Dave Nicolette

Effective software development and delivery

All evidence is anecdotal

JANUARY 11, 2012FEBRUARY 28, 2012 ~ DAVE NICOLETTE

It's a commonplace for people to disparage the direct experience of practitioners and to favor the indirect observations of researchers. This has always struck me as a rather strange attitude. A person who does a thing every day, and whose livelihood depends on doing it well, surely knows what has worked and what has not worked well in practice, at least in his/her own experience. Someone who takes the work seriously will naturally remain open-minded about learning new techniques and methods throughout his/her career, building up a deep and reality-based understanding of the work over time. People working in different domains will have had different experiences and will have solved different problems, so if you listen to several experienced individuals representing different industry sectors you will end up with a pretty good cross-section of practical knowledge.

A practitioner who explains what he/she has found useful in the field is giving you an anecdotal report of his/her experience. Is a researcher giving you any better information when he/she publishes the findings of a study? Isn't the researcher giving you, for all practical purposes, an anecdotal report of his/her experiences in preparing a research paper? Both forms of evidence are anecdotal, are they not? Would you really dismiss the opinions of a 35-year veteran of software development in favor of a report prepared by a couple of graduate students who had never written a line of code in anger, and whose main purpose in doing the study had been to learn how to "do a study?" Are the hundred-odd real-world projects the veteran has completed really less informative than the two or three sets of apples-and-oranges observations the graduate students made in their study? Anecdotal evidence is still anecdotal, even after someone publishes it. The question is whose anecdotes carry more weight.

At the Agile 2007 conference in Washington DC, I scheduled my activities so that I could attend some of the talks on the research track. I felt there was too little cross-pollination between practitioners and researchers, and I was curious to know what was happening in research. One presentation I attended reported the results of the study, "A Longitudinal Study of the Use of a Test-Driven Development Practice in Industry" (http://agile2007.agilealliance.org/index.php%3Fpage=sub%252F&id=860.html), conducted by Laurie Williams of North Carolina State University, and Julio Cesar Sanchez and E. Michael Maximilian of IBM. I knew that Dr Williams had run one of the *very* few credible studies of pair programming (http://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF), back in 2000 at the University of Utah, so I had high hopes that this study would provide some useful insights about TDD. The IBMers were professional researchers with good knowledge of the field, and not graduate students who were just learning how to do studies. Max was the presenter.

The researchers noted a correlation between the use of TDD and the control of cyclomatic complexity in the resulting code. This observation is hardly news to anyone who writes software for a living. In our informal discussion after the presentation, I was quite astonished at how pleased the researchers were with themselves; they kept saying they had "discovered" the correlation, and that no one had ever noticed it before. It was a breakthrough! I pointed out that this was already well known to practitioners; it was one of the reasons we choose TDD as a preferred software development practice. If TDD didn't give us results like that, we wouldn't bother to do it. We didn't wait for a study to tell us so; we knew it to be so based on direct experience. It is a result of incremental refactoring, which is done as part of the basic red-green-refactor cycle. Cyclomatic complexity is just one sort of design debt that we manage through the TDD cycle. They said, well, no one has *published* the observation before. They excitedly suggested that I write a paper about it.

I felt genuinely puzzled. The publication of a paper does not cause reality to exist. Reality exists first, and then researchers eventually notice it and write it down. The first one to write something down gets to name it after him/herself. Practitioners don't stop delivering value to customers so that they can write papers. Practitioners don't wait until someone publishes a paper to "prove" the value of a technique before they try it and decide for themselves whether the technique helps them. If they did, then researchers would never have anything to observe. The discussion drove home the fact a wide gulf exists between software development practitioners and the research community.

