What Software Engineering "Best Practices" are we Teaching Students - a Systematic Literature Review

Maíra Marques[1] and Javier Robledo[2]

***Abstract*— This research presents that teaching software engineering can be a demanding challenge for instructors, considering that the software industry grows rapidly and there are new development technologies being released constantly. Some of the methodologies being used in industry with time are converted in "best practices". This systematic literature review (SLR) focuses on finding what are the software engineering best practices being taught to students in academia. It is expected to show with this SLR what are the best practices being taught and how, so other instructors and the teaching staff can evaluate what to choose and level up their courses to the extent of what is being used and is already tested. To perform this SLR, a well-known protocol was used with the search string: "software engineering education" and "best practices" (variations and synonyms of the words were also used), the search was performed in six well-known databases. It is surprising that the amount of primary studies found in this SLR was not what was expected, seventeen primary studies were identified. These studies mentioned a total of seventy best practices that are being used in academia to teach software engineering, some of them are mentioned in more than one paper. But the granularity of the primary studies was quite different some of the best practices are really software engineering best practices and others are instructional best practices. It was also evaluated how the use of these practices was validated and reported, and the results are very diverse, some of them did not have a validation, others have qualitative data, and a few qualitative and quantitative data.**

I. INTRODUCTION

Software engineering education is concerned with finding good ways to teach students how to analyze problems, evaluate alternatives and apply the technical skills they have learned (related to computer science and its application domain) to be able to develop high-quality software solutions for their clients.

To do that, a lot has been said about the use and teaching of software engineering "best practices". Frezza states that [7] "new software engineering's entering the workforce are often at the top of the technology transfer cycle: not only do they need to be able to learn best practices before they join the workforce, they also need to be able to champion transferring these practices into their organizations." But, what are software engineering "best practices"? Where can they be found? Surendran [29] considers the Software Engineering Body of Knowledge (SWEBOK) [1] as a reference of good practices. But, as stated by Hilburn et al. [12], "with the increasing demand for software engineers and other computing professionals, many of these issues are obscured and it is difficult for many computing educators to understand and appreciate the importance of software engineering education". Also, the ACM-IEEE Software Engineering Curriculum Guidelines [16] can be considered a source of good practices, but not all the universities are involved in a context that allows for the implementation of these guidelines: some universities have to teach all the sub-disciplines of computing (Computer Science, Computer Engineering, Information Systems, Information Technology, and Software Engineering) as part of just one curriculum.

As stated by Bailey et al. [2], "as the software industry matures, new development methodologies are invented and some of these methodologies transition into best practices. Our role as university education is to teach these best practices". It is wanted from students to understand, appreciate and continue using these methodologies after they finish their studies, so academia can have a relevant impact on software engineering in general.

Thompson and Fox [30] found in their investigation of leading edge practices in Software Engineering, a lack of guidelines on how to approach and execute a software project, and that there was no clear indication of the best practices being used in industry. Therefore, the concern about what are the best practices remains an open question.

The main goal of this work is to clarify what software engineering best practices are being taught and how they are being taught. To do so, a systematic literature review (SLR) was conducted on software engineering "best practices" being taught in academia. A set of four research questions were proposed with the aim of clarifying where "best practices" stand at the moment.

The rest of the paper is organized as follows. In Section II the ACM - IEEE Curriculum Guidelines for Software Engineering was briefly described, Section III presents the research questions. Section IV describes the search strings as well as the search strategy followed. In Section V the selection criteria is enumerated. Section VI presents the results and Section VII discusses the threats to the validity. The final Section VIII presents the conclusions.

II. ACM - IEEE - CURRICULUM GUIDELINES

The main idea of the ACM - IEEE Software Engineering Curriculum Guideline is to provide guidance on what should be taught in an undergraduate software engineering program. The guidelines are hierarchically divided into three levels: knowledge area (a particular sub-discipline of SE), units

[1]Maíra Marques is with the Computer Science Department, Universidad de Chile, Beaucheff 851, Santiago, Chile mmarques@dcc.uchile.cl

[2]Javier Robledo is with the Computer Science Department, Universidad de Chile, Beaucheff 851, Santiago, Chile jrobledo@dcc.uchile.cl

