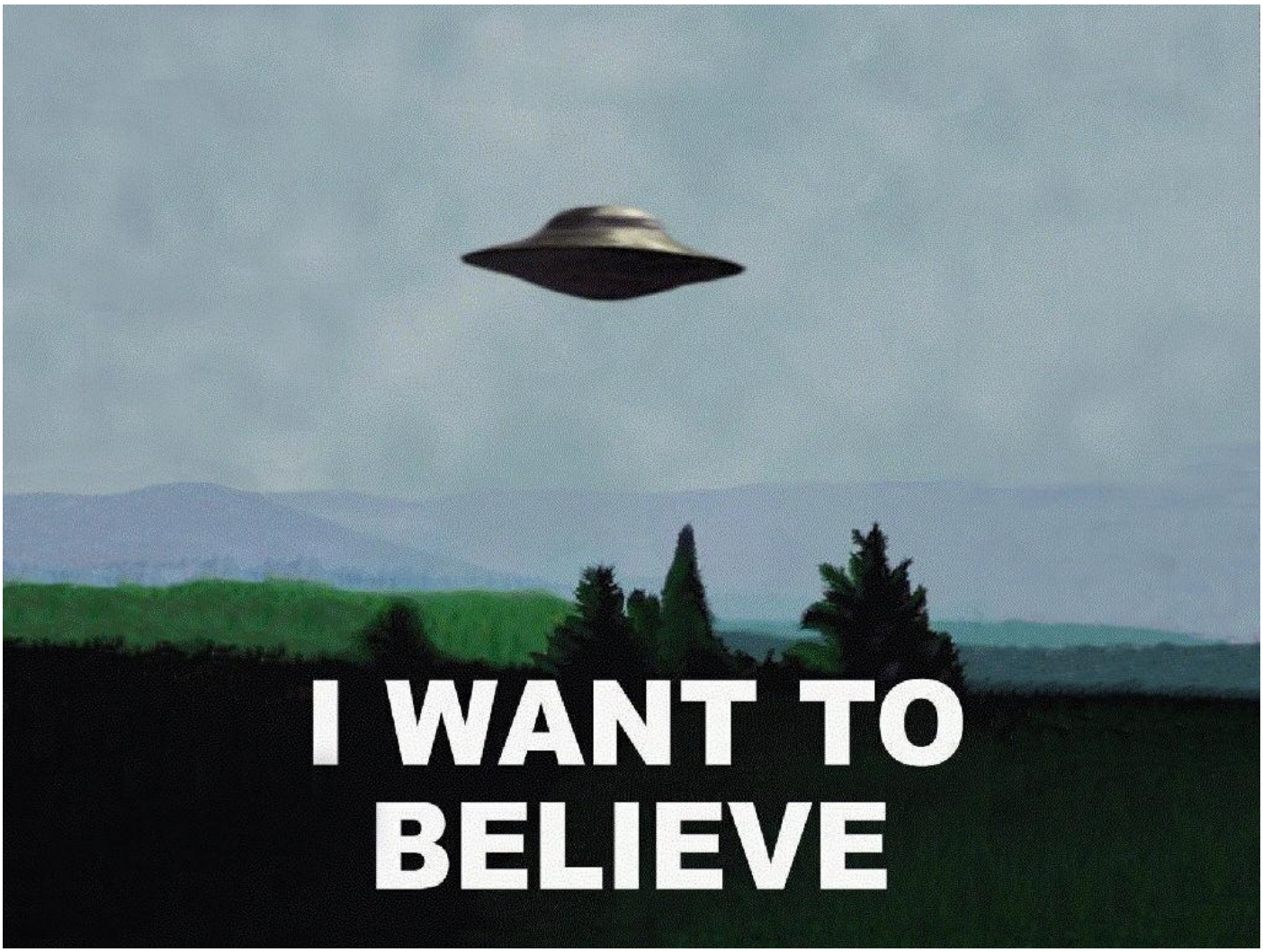
Believing you don't know

11 February 2011 · belief, homeopathy, pre-scripted testing, quack, Scientific Method



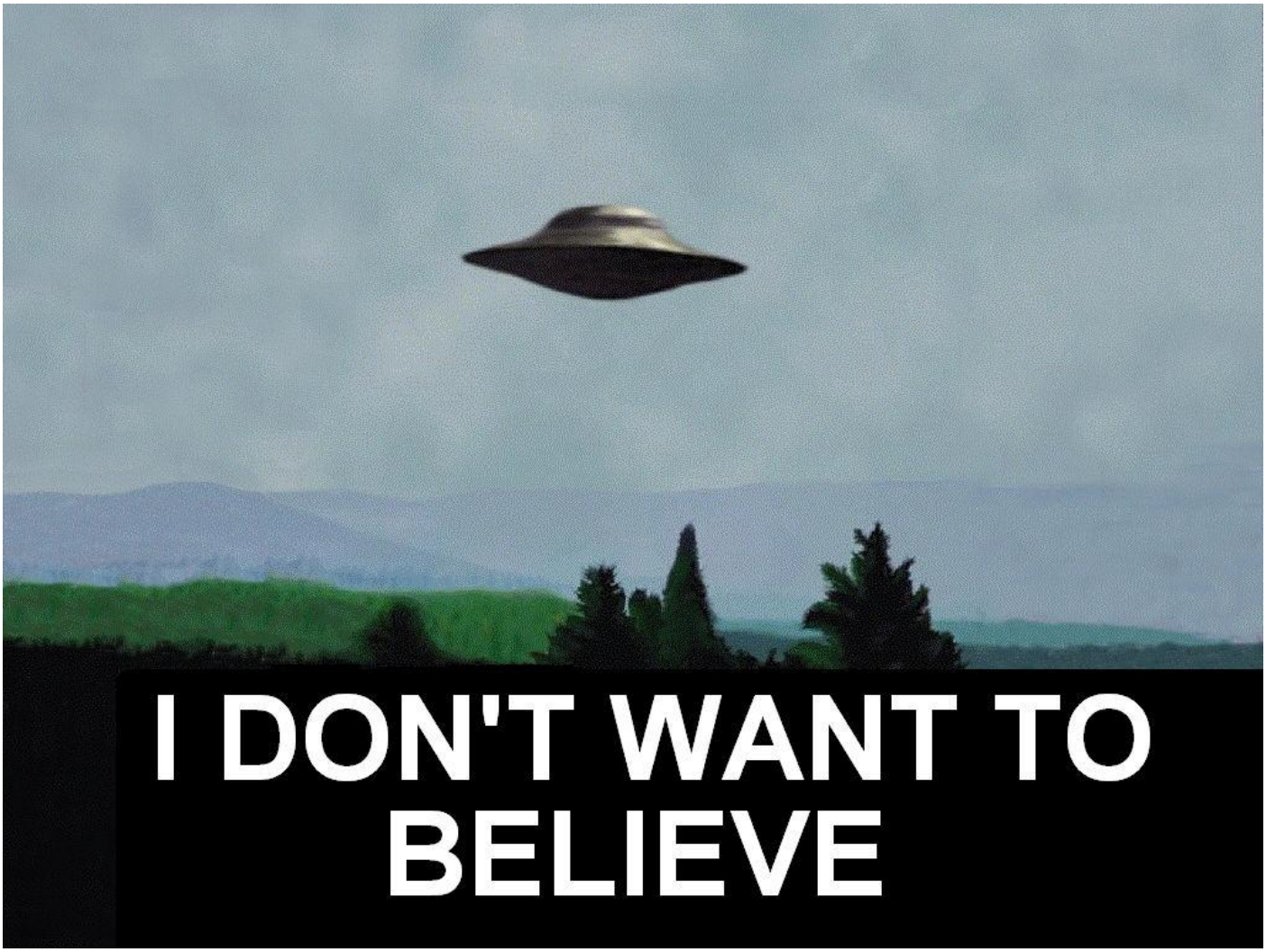
People want to believe. If you are a tester then you've probably seen this in your work. A new build of the software has been delivered. "Just check it works" you're told, It's 'a given' for most people that the software will work. This isn't unusual, it's normal human behaviour. Research has suggested (http://www.spring.org.uk/2009/09/why-you-cant-help-believing-everything-you-read.php) that it's in our nature to believe first. Then secondly, given the opportunity, we might have doubt cast upon those beliefs.

We see this problem everywhere in our daily lives. Quack remedies thrive on our need to believe there is a simple [sugar] pill or even an mp3 (http://www.quackometer.net/blog/2007/08/will-homeopathy-and-itunes-cure-aids.html) file that will solve our medical problems. Software like medicine is meant to solve problems, make our lives or businesses better, healthier. Software

release are often presented to us as a one-build solution to a missing feature or a nasty bug in the code.

As teams, we often under-estimate the ‘unknown’ areas of our work. We frequently under-estimate the time taken to test and fix the features we create. I suspect the more ‘unknowns’ we can think of the less we actually prepare for them. We fail to see the potential link between the idea of an ‘unknown’ and its inherent ambiguity (It’s unknown for a reason - probably the new function is not easy to ‘know’ or understand). As such, many development projects will deliberately avoid planning for the fixing of bugs. Refusing to accept that they can be accounted for until they ‘exist’. Not realising that ambiguity, issues and bugs almost always do get uncovered, and therefore already ‘exist’ before a single line of code is written.

