Prog Artif Intell (2016) 5:307–314
DOI 10.1007/s13748-016-0092-2

REGULAR PAPER

Data mining for software engineering and humans in the loop

Leandro L. Minku[1] · **Emilia Mendes**[2] · **Burak Turhan**[3]

Received: 8 March 2016 / Accepted: 29 March 2016 / Published online: 16 April 2016

Abstract The field of data mining for software engineering has been growing over the last decade. This field is concerned with the use of data mining to provide useful insights into how to improve software engineering processes and software itself, supporting decision-making. For that, data produced by software engineering processes and products during and after software development are used. Despite promising results, there is frequently a lack of discussion on the role of software engineering practitioners amidst the data mining approaches. This makes adoption of data mining by software engineering practitioners difficult. Moreover, the fact that experts' knowledge is frequently ignored by data mining approaches, together with the lack of transparency of such approaches, can hinder the acceptability of data mining by software engineering practitioners. To overcome these problems, this position paper provides a discussion of the role of software engineering experts when adopting data mining approaches. It also argues that this role can be extended to increase experts' involvement in the process of building data mining models. We believe that such extended involvement is not only likely to increase software engineers' acceptability of the resulting models, but also improve the models themselves. We also provide some recommendations aimed at increasing the success of experts involvement and model acceptability.

Keywords Data mining · Machine learning · Software engineering · Software analytics

1 Introduction

The twenty-first century has been experiencing a rapid growth in the amount of data produced by sensors, processes and activities. Data are everywhere. They are produced by cameras monitoring public pathways, sensors monitoring industrial machinery, customers making purchases in supermarkets, financial markets, hospital logs, elderly patients at home, etc. Data have also been (and are being) collected from software engineering processes and products, during and after software development. We refer to these data as software data.

Software data have the potential to provide useful insights into how to improve software engineering processes and software itself, supporting decision-making. For instance, they can be used to gain insights into what software modules are most likely to contain bugs [15,34], what amount of effort is likely to be required to develop new software projects or Web applications [12,29], what software changes are most likely to induce bugs [2,19], how the productivity of a company changes over time [37], etc.

The data availability combined to the difficulty of manually browsing data to retrieve knowledge and insights resulted in the emergence of research communities exploring the use and development of data mining approaches for software engineering. For instance, the Working Conference on Mining Software Repositories (MSR) has experienced an

✉ Leandro L. Minku
leandro.minku@leicester.ac.uk

Emilia Mendes
emilia.mendes@bth.se

Burak Turhan
turhanb@computer.org

[1] Department of Computer Science, University of Leicester, Leicester LE1 7RH, UK

[2] Blekinge Institute Technology, 371 79 Karlskrona, Sweden

[3] Department of Information Processing Science, University of Oulu, POB.3000, 90014 Oulu, Finland

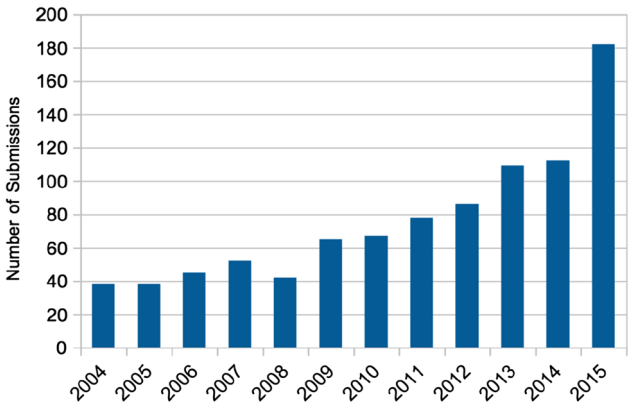

Fig. 1 Number of paper submissions to the Working Conference on Mining Software Repositories since its first chapter in 2004 until its most recent chapter in 2015

increase in the number of paper submissions since its first chapter in 2004 (see Fig. 1).

Despite the promising results being achieved in the field of data mining for software engineering, there is frequently a lack of discussion on the role of software engineering practitioners amidst the data mining approaches. Such limitation can make industrial adoption of data mining for software engineering difficult. Moreover, despite the fact that software engineers have valuable knowledge, such knowledge is usually ignored by data mining approaches. Together with the lack of transparency of the models created by most data mining approaches and of the data mining approaches themselves, this can be an additional factor to hinder the acceptability of data mining by software engineering practitioners.

