System Design Document Template
CMSC 421, Spring 2003

CMSC 421 Staff[1,2,3]

1 General Instructions

- Provide a table of contents.
- Provide a cover page that includes the phase, name, and the current date.
- Number the pages of the document.
- Number and label all figures. Refer to the figures by number in the text.
- All sections should have an introductory sentence or two.
- Do not use vague words and phrases such as may, might, could, possibly, should, assumed to be, some, a little, and a lot. Use strong, definite words and phrases such as shall, will, will not, can, and cannot.
- Watch your spelling, punctuation, and grammar. It is a reflection on your professionalism.
- As before, your document must be in either *Adobe Portable Document Format* (PDF) *PostScript* (PS) *ASCII text formatted to 72 columns* (txt) or *GIF / JPEG / TIFF* (for images or scanned-in handwritten stuff, not recommended)
- Provide definitions of all terms, acronyms and abbreviations needed for the SDD.

2 Be sure that your document is

- Complete - No information is missing
- Clear - Every sentence's meaning must be clear to all parties
- Consistent - The writing style and notation is consistent throughout the document and the document does not contradict itself.
- Verifiable - All facts stated are verifiable

[1]The template for this system design document (SDD) was adopted from the IEEE Software Engineering Standards Collection, IEEE Press and other SDD templates for CMSC 345 by Ms. Mitchell. It is further adopted for CMSC 421.
[2]Original Copyright : Copyright 1996-1998 Construx Software Builders, Inc.
[3]Please send all questions to cmsc421@cs.umbc.edu.

3 Your document shall have following major components

3.1 Introduction

Provide an overview of the SDD and a description of the scope of the software.

- Purpose - define the purpose of this SDD and specify intended readership.
- Scope - identify the software products to be produced by name; explain what the proposed software will do (and not do, if necessary); describe the relevant benefits, objectives and goals as precisely as possible; and be consistent with related higher-level specifications.
- References - provide a complete list of all the applicable and referenced documents. If you used any other types of documents to arrive at this design, list them here.
- Overview - describe what the rest of the SDD contains and explain how the SDD is organized.
- Constraints - Briefly describe any restrictions, limitations or constraints that impact the design or implementation

3.2 System Overview

Briefly introduce the system context and design, and discuss the background to the project.

3.3 System Architecture

- Architectural Design: A diagram showing the major subsystems and data repositories and their interconnections. Supplement with text as needed. Provide descriptions of the design components.
- Architectural Alternatives: Discuss other architectures that were considered.
- Design Rationale: Discuss the rationale for selecting the architecture, including the critical issues and trade/offs that were considered.

3.4 Data Design

- Global Data Structures: Describe any data structures that are a major part of this system. This should include major data structures that are passed between components. That is, it is not restricted to truly *global* data structures.
- Functional: List all functions and function parameters shown in the structural decomposition diagrams. For functions, give function input and output names in the description. Refer the reader to the decomposition diagrams.
- Database Description: Describe the database(s) which is/are part of the system.

3.5 Human Interface Design

Overview of the User Interface: Describe the general functionality of the system from the user's perspective.

3.6 A couple of checklists courtesy of Construx Software Builders, Inc. (modified slightly)

Architecture

- Is the overall program organization clear, including a good architectural overview and justification?
- Are modules well defined including their functionality and interfaces to other modules?
- Are all major data structures described and justified?
- Is the database organization and content specified?
- Are all key algorithms described and justified?
- Is the user interface modularized so that changes in it won't affect the rest of the program?
- Is a strategy for handling I/O described and justified?
- Is a coherent error-handling strategy included?
- Are error messages managed as a set to present a clean user interface?
- Is the architecture designed to accommodate likely changes?
- Are the major system goals clearly stated?
- Does the complete architecture hang together conceptually?
- Are motivations given for all major decisions?
- Are you, as a programmer who will implement the system, comfortable with the architecture?

High-Level Design

- Have you used round-trip design, selecting the best of several attempts rather than the first attempt?
- Are you satisfied with the way the program has been decomposed into modules ?
- Are you satisfied with the way that modules have been decomposed into routines?
- Are subprogram boundaries well defined?
- Are subprograms designed for minimal interaction with each other?
- Does the design differentiate between the problem-domain component, the user-interface component, the task-management component and the data-management component?
- Does the design keep the degree of component coupling as low as possible?
- Does the design keep the degree of component cohesion as high as possible?
- Will the program be easy to maintain?
- Does the design account for future extensions to the program?
- Are subprograms designed so that you can use them in other systems?
- Is the design lean? Are all of its parts strictly necessary?
- Does the design use standard techniques and avoid exotic, hard-to-understand elements?