Improving software technology transfer: making decisions based on evidence

Winifred Menezes
Q-Labs, Inc.
6301 Ivy Lane
Suite 210
Greenbelt, MD 20770 USA
+1 301 982 9536
winifred.menezes@q-labs.com

Shari Lawrence Pfleeger
President, Systems/Software, Inc.
4519 Davenport St. NW
Washington, DC 20016-4415 USA
+1 202 244 3740
s.pfleeger@ieee.org

ABSTRACT

The diffusion of innovation in the field software engineering is not well understood. This paper looks at how the legal field treats evidence in an attempt to identify ideas useful to both software engineering researchers and practitioners.

Keywords
Technology transfer, adoption, evidence, empirical software engineering, experiments

1 INTRODUCTION

Much research in software engineering is carried out in universities and research institutes around the world. However the software that reaches the consumer, be it a stand-alone product or embedded software is often developed and maintained with little regard to new processes, method and techniques. At the 1995 International Conference on Software Engineering, David Parnas [1] claimed that neither the papers voted most influential, nor any other papers had any influence on practitioners of software engineering. According to Parnas, the papers' span of influence was limited to the universe of other researchers. There appears to be a communication gap between software engineers in industry and software researchers doing empirical research.

The problems of technology transfer are not specific to software engineering. The business community has studied this topic from various perspectives, addressing some of the interaction between people and technology [2]. Moreover the legal community, has well developed rules for evaluating and combining evidence.

In this paper we have looked at how the legal community treats evidence and applied a similar treatment to empirical software engineering results. We wish to be able to:

- evaluate existing results
- design studies to generate new results
- evaluate collections of results as bodies of evidence in favor of a particular technique

Based on this foundation we ask the following questions:

1. Is it feasible to apply the legal treatment of evidence to software engineering research?
2. Would the application benefit from adaptations? What could these be?
3. Are there other ways of treating evidence that would achieve our stated objectives?

4. Are there evaluation methods that would be well received and easily assimilated by industry?

2 EVIDENCE

Technology adoption is a process that starts with awareness knowledge and progress through various stages, either to the technology being assimilated or rejected. Decisions are made at each stage of the adoption process. These decisions are based on the evidence at hand. Although social science research suggests that experts make decisions [3] in non-analytical ways, the decisions are based on acquired knowledge and perception. The input to and basis of the decision is information that has been communicated to the potential adopter. Schum [4] describes in great detail how lawyers analyze evidence and construct arguments based on the available evidence.

Schum classifies evidence in five categories based on how the users of the evidence stand in relation to it:

- *Tangible evidence* can be examined to determine what it reveals. This category includes objects or things that can be examined, including documentary evidence and demonstrative evidence.
- *Testimonial evidence* comes from another person. Schum distinguishes three cases based on how the person acquired the evidence:
    - direct observation, i.e. she participated in the event in question and observed, via one of her senses, the evidence we now hear;
    - second-hand, i.e. she has obtained information from another source; and
    - opinion, i.e. no primary source of evidence,
- *Equivocal testimonial evidence* where the person providing the evidence either does not remember or expresses the testimony in probabilistic terms.
- *Missing evidence* where expected evidence is either not found or not produced. Missing evidence could be either tangible or testimonial
- *Accepted facts*: information that is accepted without further need for proof, such as physical constants, mathematical formulas, etc.

The credibility of evidence is essential to decision-making. Credibility of tangible evidence can be assessed by examining both its authenticity and accuracy. An object is authentic when it is exactly what it appears or claims to be. In order to extend the range of human observability, devices such as gauges, meters, clocks, or photographs, are used. The accuracy of tangible evidence refers not only to the accuracy of these devices, but also to how the data are presented and interpreted.

The credibility of testimonial evidence is directly related to the credibility of the human source providing the evidence. Schum distinguishes between the actual occurrence of an event, the individual's belief that the event occurred, and the reporting of the occurrence by the individual. Each of these clauses influences an individual's credibility. Thus Schum suggests that an individual credibility can be established by assessing her:

- Veracity or truthfulness. In other words, does she believe what she is telling us? An individual may very well report an event that did not occur because of inaccuracy or unobjectivity.

- Objectivity. In other words, is her perception of events influenced by expectations or some other motivation? The Hawthorne effect is a good example of objectivity problems. Memory-dependent evidence can call into question the individual's objectivity.
- Observational sensitivity. In other words, how accurate was the measuring device? The evidence from our senses is not perfect and can be inconclusive to some degree.

The judicial system has non-statistical ways of assessing a person's credibility, such as previous experience, or by asking the individual a number of relevant questions. An individual's veracity, objectivity and observational sensitivity are entirely context dependent. In law, competence also plays a role. An individual may be perfectly credible (i.e. truthful, objective and have excellent means of sensory perception), but may not be competent in a particular domain. Competence and credibility are two orthogonal attribute of human source of evidence.

Evidence is relevant if it allows us to revise our beliefs about an existing hypothesis or revise the hypothesis itself or generate new hypothesis. Schum distinguishes three degrees of relevance.