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

TABLE I
ACM-IEEE SOFTWARE ENGINEERING CURRICULUM GUIDELINES CONSIDERED

Knowledge Area	Units of Knowledge
Professional Practice	Group Dynamics and Psychology, Communication Skills, Professionalism
Requirements Analysis and Specification	Requirements Fundamentals, Eliciting Requirements, Requirements Specification and Documentation, Requirements Validation
Software Design	Design Concepts, Design Strategies, Architectural Design, Human Computer Interaction Design, Detailed Design, Design Evaluation
Verification and Validation	Verification and Validation Foundations, Review and Static Analysis, Testing, Problem Analysis and Reporting
Software Process	Process Concepts, Process Implementation, Project Planning and Tracking, Software Configuration Management, Evolution Process and Activities
Software Quality	Software Quality Concepts and Culture, Process Assurance, Product Assurance
Security	Security Fundamentals, Computer and Network Security, Developing Secure Software

TABLE II
RESEARCH TYPES

Category	Description
Experience Report	Authors personal experience describing what and how something was done. This usually includes a lesson-learned section
Case Study	Report of investigation of a real-life phenomenon. It may include quantitative evidence, relies on multiple sources of evidence, and also may report one or several cases
Action Research	Problem solving actions implemented in a collaborative context with data-driven analysis or research, with the intent to enlighten underlying causes enabling future trend prediction
Experiment/Quasi-Experiment	Manipulation and controlled testing for causal analysis. Normally, one or more variables are manipulated to determine their effect on the dependent variable
Solution Proposal	A solution is proposed and the results of using the proposed solution are reported. An example or a line of argumentation must show the applicability of the solution

(individual thematic modules) and topics. So, a body of core knowledge was defined, and described by LeBlanc et al. as [16] "the essential material that professionals teaching software engineering agree is necessary for anyone to obtain an undergraduate degree in this field". With that being said , in this work the SE Curriculum Guidelines were considered as the basis of comparison of software engineering "best practices" being taught. The guidelines contain 10 knowledge areas (KA); and 7 of them contain "best practices" that should be considered in a software project (Professional Practices, Requirements Analysis and Specification, Software Design, Verification and Validation, Software Process and Software Quality). Within these 7 knowledge areas there are 34 units of knowledge in total, and these units of knowledge were considered to be a set of "best practices". The knowledge areas and the units are listed in Table I.

III. RESEARCH QUESTIONS

The main goal of this literature review is to find which software engineering best practices are being taught to students. However, this goal is too broad, so it was subdivided, in order to focus on specific aspects of the evaluation.

To define the research questions the Population, Intervention, Comparison, Outcome and Context (PICOC) structure was followed [26]. A comparison among the findings was not considered since this is not the goal. So, the focus will be on the following set of research questions about software engineering best practices being taught.

- RQ1 - Which software engineering best practices are reported as being taught in academia?
- RQ2 - Are they taught in a practical way? RQ2.1 - How?
- RQ3 - The use of software engineering "best practices" being taught was validated somehow? RQ3.1 - Number of teams reported? RQ3.2 - What is the duration of the time frame reported in the paper? RQ3.3 - What was the research approach used to report? (Table II shows a description of the research type categories used to classify the articles. These categories are based on those proposed by Petersen et al. [25] and Nacimento et al. [21]).
- RQ4 - Are there any "best practices" being taught that are part of the knowledge units of ACM-IEEE SE Curriculum Guidelines?

IV. SLR METHODOLOGY

Nowadays systematic literature reviews are being used as a means to consolidate scientific knowledge. One of their major strengths is that they follow a defined protocol that allows verifiability and rigor. In this case, the guidelines created by Kitchenham et al. [14] were followed, even though they are not detailed here for brevity.

A. Search Strategy

Based on these questions, keywords were identified to be used to search for primary studies. Table III shows the words that are being looked for and related synonyms that are considered in the search.

TABLE III
SEARCH STRING

Term	Keywords	Synonyms
A	software engineering education	"software engineering education"
B	best practices	"best practices", "recommended practices", "edge practices"

TABLE IV
SPECIFIC SEARCH STRINGS FOR EACH DIGITAL LIBRARY

Digital Library	**Search String**
ACM	("software engineering education") and ("best practices" or "edge practices" or "recommended practices")
IEEE Xplore	(("software engineering education") AND ("best practices" OR "recommended practices" OR "edge practices"))
ScienceDirect	(("software engineering education") AND ("best practices" OR "recommended practices" OR "edge practices"))
Scopus	TITLE-ABS-KEY ("software engineering education") AND TITLE-ABS-KEY ("best practices" OR "recommended practices" OR "edge practices")
Springer Link	("software engineering education") AND ("best practices" OR "recommended practices" OR "edge practices")
Wiley	"software engineering education" in All Fields AND "best practices" OR "edge practices" OR "recommended practices" in All Fields

The search for primary studies was done on the following digital libraries: ACM Digital Library, IEEE Xplore Digital Library, ScienceDirect, Scopus, Springer Link and Wiley. These libraries were chosen because they are among the most relevant sources of scientific articles in several computer science areas [18]. Limitations in terms of publication year were not considered and the search was performed at the beginning of January 2018, so only publications done before December 31, 2017 were considered.