To overcome these problems, this position paper provides a discussion of the role of software engineering experts when adopting data mining approaches. It also argues that this role can be extended to increase experts' involvement in the process of building data mining models. We believe that this extended involvement can (1) increase the acceptability of data mining approaches in software engineering in practice, and (2) further improve the field of data mining for software engineering by facilitating the integration of knowledge automatically retrieved from data with human knowledge. We also provide some recommendations aimed at increasing the success of software engineering experts involvement and model acceptability.

This paper is further organised as follows. Section 2 provides an overview of current work on data mining for software engineering. Section 3 discusses the current role of software engineering experts in data mining for software engineering. Section 4 argues in which ways software engineering experts' participation could be increased. Section 5 provides recommendations for involving software engineering practitioners. Section 6 concludes the paper with final remarks.

2 Current work on data mining for software engineering

Data mining has been used for several software engineering problems. This section provides a brief overview of work done in three of the software engineering problems most studied from the data mining perspective: software effort estimation, software defect prediction, and prediction of bug-inducing software changes.

2.1 Software effort estimation

Software effort estimation is the task of estimating the effort (e.g. in person-hours, person-months) required to develop a software project. It is a task of strategic importance for software companies, given that effort is the main contributing factor for project cost. For instance, overestimations could cause companies to waste resources or loose bids for projects. Underestimations could cause companies to be unable to complete software projects.

Data mining can be used to create software effort estimation models based on data describing previously completed software projects. These data may contain project features such as estimated software size, team expertise, programming language, memory requirements, etc, besides the actual effort required to develop the completed software projects.

There has been more than 400 studies in the effort estimation field, investigating new prediction techniques, and or comparing techniques. Jorgensen and Shepperd provide details on a mapping study in this topic [16]. A seminal work is that of Boehm, who proposed a regression-based model called COnstructive COst MOdel (COCOMO) [7]. This approach learns an equation for estimating effort, which can be easily interpreted by software engineering experts. Another landmark study is the work of Shepperd and Schofield, who used k-Nearest Neighbours for software effort estimation [46]. Their approach was shown to usually outperform more traditional stepwise regression models. Chulani et al. proposed to use a Bayesian approach to combine a priori information based on expert knowledge with a linear regression model based on log transformation of the data [10]. Their approach showed promising results, outperforming linear regression models based on log transformed data. However, their study was based on a single data set and its extended version. It is difficult to know how much the results would generalise to other software projects.

Dejaeger et al. provide a comparison of several machine learning approaches applied to software effort estimation [12]. Their results show that ordinary least squares regression in combination with logarithmic transformation obtain competitive results. Studies involving ensembles of learning machines have also been showing promising results when applied to software effort estimation [23,36]. In particu-

lar, Minku and Yao showed that combining the power of ensembles with local learning through the use of bagging ensembles of regression trees outperformed several other machine learning approaches [36]. A problem of ensemble-based approaches is that they are difficult to interpret by software engineering experts. Alternatively, despite not being the best ranked approach for software effort estimation, regression trees can also obtain competitive results. These interpretable models have shown to be rarely considerably worse than the best approach for a given software effort estimation data set [36].

Software effort estimation has also been investigated as a transfer (cross-company) learning problem [22,37,52]; an online or incremental learning task, where more software project data are used for training over time [24,35,37]; a problem where estimations for latter phases of the project can be improved based on the actual effort spent on previous phases [13,25,51]; a multi-objective learning problem [38,45], and a semi-supervised learning problem [21].

Finally, many of the techniques that were applied to software effort estimation have also been used for Web effort estimation [4,27], which also include the use of ensembles [5]. Mendes pioneered this field, and also led the creation of the Tukutuku database, which is to date the only cross-company database on Web project data [31]. The Tukutuku database has been used in numerous studies to compare different effort prediction techniques. Mendes has also applied one specific technique Bayesian Network, to building Web effort prediction models using as basis expert knowledge [27]. Further details are given in Sect. 3.

