Teaching Evidence-Based Software Engineering: learning by a Collaborative Mapping Study of Open Source Software

Daniela Castelluccia
University of Bari
Department of Computer Science
70125, Bari, ITALY
+39 080 5442137
daniela.castelluccia@uniba.it

Giuseppe Visaggio
University of Bari
Department of Computer Science
70125, Bari, ITALY
+39 080 5443270
giuseppe.visaggio@uniba.it

ABSTRACT

In this paper, we share our experiences about teaching evidence-based software engineering to students of a Master degree program in Computer Science. We provided a semester-long course, composed of lessons about empirical and experimental methods. It also included a collaborative project concerning a systematic mapping study of the challenges in the adoption of open source software in a business context. All students collaborated on the project by analyzing emerging results in the scientific literature. They evaluated the proposals in terms of level of novelty and evidence and delivered a complete report, which summarized the risk factors in the adoption of open source software and offers technical knowledge about evolutionary patterns and development community support, with practical implications. As a side effect, this problem-based learning approach provides a positive impact in terms of students' participation, teamwork attitude, professional interest in open source software, and exam passing.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education.* D.2.12 [**Software Engineering**]: Reusable software.

General Terms

Design, Experimentation, Evaluation.

Keywords

Evidence-based software engineering, empirical methods, systematic mapping study, systematic literature study, software engineering, open source software, education, decision making.

1. INTRODUCTION

The software engineering research community has long-argued that the software industry frequently adopts technologies without first undertaking a systematic evaluation of those technologies. At the same time, the research community has also recognized that researchers themselves are prone to propose new methodologies and tools without undertaking an empirically-based evaluation of those solutions [1, 2]. With the aim of spreading an evidence-based approach to software engineering, several universities annually conduct a teaching module about evidence-based software engineering (EBSE). Dyba [3] and Jørgensen [4] report anecdotal evidence on the success of teaching EBSE to undergraduate students at a Norwegian university. Also other researchers have reported on a preliminary study of the use of EBSE by students at a UK university [5].

In the March issue of ACM Software Engineering Notes we read a paper [6] on teaching EBSE to master students of a Turkish University. The author advocates teaching EBSE as a single lecture within a Software Architecture course instead of an entire semester-long course, with the belief that EBSE may be boring for some of the students because they would like to see topics that can be used in industry. After the 2-hours lecture, limited to procedures of systematic literature review and systematic mapping, each student selected a specific software architecture topic and individually prepared a systematic mapping study of the specific topic; this project was to be delivered at the final exam, two months later. The author states promising results because the students learned by project and were motivated to approach EBSE due to the choice of a research topic useful for their master thesis.

In this paper we share our experiences teaching EBSE to students of the Master degree program in Computer Science at the University of Bari in Italy. We assert that problem-based learning is necessary and efficient, moreover we are convinced that a strong background in the full EBSE methodology is a precious asset that will support students throughout their professional paths. With this belief, we advocate that combining a balance of lessons and projects into a semester-long teaching module is the most effective choice.

The reason can be found in the idea that EBSE is not only a collection of emerging results from the scientific literature, because systematic mappings and systematic literature reviews are only the first step aimed to provide a fair, evidence-based evaluation of an open issue and related solutions by using a trustworthy, rigorous, and auditable methodology [7]. Moreover, EBSE gives practical guidance for integrating the appraised evidence with experience, the customer's values, and circumstances in order to make decisions about technology adoption, to evaluate performance and seek ways to improve it.

In general, EBSE is not only concerned with determining what works, when and where, in terms of software engineering practice, tools and standards, but also it is one of the key factors of success for a software engineer or project manager: in fact, by means of EBSE methods, he will acquire the ability to find and assess an appropriate technology prior to inclusion of that technology in a software architecture or a process improvement program. Providing the means for decision making for future software engineers is the professionalizing objective of our teaching module on EBSE.

