Backyard Splashpad Specifications Document

Braydan J. Allen
James S. Humble

October 24, 2018

Acknowledgements

No-one has helped us significantly enough to deserve to be mentioned in the Acknowledgments.

Signature Page

Signature	*Signature*	*Signature*
Date and email	*Date and email*	*Date and email*

Revision History

Revision	Description	Author	Date	Approval
1	Initial Revision	James and Braydan	Oct 24, 2018	
2				
3				
4				
5				
6				
7				
8				
9				
10				

Contents

1 SCOPE **7**

2 APPLICABLE DOCUMENTS **8**

3 STAKEHOLDER REQUIREMENTS **10**

3.1 Stakeholders User Stories 10

4 ENGINEERING REQUIREMENTS **11**

4.1 Interface 18

4.1.1 Interface Method Design Constraints 18

4.1.2 Interface Design Requirements 18

4.1.3 Interface Support Design Constraints 18

4.1.4 Interface Support Design Requirements 18

4.2 Functional Requirements 18

4.2.1 Functional Method Design Constraints 18

4.2.2 Functional Design Requirements 18

4.3 Support Requirements 18

4.3.1 Support Method Design Constraints 18

4.3.2 Support Design Requirements 18

5 VERIFICATION OF REQUIREMENTS **19**

5.1 Verify Coverage of Stakeholder Requirements 20

5.2 Interface 21

5.2.1 Functional Interface Constraints 21

5.2.2 Functional Interface Requirements 21

5.2.3 Support Interface Constraints 21

5.2.4 Support Interface Requirements 21

5.3 Functional Requirements 21

5.3.1 Functional Method Constraints 21

5.3.2 Functional Design Requirements 21

5.4 Support Requirements 21

5.4.1 Support Method Constraints 21

5.4.2 Support Requirements 21

Specifications

1 SCOPE

(a) **General: This document describes the design and verification requirements of the Backyard Splash Pad. The Backyard Splash Pad is used to provide backyard entertainment for adults and children alike.**

(b) **Acronyms:**

BSP: Backyard Splash Pad

2 APPLICABLE DOCUMENTS

This section is often poorly understood and poorly implemented. It is a list of documents that are a critical part of understanding the item or requirements imposed on the item. Every document listed must have a text reference in the body of the spec further describing and limiting how it is to be applied. Conversely, no document is to be referenced in the spec unless it is listed here. Don't put items here that are background information or of general interest. Always obtain and review all items listed here. This section often has the following statement:

> *The following documents shown shall form part of the specifications for this project. In the event of a conflict between requirements, priority shall first go to the contract, second to this document, and lastly to these reference documents.*

There are lots of MIL-STD(standards) and MIL-HDBK(handbooks) that cover an amazing range of subjects. Here is a website that has them plus NASA documents and others all available (every government specification and handbook you can imagine) for free. Other groups publishing standards include Institute of Electrical and Electronics Engineers (IEEE), American Society of Mechanical Engineers (ASME), Society of Automotive Engineers (SAE), American National Standards Institute (ANSI), American Society for Testing and Materials (ASTM) and Aeronautical Radio, Incorporated (ARINC) to name more than a few.

Another thing to note is that referencing documents will often save you time (and money). For example, a referenced document can contain a complete set of environmental tests. Stating that your system has to be tested to the requirements of MIL-STD xxx is a lot easier than making up the series of tests yourself. The process is akin to what you do when you use a library in C programming: someone has provided software that meets your needs. Why reinvent the wheel? Particularly in something as difficult to get right as environmental testing.

Another benefit to using standards documents is that it connects your project to what is typically done in industry. While this should be obvious it is sometimes forgotten. The designers that you hand the spec over to may already know the standard that you have specified and know how to meet these standards. The simple adherence to standard could save you a lot of money.

