Some parallels between empirical software engineering and research in human-computer interaction

Judith Segal

j.a.segal@open.ac.uk

Computing Department

Faculty of Mathematics and Computing

The Open University

Walton Hall

Milton Keynes

MK7 6AA

UK

Some parallels between empirical software engineering and research in human-computer interaction

Abstract

Both empirical software engineering and human-computer interaction (HCI) are applied sciences: studies conducted within these disciplines are futile unless they enhance, either directly or indirectly, the practice of software engineering in the former case, and computer support for human endeavours in the latter. The main thesis of this paper is that there should be improved communication between the two disciplines. We argue that a major current concern of researchers in empirical software engineering – that empirical studies do not sufficiently inform practice – and the current emphasis on studies following a traditional scientific experimental design, is very similar to the major concern and methodological emphasis of HCI in the late 1980s/early 1990s. HCI researchers responded to this concern by borrowing tools and techniques from other disciplines, as is currently being advocated by some in the world of empirical software engineering. Although this response has not been unequivocally successful in its aim of closing the gap between studies and practice, we believe that researchers in empirical software engineering might benefit from reflecting on the HCI experience.

Keywords: empirical software engineering, human-computer interaction, field studies

1. Introduction

Given that empirical software engineering and human-computer interaction are both applied sciences, in that studies in these disciplines are futile unless they inform the relevant practice, it is not surprising that they face similar challenges.

The review paper by Perry, Porter & Votta, 2000, identifies the major challenge facing empirical software engineering at the start of the new millennium, to be the closing of the gap between theory and practice; the design of empirical studies which produce results both relevant to, and usable by, practitioners. I argue in section 2 herein that empirical studies which follow the traditional scientific method of testing hypotheses often by the use of experimental and control groups, must be complemented (if not supplanted) by field studies. The HCI community faced the same challenge, and relied on the same traditional scientific method, in the late 1980s/1990s. In section 3, I describe how the community responded to this challenge by importing theoretical frameworks and methods from other disciplines. This response has, according to Rogers, 2001, improved the discourse between designers and clients and made designers more aware of the importance of context in carrying out human endeavours. However, the challenge of reducing the gap between studies and practice remains. This may be due in part to the problems posed in conducting and interpreting field studies, as I discuss in section 4. In section 5, I conclude with the plea that there be more interaction between the two communities of empirical software engineering and human-computer interaction.

2. Empirical studies of software engineering

The main argument of this section can be summarised by the following quote from Ball & Ormerod, 2000. These writers were concerned with designers, but if we replace the terms 'design', 'design activity ' and 'design expertise' by the term 'software engineering', I maintain that the quote still makes perfect sense.

> ‘Given the highly contextualized nature of design activity, it remains paradoxical that the majority of existing studies of design expertise have ignored the role of situational and social factors in design in preference to carrying out laboratory-style investigation in which such factors are controlled for’ [Ball & Ormerod, p.148]

I shall begin by reflecting on two recently published papers on current issues in empirical software engineering. The first, Perry, Porter & Votta, 2000, reviews the state of the subject at the turn of the millennium, and the second, Kitchenham, Pfleeger et al., 2002, proposes a set of guidelines for the conduct of empirical studies in software engineering. In my opinion, Perry et al. take a very traditional view of the discipline, whereas Kitchenham et al., while still quite traditional, are aware of some of the pitfalls. I then describe some moves to break away from tradition.

2.1. Some problems with the traditional view

Perry et al. take a very traditional view in my opinion, that is, they appear to view empirical studies of software engineering in the same light as traditional scientific experiments whereby hypotheses are usually tested by means of comparing experimental and control groups. There seems to me to be a fundamental flaw with this view, which is that it ignores the fact that software engineering is essentially a human activity. The human element is usually discounted from (say) a physics experiment: every action has an equal and opposite reaction independently of who is making the observation, or the organisational, social and cultural background of the observer. This simply isn’t true in software engineering: cultural, organisational and social issues effect the way that software is developed. As Bannon is quoted as saying in Sim et al., 2001,

> '[In empirical software engineering] Social issues relating to work practices, group dynamics, discourse between group members, and political issues among group members are all very important and should not be ignored' [p.89].

