Project Guidelines

Excerpts from the Chapter 4 of the book

i want 2 do project. tell me wat 2 do
by Shakthi Kannan
2014 / 135 pages / Paperback / Rs. 398 (at Amazon india) / Self Published

Adding manpower to a late software project makes it later.

~ Brook's Law

Write abstract, code, documentation and presentations

If you really want to do something, you'll find a
way. If you don't, you'll find an excuse.
~ Jim Rohn

A student approached me for help in creating an abstract for a project. After an initial discussion on his idea, I provided him a brief write up. Few weeks later, he pinged me online on Internet Relay Chat (IRC) for help in coding the project. I had worked out a proof of concept (POC) implementation for him. Later, the faculty wanted him to write some documentation for the project and he approached me again for help. I wrote a page explaining the idea and citing reasons for the design and implementation. After a month, the student again came back to me asking for a presentation! Mentors are there only to guide you in your project. They cannot write abstract, code, documentation and make presentations for you! Even if you are time bound, you need to learn to plan your work. It is very important to be clear on what you must do and what mentors can do for you. The effort has to come from you, and that is when you actually learn. Self-motivation is thus very important when working with free and open source software projects.

Small tasks before handling big tasks

Little drops of water make the mighty ocean.
~ Julia A. Fletcher Carney

Newbies with little practical experience can feel daunted by any task given to them. Handling complex problems or making an assessment can be hard. It is thus important to break a task into smaller sub-tasks and work on them individually. You can also request the project team members for beginner tasks to be worked upon stating that you are new to the project. This approach can get you started on a project. Working on smaller problems can also help you understand the tools used in the project and the workflow. It is also convenient for the team members to take time to review and correct smaller mistakes rather than having to go through a larger modification. Correcting smaller mistakes at an early stage is easier than to undo or redo a lot of changes before a project delivery. The scope of smaller tasks also helps you to focus and understand a module, its design and implementation without having to worry about the larger pieces of a project.

Last minute work

The early bird catcheth the worm.
~ John Ray

I always ask a question in my lecture sessions to know how many students study just a few days before an examination. Most of them raise their hands. Some of them ask for important questions that may appear in an examination, and study only those relevant topics. Some students study only to obtain a pass mark and hence feel that whatever is sufficient to study can be done just days before the examination. This approach doesn't work in real life though. To manage your work--life balance, you need to plan early and work systematically. Only this will help you manage both time and resources. If you work every day, sharing the workload over a period of time, you will reap the benefits before a deadline. You will also not feel the burden of it. Any last minute work done under pressure will not be a whole-hearted effort, and that will reflect on your work. It will be very obvious. The human body has a biological cycle and rhythm. If you allot time every day for your tasks, you will automatically fall into the cycle of doing work regularly. Study every day and do your homework daily. Avoid any last minute work!

Steep learning curve

The students should be reaching up to it because
success in life demands the use of intellect
under pressure.
~ Bill Cosby

The methodology, tools and approach in free and open source software can be entirely new or different from what you are used to. You may not understand everything when you are starting. You will feel frustrated at times. It is very important to be patient, and reason why things do not work the way you expect them to. When you learn to ride a bicycle, you will fall and hurt yourself. Only when you fall, you will learn through your mistakes. The learning curve is very steep. But, learning indeed is hard. It is thus important to persevere and understand if you wish to receive the benefits of free and open source software. If it becomes overwhelming, take a break and do some other work. You can return to the task with a fresh mind, and you may be able to see the problem from a newer perspective. Persist and survive. Everyone has gone through the same steep learning curve, and you are no different.

No blunt info

If you can't explain it simply, you don't
understand it well enough.
~ Albert Einstein

The free and open source software mailing lists are usually run by volunteers. If you have a problem in your work and need assistance, you must give as much information on the same, stating what you are trying to solve. If you are working on a specific feature request or bug, provide a link to the same. You should give a context to the problem, some background information on the task that you are working, any prior relevant communication with the project team members, the approach you have taken and any errors or output that you have observed. This can serve as documentation for you and others involved in the project. If people face a similar issue in future, a search in the mailing list will return these relevant discussions. Try to give detailed technical information when posting a question related to your work.

