Sample Software Design Document
2004

By Ofer Faigon
www.bitFormation.com

Sample High Level Design

Last modified: 18 Mar 2005
[See explanation]
[Download as a Word document]

...Table of contents...
[Any decent word processor will happily generate the table of contents automatically]

1. Introduction

1.1 Background

A well-written background should not cover more than a single page. As it is the opening section of your design document, the background must refer to the following questions:

- What the new system is
- The social and technological environment in which the system will function
- Its advantages over older systems
- Who the potential users are, and how they will benefit from it

Although some of these points were probably discussed at length elsewhere, their brief mention in the background is invaluable to the clarity and effectiveness of your document.

1.2 Design Goals

This section should outline the technical manifestation of the system requirements, and list the goals your design is intended to achieve.

2. Architecture

2.1 Introduction

Should contain details such as:

- The type of the system (distributed, client-server, etc.)
- What platform(s) the system will run on
- The major inputs and outputs
- What user interfaces the system will have and in what form (web, Windows GUI, etc.)
- The distances between components - on different PCs, on a LAN, on the web
- A rough estimate of the number of instances of each part (modules, threads, processes, clients, etc.)

A block diagram of the modules and the relationships between them can be very useful here. Try to point out the dynamic aspects even though this view is mostly static: include arrows to indicate flow of data and/or control, multiple boxes to indicate multiple instances of a thread or a module, etc.

2.2 Data

This section describes the persistent data, and/or any other data important enough to justify a separate section. Such a section will not be needed when there isn't a central database, or any other bulk of permanent storage.

2.2.1 Introduction

Should explain the need for a database, the considerations which led to the choice of a particular type of database; should contain a short description of the data stored in there, an estimate of the size and frequency of updates, some special considerations like security requirements, recovery, interfacing with external systems, report generation, etc.

2.2.2 Schema

Should give a list of tables and columns in each table, a description of each of the following - data type, size, number of records, what data it keeps, which parts of the software read it and why, which parts of the software write it and when, an estimate of the update frequency.

2.2.3 File and Data Formats

Most systems deal with external data stored in files, the majority of these notably configuration files and imported or exported data files. The files should be listed, as well as which module reads/writes them, at what instances and for what purpose. This section should give the name, or a detailed description, of the file formats.

2.3 Communication

Whenever modules communicate, be it using TCP/IP or some other protocol, this section should indicate the following:

- Which modules do so, at what instances and for what purpose.
- Who listens and who connects.
- Which protocol is used for communication. If relevant, give a detailed description of both the protocol and the format of the exchanged data. Furthermore, explain how the protocol may accommodate future changes.

2.4 Code

2.4.1 Introduction

Should mention any special considerations that led to this specific design, or that should be kept in mind while implementing it. It has to include general comments about the code in addition to notes that do not fit in any specific sub- section.

2.4.2 Modules

A general description of each module and where it fits in the global picture. There should be a description of what each module is responsible for, what inputs it takes, where it takes these from, what format they are in, and what method it uses to take these inputs (TCP, read from a file, a pipe, direct function call…) The same description should appear for outputs: what outputs each module produces, where these outputs go, in what format, using what method. This section should also specify the following: what other modules each module uses and for what purpose; what non-trivial algorithm each module uses; what non-trivial data structures it uses and for what purpose.

2.4.1.n Internal Functions

This section should be repeated for each module description - a comprehensive listing of the module non-trivial internal functions (functions that are not part of any interface). Be sure to refer to each item mentioned below:

- General description of the function and what it is used for.
- The name of the function.
- The return type.
- Ranges of return values and their meanings.
- Parameter names, types, whether the parameter is input, output or both and under what circumstances it is read or written.
- Assumptions on the parameter values.
- Assumptions on other conditions, such as global data or system state.
- Input validations that the function performs.
- Side effects of the function.
- Exceptions the function might throw and under what conditions.
- Non-trivial algorithms used.
- Non-trivial data structures used and for what purpose.
- Other non-trivial functions that the function calls.
- If the software has a layer structure, or some other inner partitioning, then to which part or layer this function belongs (this information should be evident from the naming convention).

2.4.3 Interfaces

Every module must expose one or more interfaces. If these interfaces are not trivial and clear from the module descriptions, they should be listed and described in an orderly fashion at this point. This description should detail the purpose of each interface and in what context it can be used. It should also mention what kind of modules or objects expose it and which ones will be using it. If using the interface requires obtaining a reference to it, there should be an explanation of how such a reference is to be obtained and what information will be needed in order to do so. Following the description of the interface, there should come a list of the interface functions, referring to the very same items mentioned above (see section 2.4.1.n Internal Functions).

It is advisable to add here a description of the scheme that will determine how the interfaces will evolve in future versions of the system in order to answer new requirements and modifications. If backward and forward compatibility is one of the requirements, explain how this scheme supports it.

3 Operation

This section should describe issues related to how the system is supposed to operate once it has been implemented.