The plethora of organisational, social and cultural factors which have the potential to impact on software engineering renders it very difficult, in my opinion, to conduct traditional scientific experiments in this context. There are difficulties in teasing out variables so that the independent variables can be clearly distinguished and one manipulated independently of the others; in setting up control groups, and in dealing with qualitative factors.

Kitchenham et al. note that teasing out variables is difficult, even when organisational, social and cultural factors are ignored:

Some parallels between empirical software engineering and research in human-computer interaction

> 'In software engineering, many of our outcome measures (defect rates, productivity, lead time) are related to each other. Thus, if our main interest is in whether a development method decreases lead time, it may still be important to investigate whether it has adverse or beneficial effects on productivity or defect rates' [p.728]

and that setting up control groups can pose problems:

> 'For software experiments, controls are difficult to define because software engineering tasks depend on human skills. Usually, there is no common default technology… for performing a task against which a new method, procedure, or tool can be compared….. At best, our empirical studies within a particular company can use its standard or current practice as the basis for evaluating a new technology…. Thus, studies of a new technology in different companies or organizations are difficult to compare because it is highly unlikely that the baselines are the same' [p.727]

In addition, there is the problem that organisational, social and cultural factors do not always lend themselves to the quantitative measures required by traditional scientific studies. This might have the effect that qualitative factors, however important, may be ignored in favour of quantitative. Kitchenham et al. are aware of this:

> 'Many inspection experiments have concentrated on assessing easily quantifiable benefits and ignored other type of benefits.' [p.723. This quote is in the context of comparing quantitative studies, which suggested that inspection meetings are not necessary to maximise defect detection, with an observational study, which identified a number of qualitative benefits of such meetings, such as on-job training and the promotion of teamwork]

Bannon, as quoted in Sim et al., 2001, puts the point more forcefully. He warns of the following 'McNamara Fallacy' :

> 'The first step is to measure whatever can be easily measured. This is OK as far as it goes
> The second step is to disregard that which can't be easily measured or to give it an arbitrary quantitative value. This is artificial and misleading.
> The third step is to presume that what can't be measured easily really isn't important. This is blindness.
> The fourth step is to say that what can't be easily measured really doesn't exist. This is suicide.' [pp 89-90]

I have argued above that traditional scientific experiments are difficult to execute in the context of software engineering because of the human (organisational, social, cultural, psychological) factors involved. I will now argue that, even when executed perfectly, traditional scientific experiments do not, of themselves, address the main concern of Perry et al., that studies fail to sufficiently inform practice.

Consider the insistence of these writers on the importance of a priori hypotheses in empirical studies. For example, in their section 3 they say

'A more fundamental aspect of this problem [that too many papers rely solely on the fact that they have lots of data] is that too many empirical studies simply lack hypotheses'

and in their section 4:

‘Our studies (no matter how they are done) should always have a hypothesis. With every study we must define what we are comparing and why’ .

This seems to me to beg a fundamental question. How does one know what the hypothesis should be? How does one know what are the important research questions to ask; the questions to which the associated answers will have the most impact on software engineering? I would argue that answers to these meta-level questions might come from empirical studies, but empirical studies *without* a priori hypotheses, that is, field studies which take full cognisance of organisational, cultural and social factors and use some sort of ethnographic method in order to study what software engineers actually do as they develop software, and where their problems actually lie.

Kitchenham et al. are aware that it is the salience, rather than the existence, of a hypothesis that matters:

'… we do not want to do research for the sake of research' [p.732]

'.. all too often the so-called hypothesis is simply a statement of the tests to be performed. For example, … "there is no correlation between cyclomatic number and faults found". We call this statement a shallow hypothesis because it does not reflect an underlying, explanatory theory. That is, whatever the result of the experiment, we will not be increasing our software engineering knowledge in any significant way.' [p.724].