(a) **Government Documents** *This is where to put MIL-Specs, MIL-STDs, NASA specs and so forth. Be sure to include the revision level and date.*

(b) **Industry Documents** *This is where to put ANSI, ASTM, ASME, IEEE, Company specifications and so forth. Both this section and government documents can be divided up into logical subcategories.*

My project is pretty simple and I don't have any applicable documents, but your project might. Did you know that in order to use USB on a commercial device that you have to pay a licensing fee? I know, your development board has a USB port on it. You can use it because the board manufacturer paid the fee. The USB spec is about 1500 pages. Saying that your device must comply with this spec (or a part of it) is a lot easier than writing it all out.

Research and write down the external specifications that you want your system to meet. Think about standards (like USB) and MIL-Specs that cover things like environmental testing and electromagnetic compatibility.

3 STAKEHOLDER REQUIREMENTS

The stakeholders for the USU ECE Controls Lab Power Amplifier Module are:

1. Dr. Don Cripps
2. Jolynne Berrett
3. The Families of Braydan Allen and James Humble
4. The USU ECE Department

3.1 Stakeholders User Stories

The primary stakeholders needs are described below.

- **Dr. Don Cripps The device:**
    1. Must be complicated enough to challenge the designers.
- **Jolynne Berrett. The device:**
    1. Must have a useful, readable user manual.
- **The Families of Braydan Allen and James Humble will be the primary users of the device. The device:**
    1. Must not present safety/shock hazards to any user.
    2. Must have an easy user interface.
    3. Must be able to run in a typical backyard.
    4. Must have multi-colored lights.
- **The USU ECE Department is funding the project. The device:**
    1. Must be low in initial cost (design and prototype).
    2. Must meet the pedagogical requirements of the course (ECE 4820/4830/4840/4850) for which it is designed.

4 ENGINEERING REQUIREMENTS

This section constitutes the meat of the document. Remember that it is "requirements" and not methods. The definition here should leave no doubt about what is needed. Engineering requirements differ from stakeholder requirements in measurability, detail, and unambiguity.

Measurability is a key attribute; without it we have no way of knowing that we have built what we said that we would build. In the aerospace industry the process of determining that what we have built meets the plan to which we were building is called verification. Verification answers the question, "Did we build what we said we would build?"

Ideally, detail in engineering requirements means that the designer can set off designing and building the system with little or no extra input from the stakeholders. Since we do not live in an ideal world additional stakeholder input will be required. When additional input is sought the specification should be revised to reflect clarification or change. Reality is why we include a Revision History Block at the beginning of the specification.

An ambiguous requirement, by its very nature, means that I could design and build the system in one, two, or many 'wrong' ways. Since the stakeholder who is paying for this system doesn't want one of the many possible wrong ways we need to ensure that requirements cannot be misinterpreted by the designer.

Understand:

1. *"An engineering requirement is a statement about the system that is unambiguous. There's only one way it can be interpreted, and the idea is expressed clearly so all of the stakeholders understand it."*
2. *"An engineering requirement is binding. The customer is willing to pay for it, and will not accept the system without it."*
3. *"An engineering requirement is testable. It's possible to show that the system either does or does not comply with the statement." (Steve Tockey)*

Additionally, requirements must be stated in one place only in this document. You can reference the requirement in another part of the document by using its paragraph number. It is often tempting to restate a requirement in another requirement to improve clarity. However, a requirement stated in two places may only get changed in one place during a revision and possibly leading to confusion or worse a flawed product.

Engineering requirements constitute contacts between between clients and design teams. Requirements are binding.

Requirements form the base that we design on. A well written requirement keeps us from ending up with a less than ideal implementation. Defining requirements forces us to think prior to design.

The engineer's job at this point is to make the stakeholder requirements of the previous section into measurable, detailed, and unambiguous. Sometimes this process requires that a stakeholder be interviewed again until a proper set of engineering requirements emerge.

An old joke in aerospace says that a specification is written and thrown over a high wall to the design group. The spec. writer then runs away from the wall (laughing). While formal adoption of this absurd method leads to disaster, we should write specifications as if we could never clarify anything. The clearer we make the specification now, the lower the cost of change later.