2.2 Software defect (bug) prediction

In 2002 IEEE Metric Panel, a group of noted researchers have agreed that fixing defects in a software product after being delivered to the customer is up to 100 times more expensive than finding and fixing them during the requirements and design phases [8,47]. They have also argued that up to 50 % of effort is spent on avoidable work, 80 % of which comes from a small number of defects (i.e. 20 %) in the system. The bottom-line is that software testing is a costly challenge and practitioners seek the knowledge of where the defects might exist before they start testing.

In this respect, defect predictors are data mining applications to help prioritising the list of software modules to be tested, to allocate limited testing resources effectively and to detect as many defects as possible with minimum effort.

Software defect prediction has been a popular area of software quality research that has drawn the attention of significant organisations including, but not limited to, Microsoft, NASA and AT&T [32,41,43]. Basically, software defect prediction models require a set of features to characterize the problem and to give estimation on the defect proneness of the system. In software quality, these attributes are referred to as software metrics and numerous previous studies demonstrated defect predictors learned from product [32] (e.g. size, complexity) and process [43,53] (e.g. code churn) metrics.

Once relevant data are available, a variety of data mining algorithms can be applied to learn defect predictors, please see Hall et al. for a systematic literature review [15]. Most defect prediction studies formulate the problem as a supervised learning problem, where the outcomes of a defect predictor model depend on historical data used for training. They can be either labels indicating that a software module is or is not likely to contain defects, or the predictive number of defects expected to be present in the software module, or a ranking of software modules according to their defect proneness.

Majority of research focus on the algorithmic models and report simulation results of defect predictors that are trained on a project and tested on a reserved portion of the same project, i.e. retrospective analyses, or the application of defect predictors to the newer versions of the same project in terms of longitudinal case studies [50,53]. These attempts for defect prediction modelling assume the availability of local project data (i.e. within project predictors). In other words, building data mining models requires a project to have a historical data repository, where project metrics and defect information from past are stored. However, this is rarely the case in reality.

To address this issue, recently a branch of defect prediction research emerged that makes use of transfer learning and deals with cross-project predictors, where the goal is to learn a predictor model from a project and then to apply the model to another project [9]. Cross-project defect prediction is a challenge with important practical aspects. One such practical aspect is that cross-project predictions may enable practitioners to use the available open-source project data for defect prediction [54], without making big changes in or investments to their existing processes for data collection, and process improvement activities. Existing studies provide empirical evidence over a wide range of software systems, advocating that cross-project defect predictors can be effective. Considering that the idea behind cross-project prediction is to make estimates of faulty locations in projects with no history, it is a viable stop-gap choice [52] in data starving project environments.

2.3 Prediction of bug-inducing software changes

More recently, researchers have started to investigate the use of data mining for predicting whether software code changes are likely to induce bugs [19]. This type of approach allows risky changes to be identified immediately after the commit of the code related to the changes take place, rather than having to wait a whole software module (e.g. software class)

to finish being implemented. As the changes have just been committed, the context of the changes is still fresh in the developer's mind, being much easier to investigate for finding bugs.

The problem of predicting bug-inducing changes can be formulated in different ways. Existing work has investigated predictions at the individual change level [19,20], and at the level of initial modification requests [40], which may consist of several changes. Researchers have also investigated prediction of whether commits are likely to lead to crash-related bugs [2], i.e. bugs that result in an unexpected interruption of the software system in users' environment.

Models for predicting bug-inducing changes for a given software can be created based on data describing previous changes for this software, which can be obtained when using version control systems. Input attributes describing changes can be change metrics [19,40], such as the number of modified directories, the distribution of modified code across each file, number of lines of code added/deleted, whether the commit is a bug fix change, number of developers that modified the changed files in the past, developer experience, etc. Existing work has also used code complexity metrics such as the ones typically used for software defect prediction (Sect. 2.2) and social network analysis metrics computed based on the dependency among the changed files [2].

The data mining approaches used for these predictive tasks include logistic regression [3,19], general linear model, naive Bayes and random forest [40], support vector machines [20] and k-nearest neighbours [3].

3 The current role of software engineering experts

This section outlines the main current roles of software engineering experts in different phases of the application of data mining for software engineering.