But where does the initial explanatory theory come from? In my opinion, the answer to this question is given insufficient weight in Kitchenham et al.'s paper. My view is that here, again, is where field studies using some sort of ethnographic method, might be of use.

2.2. Multidisciplinary approaches to empirical software research.

Mine is not the only voice crying in the wilderness that the traditional scientific experimental approach is not always appropriate for empirical software engineering, and that field studies might, in certain circumstances at least, be of more relevance. Harrison et al., 1999, in the course of a SWOT (Strengths, Weaknesses, Opportunities, Threats) analysis of empirical software engineering, assert that

‘The discipline is intrinsically multi-disciplinary as solutions to real-world problems are necessarily holistic and complex’ [p.408]

and

'The opportunity to evaluate systems from multiple viewpoints, using different techniques together with both qualitative and quantitative data should be grasped whenever it is available' [p. 407]

Some parallels between empirical software engineering and research in human-computer interaction

In 2000, ICSE included a workshop on multidisciplinary approaches to empirical software engineering, as reported in Sim et al., 2001. This workshop was based on the premise that empirical software engineering should be able to 'beg, borrow or steal' from other disciplines with long traditions of empirical research. Three themes emerged from the workshop: the importance of education in empirical software research; the need to understand fully techniques from other disciplines before attempting to apply them, and the role of evidence. It is this latter theme which I think is of most relevance to this paper. Petre is quoted in Sim et al. as saying that the important point is to determine the research question, then to determine the type of evidence that will answer the question, and finally, to determine the research method which will deliver the right sort of evidence.

We noted above the concern of both Bannon and Kitchenham et al. that qualitative data might be ignored in favour of quantitative. Petre goes further than this:

> 'Qualitative knowing is absolutely essential as a prerequisite foundation for quantification in science' [p.88]

That is, quantitative data can only be interpreted in the light of qualitative understanding of a domain.

Another salient comment made in this workshop concerns the potential difficulty of gaining access to suitable field study sites, a concern echoed by Harrison et al., op. cit.

I now turn my attention to some relevant research in HCI. The major problem identified by Perry et al., that empirical software engineering does not seem sufficiently to inform software engineering practice, has a parallel in HCI, and I shall now discuss this parallel and how the HCI community has attempted to address it.

3. Some parallel issues in HCI research

HCI research, like empirical studies of software engineering, is fundamentally applied. It is futile unless it has the potential of improving the way computers can support human work, just as empirical software engineering is futile unless it can inform software engineering practice.

In the late 1980s/early 1990s, there was much concern expressed in the HCI community, paralleling the current concern of the software engineering community, that empirical studies did not sufficiently inform practice, see, for example, Rogers, 2001, and Kuutti, 1996. In the 1980s, many of these studies were couched within the then dominant theoretical framework for HCI, cognitive psychology. This framework focuses on the individual and lends itself to studies in the classical scientific experimental paradigm, with control and experimental groups. As the following quote shows, however, results from such experiments often did not translate into the 'real world':

> 'In basic research settings, behaviour is controlled in a laboratory in an attempt to determine the effects of singled out cognitive processes. The processes are studied in isolation and subjects … are asked to

> perform a specific task, without distractions or aids at hand. In contrast, the cognition that happens during human-computer interaction is much more 'messy', whereby many interdependent processes are involved for any given activity. Moreover, in their everyday and work settings, people rarely perform a task in isolation. Instead, they are constantly interrupted or interrupt their own activities… The stark differences between a controlled lab setting and the messy real world setting, meant that many of the theories derived from the former were not applicable to the latter' [Rogers, section 2]

The response of the HCI community to this concern was to look outside the focus on an individual and cognitive psychology to a wider focus and a plethora of other theoretical frameworks, such as extensions of cognitive psychology to a consideration of full socio-technical systems (distributed cognition) or to the effect of external representations on cognitive tasks (external cognition/ interactivity theory); ecological psychology; social psychology (activity theory), and sociology and anthropology (situated action and ethnomethodology). The dominance of laboratory studies began to be challenged by field studies.

