

On Establishing the Essential Components of a Technology-dependent Framework: A Strawman Framework for Industrial Case Study-Based Research

Laurie Williams¹, Lucas Layman¹, Pekka Abrahamsson²

¹North Carolina State University, Department of Computer Science
{lawilli3, lmlayma2}@ncsu.edu

²VTT Technical Research Centre of Finland, pekka.abrahamsson@vtt.fi

ABSTRACT

A goal of evidence-based software engineering is to provide a means by which industry practitioners can make rational decisions about technology adoption. When a technology is mature enough for potential widespread use, practitioners find empirical evidence most compelling when the study has taken place in a live, industrial situation in an environment comparable to their own. However, empirical software engineering is in need of guidelines and standards to direct industrial case studies so that the results of this research are valuable and can be combined into an evidentiary base. In this paper, we present a high-level view of a measurement framework that has been used with multiple agile software development industrial case studies. We propose that this technology-dependent framework can be used as a strawman for a guideline of data collection, analysis, and reporting of industrial case studies. Our goal in offering the framework as a strawman is to solicit input from the community on a guideline for the essential components of a technology-dependent framework for industrial case study research.

1. INTRODUCTION

The first speaker [29] at the Empirical Assessment of Software Engineering (EASE) 2004 conference presented an industrial case study of a small, co-located team that used Extreme Programming (XP) [2]. In this study, five metrics were collected. The second speaker [31] also presented an industrial case study of a small, co-located team that used XP. A larger set of metrics was collected in this case study. However, the intersection of the metrics presented by the first and by the second speaker contained no metrics. Such a scenario is commonplace in software engineering research today.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

REBSE'05, May 17, 2005, St. Louis, Missouri, USA.
Copyright 2005 ACM 1-59593-121-X/05/0005...\$5.00

As will be discussed, the inclusion of industrial case studies in evidence-based software engineering (EBSE) is not without its challenges [15]. However, as software engineering researchers we lose the opportunity to accumulate evidence from our research if we do not establish technology-dependent frameworks for metrics collection, analysis, and reporting. In our experience, as will be discussed, generic frameworks are not feasible; technology-dependent frameworks are needed to capture important elements of the technology under study.

Two of the primary goals of EBSE are (1) to provide a common goal for individual researchers and research groups to ensure their research is directed to the requirements of industry and other stakeholder groups and (2) to provide a means by which industry practitioners can make rational decisions about technology adoption. Over a decade ago, Colin Potts [26] questioned why software engineering research was failing to influence industrial practice. He posited that the dominant sequential “research-then-transfer” mode of research can cause a failure to address significant practical problems. Instead, he proposed an “industry-as-laboratory” mode of research in which researchers are closely involved in industrial projects.

Five years after Potts’ challenge, Zelkowitz also commented on the lack of influence that software engineering research results have on industry. His survey of 90 software engineering researchers and practitioners [34] revealed that industry is influenced by compelling evidence on the effectiveness of a technique in live situations in an environment such as their own. Similarly, Pfleeger asserts that the audience for our evidence should help us decide what kinds of studies to perform and in what contexts [25]. Furthermore, Mathiassen [23] advocates research as a means for establishing a collaborative relationship between the practitioner and research communities. Results of sound industrial case studies are important for achieving the goals of EBSE because laboratory experiments are not considered to provide compelling evidence [5]. However, empirical software engineering is in need of guidelines and standards to direct industrial

case studies so that the results of this research are valuable and can be combined into an evidentiary base.

Our goal is to solicit input from the community on guidelines for the essential components of a technology-dependent framework for industrial case study research. In this paper, we present a high-level view of a measurement framework, the Extreme Programming Evaluation Framework (XP-EF) [32], that has been used with three agile software development industrial case studies [20, 21, 31]. *We propose that this technology-dependent framework can be used as a strawman for a guideline for data collection, analysis, and reporting of industrial case studies.* With such a guideline, groups of researchers interested in the same research question(s) can customize a common framework, allowing for their results to be combined. Each case study might have small evidential force, but together a family of related case studies can demonstrate a point more strongly [1, 24], particularly if a common framework is used.

