

Mission Possible ... With Good Requirements

Wayne Turk

Suppose that you gave a dozen contractors a single requirement: "Build a vehicle to cross the English Channel." What would you get?

It might be something that would fly—a balloon, a helicopter, an airplane, or a rocket. It might be some kind of a boat or barge. Or it might even be a submarine or something that crosses on the sea bottom. But whatever you get might not be big enough, fast enough, or carry enough people or cargo. It might be easily detectable and too vulnerable to hostile fire. It might be too expensive. It would meet your stated requirement but not all your needs. Why? Because it takes a good comprehensive set of requirements to get the right final product that meets the users' needs.

Nobody would ask for something that is based on a single requirement (although there might be rare times when you could). There are usually hundreds or even thousands of requirements. But given the way that some of those requirements are sometimes submitted, there might as well be only one. Too often, requirements are poorly written. They are ambiguous, vague, or not understandable. There may be contradictory requirements. And there may be ones that are not feasible technically or financially.

Anatomy of a Good Requirement

What constitutes good requirements and how do you develop them? This article cannot be comprehensive, but the following should give you a working knowledge of what constitutes good requirements and how to develop them, whether you are the one who has to write them or the one who must build to them.

Turk is a retired Air Force lieutenant colonel and a manager with SRA International supporting National Guard Bureau information technology projects and distance learning classrooms. He has supported projects for DoD, other federal agencies, and non-profit organizations. Turk is a frequent contributor to Defense AT&L.

Requirements That Get Results

- Each requirement must be concise and written in complete sentences.
- Use active voice and good grammar.
- A requirement must stand alone as a complete requirement.
- Requirements must be clear, understandable, and unambiguous.
- Don't combine requirements using words like *and, or, also, with*.
- Avoid using *etc.*, which opens the way for interpretation.
- *Shall, will, and must* make requirements mandatory.
- Avoid terms that invoke possibilities: *may, might, could, should, perhaps, and probably*.
- Don't use words like *except, if, when, unless, or but*, which provide escape clauses.
- Use defined terms such as *no greater than* or *no less than*. Avoid vague or undefined terms: *greatest extent possible, maximum, minimum, state-of-the-art, flexible, user-friendly, efficient, several, improved, adaptable, adequate, and simple*.
- Each requirement must be verifiable (think testable, but there are other verification strategies).
- Don't gold plate requirements.
- Avoid wishful thinking or impossible goals.
- Do not design the system or product in the requirements; just give the results required, not how to get those results.
- Use the same level of granularity for each requirement.
- Ensure that requirements are not contradictory or mutually exclusive.
- Ensure that requirements are organized, structured, and numbered.

Be Necessary

The first characteristic of a good requirement is that it is absolutely necessary. With today's fiscal constraints, there is rarely any room for nice-to-have, desired, or frivolous requirements. A requirement like "the gross takeoff weight of the aircraft shall not exceed 160,000 pounds" is imposed for a reason. It might be based on runway surface restrictions, deck restrictions on an aircraft carrier, or some other constraint. Another part of necessity is the need to solve a problem. For example, a requirement to have an individual ID other than a social security number was necessary for DoD's electronic medical record. While the SSN should have been enough, it turned out that there were too many errors and potential problems.

Be Correct

The requirement must be accurate as to what the product needs to deliver. The normal source of information is the customer or end user. Only a knowledgeable user can determine if a requirement is correct. That's why having users and functional experts involved throughout the require-

ments process is a very good idea. It can save a lot of pain and wasted effort. Otherwise you are just guessing.

Be Unambiguous

Requirements must be unambiguous. Multiple readers should come to the same understanding of what each means. If a requirement can be interpreted more than one way, you are in trouble—chances are that the developer or builder will interpret it the wrong way. Terms like “user-friendly,” “fast,” “easy,” “flexible,” “state-of-the-art,” “maximize,” “minimize,” or “efficient” mean different things to different people. Avoid them like the plague. Don’t allow the customer or user to include them. Get a specific definition of what is really needed—and get it in simple language.

Be Attainable

All requirements must be feasible, attainable, and achievable. These words are almost synonymous and, in this case, mean that the product can be produced with today’s technology and with the time and money available. A few years ago, stealth technology or wireless computers were not technically feasible. Advancements in technology rapidly change what can be done, so a little flexibility is needed. Who knows, within the next few years a Star Trek phaser or transporter may be feasible. But don’t get ahead of technology.

