Five Days of Empirical Software Engineering: the PASED Experience

Massimiliano Di Penta*, Giuliano Antoniol**, Daniel M. German***, Yann-Gaël Guéhéneuc**, Bram Adams****
*University of Sannio, Benevento, Italy
**École Polytechnique de Montréal, Québec, Canada
***University of Victoria, Victoria, BC, Canada
****Queen's University, Kingston, ON, Canada
dipenta@unisannio.it, antoniol@ieee.org, dmg@uvic.ca, yann-gael.gueheneuc@polymtl.ca, bram@cs.queensu.ca

***Abstract*—Acquiring the skills to plan and conduct different kinds of empirical studies is a mandatory requirement for graduate students working in the field of software engineering. These skills typically can only be developed based on the teaching and experience of the students' supervisor, because of the lack of specific, practical courses providing these skills.**

To fill this gap, we organized the first Canadian Summer School on Practical Analyses of Software Engineering Data (PASED). The aim of PASED is to provide—using a "learning by doing" model of teaching—a solid foundation to software engineering graduate students on conducting empirical studies. This paper describes our experience in organizing the PASED school, i.e., what challenges we encountered, how we designed the lectures and laboratories, and what could be improved in the future based on the participants' feedback.

***Keywords*-Empirical Software Engineering, Software Engineering Education.**

I. Introduction

Empirical research has become crucial for today's software engineering researchers. Virtually any software engineering research publication is expected to have some form of empirical validation, *e.g.,* a case study or a controlled experiment. Moreover, software engineering is gaining maturity, moving from craftsmanship towards a consolidated scientific discipline, where conclusions are drawn based on empirical evidences rather than on common wisdom.

In recent years, many authors have provided guidelines on conducting empirical software engineering research [1], [2], [3]. Some universities provide empirical software engineering courses as part of their graduate curricula, *e.g.,* Easterbrook's CSC2130: Empirical Research Methods in Software Engineering at the University of Toronto (2009)[1], Herbsleb's 08-803: Empirical Methods for Socio-Technical Research at CMU (2010)[2], or the course described by Jaccheri and Østerlie [4]. However, the large majority of software engineering graduate students still do not have access to such courses and must rely on generic statistics/data mining courses from computer science/software engineering curricula. Such courses do not completely satisfy the needs of empirical software engineering researchers, because:

- developers and users are at the center of the development activities (either as producers or consumers), making software engineering research more similar to social sciences than to natural sciences research;
- many software engineering studies deal with data sets for which parametric statistics (taught in typical statistics courses) are not suitable. Thus, empirical software engineering research courses should provide solid foundations on non-parametric statistical procedures;
- empirical software engineering researchers require specific skills to extract information from software repositories containing a mixture of structured and unstructured data, often incomplete and imprecise [5], [6].

Consequently, similar to others, we identified the need for teaching students the ability to plan and conduct different kinds of empirical studies. We acted by organizing the first Canadian Summer School on Practical Analyses of Software Engineering Data (PASED)[3] from July 16 to 20, 2011 in Montréal, Québec, Canada, with the support of MITACS Networking and Training Initiative[4]. We aimed at offering a venue in which graduate students, as well as young faculty members, could learn the sound foundations of empirical software engineering research from world experts, with a particular emphasis on the analysis of data from software repositories and on conducting studies involving human subjects.

The school attracted 44 participants from 25 Universities (9 countries) across the world. It featured morning lectures on general empirical software engineering topics, such as experimental design and data analysis, and also on specific topics, such as mining software repositories, natural language processing, and machine learning. All lectures were complemented by afternoon laboratories where students could practice the concepts they had learned in the morning. It also included keynotes by leading researchers in software engineering and industrial partners interested in the field, a poster session where participants could showcase their current work, and several social events to foster discussion

[1]http://www.cs.toronto.edu/~sme/CSC2130/index.html
[2]http://herbsleb.org/web-courses/courses.shtml
[3]http://pased.soccerlab.polymtl.ca
[4]http://www.mitacs.ca/opportunites/13

among participants.