Processing Stakeholder Requirements into Engineering Requirements

Stakeholder requirements in the form of user stories are typically poorly organized and indefinite. A designer cannot design to an indefinite requirement. How would a designer know if an indefinite requirement was met? Also, different stakeholders may have requirements that effect overlapping areas of the system but because of the nature of story collection the overlap in these requirements may not be obvious. There are many more issues and a systematic approach to this processing problem should help.

The work of sorting and firming up requirements can be considered tedious. Engineers are famous for hating tedious work. It is why we are constantly automating things that we don't want to do. And when we encounter a tedious task we have a tendency to create a process to limit the amount of tedium that we must endure. Tedious work reduced to a process becomes less tedious if the process is well designed. In order to translate the scattered and indefinite stakeholder requirement into organized and definite engineering requirements we are going to follow a two step process:

1. *Identify and sort stakeholder requirements into fixed categories.*
2. *Rewrite the sorted requirements into definitive engineering requirements.*

The details of the process are explained in the following sections.

Identifying and Sorting the Stakeholder Requirements

At this point in the process we are faced with the task of identifying and sorting the requirements found in the user stories. Finding the requirements is pretty easy. The user wants

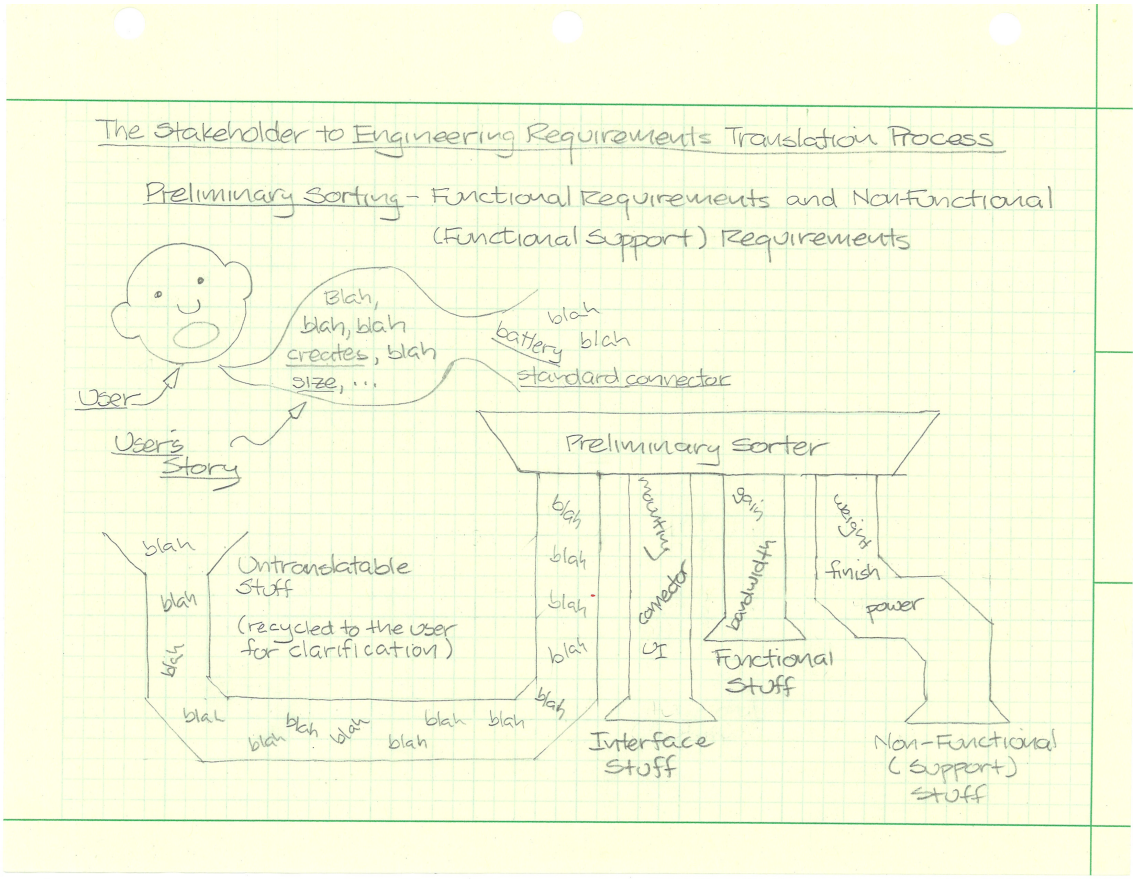

Figure 1: A machine that sorts user stories into appropriate categories.

us to do something and that something is a requirement. Sorting the requirements is only slightly more difficult because we need bins for the different categories of requirements. Different companies come up with different bins so I don't feel too bad at coming up with my own set of bins.

It helps me to think of the process of translating stakeholder user stories into engineering requirements as a machine with several stages. We take the often less than definite requirements that we find in user stories, clarify the untranslatable stuff, and sort the results into applicable categories. The first stage of the process is illustrated in Figure 1.

We will be sorting the requirements into four categories:

- *Untranslatable Stuff*
- *Interface Stuff*
- *Functional Stuff*
- *Non-Functional (Support) Stuff*

The categories themselves are intended to cover all aspects of the project in a systematic and an understandable fashion. Properly chosen categories allow the designer working from this specification to verbally see the system (don't worry, we will also draw a few actual pictures of the system). I'll explain the categories in the next sections.

Untranslatable Stuff