Be Orderly

The requirements for any project must be prioritized. This priority is normally set by the end user, but the program manager may have a say. That is especially true when the user sets the same priority on a number of requirements. Along with operational needs, other factors can influence priority. For example cost can play a huge role. If meeting one requirement will cause the expenditure of 75 percent of the budget, you are probably not going to have that as your highest priority. Technical risk and schedule impact are other influencing factors. You may have to weigh these factors and work with the users to make them understand the effect on priorities. And if you are the user, be willing to listen. You want the product to be what you need and can use.

Be Measurable

Another necessary characteristic is that all requirements must be quantifiable, measurable, and verifiable in some way—through inspection, analysis, demonstration, simulation, and testing, among others. In most cases, we look toward testing, but that can be very expensive and time-consuming. A requirement for size is a perfect example of one that is easily quantifiable and measurable. Inspection can determine if a tank will fit on an aircraft. A trained soldier could be used to verify by demonstration whether a radio is repairable using the provided documentation and available spare parts. Computer simulation can provide answers without destroying components. Testing may

be something as “simple” as firing a missile at a target, or it may require weeks or months, as in the case of integration testing of complex software that has to interact with other software applications. Just remember that every requirement must be verifiable in the most expeditious and least expensive manner possible.

Be Organized

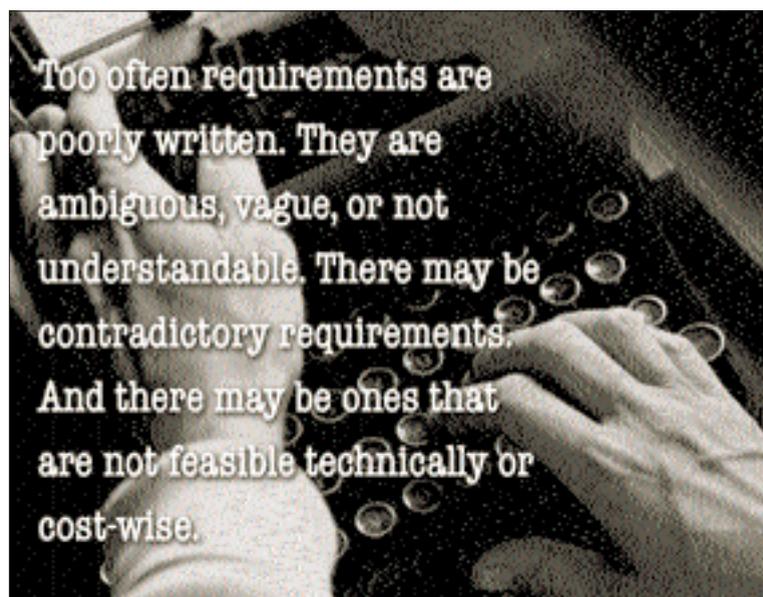
Verifiable is related to traceable. While especially critical in software development, in any project you should be able to trace a requirement from identification through development to verification. Requirements need to be written with the same terminology and the same standards throughout. It also helps for them to be organized and grouped into defined categories. This allows you to find duplications, inconsistencies, and contradictions. For software, linking to the design elements, source code, and test cases can be a time-consuming but important function. If you can’t link it from beginning to end, how do you know whether you have met the initial requirement?