The educational objectives of the school were to:

1) learn to plan and conduct software engineering experiments with human subjects and collect related data;
2) learn to plan and conduct software engineering experiments involving the mining of data from (un)structured software repositories;
3) learn to build prediction and classification models from the collected data, and to use these models.

Mastering each of these objectives would have, on average, required at least months of individual study by a student and tutoring by the student's supervisor. Thus, we faced the challenge of teaching the students how to apply the appropriate empirical methods and tools to their everyday research work. Therefore, rather than providing deep details on the various existing methods and tools, we showed the students, using a "learning by doing" model of teaching, how problems recurring when carrying out empirical research can be solved, building and relying on a proper skill and tool set.

In the following, Section II describes the main challenges that we encountered when planning and organizing the PASED school. Then, Section III illustrates how the school was conducted. Section IV summarizes the lessons learned from organizing the school, distills guidelines on planning and organizing such a school based on the participants' feedback, and concludes the paper.

II. Challenges

When planning the PASED School, we dealt with three major challenges related to the school's content and organization.

A. Choosing the School Topics

The first challenge was choosing the concrete topics of the school. Empirical research in software engineering is a broad area. Different researchers conduct different kinds of empirical studies and, thus, focus on different aspects of empirical software engineering. Based on our experiences, on the background of potential attendees (collected through a pre-school survey as explained in Section III), and on the recent empirical studies published in major software engineering venues, we identified three course requirements:

1) **Req 1:** Extracting facts from software repositories;
2) **Req 2:** Designing different kinds of empirical studies, including those involving human subjects;
3) **Req 3:** Analyzing and presenting empirical study results.

B. Combining Theory and Practice Effectively

The second challenge was providing the right mix of theoretical and practical content. While the school provided material on performing statistical analyses, it would have been impossible, given the available time frame, to include complete courses on the covered topics (*e.g.,* on statistical techniques or mining software repositories) and on their application to software engineering data.

Thus, we tried to provide direct answers to questions that students would typically ask during their research (and that we asked ourselves), *e.g.,* What is an experimental design? How can I compare the usefulness of two techniques for developers in a controlled experiment? What kind of variables should I measure? Given two data sets containing the test coverage achieved using different techniques, how can I statistically compare them? What test should I use? What kind of graph is most appropriate to show such a dataset? How can I implement the test using available tools?

C. Heterogeneous Participants

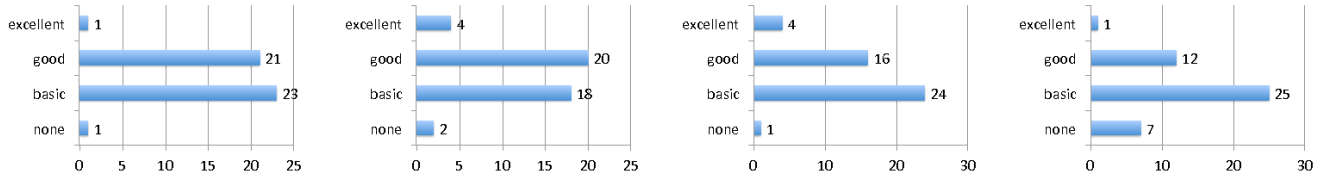
The third challenge was the participants' varied backgrounds. The school received applications from students working on a wide variety of software engineering research topics and with very different levels of experience. Some students were already experts in mining software repositories, while others had significant expertise in statistical analyses. Hence, our goal was to make sure that all students could learn from the school and that nobody would find its content too trivial or too complicated. We (partially) addressed this last challenge by reviewing each participants' backgrounds and expertise levels before the school to fine-tune the covered topics.

III. The School Structure and Contents

To design the detailed curriculum of the school, we asked each participant (when signing up, at least three months before the school) to fill a form describing his or her level of expertise in topics related to the school, *i.e.,* statistical techniques, mining software repositories, machine learning, and experimental software engineering. Figure 1 provides a summary of the pre-school survey. It shows that most students claimed to have "basic" or "good" knowledge on the topics, with some "none" (especially on machine learning) and some "excellent". We adapted the course structure and contents to these answers. Also, we explicitly got in touch with prospective participants who claimed to have an excellent knowledge of some topics to make sure that they understood the goals of the school.

