Understanding the role of software metrics in an empirical software engineering course for PhD students

Letizia Jaccheri* and Sandro Morasca**

I. Context and motivation

This work was carried out in the context of an Empirical Software Engineering (ESE) course for PhD students which has been held four times at NTNU and Simula (Uio) from 2002 to 2005 [6]. The general pedagogical goals of the course are to let students learn as much as possible about ESE topics. We used Bloom's taxonomy to reflect about and assess the learning level of students. This taxonomy grades six different levels of thinking one uses when learning. At the lowest level one can remember after memorizing verbatim, while, at the highest level, one can make decisions and their rationale based on the acquired knowledge.

The reading material consists of a dozen of scientific articles and one book. It was designed in 2002 and has evolved over the years to reflect a wider variety of topics and the evolution in the field, based on the feedback obtained during the iterations of the course. The topics of the course encompass quantitative and qualitative methods for investigations and examples of applications. Software metrics are covered as a supporting field. Information about reading material and course organization is available at[5].

We have used a variety of teaching methods, from lectures to role playing, log writing, presentations, filming, dialogue sessions, and essay writing. In general we seek a balance between lecture based teaching and constructivist[2] learning.

In this ESE course we have designed a questionnaire that we have used in three iterations of the course.

II. Goals of this work

The goal of this work is to reflect on the software metrics part of the ESE course, and specifically to investigate the role of software metrics in an ESE course. What contents do students need to know? Is our reading material appropriate? And what about the teaching methods? Is the length of the class adequate?

III. The course (4th iteration) with emphasis on the software metrics part

Five PhD students signed up for the fourth iteration of the ESE course and three of them have taken the exam so far. Students were informed about the reading material one month before the course started and they were required to read it. The course is organized over three days. The first day is devoted to content understanding, motivation, and team building. The second day is devoted to software metrics, and the third one to general issues about setting up empirical investigations. There are almost no classic lectures, but the students are instead asked to prepare presentations about selected parts of the reading material and interact with the teachers. The students are asked to write personal logs after each day. Before the final exam, the students must write an essay about the main lesson learnt. Here, they must reflect about their own learning process and explain what main sources (papers, presentations, discussions, events that happened before) guided them to achieving this knowledge. The final exam is oral. Details about the list of papers, event schedule, essay, and evaluation data of the previous iterations are available at the course web site.

The software metrics part is based on [8][1][3][4][7]. These papers cover most of the basic aspects of software metrics. The entire course is organized as a seminar spanning over three full days, one of which is devoted to software metrics.

Previous iterations of the course and their evaluation show that the students' awareness of software metrics reached during the course needed need to be improved.

IV. Evaluation

The entire course is evaluated each year according to questionnaire which we distributed after the final exam. The raw data are available at the course site. No attendees mentioned "software metrics" while discussing the course goals. Examples of answers to question about course objectives are: "An overview of status, methods and theory related to empirical software engineering research, which seems to a large extent to directed at quantitative studies with statistical analysis." or "To know why we need empirical software engineering, what is empirical software engineering and learn how to do empirical studies."

To evaluate the pedagogical effects of the software metrics part of the course, we have implemented a specific evaluation in addition to the general questionnaire. This consists of a simple test (pre-test and post-test) for the metrics part of the course. In the pre-test, the students are asked to answer questions about their previous knowledge in statistics, mathematics, software metrics, and software in general, knowledge about course themes, expectations and motivations for taking this course, professional interest for the course.

In the post-test, the students are asked to discuss how their expectations have been satisfied and how their knowledge

* L. Jaccheri is with Department of Information and Computer Science, Norwegian University of Science and Technology, Trondheim, Norway. *letizia@idi.ntnu.no*

** Sandro Morasca is with Università degli Studi dell'Insubria Dipartimento di Scienze della Cultura, Politiche e dell'Informazione Via Valleggio 11, I-22100 Como, Italy, *sandro.morasca@uninsubria.it*

about the above themes (statistics, mathematics, software metrics, and software in general, knowledge about course themes) has changed during the day. The raw data for these tests are also available at the course web site.

