

Performing and Reviewing Assessments of Contemporary Modularization Approaches: What Constitutes Reasonable Expectations?

Robert J. Walker

Department of Computer Science, University of Calgary, Calgary, Canada
rwalker@cpsc.ucalgary.ca

Abstract

The inherent difficulties in assessing contemporary modularization (CoM) approaches are considered. The motivation is provided for a model relating assessment methodologies to the maturity of the CoM approach.

1. Introduction

Some software engineering research is aimed at the current state-of-the-practice and some is applicable only in the long-term, after many practical issues have been addressed. For example, consider the case of aspect-oriented software development (AOSD) [4], a maturing, contemporary modularization (CoM) approach with the potential to make software easier to develop, change, and reuse. In the mid- to late-1990's AOSD was in its infancy: the nascent AOSD research community did not understand many issues about what constituted an aspect, how aspects and base modularity should and should not interact, how or whether developers would be disoriented by the separation of cross-cutting concerns or not, and so forth. As a result, developing industrial-strength tools—for the development of aspect-oriented programs, or for the use of the aspect-oriented paradigm in other facets of software development (e.g., during requirements engineering)—was rightly viewed as a risky proposition. This led to a conundrum: to truly demonstrate the efficacy of the aspect-oriented paradigm would require industrial deployment of the approach; to deploy the approach industrially would require a major investment in tool and training support; and to justify a major investment in tool and training support would require a demonstration of the efficacy of the aspect-oriented paradigm.

The conundrum was dealt with incrementally: a bit of demonstration, to justify a little investment in tools, to promote some industrial development, to permit slightly stronger demonstration of efficacy, and so forth. There are two key problems with this incremental approach: (1) many reviewers will object to it¹; and (2) it risks failing to notice that key claims have

not (yet) been well-validated. On the other hand, failing to attempt such an incremental approach for a promising technology will lead to stagnation. Notkin states [9], “We need to have a sense of the lifetime of different ideas. Ideas in their early stages need to be given some room to grow, without overly restrictive expectations about their evaluation. More mature ideas need more mature evaluation. ... one size does not fit all.” Briand *et al.* state [1, p. 398], “Each discipline needs to develop its own body of experience and strategies to answer its most pressing research questions.”

2. Formal experimentation?

Some researchers have called on the software engineering community to adopt the methodologies of other fields, such as psychology [2] or the physical sciences [8]. These calls overlook what is different about software engineering research in general, and particularly about software engineering research into novel paradigms like CoM approaches.

The traditional model of formal experimentation calls for controlling all variables but the one that is to be measured, and for randomizing any variables that cannot be controlled. The size and complexity of real-world software and the unpredictability of human developers are clearly difficult issues to cope with in experimenting in any software engineering context. While difficult, various software engineering researchers have successfully performed formal experiments.

To see why such success will not immediately transfer to the assessment of immature CoM approaches, first consider an example of a software engineering technique—code inspection—that happens in industrial practice. If one is interested in how industry actually performs code inspections, one can behave as would a natural scientist, and observe the phenomenon. One needs to worry about how representative the sample (i.e., the particular people being observed, the particular organization, the domain of operation, etc.) under observation is of the population of interest (e.g., all code inspections performed by all organizations in all domains), but careful sampling can deal with this, or expressing the possible limitations on the generaliza-

¹ Most problematically, on the basis that such research is unworthy of publication until the final stages, when tools (at least prototypes) are in the hands of industrial practitioners for evaluation.

bility of the results can lead other researchers to broaden the study to a wider sample. One also needs to be concerned about the possible impact that the act of observation can have on the people being observed (e.g., they will change their behaviour to conform to what their managers expect of them [3]), but the means for coping with such issues exists (such as unobtrusive observations, guarantees of confidentiality, etc.).

