When *STP Magazine* Editor Eddie Correia invited me to write this, he described the Future Test column as a thought-provoking opinion piece about the future of testing, especially with respect to automation.

This article is about requirements. So, the first thought I hope I'm provoking is: What's an article on requirements doing in a testing magazine?

Requirements do seem increasingly to occupy the attention of testers. In fact, I recently presented a featured speech whose title I'd overheard a conference attendee comment, "I Went to a Testing Conference and All They Talked about Was Requirements."

Testers Need to Know

Certainly testers have reason to be concerned about requirements. After all, Black Box Testing is charged primarily with confirming that developed systems meet requirements. Consequently, in order to create tests that demonstrate the delivered system meets the requirements, it is essential for testers to know what said requirements are. Too often, though, testers are expected to create tests without having been informed what the requirements are, or at least not in a timely manner.

Without a reliable requirements basis for determining what content to demonstrate, testers tend to be relegated to guessing what the system is supposed to do. The tester's experience with the organization's business and with testing in general can increase chances of good guesses, but guessing is bound to miss things. Moreover, the less the tester knows about the system's intended functionality "guts" content which provides most of the system's value, the more the testing is likely to concentrate on GUI graphical user interface format characteristics. While usability surely is important, usability is mainly relevant within the functionality's context, which stems from the requirements.

Despite some popular efforts to characterize testing without requirements as a preferred practice, the fact that it often reveals errors is more an indictment of poor development than a testament to the testing. Of course, too, defects are especially likely when developers also proceed with inadequate understanding of requirements,

Format Factors that Restrict Knowing

A tester's ability to find out the requirements similarly is affected by format and content aspects. Format factors include organizational structure/politics and physical logistics. At conferences, much attention is devoted to lamenting organizations' failure to provide testers needed requirements information but seldom provides assistance beyond sympathetic exhortations that testers should be treated better and others should care more about quality.

Logistics—the physical ability to access documented requirements--are easier to address than politics and can be automated. At a minimum, the requirements must be captured in some retrievable form, for example on paper, in an electronic word processing document

or spreadsheet, or carved in stone tablets. A number of commercially available automated requirements management tools capture requirements in a database which facilitates additional capabilities, such as annotating with priority and other descriptive characteristics, identifying and controlling changes, and producing traceability matrices which cross-reference to the artifacts where each respective requirement is implemented and tested.

Since a system can be only as good as its requirements, for years many authorities (and tool vendors) have touted *requirements management*, and by implication requirements management tools, as the biggest single determinant of effective system development. Certainly, both developers and testers have a better chance of delivering quality when the tool enables them to know what the requirements are.

Content

Requirements management tools support administrative/clerical activities but have little to do with the more important content issue of whether the requirements are correct. Testers wishing to address issues of content generally do so from the perspective of their potential role as reviewer of requirements which someone else has defined.

To date, tools mainly have been incidental to reviews, primarily just using an existing requirements management tool to capture review comments next to stored requirements. Some newer tools analyze requirements text to identify clarity and consistency problems, which often is characterized as content but really is mainly format. That is, a requirement can be clear and consistent but wrong; and clarity and consistency are irrelevant for requirements which have been overlooked. Additionally, some tools are based on analyzing designs, which may be called requirements but are not the REAL requirements.

Again, the most commonly articulated challenge often seems to be many organizations' reluctance to involve testers in requirements reviews. Such omissions may reflect bigger issues that testers seldom recognize. For instance, testers who lack relevant subject area knowledge may be perceived (rightly) as not contributing to the review. Moreover, conventional testing industry wisdom is that *testability* is the main requirements issue. Testability is a form of clarity, and others often find testers' harping on testability to be annoyingly trivial nitpicking that may warrant again excluding testers from reviews.

However, I've noticed recent trends shifting emphasis toward defining requirements content. Testers increasingly seem to be deciding defensively that the only way to be sure of having the requirements they need is to define them. This raises issues such as duplication of effort, whether testers have suitable skills, and impacts on testing due to testers' time being diverted to defining requirements.

Also, several prominent tool vendors have begun offering tools aimed at assisting requirements content definition; and I think it's safe to say that such tools will gain the attention that the format-oriented requirements management tools never achieved.

While automation is helpful, ultimately the most important requirements tool is the one we so often resist using—the tool we carry around all the time between our ears.

Robin F. Goldsmith, JD is President of Go Pro Management, Inc. consultancy, creator of the Proactive Testing™ methodology, and author of *Discovering REAL Business Requirements for Software Project Success*. Reach him at robin@gopromanagement.com.