We offer as a strawman the XP-EF framework. The initial version of the XP-EF was based upon the examination of software engineering literature, such as [9, 14, 18, 19]. The XP-EF was then refined through our experiences of multiple, longitudinal case studies. We have also revised the framework based upon input from anonymous reviews of our publications (accepted and otherwise), from researchers in the community, and through discussions with members of the International Software Engineering Research Network (ISERN)¹. Additionally, numerous presentations have been made at conferences and industrial sites. Through this peer review process, the research and practitioner communities have already provided input on the components of the framework. Through additional input from the EBSE community, we feel that the essential elements of industrial case studies measurement frameworks in the general case can be extracted from this exemplar framework.

The remainder of this paper is as organized as follows. Section 2 discusses the importance of technology-dependent frameworks for industrial case study research. Section 3 presents a high-level view of the strawman framework, the XP-EF. In Section 4, we present our experiences using and modifying the XP-EF for other software development methodologies. We conclude in Section 5.

2. IMPORTANCE OF TECHNOLOGY-DEPENDENT MEASUREMENT FRAMEWORKS

Case studies (or $N=1$ experiments [10]) can be viewed as “research in the typical” [8, 17] and are an evaluative technique carried out with realistic tasks and realistic subjects in a realistic environment [7, 28]. As such, case study research avoids the limited relevance of small-scale software engineering experiments. However, case studies cannot be performed with rigor of such experiments. Consumers of case-study based research must realize that the results must be appropriately qualified by the researchers and not taken to be

valid in the general sense. Additionally, it may be difficult to disentangle the research results from the particular context in which the case study took place. Finally, even a family of case studies is not likely to yield statistically significant results, though Harrison [11] observes that practitioners are not necessarily motivated for technology transitions solely due to the presence of statistically-significant research results in support of a technology

Measurement frameworks have been published in software engineering [9, 14, 19, 32]. However, except for the XP-EF [32] no actual use of the framework was found in the literature. Kitchenham et al. [15] contend that more guidelines and protocols are needed for EBSE. Rigor is important in any research that purports to be relevant [5] Selecting the proper measurements for a framework is central to ensuring construct validity. A common concern with case studies is that they can lack rigor and the use of systematic procedures [33]. The use of detailed instructions and a metric suite defined in a framework can dissuade this concern as well as can reduce experimenter and subject bias due to pre-defined measures. The metrics defined in the framework should be theoretically defensible and based primarily upon validated metrics [16, 27]. Metrics which are not validated should be validated as part of the case study research.

The development of a framework enables the replication of empirical studies and the meta-analysis of the results. Framework artifacts are meant to be used in different laboratories and in different industrial locations.

An isolated study remains virtually meaningless and useless in itself. . . If a study is worth doing at all, it's worth doing twice. [22]

An important requirement for any empirical study is repeatability. Repeatability ensures that the results can be checked independently and addresses threats to experimental validity [1]. Furthermore, researchers become more confident in a theory when similar results emerge in different contexts [17, 30]. Alternately, the inability to replicate can serve to refute prior findings or to limit the context in which prior findings are relevant. Replicating studies is a means for establishing the range of different conditions under which the findings hold [22].

Finally, the development of a framework also drives increased collaboration between researchers. Researchers work together to build consensus on what should be in the framework and to compare research results in a wider research agenda than their own single case study.