B. Particular Searches

Not all search databases operate in the same way or respect the same rules for the search, so some adjustments were needed during the first step of this work. The adjustments were different for each database but due to space restrictions, it was not possible to describe all of them here.

Table IV shows the specific search strings used in each database considering the restrictions that were mentioned earlier. Table V shows the results obtained performing the searches in the databases. A total of 1452 primary studies were selected.

V. SELECTION CRITERIA

After performing the search on the selected databases, the relevant primary studies found according to the following inclusion and exclusion criteria were evaluated:

TABLE V
NUMBER OF THE PAPERS RETRIEVED FROM EACH DIGITAL LIBRARY

Digital Library	**Search Results**
ACM	119
IEEE Xplore	397
ScienceDirect	83
Scopus	42
Springer Link	786
Wiley	29
Total	**1452**

1) Inclusion criteria: the paper is in English, the paper is a peer-reviewed article and it was obtained from a journal, conference or workshop. The paper was published on or before December 2017, documents reporting teaching/learning software engineering experiences, documents presented as full papers; short papers and work in progress, where reported approach and its validation are reasonably present.
2) Exclusion criteria: the paper is not available online, documents whose full text is not available, documents that are only available as an abstract, documents describing a panel, documents that are a letter from the editor, documents in which the context is not related to software engineering education, documents that report the use of software engineering as a means of teaching a specific programming language or framework, documents that report opinion papers, and documents proposing/reporting curriculum description, massive online courses and philosophical theories.

After applying the inclusion and exclusion criteria on titles and abstracts, 74 papers were selected for the next phase. In the next phase the papers were fully read and the inclusion and exclusion criteria were applied again. In the end a corpus of 17 papers was obtained for the final investigation. These numbers are summarized in Tab. VI.

TABLE VI
RESULTS OBTAINED AFTER THE USE OF THE GUIDELINES

Phase	**Amount of papers**
All papers	1452
After paper screening	74
After full reading	17

VI. OBTAINED RESULTS

This section presents the results produced by the SLR. Since the publication of Thompson and Fox [30], a little bit more than ten years ago, there is a trend of relevance in the topic since 41% of the primary studies selected were published in the last three years; the others were published in a time span of 21 years. Table VII shows the primary studies selected in the systematic literature review.

Now each one of the research questions proposed were analyzed. It is worth mentioning that it was surprising that from 1452 studies only 17 (1.17%) ended up being useful. A small sample of search in some of the selected papers

TABLE VII
PRIMARY STUDIES SELECTED

Year	Venue	Authors
1996	Conference	Tomayko [31]
1999	Conference	Upchurch et al. [32]
2001	Conference	Groth et al. [11]
2003	Conference	Bailey et al. [2]
2005	Journal	Schneider et al. [28]
2007	Conference	van der Duim et al. [33]
2009	Conference	Koolmanojwong et al. [2]
2011	Journal	Chen et al. [3]
2012	Conference	Neto et al. [22]
2013	Conference	Gary et al. [10]
2015	Conference	Rost et al. [27]
2015	Journal	Lehtinen et al. [17]
2015	Workshop	McDonald et al. [20]
2016	Conference	Mathies et al. [19]
2016	Journal	Flores et al. [6]
2017	Conference	Eddy et al. [5]
2017	Conference	Neyem et al [23]

on the first phase was made to see what happened; and it was found out that many papers only mention the words best practices but they did not clearly state what they mean by best practices, what they are or anything else. Some examples: [8] states that "Students work with the faculty team members to make sure that best practices are being used and that academic requirements are met."; in [13] reports that "... students can experience and appreciate the benefit of design principles and best practices only in a frame of larger projects than assignments"; and in [24] it was found that "Such processes should expose the students to well-known notations, techniques, and best practices, offering them practical grounds to decide about their adoption in actual work situations". This is also mentioned by the author of one of the selected primaries studies [19]: "There is little research yet on what constitutes best practices for agile metrics"; their conclusion is related to the agile world, which with the result of this work could be extended to a broader focus.