Even disciplined teams will often slip into their ‘belief system’ when the names are changed. For example, A new feature may be tested thoroughly, but a series of major ‘bug-fixes’ to the same system skip through testing with barely a glance. For all we know the programmer checked in the ‘old code’ and the feature is now completely gone! Even if you have an acceptance test in place - every non-acceptance tested execution path maybe broken!



In the software business we are starting to be seen as quacks. We often churn out half baked remedies that don't meet the customers needs. The customer can't even look at the label, and see the possible side effects. The testing has been so cursory and confirmatory that we didn't find any issues (There it is again: Lots of unknowns=no problems). If a doctor gave you a potent medicine, and when you read the label it didn't mention a single possible side-effect, would you really believe it was 100% safe?

The same applies to software... Until we question the hypothesis that it's all going to be OK (http://www.investigatingsoftware.co.uk/2011/01/hunky-dory-hypothesis.html); Until we put our proposed solutions through a 'trial' with testing, then we will remain charlatans. This questioning process can't be a series predefined checks. Much as a medical trial should not be to check that a drug like Thalidomide is -only- a potent antiemetic (http://en.wikipedia.org/wiki/Thalidomide#Other_side_effects), or that an antibiotic like Penicillin was -only- 'good (http://en.wikipedia.org/wiki/Penicillin#Adverse_effects)'. We'd hope they would check the drug for other potential problems, look deeper and learn about it's good and bad attributes, by building and testing hypotheses as they learn.