3. STRAWMAN: XP-EF

To structure a family of XP case studies, we created the XP-EF [32], a framework for expressing (1) the context of the case study; (2) the extent to which an organization has adopted and/or modified XP practices; and (3) the business-related results of this adoption. We offer the XP-EF as a strawman for examination by the EBSE community. Through this examination, we solicit input from the community on a guideline for the essential components of a technology-dependent framework for industrial case study research. For the XP-EF, the technology is the XP software development methodology. Space prevents a full enumera-

¹ <http://www.iese.fhg.de/ISERN/>

tion of the XP-EF in this position paper. A more detailed discussion of the XP-EF, its creation, rationale, and shortcomings may be found in [31]. Instructions and templates for measuring and reporting an XP case study via XP-EF Version 1.4 have been documented in [32].

Via collaborative research [23], the framework was created by a team of researchers and one practitioner. The XP-EF is composed of three parts: XP Context Factors (XP-cf); XP Adherence Metrics (XP-am); and XP Outcome Measures (XP-om). Each of these parts is now discussed.

3.1 Context factors (XP-cf)

Recording context factors is essential for fully understanding the generality and utility of the case study findings as well as the similarities and differences between the case study and one’s own environment. Through the XP-cf, practitioners can consider, “*Is this team like my team?*” and researchers are enabled to combine results.

In the XP-EF, researchers and practitioners record essential context information about their project via the XP-cf. Software engineering has no well-defined standards for determining what contextual information should be recorded [18]. Jones [14] states that software projects can be influenced by as many as 250 different factors, but that most projects are affected by 10-20 major issues. Based on Jones’ organization of key context factors for software development, we organized the XP-cf into his six categories: software classification, sociological, project-specific, ergonomic, technological, and geographical. We also include developmental factors that use a risk-driven approach [4] to determine whether a project would be most successful using an agile or plan-driven approach.

In total, there are over 50 context factors recorded in the XP-cf. The majority of these are easy to record, such as team size, geographical dispersion, and the name of the development process. Others are more time consuming to gather, such as lines of code and number of test cases. Such a large number of context factors could lead to a combinatorial explosion of information [5]. However, we focus our comparison on five context factors outlined by Boehm and Turner [4] as those they believe particularly relate to the suitability of the use of agile practices for that particular team. These five factors are as follows:

- *Size*. The number of the people on the team.
- *Criticality*. The potential impact of a software defect in terms of comfort, money, and/or lives.
- *Dynamism*. The degree of requirements and technology change.
- *Personnel*. Skill level and experience of team.
- *Culture*. Whether the individuals on the team prefer predictability/order or change.

Boehm and Turner graphically display a team’s five critical values for the context factors on a polar chart, as shown in Figure 1. Each of the axes is labeled with carefully-chosen values based on the Boehm and Turner’s history of working with industrial projects. When a project’s data points for each factor are joined, shapes distinctly toward the graph’s

center suggest using an agile method. Shapes distinctly toward the periphery suggest using a plan-driven methodology. More varied shapes suggest a hybrid method of both agile and plan-driven practices. In our XP case studies, we focus on the comparison of the shape of the polar chart. However, the remaining context variables recorded in the XP-cf provide rich information on the team. In our work, if we find disparities such that two teams with similar “shapes” have different results, we will dig deeper into the context information to determine what other factor(s) can explain the variability of results.