A. Tutorial Lectures

Based on the requirements described in Section II-A, the school was divided into five different topics, each covered in a different way: mining software repositories, experiment design, text mining, statistical techniques, and machine learning. As mentioned before, with few exceptions, participants had basic-to-good knowledge on the school topics. We therefore had to make the school understandable for people having basic knowledge and not boring for the others.



(a) Statistical techniques (b) Empirical Sw. Eng. (c) Mining Software Repos. (d) Machine learning

Figure 1. Level of knowledge of PASED participants.

We thus divided the 5-day school in ascending order of complexity, according to the participants' answers.

Day 1 was dedicated to Mining Software Repositories. Its objective was two-fold: first, to motivate the need for empirical research by providing an overview of the kind of high-level knowledge that decision makers need in their daily practice; second, to introduce to students software repositories, the data they contain, and various techniques used to mine them [7].

Day 2 was dedicated to experiment design. Its goal was to describe how to design and plan an experiment and to introduce students to guidelines provided by well-known papers and textbooks [1], [2], [3]. The emphasis was on defining experiment variables, choosing the right research method, and properly designing and planning the study.

Day 3 was used to introduce students to information retrieval methods and their applications, in particular to the use of Vector Space Models [8] and Latent Semantic Indexing [9] to recover traceability links and to cluster software artifacts.

Day 4 was dedicated to statistical procedures for software engineering data and their analysis, using examples introduced in the previous days. This was not a thorough course on statistics but rather focused on providing notions of: (1) the kind of statistical tests to use to test different kinds of hypotheses, (2) the preconditions for applying such tests, (3) the implementation of these tests in *R* [10], and (4) the presentation of results.

Day 5 was intended to provide an overview of machine learning models to classify software engineering data. First, it explained the basic methodology for training and testing models. Then, using a running example, it introduced three progressively more powerful, common models (ZeroR, decision trees, and logistic regression). The lectures explained the main principles and pitfalls of each model as well as interpretations of the performance of the models using measures such as Receiver Operating Characteristic (ROC) curves and confusion matrices.

To ease the learning and allow students to review the lectures after the school, lecture materials and videos were made available on the school web site.

B. Laboratories

Every day the school ended with a 2.5 hours laboratory. The goal of each laboratory was to apply the concepts introduced during the morning lectures. All materials are available on the school web site.

The mining software repository laboratory focused on extracting facts from Git repositories. It started with a short introduction on the basic Git commands, then focused on building simple scripts to analyze the activities of a developer. As a laboratory task, participants, working in pairs, had to mine for interesting facts about PostgreSQL developers, using a cloned PostgreSQL Git repository made available to them. Participants were asked to submit their report within 24 hours from the laboratory and the two best pairs were awarded a prize, which proved to a good motivator to promote students' involvement and initiate their interaction with other students.

In the experiment design laboratory, we asked students to design an experiment evaluating the use of unit-test cases for program comprehension purposes. The experiment topic was introduced in a short presentation [11]. Students, working in small groups, developed a document with the experiment definition, the research questions and hypotheses, the dependent and independent variables, the experiment design, and the experiment material and procedure.

The subsequent three laboratories (text mining, statistical procedures, and machine learning) were oriented towards performing practical analyses using the *R* [10] and *Weka* [12] tools on some real datasets. Students applied clustering techniques to bug reports, analyzed results from a previously-published experiment [13], and built bug prediction models using real datasets from the PROMISE repository[5]. To deal with the challenge of making students productive under time-constraints, we provided them with step-by-step instructions—*e.g.,* sequences of R commands with appropriate comments—to perform some of the analyses. Then, we asked students to perform additional analyses consisting of variations of the ones showed in the instructions, *e.g.,* to repeat a statistical test on a different dataset, build a model using different variables, and so on.