Another example is the study, "Investigating the Usefulness of Pair-Programming in a Mature Agile Team," carried out by Irina Coman, Alberto Sillitti, and Giancarlo Succi at the Free University of Bozen-Bolzano, Italy, and presented at the XP 2008 conference in Limerick, Ireland. The paper is published in the *Proceedings* of that conference, ISBN 978-3-540-68254-7, Springer-Verlag, pp. 127-136. The study was based on observations of a team of 16 experienced developers at an Italian company over a period of three months. As a way to avoid subjective reporting of when pair programming was taking place, the researchers used a software tool that recorded the times and the focus window at one-second intervals, if someone was actively using the workstation. They discarded the data from 2 of the developers, as they were newcomers who joined the team after the period of study had begun.

There are several problems with this study. Firstly, the researchers only observed one team and only for a limited time. Secondly, the data collection tool could not really tell whether appropriate pair programming practices were in use. Only a qualified human observer would be able to tell whether a pair was engaged in real collaborative work, or if they just happened to be sitting near one another. Thirdly, although the team claimed to be a "mature agile team," they did not in fact use pair programming in a rigorous, disciplined way. They used it "on an as-needed basis." In my experience (anecdotal evidence), this is a euphemism for "we're great programmers and we don't *need* to pair in order to produce good code." This is not the definition of "mature agile team." They actually did pair programming (or whatever passed for pair programming) less than half the time. This raises questions about whether the observations can tell us anything about the relative value of paired work versus solo work. Thirdly, the researchers were not experienced, professional software developers (past or present), able to relate on a deep level to the activities they were studying. They were young graduate students, learning how to conduct studies, how to apply statistical analysis techniques to their observations, how to format and submit a research paper for publication, and how to deliver a talk in front of an audience. Those are all good things to do, of course. They just don't result in a study that business people can depend on to understand the value of the programming technique in question.

These are just two examples among many.

One of the criticisms people express about anecdotal information is that any given report reflects just one set of observations interpreted from just one perspective. A formal study, they argue, compiles information obtained from many sources, many observations, many situations. It normalizes the results, removes statistical outliers, and determines general conclusions that may be drawn based on objective criteria. Really? The pair programming study mentioned above reflects just one set of observations. It is based on observations of a single team. It draws conclusions from just one perspective. It represents the anecdotal report of its three authors. The TDD study mentioned previously also only reports on observations of a single project.

A formal paper published in a respected journal *seems* more impressive than a comment made by a practitioner over beers after work. I have to wonder, though, whether a formal study is any more meaningful than the practitioner's comments, if even *as* meaningful.

It's all anecdotal, when you come right down to it. Whose anecdotes do you trust?

EVIDENCE STUDIES

11 thoughts on "All evidence is anecdotal"

1. **Alex Singh**
   SAYS:
   JANUARY 11, 2012 AT 11:47 AM
   It's actually worse, Dave. You first have to sift through dozens of research papers — papers that often reach different conclusions. For newcomers to the field, this poses a challenge of who to believe. To a practitioner, reading through the reams of equations and technical mumbo-jumbo is a waste of time — essentially nothing new has been said and whatever has been said has been said in a manner that often makes it difficult to understand.

   Reply
2. **Laurent Bossavit**
   SAYS:
   JANUARY 11, 2012 AT 12:43 PM
   FWIW, I disagree with you on the conclusions, even though I agree on the premises, as far as the TDD study is concerned.

   It's not that the anecdotal evidence is good, but more that the research is bad. This is not an isolated case, and most of what passes for research in software engineering is horrible.

   Research in this field has two huge problems, relevance and validity. On relevance, let me just quote the words of one academic who rejected a colleague's proposal for a workshop on TDD: "TDD is of great interest to the software engineering community, but it seems a topic too close to industrial practice to be interesting to a research conference". (Yeah, I WTFed at that too.)

   On validity, things like using students as test subjects keep popping up. The problem is that software engineering must first decide on what kind of insight it wants to give us about the world, not blindly follow the same fads and fashions that sweep the industrial community. Recently "evidence based

software engineering" or "empirical research" seems to have become all the rage, but academics working on these topics are bent on blindly aping medical research, with results that are comical, because practices like TDD are not pills and teams are not bodies or "patients".