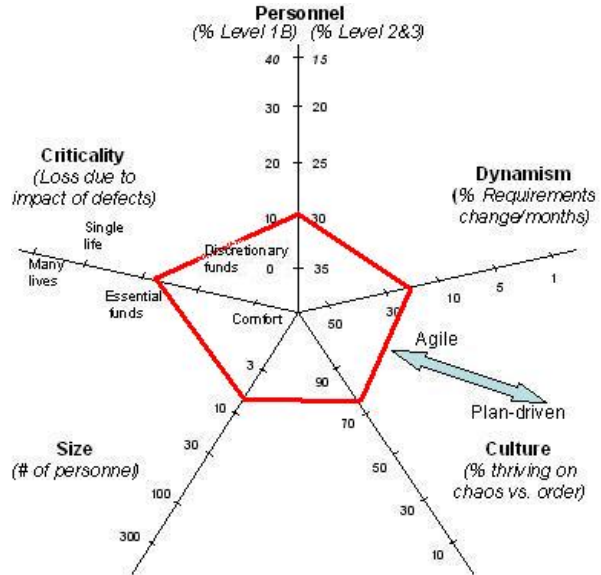


Figure 1: Boehm/Turner Polar Chart

3.2 Adherence metrics (XP-am)

The XP-am is a set of objective and subjective measures of process conformance relative to the team’s actual execution of a predefined process. Through the XP-cf, consumers of the case study can learn, “*But, what did the team really do?*”

Teams can say “We do XP.” But what does that mean? Khaled El Emam [6] surveyed project managers, chief executive officers, developers, and vice-presidents of engineering for 21 software projects. El Emam found that none of the companies adopted agile practices in a “pure” form. Project teams chose which practices to adopt selectively and developed customized approaches to operate within their particular contexts. So, not all XP teams are “equal.” Just as a doctor running a clinical trial needs to know if the subject actually took the medication, software engineering researchers need to know what the team really did and why they chose to do what they did.

Recording the team’s adherence to the ideal practice/method will help validate whether the case study results can be attributed to the practice/method under study or to some other factor. The adherence metrics also help to investigate which XP practices are best suited to a team’s particular project

characteristics. The framework is a measure against the pure (fully implemented) use of the technology.

As mentioned, the adherence metrics are both objective and subjective and are both qualitative and quantitative. The subjective metrics take the form of qualitative data obtained from a web-administered, developer survey that asks how often each XP practice is used. We also conduct semi-structured interviews (the interview protocol is part of the XP-EF definition) with team members to discuss the aspects of their project that enabled or prevented them from using a particular XP practice. The objective metrics are primarily based on process artifacts (such as source code), though we have found it difficult to find automated, objective metrics for all XP practices. Evolving this portion of the adherence metric suite has been a continuous activity.

3.3. Outcome measures (XP-om)

The XP-om enables one to assess and report the business-related results of a team when using a full or partial set of XP practices. Through the XP-om, the consumers of a XP case study can learn, "Was the team successful?" The XP-om consists of traditional external software development metrics, such as productivity and quality. These results can be used to provide evidence in support of or to refute theories about quality, productivity, customer satisfaction, and other purported benefits of the XP process.

4. EXPERIENCES USING AND MODIFYING XP-EF

XP-EF evolved from Version 1.0 to Version 1.4 through actual use with our case studies, through formal feedback on peer reviewed publications, and through informal feedback from practitioners and researchers. We are currently creating the XP2-EF based upon the second edition of Kent Beck's [3] book which redefines the initial definition of XP. The XP2-EF is a collaboration with researchers at University of Cagliari in Italy who have sent a doctoral student to North Carolina State University (NCSU) for a period of almost three months. This collaborative effort represents the unification of research agendas and active international collaboration that can take place around a common framework.

Additionally, we are working with researchers and practitioners at ABB to define the Team Software Process [12] Evaluation Framework (TSP-EF) and at IBM to define the Rational Unified Process [13] Evaluation Framework (RUP-EF) based upon the XP-EF. Both the TSP-EF and the RUP-EF will be published as NCSU Technical Reports during the first quarter of 2005. Our experiences with the XP2-EF, TSP-EF, and RUP-EF have lead us to consider the following generalities about adapting the framework for other processes:

- **Context factors.** Minimal, if any, change is necessary to adapt the context factors between processes. The context factors recorded in a case study are to some extent a function of what is being studied. However, we believe there is a minimal set of essential, omnipresent metrics that must be recorded for each study of software engineering technology, whether it be of a soft-

ware development methodology (such as XP) or of a specific practice (such as pair programming).