A. RQ1 - What are the best practices taught in software engineering education?

The classification schema of the answers found for the first research question is on Tab. VIII and Tab. IX. In Tables VIII and IX, the Primary Studies selected are named PS and the publication year and the best practices reported were also presented.

Analyzing the 17 primary studies selected it was expected to see the best practices software engineering taught to students (practices that students can use/incorporate in their developments), but ii was also found some instructional best practices used to teach software engineering. The best practices were split into instructional best practices (teaching methodology) and students applicable best practices. In Table VIII see the students applicable best practices and in Table IX it is possible to see the instructional best practices. Some authors reported these two types of best practices in the same paper [10][22][33].

TABLE VIII
CLASSIFICATION OF RESEARCH QUESTION 1

PS	Year	Reported Best Practices (applicable by students)
[31]	1996	**reflexive practices**, **use of inspections**, formal methods, **integrating process**, mentors, client satisfaction, boot camp, **requirement**s, PSP
[32]	1999	project postmortems
[11]	2001	prototyping, **quality assurance**
[2]	2003	**code inspection**
[28]	2005	eXtremme Programming practices (**planning game**, small releases, metaphor, simple design, **testing**, refactoring, **pair programming**, collective ownership, **continuous integration**, forty-hour week, on-site customer and coding standards)
[33]	2007	contacts, reciprocity, feedback, time tasks, respect
[15]	2009	**requirement management**, object oriented analysis and design, risk management, **quality management**, peer reviews, configuration management, and value based software engineering
[22]	2012	on-line daily meetings, group leader, iterative and incremental development, activity redistribution, version control, self manageable groups, **on-line pair programming**
[10]	2013	preparation, **reflection**
[17]	2015	root cause analysis
[27]	2015	**requirements**, **planning, design**, evaluation, realization, control, project planning
[20]	2015	**requirements**, analysis, **design**, **testing**, implementation, maintenance , security
[19]	2016	conformance metric
[6]	2016	design patterns
[5]	2017	static analysis, unit testing, integration testing, **continuous building**

TABLE IX
CLASSIFICATION OF RESEARCH QUESTION 1

PS	Year	Reported Best Practices (instructional)
[33]	2007	active learning, high expectations, respect
[3]	2011	industrial involvement
[22]	2012	internal and external groups training
[10]	2013	practice (labs) and project centered learning
[23]	2017	projects with real clients, communication, project management

From all the primary studies selected there are a total of 70 best practices reported; from these 10 are repeated so it was found that 57 best practices reported being taught in software engineering education. Practices reported being taught in more than one primary study (they are in bold text in Table VIII and are presented in more detail in Table X).

The best practices found are described with different granularity in different papers. For example, it is clear what "code inspections" means, but it is not as clear when "requirements" is mentioned provided that requirements is a knowledge area that can have several subareas and it can even have its own specific course [16].

TABLE X
BEST PRACTICES REPORTED IN MORE THAN ONE PS

Reported Best Practices (applicable by students)	PS1	PS2	PS3	PS4
reflective practices	[31]	[10]		
inspections	[31]	[2]		
integration	[31]	[28]		
requirements	[31]	[15]	[27]	[20]
quality	[11]	[15]		
pair programming	[28]	[22]		
continuous building (integration)	[5]	[28]		
design	[20]	[27]		
planning (or planning game)	[28]	[27]		
testing	[28]	[20]		

TABLE XI
CLASSIFICATION OF RESEARCH QUESTION 2 AND 2.1

PS	TiP?	How?
[31]	Yes	Studio (one year project course where students assume distinct roles over time and have industrial clients)
[32]	Yes	Reflexive writings at the end of the project based course
[11]	Yes	Project based course, reporting and monitoring formalized in four different paths
[2]	Yes	Individual project using inspections
[28]	Yes	The paper only states that they are taught
[33]	Yes	The paper only states that they are taught
[15]	Yes	The paper only states that they are taught
[3]	Yes	Project based course in collaboration with industry
[22]	Yes	The paper only states that they are taught
[10]	Yes	The paper only states that they are taught
[17]	Yes	Project based course and use of cause effect diagrams
[27]	No	A template of architecture best practices is presented
[20]	Yes	Project based course in parallel with a security based course
[19]	Yes	Single project based course
[6]	Yes	Project based course with and without best practices usage
[5]	Yes	Individual laboratories activities