Let's face it we're human and our project stakeholders are human. Being human we don't always communicate what is either meaningful or useful. Sometimes, as engineers, when we hear or read a stakeholder statement it may seem vague or more like a wish. Sometimes the statement may be incredibly obvious, but hard to translate into a measurable requirement. When we encounter such statements it is up to us to approach the individual stakeholder and ask (politely) what on earth the statement means in terms of the project deliverables.

Interacting with our stakeholders early in a project is a good thing. We establish the lines of communication that we will most definitely need sometime in the project.

Interface Stuff

We design and build things to perform some useful or interesting function. Interesting things interact or interface with their environment to send information, provide motive power, receive and process information, receive power, fit into an existing slot, etc. Interfaces make up the useful pieces of a system and are critical to the system.

The documentation for many complex systems includes a very useful document called an Interface Control Document, or ICD for short. This document contains all the interface information for the system. It is a very useful document and helps designers avoid a host of errors. If such a document is available when the specification document is being written then it is simply referenced in the Applicable Documents Section of the specification. For our purposes we will define the interface requirements as part of this specification document.

Functional Stuff

The devices we make are intended to do something. The something that they are supposed to do is defined as functional constraints (designer's hands tied) and functional requirements (designer is free to choose the method).

Non-Functional (Support) Stuff

The engineered system is designed to produce certain things and interface in certain ways. To accomplish its function the system needs a lot of support. For example, a system is designed to process input data in a specific digital format and output the processed data in a specific digital format. The format constitute interface constraints, but the connector for the digital formats constitute non-functional or functional support interface constraints.

The box that keeps the system from getting wet constitutes a non-functional or functional support requirement while the data processing requirements constitute a functional requirement. Meeting non-functional requirements is as necessary to the system as meeting functional requirements.

Let's return to the example and start sorting the stakeholder user stories into the four categories. The stories will be highlighted using four colors with the following meaning:

1. Untranslatable Stuff
2. Interface Stuff
3. Requirements Stuff
4. Non-Functional Stuff

Beginning with the Course Instructor's Pedagogical Requirements. The marked version is:

As a course instructor I need equipment in the control systems lab that provides students with a useful hands-on experience.

In order to be useful the lab equipment used must not be mysterious (or overly 'black box').

