2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)

Using Q&A Websites as a Method for Assessing Systematic Reviews

Bruno Cartaxo*†, Gustavo Pinto‡, Danilo Ribeiro*, Fernando Kamei±,
Ronnie E. S. Santos*, Fábio Q. B. da Silva*, Sérgio Soares*
IFPE, Brazil† UFPE, Brazil* IFPA, Brazil‡ IFAL, Brazil±
{bfsc, dmr, ress, scbs, fabio}@cin.ufpe.br, gustavo.pinto@ifpa.edu.br, fernando.kenji@ifal.edu.br

***Abstract*—Questions and Answers (Q&A) websites maintain a long history of needs, problems, and challenges that software developers face. In contrast to Q&A websites, which are strongly tied to practitioners' needs, there are systematic reviews (SRs), which, according to recent studies, lack a connection with software engineering practice. In this paper, we investigate this claim by assessing to what extent systematic reviews help to solve questions posted on Q&A websites. To achieve this goal, we propose and evaluate a coverage method. We applied this method to a set of more than 600 questions related to agile software development. Results suggest that 12% of the related questions were covered. When considering specific agile methods, the majority of them have coverage below 50% or were not covered at all. We also identified 27 recurrent questions.**

I. Introduction

Questions and Answers (Q&A) websites empowered software developers to increase the pace of learning, allowing them not only to be more productive and more effective, but also more fulfilled [1], [2]. The software engineering (SE) community has long recognized the importance of these websites, and produced contributions related to both social aspects (*e.g.,* personality traits [3], reputation [4], and gender [5]) and technical aspects (*e.g.,* documentation [6], debugging [7], or even energy consumption [8]) of software development.

On the other hand, there are systematic reviews[1] (SRs), which aim at synthesizing the best existing research to practice [9]. Unfortunately, some researchers argue that there is a lack of connection between a significant number of systematic reviews and software engineering practice [10], [11].

In this paper, we investigate how systematic reviews are connected with SE practice. To achieve this goal, we propose a coverage method that consists of matching the findings of systematic reviews with SE related questions posted on Q&A websites. Specifically, the question we are trying to answer is:

> **RQ.** To what extent do systematic reviews **cover** software engineering related questions posted on Q&A websites?

By **coverage** we mean: at least one finding of a systematic review offers knowledge that *helps to solve* a SE related question. By any means, however, we are not suggesting that systematic reviews should provide definitive evidence to answer these questions. Nevertheless, we believe that systematic reviews can provide valuable insights, that practitioners can use to get acquainted with possible solutions and, therefore, seek further evidence by themselves.

Our method works as follows (details at Section II): (1) we extracted key findings of a SR, (2) we identified SE questions related to the systematic review, and (3) we applied a qualitative research approach to match whether the findings of the systematic review cover SE related questions.

We investigate SE related questions posted on five StackExchange websites. StackExchange is an umbrella of over 160 high-quality Q&A websites. We selected Q&A websites particularly relevant to SE practice (details at Section II-A). Although in its early stages, this study provides important contributions:

1) A coverage **method** used to assess systematic reviews using Q&A websites.
2) A preliminary **study** of how systematic reviews cover SE related questions.
3) A reusable **dataset** related to the analysis presented in this paper (available at http://bit.ly/2dMaaRJ).

II. Method

A. StackExchange Websites Selection

We analyzed the official description of each one of the 160 Q&A websites and selected those that are related to SE, according to SWEBOK [12]. The five selected Q&A websites, and their characteristics are presented at Table I. We did not use StackOverflow since it is focused on specific coding issues, which is rarely the target of systematic reviews.

According to Area51 [18], websites with A/Q ratio above two are considered as *good*, and above one are *okay but need improvement*. The first three listed websites were classified as good, and the remaining ones were okay but need improvement. However, when considering metrics such as number of visits per day and number avid users, RE and SREC were considered excellent, with 199 avid users and 1,905 visits per day, and 394 avid users and 4,384 visits per day, respectively.

B. Systematic Reviews Selection

Our set of systematic reviews are based on the tertiary study of Da Silva [11], that identified 120 SRs in SE. However, we excluded 88 SRs that, according to Da Silva, do not present guidelines to practitioners. Next, we excluded 8 SRs that do not report their search strings, since we need them on further

[1]By Systematic Review we mean, any kind of secondary study, such as: systematic mappings, meta-analyses, and systematic literature reviews.

978-1-5386-1544-7/17 $31.00 © 2017 IEEE
DOI 10.1109/MSR.2017.5

TABLE I: Relationship between StackExchange communities and SE areas. Q means questions, A means answers.

