SRS Frequently Asked Questions

Spencer Smith

November 3, 2022

A list of Frequently Asked Questions are given below. The answer to the question is shown below the question, in italic font.

1. If we are solving an "intermediate problem," should we state the output(s) of that step or we should only talk about the raw inputs of the software and ultimate outputs?"

   Without any other qualifying information, I would say that the intermediate steps are not relevant in the SRS. The SRS should be abstract. It should say what we want, not how to calculate it. However, the world of requirements specification doesn't always lend itself to an absolute answer. :-) There are definitely cases where the best course of action is to describe how to get from a to c by first passing through b. The cases where specifying the intermediate steps makes sense are as follows:

   (a) *If the intermediate steps have value on their own, then they should be included as a goal. For instance, in the Solar Water Heating System example the energy in the water and the PCM depends on the temperatures. If the ultimate goal was to know the energies, it still would make sense to output the temperatures, since they have value on their own.*

   (b) *If the intermediate steps make the specification clearer, or easier to understand, they should be included. Although a specification should say what is desired, there are cases where the appropriate way to specify "what" is to say "how." This is called an operational specification. An operational specification gives the steps for the intended behaviour. All other things being equal, our preference is still a descriptive specification. A descriptive specification*

says what is required in terms of the desired properties, without reference to how it will be done. As an example, a descriptive specification for finding the minimum of a list would describe what is meant by minimum. A descriptive specification would say that the output is the element of the list that is equal to or less than all other elements. The corresponding operational specification would give the algorithm for finding the minimum. The topic of operational versus descriptive specification is covered further in these slides (starting around slide 22). One of the points on operational specifications that sometimes confuses students is that the operational specification is phrased in terms of "how," but the intention is still to say "what." That is, the implementation is not required to use the steps given for how, as long as the results match the what. In the above example, the operational specification for the minimum of a list gives an algorithm, but the implementation is free to use any algorithm, as long as the output returned matches the output that would be returned by following the operational specification.

In the SRS intermediate calculations can come through supporting theoretical models, general definitions, instance models and/or data definitions. The short answer to the original question is that sometimes it makes sense to include an intermediate problem in the specification, and sometimes it doesn't. A generalization isn't really possible. The answer depends on the specific problem at hand.

2. If your work depends on other applications or libraries, how does this show up in the SRS?

   The first thought may be that you should assume that the libraries are available; that is, that you should capture this information through assumptions. I can see why assuming that a library exists might be called an assumption, but this isn't the best location for this information. Assumptions are used to refine the scope; they take a very general problem and turn it into something that we have a hope of solving. For instance, assuming that a function is differentiable is needed for many theoretical proofs. This assumptions is saying that the software won't work for all functions; the class of functions that will work has been restricted. Assumptions refine "what" you are solving; they don't refine "how" you

are going to solve the problem. Building on the work of others is relevant to make the problem feasible, but it isn't technically needed to make the problem solvable.

There are two spot in the SRS to show that you are building on the work of others: System Context and System Constraints. The System Context shows the boundary between your software and the environment in which it works. If the software will depend on services from other libraries, the availability of these services should be mentioned. In the system context diagram there would be other boxes that your program would point to. You are then explicitly saying that your program will depend on these services. Ideally, the services will be given generic names. That is, unless you have no choice, you should leave it open as to which library will actually provide the services. In this way you keep your document abstract. However, there are cases where the design decision is imposed on you. In these cases you would include the names of the specific library, or libraries, as System Constraints (and list them in the corresponding section of the SRS).

3. As I work on my Verification and Validation plan I find part of the SRS should be changed. What should I do?

   This is completely natural and expected. Students in the class make this observation every year. We cannot really have perfect knowledge when we are writing the first draft of our SRS. We learn by jumping in and starting the documentation. As we get further in the project our understanding improves.

   This is why we have two "official" iterations of the SRS in the course. You should also have many sub-iterations as the term goes along. Our aim is to have documentation at the end of the course that fakes a rational design process.