Students need to see the inputs and outputs of the system and understand that they can replicate similar systems in their careers.

The power amplifier module must be constructed of a technology that is familiar to both mechanical and electrical/computer engineering students at a late-junior year level. In order to meet this familiarity requirement the design is constrained to use a power operational amplifier.

The power amplifier needs to be a unity voltage gain non-inverting buffer.

The power amplifier must provide a minimum continuous output current of 8 amps.

The input impedance of the power amplifier must be greater than or equal to 10 kilohms.

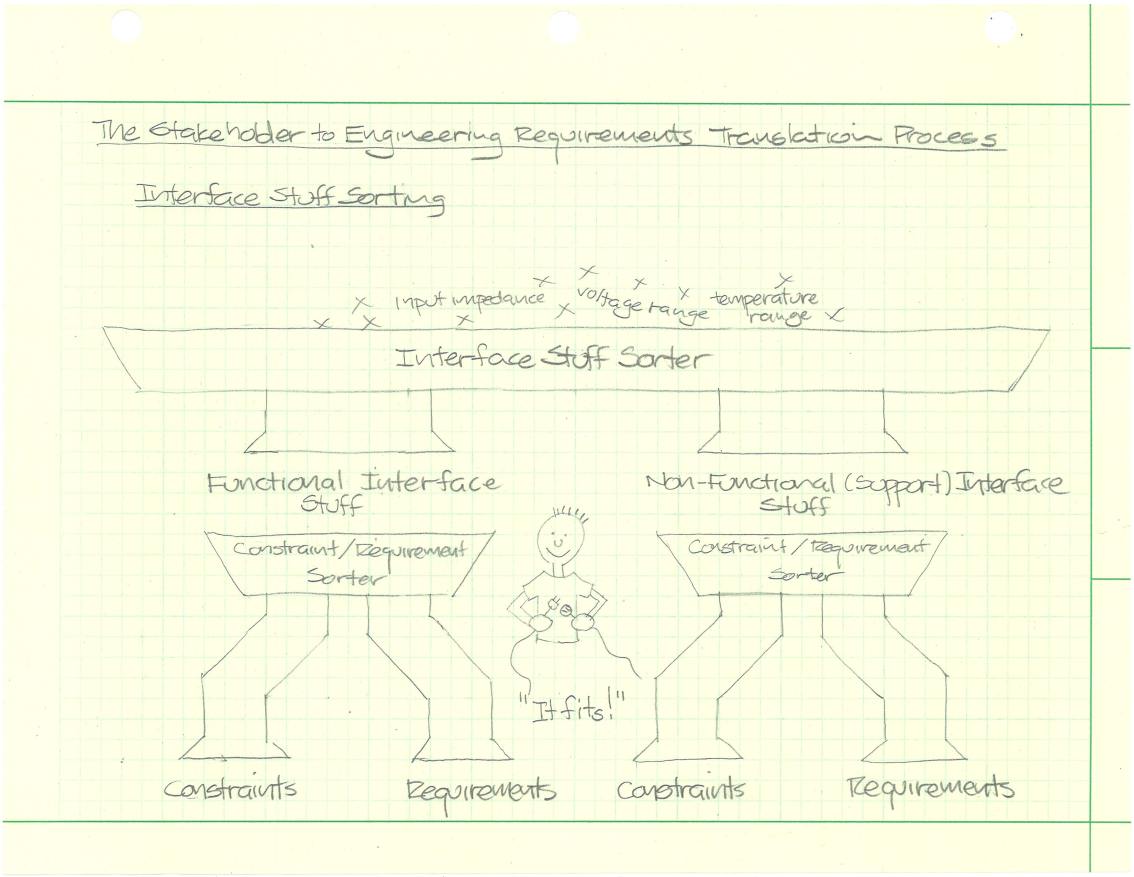

Figure 2: A machine that sorts Interface Stuff into appropriate categories.

The largest time constant of the power amplifier must be less than 0.001 seconds.