The key point about the context of such research is that either the population of interest *does* exist now (i.e., code inspection does occur in some portion of industry) or that a population exists that is a close *model* for the population of ultimate interest (e.g., when we want to know if a small change to inspection techniques results in better inspections, we can provide minor training to some of the people doing inspections now and then compare the results). This is where research into novel CoM approaches faces problems so very different to those of software engineering research like that on code inspections: no population exists that will obviously and immediately serve as a perfect or nearly-perfect model for the population of interest (i.e., strong industrial developers who are used to working with the “novel” approach).

Consider what the effects might be of dropping industrial developers from the 1960s (assuming the availability of a handy time machine) into a modern development environment and context: they would immediately be overwhelmed by the proliferation of graphical user interfaces, the scale of applications, new languages, new tools, and new paradigms. But a misguided attempt at using such people as a model population to assess the value of modern tools, languages, etc. would likely lead us to believe that such tools, languages, etc. were flawed.

We formulate this idea in terms of the following two hypotheses:

H1: *An immature CoM approach must be evaluated against a model population that approximates the ultimately intended, real population.*

H2: *The more immature the CoM approach, the more approximate will be the model population.*

We contend that H1 is a tautology: an immature CoM approach will not have the tools and experts appropriate for real industrial development and, therefore, any assessment cannot be performed on the ultimately intended, real population (i.e., expert industrial developers with real tool support). We expect that H2 will represent the general trend, but that outliers will occur due to the imprecise definition of “the level of maturity”. Nevertheless, the consequences of premature attempts at performing formal experimentation are immediate:

H3: *The more immature the CoM approach, the weaker will be the external validity of the results of a formal experiment on any population that we try to use as a model population.*

Such hypotheses are by no means a complete analysis, let alone validated, but they point towards the sort of theoretical framework that assessment of CoM needs.

3. The Road Ahead

The idea of “rigorous” assessment will immediately appeal to a wide audience; the danger is that “rigour” will be interpreted as something that is impossible to achieve for any less-than-fully-mature approaches.

Many alternatives to formal, quantitative experiments are available. Kitchenham *et al.* [5] recognize the value of “observational studies” in addition to formal experimentation, but emphasize industrial-context, quantitative evaluation, and statistics; their guidelines do not aid a software engineering researcher who attempts to evaluate technology that is not sufficiently developed to be deployed in industry. Seaman [7] promotes the value of qualitative evaluation. Murphy *et al.* [6] suggest that a different treatment is needed for emerging technologies than for more mature ones. What we need to develop are guidelines for treatment of the full spectrum of maturity possible: a research-maturity/assessment-methodology model. A quantitative, formal model would be the ideal, but an iterative approach to developing the model, starting from a rough, cognitive framework, seems best to make progress. Whether the ideal is achievable is to be seen.

Innovation cannot be stifled lest rigour become rigor mortis; creativity cannot deny the value of skepticism. As with much of life, balance is key.

4. References

- [1] L. Briand *et al.* Empirical studies of object-oriented artifacts, methods, and processes. *Empirical Softw. Eng.* 4(4):387–404, 1999.
- [2] A. Brooks. Meta analysis—a silver bullet—for meta-analysts. *Empirical Softw. Eng.* 2(4):333–338, 1997.
- [3] R. Gillespie. *Manufacturing Knowledge: A History of the Hawthorne Experiments*. Cambridge University Press, 1991.
- [4] G. Kiczales *et al.* Aspect-oriented programming. In *Proc. Europ. Conf. Object-Oriented Progr.*, pp. 220–242, 1997.
- [5] B. Kitchenham *et al.* Preliminary guidelines for empirical research in software engineering. *IEEE Trans. Softw. Eng.* 28(8):721–734, 2002.
- [6] G. Murphy *et al.* Evaluating emerging software development techniques. *IEEE Trans. Softw. Eng.* 25(4):438–455, 1999.
- [7] C. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Trans. Softw. Eng.* 25(4):557–572, 1999.
- [8] W. Tichy. Should computer scientists experiment more? *Computer* 31(5):32–40, 1998.
- [9] R. Walker, L. Briand, D. Notkin, C. Seaman, and W. Tichy. Empirical validation—what, why, when, and how. In *Proc. Int’l Conf. Softw. Eng.*, pp. 721–722, 2003.