**Template for Software Design and   
Implementation Documents**

***Timothy A. Gonsalves  
Professor, Department of Computer Science & Engineering***

***TeNeT Group  
Indian Institute of Technology, Madras-600 036***

**E-mail: tag@ooty.tenet.res.in**

***November 2000***

**Table of Contents**

[**Introduction**](https://courses.iitm.ac.in/pluginfile.php/51014/mod_resource/content/1/Soft_Dsn.html?embed=1#Intro)

Formatting Conventions

Title

**1. Preamble**

1.1 Project Overview

1.2 Software Overview

1.3 Purpose

1.4 Requirements

1.5 Document Outline

**2 Interface Specifications**

2.1 Classes

2.1.1 Hierarchy

2.1.2 ClassA

2.2 Functions

2.2.1 aFunction

**3 Design**

3.1 Modular Decomposition

3.2 Data Structures

3.3 Algorithms

**4 Implementation**

4.1 Files

4.2 Formats

4.3 Details

**5 Dependencies**

**Appendix A Files**

**Appendix B Global Variables**

**Appendix C Classes**

**Appendix D Functions**

**Appendix E Call Tree**

**Appendix F Called-by Tree**

**References**

**Template for Software Design and Implementation Documents**

**Introduction**

This generic template is intended as a guide for preparation of software design and implementation documents. The team leader may modify the template for specific documents to improve the clarity of presentation, or to cater to peculiarities of a module. However, all the information specified below should be contained in the document. If certain information is present in another document (e.g. Interface specification or protocol standard) you need only cite that document.

As you write this document, keep in mind its purpose: a programmer who needs to make changes or fix bugs in the module in your absence must get a complete understanding of your software by reading the document and code. Hence, take care to explain your decisions especially the non-obvious ones. In addition to justifying your design choices, state why other alternatives were not chosen. Think of the questions you would ask, and ensure that the answers can be easily found.

**Formatting Conventions**

All file names and user type-in to be in Courier font.  
All variable, function, classes, type and constant names in the text to be in *italics.*

**Title**

Follow the title-page format used in your project. Register the document name and number with your Software Librarian. The "Authorship" should include all team members who have contributed significantly to the software or the document in decreasing order of contribution.

**1 Preamble**

**1.1 Project Overview**

Start with a 1-paragraph description of the overall project, citing appropriate overview documents(s).

**1.2 Software Overview**

Give a 1-paragraph description of the overall software, citing appropriate overview document(s), showing the place of this module in the overall scheme and stating its important functions. Cite the functional specification documents for this module.

**1.3 Purpose**

State the purpose of this document and the intended audience.

**1.4 Requirements**

If your module requires specific hardware or software, give as much detail as possible.

**1.5 Document Outline**

Give a section-wise overview of the rest of the document.

**2 Interface Specifications**

Describe the programming interface that other modules use to obtain service from this module. Describe the philosophy of usage, if any.

**2.1 Classes**

If your module uses classes, then give details of the public classes used by other modules.

**2.1.1 Hierarchy**

Give a pictorial representation of the public classes.

**2.1.2 ClassA**

For each class (e.g. classA) give the following:

Name - with a 1-line description.

Syntax - the class statement.

Purpose - why does this class exist.

**2.2 Functions**

For each function, give details as below. The functions may be logically grouped, or listed alphabetically.

**2.2.1 *aFunction***

Name - with a one line description

Syntax - function prototype

Description - state what it does

Arguments - describe each argument

Returns - Specify the return values

Exceptions - list behaviour in case of various errors

Bugs - list known bugs, limitations and restrictions.

If the interface is message-based, give the complete format of the messages and describe the fields and purpose of each message in a manner similar to the above.

Each function or message name should be in subsection title so that it appears in the table of contents (2.1, 2.2, ...) If necessary, use subsections and sub-subsections to improve clarity.

**3 Design**

Outline the design philosophy - is it state-machine based, client-server, interrupt-driven, etc.

**3.1 Modular Decomposition**

Give the break-up of your software into logical modules, accompanied by a figure. Show control flow with broken lines, data flow with solid lines.

**3.2 Data Structures**

Describe all important data structures, especially global ones. Use pictorial representation for clarity. Explain why the particular data structures were chosen. Cite references to text-books if they are helpful.

**3.3 Algorithms**

Describe important algorithms and state machines (FSMs). Explain the reasons for choice of algorithms and their properties such as complexity. Give pseudo-code or flow-charts as appropriate. Cite references to text books. It is often important to document why other alternatives were not chosen.

**4 Implementation**

Specify the hardware and software platform(s) and specific versions of tools such as compilers.

**Note:** Use Courier font for file names and command names (e.g. prog.c, gcc), italics for variables and function names (e.g. *pktCnt, GetString(), cardStatus[]).* Add `()' after a function name and `[ ]' after an array name.

**4.1 Files**

Describe how the modules (in section 3.1) are divided into source and header files. Give the directory structure used for the source code, if any.

**4.2 Formats**

Give packet and message formats. Neat picture are often best, perhaps augmented with C or other type declarations.

**4.3 Details**

There may be several sections and sub-sections here (very implementation dependent!).

Give non-obvious aspects of implementation. Include code-fragments (well-commented), pseudo-code, local variable and data type declarations. If you had difficulty in working out some detail, it is worth documenting it to save others from having to go through the same suffering.

Describe special debugging and test methods used, and hooks put into the code for this purpose. (This is distinct from the test and validation documents that are for use by system-level QA people).

**5 Dependencies**

Give a list of the other software or hardware modules/subsystems used by your software (typically, the layer beneath yours). For each, give the specific functions, services and features that you rely on. Cite appropriate functional specification documents and version numbers.

**Appendix A Files**

A list of all files that are part of your software, including, source, batch files, make files and data files, with a 1-line description of each. Arrange them alphabetically, or by category and alphabetically within each category.

**Appendix B Global Variables**

A list of all global variables. For each variable, give its type, filename and 1-line description. Arrange alphabetically and/or by category.

**Appendix C Classes**

Give the class hierarchy pictorially. Follow this by a list of classes. For each, give its prototype, filename and a 1-line description. Arrange alphabetically.

**Appendix D Functions**

A list of all functions. For each function, give its prototype, filename and a 1-line description. Arrange alphabetically and/or by category.

**Appendix E Call Tree**

A tree showing which functions are called by each function.

**Appendix F Called-by Tree**

A tree showing which functions call each function. This is useful if the interface to a function *F() i*s changed-we need to then modify all functions that call *F()*.

**Note:** For Java programs, the javadoc utility generates HTML version of many of the Appendices.

**References**

For each reference, give enough information so that the reader can locate it: author, title, name of journal, organisation/publisher, document volume and number date, page number. Reference to an online document should include the URL with the complete path and filename. The references should be either in alphabetical order, or in order of citation in the document. For example:

[1] T. Liao, " Multipoint Information Distribution Protocol ", unpublished paper, Nov 1998. http://webcanal.inria.fr/midp/midp.html.

[2] G. Ash, *Dynamic Routing in Telecommunications Networks*,  
McGraw-Hill, 1998.

[3] A. Manager, " Next Generation Access Centre Specs ",  
v: 1.3, A TeNeT Company (p) Ltd., 20 Aug 2001.