Lab-based Action Design Research

Paul Ralph
Lancaster University
Lancaster UK
paul@paulralph.name

ABSTRACT

This paper proposes a research methodology, Lab-based Action Design Research, which combines organizational intervention (action research), building innovative artifacts (engineering research) and studies of software development practice (behavioral research) within a laboratory environment. Seven principles for successful Lab-based Action Design Research are proposed – attract funding with a win-win scenario; select inspiring projects; conduct simultaneous studies; mix methods; use longitudinal, quasi-experimental designs; use enterprise-level technical infrastructure; use established project management infrastructure. Initial evaluation indicates that the proposed approach is practical and may produce improvements in internal validity and theoretical generalizability.

Categories and Subject Descriptors
D.2.8, D.2.9, D.2.10 [**Software Engineering**]: Metrics, Management, Design.

General Terms
Design, Measurement, Experimentation, Human Factors, Theory.

Keywords
Methodology, Metrics, Empirical Software Engineering, Design.

1. INTRODUCTION

Diverse software engineering research initiatives are frustrated by the limitations of conventional lab and field study methods. Field studies involve real problems, constraints and complexity but limit researcher control and inhibit establishing causality. Lab environments meanwhile provide substantial control and good internal validity but lab studies lasting a few hours or even a few days cannot capture the complexity of even modest projects, which occupy several developers for months or years. Even field experiments rely on either limited experimental controls or rare natural controls. These limitations may contribute to the proliferation of papers which formulate processes, methods or algorithms but exhibit problematic or no evaluation [6, 7, 10, 19].

The complexity-control tradeoff between lab studies (high control, low complexity) and field studies (high complexity, low control) highlights the need for a middle-ground (medium control, medium complexity). This motivates the following, research question.

Research Question: *Can a research methodology practically combine realism and complexity with strong experimental controls?*

Consequently, this paper proposes a novel methodology, termed *Lab-based Action Design Research* (LADR). The paper proceeds by reviewing background on action research and design research (§2), then describing (§3) and evaluating (§4) LADR and its key principles. The paper concludes with a summary of its contributions and suggestions for future studies.

For our purposes, a *lab study* is any research conducted in a setting substantially created and controlled by the researcher, including controlled experiments, quasi-experimental designs and informal evaluations, regardless of whether participants are amateurs or professionals. In contrast, a *field study* is any research conducted in an organizational setting over which the researcher has little control, including ethnography, case study and grounded theory. Meanwhile, a *control* is an attempt to limit the effect of one or more variables on a dependent variable or on a relationship between variables. Finally, a system is *complex* to the extent that it exhibits emergent behaviors not evident from its components [2].

2. ACTION, DESIGN, AND ACTION-DESIGN RESEARCH

Action Research is a type of field study in which "the researcher enters a real-world situation and aims both to improve it and to acquire knowledge" [5]. It differs from ethnography in that the researcher intervenes in the research context to achieve a practical goal and reflects on this intervention. Action researchers use participant observation as their primary data collection mechanism. Action research has been praised for its utility in practical problem-solving and engaged scholarship [18], but criticized for researcher bias and lacking rigor [12].

At least three kinds of software engineering studies resemble action research. First, proof-of-concept field studies where a researcher deploys a novel tool, algorithm or method in an organizational setting is a kind of action research. Second, when a researcher discovers important concepts or evidence during a consulting project, it can be presented post hoc as action research. Third, field studies where the researcher has traded consulting for access may be better presented as action research.

Design Research is the information systems community's term for research centered on developing innovative technologies [9]. Seven core guidelines have been proposed – 1) produce an artifact; 2) address a relevant problem; 3) evaluate the artifact; 4) provide a clear and verifiable research contribution in the form of an artifact, a methodology, or design knowledge; 5) construct and evaluate the artifact rigorously; 6) design artifacts by searching for the best alternatives; 7) communicate the research in a manner understandable by both technical and managerial audiences [9]. Design research has been praised for promoting innovation but criticized for lacking rigor, especially where artifact evaluation is limited [8].

