Documenting software

Hans Rischel
Department of Information Technology,
Technical University of Denmark

High-quality documentation is vital for any software product, and preparing software documentation is hence an important task for a software engineer. The software engineering student should therefore invest great effort in trying to learn how to make good software documentation.

Documenting software must be learned by practice where the student prepares documentation and gets feed-back from the teacher – it can not be learned by reading a text book. These notes contain a collection of guidelines on how to design documentation together with some criteria to be used in judging the quality of documentation. These criteria have shown to be quite useful – at least in defining a benchmark to be used in the (often quite heated) debate between student and teacher on the merits of documentation created by the student.

The goal

The goal of any software documentation can be paraphrased in two points:

- The documentation should be useful for the user, i.e. when reading or glancing through the documentation the user should
  - get the information needed
  - don't get information which is not needed
- The documentation should be easy to update to a new version of the software product.

Note that these goals are conflicting, e.g. repeating the same information at several places in the documentation may be useful for the user – but it is a nuisance when the documentation should be updated. Hence, one will have to find a compromise between these two goals.

Many kinds of users

There will always be many kinds of users of a software product. For instance, an office system used for preparing letters in a company has (at least) the following kinds of users: (a) the secretary using the system to prepare company mail, (b) the software specialist making the company specific adaption of the product (company logo on letters, etc.), (c) suppliers local software specialist adapting the product to the national language and the national characters, and (d) suppliers software developer maintaining the source code of the product, etc.

Different kinds of users have different requirements to the documentation as they differ in their use of the product and often also in their general knowledge of software. When documenting software you must hence be able to make descriptions on different levels of the same product, and on each level you must be able to write on the premises of a particular kind of users.

User demands

Software documentation is used in two distinct ways:

- *Tutorial reading* where the user wants to get a general overview of the software.
- *Reference use* where the user searches specific information about the software.

In either case the user should have the greatest possible benefit with the least effort when using the documentation. The tutorial parts should hence be easy to read and they should provide a general understanding while presenting a minimum of boring technical details. The reference parts, on the other hand, should provide fast and convenient means to locate and find a specific piece of information with all relevant technical details.

Terminology

The *terminology* is the set of terms used to denote concepts pertaining to the software product together with their precise meaning when used in this documentation. Graphical symbols used in figures are also part of the terminology.

A software product should, whenever possible, borrow the terminology of the *application domain* (e.g. the documentation of a text processing system should use the terms "paragraphs" and "sections" if these terms are commonly used in describing the composition of a letter). However, the terms from the application domain have to be used in a more stereotype and specific way as the terminology is now used to document the communication with a program (while the terminology of the application domain is used in communication among people) . Often it is also necessary to express finer distinctions with more concepts, so one will have to invent new terms – or to use terms which are otherwise considered synonymous with different meanings.

The basic rules for terminology in software documentation are:

- Always use the same term (or symbol) to denote the same concept.
- Use different terms (or symbols) to denote different concepts.

Note that these rules are in conflict with a stylistic rule requesting the use of a varied language to create a fluent text. This stylistic rule does *not* apply to the *terminology* of software documentation where consistent use of terms (and symbols) is much more important than stylistic qualities of the language.

Structure

The *structure* is the division of the documentation into sections with title and intended contents of each section. The structure provides the means for the user to access a selected part of the documentation (via the table of contents). There can be sections at several levels (e.g. chapters, sections, subsections, etc.).

The following rules should guide the design of the document structure:

- Each section should address a specific subject and different sections should address different subjects.
- Each section (on the lowest section level) should be written to address a specific user demand.
- Repetitions should be avoided when possible (in order to ease the maintenance of the documentation).
- Forward references should be avoided when possible (to ease a linear reading through the documentation from beginning to end).
- Parallel sections (at the same section level) should have comparable contents (a book on zoology should e.g. not have two *parallel* sections entitled “animals” and “monkeys”).

Language and style

The following language rules should be used when writing software documentation:

- Use short sentences – avoid long convoluted sentences.
- Avoid chattering "small-talk" – and remove sentences which are not essential for conveying the message of the present section.
- Use figures when appropriate.
- The use of figures must be consistent with the written text.
- Use examples instead of verbose explanations in tutorial material.
- Use tables in the text to arrange referential material in a systematic way.

Literary skills are rather useful when documenting software, but don't get carried away – use a short and concise language – you are *not* writing a novel.

Typography

Most companies have a *corporate standard* for setting up documentation, and you will, of course, comply to this standard when working in a company.

At DTU we stick to common typographical rules for technical reports:

- Write on A4 sheets.
- Put title of report, title of course, date, and your name and signature on the front page.
- The margins should have room for holes used to put the report into a binder.
- Use an easy readable set of characters – the type font should be at least 11 pt.
- Use page numbering.
- Chapters, sections and subsections should be numbered.
- Use page footing and/or headings.
- Larger reports should have table of contents and index.

Source programs

Programs should be short and elegant. Use of sub-functions may ease the readability.

Comments should only give informations which can not be read directly from the program like:

- The intention of specific parts of the program.
- Invariants of a data representations.

Comments should be brief and they should not clutter the program. The program must be arranged in a way to make it easy for a reader to locate and find any specific part of the program.

Proposed structure of a report

The following table of contents will do for most technical reports documenting a program. The "*Table of contents*" and "*Index*" are usually omitted in shorter reports. Also, the sections "*The problem*" and "*User's guide*" can be merged into one section called *User's guide*. This section should then contain a (tutorial style) prelude explaining the problem to be solved followed by a (referential style) user's guide for this specific solution of the problem.

	Style	**Comment**
Front page	referential	Title of report and course, date, name, signature.
Table of contents	referential	Section titles with page numbers.
Introduction	tutorial	The aim of the report, required prerequisites of the reader, short summary of the report.
The problem	tutorial	A description of the problem solved by the program.
User's guide	referential	A guide on how to use the program.
Examples	tutorial	Examples of use of the program.
Technical documentation	referential	Program listing, tests, plus internal documentation needed to modify the program.
Conclusion	tutorial	Assessment of the state of the solution and what could be done to improve it.
Index	referential	Alphabetic list of terms with page references.