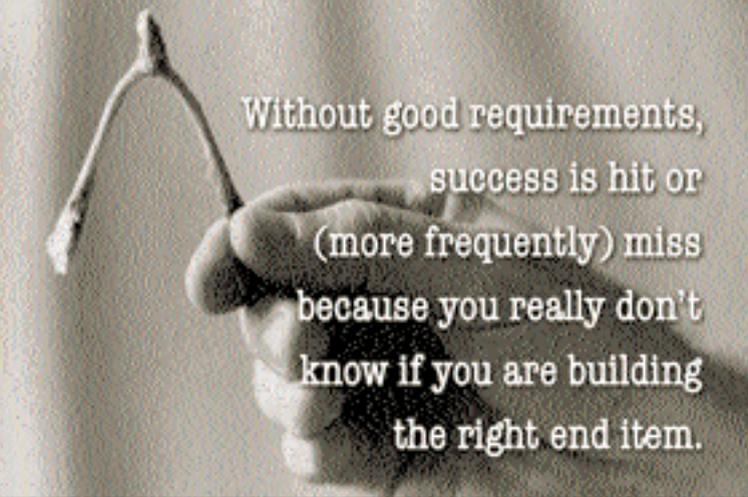
Be Results-Oriented

Finally, requirements must be results-oriented. The objective of the complete requirements package is to provide a product that meets the users’ needs and/or solves a problem. It doesn’t have to look good, involve the latest technology, or do all kinds of extra things. It must provide the results and the product that are wanted. If a radar system can track a hundred targets of a specified size at a defined distance but can’t present the data in a way that is understandable, it doesn’t have the results that are needed by the user, and it will be deemed a failure.

The Requirements Package

Some type of a formal requirements package is necessary. In most government agencies, there are specified documents for the task. It may be a system requirements specification, functional requirements document, opera-





Without good requirements,
success is hit or
(more frequently) miss
because you really don't
know if you are building
the right end item.

tional requirements document, or some other similar document or series of documents. Whatever it is, it will become the bible for the project. As mentioned earlier, it needs to be organized with requirements grouped in some logical fashion.

The project will also need a tool for tracking requirements from initial identification through deployment. Your organization will want to look at what tool best meets your needs—preferably one you already own to avoid incurring extra costs. You want a tool that allows identification and tracking throughout the process and can provide an audit trail of all changes, who made them, and when they were made. It should have the capability to sort in different ways. For a small project, a simple spreadsheet would probably work fine, but for a large and complex program with hundreds or thousands of requirements, you need a tool designed specifically for requirements management. But however you track requirements, keep the audit trail up to date. Keep a record of both current and historical requirements, including any that are deleted because many times requirements resurface.

Scope creep and changing requirements can be slow poison to a project. A simple change here can lead to another there until the project is in deep trouble, and the final product bears only a faint resemblance to what was originally planned. These are insidious problems that can cause schedule slips, cost overruns, and unhappiness for all concerned. Yes, some flexibility is needed, especially with a project that stretches out over time. Needs change, as does technology. Organizations restructure or reorganize. Vendors come and go. Budgets wax and wane. Customers and their level of support may be in flux. All of these things happen, and you must accept some change—but try to keep changes to the *requirements* to a minimum.

Potential Traps

There are a number of obvious and not-so-obvious traps that a requirements writer can fall into. The most obvious is poor word choice, which leads to ambiguity. Does the following requirement say what is *really* meant: “The fire alarm shall sound when smoke is detected, unless

the alarm is being tested or the engineer has suppressed the alarm”? That one could lead to a very dangerous situation.

Rambling requirements can also cause confusion, especially when terms are not defined well or they have unclear antecedents. Remember, clarity is your goal. “Provided that the designated data from the specified columns are received in the correct order so that the application is able to differentiate among the data, the resulting summary data shall comply to the required format in section 2.3.1.” I give up—what does that mean?

Supplying too much data or being too specific may force the design of the system into a predetermined path and stifle innovation. This happens frequently and is usually marked by naming specific required materials, components, software objects, or database fields. In some cases, requiring a specific component is necessary for compatibility, maintenance, cost savings, or supply capability; but be alert, and don't fall into the trap of requiring something just because you like it or think that it would be a perfect solution.

Beware of wishful thinking. Nothing is 100 percent reliable, able to handle all unexpected failures, able to run on all platforms, or is guaranteed upgradeable to all future versions. You can see the common theme: “all.” Just as bad are “none,” “zero,” and “never.” “The brakes will be 100 percent effective in normal situations.” “The network shall handle all unexpected errors without crashing.” Dream on; it isn't going to happen. While it is possible to write requirements for 100 percent reliability in some products, it will require redundancy (usually multiple redundancy), and they will be expensive to build.

Without good requirements, success is hit or (more frequently) miss because you really don't know if you are building the right end item. Sure, it's time-consuming to write good requirements, but it's well worth the effort because time spent in the beginning can actually save time later. Good requirements writing comes with practice, thoughtful consideration, and plenty of review and discussion. And by following the basic rules, of course:

- Keep users involved.
- Develop and refine requirements.
- Define and use consistent terminology.
- Organize requirements.
- Monitor/track development and changes.
- Document all requirements and changes and why they changed.
- Make requirements management one of your repeatable processes.

The author welcomes comments and questions. Contact him at wayne_turk@sra.com.