Design Research is presented as a Waterfall-like sequence of problem definition, artifact development and evaluation [8]. Recognizing that artifacts are often simultaneously shaped by researchers and use organizations, Sein et al. proposed *Action Design Research* (ADR), where artifact construction, organizational intervention and artifact evaluation occur concurrently [17]. They provide seven principles for ADR – 1) practical problems are addressed; 2) artifact design and evaluation are informed by theory; 3) artifacts are mutually shaped by the researcher and the use organization; 4) mutual learning between researchers and practitioners; 5) concurrent evaluation; 6) treating design as a process of guiding emergence; 7) generalizing specific outcomes to problem classes, solution classes and design principles [17]. Here, "guiding emergence" refers to managing unpredictable behavior or unintended consequences rather than pretending the researcher can plan for every contingency.

ADR's use of concurrent construction, intervention and evaluation simultaneously hinders researcher control and facilitates engagement, relevance and realistic complexity. ADR may therefor provide a suitable foundation for a novel methodology combining realistic complexity with greater researcher control.

3. LADR PRINCIPLES

Lab-based Action Design Research (LADR) is a research methodology that combines the organizational intervention of action research, the artifact-centricity of design research and the concurrent construction/intervention/evaluation of action design research with some of the researcher control available in experimental, lab-based studies. It is characterized by:

1. designing a technological artifact (e.g. a software system) in a laboratory context where the researcher exerts substantial control over the project, process and environment
2. intervene in a real-world situation with the designed artifact
3. concurrent development, organizational intervention, artifact evaluation and reflection on the intervention
4. generating new knowledge about the organization, the artifact, design practice, or theories thereof

Conducting LADR requires an organization (the lab) comprising one or more research teams, software development teams, research projects, development projects, as well as research, technological and management infrastructure and physical space. The development teams use the managerial and technological infrastructure to complete development projects while the research teams use research infrastructure and the development projects to complete research projects. Ideally, the lab space is within the organization where the artifact will be deployed.

The core principles for action design research apply equally to LADR. This remainder of this section describes seven principles specific to LADR, based on the study described in Section 4.

3.1 Attract Funding with a Win-Win Scenario

LADR requires moderate funding to cover participant salaries, infrastructure and research and incidental expenses. While attracting external grants is ideal, internal funds may also be available given the right pitch.

Effectively deployed, LADR manifests significant benefits for diverse stakeholders in the university context. If the development team is composed of university students, they receive financial support and relevant work experience. Meanwhile, the university benefits not only from more experienced graduates (improved reputation) but also from the software projects the development team completes (as we select projects based on the university's needs). In addition, LADR creates research infrastructure available to numerous faculty and PhD students. Meanwhile, both the student-participants and the university benefit from the design expertise brought to the project by the research team. Construed this way, modest internal funding may be forthcoming.

3.2 Select Inspiring Projects

Project selection involves balancing the concerns of the funder and participants with the needs implied by the research questions. It is tempting to believe that if the primary research question is about design practice (e.g. *does tool X increase developer productivity?*) then the details of the development project are irrelevant. This is a grievous error. Aside from the importance of project complexity as a moderating variable, developers quickly disengage from simple or boring projects (e.g. digitizing paper-based forms). Selecting an inspiring project also helps to justify the lab's funding, create buzz in the organization, attract more and better participants, and increases potential research output (§3.3).

Moreover, the ideal project is not a simulation but an actual project for real stakeholders who intend to use the results. Realistic organizational complexity comes from doing a real project for a real organization. However, the lab may have less internal complexity than working in a large or bureaucratic firm.

3.3 Conduct Simultaneous Studies

LADR allows researchers to conduct multiple, separate but interconnected studies simultaneously. In the simplest instance, at least two research projects may be combined. First, the development team may build and evaluate an innovative artifact that is, itself, a research contribution. Second, the research team may theorize about some aspect of the development team's practice and test the theory. For example, the development team may be building and evaluating a novel recommender system to help students select courses; while the research team may be exploring coevolution behavior [14]. Moreover, multiple studies may require multiple types of reflection, e.g., reflection on: 1) the organizational intervention; 2) the design artifact; 3) the developers and their process; 4) the research method.

