Project Management

Kaya Oğuz

The purpose of this document is to discuss the day-to-day management of a software project that you will be developing for SE302 and CE316 courses.

A project requires resources; the most important resources are humans (in this case developers), time, and funding such as money (or grades in this case). The management of these resources is crucial for the outcome of the project; either success or failure.

Source Code

Developing software as a team requires a "version control system" that can keep track of the software source code while it is being developed. Currently "git" is the most popular version control system available.

Git, and other version control systems, work with a remote repository where the code is stored. Each developer gets a copy of the source code in the repository and works on their local copies. Once a task is complete, the developer sends (pushes) their code to the remote server so that others can get the latest version and get synchronized.

Since git is very popular, there are several documents, videos and tutorials available online. Instead of discussing what it is, I'll refer to these documents online.

The git, as a software, is available at https://git-scm.com - this web site has a lot of great documents and videos, available at https://git-scm.com/doc. While you need the git commands (software) to be able to use git, I suggest that you use one of the methods I will mention below, so do not install git using this website.

While you can have your own git server, we almost always use an online service, such as GitHub, GitLab, or BitBucket. Among these GitHub, perhaps, is the most popular one. I strongly suggest that you create an account on GitHub and download its client. This client will enable you to connect to GitHub and run git commands using a graphical user interface.

Your project must be developed on a git server. There are no exceptions. You can use any service you like (GitHub, GitLab, BitBucket, or anything else) but you should be able to make it public when needed. I can ask you to make it public so that I can access the code.

Issues

Issues are a general name for "bugs" - if there is a bug in the software, it must be reported. This is usually done by testers, but I also want you to enter all the bugs and errors you come across to the "issues" in the GitHub project page (or anything similar in the service you will be using).

There are different kinds of issues. Most of them are "run-time" errors. These happen as the program runs. The other ones are "compile-time" errors - these will be reported by your compiler.

The run-time errors are reported as follows so that the developer can reproduce the bug.

1. Steps to reproduce the bug (such as: enter this screen, enter this text to the text field, and click “process”).
2. What is expected?
3. What has happened?
4. Some visual or text output.
5. Severity or priority of the error.

You have to have a list of issues as you develop your project. I will check projects regularly, and I’m expecting to see errors. You cannot use “Whatsapp” or another tool where I cannot see your issues. They have to be available online.

Project Management

The project management can be done with tools such as Trello or Asana. These tools have tasks and assignments so that you can manage the project and see you progress. I can ask you to show me your management tool anytime during the semester. So **you have to use a project management tool as you develop your project.**

Task Point System

Task Point System (TPS) was an online tool to keep track of your contributions to the project. Every member must contribute almost equally to the project. TPS aims to maintain this. It basically works by assigning yourself a task and completing it. The more tasks you complete, the more points you get. However, these total points for each developer must be around the same; so this means (i) you should not do everything, and (ii) you should not do nothing.

Here is an example. Let a team have students A, B, and C, and let’s assume that each have completed the following tasks for the project. The Task column should contain detailed information. For the sake of simplicity of the table, I’m just going to list the tasks as 1, 2, and 3… The “Points” column is how hard the task is for the developer. It has to be between 0 and 100. 0 means very very very easy, and 100 means very very very hard. I am expecting most of the tasks to be around 50. Very high and very low values are suspicious, and I can ask you to explain why a task is assigned such a value.

Student	**Task**	**Points**
A	Task 1 (this should include detailed information)	50
A	Task 2	50
B	Task 3	40
C	Task 4	60
A	Task 5	60

In this case, A has completed 3 tasks and collected 160 points. B has collected 40 and C has collected 60 points. The team has completed 5 tasks and collected 260 points. Since there are 3 developers, the average is 260/3 = 86.6. This value is the expected points for each

developer. Since A has 160 points, his score is around 184, but since he cannot get over 100, it is cut off at 100. B gets around 46 and C gets around 70. A has overachieved and his work on most of his tasks has gone to waste. B and C did not do much, and they got lower grades. Instead each should have contributed around the same points, sharing all the work. If they have collected, say, 90, 80 and 100, their sum would be 270, expected score would be 270/3 = 90. Then, A would get 100 again, but doing much less work, B would get 80/90*100 = 89, and C would get 100, too, but his excess work would be minimal.

Another advantage of TPS is to give you the opportunity to make promises. The "tasks" are promises to your team. You are expected to assign yourself a task in the following manner:

1. You report a detailed description of the task to your team. Your team might ask for revisions. You should come to a final version of this description. For example, you can say that "In this task I will provide a database connection using a Database class. This class will have the following methods (i) one that will return a list of string objects for the users in the database; (ii) one that will accept a user name and a password and store it on the database; (iii) one that will return a boolean value depending on the user name and password parameters match a user in the database." Your team might ask for more information or ask for more methods.
2. Once the description is set, you should make a promise to finish it on a specific date. For example, you can say that "I will finish this on Friday, 13th." Your team must approve this date.
3. Then, you should decide on the difficulty of the task. As I have mentioned above, most of your tasks should be around 50. Your team must approve this value, too.
4. Once all are approved, you start working on it.
5. When you are done, you tell your team that you have finished the task (hopefully on time) and provide links so that they can check what you did.
6. Your team must evaluate your work and they can ask for revisions. For example, if your boolean method has some errors, they can ask you to complete it. You can only move on to the next task once they all accept it.

Think about this process from the other developers, too. If one of your teammates comes with a task, you are responsible for checking what they are doing. The project belongs to all of you.

Since we don't have the web application any more, you have to submit all your tasks on a spreadsheet along with your project. The sample file is available here: https://docs.google.com/spreadsheets/d/1hxvZRr2UBIkPNnZXKzrSPtBF7goD0iEY-99s8gQC1Qo/edit?usp=sharing

Notice that there are two sheets on the file; "Task List" and "Developer List" - the Task List must contain all your tasks as a team. The Developer List must contain the developer names and their total points. Notice that the columns "Score" and "Team Average" on the Developer List sheet are calculated using a formula.

You are expected to fill out this spreadsheet correctly. Do not do it at the end of the project. Use the project management tool to assign yourself a task and discuss it with your teammates. Then fill out this spreadsheet. You can use Google Sheets so that each developer can access. Please remember that when you submit this file as a team, I will assume that you all agree to these tasks.

A warning! A common approach in previous years was to do the task first, and then create the related information. This takes away the opportunity to learn how to estimate a due date for your tasks. Estimation is very important because in a real project your project leader will

ask for a completion date. If you cannot tell when you would be able to complete a task, then you can come up with shorter or longer durations. This is an opportunity to practice your estimation skills. Don't take my word for it, read Joel Spolsky's article here: https://www.joelonsoftware.com/2007/10/26/evidence-based-scheduling/

Here's another important part of TPS: you should know what others are doing, and others should know what you are doing. "X has done that part" or "Y was responsible for that" is not an excuse in a real project. In real life, the project belongs to everyone and everyone is responsible. Therefore, when someone completes a task, study it carefully. If they did it in a sloppy way, **ask them to correct it.** That last sentence is in bold letters: "ask them to correct it" - it is much harder to say this to your best friend.