Q&A WEBSITE	DESCRIPTION	SE AREA	# Q	# A	# A/Q
Programmers (PROG) [13]	Q&A for professional programmers who are interested in getting expert answers on conceptual questions about software development	Software Design Software Construction	35,560	128,199	3.6
Project Management (PM) [14]	Q&A for project managers	Software Management	2,362	8,420	3.56
Quality Assurance & Testing (SQA) [15]	Q&A for software quality control experts, automation engineers, and software testers	Software Testing Software Quality	2,642	6,333	2.39
Reverse Engineering (RE) [16]	Q&A for researchers and developers who explore the principles of a system through analysis of its structure, function, and operation	Software Maintenance	1,745	2,751	1.57
Software Recommendations (SREC) [17]	Q&A for people seeking specific software recommendations	Software Tools	4,434	4,894	1.1

steps of our method (see Section II-C). Finally, we restricted our search to SRs related to Agile Software Development, resulting in four SRs. We chose SRs related to agile development since it is the topic that has the highest number of SRs associated with. They are: SR1 [19] focusing on empirical evidence about agile in general; SR2 [20] focusing on the effectiveness of pair programming; SR3 [21] focusing on the use of Scrum in global software development; and SR4 [22] focusing on the automated acceptance testing practice.

C. Related Questions Selection

We used the StackExchange dump of August 2015. The five selected websites have 46,743 questions. To avoid low-quality questions, we excluded the ones scoring below the median of their website, resulting in 26,687 questions. Finally, we used the search strings of each selected SR and found 641 questions. Due to space constrains we omitted the search strings used, although they can be found at the SRs. To remove false-positives, we manually tagged questions as *Related* or *Not Related* to agile development. This reduced our sample to 293 related questions. To avoid bias, this process was conducted in pairs, followed by conflict resolution meetings. The Kappa value was 0.85 which means an *Excellent Agreement* level [23]. Table II shows the results of this classification.

As we can see, SR4 presents a high number of false-positives. When analyzing the search strings of SR4, we found that it is composed of many ordinary terms such as "fit" and "software", which reduced the effectiveness of the search. Finally, we found that some SRs selected the same questions. After removing these duplicated questions, we ended up with **284 unique related questions**. We refer this group throughout the paper as our **related questions**.

D. Coverage Method

After defining the related questions, we established the *Match Procedure* to identify whether the selected SRs cover, that is, help to solve, the selected questions. This procedure is based on the *open coding* technique which uses inductive logic to construct analytical codes and infer categories from the data. Following are the steps that compose the match procedure:

1) We extracted the findings of each SR and applied open coding [24], [25] to define their *Key Points*.
2) We applied open coding [24], [25] with the related questions to also defined their *Key Points*.
3) We matched the *Key Points* of each related questions against the *Key Points* of each SRs findings, to establish the coverage.

These steps provide quantitative data about how SRs cover the related questions. To understand additional characteristics of the (not-) covered questions, we conducted another round of open coding. We applied constant comparison technique [24] to group recurrent questions. To avoid bias, this process was conducted in pairs: one researcher conducted the procedures, and another one revised. Figure 1 depicts the entire process.

TABLE II: Number of related and not related questions.

SR	RELATED		NOT RELATED	TOTAL
	#	%	#	#
SR1 [19]	217	54.8%	179	396
SR2 [20]	45	57.7%	33	78
SR3 [21]	7	63.3%	4	11
SR4 [22]	24	15.4%	132	156
TOTAL	293	45.8%	348	641

III. Case Study

Throughout the section we provide discussions about *low* and *high* coverage. However, we are not suggesting that a low coverage is bad or a high coverage is good. Instead, we believe that the coverage of systematic reviews might indicate whether such systematic review has immediate implications for SE practice, or whether the subject that it touches is, for instance, not yet widely adopted. This might help researchers to better orientate their research efforts.

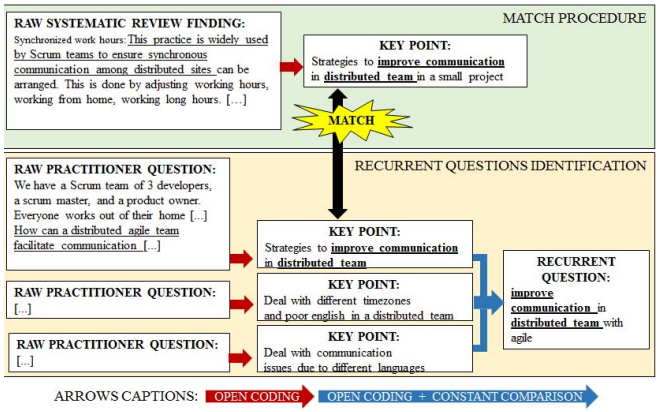

Fig. 1: Coverage Analysis Procedure

TABLE III: Overall Coverage.

