CSE504 Homework 1

Haoran Cai [1]

[1] Department of Statistics
University of Washington, Seattle, WA

January 11, 2016

Write and update code documentation

Problem overview

Writing code documentation is important for both software developers and the success of softwares. The document here refers to source code documentation, API documentation as well as algorithm documentation. For developers, it can help original author organize thoughts and contributors work more efficiently. For the software itself, clear and up to date documentation can greatly attract people to use and contribute. However, writing documentation for the software is not the most interesting thing that software developers like to do. Because it takes long time to write in a quality that other people can easily understand. Furthermore, a more challenging question is when the code itself keeps getting updated, the documentation may not. As a result, the software may either lack detailed human understandable documentation or the documentation is out of date. We can even see these issues in very popular open source software projects. When the number of software contributors getting large or the development cycle getting longer, it is very hard to efficiently control the quality of the document and force the documentation up to date.

Force developers write documentation

To encourage software developers writing high quality documentation, I propose to use or develop a documentation style checking tool every time when the code gets committed into the code base. Because almost every software development requires version control (git, SVN), so this necessary extra checking can make sure only the high quality code can be committed. This documentation checking will for example check if there is enough documentation in each function, class. Furthermore, it will evaluate the documentation style. To lower the burden of the developers, this checking has many levels and the code will be automatically assigned an importance level based on the structure of the whole software, higher level indicates that this part of code needs to write more detailed documentation.

Make documentation up to date

To make the large documentation up to date, I propose two ways to solve this issue. One way currently widely used is to make a better visualization of the documentation. For example, Sphinx will automatically extract the code documentation from source code and allow users to convert those information into different formats that easier for developers and other people to view. For example, a popular choice is to convert the documentation into html format, people can read and track all the documentation in a browser, thus it is easier for them to understand the whole structure of the code base. Visualization can definitely remind developers to update required and useful documentation, however a tricker problem is that when code base gets larger and larger, developers may even forget what part of documentation needs to be updated. This frustration may even make developers and contributors not willing to write high quality documentation. Combining the tool I mentioned in last section, I purpose to construct a "documentation network" that track the dependency of each documentation. When a high quality documentation passes the test and gets successfully committed into code base, we either create a new node to represent this documentation or update the node if this documentation has been previously committed. The edges in this graph obviously represents the dependency of documentations. For example, in the API documentation, if one class inherits another class, then these two chuck of documentation (nodes) may have an edge to connect them. When the code base gets growing, this network will also automatically expand. When developer wants to commit a code or documentation, the tool will automatically remind him what other parts of documentation needed to be updated by querying the "documentation network".