What is Good Documentation

The IDEAS Scientific Software Productivity Project

ideas-productivity.org/resources/howtos/

Motivation: Software documentation is essential for three reasons:

(1) To identify the purpose and role of the software and its requirements

(2) To clarify what each component does, what is needed in order to maintain it, and how (or whether) it can be reused elsewhere

(3) To provide user support and thus minimize unnecessary handholding of users

Scientific software has an additional reason: to ensure that the software is used within its region of validity so that the possibility of producing spurious scientific results is minimized.

Categories: In the wider software world a good place to start for a general description of documentation practices is [1]. For scientific software many categories for documentation are listed below.Categories: In the wider software world a good place to start for a general description of documentation practices is [1]. For scientific software many categories for documentation are listed below.

- **Requirements:** The objectives of the software in terms of target applications and expectations from it in terms of features such as extensibility and composability.
- **Models and Algorithms:** Mathematical equations being solved and the numerical algorithms being used to solve them. These should include the range of validity for the models and algorithms.
- **Design Documents:** Design choices in terms of software architecture, data structures, parallelization techniques, and other similar relevant details.
- **Users Guide:** Specification of how to use the software, how to provide input, how to configure a specific problem to solve, and, if applicable, how to customize the software.
- **Reference Manual:** List of the interfaces and/or routines and explanation of their functionality. This can be automatically generated if the information is embedded in the code.
- **Embedded Documentation:** In nonmathematical software one may be able to do away with this kind of documentation and instead write code that is self-explanatory. However, a mathematical model or even numerical algorithm does not have a straight map to implementation. Inline documentation is the way to provide this map and is therefore indispensable in scientific software.
- **Readme Files:** These descriptive files are typically included with the source code of the software. They are often used to describe the section of the code (usually within the directory) where they reside.
- **Developers Guide:** If the software is extensible, a developers guide is helpful for users to extend the software for their own purposes.
- **Installation Guide:** If the software has any component or dependencies that need installation prior to use, they should be described explicitly in some document. It can be a simple text file or a more complete guide, depending on the complexity of installation.

This material is based upon work supported by the U.S. Department of Energy Office of Science, Advanced Scientific Computing Research and Biological and Environmental Research programs.

Version 0.2, April 25, 2016

- **Process Documentation:** For any nontrivial software effort with even a few developers, documenting the software process is useful. In large efforts with changing and/or transient developers, such documentation is necessary. This kind of documentation may include definitions and practices such as licensing, code contribution, verification, coding standards, and code reviews.
- **Tutorials:** These consist of the entire necessary input to set up, run, and (as appropriate) visualize the output of software. They may include source code (for a library) and/or input files (for an application code). Often tutorials are the most useful documentation for users and can double as regression tests.

Trade-offs: Documentation is the least immediately productive aspect of code development. In theory a code can be developed and perform perfectly without any documentation. In reality, for any nontrivial code, remembering all the details is impossible. Lack of documentation will eventually lead to the code's not meeting its design and performance goals. Arguably, however, generating documentation involves an overhead. Therefore, each team should decide how much documentation is optimal for them. For any scientific software, embedded documentation (comments inlined in the code) is necessary: developers can, over time, forget why they wrote what they did. But for all others the teams should weigh their options. For example, a small team with only internal users may opt to spend all their documentation efforts in specifications of models and algorithms, whereas the developers of a community code may want to spend greater effort in producing a users guide. Many teams use code annotations to automatically generate a users guide by doing *literate programming* [2].

Pitfalls: Code projects that have released code without adequate documentation have faced many challenges.

(1) Users tried the software and then dropped it from frustration.

(2) The software was too useful to drop, so the developers ended up fielding innumerable queries from users.

(3) Users did not understand the limitations of the software and obtained wrong results, which ended up casting doubt on the quality of software.

(4) Documentation was not maintained and hence became incompatible with the actual software over time, thereby misleading users.

(5) Better, more easily usable software came along, and users switched to the other software.

Many of these challenges can lead to software becoming obsolete, and as a result the developers may even lose their funding.

References

[1] *https://en.wikipedia.org/wiki/Software_documentation*

[2] *http://www-cs-faculty.stanford.edu/~uno/lp.html*

This document was prepared by Anshu Dubey with key contributions from Roscoe A. Bartlett, Ulrike Yang, and Ethan Coon.

This material is based upon work supported by the U.S. Department of Energy Office of Science, Advanced Scientific Computing Research and Biological and Environmental Research programs.