3.1 Problem definition

An important role of the experts is to help defining the software engineering predictive problem itself. To address problems that are relevant to software engineering practice, qualitative research in the form of interviews and questionnaires can be performed to understand software engineering practitioners' needs. As explained by Runeson and Host [44], software engineering case studies differ from other areas in that the study objects are developing rather than using software, they are project oriented rather than line or function oriented, and the subjects have advanced software engineering knowledge, rather than being people performing routine work. Interviews and questionnaires should be prepared with that in mind.

3.2 Data collection

Software engineering experts also play an important role in creating and providing data that can be used by data mining approaches. They have useful knowledge regarding data quality, which can be provided for data mining experts to decide how to best process the data before applying data mining approaches. As explained by Bener et al. [6], data mining professionals can decide whether or not to include certain parts of the data in the training set based on software engineering experts' knowledge. Given that poor data quality is likely to result in poor predictive models, software engineering experts may also have the key knowledge to identify the reasons for possibly poorly performing predictive models.

3.3 Model building

A few studies use software engineering expert knowledge during data-driven model building, which aims to select the best choice of data/variables/relationships based on their expertise [10,30]. However, most work on data mining for software engineering implicitly assumes that the role of the software engineering experts in creating predictive models should be minimised. It is typically considered that software engineering experts should be able to press a button that will not only build, but also automatically fine-tune the parameters of data mining approaches. Software engineering practitioners are unlikely to have the data mining knowledge required to fine-tune parameters of certain data mining approaches, and such parameters can significantly affect the performance of software engineering predictive models [48]. Therefore, studies on optimisation algorithms to automatically tune parameters [11,42,49] are likely to be useful in practice.

3.4 Model usage and decision-making

The main purpose of data mining for software engineering is to create models which are able to provide actionable insight into support decision-making related to software [14]. In this context, the next role of software engineering experts after the predictive models are created is to use the predictive models to support decision-making. Software engineers can use predictive models in different ways. For instance, they can use isolated predictions for a given set of input attributes. For example, several studies investigated the use of probabilistic models for decision-making in software engineering, within the context of effort estimation [26,27]. They may also try several different sets of input attributes to find which of them would lead to the most desirable outcome. When predictive models are transparent, software engineering experts may also use them to gain insights into significant correla-

tions between input attributes and the variable of interest. The transparency of predictive models is also likely to increase their acceptance by software engineering experts, as experts can understand the model and how it makes predictions. However, care must be taken not to confuse correlations with causations. We believe that software engineering experts can use their domain knowledge to explain whether certain correlations are likely to represent causations or not.

4 The potential role of software engineering experts

Even though it is usually assumed that the software engineering experts' role in model creation should be minimised, we argue that, in software engineering:

1. involving experts in the process of building predictive models is likely to increase the acceptability of data mining in software engineering and
2. experts have valuable knowledge that can improve predictive models, further improving the field of data mining for software engineering.

A few studies can be used to support our arguments [6,10,18,26,27]. One of the authors of this paper (Mendes) had the opportunity to collaborate with six different companies in New Zealand and Brazil building expert-based Web effort estimation models based on Bayesian networks [26,27]. The software engineering experts who took part in the case studies were all project managers of well-established Web companies in either Auckland (New Zealand) or Rio de Janeiro (Brazil), each with at least 10 years of experience in project management. These companies varied in their size, measured as the total number of employees. In addition, all six companies were consulting companies and as such, developed a wide range of Web applications, from static and multimedia-like to very large e-commerce solutions.

When approached, all six companies were looking at improving their current effort estimates, and agreed to participate in the study for two main reasons: (i) because the models to be created were geared towards their specific needs; and (ii) because their expertise and participation were acknowledged as essential to eliciting the models [26,27]. This shows how important the inclusion of software engineering experts in the process of building models is to increase their openness to the use of data mining. Given that many software engineers will have been working in their field of several years, we believe that the importance of acknowledging their expertise to support model building also extends to other software engineering tasks than Web effort estimation. Even though software engineers may lack the expertise to tune parameters of certain data mining approaches, other data mining approaches such as Bayesian networks lend themselves for domain experts input.