B. RQ2 - Are they taught in practice?

In the second research question it is wanted to evaluate if these practices were being taught in practice and how they were taught. In Table XI it is shown the Primary Studies selected named PS, if they are taught in practice (column TiP) and how they are being taught. It was found that 94% of the primary studies selected reported that the best practices were being taught in practice, but 31% of the studies did not report **how** these best practices were being taught. 34% (6 PS) of the studies reported that best practices were being taught in project based courses and three studies (17%) reported that the best practices were being taught in an individual project (individual homework style).

TABLE XII
CLASSIFICATION OF RESEARCH QUESTION 3, 3.1 AND 3.2

PS	Validated?	N teams	Time Frame
[31]	No	–	5 years
[32]	No (some qualitative data)	-	6 semesters
[11]	No	-	-
[2]	Yes (quantitative)	individual	2 semesters
[28]	No	-	-
[33]	No	-	-
[15]	No	15-20	-
[3]	Yes (quantitative and qualitative)	-	2 semesters
[22]	No	$\pm$30	5 year
[10]	No	-	-
[17]	Yes (quantitative and qualitative)	11	1 academic year
[27]	No	-	-
[20]	No	-	1 semester
[19]	Yes (quantitative and qualitative)	1 team (38 students)	1 semester
[6]	Yes (quantitative)	2 (13 students each)	1 semester
[5]	Yes (quantitative)	individual	1 semester

C. RQ3 - The use of software engineering "best practices" being taught was validated somehow?

In the third research question the interest was to discover the quality of the selected studies, so it was evaluated if the studies selected were somehow validated. In Table XII the primary studies selected are shown, if they were validated, the number of teams reported and the time frame. It can be seen that 65% of the studies did not report any validation at all; 18% reported qualitative and quantitative validation. Only 29% of the studies reported the number of teams that were being analyzed and 59% reported the time frame considered for the study. None of the primary studies selected analyzed the validity, generalizability or applicability, which shows that there is a lack of maturity in the research work published within the subject of this SLR. It is also possible to see the number of teams reported in each primary study selected (RQ3.1) and the time frame being reported (RQ3.2).

In table XIII it is possible to see the research approach that was used in each of the primary studies selected to report their work. The majority of the primary studies selected, were considered experience reports (47%); because they did not state any particular research approach and are reporting experiences of what they did; and two PS out of the seventeen were stated as case study; two experiments; one as solution proposal and one as a lessons learned.

TABLE XIII
CLASSIFICATION OF RESEARCH QUESTION 3.3

PS	Research Approach
[31]	Experience Report
[32]	Experience Report
[11]	Experience Report
[28]	Experience Report
[33]	Experience Report
[15]	Experience Report
[3]	Case Study
[10]	Solution Proposal
[17]	Experiment
[27]	Lessons Learned
[20]	Experience Report
[19]	Case Study
[6]	Experiment
[5]	Experience Report

D. RQ4 - Are there any "best practices" being taught that are part of the knowledge units of ACM-IEEE SE Curriculum Guidelines?

In the fourth research question it was addressed that the resulting comparison of the "best practices" that were found out as being taught with the knowledge units proposed in ACM-IEEE curriculum guidelines. The comparison was not straightforward because the granularity of the "best practices" found and the knowledge units of the curriculum are quite different. As an example; table VIII demonstrate the requirements as one of the "best practices" being reported, but there is no detail on what is done exactly to teach requirements and in the curriculum guidelines there is a whole knowledge area (KA) about requirements (Table I), and only this knowledge area has four knowledge units. So, a general comparison analysis of knowledge areas and "best practices" was first performed . This was made according to the interpretation of the meaning of each "best practice" reported. The result of this comparison is in Table XIV; shows that all the six knowledge areas that are related to software projects have some "best practice" that can be classified as being taught. In a second analysis the units of knowledge that each KA has with the list of "best practices" were compared, the comparison is in Table XV where it is possible to see that from the 27 units of knowledge associated with software projects; 18 of them have a related "best practice". Which means that almost 67% of the knowledge units of the ACM-IEEE Curriculum Guidelines are covered with some "best practice". It represents more than a half of the knowledge units, but it is not a high amount of coverage, that one as a practitioner should expect. Why does this happen? At least three possibilities were visioned; the "best practices" that are the guidelines are not taught; the "best practices" that are on the guidelines are taught but not reported or the "best practices" that are on the guidelines are not relevant.