3.4 Mix Methods

LADR creates empirical infrastructure that can be leveraged for diverse studies including experiments and ethnographies as well as action research and engineering research. Combining several approaches facilitates data triangulation, mitigates mono-method bias and may produce deeper insights. For example, simultaneous theory testing using a (positivist) quasi-experimental design and related theory-building using an (interpretivist) ethnographic approach prompts reflection on the same phenomenon through heterogenous theoretical lenses and philosophical perspectives, spurring creativity and demanding more nuanced analysis.

3.5 Use Quasi-Experimental Designs

LADR is inherently longitudinal – completing inspiring projects will require months of sustained effort. Furthermore, the unit of analysis for many LADR studies will be a project, team, or design artifact. Therefore, having a control group and enough participants for cross-sectional statistical analysis may be impractical.

Quasi-experimental, time series studies may mitigate this limitation. For example, to study whether peer programming decreases coding errors, we could randomly assign participants to two groups, have the treatment group code in pairs while the control group codes individually and comparing the mean error across the two groups. The same research question might be studied by randomly assigning days to two groups and having the team code in pairs on treatment days and individually on control days, and then comparing means across days.

3.6 Use Enterprise Infrastructure

Keeping track of a LADR project's data and progress is challenging. Using sophisticated technical infrastructure may help. For example, a web application development project may benefit from an integrated stack involving a version control system, static analysis tools, testing tools, reporting tools, a continuous integration server, web hosting and a project management suite. The data available via the project management suite and reporting tools forms a convenient set of dependent variables appropriate for myriad research questions.

3.7 Use Project Management Infrastructure

Unless manipulating project management practices is part of the research design, adopting as established management approach may be helpful. For example, mounting evidence indicates that Scrum [16] is positively related with developer productivity [4]. Scrum may be combined with many best practices including Extreme Programming [3] and lean engineering [13]. In contrast, using homegrown or ad hoc development practices may increase management overhead, distract from research goals and even negatively impact project success [1].

4. EVALUATION OF LADR

4.1 Conceptual Evaluation

A core benefit of LADR is increased internal validity in studies of design practices and projects, where LADR allows more researcher control than a field study. However, internal validity depends on how LADR is used. Strictly observational approaches will have validity characteristics similar to that of an ethnography, while intervention-oriented approaches will have similar validity to Action Research. However, LADR also supports quasi-experimental time series designs, which can produce strong evidence of causality. The primary threat to internal validity for this design is an unnoticed event occurring contemporaneously with the treatment, which contributes to the observed effect. The researcher can mitigate this threat and increase internal validity by combining the quasi-experimental design with direct observation or participant observation. Pragmatically speaking, an engaged researcher should notice such third variables. Additionally, as with all longitudinal designs, mortality threats may apply.

LADR also has interesting generalizability characteristics. Generalizability refers to several types of scientific inference [11]. Statistical generalizability, which involves inferring properties of a population from a representative sample, is rare in software engineering. Even in survey research, representative sampling is hindered by the absence a population list from which to sample. Experiments and field studies rarely support statistical generalizability, either because the sample size is too low (field studies) or not representative (experiments). Similarly, LADR does not facilitate statistical generalizability. However, scientists also generalize from data to descriptions (DD), from description to theory (DT), from theory to description (TD) and from concepts to theory (CT) [11]. Action Design Research involves DD and DT, specifically "(1) generalization of the problem instance, (2) generalization of the solution instance, and (3) derivation of design principles from the design research outcomes" [17]. Interpretive and exploratory studies predominately involve DD and DT, i.e., generalizing from observations. Theory testing studies predominately use DD and TD – statistical generalization is rarely used without random sampling. LADR supports DD, DT and CT.

Moreover, LADR should have minimal impact on reliability. Pseudo-experimental designs with accurate measures should have relatively strong test-retest reliability while observational studies will generally have weaker test-retest reliability (as each researcher will intervene in organizations differently). Reliability of subjective analyses may be enhanced in by having two researchers code data independently and computing inter-rater reliability. The process of reconciling inter-rater disagreement may further enhance reliability.

