Short Guidelines for Software Documentation
Version 6*

Sabih H. Gerez
University of Twente, Department of Electrical Engineering
Laboratory for Network Theory
email: s.h.gerez@el.utwente.nl

September 21, 1998
Document no: grz98-20

This short document is primarily addressed to students that work on projects under my supervision, but may be interesting for others as well.

Specifications expressed in formal languages (e.g. programming languages or hardware description languages) are not always easy to understand. For this reason additional documentation is often essential. In organizations whose main activity is writing software, often standards exist according to which documentation should be written. In this text, a rather informal set of rules for writing such documentation is presented.

Often code is distributed throughout different files. Make this distribution in a structured way. Use 'header files' for the definition of data structures, global variables etc. Put functions that are related in the same file. The documentation of which files exist and why (the "global organization") can be part of your report (use an appendix!).

Apart from the global issues that you explain in the report, the code should be documented by means of comments. Comments can be divided into several groups:

1. *File header.* The file header contains: a line that shortly describes the file, the file name, the author, the date of creation and the dates of major modifications. Besides you can put here more detailed information on the whole file. Below an example of a file header is given[1]:

```
;---------------------------------------------------------------------
; FUNCTIONS FOR POSTSCRIPT INTERFACE
;
; File: @(#)pscript.lisp
; Version: @(#)11.3
; Last update: @(#)3/31/89
; Creation date: March 18, 1988
; Author: Sabih Gerez, University of Twente
; Original Version: Pieter Vorenkamp
;---------------------------------------------------------------------
```

*This document can be obtained through World Wide Web via the author's home page located at: http://utelnt.el.utwente.nl/links/gerez/.

[1]All examples below have been taken from files of Lisp code; similar things can be done in any language, provided that the language allows the use of comments.

2. *Functions.* Before giving the code of the function, give the function name followed by a one-sentence explanation of its main goal. Then list the arguments with a short description, the return values of the function and the main steps in the function body. The more tricky your code is, the more comment you need to explain it. Also local variables might need a short explanation. Sometimes it is wise to add some comment locally within the function body, to explain something special. Example:

```
;---------------------------------------------------------------------
; ps-hor-axis: draw a horizontal axis, between two points, indicating the
;   units and emphasizing every five units.
;
; args: - stream-id: the open stream to which PostScript code is written;
;       - x1, x2: the minimal and maximal x-positions between which the
;         axis is drawn;
;       - y: the height of the axis.
;
; res: <the return value is irrelevant>
;
; effs: - a horizontal line is drawn;
;       - in a loop, perpendicular segments of the axis are drawn at
;         integer positions, the segments are longer when the position
;         is a multiple of five
;---------------------------------------------------------------------

(defun ps-hor-axis (stream-id x1 x2 y)
  (declare (float x1 x2 y))
  (ps-hor-line stream-id x1 x2 y)
  (dotimes (i (1+ (- x2 x1)))
   (declare (fixnum i))
   (if (= 0 (mod i 5))
       (ps-ver-line stream-id (+ i x1) (- y 0.5) (+ y 0.5))
       (ps-ver-line stream-id (+ i x1) (- y 0.25) (+ y 0.25)))))
```

3. *Data structures.* Describe the data structure as a whole in one sentence and then list each field of the structure with a short description. Example:

```
;---------------------------------------------------------------------
; material: contains all specific information about a certain type
;   of material. For the time being this information solely
;   consists of information for graphics output.
;
; - color: the color for GPR output on the screen;
;
; - ps-priority: the plotting priority for PostScript output;
; - ps-half-width: half the width of the wire for PostScript output;
; - ps-gray: the gray shade in which the material should be plotted for
;   PostScript output.
;---------------------------------------------------------------------

(defstruct material
  (color 0 :type fixnum)
  (ps-priority 0 :type fixnum)
  (ps-half-width 0.0 :type float)
  (ps-gray 0.0 :type float))
```

4. *Global variables.* Provide a short description of the global variable before introducing it. Example:

```
;---------------------------------------------------------------------
; - *priority-array*: an array that has an entry for each type of primitive
;   to be plotted; each entry contains a list of rectangles;
;   the rectangles in the first entry should be plotted first, followed by
;   the rectangles in the second entry, etc.
;---------------------------------------------------------------------

(defvar *priority-array*)
```

It is strongly advised to introduce the documentation in your files at the same time that you write the code itself. If you postpone it for later, you might forget essential details. Besides, there is never time for working on software documentation when you are about to finish the project. A necessary but not sufficient condition for your software to be well-documented is that 30 to 60 % of your files should consist of comment lines.