3.1 User types

Should list the various types of users that will use the system (e.g., administrators, operators, managers, employees, customers...) Should also describe the user interface each user type would need to employ, and what prior knowledge and experience each type would require. Furthermore, this section should specify for each authenticated type/user which operations they may be allowed to perform, and which parts of the system, or data, would remain closed to them. Another important clarification should be the number of users the system may support and how many of them would be able to work concurrently.

3.2 Scenarios

Should describe a few typical scenarios of how the system works. For example, what happens in the system while a user logs in, gives a short sequence of typical commands and logs out - what data flows from which module to where, what triggers which actions, etc. If the system is not very simple, this description can make the difference between utterly confused and fully comprehending readers.

3.3 Installation

Yet another section that most designs ignore. It should explain how the system is installed, who can perform the installation (i.e., does it require some special skills, or can it be done by anyone?), what resources are needed for a successful installation, what medium would be used for the installation (e.g., a CD, download modules over the Internet, etc.), what user interface would be used during the installation (e.g., a Unix shell script, a Windows-like install UI, etc.), whether or not the installation would require a serial number. An important detail to include here is whether or not you allow several installations to co-exist on the same machine or LAN.

3.4 Licensing

Most commercial software is protected by a licensing scheme. If this is the case with your system, you should describe the scheme you are going to use, particularly the way a license will be validated (by some internal consistency check, or by an online query to a central server), at what times (during installation, at program start up, at regular intervals) and what should happen when validation fails.

3.5 Upgrades

This section should describe the way the system would be upgraded to newer versions. It should also relate to the following questions:

- How will these upgrades be distributed (Internet, CD)?
- How (if at all) will an appropriate license be distributed and verified?
- Who will be authorized to install the upgraded version (i.e., which of the system's user types mentioned in section 3.1)?
- How complicated will it be to reinstall the previous version in case an upgrade causes problems?

- How can a partial upgrade be installed? (e.g., in the case of a distributed system where it is unreasonable to expect the server and all the clients to be upgraded at the same time)
- How will the design handle version conflicts (between a server and a client, between two modules, between stored data and a module or between imported data and a module)?
- How will the design ensure that the users do not experience data loss or downtime during an upgrade?

If time-limited ("try before you buy") or demo versions are planned, they require special attention. You must consider all possible scenarios, such as upgrading from a demo to a full version, attempting to install a demo over a full version, the coexistence of a demo and a full version on the same machine, etc.

3.6 Uninstall

This section should describe the way the uninstall would be done, who would be authorized to perform it, what resources would be needed, what data would be left behind, and what would happen to that data when re-installing on a computer after an uninstall. Don't forget to consider the implications of your licensing scheme.

4 Development

[Some companies require that this chapter be included in the design; others expect this information to appear in a separate document.]

This section lays out a plan for the development process. It can be quite complex if the project has modules that cannot be developed or tested before others are completed, or if it depends on the availability of external resources like developers, machines, access to special services or real-world objects, etc.

For example, you may choose to start by developing a mock-up version of some module, in order to allow the development and testing of another. Once this is done, you can develop the two modules in parallel using two independent teams. Another common technique is to manually prepare a data file that will be used as the input to some module until the UI is mature enough to supply real data.

The design should describe the necessary resources - number of developers, their required skills, the hardware, environment and development tools required for the development process. It should also say when and for how long each resource would be needed. This is the place to include development time estimates.

5 Miscellanea / Appendices

5.1 Conformance with standards

Must contain the lists of both the standards the system should conform to and the references as to where these standards may be obtained.

5.2 Interoperability with other systems

Must contain the list of the external systems that the new system should interact with in addition to a description of the way in which it should do so.

5.3 Expandability

If applicable, this section should explain how a third party would be able to extend the system. This may be done by writing plug-ins or scripts, or by writing some instructions in a configuration file.

5.4 Debugging

Most of the development time will no doubt be spent on debugging. The larger the system, the more essential it becomes to have some built-in mechanism to help debug the complete system. It can be a mechanism of log files, or special functions in each interface, or a special mode in the UI. This section should describe your choice of mechanisms, the ways in which they would be used, and what the programmers should do in order to make their module part of the game.

5.5 Security

Most software systems today face one or more security threats: spoofing, identity theft, password stealing, eavesdropping, sniffing, spamming, data theft, web site defacing, denial of service attacks, password breaking, fraud, forgery, hacking, viruses, worms, trojans - just to name a few.

This section should describe the security threats you foresee and intend to deal with. It should specify your assumptions regarding the environment (whether the computer is behind a firewall, who has physical access to it, etc.) and the means you plan to employ in order to protect the system (authentication, data encryption, input validation, internal sanity checks), etc.

5.6 Open Issues

There are always some issues left open. Sometimes information needed for making certain design decisions is not available in time or even at all. Sometimes decisions are delayed for a more convenient time. This section should list all the open issues in the design, and, if possible, point out what is required in order to resolve each one.

5.7 Glossary

List all the technical terms, concepts and acronyms that appear in the document or that are relevant to it, for the sake of the uninformed reader. The explanation of each term/concept/acronym should not exceed 4 lines. People will thank you for not having to spend hours on looking up unfamiliar terminology.

5.8 Bibliography

If applicable, list documents, publications, books and other information sources that your readers may find useful when trying to understand the full implications of your design.