SPECIFICATION DOCUMENT (D1)

Introduction: BFUD v Agile
The role of the specification document varies depending on the process. In a waterfall-style (AKA "big up-front design") process, the specification document can take months, or years, to write. It is complete, detailed, and, when finished, it is static. The challenge, of course, is to get it right.

Recognizing that 1) it is nearly impossible to gather all requirements up-front, 2) the requirements are likely to change as the development team continues to show prototypes to the client, and 3) the time and effort spent in creating a BFUD specification is expensive'; the agile way is to gather requirements throughout the development effort.

In an agile process the specification should be a living document. That is, in response to an evolving understanding of the problem, the specification should change. For this reason, the specification "document" should be as lightweight as possible. On the other hand, there should be a point at which the team has a solid enough understanding to cast some aspects of the requirements in stone.

What is a "specification document"?
A "specification document" (or just "specification") is not necessarily written as a single document – but it could be. The specification is the collection of all documentation the team has assembled to represent the problem (not the solution) as described by the client. This could be interview transcripts, field notes, work models, use cases, user stories, pictures, diagrams, text, …

To remain agile, the team should select the appropriate means for capturing all the specifications. That is, there could be different documents for different aspects of the specification, and there could be different files, documents, etc. The goal is to maintain, share, and use the specification throughout the development effort. In addition to the living (i.e., maintained, shared, and used) specification artifacts, the team will prepare a formal specification document (referred to as "D1").

D1: The Specification Document
The D1 document must have the sections below, in the order below, with the section numbers as below. After each element is a word, or words, in square brackets suggesting the presentation of that item. Note the list of qualities of a good technical document at the end of this document. Keep these in mind when writing the document.

1. **Elevator pitch** – "a story short enough to explain during an elevator ride (30 to 60 seconds), yet still complete and understandable". The elevator pitch appears first in the D1 document, but probably should be written last. [paragraph(s)]

2. **Functional requirements**
   2.1. Identify **user classes** – just a list [bullet list / table]
   2.2. Identify **tasks** by user class – title and short description of a function. Should be written as:
   `A person in <user class> can perform <task with system> for <reason>.`
   [bullet list / table]
   2.3. Group tasks into **functional groups** – a collection of functions that make sense to consider together because of their relationship within the architecture of the product. Each group should be assigned an identifier, preferably a mnemonic. [table]
   2.4. Describe the **test plan** for each functional group [table / paragraphs]

2.5. **Estimate effort** (development and testing) for each functional group in person-hours [table – three columns: function group id, development PH, testing PH]

3. **Non-functional requirements**

3.1. Security/data-privacy issues [table / paragraphs]

3.2. Data longevity issues [table / paragraphs]

4. **Non-executing products**

4.1. Identify **non-executing products** (e.g., external documentation) [paragraphs]

4.2. **Estimate effort** (development and testing) for each non-executing product in person-hours [table]

Using the D1

The D1 is used to communicate with the client. You will ultimately show the D1 to your client to:

- Validate your description of the product
- Communicate the initial estimate of the product size – they need to know if the product is too big.
- To prioritize features – what is most important
- Determine the minimum viable product (MVP) – work with the client to determine the minimal set of features that make a working product. This is your target for version 1. Everything else should be saved for version 2.

A good technical document:

- is concise
- supports random access (headers, bullet lists, tables, whitespace, images, …)
- uses good grammar
- has no technical errors / contradictions
- has version number / date
- has page numbers / footer / header