Journal or log of activities

The errors in our own history make us open to new
ideas, open to unusual ways of doing things.
~ Neil Lewis

Contributors to free and open source software projects may have a day job and they may be able to work on their projects only during weekends. Their job may require them to be online during weekdays, but that does not necessarily mean they are available for a discussion. Depending on the nature of work, they may or may not be able to answer your queries during weekdays. It is thus very important for you to keep a journal of your activities. You can create and update your journal as and when you complete your tasks. Even if you are stuck with a problem, you can write about it in the various communication channels and people might give you solutions. Mentors who are busy during weekdays can see your blog post when they have time and know about your progress. If there aren't any frequent updates, then it is a sign that there is a problem. Also, the blog acts as a tool to showcase your work. You will need to have accomplished some task before you can write something. Search and archives of the posts serve as documentation for you and others. An update must be provided at least once every week.

Never make your own decisions

We promise according to our hopes and perform
according to our fears.
~ François de la Rochefoucauld

When you are young and energetic, you have a lot of enthusiasm and interest. But, you have to be cautious about being over-confident. People with experience have immense knowledge that they can guide you well. It is very important to see different points of view before making a decision. Never commit to something without thoroughly analysing the scope and impact of your changes. Always try to seek a second or third opinion. If you have made a decision, you must be 100% sure that you can defend yourself no matter what questions are asked. Research well and do your homework. Some projects have maintainers for specific modules and they are responsible for it. You will need to get their approval before the project can accept your changes. It is thus very important to have a good rapport with your team members. The more you deviate from the goals of the project and make your own decisions, the more you will feel left out. Always consult others when you are new to a project. It takes years of experience to become a master and commander of a project.

Forget to CC members

Everybody knows everybody, and there is a very high
degree of transparency.
~ Yossi Vardi

Free and open source software development is a double-edged sword. The openness in the communication tools provides transparency for any dispute that may arise. It is thus very important to have all the project discussions in the respective project mailing lists. You can use private discussions only if you wish to discuss personal matters. If a project team is small and they prefer to use e-mails for communication, it is important to CC every one in the discussion. Forgetting to include

members in the discussions will cause confusion among the team members. It is very important for everyone to be on the same page. If your development module affects other projects or consumers of your software, it is important to inform them about any upcoming changes. Always keep everyone informed!

Repeat same mistakes

Go ahead and do what you think is right. If you
make a mistake, you will learn from it. Just
don't make the same mistake twice.
~ Akio Morita

Working with free and open source software does require you to be a self-starter. The methodology of work is not suitable for everyone. Some people like to be given tasks to work on, and they will follow that strictly. There is nothing wrong with that. But, you should know what your strengths and weakenesses are. Only by accepting your mistakes can you start to think of a solution for the same. You can write about them in your journal or blog so you remember not to repeat them in future. If you do keep committing the same mistakes repeatedly, then the task may not be suitable for your style of work. It is perfectly fine to move on to other tasks or even a different project after informing the respective project team members. It is very normal for people to change projects and domains if they feel there is a need to do so. Repeating the same mistakes is a concern for you and the project.

Can friends also join the project?

"Me and my friend studied in the same school. We
went to the same tuition centre. We joined the
same college and department. Can we work on the
same project?"

The friends circle vary in numbers in an institution. It is very common for a group of friends in a class to decide to work together on a project. They usually decide among themselves as to who will work on which part of the project. It might also be the case where only one person actually does the work, and the other team members just support the cause. The transparency in free and open source software projects doesn't encourage the above behaviour. Every project team member's contributions are visible. Each individual has his own clone of the project repository in a decentralized version control system. The individual commits made by them can be observed. The pull requests will let us know whose changes were absorbed, if the work is a contribution to a larger project. While it is fine with friends to work together on a project, it is important to learn to work with people outside your comfort zone. The free and open source software projects enable you to network and work with people from different backgrounds and countries. You need to make the best use of this opportunity. This experience will teach you how to build relationships with newer people in a project. It is acceptable for a group of friends to work together, but, make sure that everybody learns and contributes to the project.

Commit early, commit often

"Code, code, code your way,
Gently down the screen,
Commit early, commit often,
And life is but a dream."

