## Sample Format Of The Requirements Specification

Based on IEEE Standard 830: Software Requirement Specifications

This document is to contain all of the details the software developer needs to design the product.

### 1. Data Model:
In this section the data entities should be described and the relationships between the entities specified. One way this can be done is with an Entity-Relationship Diagram. Another way this can be done is with a Class Diagram. Choose one of the two ways to include in the document.

### 2. Functional Modeling and Information flow:
Specify the information flow and the transformations that are applied as data moves from input to output either using a set of Sequence Diagrams or a level 0 Data Flow Diagram.

Note for sections 1 and 2. You must be consistent in your choices; if you choose an ER Diagram for the data model you must use a level 0 Data Flow Diagram for your functional modeling, as you have chosen the classical paradigm. Obviously, if you choose a Class Diagram for your data model you must choose the Sequence Diagram set for your functional model, as you have chosen the object oriented paradigm.

### 3. Specific requirements:

#### 3.1 System Features
This section describes how the transformation of input to output is achieved. The following table of information is given for *each* individual function.

<table>
<thead>
<tr>
<th>Name of Function</th>
<th>Description:</th>
<th>State the purpose and objective of the function.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inputs:</td>
<td>Listing of all input items (using the naming conventions specified in the data model), any valid ranges know should also be specified.</td>
</tr>
<tr>
<td></td>
<td>Source of input:</td>
<td>How/where/who provides the input items (external entity or actor)</td>
</tr>
<tr>
<td></td>
<td>Outputs:</td>
<td>Listing of all output items, both data and control information</td>
</tr>
<tr>
<td></td>
<td>Destination of output:</td>
<td>How/where/who receives the output (external entity or actor)</td>
</tr>
<tr>
<td></td>
<td>Processing:</td>
<td>List of processing steps that must be followed to transform inputs to outputs. Includes error checking, calculations and validation.</td>
</tr>
<tr>
<td></td>
<td>Requirements:</td>
<td>Listing of other information (data or system) which this function will require that is not directly input. These include data items that must be retrieved from a data store in response to some input.</td>
</tr>
<tr>
<td></td>
<td>Pre condition:</td>
<td>What events/control/conditions must exist upon entry to the function.</td>
</tr>
<tr>
<td></td>
<td>Post condition:</td>
<td>What events/control/conditions take place as a result of the function.</td>
</tr>
<tr>
<td></td>
<td>Side effects:</td>
<td>Listing of any events, which might occur that are not directly related to output items.</td>
</tr>
</tbody>
</table>
Responses to Abnormal behavior: The action to be taken should invalid data be received or abnormal processing occur.

Stability: Stable/Volatile: Denotes the stability of the function. A process is “stable” when it is thought that the needs it addresses will not change during the expected life of the software. It is “volatile” if it is subject to change.

Necessity: Reflection of the “desirability” of the desired function. “Mandatory”: implies that the software will not be acceptable without the feature. “Desirable”: implies that the requirement would enhance the software but the product would be acceptable without the feature. “Optional”: implies a class of functions that may or may not be worthwhile.

3.2. Non-Functional Requirements:

3.2.1. Design Constraints:

3.2.1.1 Standard Compliance: Existing standards or regulations that must be followed and the requirements that result from these.

3.2.1.2 Hardware limitations: A description of the characteristics of the hardware, as far as they lead to software requirements.

3.2.2. External interface requirements:

3.2.2.1 User Interfaces: A description of the characteristics of the user interfaces, such as screen layout, function keys, help functions. For each report to be produced, a Warnier-Orr Diagram should be included.

3.2.2.2 Software Interfaces: A description of the software needed such as a certain operating system or software package. Interfaces to other application software are also discussed here.

3.2.2.3 Communication Interfaces: An example is a communication protocol.

3.2.3. Quality Attributes: In this section, particular attention is paid to quality aspects. These requirements must be measurable and verifiable.

3.2.3.1 Availability: Factors that guarantee a certain level of availability, such as restart procedures, redundancy, etc.

3.2.3.2 Security: Requirements regarding unauthorized access and other forms of misuse. These include cryptographic techniques, constraints on the access and communication.

3.2.3.3 Portability: Requirements regarding the portability of the software to different hardware platforms and/or different operating systems.

3.2.3.4 Maintainability: requirements to guarantee a certain level of maintainability of the system.

3.2.4 Performance Requirements: Performance requirements encompass both static and dynamic requirements. Static requirements concern, amongst others, the number of terminals to be connected and the number of users handled concurrently. Dynamic requirements concern operational performance of the system, how frequently will certain functions be called for and how fast should the system’s reaction be. These requirements should be stated in measurable terms.
The purpose of the acceptance test plan is to state the tests that must be performed for each function to demonstrate that the product meets its requirements. All features of all functions must be demonstrated as functional in order for the product to be accepted and these tests form the basis of that plan. For each function in your specification, individual test plans should be developed for testing correctness and robustness. They should include data which exhibit both normal functionality and error conditions.