The main contributions of this paper are the following:

- help to promote the teaching and publicize the value of the full EBSE methodology;

- help to promote the problem-based learning in EBSE by means of mapping studies;
- evaluate the level of evidence in the current research results on open source software.

The remainder of the paper is organized as follows: the next section discusses our semester-long course on EBSE and explains the motivation and the organization of the collaborative project; section 3 reports the results of the project; section 4 provides overall insights and some directions for further research.

2. AN ENTIRE SEMESTER-LONG COURSE

At the University of Bari, we usually teach an EBSE course for students of the Master Degree program in Computer Science with two objectives:

1. Provide the ability to design and perform empirical studies in Software Engineering;
2. Provide the ability to search, summarize and formalize reusable knowledge and professional experience.

The course is semester-long and is composed of lessons that cover the following topics:

1. Empirical studies to support decision making in process and product innovation: types of empirical studies for "research in the large" (survey and retrospective analysis), "research in the small" (controlled experiment) and "research in the typical" (case study); procedures for designing and performing empirical studies, threats to validity and statistical analysis; replication of empirical studies and family of experiments; definition of the theory and practical use of empirical knowledge;
2. Methods and tools to extract, formalize and share knowledge from empirical studies: systematic literature reviews and mapping studies;
3. Technological innovation: software product innovation and software process innovation; principles, best practices and open issues in decision making; experience factory and socialization of knowledge.

In addition, in the last academic year we investigated the benefits of complementing the course with a collaborative project concerning a systematic mapping study of the challenges in the adoption of open source software (OSS) in a business context, in order to underline how strategic is the stage of evaluation and selection of an OSS. The assignment consisted of classify risk factors found in the scientific literature, searching out empirical studies and evaluating the empirical evidence of the various proposed solutions or technologies.

In detail, the project was first introduced by a lecture concerning the history of OSS and the recent legislative initiatives for its adoption, the licensing issues, the business models of OSS, several problems related to integration into the technological and the organizational context of a company that aims to adopt OSS. Also the lecture focused on the research question of the mapping study:

- identify the risk factors grouped by topic, such as the internal quality of the OSS component, the level of support from the development community, the evolutionary patterns of the OSS, and the adoption context (in terms of technological and organizational barriers).

A systematic procedure started by this research question and conducted to results then plotted in a bubble chart, which summarizes the state of the art and practice and highlight findings of the mapping study.

The students worked on the project in groups of pairs. We assigned to each group a sub-set of Relevant Papers extracted in a five-year period (2008-2012) by applying the following search string on Google Scholar:

"open source software" AND (adoption OR selection OR acquisition OR migration OR integration OR evolution) AND (risk OR factors OR barriers OR challenges OR decisions)

This search string was applied to an optimized set of conference proceedings and journals with a high impact factor and different styles of writing, in order to enrich the experience of literature analysis by the students.

	5-Year Impact Factor	# PAPERS (2008-2012)
ACM Computing Surveys	9,169	6
IEEE Transactions on Software Engineering	3,038	33
ELSEVIER – Information Science	2,984	9
Communications of the ACM	2,113	32
IEEE Software	1,443	44
ICSE & workshops – International Conference on Software Engineering		65
CSRM – Conference on Software Reuse and Maintenance		31
ICSM - International Conference on Software Maintenance		43
TOTAL		263

Table 1: Set of selected journals and conferences

A set of 121 Relevant Papers passed the screening process based on exclusion criteria, aimed at excluding not-primary studies, introductions, posters, papers with OSS citations only in references or biographies and so on.