Once the study finished, all companies except for the company in Brazil were contacted for postmortem interviews. The interviews revealed that not only the resulting models but also the process of building the models was itself advantageous to the companies. This is because the process enabled software engineering experts to think deeply about their effort estimation process and the factors taken into account during that process. This has been pointed out by all the software engineering experts interviewed after the study finished. All the companies remained positive and very satisfied with the results of the study once it finished. In particular, the software engineering experts from the largest company in terms of number of employees presented their effort estimation models to their development teams and asked them to adopt the model in all of their effort estimations. We believe that the successful development of these six Web effort Bayesian network models was thanks to the involvement of software engineering experts in the loop. It was greatly influenced by the commitment of the participating companies, and also by the software engineering experts' experience in estimating effort.

The Bayesian networks from the study above were created entirely based on expert knowledge. However, given that software engineering experts can have their estimations influenced by irrelevant and misleading information [18], knowledge acquired from data could be used to improve expert-based predictive models further. Knowledge acquired from data could also be used to improve upon less experienced software engineers' knowledge. Meanwhile, software engineering is a domain where there is relatively limited data [6]. It is not uncommon for datasets produced by a given company to contain less (or much less) than 50 projects [1,17,31,33,39]. Therefore, models created solely based on data can perform poorly. Knowledge from software engineering experts can be used to overcome the problem of little data [10]. Carefully combining expert knowledge with knowledge automatically retrieved from data could bring the best of both worlds in software engineering.

5 Recommendations for practitioners involvement

Based on previous experiences, we would like to share some lessons gained in terms of involving software engineering experts and ensuring successful adoption of the resulting models by practitioners.

5.1 Initial engagement

To reach out to industry, researchers can organise seminars to provide an introduction to the approaches to be used, how

they use software engineering experts knowledge and what can be achieved by companies using such approaches. These seminars can help practitioners to understand the value of the approach. This has been successfully done in previous work [26,27], where companies saw the immediate value in expert-based Bayesian networks, in particular because it enabled the very close and fundamental participation of in-house software engineering experts while building and validating a company-specific model.

5.2 Experts data collection

Besides collecting software engineering experts' knowledge through meetings, interviews and surveys, recent work has also successfully used decision-support tools as a way to collect data on software engineering experts decisions [28]. For the software engineers, such tools can be designed with the external purpose of helping software engineers to organise their tasks, visualise data, record a diary of decisions, etc. For the data miners, these tools can then collect and process data produced by the software engineering experts. An analogy of such tools can be made with software bug report tools, which can be used to collect information about bugs from software.

5.3 Initial results

As highlighted by Bener et al. [6], it is advisable to involve software engineering experts in discussions about the initial results achieved with the models and any concerns related to them. Given software engineering experts' knowledge about the problem and the data being used, they may be able to provide a clear explanation for poor performance or results that are unexpected to the researcher.

5.4 Model improvement

Once a model has been validated and put into use, it is important to obtain feedback from developers and managers on its use, as the model may need to be updated at some point. Those who have participated in building the model should ideally be the ones engaged in any model updates that take place.

5.5 Company-wide model availability

Previous research building expert-based effort prediction models in collaboration with several software companies [27] suggests that once such models have been built and validated, it is important that they do not remain within the boundaries of a single development team and project manager (assuming the company has several development teams and project managers). Our anecdotal evidence from postmortem interviews with some of the companies with whom one of the authors collaborated building such models provided us with a range of concrete and industry-informed choices that can be used for that. For instance:

- To increase the chances of successful adoption of a model, it is advisable to presenting a seminar to all developers and project managers who participate in the development and management of the types of applications that were the focus of the expert-based model built. One of the goals of this seminar is to elaborate on the value that the entire company can gain from using such models.
- We suggest that the seminar focuses around presenting and detailing the model and various what-if scenarios based on their most recent projects for which the model was used. This provides concrete examples of how the model is being used in the company. It is in our view also important to detail all the factors and categories that were defined by the software engineering experts who participated in building the model, such that all those attending the seminar become familiar with the terminology.
- Once the seminar takes place, the documentation relating to the model (description of factors and how to use the model) should become available for all the participants. The tool that is used to run the model and the model itself should also be made available to all development teams, so they can all run what-if scenarios obtained using the model.
- The nomenclature that was defined in the model should also become a common vocabulary for all teams, to be used whenever they need to discuss anything relating to the models. This is quite important as it guarantees the model's uptake by all relevant developers and managers.
- Depending on the purpose of the model being built, the model can also be presented to clients of the software company. For example, software effort estimation models can be presented as a way to provide clients with reassurance that the effort estimates being put forward are not simply guesses. Project managers and/or requirements analysts can take the model to requirements elicitation meetings and use it as a guide to obtain some of the evidence to be entered in the model to obtain an effort estimate. Such approach can be effective and help making the elicitation meetings focused, in particular whenever clients want quick cost estimates based on very short elicitation meetings.
- Seminar to other branches and/or events on best practices can be organised within the company as part of a wider strategy to use such modelling approach. In addition, such model, or experiences from using it, can also be presented at industry events as examples of best practice.