The Course Instructor's Pedagogical Requirements has one item classified as Interface Stuff dealing with the input impedance of the amplifier. This item must be further sorted by the machine in Figure 2. This machine determines if the item is a constraint or a requirement.

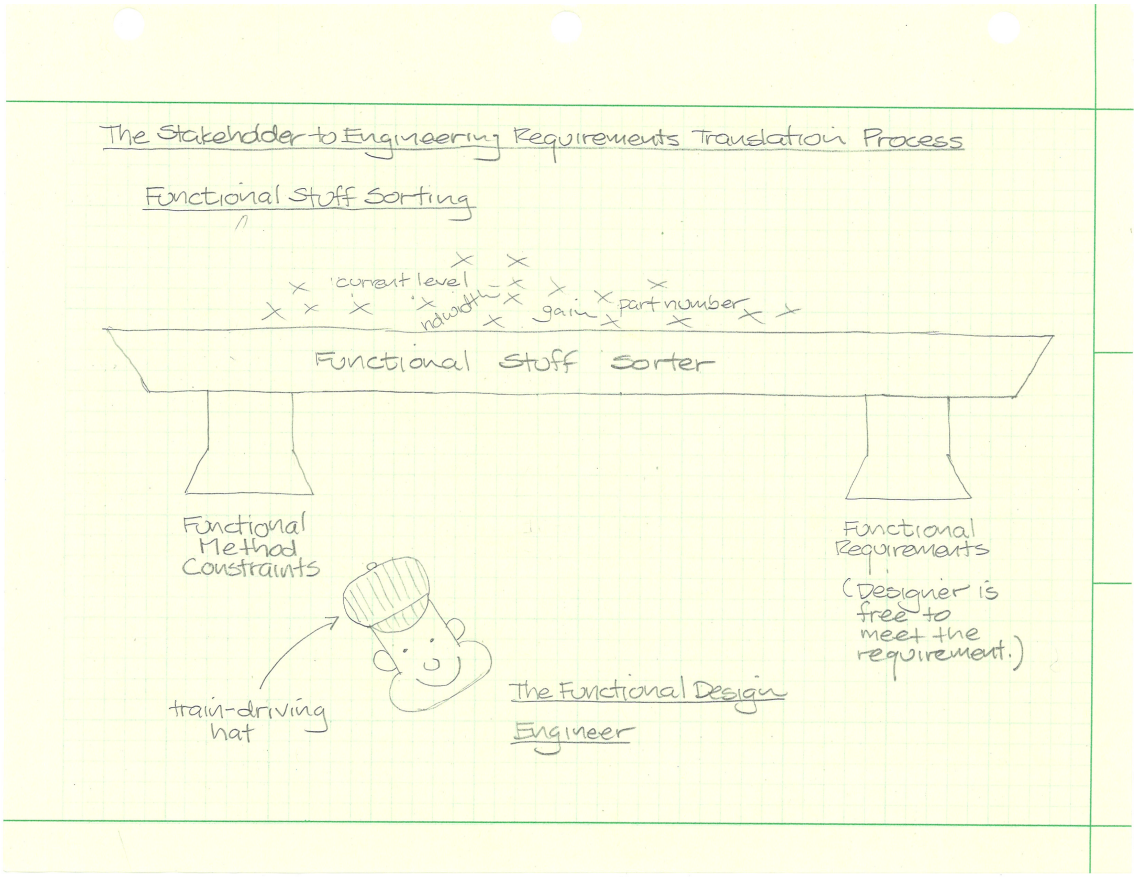

Figure 3: A machine that sorts Functional Stuff into appropriate categories.