One of the Relevant Papers was used by the teacher to show how to apply the following procedure. The remaining 120 Relevant Papers were subdivided into 12 subsets. Each subset of 10 Relevant Papers was assigned to one group and analyzed using the following steps:

1. Keywording: find keywords about context, addressed problem, proposed solution and results; these keywords are useful to support the building *in-itinere* of the classification scheme;

ID	Publisher	Title	Authors	Journal	Year	Keywords			
						Context	Problem	Solution	Results

Table 2. Fields of the table for Keywording

2. Classification: classify by
   a. Content, structured in topic, open issue and risk factor;
   b. Novelty = type of proposed solution, like overview (typical in survey), methodological approach, tool, empirical study (using metrics for validating theory or conjectures), metric (definition of new metrics);

c. Evidence = source of empirical results, like a survey, a case study, or a controlled experiment, grouped by preliminary results (field-limited or without statistical analysis), empirical results, scientific results (replication of empirical studies).

ID	CONTENT			NOVELTY	EVIDENCE	Note
	Topic	Open issue	Risk Factor			

Table 3. Fields of the table for Classification

First, students individually carried out the Keywording and the Classification of the set of 10 papers assigned to each group. Then in pairs, students were reviewers of their results and, after a meeting and discussion, they provided a single version of their classification.

By integrating results coming from the work performed by all students, a complete classification table was obtained from the mapping study and then a bubble chart was plotted for providing an interesting answer to the research question, highlighting both future directions for researchers and practical implications for practitioners.

3. RESULTS FROM MAPPING STUDY

In general, the distribution of Relevant Papers in the last 5 years shows an increase of interest from 2008 to 2009, followed by a moderately constant interest within research community. This pitch reflects the impulse in private investments to the OSS community, following the birth and approval of the GPL license: in 2009 private investments by IBM, HP, Intel, Oracle and others reached 2.5 billion dollars, so this context contributed to an increase of researchers' interest and to a doubling of the number of published papers.

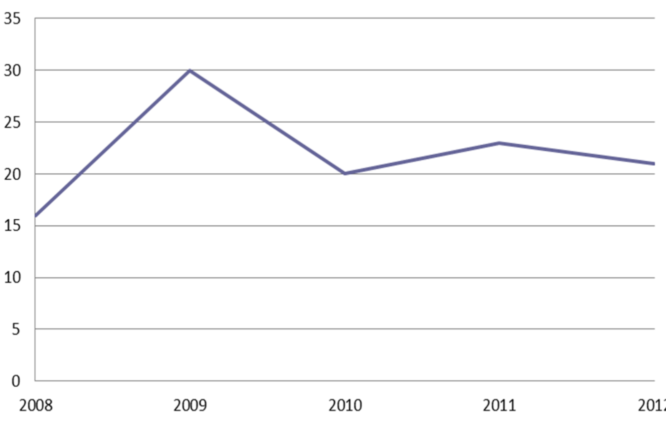


Figure 1: Trend of Relevant Papers per year

The distribution of Relevant Papers per Topic shows that the interest of the research community focused on adoption of OSS by analyzing the risk factors mainly related to OSS evolution and community support.

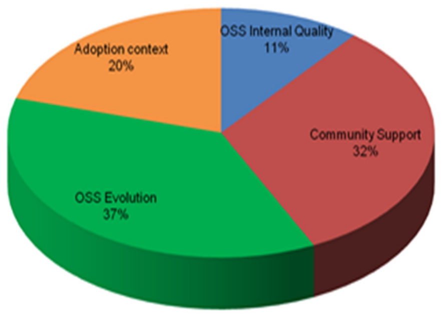


Figure 2: Distribution of Relevant Papers per Topic

Evaluating an OSS component rather than a classical software component entails risk factors which aren't correlated to specific metrics for internal quality, they are correlated to the practices of distributed development known as "open collaboration". These practices are quite different from classical software processes and, although they benefit from a higher availability of developers, they spawn problems due to coordination of distributed teams, lack of documentation, bad code reuse practices without license compatibility verification, unplanned evolution of OSS functional and non-functional properties, hard evaluation of change impact. These problems could affect the future maintainability of OSS components, so that they can be considered the main risk factors to be evaluated during OSS selection with the purpose of adoption in a business context.