[5]http://promisedata.org

C. Keynotes and Networking Opportunities

To better put the lecture content into perspective and to provide inspiration to students about challenging research problems that can be investigated by applying proper empirical methods, the school featured one keynote per day, from both academia and industry. The two first keynotes were given by experts in (empirical) software engineering: G. Murphy, UBC, and P. Devanbu, UC Davis, while the next three by industrial partners: Benchmark Consulting Inc., SAP Labs., and Google Inc.

To foster exchange of ideas among the school participants and lectures/keynote speakers, we organized a poster session in which each student presented his or her own early research ideas. Every student was required to present a poster.

IV. Lessons Learned and Conclusions

At the end of the school, we asked participants to fill out a feedback form. Rather than using scales to allow empirical analysis of the forms, but constrain participants into categories, we preferred free-text forms to first understand the categories with which such a school can be evaluated and to improve the next editions.

Students were excited about the "learning by doing" teaching model, as this model helped them to concretely apply the various methods taught in the courses and to reinforce learning. They also liked the variety of topics chosen and the technical depth of the lectures.

Having laboratoires with each lecture and with step-by-step instructions was probably the part of the school that students liked the most, because they could (1) apply what they learned during the lectures, (2) learn tools that they never used before, *e.g., R*, and (3) interact with speakers and teaching assistants.

We also obtained suggestions to improve future editions of the school. Students liked laboratories and, thus, we would like to make them longer, possibly complemented with some general tutorials on the tools being used. Also, students asked for keynotes on writing empirical research papers, possibly explicitly related to the specific topics of the lectures. Another interesting remark was the request to see "what not to do" when conducting empirical research, possibly by referring to wrong (and published) empirical research. Finally, participants would like to see industry talks that are directly connected with the topics of the lecture, to understand their relevance in the industry.

Last, but not least, we should remark that, thanks to sponsorships and having the school held at the École Polytechnique de Montréal, with its meeting rooms and student housing, we were able to charge a very low fee (CAD$250/CAD$200, tax included, including/excluding room and board). We believe that this low fee was an important factor that contributed to the success of the school. We recommend others thinking of organizing a similar event to keep the prices low (or to provide some bursaries to students who would not be able to afford its cost, otherwise).

We believe that the model that we implemented in the PASED school is useful for both organizing similar summer schools in the future and introducing topics related to empirical software engineering into software engineering curricula.

References

[1] B. Kitchenham, S. L. Pfleeger, L. Pickard, P. Jones, D. C. Hoaglin, K. E. Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Software Eng.*, vol. 28, no. 8, pp. 721–734, 2002.

[2] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering - An Introduction*. Kluwer Academic Publishers, 2000.

[3] N. Juristo and A. Moreno, *Basics of Software Engineering Experimentation*. Kluwer Academic Publishers, 2001.

[4] M. L. Jaccheri and T. Østerlie, "Can we teach empirical software engineering?" in *Proc. of the 11th IEEE Intl. Symp. on Software Metrics (METRICS)*, Como, Italy, September 2005, p. 25.

[5] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement?: a text-based approach to classify change requests," in *Proc. of the conf. of the Centre for Advanced Studies on Collaborative Research (CASCON)*, Richmond Hill, Canada, October 2008, p. 23.

[6] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. T. Devanbu, "Fair and balanced?: bias in bug-fix datasets," in *Proc. of the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Intl. Symp. on the Foundations of Software Engineering (ESEC/FSE)*, Amsterdam, The Netherlands, August 2009, pp. 121–130.

[7] A. E. Hassan, "The road ahead for mining software repositories," in *Frontiers of Software Maintenance (FoSM)*, Beijing, China, 2008.

[8] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison-Wesley, 1999.

[9] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[10] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2011, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org/

[11] F. Ricca, M. Torchiano, M. Di Penta, M. Ceccato, and P. Tonella, "Using acceptance tests as a support for clarifying requirements: A series of experiments," *Information & Software Technology*, vol. 51, no. 2, pp. 270–283, 2009.

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[13] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, M. Ceccato, and C. A. Visaggio, "Are fit tables really talking?: a series of experiments to understand whether fit tables are useful during evolution tasks," in *Proc. of the 30th Intl. Conf. on Software Engineering (ICSE)*, Leipzig, Germany, 2008, pp. 361–370.