applications. Most of them will not be of high quality if students were encouraged to apply early and there will always be students who miss the deadline. It is important to make the students aware of this during the application period.

3.3 The Application Review Period

To give all applications a fair chance the actual voting on applications should not start before all applications are received. Otherwise early submissions have a big advantage.

For a large organisation like KDE it has proven to be very successful to let each subteam decide on their own proposals. They will know best which feature they need to be developed and which student will likely successfully do it. Also starting at around 50 applications it is not possible for everyone to review every application. It should however be encouraged that everyone reviews at least a few applications outside his team to make sure they fit within the larger goals of the organisation for example.

3.4 Struggling Subteams

There are always subteams struggling more or less to keep their community alive. It is tempting to give slots to them to help them gain momentum again. Overall however this is a bad idea. The student will have a hard time finding the help he needs if there is no real team to support him. If the subteam is struggling because the contributors do not have enough time for it the same will likely happen with mentoring the student. Feedback to organisation admins and program evaluations will be hard to get. The student deserves better and the success rate will decline. An exception to this is accepting students who have worked in that team before and know what to expect and will not need a lot of help. But even in this case it is not recommendable.

3.5 Oldies vs. Newbies

It is tempting to give a lot of slots to people who are already contributing to the project for a while assuming they are a safe bet and will finish their work without needing much help. This assumption is however unfortunately wrong. There have been (too many) examples of longterm contributors not finishing their project. They are not immune to it. Also as shown in section 2.4 there is a risk of losing them forever. Of course many of them finish their project and do things a newcomer would not be able to do.

People completely new to the project will likely need a lot of help at the beginning but it is time well spent on integrating a new team member. Also they are often joining the team with a lot of much needed enthusiasm. On the other hand it is very hard to tell if they will likely finish their project.

It is a fine line to walk.

3.6 Who Should Mentor?

Mentoring should always be done by someone who is known in the team and has worked with it for a longer time - preferably a long-time contributor. This is likely the best way to ensure the student gets well integrated into the team.

Each mentor should only have one student. Preferably each student has a mentor and a back-up mentor assigned. If a mentor needs to mentor more than one student it should not be in his first year. The necessary time investment is often underestimated.

3.7 The Community Bonding Period

The community bonding period should be spend on getting the student set up with a development environment, get them commit rights (KDE's policy is very permissive here), subscribed to mailing lists and other important development tools.

It is essential to not let the student get discouraged here already as those are potentially the first real interactions with the wider community.

3.8 The Coding Period

Students should be encouraged to post regular status updates. Once a week has been found to be a good time span. It helps the mentor keep track of the progress and more importantly if it is public it keeps the community in the loop and gives them an opportunity to give vital feedback. It also urges the student to actually get something ready to show at the end of the week and makes it easier to spot when he is not doing well with regards to the proposed timeline and needs help. A lot of students will not actually ask for help out of fear or not wanting to annoy. The mentor needs to be proactive in those cases and regularly make sure the student gets the help he needs.

It is important to break down the project into manageable tasks and adjust goals where needed.

Code should be committed early, often and in a publicly accessible repository.

3.9 The Evaluations

Evaluations should be done based on several criteria. Did the student achieve his goal? Did the student make an effort? Is the result useful to the project?

The student should never be in doubt if he is going to fail or pass. In case a student is on the verge of failing there should have been a lot of measures taken and talks done with him to prevent him from failing beforehand. This needs to happen as soon as there are signs of trouble.

3.10 The After-GSoC Period

The figures in section 2.4 lead to two conclusions. The first one is that it is essential to give the students tasks for at least three months after the program ends. The second one is that everyone who is still with the organisation three months after the program ends is definitely worth investing more time into and can be expected to stay.

One of the easiest ways to keep the student motivated and engaged at this point is probably to get his code into a released version distributed to thousands of users.

3.11 Incentives

Clearly one of the big incentives to contribute to a project in GSoC is money but it is not the only one. There are other big motivators. One of them is the t-shirt students and mentors get. The other one of course is being part of a great team producing something of value for thousands of users. The publicity and feedback from the community and users are incredibly valuable here.

SoK shows nicely that money isn't the only motivation and likely not the biggest one. Someone believing in the student is often the biggest gift they can get. "Being allowed" to work with the project is not an unusual way of seeing it from the student's perspective.

3.12 Determination and Time

Being a GSoC admin for a large organisation takes a lot of time and determination. Getting ideas, applications, reviews and evaluations needs to be done and people reminded, often repeatedly. Most of it will happen fairly smoothly but there are always a few who need personal following up on. Those however are the ones who really need the attention to make sure no-one falls through. Keeping a written record of everyone's status is essential here to not lose overview.

The admin has to be responsive! Especially in the weeks around the application period there will be a lot of questions both from mentors and students. Well known points of contact (dedicated mailing list, IRC channel, ...) can help a lot here as can a FAQ.

3.13 Have Faith!

One of the things needed most in mentoring is faith. Believe in the mentor doing the right thing. Believe in the student being able to complete the task. It will work out just fine most of the time.

4. SUCCESS STORIES

KDE has had many very successful students in the past five years. The following are a selection showing what is possible if one is determined and motivated.

4.1 Akarsh

Akarsh worked on KStars as part of GSoC 2008. Quickly after that he became its maintainer and still continues to push it forward. He also mentored GSoC students in the following years.

4.2 Chani

Chani started her KDE contributions in the Kopete team and did a GSoC project there in 2006. After that she moved on to work with the Plasma team doing a GSoC project in 2008 and 2009 and becoming one of the core Plasma developers. In 2010 she is mentoring a SoK student.

4.3 Lukas

Lukas worked on Krita as part of GSoC 2008 and 2009. He is mentoring in 2010 and the Krita team had a very successful fund raiser to help him work on Krita full-time for three months.

4.4 Leo

Leo is the one who completed all three levels of GSoC. He started as a student for Amarok in 2007, became one of the main developers and went on to mentoring for Amarok and being one of the organisation admins for KDE.

4.5 Teo

Teo applied for GSoC in 2008 for Amarok but was unfortunately not selected. He took on an Amarok project as part of SoK which his university also counted as part of his degree. He became a part of the core-team and in 2009 and 2010 his GSoC applications were successful and he continues to do great work for Amarok.

4.6 Vera

Vera took part in SoK in 2009 working on Krita. She successfully completed her project and SoK helped her get an internship.

5. ACKNOWLEDGMENTS

I would like to thank Google for making GSoC possible, the KDE admins and mentors for endless hours of work and patience and last but not least each of our students for their work and enthusiasm. It is contagious.