Similarly, LADR should have little effect on construct validity. However, LADR may permit more robust operationalizations of constructs than are possible in a lab study. For example, if we hypothesize that peer programming increases team cohesion over time, a four-hour lab study would limit operationalization of time compared to a six month LADR study.

From a different perspective, LADR may be more realistic than a conventional lab study in three ways – 1) realistic timescale; 2) real projects; 3) realistic complexity. Generalizing from lab studies to practice is hampered by toy problems, lack of problem framing, short durations and unrealistic tidiness. LADR results are intuitively more transferable to real projects because LADR involves completing real projects. However, LADR teams are unlikely to have the same internal politics as development firms.

In summary, LADR supports higher internal validity than a field study (via quasi-experimental time-series designs), but not as high as a randomized controlled experiment. Inversely, LADR supports more realistic trials of new technologies than lab studies but not quite as realistic as field trials. Statistical generalization is not normally supported by field studies, lab studies or LADR studies. LADR should have little effect on reliability of construct validity.

4.2 Empirical Evaluation

I conducted a nine-month trial of LADR beginning in Feb 2012. Briefly, a team of seven undergraduates and postgraduates developed a mobile application for internal university stakeholders using a Scrum-like approach [16]. I managed the team while conducted observational research on design practices, tools and metrics. This evaluation demonstrates that LADR can be implemented in practice but does establish its validity.

The lab facilitated substantial data collection including video recordings of team meetings, audio recordings of stakeholder consultations and images of diagrams created by the team. All emails sent within and between team members were archived using a Google Group. All code, documentation and changes thereof were recorded using Git revision control. Other documents created by the team, including user stories, proposals and notes, were captured using a combination of Google Docs and a shared Dropbox. As no one system seemed capable of organizing all of the transcripts, notes, documents and media collected, a combination of Evernote, iTunes and directories were used with pointers in Evernote to content in other systems. In hindsight, it would have been wise to consider data storage infrastructure more carefully prior to the study.

LADR also facilitated substantial control over the research environment including the project management framework (Scrum) and software (ScrumWorks Pro), project selection, layout of physical workspace, and use of specific practices. Participants understood that it was a research project as well as a development project and were very accepting of constraints and suggestions from the research side. Programming languages, coding styles, documentation styles, toolsets, testing practices, etc. were all potentially manipulable.

However, several nontrivial challenges emerged. First, simultaneously operating the lab and collecting and analyzing data requires a kind of doublethink where the researcher must oscillate between practical and theoretical reflection. This was very challenging at times, especially when events unfolded rapidly, and the researcher experienced the desire to stop time long enough to reflect and write notes. Second, many variables of interest are available via version control systems and the toolsets that run on them. However, these toolsets are language specific. As the team initially used a proprietary XML variant rather than a more popular language (e.g., Java, C++), the lack of available tools hindered quantitative data collection. The team later switched to a popular object-oriented language, intending to purchase an integrated infrastructure including everything from version control to project management and hosting, preferably with good integration across tools. For example, the backlog items in the project management system should be tied to related code fragments in the version control system. This revealed a third challenge: despite intense interest in cloud-based systems, the kind of turnkey solution we hoped for remains elusive.

5. DISCUSSION AND CONCLUSION

This paper proposes Lab-based Action Design Research, a research methodology where an artifact is concurrently constructed in a lab setting and evaluated in an organizational setting. The core contribution of this paper is the description of LADR and its seven principles – attract funding with a win-win scenario; select inspiring projects; conduct simultaneous studies; mix methods; use quasi-experimental designs; use enterprise-level technical infrastructure; use project management infrastructure.

LADR manifests at least three benefits. First, it permits more realistic projects, in terms of size, complexity and inclusion of problem framing, than conventional lab studies. Second, it facilitates greater control and internal validity than conventional field research. Third, it allows researchers to simultaneously run multiple studies, including organization interventions, building innovative artifacts and studying design practice.