Rogers, op. cit., assessed the extent to which each of these theoretical frameworks have informed the practice of interface and software designers. She argues that these new approaches have, in general, enriched the language of design, by, for example, the use of concepts such as affordance, originating in ecological psychology, and cognitive dimensions, see for example, Blackwell et al., 2001. They have also made designers more aware of the importance of the work context. For example, it is clear that research in the fields of situated action and ethnomethodology have influenced user-centered design methodologies such as contextual design (Beyer & Holtzblatt, 1998). On the whole, however, Roger's conclusions are disappointing: the road from a theoretical framework to information which is directly useful to a designer, remains rocky and tortuous. In practice, many designers rely on ad-hoc empirical methods, for example, investigating which of two features might be preferred by a potential user of a system. This has the drawbacks that, without an underlying theory, what is tested might be quite arbitrary (why two features in the example above? Why those particular two features?) and the results of the testing may not be generalisable.

I now turn my attention to the conduct of field studies. Although traditional scientific studies may not sufficiently inform our knowledge of the 'real world', we do (theoretically) know how to conduct them. The situation with field studies appears to be reversed, as we shall now discuss.

4. Some methodological problems with field studies

I noted above that the erstwhile dominance of laboratory studies in HCI is being challenged by field studies. There is widespread acknowledgement of the necessity for field studies before constructing tools or methods to support work, see, for example, Beyer & Holtzblatt, 1998, Lethbridge & Singer, 1998, Luff, Hindmarsh & Heath, 2001, and Ball & Ormerod, 2000.

Field studies are not without problems. As I noted in 2.2. above, there is a potential problem with gaining access to a suitable field site. There is also the issue of how field studies should be conducted. Ball & Ormerod, op. cit., argued that pure ethnographic studies are inappropriate in the context in which the field studies are precursors to some intervention, for example, to the introduction of some new tool or process to support work. Pure ethnographic studies are slow and expensive, demanding long term immersion of the observer in the observed culture; may result in the observer drowning in data, as s/he is supposed to have no a priori plan of the type of data to collect; and, perhaps most importantly of all in the context of this paper, to be so situated in a particular historical, cultural and organisational context as to be ungeneralisable. Ball & Ormerod addressed these issues by modifying pure ethnography into a methodology they call 'cognitive ethnography', in which specific data is collected with some purpose in mind (to inform the construction of software to support reuse, in their case), and the interpretation of the data is validated by the application of different analytic methods to the same data.

The potential lack of generality is, perhaps, the greatest problem with those field studies intended to enrich our knowledge of software engineering/HCI, as opposed to simply informing some intervention in a particular context. Kitchenham et al. assert that the context of a field study has to be fully described so that the range of applicability of its results might be apparent. There are, however, two main problems with this: how the context might be delineated – what factors might be relevant, and what not – and, once delineated, how described. Delineation is difficult. In a field study I conducted of research scientists constructing analytical instruments and developing the concomitant software, the scientists preferred photos and videos of the stages of the construction, rather than notes taken at the time, since it wasn't clear a priori which contextual factors would be important ('this component failed subsequently because when it was being constructed, it just happened to be near a window with the sun beating down '). Description of a context in a form which is meaningful to other researchers and to practitioners, is especially difficult in domains such as software engineering and HCI, where even the terms specific to the domain don't have a precise, generally accepted, meaning. Kitchenham et al. cite 'lines of code' as a term in software engineering, which though apparently very precise, is, in fact, ambiguous. An example in HCI is the term 'novice'. When we are talking about novices, as opposed to experts, do we mean people with little experience of computing? or with little experience of a particular system or programming language? or merely first year undergraduates, as opposed to second years? Nardi, 1996, addresses this concern with vocabulary by suggesting that all researchers and practitioners adopt her preferred theoretical framework (activity theory), but this seems rather dogmatic. I should suggest, rather, that publishers of field (and any other) studies be very aware of the ambiguities of the language they use, and explicate fully any potentially ambiguous term.

