

## Who Lost the Requirements?

A couple of years ago I was parachuted onto a government project to review their requirements. They were getting fed up, having waited two years for the supplier to deliver more than promises.

I went to do a 'gap' analysis. The idea was to find out if the business requirements would be met by the code that was being developed. Unfortunately, the business requirements had been 'misplaced'. On day one there was a lot of talk about the requirements being contained in a complete series of 'Casewise' models on a site in another town. I printed out one of these models, it took up 12 A4 pages. I painstakingly cut the pages so they could be taped together. I poured over the diagram late into the evening, and thought I sort of understood the runes.

In the morning we went down to the business site and passed a very amiable 20 minutes with the Casewise manager. As the meeting was coming to an end, I got the big diagram out and unfolded it on his desk and said "OK, before we go, I was hoping you would just walk me through this diagram so I can assure myself I understand it". He looked at it. Silence. I said, "do you understand this diagram?" He said "no". I said "who does?" He said "well the business expert understands the process, and he sat with the Casewise operator and they made this diagram together." So, the modeller understood the Casewise notation, but not the process, and the domain expert understood the process, but not the model. Great. The Casewise models were useless. Not a great start.

As my time on that project continued I learned a lot of things. Dates came and they went without any release. Once a date had passed, it wasn't mentioned again. On the outside the project team resolutely chose to look on the bright side. The code release remained only days away. I asked to see what had been done; asked to see anything at all. My emails weren't returned; when I went to find the design manager, he was in a meeting. The meeting lasted eight weeks. I came to recognise this as the phenomena of the 'tyranny of optimism'. The tyranny of optimism was fed by the 'tyranny of experts' where overpaid consultants march around hard wood floors in polished shoes producing no visible work and blaming the lack of progress on the incomprehension of mere mortals. Where the coders view themselves as a band of heroes who are the only ones standing between civilisation and the abyss, the project is doomed.

I didn't know it at the time but by the time I got there it was too late. The requirements such as they were, recorded the existing business process without anyone ever bothering to ask 'why do you do it that way?' The existing process had grown up over time, it was full of manual workarounds to accommodate flaky automation, and they were preparing to reintroduce the workarounds in the new system. I worked day and night to rewrite the requirements together with a talented fellow from the business. We shared the results of our work, but it went down like a lead balloon.

The Senior Responsible Owner (SRO) needs to have the confidence to say, "you may understand this, but I don't. Do it again". The customer must own their business requirements and they must understand the delivery process enough to

know when they are being strung along. They need enough courage and authority to say “you promised me Monday, now it’s Wednesday, what’s going on?” They need to know how to tell the difference between a reason, an excuse and a diversion; it’s known as a personal locus of evaluation. The SRO needs to have seen good artifacts to know when they are being presented with poor ones. Sorry Gordon, it isn’t possible for the supplier to sub-contract to perform the duties of the customer. It takes two to tango, no matter how hard one pretends. The dance requires both parties learn the steps.

Sure, this project was well managed; all the project management artifacts were produced, filed and forgotten. When the auditors came in they found nothing amiss. But it didn’t deliver anything. Where there is no connection between the project’s management and the underlying software engineering methodology, it’s not a dance it’s a scrum that has misplaced the ball.