These insights are confirmed by analysis of trends of Topics per year, showing the topic "OSS Evolution" to be not only the most interesting topic but also this interest has been increasing in recent years. Results shows that "prediction of properties", "aggregate metrics" and "change evolution analysis" are the most emergent open issues in OSS evolution, together with OSS integrability and licensing.

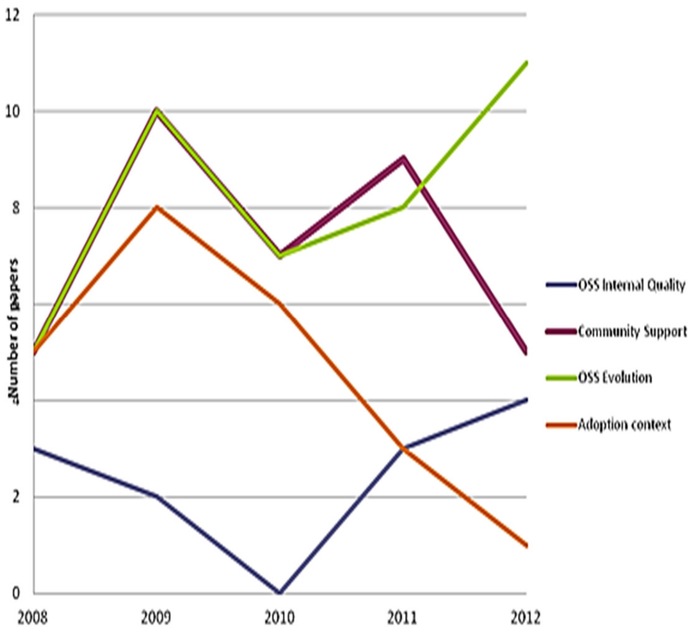


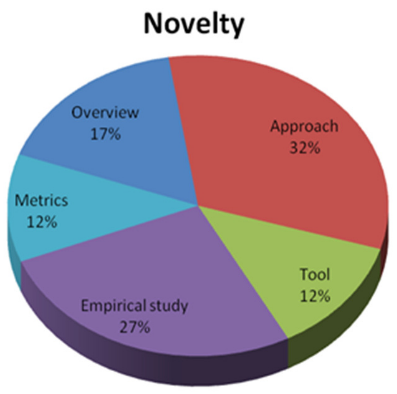
Figure 3: Trends of Relevant Papers per Topic per year

A detailed description of results from the mapping study and classification of the risk factors in the adoption of OSS in a business context will be published soon by the authors. Here we aim to highlight that, by means of EBSE and problem-based learning, students were able to acquire knowledge and practices about OSS.

Moreover, students noticed that few researchers address open issues by undertaking empirical studies in a search for reasons and peculiarities of a problem. On the contrary, researchers often avoid this step and publish papers presenting their new approach, methodology or automated tool. Only sometimes they validate the efficacy of their solutions into a limited field (applied in few OSS project). The distribution of Relevant Papers per Novelty and the distribution of Relevant Papers per Evidence show that less than 50% of papers are based on an evidence-based research approach.

Without focusing on evidence-based results and selecting the most empirically-validated proposals, the research community isn't able to assess and endorse one solution above all others. This can be called

Technological Pollution, which contributes to inundating the state of the art and the practice with numerous non-validated technologies and lengthens the path toward an effective resolution of the issues that remain "open" for years.