SR	COVERED		NOT COVERED	TOTAL
	#	%	#	#
SR3 [21]	5	71.4%	2	7
SR4 [22]	7	29.1%	17	24
SR1 [19]	20	9.2%	197	217
SR2 [20]	4	8.8%	41	45
TOTAL	36	12.2%	257	293

A. Overall Coverage

Table III shows the coverage of the related questions, grouped by each SR. As we can see, about 12% of the 293 questions were covered. Three out of the four SRs covered less than 30% of the related questions.

B. Coverage of the Agile Methods

Table IV describes the coverage per agile method found. Questions not related a specific agile method are grouped under *General*. As we can see, only 5/11 agile methods found have at least one question covered by the findings of the SRs.

Automated Acceptance Testing is the agile method with the highest coverage rate. This happens because SR4 [22] is focused on this practice. Despite the fact that **TDD** presented the second highest coverage (25%), only one question was covered. This question is about the benefits of TDD, in order to *"[...] help [him] in what is normally a very short conversation of trying to sell TDD [to his manager]"*. The low number of questions related to TDD occurred because there was no SRs about TDD (SR1 [19] is broader in scope).

Moreover, 11% of questions related to **Pair Programming** were covered. This occurred even with an SR focused on this practice (SR2 [20]). Most of the related questions were focused on the dynamics of this practice. For instance, (1) practitioners asked if pair programming can be used to transfer knowledge between members of the pairs; (2) if pair programming hinder concentration; or (3) strategies and tools to enable pair programming with distributed pairs. SR2 does not provide evidence to support any of these questions.

About 10% of the questions related to **Scrum** were covered. Again, there was no systematic review focusing specifically on Scrum, which indicates an opportunity for researchers, since it is the most recurrent agile method (79 out of the 293 questions are about Scrum). SR1 [19] and SR3 [21] present some findings, although they are not about Scrum. Some not covered questions about to Scrum are: (1) strategies to deal with the challenges of use Scrum in a distributed team; (2) applicability of Scrum in projects with specific characteristics, for example embedded software development; and (3) strategies to mix Scrum with other agile methods.

TABLE IV: Coverage of the Agile Methods

AGILE METHODS	COVERED		NOT COV.	TOTAL
	#	%	#	#
Automated Acceptance Testing	6	38%	10	16
TDD	1	25%	3	4
Pair Programming	4	11%	34	38
Scrum	8	10%	71	79
XP	1	6%	16	17
BDD	0	0%	3	3
FDD	0	0%	2	2
Kanban	0	0%	3	3
Lean Software Dev.	0	0%	2	2
RAD	0	0%	1	1
General	16	13%	103	119

Only one related question about **XP** (6%) was covered. Although SR1 [19] presents a rich set of findings about the benefits of XP, we found that the related questions are diffuse, varying from: (1) the impacts of low documentation in a XP projects; (2) applicability of XP in projects with specific characteristics; or even (3) questions about trends in agile, for instance if XP is declining in favor of other agile methods.

Kanban, BDD, Lean Software Development, FDD, and Rapid Application Development (RAD) were not covered at all. Together, **Lean Software Development, FDD, and RAD** were associated with only five related questions, although they were terms of the search string. Still, six questions related to **Kanban** and **BDD**, even though there was no term related to them in any search string. We believe there is a need of studies aimed at better understanding if the low amount of questions occurred because they are indeed not often used in practice, or if the SRs indeed do not cover them well.

C. Coverage of the Recurrent Questions

Table V provides a complete list of 27 recurrent questions. **Miscellaneous** are questions that did not have their own group. Questions asking agile definitions, such as *"In pair programming, what is each role named, and why?"*, were classified as **Definitions**.

From the 27 recurrent questions, 10 were covered in some degree. Among them, only five recurrent questions presented a coverage higher than 50%. They vary from **communication in distributed teams**, **negative impact of agile methods**, and **benefits of agile methods from a *specific perspective***. This last recurrent question, in particular, corroborates with the idea that empirical evidence should comprise not only data about the effectiveness of an intervention, but also useful information for the target audience [26].