One of the main principles of working in free and open source software development is to commit early and commit often. It is easier to correct mistakes at an early stage rather than having to complete the project and present it to users, only to realize that it is not what they had asked for. Project team members who are voluntarily contributing to the project will be able to take time to review smaller changes. If a project has test suites, then the small changes can trigger invocation of test runs and you can detect any failures immediately. When you return from vacation, it takes time for you to recollect what you have been working on the project. If you made commits often, then it will be easy for you to review the changes. If there are dependency modules in a project, then committing often and checking the test runs early can help detect any issues. The concerned users of the project will be able to review the changes and provide any feedback. This also helps avoid doing any last minute work. Being able to detect problems at an early stage also gives you an opportunity to rectify them, and helps you plan your time and project deliverables.

Never make assumptions

Question your own assumptions and the assumptions
of others.
~ Alan Kelly

It is very important to do your homework and provide justification wherever necessary. You should be able to give sufficient arguments in case of a dispute. When building a solution, you must review the steps to ensure that you have all the corner cases addressed. It is also important to be honest, and say "no" if you are not sure. If you decide to work on a

new feature request, first attempt a proof of concept (PoC) implementation. Research on the available approaches and do a feasibility study. Learning from others' experience on something which does not work can save you time and effort. In free and open source software development, you are encouraged to use open standards since the specifications are freely available. What works in one proprietary system, may or may not work in other systems. It is thus very important to run tests before you provide a solution. Proofs are not the same as tests though. Review your work to make sure you haven't made any assumptions. If you have, note them down as constraints for the specific implementation.

Never hesitate to ask questions

The scientist is not a person who gives the right
answers, he's one who asks the right questions.
~ Claude Lévi-Strauss

There is no question as a stupid question. The more questions you ask yourself, the more you will learn. But, free and open source software project team members may be working voluntarily in a project. It will be wise to do your homework before you ask questions. You can mention the various sources and links that you had referred to, and what you didn't understand in them. It gives an indication that you are serious about solving the problem and are eager to seek answers. You should avoid making direct requests in a mailing list asking for an immediate answer. The list members have no obligation to answer them. They may refer you to an appropriate source of information, or provided you useful links. It is also good to think in terms of the Five Ws and one H (Who, What, When, Where, Why, and How) when researching on a given problem. Always reason, and never hesitate to ask questions.

Priorities

The most important and basic thing is to learn and
understand the needs of a customer.
~ Vinod Gupta

Developers working on free and open source software projects work with a lot of passion and interest. They are more concerned about doing things right even if it takes time. A project may have release schedules where users may expect certain features. In such scenarios, it is necessary to note the priorities of the tasks. Measuring complexity is indeed very hard. Every individual will have a different opinion on the time it will take to complete a task. But, if it is a customer-facing project, understanding the priorities helps in planning your work. You can also assign priorities for sub-tasks that you work on, on a daily or weekly basis. Understanding the users, project requirements, and their importance can help set your priorities right.

GTD

A bitzui'ist is someone who just gets things done.
~ Dan Senor and Saul Singer

It is very important to do what it takes to get the work done. You should have, of course, done your research and home work before starting on your work. Planning and time management are crucial to keep track of your project work. Doing things in the last minute will reflect poorly. A well thought and planned work can be showcased to others as an example. The following are good books that I recommend on managing your time and work:

- Stephen R. Covey. "The 7 Habits of Highly Effective People." [1]
- Fergus O'Connell. "How to Get More Done: Seven Days to Achieving More." [2]
- Chad Fowler. "The Passionate Programmer." [3]
- Andrew Hunt, and David Thomas. "The Pragmatic Programmer: From Journeyman to Master." [4]
- Kevlin Henney. "97 Things Every Programmer Should Know." [5]

As much as you can do advocacy for a project, you should equally work on the project tasks and get things done.

References

1. Covey, Stephen R. 2000. *The 7 Habits of Highly Effective People*. Simon & Schuster.
2. O'Connel, Fergus. December 2007. *How to Get More Done: Seven Days to Achieving More*. Prentice Hall.
3. Fowler, Chad. 2009. *The Passionate Programmer*. Pragmatic Bookshelf.
4. Hunt, Andrew, and Thomas, David. 1999. *The Pragmatic Programmer: From Journeyman to Master*. Addison Wesley.
5. Henney, Kevlin. 2010. *97 Things Every Programmer Should Know: Collective Wisdom from the Experts*. O'Reilly Media.