TABLE XIV
ACM-IEEE SE CURRICULUM X "BEST PRACTICES" REPORTED

Knowledge Area	Related Best Practices
Professional Practice	reflexive practices, mentors, project postmortems, metaphor, pair programming, collective ownership, forty-hour-week, contacts, reciprocity, feedback, respect, peer reviews, group leader, activity redistribution, self manageable groups
Requirements Analysis and Specification	requirements
Software Design	simple design, design, design patterns, refactoring, object oriented analysis and design
Verification and Validation	inspections, prototyping, peer reviews, static analysis, testing, unit testing, integration testing
Software Process	PSP, formal methods, root cause analysis, realization, implementation, project planning, continuous integration, configuration management, version control, continuous building
Evolution Process	iterative and incremental development, maintenance
Software Quality	client satisfaction, quality assurance, coding standards, quality management, evaluation, conformance metrics
Security	security

VII. THREATS TO VALIDITY

Construct validity is related to what is being investigated according to the research questions. Since the search strings are constructed based on the research questions, there is no threat directly related to construction. But there is a threat related to the data since there is only data before December 31, 2017.

Data reliability focuses on the papers collected and analyzed in order to see if these processes were conducted in a way that they can be repeated. The search terms were defined and the procedures applied in this study followed the previously described guidelines. The corpus of literature used as a primary study is reported, therefore other researchers can replicate this study.

As stated earlier, the automated search engines have limitations, the searches are performed manually in each engine; and each engine works in a different way, some consider word steaming, others do not; some have precedence operators other did not have or say anything about it; and this could be considered a threat to the results, besides the fact that the inclusion or exclusion of the studies according to the predefined criteria was not always unambiguous. Finally, since there is no analysis of the collected data, there is no statistical assumptions and there is no generalization; therefore it was considered that there is no internal or external validity threat.

VIII. CONCLUSIONS

This paper presents a systematic literature review that characterizes the software engineering best practices reported

TABLE XV
ACM-IEEE SE CURRICULUM X "BEST PRACTICES" REPORTED

Units of Knowledge	Related Best Practices
Group Dynamics and Psychology	reflexive practices, mentors, pair programming, collective ownership, forty-hour-week, peer reviews, group leader, activity redistribution, self manageable groups
Communication Skills	project postmortems, metaphor
Professionalism	contacts, reciprocity, feedback, respect
Requirements Fundamentals	requirements
Eliciting Requirements	
Requirements Specification and Documentation	
Requirements Validation	
Design Concepts	simple design
Design Strategies	design, design patterns
Architectural Design	refactoring, object oriented analysis and design
Human Computer Interaction Design	
Detailed Design	
Design Evaluation	
Verification and Validation Foundations	inspections, prototyping
Review and Static Analysis	peer reviews, static analysis
Testing	testing, unit testing, integration testing
Problem Analysis Reporting	
Process Concepts	PSP, formal methods, root cause analysis
Process Implementation	realization, implementation
Project Planning and Tracking	project planning
Software Configuration Management	continuous integration, configuration management, version control, continuous building
Evolution Process	iterative and incremental development, maintenance
Activities	
Software Quality Concepts and Culture	coding standards, quality management
Process Assurance	conformance metrics
Product Assurance	client satisfaction, quality assurance, evaluation
Security Fundamentals	
Computer and Network Security	
Developing Secure Software	security

as being taught in academia. A corpus of literature of 17 primary studies that were published in the main conferences and journals of the area was used, and to select these studies well-known guidelines were followed. The results show that there are a lot of studies that mention the words "best practices" in their content but they do not describe them. And in the few papers where they do describe which best practices are being taught, they do not validate the relevance of teaching these practices or how others can learn and apply the same practices. Sometimes the best practices are even mixed between instructional best practices and software engineering best practices. There is also the problem of granularity, some of the practices reported are self contained and others are a whole knowledge area.

It was established that there are some practices that were reported as being used in more than one primary study, which can be considered a good indicator that they are really software engineering "best practices" that should be taken into consideration within the context of software engineering education, they are: integration, quality, reflection, requirements, pair programming, testing/inspections, design, planning, testing and continuous integration (building).