We are building on sand: as I show in "Leprechauns" the best-known research resuts in software engineering are patched together from data points that are not comparable with each other, for instance research on "defects" cannot be judged valid to the extent that no two people, let alone two different businesses, will even agree on what counts as a "defect".

There is no such "thing" as a defect, if by "thing" you mean something you could touch or see with your eyes or probe with a beam of electrons. A "defect" is a social convention, a negotiated agreement to perform what we (equally conventionally) choose to call a "fix". The same thing can be a "defect" to one person, a "missing feature" for another, a "feature" to a third.

Software engineering research hasn't even been able to pin down its ontology, the objects it purports to study. No wonder you think all of it is anecdotal. 🙂

But then, "research" is also a social institution, a machine which must do its work and perform its duties of transforming students into doctors, postdocs and tenured professors, award winners and retired dons (except, of course, the large fraction who escape to industry upon getting their PhD).

The entire field is not allowed to grind to a halt just because it's not producing good enough results; the peer review system is not allowed to reject every paper that does not address elementary problems of construct validity such as how "TDD" or "pair programming" or "defects" are characterized, because that would leave publications with nothing to publish.

So the system grinds its wheels, and will continue to do so until either some bright bunch of people manages to energize it with a proper research programme (and I suspect the Agile bunch have the potential to do that, but that potential is still only that for the time being), or it collapses under the weight of its own uselessness.

Meanwhile I recommend reading Kuhn, Lakatos, Latour, and obviously "The Leprechauns of Software Engineering". 🙂

Reply

1. **Dave Nicolette** ☐
   SAYS:
   JANUARY 11, 2012 AT 1:24 PM
   "…the peer review system is not allowed to reject every paper that does not address elementary problems of construct validity … because that would leave publications with nothing to publish."

   I appreciate your comments generally, but I keep mulling over this bit. I'm sure this isn't what you meant, but it sounds as if there's some sort of imperative to keep publications in business, even when they can't tell the difference between valid and invalid content. Why should they not be allowed to fail on the merits?

   In any case, it sounds as if practitioners and researchers can go merrily on their separate ways, pursuing their own completely disconnected goals. No harm done. Either that, or "the system … collapses under the weight of its own uselessness." That works, too.

   Reply

3. **Laurent Bossavit**
   SAYS:
   JANUARY 11, 2012 AT 1:35 PM
   "Why should they not be allowed to fail on the merits?"

   Oh, they *should* all right. But don't expect that they *will*, not if you want your expectations to track reality…

   Reply
4. Pingback: Interesting: Anecdote, Evidence and Research – Agile Advice – Working With Agile Methods (Scrum, OpenAgile, Lean)
5. **Joseph Pelrine**
   SAYS:
   JANUARY 14, 2012 AT 12:37 AM
   Great post, Dave. Very true!

   Reply
6. Pingback: Interesting: Anecdote, Evidence and Research | The Agile Radar
7. Pingback: With the benefit (and bias) of hindsight « Dave Nicolette
8. **Overt**
   SAYS:
   OCTOBER 25, 2012 AT 1:35 PM
   Very good points. As far as medical evidence is concerned for drugs and treatments I find that personal testimonies on internet forums are much more useful than even a doctor's advice. The thing about doctors is that even though they have direct experience they, like the companies that conduct and publish scientific drug trials, have a vested interest in the drugs they prescribe. On top of that patients will often lie to doctors about side effects out of embarrassment or they may lie in order to obtain for of a drug in order to abuse it or sell it. Random people online have virtually no reason to lie about their experience with a certain drug or treatment.

   Reply
9. Pingback: I am NOT a Lab Rat! | ADD . . . and-so-much-more
10. Pingback: scientific evidence vs anecdotal | Answer and Guide