Six Impossible Things Before Breakfast

Fiona Charles

Originally published in Tea-time with Testers, June, 2011.

> *Alice laughed. "There's no use trying," she said: "one CAN'T believe impossible things."*
>
> *"I daresay you haven't had much practice," said the Queen. "When I was your age, I always did it for half-an-hour a day. Why, sometimes I've believed as many as six impossible things before breakfast..."*[1]

"Aha!" I thought, "Then you must have worked on software projects!"

Waiting for a bus a couple of years ago, I began listing software project lies. The exercise kept me nicely occupied until the bus arrived and through most of the ensuing ride. I ended up with an astonishing thirty categories of lies.

In case you think that sounds like an improbable number, I'll begin by outlining the parameters I set. First and foremost, I had personally to have heard the lie told one or more times. Each lie or category of lies had to be material to a software project, though it could have been told to make a sale before a project began or to describe a project after its end. Each lie had to be relatively common in the industry—or at least not rare. A lie had also to be significant to a project: to have influenced perceptions and/or decisions. And finally, I excluded malicious lies intended to subvert or sabotage an individual on a project, though I have both heard and (on one unhappy project) been the object of some of these kinds of lies.

Reviewing the list as I prepared to write this article, I immediately added a couple more.

The (depressing) reality is that in a career spanning three decades, I cannot recall a single project where there was not at least one significant and material lie. I thought I remembered consulting on one very nice-minded and squeaky-clean project, but then I recalled the programme manager telling a PM, "We're pushing the date of your project out, but it's vital that you do not tell your teams. Everyone needs to believe in the original date so they don't slack off."

Does the number of lies on my list horrify you? Am I exaggerating? Or could it be that we have all heard so many lies so often on software projects that we've become desensitized? I mentioned the "don't tell the team" lie to a notably honest programmer friend, and he said, "Well... we hear that one so often it almost doesn't qualify as a lie."

"Right", I said. "And how is it different from those other oldies but goodies:

> 'We're running 3 weeks behind, but it won't impact the end date.'

and

'We just need to work a couple of weekends to sort out all the quality problems.'"

(A project manager I once worked with used to say, "Show me where on your project plan it says 'A miracle happens here'!")

Perhaps you'll say team lies like this aren't deliberate lies—that the techies or the testers are just being over-optimistic, persuading themselves that the commitments they're making aren't patently impossible. We've all done it, haven't we? And yes, we probably have. But self-deception is still deception. A lie to one's self is no less a lie.

How far really are all those "we're going to make it" team lies from the infamous bait-and-switch scam some consulting firms routinely practice, or the deliberate underbids put forward to make a sale? ("We'll more than make it up in change requests," the sales people say.)

Lying of various kinds is quite common on software projects. Past the sale, the lies often continue with management arbitrarily slashing estimates and imposing upfront commitments to deliver fixed scope with fixed staffing within unachievable timeframes and budgets.

On projects that start out with fraudulent commitments, lies are propagated in the hotbed of fear. Managers demand certainty from people who are often barely in a position to give better than rough estimates, plus or minus fifty percent. Programmers and testers make desperate commitments they know they can't really achieve, and then, week after week, over-optimistically report their progress and status (lie). People lie—or avoid telling the truth, which is pretty much the same thing—to deflect blame and to get management off their backs, hoping to put off well-founded suspicions that they aren't going to deliver the impossible, and anxious to get on with productive work.

How many so-called "troubled" or "failed" projects would actually have gone much over time and over budget if they hadn't started out committed to fiction?

We may think lies like these are symptomatic of big waterfall projects, and indeed that is often true. But Agile projects are not immune to deception. A quick scan of the Agile blogosphere reveals plenty of discussion about impossible project or sprint commitments made by stakeholders outside the project teams or even by the teams themselves.

The rapid feedback built into an Agile process leaves much less room for practitioners to hide impossible estimates or falsify status. But teams calling themselves Agile have been known to play games with the concept of "done", redefining it to meet the actual state of the work when they haven't achieved their goals. Human nature is what it is, on Agile or waterfall projects.

It's a healthy sign, therefore, that two recent books by well-known and respected authors robustly address the topic of dishonesty on software projects and by software practitioners.

The Dark Side of Software Engineering: Evil on Computing Projects,[2] by Johann Rost and Robert L. Glass, is a survey of the bad things people do on software projects, plus some other software-related evils like hacking and computer scams. The range of subjects means the book is a bit of a hodgepodge, but it's a fascinating read. There's

some numerical analysis I found less than compelling, mainly because the samples are too small for the numbers to have real statistical significance. This doesn't detract from the book overall, which has lots of good stories, qualitative analysis and suggested remedies from which we can learn important lessons. Among other benefits, you can learn to spot patterns of certain kinds of nefarious behaviour you may not previously have noticed on projects.