6 Conclusions

The current role of software engineering experts is usually not discussed in papers on data mining for software engineering, making adoption of approaches developed in this field difficult in practice. To fill in this gap, this paper provided a discussion on the role of software engineering experts when adopting data mining approaches. Even though this role ranges from problem definition to decision-making, there is a lack of involvement of software engineering experts in the process of building data models. This lack of involvement is a hindering factor when it comes to the acceptability of data mining approaches by software engineering practitioners.

We argue that the involvement of software engineering experts in the process of building data models is likely to not only help increasing acceptability of data mining for software engineering in practice, but also to improve the resulting data models themselves. This is because data mining for software engineering is a particular field in the sense that (1) many software engineering experts will have valuable knowledge that can be used for performing software engineering tasks, despite being potentially affected by irrelevant information, and (2) several software engineering tasks have a certain level of data scarcity. Knowledge acquired from data can help to overcome potential mistakes made by software engineering experts and provide useful insights that they may otherwise not have identified. At the same time, experts' knowledge can help to overcome the problems resulting from little data. Our argument is supported by previous successful collaborations with industry and papers containing initial results on integrating software engineering experts' knowledge with knowledge acquired from data. We also share some lessons gained in terms of involving software engineering experts in data mining studies and ensuring successful adoption of the resulting models by practitioners.

References

1. Albrecht, A., Gaffney, J.E.J.: Software function, source lines of code, and development effort prediction: a software engineering. IEEE Trans. Softw. Eng. **9**(6), 639–648 (2016)
2. An, L., Khomh, F.: An empirical study of crash-inducing commits in mozilla firefox. In: Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE), pp. 5.1–5.10 (2015)
3. Aversano, L., Cerulo, L., Del Grosso, C.: Learning from bug-introducing changes to prevent fault prone code. In: Proceedings of the International Workshop on Principles of Software Evolution, pp. 19–26 (2007)
4. Azhar, D., Mendes, E., Riddle, P.: A systematic review of web resource estimation. In: Proceedings of the 8th International Conference on Predictive Models in Software Engineering (PROMISE), pp. 49–58 (2012)
5. Azhar, D., Riddle, P., Mendes, E., Mittas, N., Angelias, L.: Using ensembles for web effort estimation. In: Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 173–181 (2013)
6. Bener, A., Misirli, A., Caglayan, B., Kocaguneli, E., Calikli, G.: The Art and Science of Analyzing Software Data: Analysis Patterns, chap. Morgan Kaufmann, Lessons Learned For Software Analytics in Practice (2015)
7. Boehm, B.W.: Software engineering economics. IEEE TSE **10**(1), 4–21 (1984)
8. Boehm, B.W., Basili, V.R.: Software defect reduction top 10 list. IEEE Comput. **34**(1), 135–137 (2001)
9. Briand, L.C., Melo, W.L., Wst, J.: Assessing the applicability of fault-proneness models across object-oriented software projects. IEEE Trans. Softw. Eng. **28**(7), 706–720 (2002)
10. Chulani, S., Bohem, B., Steece, B.: Bayesian analysis of empirical software engineering cost models. IEEE Trans. Softw. Eng. **25**(4), 573–583 (1999)
11. Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., Mendes, E.: Using tabu search to configure support vector regression for effort estimation. Empirical Softw. Eng. **18**(3), 506–546 (2013)
12. Dejaeger, K., Verbeke, W., Martens, D., Baesens, B.: Data mining techniques for software effort estimation: a comparative study. IEEE Trans. Softw. Eng. **38**(2), 375–397 (2012)
13. Ferrucci, F., Gravino, C., Sarro, F.: Exploiting prior-phase effort data to estimate the effort for the subsequent phases: a further assessment. In: Proceedings of the 10th International Conference on Predictive Models in Software Engineering (PROMISE), pp. 42–51 (2014)
14. Gall, H., Menzies, T., Williams, L., Zimmermann, T.: Software Development Analytics (Dagstuhl Seminar 14261). Dagstuhl Rep. **4**(6), 64–83 (2014)
15. Hall, T., Beecham, S., Bowes, D., Gray, D., Counsell, S.: A systematic literature review on fault prediction performance in software engineering. IEEE Trans. Softw. Eng. **38**(6), 1276–1304 (2012)
16. Jorgensen, M., Shepperd, M.: A systematic review of software development cost estimation studies. IEEE Trans. Softw. Eng. **33**(1), 33–53 (2007)
17. Jureczko, M., Madeyski, L.: Towards identifying software project clusters with regard to defect prediction. In: Proceedings of the 6th International Conference on Predictive Models in Software Engineering, PROMISE '10, pp. 9:1–9:10. ACM, New York (2010). doi:10.1145/1868328.1868342
18. Jrgensen, M., Grimstad, S.: The impact of irrelevant and misleading information on software development effort estimates: a randomized controlled field experiment. IEEE Trans. Softw. Eng. **37**(5), 695–707 (2011)
19. Kamei, Y., Shihab, E., Adams, B., Hassan, A., Mockus, A., Sinha, A., Ubayashi, N.: A large-scale empirical study of just-in-time quality assurance. IEEE Trans. Softw. Eng. **39**(6), 757–773 (2013)
20. Kim, S., Whitehead Jr., E., Zhang, Y.: Classifying software changes: clean or buggy? IEEE Trans. Softw. Eng. **34**(2), 181–196 (2008)
21. Kocaguneli, E., Cukic, B., Menzies, T., Lu, H.: Building a second opinion: learning cross-company data. In: Proceedings of the 9th International Conference on Predictive Models in Software Engineering, pp. 12.1–12.10 (2013)
22. Kocaguneli, E., Menzies, T., Mendes, E.: Transfer learning in effort estimation. Empirical Softw. Eng. **20**(3), 813–843 (2014)

