How to Constructively Read a Spec

by Yaakov Ellis

When you first think about it, reading a spec is not something that sounds like it should be so complicated. And indeed, reading a spec *the wrong way* is an extremely uncomplicated process:

The Wrong Way to Read a Spec

1. Read it from start to finish (or if you are lazy just skim it, or if you are really lazy just hit the tldr and stop there)
2. You are done

If all you do when you read a spec is to read it, then unless this spec is already perfect, your perusal has not helped make the spec better. And although acquiring the knowledge that the spec is intended to convey is always a primary goal of reading a spec, an additional primary goal should always be to improve the quality of the spec. (To be clear: improving the quality of a spec is not an end unto itself; rather, having a high quality spec is an essential part of executing any non-trivial project).

So What is a Quality Spec?

I'm glad you asked! A quality spec will do the following:

1. Clearly state the issue being addressed
2. Clearly define the desired outcome on a functional level (and technical solution if it is known)

In its simplest form, that is what it really boils down to. Of course, that is a very high-level definition. In order to achieve these goals, many factors come into play:

- Provide context for the issue being addressed
- Give reasons and arguments for the solution (potentially exploring alternatives)
- Define important terms and concepts
- Break down the way in which the solution will need to be implemented (always <u>needs to be done</u>, but might take different forms depending on whether this is a technical or functional spec, and who the audience is)
- Define success criteria
- Where appropriate, give user stories
- Using language throughout that promotes clarity, and brevity while avoiding redundancy and confusing language

So if those are the ingredients of a quality spec, what are factors that can reduce the effectiveness of a spec at achieving its goals?

- Lack of context/introduction
- Unclear definition of the issue being addressed
- Lack of, ill-defined, or poorly made reasons and arguments for solution. Lack of exploration of alternatives where appropriate.
- Ambiguous or missing definitions of key terms and concepts
- Lack of or incomplete breakdown of the solution that is to be implemented. The breakdown can be there, but might not go down to the detail level needed for more complicated issues.
- Missing or incomplete success criteria
- Internal contradictions or Redundant/confusing language in details of the spec (in and between any of the sections)

So therefore, reading a spec the right way means…

…reading it with a critical eye while having the goals of:

1. Calling attention to, clarifying and fixing any factors that can reduce the effectiveness of the spec
2. Encouraging all of the success factors for the spec

It's that easy :-)

Believe in Yourself

Of course, when put this way, reading a spec changes from an easy to execute, passive task into a complicated endeavor. It is the easiest thing in the world to go back to the uncomplicated (and ineffective) way of reading a spec. Before doing so, there are some things to keep in mind:

Take it one sentence at a time

Like any large task (including executing on the project itself), it is much more achievable when you take it one bit at a time. Read the summary, then the problem definition, solution summary, breakdown, etc. While going through it look for the items that are missing, that are contradictory, the ill-defined edge cases, etc—and when you see them, write a comment or make a suggested edit. If something is not clear, write a comment or make a suggested edit. And then move on. One paragraph at a time.

Your opinion is important & there is no such thing as a stupid question/comment

You were hired or chosen to work on this project/team for a reason: because your opinions, skills and knowledge are respected and valued, and your input is something that people are interested in. Otherwise you wouldn't have been copied on the spec, you wouldn't have been hired, and your opinion wouldn't have been solicited. So, if you see something in a spec that requires attention, then it is your duty to comment on it.

Often people will have questions when reading a spec and will decide not to leave them while making the argument: if it was a question that needed to be asked then surely someone else would have asked it (and since no one else did, it is an obvious question that will only highlight my own imagined inadequacies). Do not let Imposter Syndrome lead you into this trap. What you have to say **has** value and others want to hear it.

Your thoughts and input on a spec do make a difference

Even if it is adding a missing comma, calling attention to one redundancy, pointing our a minor inconsistency or asking about one seemingly far-fetched edge case—every little bit makes a difference. Don't underestimate this and don't use this as an excuse for neglecting details that require attention.

Assume that you are the only one who will be **really reading** the spec. You can make a big difference through by just reading carefully and letting others know what you think in a constructive way.

The Author of the Spec Also Plays a Key Role

Of course, comments and suggestions are only of value if they are a catalyst for change and improvement. It is up the author or caretaker of the spec to review all changes and use them to update the spec where appropriate.

Some important things to remember:

It's Not Personal

If comments and suggestions are made on a spec, it is nothing against the author. Every spec is imperfect after its first draft. Perfecting a spec and making it a more effective tool for executing on a project is a team effort. Comments and suggestions are merely the form in which this collaboration is manifested. Of course, comments and suggestions should always be made relative to the content, and should definitely not be made in any way sort of aggressive way. As long as everyone is on board with the goals, this should not be a problem.

If a comment or suggestion does not lead to change, then it never really happened

Very often someone will leave a comment or ask a question on a spec. The spec author will respond to the comment. Maybe there will be a back-and-forth and several rounds. And then seeing that the question has been answered, the spec author will *delete the comment* without doing anything else.

This is a **mistake**.

If one person had a question or comment on a suggestion, then it means that something was not clear with the spec and needs to be clarified. This is true even when the answer to the question was "no, that is not a change that we will make". It is very likely that other people will have the

same question (and might not be as conscientious about reading specs as you are). Resolving a comment without changing the spec or leaving something to address the original question makes it as if the comment was never made. Even the original participants in the comment exchange will be hard-pressed to recall the details even a short while later, and of course those who never saw the original exchange will realize no benefit whatsoever.

The changes that we are talking about don't have to be significant in nature. Very often a footnote or parenthetical aside, or a few words of clarifying language is all that is needed in order to leave a meaningful relic of the erstwhile discussion. However, these are important pieces of the evolution of a spec as it moves closer towards its final state, and will introduce an additional level of clarity for future readers and consumers.

Final Thoughts

Most developers hate (or at least extremely dislike) writing specs. They are not fun to write. They are no fun to read. Many recognize them as a necessary evil, and thus grant them the minimal time and effort possible before moving on to more "productive work".

I also really don't like writing specs. I find every reason not to work on them and procrastinate an extraordinary amount. But over the years I have come to recognize that it is extremely hard (if not impossible) to complete a complicated project on time and budget, to meet the functional expectations of stakeholders to deliver a quality product without devoting the necessary time and energy *up front* to the spec writing process. It is my hope that after reading this far (for which I sincerely thank you) that the next spec you interact with will end up in a better state through your efforts and attention.