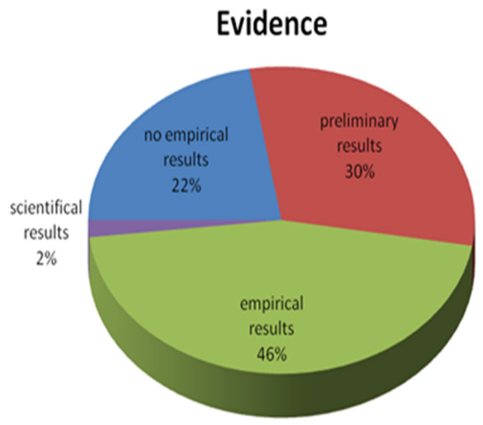


Figure 4: Distribution of Relevant Papers per Novelty and per Evidence

4. CONCLUSIONS

EBSE teaching modules are offered annually by several universities in the US and the EU. We aim to share our experiences in teaching a semester-long course on EBSE to master students in Computer Science. A combination of lessons about the full EBSE methodology and a collaborative mapping study about emergent professional topics exploits benefits from both a strong methodology background and the problem-based learning.

The risk factors in the adoption of OSS in a business context are collected with the collaboration of students, who learned to analyze several types of scientific literature, evaluate solutions in term of novelty and evidence. The students also came to appreciate EBSE as a means of evaluating product and process innovation and as a precious asset in support of their professional growth.

As a side effect, the collaborative project provides a positive impact in terms of teamwork, interest for OSS concerns and exam results. In fact, it is significant that all participants of the project passed the exam during the first session of exam calls at the end of the course.

As far as research community concerns, the results from the systematic mapping study prove that many researchers don't perform empirical studies or replications for solving the open issues. Indeed, empirical studies could be useful to search, evaluate and validate the findings and solutions just published. On the contrary, researchers often focus on producing their own methods or tools, sometimes without validating their efficacy by means of empirical examination. This research approach is not to be considered properly "scientific". It doesn't contribute to the increase of knowledge nor to the resolution of the open issues by means of empirically-validated solutions.

This lack of evidence produces a poor trustworthiness in research results by software practitioners which remain without guidance when making technology-adoption decisions in their projects or organizations. However, from an ideal consumer point of view, it would be desirable to adopt a software component as if it was an electronic tool (like a sensor for example), which usually comes supplied with a specific datasheet including features and measures of throughput, efficiency and savings. These measures are validated by experimental tests, which are reported as public results, not restricted. Applied to a software component, the ideal datasheet could include measures of size, structural complexity (coupling, cohesion...), maintenance rate, upgrade rate, vulnerability and so on. These values could be provided and validated by means of EBSE methods. In this way, we think the EBSE approach could provide a competitive advantage both for researchers and software professionals. By means of EBSE, Software Engineering could be truly an engineering science.

REFERENCES

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

[2] Höfer, A. & Tichy, W. F. 2007. Status Of Empirical Research In Software Engineering. In Basili, V. R., Rombach, D., Schneider, K., Kitchenham, B., Pfahl, D. & Selby, R. W. (Eds.) *Empirical Software Engineering Issues: Critical Assessment And Future Directions*. Springer.

[3] Dyba, T., Kitchenham, B. A. and Jorgensen, M. 2005. Evidence-Based Software Engineering for Practitioners. *IEEE Software*, 22(1), 58-65.

[4] Jorgensen, M., Dyba, T., and Kitchenham, B. 2005. Teaching Evidence-Based Software Engineering to University Students. In *Proceedings of the 11th IEEE International Software Metrics Symposium* (METRICS '05). IEEE Computer Society, Washington, DC, USA.

[5] Rainer, A., Beecham, S. & Hall, T. 2007. Assessing Undergraduate Students' Use Of Evidence Based Software Engineering. Hatfield, U.K., University Of Hertfordshire. Technical Report CS-TR-462.

[6] Catal, C., 2013. Teaching Evidence-Based Software Engineering to Master Students: A Single Lecture within a Course or an Entire Semester-Long Course? ACM SIGSOFT Software Engineering Notes, March 2013 Volume 38 Number 2.

[7] Kitchenham, B. (2004) Procedures for Performing Systematic Reviews. Keele University, Technical Report TR/SE-0401 and NICTA Technical Report 0400011T.1