CCT490, Fall 2011

Open Source Project Fact Sheet

Due Date and Submission Information

This assignment is worth 10% of your grade and must be done individually. It is should be submitted by **10:10 am on November 2**. There will be a penalty for lateness of **10% deducted per day.** Assignments not handed in one week (168 hours) from the time they are due will not be accepted. An assignment will be considered late if not submitted at the stated due time, that is by 10:10 am on the due date. "Per day" means "per 24 hour period."

Normally, students will be required to submit the assignment to TurnItIn for a review of textual similarity and detection of possible plagiarism. In doing so, students will allow their the assignment to be included as source documents in the Turnitin.com reference database, where they will be used solely for the purpose of detecting plagiarism. The terms that apply to the University's use of the Turnitin.com service are described on the Turnitin.com web site. Students who do not wish to use TurnItIn for submitting their work should approach the instructor **immediately** to discuss alternative ways to establish the originality of the paper. This discussion needs to happen before the student begins working on those assignments.

The assignment will need to be submitted as a PDF file.

The Task

Your task is to prepare a fact sheet / memo of **up to 2,000 words** (not counting the bibliography) summarizing information about the open source project that you chose earlier in class. You should cite all of your sources. If there are academic sources relevant to your project, then you probably should be drawing on them. You can draw on non-academic sources for topics that academic sources do not cover.

Your fact sheet should the information into sections described above. Depending on the nature of your project, you may find that there is more to be said for some sections and less for others. It is ok for the sections to be of somewhat different size, but they should be *roughly* balanced and you should provide information for each section. If you cannot find relevant information for some of the sections, please describe what you did to try to find the information.

You do not need to answer every single one of the questions listed under each section but they should give you an idea of what is expected for each section and are a good starting point.

1. Mission and Software

What kind of software does the project provide or aims to provide? Is there a "mission"? If so, what is it?

2. History

How did this project come to be? Describe the relevant context. Did the goals of the project change over time?

3. License

Under what license does the project distribute its software? What kind of rights does this license provide to the users? How did the project come to adopt this particular license? (Did it always use the same license?)

4. Contributors

Who contributes code to the project? Why do they do this? Are they volunteers or are they paid for their work? Who contributes money? Why do they do it? Are there any for-profit entities contributing money to the project? If so, how do they make their money?

5. Governance

How are technical decisions made? How are financial decisions made?

If the project is supported by a foundation (or association), then here are some questions to answer about it: When and where was the foundation incorporated? What is its official mission? Are donations to the foundation tax-deductible? Is the foundation a "registered US 501(c)(3)"? Who makes decisions on behalf of the foundation? Are those people elected? If so, by whom? Does the foundation hold trademarks for any of the projects that it supports? Does the foundation employ any people? If so, what do they do? If not, who does the work?

If the project is *not* supported by a foundation, why does it not need one?

6. Communication Strategy

What kind of tools does the project employ to support communication between the relevant parties? Is there a website? If so, how is it used? Are there mailing lists? If so, what kinds? What software is used to manage the mailing list(s)? Is that software itself free / open source? Where are the archives for the mailing list? How many messages (roughly) are sent to the list in a typical month? What kind of messages are most common? (E.g., is it mostly question-and-answer messages? Is it mostly announcements?) Does the project use any other tools?

7. Code Management

How are users supposed to get the software? Does the project offer *compiled* software for download (e.g., .exe)? If so, for what operating systems? Do the installation methods differ substantially between the different operating systems or are they all roughly the same? Does the project offer a way to download source code as an archive file (e.g., .zip, .tar, .tgz)? If so, where? How often are such archive files released? How would one get the source code with the history of revisions? What revision control software would one need for that? How often are revisions made?

8. Bug Tracking

Does the project provide the users with instructions on how to report bugs? How are they asked to do this? Do they actually use this method? If not, how *do* they report bugs?

9. Preliminary Analysis

Is there anything about this project that seems particularly interesting and worth writing a course paper about?

Formatting

Please make sure your assignment satisfies the following basic requirements:

- saved as PDF,
- showing your name in the top right corner,
- using 12 pt serif font (like Times New Roman),
- using 2 cm margins,
- showing the number of words used under the title (don't count the words in the bibliography)**.**

Assignments that fail to satisfy those requirements may be returned. The students will be asked to resubmit the assignment and will incur late penalty.

Please state the number of words at the beginning of your assignment, right after the title. Please note that 2,000 is a short word limit for the amount of information that you'll need to summarize. Therefore, you should aim to be succinct and to the point, using your words wisely. Including the most important information and omit the less important. You should plan to writing more than 2,000 words at first and then to condense the text.

When answering the questions, make sure to cite *all sources* on which you are basing your answer. Use **APA style** and include page numbers. (E.g. "Raymond, 1998, p. 2.") For readings included in the course pack with unnumbered pages, assume that the first page included in the course-pack is p. 1, then count from there. Include an APA-style bibliography section at the end of your assignment.

Avoiding Plagiarism

Any sequence of two or more words copied from any source (readings, slides, other students notes, etc.) and not originally written by you individually **must be included in quotation marks**. Failure to use quotation marks will be considered an academic offense and will be handled accordingly. *This rule applies whether or not the source is identified.* (Failure to identify the source makes the offense yet more serious, but is not a necessary component of plagiarism.) Here are some examples of proper and improper use of sources:

> According to Raymond (1997, p. 5), with a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone. **[Plagiarism: verbatim copy without quotation marks.]**
>
> According to Raymond (1997, p. 5), if you have enough beta-testers and co-developers nearly all problems will be characterized quickly. The fix will obvious to some person. [**Plagiarism: the text shows only trivial differences from the original.]**
>
> According to Raymond (1997, p. 5), with enough beta-testers and co-developers it becomes possible to quickly characterize the problems: for some person the fix will be obvious. [**Border line: the text shows differences from the original, but it is fairly obvious that the author started with a direct quotation and then made some changes just so that it is not a verbatim copy. Multiple sentences of this kind can constitute plagiarism.]**
>
> According to Raymond (1997), "given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone" (p. 5). **[Not plagiarism: the author uses quotes to indicate that those are Raymond's words. However, this answer does not make it clear the author understands what Raymond actually means. The author may get less than full credit.]**
>
> Raymond (1997) argues that having a lot of people testing the code and contributing as "co-developers" allows all problems to be found, understood, and fixed quickly. **[Good]**