- **Adherence measures.** Significant changes are necessary between processes. The adherence metrics are focused on the specific software development practices that comprise the software development methodology or practice. Practices can vary, sometimes significantly, between processes. Case studies of singular practices, such as inspections or pair programming, would have a smaller set of adherence metrics than would a complete software development process.
- **Outcome measures.** Minimal, if any, change is necessary to adapt the outcome factors between processes. Most often business stakeholders are looking for similar business-related outcomes regardless of the practice or process. There are exceptions, however. For example, the TSP-EF has an outcome measure of estimation accuracy. TSP stakeholders consider predictability to be of prime importance, a business-related outcome not as explicitly shared by XP and RUP.

5. SUMMARY

Software practitioners desire compelling evidence of the efficacy of software development methods based upon realistic empirical studies in a context similar to their own. In this paper, we point the reader toward an industrial case study measurement framework, the XP-EF, designed for use with XP industrial case studies. An international team of researchers has utilized this framework for three XP case studies and more are in process. Commentary on the current version of this framework has been obtained by anonymous reviewers and by researchers and practitioners. More recently, the XP-EF is undergoing adaptation for the publication of the TSP-EF, the RUP-EF, and the XP2-EF. Again, an international team of researchers and practitioners are participating in authoring these frameworks.

We feel a measurement framework such as the XP-EF is necessary to unite researchers for a common research agenda and to allow them to structure families of case studies examining a particular technology. We invite the examination of this framework such that the essential elements of a measurement framework can be identified for the research community. We envision that the examination and refinement of the framework and the extraction of essential components can be the focus of a EBSE working group.

ACKNOWLEDGEMENTS

This research was supported by the North Carolina State University Center for Advanced Computing and Communication Grant 02-02 and 04-06.

REFERENCES

- [1] V. R. Basili, F. Shull, and F. Lanubile, "Building Knowledge Through Families of Experiments," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 456 - 473, 1999.
- [2] K. Beck, *Extreme Programming Explained: Embrace Change*. Reading, Mass.: Addison-Wesley, 2000.