However, LADR is limited in several ways. First, it requires moderate funding to cover the costs of participant salaries. Second, having a control group or multiple treatment groups is financial impractical and methodologically problematic. Third, simultaneously participating in and analyzing the results is challenging. Researchers can mitigate these limitations by making a case for funding to their universities, using longitudinal quasi-experimental designs coupled with time-series analysis and recruiting PhD students or other partners for team-based research.

More research is needed to explore LADR's methodological properties and implications, to establish its usefulness and to produce additional guidance. This paper meanwhile describes LADR in the hope that it may be beneficial to others struggling with the limitations of contemporary research methods.

Finally, my experience developing LADR has reconfirmed the maxim that not everything that is measurable is important and not everything that is important is measurable. We lack good measures of software project success, although its dimensions are becoming more clear [15]. We also lack good measures of productivity, or even work done. Existing measures of work and software quality were unhelpful with LADR. In conclusions, the lack of meaningful measures for key variable remains a crucial challenge for future work involving LADR and for empirical software engineering research in general.

6. ACKNOWLEDGMENTS

Thanks are due to all of the Lancaster staff who made this project possible (especially Jon Gallagher) and to all of the participants of the Lancaster Design Practice Lab. This research was funded by Lancaster University.

7. REFERENCES

[1] Ambler, S. 2010. 2010 Agile project success rates survey results. *Ambysoft*. http://www.ambysoft.com/surveys/ agileSuccess2010.html, (2010).

[2] Anderson, P. 1999. Complexity Theory and Organization Science. *Organization Science*. 10, 3 (1999), 216–232.

[3] Beck, K. 2005. *Extreme Programming eXplained: Embrace Change*. Addison Wesley.

[4] Cardozo, E., Neto, J., Barza, A., França, A. and da Silva, F. 2010. SCRUM and Productivity in Software Projects: A Systematic Literature Review. *Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering (EASE)*.

[5] Checkland, P. and Holwell, S. 1998. Action Research: Its Nature and Validity. *Systemic Practice and Action Research*. 11, 1 (1998), 9–21.

[6] Dyba, T. and Dingsøyr, T. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology*. 50, 9-10 (2008), 833–859.

[7] Glass, R.L., Vessey, I. and Ramesh, V. 2002. Research in software engineering: an analysis of the literature. *Information and Software Technology*. 44, 8 (Jun. 2002), 491–506.

[8] Hevner, A. and Chatterjee, S. 2010. *Design Research in Information Systems: Theory and Practice*. Springer.

[9] Hevner, A., March, S.T., Park, J. and Ram, S. 2004. Design Science in Information Systems Research. *MIS Quarterly*. 28, 1 (Mar. 2004), 75–105.

[10] Kampenes, V.B., Dyba, T., Hannay, J.E. and Sjøberg, D.I.K.I.K. 2009. A systematic review of quasi-experiments in software engineering. *Information and Software Technology*. 51, 1 (Jan. 2009), 71–82.

[11] Lee, A.S. and Baskerville, R.L. 2003. Generalizing generalizability in information systems research. *Information Systems Research*. 14, 3 (2003), 221–243.

[12] McKay, J. and Marshall, P. 2001. The dual imperatives of action research. *Information Technology & People*. 14, 1 (2001), 46–59.

[13] Poppendieck, M. and Poppendieck, T. 2003. *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional.

[14] Ralph, P. 2013. The Sensemaking-Coevolution-Implementation Theory of Software Design. *arXiv*: 1302.4061 [cs.SE].

[15] Ralph, P., & Kelly, P. 2014. The Dimensions of Software Engineering Success. *Proceedings of the 2014 International Conference on Software Engineering*. ACM.

[16] Schwaber, K. 2004. *Agile Project Management with Scrum*. Microsoft Press.

[17] Sein, M., Henfridsson, O., Purao, S., Rossi, M. and Lindgren, R. 2011. Action Design Research. *MIS Quarterly*. 35, 1 (2011), 37–56.

[18] Van de Ven, A.H. 2007. *Engaged Scholarship: a Guide for Organizational and Social Research*. Oxford University Press.

[19] Zelkowitz, M.V. and Wallace, D. 1997. Experimental Validation in Software Engineering. *Information and Software Technology*. 39, 11 (1997), 735–743.