Only three students of those taking the course have taken their final exam and only two students have passed the exam so far. As a result, we have pre- and post-tests for five students, while we have the general evaluation of the course for only two students.

In general all students have had at least one university course in mathematics and statistics. Nobody declared to have theoretical knowledge of software metrics through courses but some students have practical experience as they have participated in experiments.

The post-test consists in this question: "Discuss how your expectations have been satisfied and how your knowledge around the themes in the pre-test has changed during the day." The students gave answers such as:

- "I have gained improved consciuosness/awareness of some foundations for empirical software engineering, both as to use of statistics, formal theory, and practice (such as GQM)."
- "During the day I have acquired insight into the many pitfalls of metrics, statistics, research methods (qualitative and quantitative) and design of study.
  I have also got a insight into the different opinions and schools of the above mentioned topics. There are many different interpretations of reality of theory and how to conduct practice.
  I have to some extent broadened my knowledge of S.E. studies and related work. And also had insights into how these interconnect with each other, as well as stand in opposition to each other (expecially use of terminology).
  I have found it expecially useful to have the authors of some of the papers present as they can tell more about background and other interesting insights into the work presented."

V. Discussion

While planning the course and when evaluating the results, we mainly discuss these topics.

- **Empirical software engineering and software metrics**: what is actually the relationship between empirical software engineering and software metrics and how much of the software metrics bulk of knowledge is needed to teach and learn empirical software engineering? Should we rather require a software metrics course before admitting PhD students to empirical software engineering courses or should we merge the two topics as we do in this course?
- **Teaching strategy**: software engineering learning is often based on a project based strategy. In this course we have tried to construct collaborative exercises. Should one design such exercises for teaching software metrics to PhD students? Or should we rely on a more lecture based approach combined with presentations of papers by students?
- **Course evaluation**: the first time the empirical software engineering course was run we had a dozen of students because we recruited students from the first, second and third year. In the subsequent years, we have on average 4 students taking the exam. This means that raw data we obtain from the evaluation questionaire are valuable as it is but it does not make sense to attempt categorizations. Moreover, while there exist an established bulk of literature about education evaluation at lower education level, we are still not aware of established methods to evaluate the effectiveness of university courses expecially at PhD levels.

These are the topics we wish to bring as a contribution to the workshop.

References

[1] Lionel C. Briand, Sandro Morasca, and Victor R. Basili. An operational process for goal-driven definition of measures. *IEEE Transactions on Software Engineering*, 12(4):1106–1125, 2002.

[2] Jacqueline Grennon Brooks and Martin G. Brooks. *In Search of Understanding: The Case for Constructivist Classrooms*. Association for Supervision and Curriculum Development, 1993. ISBN: 0871202115.

[3] Norman Fenton. Software Measurement: A Necessary Scientific Basis. *IEEE Trans. on Software Engineering*, 20(3):199–206, March 1994.

[4] Alfonso Fuggetta, Luigi Lavazza, Sandro Morasca, Stefano Cinti, Giandomenico Oldano, and Elena Orazi. Applying GQM in an industrial software factory. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 7(4):441–448, 1998.

[5] Letizia Jaccheri. Empirical software engineering course web site. web: http://www.idi.ntnu.no/∼letizia/empse2005, September 2005.

[6] Letizia Jaccheri and Thomas Osterlie. Can we teach Empirical Software Engineering? In *11 International Symposium on Software Metrics, Como, Italy*. IEEE-CS Press, September 2005.

[7] J. Miller, J. Daly, M. Wood, M. Roper, and A. Brooks. Statistical Power and its subcomponents – Missing and Misunderstood Concepts in Empirical Software Engineering Research. Technical report, University of Strathclyde, Glasgow, UK, 1996. Report no. ISERN-96-12.

[8] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, 2000.