23. Kultur, Y., Turhan, B., Bener, A.: Ensemble of neural networks with associative memory (ENNA) for estimating software development costs. Knowl. Based Syst. **22**, 395–402 (2009)
24. Lokan, C., Mendes, E.: Applying moving windows to software effort estimation. In: International Symposium on Empirical Software Engineering and Measurement, pp. 111–122. Lake Buena Vista, Florida (2009)
25. MacDonell, S., Shepperd, M.: Using prior-phase effort records for re-estimation during software projects. In: Proceedings of the Software Metrics Symposium, pp. 73–86 (2003)
26. Mendes, E.: Using knowledge elicitation to improve web effort estimation: lessons from six industrial case studies. In: Proceedings of the International Conference on Software Engineering, pp. 1112–1121 (2012)
27. Mendes, E.: Practitioner's Knowledge Representation: A Pathway to Improve Software Effort Estimation. Springer, New York (2014)
28. Mendes, E.: Estimating the value of decisions relating to managing and developing software-intensive products: talk at CREST open workshop on predictive modelling for software engineering (2015). http://crest.cs.ucl.ac.uk/cow/44/videos/mendes_cow44_720p.mp4
29. Mendes, E., Mosley, N.: Web Engineering. Springer Science & Business Media, New York (2006)
30. Mendes, E., Mosley, N.: Bayesian network models for web effort prediction: a comparative study. IEEE Trans. Softw. Eng. **34**(6), 723–737 (2008)
31. Mendes, E., Mosley, N., Counsell, S.: Investigating web size metrics for early web cost estimation. J. Syst. Softw. **77**(2), 157–172 (2005)
32. Menzies, T., Greenwald, J., Frank, A.: Data mining static code attributes to learn defect predictors. IEEE Trans. Softw. Eng. **33**(1), 2–13 (2007)
33. Menzies, T., Krishna, R., Pryor, D.: The promise repository of empirical software engineering data (2015). http://openscience.us/repo
34. Menzies, T., Milton, Z., Turhan, B., Cukic, B., Jiang, Y., Bener, A.: Defect prediction from static code features: current results, limitations, new approaches. Autom. Softw. Eng. **17**(4), 375–407 (2010)
35. Minku, L., Yao, X.: Can cross-company data improve performance in software effort estimation? In: Proceedings of the 8th International Conference on Predictive Models in Software Engineering (PROMISE), pp. 69–78 (2012)
36. Minku, L., Yao, X.: Ensembles and locality: Insight on improving software effort estimation. Inform. Softw. Technol. **55**(8), 1512–1528 (2013)
37. Minku, L., Yao, X.: How to make best use of cross-company data in software effort estimation? In: Proceedings of the 36th International Conference on Software Engineering, pp. 446–456 (2014)
38. Minku, L.L., Yao, X.: Software effort estimation as a multiobjective learning problem. ACM Trans. Softw. Eng. Methodol. **22**(4), 35.1–35.32 (2013)
39. Miyazaki, Y., Terakado, M., Ozaki, K., Nozaki, H.: Robust regression for developing software estimation models. J. Syst. Softw. **27**(1), 3–16 (1994)
40. Mockus, A., Weiss, D.M.: Predicting risk of software changes. Bell Labs Tech. J. **5**(2), 169–180 (2000)