5. Summary and conclusion

Some parallels between empirical software engineering and research in human-computer interaction

In this paper, I have drawn parallels between the current state of empirical software engineering, with its concern for the gap between studies and practice and its emphasis on the traditional scientific method, and the state of HCI in the late 1980s/early 1990s. Although I have argued that field studies have an essential place in empirical software engineering, for example, in order to ascertain exactly how software engineers work and how this work might best be supported, such studies are not without problems. For example, the richness of context of field studies makes it difficult to compare or to aggregate them or to transfer their findings to other contexts. Delineating and then describing contexts are not easy tasks, as I have discussed above. These methodological problems may go some way towards explaining the limited impact that such studies have had on the practice of HCI, as described in Rogers, op.cit.

There are clearly no easy answers. But given that the disciplines of empirical software engineering and human-computer interaction are both about improving human endeavour, and given that they face similar methodological problems and the same major challenge of transforming the results of academic studies into information that practitioners can gainfully use, may I make a plea for more interaction between the two communities?

References

Ball LJ & Ormerod TC, 2000, 'Putting ethnography to work: the case for a cognitive ethnography of design', Int. J. Human-Computer Studies, 53, 147-168.

Beyer H & Holtzblatt K, 1998, Contextual Design. Defining Customer-Centered Systems, Morgan Kaufmann

Blackwell AF, Britton C, Cox A, Green TRG, Gurr CA, Kadoda GF, Kutar M, Loomes M, Nehaniv CL, Petre M, Roast C, Roes C, Wong A & Young RM, 2001, 'Cognitive Dimensions of Notations: Design tools for cognitive technology'. In M. Beynon, C.L. Nehaniv, and K. Dautenhahn (Eds.) Cognitive Technology 2001 (LNAI 2117). Springer-Verlag, pp. 325-341.

Harrison R, Badoo N, Barry E, Biffle S, Parra A, Winter B, Wuest J, 1999, 'Directions and methodologies for empirical software engineering research', Empirical Software Engineering, 4(4), 405-410

Kuutti K, 1996, 'Activity Theory as a potential framework for human-computer interaction research' in Context and Consciousness: Activity Theory and Human Computer Interaction, B. Nardi (ed), MIT Press: Cambridge. 17-44.

Kitchenham BA, Pfleeger SL, Pickard LM, Jones PW, Hoaglin DC, El Eman, K, Rosenberg J, 2002, 'Preliminary Guidelines for Empirical Research in Software Engineering', IEEE Transactions on Software Engineering, 28(8), 721-734.

Lethbridge T, Singer J, 1998, 'Studying work practices to assist tool design in software engineering', Proceedings of IEEE 6th International Workshop on Program Comprehension, 173-179.

Luff P, Hindmarsh J, Heath C (eds.), 2001, Workplace Studies: recovering work practice and informing system design, Cambridge University Press, 150-166.

Nardi B, 1996, 'Activity Theory and human-computer interaction' in Context and Consciousness: Activity Theory and Human Computer Interaction, B. Nardi (ed), MIT Press: Cambridge.

Perry DE, Porter AA, Votta LG, 2000, 'Empirical studies of software engineering: a roadmap', in Proceedings of the International Conference on The Future of Software Engineering, A. Finkelstein (ed.), ACM Press, 345-355

Rogers Y, 2001, 'Knowledge transfer in a rapidly changing field: what can new theoretical approaches teach HCI?' available from http://www.cogs.susx.ac.uk/users/yvonner/papers/YR_theory.pdf, accessed 2/1/03.

Sim SE, Singer J & Storey M-A, 2001, 'Beg, Borrow or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research', Empirical Software Engineering, 6, 85-93