## Zero, one, or more testable requirements

There was a project, there was a deadline, the software was delivered late, it was the holiday season, and the person responsible for the system integration test was on holiday. So I was asked to perform a system integration test of a new protocol against the requirements set for the protocol. According to the still-prevailing practice in many organizations today, the requirements were written in, and made available as, a plain text document.

It struck me that many of the requirements were not testable.

For example, I found the requirement:

> *A configuration message includes for each service a list with 0, 1, or more objects that are involved with the execution of the service.*

Well, no matter what test I designed, the requirement would always be met, while a test should potentially have either a positive or a negative outcome. I wanted to change the requirement to something like this:

> *A service in a configuration message must include a list with the objects that are involved in the execution of the service.*

If no objects were involved with the execution of the service this rule would obviously not apply so the *zero* situation would be covered as well. But that was not my assignment, and I would do what I normally do: write business rules.

But I had a bad feeling about this situation because I like to do a good job. What was the cause? Were these requirements bad? Should we update the requirements document? Well, when the tester updates the requirements we then need to have a new tester to test the updates of the tester … that does not make any sense.

Given my background, my reaction was not surprising. I am used to working with business rules and I always check that they are testable. But does that also apply to requirements? Apparently not.

A survey of the literature and some *google* searches confirmed my intuition. While business rules are required to be testable,[1] a requirement *may* be testable. There are "tips for writing testable requirements"[2] and there are criteria to distinguish between testable and *non-testable* requirements.[3] However, a requirement does need to be *verifiable*,[4] being defined as:

> *The implementation of the requirement can be determined through basic possible methods: inspection, demonstration, test (instrumented), or analysis (to include validated modeling & simulation).*

Does this confirm that requirements and rules are just very different things as most experts in both communities agree? Or can we learn something from this example? When I read the requirement I couldn't resist asking myself what the intent of the author must have been. I believe the author must have intended to say:

> *The system must be able to process a list of the objects that are involved in the execution of the service.*

which is much more precise AND testable — although the word 'process' is much too generic in my opinion. What does it mean? To be fair, I don't know exactly, so that's why I started with that word. But if I were the analyst this would be on my list to find out.

Also, it would be good if this statement could be complemented with a business rule stating when the optional object is mandatory and what objects to expect:

> *A service in a configuration message must include a list with all objects that are involved in the execution of the service.*

Now, being a business rules expert (and thus very biased!), I believe these latter two statements are simply more precise than the original. And isn't that what would be best in all situations related to system development?

The irony is that I was inspired and influenced some ten years ago by the writing guidelines in *The Handbook of Ambiguity*[5] which is all about requirements. We have used these guidelines and further restricted business rule authors by suggesting rule patterns and promoting rigid knowledge representation techniques like decision tables. And now I conclude that requirements engineers (or authors) should learn from our experience with these more controlled natural languages.

Here are some resources for further reading:

- *The Handbook of Ambiguity* is a great source of examples about the consequence of loosely-written requirements.
- SBVR Structured English,[6] RuleSpeak,[7] and TableSpeak[8] are examples of controlled natural languages developed in the business rules community. Many organizations use a subset or extension of these language patterns.
- Do you want only one page to start with? From my website, you can download and print (and follow) these most important guidelines for avoiding ambiguity.[9]

**Interested in this topic? Let me know by sharing this post!**

*This article was originally published by BRCommunity (link).*

## References

[1] The IIBA's *BABOK Guide: Appendix A – Glossary* defines *business rule* as: a specific, actionable, *testable* directive that is under the control of the business and supports a business policy.
URL: http://www.iiba.org/babok-guide/babok-guide-online/appendix-a-glossary.aspx
[2] From a presentation on the CQAA website. URL: www.cqaa.org/Resources/Documents/Presentations%202010/Writing%20Testable%20Requirements.pdf
[3] From Wikipedia, on "Software testability": http://en.wikipedia.org/wiki/Software_testability
[4] From Wikipedia, on "Requirement": http://en.wikipedia.org/wiki/Requirement
[5] Dr, Daniel M. Berry, *From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity*, (The Ambiguity Handbook).
URL: https://cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf
[6] "Annex A – SBVR Structured English," *Semantics of Business Vocabulary and Business Rules (SBVR)* v1.2, Object Management Group (Nov. 2013). URL: http://www.omg.org/spec/SBVR/1.2/

[7] Business Rule Solutions, *RuleSpeak® Practitioner Kits*, Business Rule Solutions, LLC,

1996-2009.  URL: http://www.RuleSpeak.com
[8]  Business Rule Solutions, *Decision Tables – A Primer: How to Use TableSpeak™*, Business Rule Solutions, LLC. URL: https://www.brsolutions.com/b_ipspeakprimers_toc2.php
[9]  Silvie's website. URL: http://www.silviespreeuwenberg.com/checklist/