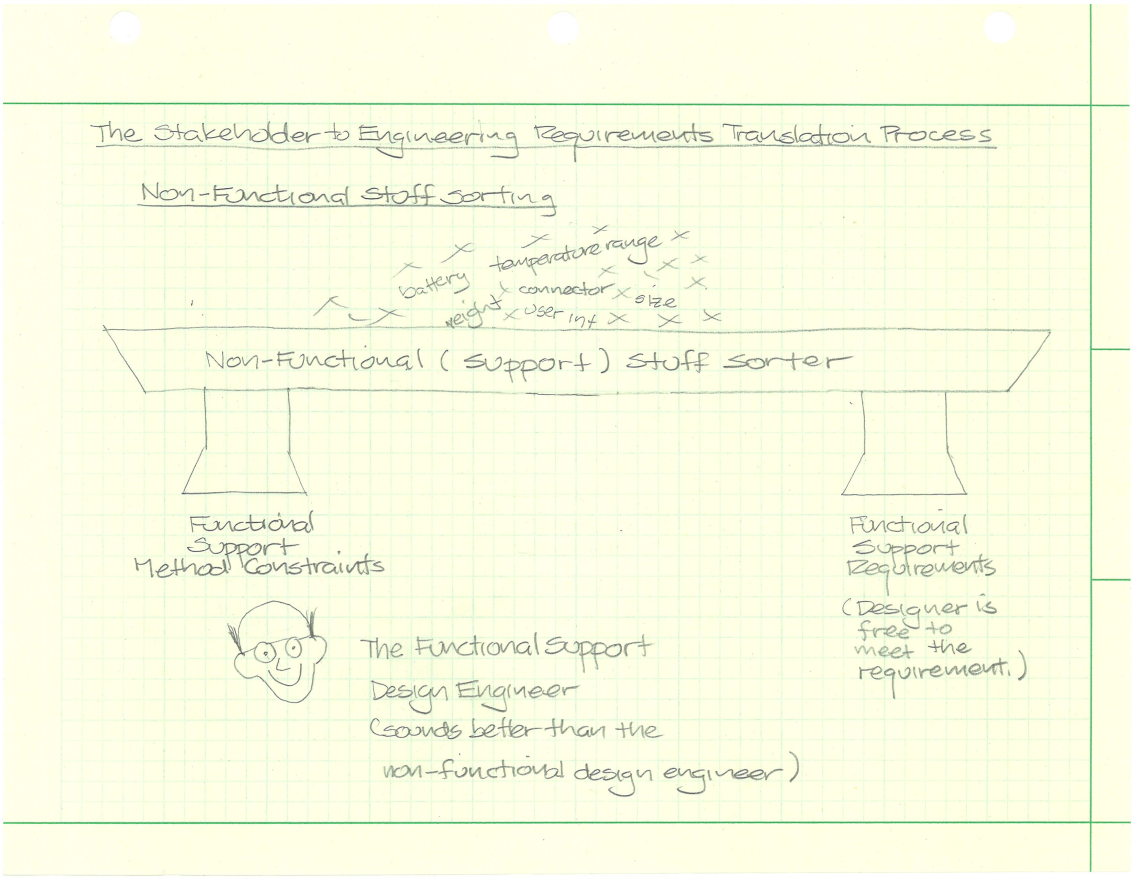

Figure 4: A machine that sorts Non-Functional Stuff into appropriate categories.

4.1 Interface

4.1.1 Interface Method Design Constraints

4.1.2 Interface Design Requirements

4.1.3 Interface Support Design Constraints

4.1.4 Interface Support Design Requirements

4.2 Functional Requirements

4.2.1 Functional Method Design Constraints

4.2.2 Functional Design Requirements

4.3 Support Requirements

4.3.1 Support Method Design Constraints

4.3.2 Support Design Requirements

5 VERIFICATION OF REQUIREMENTS

We have to know that we designed what we were required to design by this document. This process is called 'verification' and it answers the fundamental question: "Did we build what we said we would build?". We verify requirements by testing to see if the requirements are met. Design teams must test every requirement in order to prove that the requirement is met. Testing each requirement means that each requirement must cast in some quantifiable way. In this section you specify how you will test each requirement to show that it has been met.