41. Nagappan, N., Ball, T.: Use of relative code churn measures to predict system defect density. In: Proceedings of the International Conference on Software Engineering, pp. 284–292 (2005). doi:10.1145/1062455.1062514
42. Oliveira, A.L., Braga, P.L., Lima, R., Cornelio, M.L.: Ga-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. Inform. Softw. Technol. **52**, 1155–1166 (2010)
43. Ostrand, T.J., Weyuker, E.J., Bell, R.M.: Automating algorithms for the identification of fault-prone files. In: D.S. Rosenblum, S.G. Elbaum (eds.) Proceedings of the International Symposium on Software Testing and Analysis, pp. 219–227. ACM (2007)
44. Runeson, P., Host, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Eng. **14**, 131–164 (2009)
45. Sarro, F., Petrozziello, A., Harman, M.: Multi-objective software effort estimation. In: Proceedings of the International Conference on Software Engineering (2016) (**to appear**)
46. Shepperd, M., Schofield, C.: Estimating software project effort using analogies. IEEE Trans. Softw. Eng. **23**(12), 736–743 (1997)
47. Shull, F., Basili, V., Boehm, B., Brown, A.W., Costa, P., Lindvall, M., Port, D., Rus, I., Tesoriero, R., Zelkowitz, M.: What we have learned about fighting defects. In: VIII International Symposium on Software Metrics, pp. 249–258. IEEE Computer Society, Washigton, DC (2002). doi:10.1109/METRIC.2002.1011343
48. Song, L., Minku, L.L., Yao, X.: The impact of parameter tuning on software effort estimation using learning machines. In: Proceedings of the 9th International Conference on Predictive Models in Software Engineering (PROMISE), pp. 9.1–9.10 (2013)
49. Tantithamthavorn, C., McIntosh, S., Hassan, A., Matsumoto, K.: Automated parameter optimization of classification techniques for defect prediction models. In: Proceedings of the International Conference on Software Engineering (2016) **(to appear)**
50. Tosun, A., Bener, A.B., Turhan, B., Menzies, T.: Practical considerations in deploying statistical methods for defect prediction: a case study within the turkish telecommunications industry. Inform. Softw. Technol. **52**(11), 1242–1257 (2010)
51. Tsunoda, M., Kamei, Y., Toda, K., Nagappan, M., Fushida, K., Ubayashi, N.: Revisiting software development effort estimation based on early phase development activities. In: Proceedings of the 10th IEEE Working Conference on Mining Software Repositories, pp. 429–438 (2013)
52. Turhan, B., Menzies, T., Bener, A., Di Stefano, J.: On the relative value of cross-company and within-company data for defect prediction. Empirical Softw. Eng. **14**(5), 540–578 (2009)
53. Weyuker, E.J., Ostrand, T.J., Bell, R.M.: Do too many cooks spoil the broth? using the number of developers to enhance defect prediction models. Empirical Softw. Eng. **13**(5), 539–559 (2008)
54. Zimmermann, T., Nagappan, N., Gall, H.C., Giger, E., Murphy, B.: Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In: van Vliet, H., Issarny, V. (eds.) Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp. 91–100. ACM (2009)