- *Conclusive*. Evidence that is directly relevant is either direct or conclusive evidence i.e. we can reach a conclusion based on the evidence, given that the evidence is perfectly credible.
- *Circumstantial*. Even if the evidence is perfectly credible. we may still not be able to draw a conclusion, i.e. the evidence is inconclusive or indirect.
- *Ancillary*. The evidence is relevant only because of its association with other relevant evidence.

Schum creates a theory of evidence based on structure and force. By structure he means the way in which evidential items are linked to the hypothesis, conclusion and to each other. When combining various pieces of evidence, one also needs to consider its strength, force or weight, i.e. how much and in what inferential direction our probabilistic beliefs need to revised based on the new item of evidence.

The legal field uses evidence to determine if a certain event or set of events occurred and to assign responsibility for the event/s. In other words a post-facto "study", with a binary outcome, guilty or not guilty.

Evidence in Empirical Software Engineering

Empirical software engineering involves gathering data about specific aspects of software development or maintenance, analyzing the data, and arriving at conclusions about the relative merits of a particular technology. It is expected that decision-makers in industry will use the evidence and related conclusions to make important decisions about how their software should be developed or maintained in the future.

Zelkowitz *et al.* [5] has developed a comprehensive taxonomy of experiments in software engineering. In this taxonomy, experiments are grouped into three main categories, each of which has several subcategories.

- *Observational methods* collect relevant data as a project progresses. There is relatively little control over the project.
  - *Project monitoring* collects and stores data that are generated during the project. This is a passive method, since there are no experimental goals and no influence or consistency in the data that is collected.

- *Case study* collects specific data focused on the experimental goal.
- *Assertion* the developer performs a simple experiment to justify the technology. There is some question as to whether this method is scientifically acceptable. Nevertheless, it is prevalent in the software engineering field.
- *Survey* or *field study*: This is a cross between the project monitoring and case study. An outside group observes the data and collects the results. The observers have less influence on the project than in the case study.

- *Historical methods* collect and analyze data from projects that have already been completed.
  - *Literature search*: The researcher analyzes the results previously published in order to arrive at a conclusion. A major weakness of this method is selection basis, that is, the tendency of researchers, authors and journal editors to publish only positive results.
  - *Legacy data*: Data from a completed project are analyzed to determine relationships buried in the data..
  - *Lessons-learned data*: This is related to the legacy data method, though less quantitative data are available. For example, post-mortems or project conclusion reports from real projects are analyzed. In some cases, participants from the project are interviewed to further enhance the qualitative data collected.
  - *Static analysis*: Artifacts produced by a project, such as source code, design documents, or test results, are analyzed to determine characteristics of the product.
- *Controlled methods* provide for multiple instances of an observation in order to ensure statistical validity of the results. This is the more classical method of experimental design used in other scientific disciplines.
  - *Replicated experiment*: Several projects are run using the alternative technologies being studied. Control variables are set in order to establish statistical validity. The results may be compromised, since the participants know they are part of an experiment.
  - *Synthetic environments*: A replicated experiment is performed in a smaller, more artificial setting. Because of the artificiality, the applicability of the results to industry is questionable. This method can however be used effectively to test the experimental design.
  - *Dynamic analysis*: This method is used on the software product itself. The product is either modified or run under controlled situations in order to draw some conclusions about the product.
  - *Simulation*: This is similar to the dynamic analysis method, except that the product is run in a model of the real environment. Artificial data are used to make simulations faster, easier and less expensive to run.

Software engineers in industry need to decide if a particular technology would be conducive to achieving a desired outcome. The industrial decision maker must transfer the technology to her/his own environment, as well as estimate the impact the technology will in future projects.

Conclusions

We can map Schum's ontology to Zelkowitz's taxonomy and then draw the following conclusions:

Observational methods create tangible evidence, most of which is documentary. Therefore, the authenticity and accuracy of the documents must be assessed. Publishing the results of a case study, assertion or experiment in a reputed journal would lend it some measure of credibility. Beyond this, software engineering has no way of assessing the authenticity or accuracy of this type of tangible evidence.

Historical methods use evidence, both tangible (in the form of data, reports, code, etc.) and testimonial (through interviews), to create new evidence. Controlled methods also create tangible evidence. Replicated experiments and synthetic experiments create documentary evidence, while dynamic analysis and simulation create demonstrative evidence.

Although there are differences between evidence usage in the legal and software engineering fields, a key problem facing the technology adopter is assessing the credibility of the evidence provided, as well as combining the evidence to reach a conclusion about the technology.

REFERENCES

1. Parnas, D.L. "On ICSE's most influential papers," *ACM SIGSOFT*, 20(3), 1995, pp.29-32. Conference proceedings
2. Pfleeger Shari L. and Winifred Menezes "Marketing Technology to Software Practitioners," IEEE Software January/February 2000, pp 27-33
3. Klein, Gary, Sources of Power: How People Make Decisions, MIT Press, Cambridge, Massachusetts, 1999.
4. Schum, David A., *Evidential Foundations of Probabilistic Reasoning*, Wiley Series in Systems Engineering, John Wiley, New York, 1994.
5. Zelkowitz, Marvin V. and Dolores R. Wallace, "Experimental validation in software engineering", Presented at the Conference of Empirical Assessment & Evaluation in Software Engineering, Keele University, Staffordshire, U.K., 24-26 March 1997.