Don't worry! It is not as bad as it sounds. Sometimes (not always!) you can verify or test a requirement simply by looking at the completed system to make sure some required thing is present.

Testing often 'takes it on the chin' in terms of project schedule. Since integrated system testing typically occurs near the end of a project, the time for testing is compressed against the deadline. People start short-cutting tests to stay on schedule. Sometimes you may get away with it but it is never a good idea either technically or ethically. Epic failures have occurred because of truncated testing. One such failure occurred during the testing of the Hubble Space Telescope. The following is an excerpt from the the official report detailing the failure.

> *Reliance on a single test method was a process which was clearly vulnerable to simple error. Such errors had been seen in other telescope programs, yet no independent tests were planned, although some simple tests to protect against major error were considered and rejected. During the critical time period, there was great concern about cost and schedule, which further inhibited consideration of independent tests.*
>
> *The Hubble Space Telescope Optical Systems Failure Report-NASA November 1990*

If you are interested the whole report is available at (`https://www.ssl.berkeley.edu/~mlampton/AllenReportHST.pdf`).

The Hubble error wasn't caught until the telescope was deployed in space. Can you imagine the cost of fixing this problem? It is not simply a case of bundling you off with your instruments and putting you up in a fancy hotel for a week or two. Some estimates set the price at about $1 billion.

The Dilbert comic strip has a similar, and darkly amusing, view of testing truncation.

(`http://dilbert.com/strip/2010-08-21`)

(http://dilbert.com/strip/2009-07-01)

The key to completing this section is that every requirement has an associated test. The best practice in this section is to match the sub-paragraph numbers in the previous section to the sub-paragraph numbers in this section, e.g the requirement in 4.3.1.6 is covered by the test described in 5.3.1.6.

Possible verification methods include:

1. *Inspection:*
   Inspection is a method of verification consisting of investigation, without the use of special laboratory appliances or procedures, to determine compliance with requirements. Inspection is generally nondestructive and includes (but is not limited to) visual examination, manipulation, gauging, and measurement.

2. *Demonstration:*
   Demonstration is a method of verification that is limited to readily observable functional operation to determine compliance with requirements. This method shall not require the use of special equipment or sophisticated instrumentation.

3. *Analysis:*
   Analysis is a method of verification, taking the form of the processing of accumulated results and conclusions, intended to provide proof that verification of a requirement has been accomplished. The analytical results may be based on engineering study, compilation or interpretation of existing information, similarity to previously verified requirements, or derived from lower level examinations, tests, demonstrations, or analyses.

4. *Direct Test:*

   Test is a method of verification that employs technical means, including (but not limited to) the evaluation of functional characteristics by use of special equipment or instrumentation, simulation techniques, and the application of established principles and procedures to determine compliance with requirements.

5.1 Verify Coverage of Stakeholder Requirements

The tester verifies that everything that the stakeholders have asked for are covered by one or more requirements. It is a good idea for the requirements author(s) to perform a similar check at this point. The tester is likely to do his own analysis or disagree on points in yours, but the exercise itself is valuable. And if you do the analysis you might as well write it down here.

5.2 Interface

5.2.1 Functional Interface Constraints

5.2.2 Functional Interface Requirements

5.2.3 Support Interface Constraints

5.2.4 Support Interface Requirements

5.3 Functional Requirements

5.3.1 Functional Method Constraints

5.3.2 Functional Design Requirements

5.4 Support Requirements

5.4.1 Support Method Constraints

5.4.2 Support Requirements

A tabulation of all the requirements and the testing method with a blank space for results is useful for whomever is doing the testing.

Paragraph Number	Test Type	Tester's Name	Pass/Fail	Date

Now read over your completed specification and make additions and corrections. Find others who will be willing to read and comment on the specification (hopefully they will still like you when they are done). The more eyes the better. Ask yourself if you handed this spec to a competent classmate what would they build?

Congratulations! You have written an engineering specification and that is no mean feat.