The "best practices" being taught that were found in the study with the ACM-IEEE Software Engineering Curriculum Guideline were also compared and found the aforementioned issue of granularity; it is not possible to perform a straightforward, objective correlation between the "best practices" and the guidelines to evaluate if there is a real correspondence, so the correlation is subjective to interpretation. In this interpretation; all the knowledge areas are covered with some "best practices"; but when the comparison was done on the level of knowledge unit x "best practices" it showed that 67% of the knowledge units are covered by "best practices".

The low amount of papers reporting "best practices" of software engineering being taught and the amount of coverage that these "best practices" have compared with the ACM-IEEE Curriculum Guidelines; three main open questions were elaborated: the software engineering "best practices are not being taught?; the software engineering "best practices are being taught but they are not reported?; the software engineering "best practices" are not relevant? Which is also questioned by [20] but in the agile security world, a small subset of the one used here.

[9] performed a review where the authors begin their study with an empty set of best practices related to the industry academy collaboration in software engineering. They state that their contribution is a set of best practices where they evaluated and labeled as best practices patterns of successful collaborations between industry-academia in a software engineering context. If the same idea is extrapolated to this work, it is possible to say that now, there is a small set of best practices that are being used to teach software engineering.

With this systematic literature review, the software engineering best practices reported as being taught in academia were presented and they resulted to be just a few. It is intended to continue the work to elucidate software engineering best practices that are currently being used. As stated by Thompson and Fox [30] "these are all indications that if best practices do exist they are certainly not as widely used as they should be, nor does it appear that they are being developed in the light of actual practical experiences", in conclusion this statement continues to be true ten years later.

REFERENCES

[1] Alain Abran, Pierre Bourque, Robert Dupuis, and James W Moore. *Guide to the software engineering body of knowledge-SWEBOK*. IEEE Press, 2001.

[2] Delbert Bailey, Tracey Conn, Brian Hanks, and Linda Werner. Can we influence students' attitudes about inspections? Can we measure a change in attitude? In *Software Engineering Education and Training, 2003.(CSEE&T 2003). Proceedings. 16th Conference on*, pages 260–267. IEEE, 2003.

[3] Chung-Yang Chen and P Pete Chong. Software engineering education: A study on conducting collaborative senior project development. *Journal of Systems and Software*, 84(3):479–491, 2011.

[4] Tony Cowling, Shawn Bohner, and Mark A. Ardis, editors. *26th International Conference on Software Engineering Education and Training, CSEE&T 2013, San Francisco, CA, USA, May 19-21, 2013*. IEEE, 2013.

[5] Brian P Eddy, Norman Wilde, Nathan A Cooper, Bhavyansh Mishra, Valeria S Gamboa, Keenal M Shah, Adrian M Deleon, and Nikolai A Shields. A pilot study on introducing continuous integration and delivery into undergraduate software engineering courses. In *Software Engineering Education and Training (CSEE&T), 2017 IEEE 30th Conference on*, pages 47–56. IEEE, 2017.

[6] Nuno H Flores, Ana CR Paiva, and Pedro Letra. Software engineering management education through game design patterns. *Procedia-Social and Behavioral Sciences*, 228:436–442, 2016.

[7] Stephen T Frezza. Issues in student valuing of software engineering best practices. In *Frontiers in Education Conference (FIE), 2016 IEEE*, pages 1–4. IEEE, 2016.

[8] Gerald C Gannod, Kristen M Bachman, Douglas Troy, Steve D Brockman, et al. Increasing alumni engagement through the capstone experience. In *Frontiers in Education Conference (FIE), 2010 IEEE*, pages F1C–1. IEEE, 2010.

[9] Vahid Garousi, Kai Petersen, and Baris Ozkan. Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review. *Information and Software Technology*, 79:106–127, 2016.

[10] Kevin Gary, Tommie Lindquist, Sunny Bansal, and Arbi Ghazarian. A project spine for software engineering curricular design. In Cowling et al. [4], pages 299–303.

[11] Dennis P Groth and Edward L Robertson. It's all about process: project-oriented teaching of software engineering. In *Software Engineering Education and Training, 2001. Proceedings. 14th Conference on*, pages 7–17. IEEE, 2001.

[12] Thomas B Hilburn, Greg Hislop, Donald J Bagert, Michael Lutz, Susan Mengel, and Michael McCracken. Guidance for the development of software engineering education programs. *Journal of Systems and Software*, 49(2):163–169, 1999.

[13] Stan Jarzabek. Teaching advanced software design in team-based project course. In Cowling et al. [4], pages 31–40.