- [3] K. Beck, *Extreme Programming Explained: Embrace Change*, Second ed. Reading, Mass.: Addison-Wesley, 2005.
- [4] B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA: Addison-Wesley, 2003.
- [5] T. Dybå, B. Kitchenham, and M. Jørgensen, "Evidence-Based Software Engineering for Practitioners," *IEEE Software*, vol. 22, no. 1, pp. 58-65, 2005.
- [6] K. El-Emam, "Finding Success in Small Software Projects," *Agile Project Management*, vol. 4, no. 11, 2003.
- [7] N. Fenton, S. L. Pfleeger, and R. Glass, "Science and Substance: A Challenge to Software Engineers," *IEEE Software*, vol. 11, no. 4, pp. 86-95, July/August 1994.
- [8] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*. Brooks/Cole, 1998.
- [9] H. Gallis, E. Arisholm, and T. Dybå, "An Initial Framework for Research on Pair Programming," International Symposium on Empirical Software Engineering, Roman Castles (Rome), Italy, 2003, pp. 132-142.
- [10] W. Harrison, "N=1: an alternative for empirical software engineering research?" *Empirical Software Engineering*, vol. 2, no. 1, pp. 7-10, 1997.
- [11] W. Harrison, "Propaganda and Software Development," *IEEE Software*, vol. 21, no. 5, pp. 5-7, 2004.
- [12] W. S. Humphrey, *Introduction to the Team Software Process*. Reading, Mass.: Addison Wesley, 2000.
- [13] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. Reading, Massachusetts: Addison-Wesley, 1999.
- [14] C. Jones, *Software Assessments, Benchmarks, and Best Practices*. Boston, MA: Addison Wesley, 2000.
- [15] B. Kitchenham, T. Dybå, and M. Jørgensen, "Evidence-based Software Engineering," International Conference on Software Engineering, Edinburgh, Scotland, 2004, pp. 273-281.
- [16] B. Kitchenham, S. L. Pfleeger, and N. Fenton, "Towards a Framework for Software Measurement Validation," *IEEE Transactions on Software Engineering*, vol. 21, no. 12, pp. 929-944, 1995.
- [17] B. Kitchenham, L. Pickard, and S. L. Pfleeger, "Case Studies for Method and Tool Evaluation," *IEEE Software*, vol. 12, no. 4, pp. 52-62, July 1995.
- [18] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary Guidelines for Empirical Research in Software Engineering," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721-733, 2002.
- [19] B. A. Kitchenham, G. H. Travassos, A. v. Mayrhauser, F. Niessink, N. F. Schneidewind, J. Singer, S. Takada, R. Vehvilainen, and H. Yang, "Towards an ontology of software maintenance," *Journal of Software Maintenance: Research and Practice*, vol. Volume 11, no. 6, November 1999.
- [20] L. Layman, L. Williams, and L. Cunningham, "Exploring Extreme Programming in Context: An Industrial Case Study," Agile Development Conference, Salt Lake City, UT, 2004, pp. 32-41.
- [21] L. Layman, L. Williams, and L. Cunningham, "Motivations and Measurements in an Agile Case Study," ACM SIGSOFT Foundation in Software Engineering Workshop Quantitative Techniques for Software Agile Processes (QTE-SWAP), Newport Beach, CA, 2004, pp.
- [22] R. M. Lindsay and A. S. C. Ehrenberg, "The Design of Replicated Studies," *The American Statistician*, vol. 47, no. 3, pp. 217-228, 1993.
- [23] L. Mathiassen, "Collaborative Practice Research," *Information Technology & People*, vol. 15, no. 4, pp. 321-345, 2002.
- [24] S. L. Pfleeger, "Soup or Art? The Role of Evidential Force in Empirical Software Engineering," *IEEE Software*, vol. 22, no. 1, pp. 66-73, 2005.
- [25] S. L. Pfleeger, "Albert Einstein and Empirical Software Engineering," *IEEE Computer*, vol. 32, no. 10, pp. 32-37, October 1999.
- [26] C. Potts, "Software Engineering Research Revisited," *IEEE Software*, no., pp. 19-28, September 1993.
- [27] N. Schneidewind, "Methodology for Validating Software Metrics," *IEEE Transactions on Software Engineering*, vol. 18, no. 5, pp. 410-422, May 1992.
- [28] D. Sjøberg, B. Anda, E. Arisholm, T. Dybå, and M. Jørgensen, "Conducting Realistic Experiments in Software Engineering," International Symposium on Empirical Software Engineering, Nara, Japan, 2002, pp.
- [29] H. Svensson, "An Industrial Case Study on Introducing XP in a Complex Software Development Environment," Empirical Assessment of Software Engineering (EASE), Edinburgh, Scotland, 2004, pp. 1-10.
- [30] W. F. Tichy, "Should Computer Scientists Experiment More?" *IEEE Computer*, vol. 31, no. 5, pp. 32-40, 1998.
- [31] L. Williams, W. Krebs, L. Layman, A. Antón, and P. Abrahamsson, "Toward a Framework for Evaluating Extreme Programming," Empirical Assessment in Software Eng. (EASE) 2004, Edinburgh, Scot., 2004, pp. 11-20.
- [32] L. Williams, L. Layman, and W. Krebs, "Extreme Programming Evaluation Framework for Object-Oriented Languages -- Version 1.4," North Carolina State University, Raleigh, NC, Computer Science TR-2004-18 <http://www.csc.ncsu.edu/research/tech/reports.php>, 2004.
- [33] R. K. Yin, *Case Study Research: Design and Methods*, Third ed. Thousand Oaks, CA: Sage Pub., 2003.
- [34] M. V. Zelkowitz, D. R. Wallace, and D. Binkley, "Culture Conflicts in Software Engineering Technology Transfer," NASA Goddard Software Engineering Workshop, 1998, pp.