A nice counterpart to *Dark Side* is *The Clean Coder: A Code of Conduct for Professional Programmers*,[3] by Robert C. Martin (Uncle Bob, as he's known in the Agile world.) I doubt anyone will be surprised to hear that Bob Martin comes out strongly against lying on software projects. You may be surprised at how he defines lying, including that it's a lie to say you'll try to meet a date when you already know you cannot achieve it. That advice alone is worth the price of the book. Martin also emphasizes the importance of learning to say "no"—an essential skill for people who want to tell the truth (one that Jerry Weinberg has been promoting and teaching for a long time).

Dark Side and *Clean Coder* are important for software practitioners of all specialities, including testers. The subject of professional ethics is vital for us all. We need to learn to recognize and stamp out lies and other ethical lapses on our projects—not just in other people, but in ourselves. These books will help. It's a bonus that they are also good reads: engagingly written and full of good stories we can relate to.

I don't know whether people on software projects are any more prone to dishonesty than the culture at large. I do know that deception in varying degrees is common on software projects. It's a dirty little secret we don't much explore, although it's an open secret in the business. I'm glad to see that other people are writing about it.

One of my articles on tester and consultant ethics[4] prompted a reader to protest that people can take grave risks telling the truth on software projects, and in a tough economy truth may be a luxury some can't afford. I believe that an expedient lie is the luxury we can't afford. Not for our professional reputations, not for our projects, not for our self respect. I think most testers would agree.

Lying hurts software projects. How many projects have you been on where "everyone knew" the schedule was fiction? Everyone except management, that is, the managers having conveniently forgotten they'd set the project up for failure at the beginning. And perhaps those managers had confidently told senior executives the fake schedule was all certain and wonderful—and now they're running out of budget and running scared. So they put pressure on their teams.

Apart from the cynicism engendered by living a lie, software people do shoddy work under pressure. Designing, coding and testing are all difficult work that requires a clear head. In my experience, the projects where people lie the most produce the worst software.

Lying hurts people too. Every time I present at a conference on "When a Tester is Asked to Lie",[5] one or two people take me aside and say, "This is so timely. It's happening for me right now and I don't know what to do." Others tell me it has already happened to them, and it's a nightmare they don't ever want to repeat. A test manager told me he'd been fired because "we don't think you're comfortable lying to the customer." ("Too right I'm not!", he said to me.)

Yes, it can be risky to tell the truth when others are lying. It can also be unexpectedly rewarding. I have more than once seen an unhappy project benefit from the act of a single tester or programmer bravely stepping forward and saying, "I'm way behind. I'm not going to make the schedule, and I'd like to explain why." Sometimes that can be all that's needed to enable others to speak openly. Though the ensuing discussion might be painful, it could lead to a realistic replanning exercise that puts a project on an achievable path to recovery.

So why don't we just stop lying? We don't have to practice believing ANY impossible things before breakfast. We don't have to convince other people to believe them.

I've loved reading *Alice* most of my life, but I've never taken the White Queen to be a role model. Have you?

What lies have you heard on software projects? Add to my list

I've written mostly in this article about schedule and status lies, but my list of thirty-plus includes many other types.

I've put it on my blog at http://quality-intelligence.blogspot.com/ so you can see the whole list and add your comments. Can you add to the list? Do you have experiences of project lies (or truth-tellings) you'd like to share?

Notes

[1] Lewis Carroll, *Through the Looking-Glass* (Kindle edition), Chapter V.

[2] Johann Rost and Robert L. Glass, *The Dark Side of Software Engineering: Evil on Computing Projects* (IEEE Computer Society, John Wiley and Sons, 2011).

[3] Robert C. Martin, *The Clean Coder: A Code of Conduct for Professional Programmers* (Prentice Hall, 2011).

[4] I've published 4 other articles dealing with the subject of ethics, all on Stickyminds.com. Copies are also on the Publications page of my website, www.quality-intelligence.com. Search for the titles:

Sophie's Choice (September 1, 2007)
Deception and Self-Deception in Software Testing (June 1, 2009)
Negative Positive (February 8, 2010)
No Compromise (June 21, 2010)

[5] Besides the conference presentation "What Price the Truth: When a Tester is Asked to Lie", which deals with a specific type of project lying, I also lead an experiential workshop on the broader topic of "Deception and Self-Deception in Software Testing".