Thoughts for the Day from Karl Wiegers

author of *Going It Alone: Essential Tips for the Independent Consultant*

ProcessImpact.com or KarlWiegers.com

1. The two most commonly used requirements elicitation practices are telepathy and clairvoyance. They don't work.
2. No software engineering technique will work if you are dealing with unreasonable people.
3. The customer is *not* always right. But the customer always has a point, and you have to understand and respect that point.
4. When it comes to software quality, you can pay me now, or you can pay me a lot more later.
5. Strive for perfection; settle for excellence.
6. Keep requirements workshops and review teams small. A large group of people cannot agree to leave a burning room, let alone agree on exactly how some requirement should be worded.
7. Why do we never have time to build software right, but we always find the time, money, and people to fix it later?
8. Don't be afraid of documenting requirements. The cost of recording knowledge is small compared to the cost of acquiring that knowledge.
9. When people are asked to work in some different way, their instinctive reaction is to ask, "What's in it for me?" But that's the wrong question. The right question is, "What's in it for *us*?"
10. A fool with a tool is an amplified fool.
11. Nowhere more than in the requirements process do the interests of all the project stakeholders intersect.
12. Everyone likes to think their team has top-flight talent, but half of all software developers are below average in capabilities. Where do all those people work?
13. Don't give someone an estimate off the top of your head. The best response to a request for an estimate is, "Let me get back to you on that after I've thought about it."
14. Without high-quality requirements, stakeholders might be surprised at what is delivered. Software surprises are almost always bad news.
15. The realistic goal is not to create perfect requirements, but to create a set of requirements that is *good enough* to let the team proceed with construction at an acceptable level of risk. The risk is having to do excessive, unplanned rework.
16. No matter how much pressure is applied, never make a commitment you know you cannot fulfill.
17. When exploring requirements, think beyond the hands-on users. Your customer once removed is still your customer.

18. Artificial intelligence is not a substitute for the real thing.
19. If you don’t get the requirements right, it doesn’t matter how well you execute the rest of the project. You will fail.
20. You’re in a stronger negotiating position when you have data available to build your case, because the other person almost certainly has no data at all.
21. If you don’t have a documented and agreed-to project scope, how do you know if scope creep is taking place?
22. Enabling process change is hard when people don’t know about the pain caused by their current ways of working. It’s hard to sell a better mousetrap to people who aren’t aware they have mice.
23. Avoid “decibel prioritization” when deciding which features to include in a product. Loud customers aren’t necessarily demanding the most important features from a business perspective.
24. Unless you record your estimates and compare them to what actually happened, you will forever be guessing, not estimating.
25. In the technology world, if you are one week ahead of the next person, you are a wizard.
26. If some capability or characteristic is not in the requirements, don’t expect to see it in the product.
27. Today’s “gotta get it out right away” development project is tomorrow’s legacy system maintenance nightmare.
28. If you’re hanging out a shingle as a consultant or a contractor, you need to inform the world that you’re available. It doesn’t matter how good you are if no one knows you’re there.
29. An estimate you give someone should not be influenced by what you think they want to hear.
30. The purpose of requirements elicitation is to bring the voice of the customer—the VOC—as close as possible to the ear of the developer—the EOD. The role of the business analyst is to facilitate bridging that communication gap.
31. Q: How many software process leaders does it take to change a light bulb?
    A: Only one, but the light bulb must be willing to change.
32. Many problems with software systems take place at the interfaces: software-to-software, software-to-hardware, software-to-people, people-to-people. Study them carefully.
33. Do not underestimate the need for—and the difficulty of—changing the culture of an organization as it moves toward new ways of working. Installing a new process is faster than instilling a new culture. You need both to succeed.
34. People talk a lot about their “rights.” The flip side of every right is a responsibility. Think and act collaboratively.
35. The first question to ask when someone proposes a new requirement is, “Is this in scope?” If yes, it must be addressed. If no, it does not, at least not now. But if the answer is “No, but it ought to be,” the scope must be adjusted to accommodate it.

This has implications for cost, schedule, resources, commitments, priorities, and tradeoffs.

36. Despite the best of intentions, improvement action plans that don't turn into actions are not useful.
37. The deliverable from requirements development is not just a requirements specification document: it is shared understanding.
38. Never let your boss or your customer talk you into doing a bad job.
39. Common sense, good judgment, and experience should trump formal process in most cases. Sometimes the process is there for a good reason, though. Find out before you decide to bypass it.
40. Quality should be your top priority. Long-term productivity is a natural consequence of high quality.
41. People don't just "gather" requirements. Requirements elicitation is a process of exploration, collaboration, discovery, and invention, not a simple collection process. A business analyst is not just a scribe.
42. An ounce of design is worth a pound of refactoring.
43. Strive to have a peer, rather than a customer, find a defect. Technical peer reviews are a proven technique for improving quality and productivity.
44. When steering an organization toward new ways of working, use gentle pressure, relentlessly applied.
45. Project stakeholders must understand the difference between discussing a possible requirement and committing to include it in a particular release.
46. Watch out for "Management by Businessweek," the rush to adopt the Hottest New Thing in software development just because someone read about the fabulous results it promises.
47. There's no specific activity called "project management." Project management is an amalgam of people management, requirements management, risk management, opportunity management, expectation management, commitment management, change management, resource management, and supplier management.
48. Two terms that should make your blood run cold are "assumed requirements" and "implied requirements." Strive to communicate requirements expectations explicitly.
49. The best motivation for changing how people work is pain. Not artificial, externally-applied pain, but the very real pain the team experiences from its current ways of working. Structure improvement activities so as to reduce the pain.
50. Requirements development demands iteration. You can't get all the requirements right with the first discussion, and you probably can't get them all right ever. Effective requirements development involves the progressive refinement of detail and clarity.
51. Unless you take the time to conduct retrospectives, study lessons learned, and continuously improve your team's process, there's no reason to expect the next project to go any better than the last project.
52. We sometimes express requirements casually because we assume the reader has a "sensibility filter" similar to our own, but people often interpret the same statements

in different ways. This ambiguity leads to mismatched expectations and surprises upon delivery.

53. You can’t change everything at once. Identify those process changes that will yield the greatest benefits, and begin to implement them next Monday. I’m not kidding about that: next Monday!
54. Adopt a “shrink to fit” philosophy with document templates. Begin with a rich template that helps you think of information that should perhaps be included, then mold it to the size and needs of each project.
55. A collection of sticky notes you found on your monitor after lunch, saved voice mail and email messages, and a half-remembered collection of random hallway conversations does not constitute a set of requirements. It’s just a pile of information.
56. Many teams are being asked to do more with less. Too often, though, they are not provided with ways to enable them to do more with less. Without training and process improvement to increase quality and efficiency, you can’t expect higher productivity to magically happen.
57. Hold your thumb and index finger one inch apart. Most of the time that is the only difference between quality and crap. It’s a matter of listening a little bit more, checking your work, running the test, referring to the checklist, reading the directions, asking one more question. Often that’s all it takes to close the crap gap.
58. You don’t have time to make every mistake that every software practitioner before you has already made. Read and respect the literature. Learn from your colleagues. Share your knowledge freely with others.
59. Informal processes that work fine for four people working in a garage do not scale to multiple development teams working on different continents.
60. If there’s one thing the software industry is repeatable at, it’s doing the same silly things on project after project. Use retrospectives to learn, to understand, and to improve continually.
61. Software development is perhaps 50 percent about computing and 50 percent about communication. But business analysis is entirely about communication. We are much better at the computing part.