[14] Barbara Kitchenham and Stuart Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-01, Keele University and Durham University, July 2007.

[15] Supannika Koolmanojwong and Barry Boehm. Using software project courses to integrate education and research: An experience report. In *Software Engineering Education and Training, 2009. CSEE&T'09. 22nd Conference on*, pages 26–33. IEEE, 2009.

[16] Richard J LeBlanc, Ann Sobel, Jorge L Diaz-Herrera, Thomas B Hilburn, et al. *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. IEEE Computer Society, 2006.

[17] Timo OA Lehtinen, Mika V Mäntylä, Juha Itkonen, and Jari Vanhanen. Diagrams or structural lists in software project retrospectives–an experimental comparison. *Journal of Systems and Software*, 103:17–35, 2015.

[18] Liana Barachisio Lisboa, Vinicius Cardoso Garcia, Daniel Lucrédio, Eduardo Santana de Almeida, Silvio Romero de Lemos Meira, and Renata Pontin de Mattos Fortes. A systematic review of domain analysis tools. *Information and Software Technology*, 52(1):1–13, January 2010.

[19] Christoph Matthies, Thomas Kowark, Keven Richly, Matthias Uflacker, and Hasso Plattner. How surveys, tutors, and software help to assess scrum adoption in a classroom software engineering project. In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 313–322. ACM, 2016.

[20] J Todd McDonald, Tyler H Trigg, Clifton E Roberts, and Blake J Darden. Security in agile development: Pedagogic lessons from an undergraduate software engineering case study. In *Cyber Security Symposium*, pages 127–141. Springer, 2015.

[21] Debora Maria Nascimento, Ken Cox, Telmo Almeida, Wendell Sampaio, Roberto Almeida Bittencourt, Richard Souza, and Christina Chavez. Using open source projects in software engineering education: A systematic mapping study. In *Frontiers in Education Conference, 2013 IEEE*, pages 1837–1843. IEEE, 2013.

[22] Crescencio Rodrigues Lima Neto and Eduardo Santana De Almeida. Five years of lessons learned from the software engineering course: adapting best practices for distributed software development. In *Proceedings of the Second International Workshop on Collaborative Teaching of Globally Distributed Software Development*, pages 6–10. IEEE Press, 2012.

[23] Andres Neyem, Juan Diaz-Mosquera, Jorge Munoz-Gama, and Jaime Navon. Understanding student interactions in capstone courses to improve learning experiences. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 423–428. ACM, 2017.

[24] Wilson P Paula Filho. A model-driven software process for course projects. In *Proceedings of the Educators' Symposium of the ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems, Montego Bay, Jamaica*, pages 33–40. Citeseer, 2005.

[25] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering*, volume 17. sn, 2008.

[26] Mark Petticrew and Helen Roberts. *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons, 2008.

[27] Dominik Rost, Balthasar Weitzel, Matthias Naab, Torsten Lenhart, and Hartmut Schmitt. Distilling best practices for agile development from architecture methodology. In *European Conference on Software Architecture*, pages 259–267. Springer, 2015.

[28] Jean-Guy Schneider and Lorraine Johnston. eXtreme Programming—-helpful or harmful in educating undergraduates? *Journal of Systems and Software*, 74(2):121–132, 2005.

[29] K Surendran, Hays H, and Macfarlane A. Simulating a software engineering apprenticeship. *IEEE SOFTWARE*, 19(5):49–56, Sep 2002.

[30] J Barrie Thompson and Anthony J Fox. Best Practice: Is this the Cinderella Area of Software Engineering? In *18th Conference on Software Engineering Education and Training, CSEE&T 2005, 18-20 April 2005, Ottawa, Canada*, pages 137–144. IEEE Computer Society, 2005.

[31] James E Tomayko. Carnegie Mellon's software development studio: a five year retrospective. In *Software Engineering Education, 1996. Proceedings., Ninth Conference on*, pages 119–129. IEEE, 1996.

[32] Richard L Upchurch and Judith E Sims-Knight. Reflective essays in software engineering. In *Frontiers in Education Conference, 1999. FIE'99. 29th Annual*, volume 3, pages 13A6–13. IEEE, 1999.

[33] Louwarnoud Van Der Duim, Jesper Andersson, and Marco Sinnema. Good practices for educational software engineering projects. In *Proceedings of the 29th International conference on Software Engineering*, pages 698–707. IEEE Computer Society, 2007.