The five recurrent questions with coverage higher than zero but below 50% aggregates the majority of related questions, as seen in the column 'T' of Table V. This might indicate an opportunity for researchers interested in tackling these not well covered topics. **Applicability of agile in specific project context** reinforces the importance of evidence with detailed contextual information [27], [28] to solve specific problems, such as: *"[..] how to apply agile methods in large complex embedded system (100+ engineers). Firmware development has some unique characteristics that make it difficult to do agile"*. **Tools for agile methods** is the second most recurrent question with low coverage. Such lack could be paved with comparative analyses to help practitioners to choose which tool fit their needs (*e.g.,* [29]).

The remaining 17 not covered recurrent questions reveal interesting insights. For instance, practitioners seem to be interested in the applicability of **pair programming to transfer**

TABLE V: Coverage of the Recurrent Questions. 'C' means Covered and 'T' means Total.

Recurrent Questions	# C	# T
Improving communication in a distributed agile team	4	4
Negative impact of agile in software design	3	3
Low customer collaboration	3	4
Benefits of agile methods from a specific perspective	3	6
Demand for scientific empirical evidence	1	2
Introducing agile in a project	5	13
Benefits of agile methods in general	3	9
Challenges to manage distributed teams with agile	1	5
Tools for agile methods	3	20
Applicability of agile in specific project context	1	20
Mixing agile with traditional methods	0	15
Effort estimation in agile in specific project context	0	6
Tasks that do not fit in one sprint	0	4
Pair programming to transfer knowledge	0	8
Tools for mixed agile methods	0	7
Mixing multiple agile methods	0	7
Impact of low detail level or absence of documentation	0	6
Pair programming hindering concentration	0	4
Impacts of constant changes in requirements	0	4
Pair programming with distributed pairs	0	4
Trends in agile	0	3
Ideal workplace layout in agile	0	3
Team rotation in agile	0	3
Ad-hoc software development as agile	0	3
Enforcing scrum to and entire organization	0	2
User stories for non-functional requirements	0	2
Pair programming as replacement to code reviews	0	2
Definitions	0	40
Miscellaneous	17	76

knowledge from more skilled developers to less skilled ones, *e.g.*, *"Is Pair Programming also used to train less experienced developers and bring them up to speed?"*. SLR2 [20] evaluated the impact of pair programming in many dimensions, but only considered pairs with the same level of experience.

Three recurrent questions revealed concerns about pair programming that are not even mentioned on the SR that focused on this topic. They are: (1) the difficulties associated with **Pair programming with distributed pairs**, (2) when **Pair programming hindering concentration** is an issue, and (3) if it is doable to use **Pair programming as replacement to code reviews**. Apart from pair programming, practitioners seem to have recurrent problems with software projects that are **Mixing multiple agile methods** or even **Mixing agile with traditional methods**. Practitioners also want to define the **Ideal workplace layout in agile** teams, and also how **Team rotation in agile** impact team productivity. None of the SRs help to answer these questions. recurrent issues with projects and companies that are **Mixing multiple agile methods** or even **Mixing agile with traditional methods**. One of the principles of agile methods is challenged when practitioners want to understand the **Impacts of constant changes in requirements** or report to have experienced **Ad-hoc software development as agile**. Practitioners also want to define the **Ideal workplace layout in agile** teams, and also how **Team rotation in agile** impact team productivity. None of the SRs help to answer these questions.

IV. Implications & Limitations

Implications. Researchers can use our method to assess whether their research topic is adopted in practice (represented by the number of related questions), or identify new research opportunities (represented by the number of low covered questions). Researchers can at least validate their search strings in Q&A websites before conducting systematic studies.

Limitations. First, the selected SRs came from a tertiary study published in 2011. Unfortunately, this is the most up-to-date study targeting software engineering SRs in general. We are aware that there are more recent tertiary studies (*e.g.,* [30]–[36]), but they are niche-specific. However, since this work is in its early stages, we focused on proposing and evaluating a methodology. We leave completeness (*i.e.* covering all SE topics) to future work. Second, the selection of questions is sensible to the search strings employed in the SRs. We mitigate low-quality questions by filtering the ones scoring below the medium of their website. Third, the analysis was mostly manual. To mitigate classification bias, we conducted it in pairs, with conflict resolution meetings.

V. Related Work

The SE community has long recognized the importance of Q&A websites, with prosperous contributions touching both technical (such as refactoring [37], testing [38], debugging [7]), and social aspects (such as age [39], gender [5], and reputation [4]) of SE. The closest work to us is from Garousi et al. [40], but it proposes the use of StackExchange as source of evidence as gray literature for SRs, not as a source to assess how SRs are connected to practice as in this work. Our work is unique in the sense that we take advantage of the rich database of Q&A websites to empower researchers that want to assess how their systematic reviews are aligned with the needs of practitioners.

VI. Conclusions

In this paper, we proposed and evaluated a method for assessing if systematic reviews can be used to help to solve SE questions posted on Q&A websites. Through a comprehensive qualitative double-reviewed process, we analyzed more than 600 questions posted in five Q&A websites, and matched them with the findings of four systematic reviews. Our study suggested that the majority of questions about agile methods were not covered. However, for future work, we are planning to extend this study to assess the coverage of more recent SRs about agile, and also analyze the correlation between the number of questions covered by SRs with their number of citation. Other researchers may want to selected SRs from more recent tertiary studies, or focused on different topics of software engineering beyond agile.

Acknowledgments

This work is partially supported by INES (grants CNPq/465614/2014-0 and FACEPE/APQ/0388-1.03/14) and CNPq (grants 304499/2016-1 and 06308/2016-0).

REFERENCES

[1] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest q&a site in the west," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 2857–2866. [Online]. Available: http://doi.acm.org/10.1145/1978942.1979366

[2] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web? (nier track)," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 804–807. [Online]. Available: http://doi.acm.org/10.1145/1985793.1985907

[3] B. Bazelli, A. Hindle, and E. Stroulia, "On the personality traits of stackoverflow users," in *2013 IEEE International Conference on Software Maintenance*, Sept 2013, pp. 460–463.

[4] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft, "Building reputation in stackoverflow: An empirical investigation," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13, 2013, pp. 89–92.

[5] B. Vasilescu, A. Capiluppi, and A. Serebrenik, "Gender, representation and online participation: A quantitative study of stackoverflow," in *2012 International Conference on Social Informatics*, Dec 2012, pp. 332–338.

[6] C. Parnin and C. Treude, "Measuring api documentation on the web," in *Proceedings of the 2Nd International Workshop on Web 2.0 for Software Engineering*, ser. Web2SE '11, 2011, pp. 25–30.

[7] F. Chen and S. Kim, "Crowd debugging," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2015, 2015, pp. 320–332.

[8] G. Pinto, F. Castor, and Y. D. Liu, "Mining questions about software energy consumption," in *11th MSR*, 2014, pp. 22–31.

[9] B. A. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-based software engineering," ser. 26th ICSE, 2004.

[10] B. Cartaxo, G. Pinto, E. Vieira, and S. Soares, "Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners," in *10th ESEM*, 2016, pp. 57:1–57:10.

[11] F. Q. B. da Silva, A. L. M. Santos, S. Soares, A. C. C. França, C. V. F. Monteiro, and F. F. Maciel, "Six years of systematic literature reviews in software engineering: An updated tertiary study," *IST*, 2011.

[12] P. Bourque and R. E. Fairley, Eds., *SWEBOK: Guide to the Software Engineering Body of Knowledge*, version 3.0 ed. Los Alamitos, CA: IEEE Computer Society, 2014. [Online]. Available: http://www.swebok.org/

[13] "Stackexchange programmers community," http://programmers.stackexchange.com, acessed in: Nov. 4, 2016.

[14] "Stackexchange project management community," http://pm.stackexchange.com, acessed in: Nov. 4, 2016.

[15] "Stackexchange software quality & testing community," http://sqa.stackexchange.com, acessed in: Nov. 4, 2016.

[16] "Stackexchange reverse engineering community," http://reverseengineering.stackexchange.com, acessed in: Nov. 4, 2016.

[17] "Stackexchange software recommendations community," http://softwarerecs.stackexchange.com, acessed in: Nov. 4, 2016.

[18] "Area 51, the stack exchange q&a site creation zone," http://area51.stackexchange.com/, acessed in: Feb. 1, 2017.

[19] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *IST*, 2008.

[20] J. E. Hannay, T. Dybå, E. Arisholm, and D. I. Sjøberg, "The effectiveness of pair programming: A meta-analysis," *IST*, 2009.

[21] E. Hossain, M. Babar, and H. young Paik, "Using scrum in global software development: A systematic literature review," in *4th ICGSE*, 2009.

[22] B. Haugset and G. Hanssen, "Automated acceptance testing: A literature review and an industrial case study," in *AGILE*, 2008.

[23] A. J. Viera and J. M. Garrett, "Understanding interobserver agreement: The kappa statistic," *Journal of the Society of Teachers of Family Medicine*, 2005.

[24] K.-J. Stol, P. Ralph, and B. Fitzgerald, "Grounded theory in software engineering research: A critical review and guidelines," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: ACM, 2016, pp. 120–131. [Online]. Available: http://doi.acm.org/10.1145/2884781.2884833

[25] J. Saldaña, *The coding manual for qualitative researchers*. Sage, 2015.

[26] N. b. Ali, "Is effectiveness sufficient to choose an intervention?: Considering resource use in empirical software engineering," in *10th ESEM*, 2016, pp. 54:1–54:6.

[27] B. Cartaxo, A. Almeida, E. Barreiros, J. Saraiva, W. Ferreira, and S. Soares, "Mechanisms to characterize context of empirical studies in software engineering," in *Experimental Software Engineering Latin American Workshop (ESELAW 2015)*, 2015, pp. 1–14.

[28] T. Dybå, D. I. Sjøberg, and D. S. Cruzes, "What works for whom, where, when, and why?: On the role of context in empirical software engineering," in *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '12. New York, NY, USA: ACM, 2012, pp. 19–28. [Online]. Available: http://doi.acm.org/10.1145/2372251.2372256

[29] C. Marshall, P. Brereton, and B. Kitchenham, "Tools to support systematic reviews in software engineering: A feature analysis," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '14. New York, NY, USA: ACM, 2014, pp. 13:1–13:10. [Online]. Available: http://doi.acm.org/10.1145/2601248.2601270

[30] A. B. Marques, R. Rodrigues, and T. Conte, "Systematic literature reviews in distributed software development: A tertiary study," in *Proceedings of the 2012 IEEE Seventh International Conference on Global Software Engineering*, ser. ICGSE '12, 2012, pp. 134–143.

[31] D. Budgen, S. Drummond, P. Brereton, and N. Holland, "What scope is there for adopting evidence-informed teaching in se?" in *2012 34th International Conference on Software Engineering (ICSE)*, June 2012, pp. 1205–1214.

[32] M. Goulão, V. Amaral, and M. Mernik, "Quality in model-driven engineering: A tertiary study," *Software Quality Journal*, vol. 24, no. 3, pp. 601–633, Sep. 2016.

[33] D. S. Cruzes and T. Dyb, "Research synthesis in software engineering: A tertiary study," *Information and Software Technology*, vol. 53, no. 5, pp. 440 – 455, 2011, special Section on Best Papers from {XP2010}. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095058491100005X

[34] F. Q. B. da Silva, A. L. M. Santos, S. C. B. Soares, A. C. C. França, and C. V. F. Monteiro, "A critical appraisal of systematic reviews in software engineering from the perspective of the research questions asked in the reviews," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '10. New York, NY, USA: ACM, 2010, pp. 33:1–33:4. [Online]. Available: http://doi.acm.org/10.1145/1852786.1852830

[35] N. B. Ali and K. Petersen, "Evaluating strategies for study selection in systematic literature studies," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '14. New York, NY, USA: ACM, 2014, pp. 45:1–45:4. [Online]. Available: http://doi.acm.org/10.1145/2652524.2652557

[36] Y. Zhou, H. Zhang, X. Huang, S. Yang, M. A. Babar, and H. Tang, "Quality assessment of systematic reviews in software engineering: A tertiary study," in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '15. New York, NY, USA: ACM, 2015, pp. 14:1–14:14. [Online]. Available: http://doi.acm.org/10.1145/2745802.2745815

[37] G. H. Pinto and F. Kamei, "What programmers say about refactoring tools?: An empirical investigation of stack overflow," in *Proceedings of the 2013 ACM Workshop on Workshop on Refactoring Tools*, ser. WRT '13, 2013, pp. 33–36.

[38] P. S. Kochhar, "Mining testing questions on stack overflow," in *Proceedings of the 5th International Workshop on Software Mining*, ser. SoftwareMining 2016, 2016, pp. 32–38.

[39] P. Morrison and E. Murphy-Hill, "Is programming knowledge related to age? an exploration of stack overflow," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13, 2013, pp. 69–72.

[40] V. Garousi, M. Felderer, and M. V. Mäntylä, "The need for multivocal literature reviews in software engineering: Complementing systematic literature reviews with grey literature," ser. EASE '16. New York, NY, USA: ACM, 2016, pp. 26:1–26:6. [Online]. Available: http://doi.acm.org/10.1145/2915970.2916008