# A Systematic Mapping Study on Software Process Education

Alberto Heredia
Universidad Carlos III de Madrid
Computer Science Department
Leganés 28911, Spain
alberto.heredia@uc3m.es

Ricardo Colomo-Palacios
Østfold University College
Faculty of Computer Sciences
Halden 1783, Norway
ricardo.colomo-palacios@hiof.no

Antonio Amescua-Seco
Universidad Carlos III de Madrid
Computer Science Department
Leganés 28911, Spain
amescua@inf.uc3m.es

## Abstract

Software professionals often face trouble when developing software products as it is a highly dynamic, knowledge-intensive complex process. The success of the software process heavily depends on the people involved, among other factors, making their education and training an interesting topic for research. The purpose of this study is to structure and characterize the state of the practice on software process education to help identify best practices and find new challenges. To do so, authors conducted a systematic mapping study to identify primary studies in the existing literature related to software process education. The analysis of results helps clarify the general characteristics of the software process education and training initiatives, the lessons learned in previous research, and the future works proposed by the authors in previous research on software process education.

## 1 Introduction

Modern societies increasingly depend on the services offered through computerized systems. The advent of smartphones, tablets, wearables and other intelligent devices makes that more and more products embed or take advantage of some piece of software. Unfortunately, software is a complex product, difficult to develop [FuNi14].

Software Engineering has the main goal of creating software products with quality, respecting time and budget constraints [Hump95]. To do so, the software development activity usually follows a software process, which can be defined as the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product [Fugg00], i.e., it describes the approach that is taken as software is engineered.

However, the controversial reports from The Standish Group continuously mention a low percentage of successful projects delivering software on time, on budget, and with required features and functions. Other forums, such as Risk Digest [Acmc15], constantly document numerous examples of software failures that could be harmful for the society, e.g., the accidental erasure of criminal records or the exposure of private data from online customer databases.

Many of these problems found in software products are unintentionally caused by people [KuFM13], as software in the end is developed by individuals and is largely dependent on human capital [CCGG09, CCMS14, CCSG13a, HeCG13]. Thus, it is worth researching how they are educated and trained on the process to follow for the development of a software product[CCSG13b, RoZS14].

Training software engineers in order for them to acquire the knowledge and skills required in professional practice depends on the stage of their careers. As an example, software engineering courses at the university usually consist of lectures along with a small software project [BaOH05], but software process is often treated as an additional module to the core curriculum. Trainings in an industry environment are, on the other hand, organized in a workshop style with theoretical and practical parts interwoven [KuFM13]. Yet it is not clear if this education and training –no matter the way it is provided– effectively prepares software process (improvement) practitioners as skilled and competent professionals for industrial life.

In fact, software engineering professionals are often unsatisfied with their level of preparation for the real-world when they start working in industry [Exte14]. Some authors point out the root of the problem lies in the way software process is typically taught at universities [AlUn14, BaOH05]; due to the time and scope constraints inherent in an academic setting, most



Proceedings of the International Workshop on Software Process Education, Training and Professionalism, Gothenburg, Sweden

20015-06-15 published at http://ceur-ws.org

course projects leaves little room for experiencing the many facets of the software lifecycle [KoCM14].

Many authors have researched on how to make improvements in software process education and training to overcome this issue using different approaches. The first one lays on a specific subject that is needed but currently missing or not properly addressed [WaSB12]. Another approach aims at bringing the class project closer to a real-world one, for instance, by intentionally applying unexpected complications during the project [Daws00] or involving external organizations [ChCh11]. A third approach uses a simulated environment in conjunction with lectures and projects for enhancing the learning and understanding of complex themes [BaOH05]. Finally, the gamification of learning has emerged as a significant trend in recent years in an effort to make education more attractive by means of incorporating game mechanics and elements [PGBP15].

Regardless of the approach chosen, it is also important to consider how instructors intend their students to learn. The most traditional delivery method consists of a series of lectures and demonstrations in which the teacher presents a particular subject and directly instructs students. This method is often contrasted to experiential learning, which is based upon the premise that the best way to learn how to do something is by actually doing it [BaOH05]. Other methods center learning around an anchor such as a case study or a problem [BSHK90], foster a situated learning in which the learning environment is closer to reality [AnRS96], focus on the aptitude of students and tailor the learning environment to their needs [Yeh12], emphasize a lateral thinking that require students to take different perspectives [Bono09], or just focus on motivating students to learn [Kell87].

We thus need to further study how software engineers learn the software process. The objective of this paper is to structure and characterize the state of the practice on software process education. In consequence, the authors of this study conducted a systematic mapping study to identify, select, classify and analyze primary studies published in scientific journals. To the best of our knowledge, no systematic mapping study on software process education has been published yet.

The remainder of this paper proceeds as follows. Section 2 describes the method followed in this research work. Section 3 analyzes and discusses the results of the systematic mapping study. The paper concludes with the limitations of this research and concluding remarks.

## 2 Research Method

The purpose of this study is to structure and characterize the state of the practice on software process education, analyzing previous works published in the literature to provide an overview of the topic and to help discover potential gaps for future research. Thus, the main research question driving this study is:

What is the state of the practice of the education on software process?

Due to the breadth of the topic, a systematic mapping study [KiBP11] is used to identify and categorize all relevant research papers (referred to as primary studies) related to software process education. The study follows the guidelines provided by Petersen et al. [PFMM08]. The following sub-sections present the different stages of the mapping study: definition of research questions, conducting the search for primary studies, screening papers based on inclusion/exclusion criteria, classifying the papers, and data extraction and aggregation.

### 2.1 Research Questions

To answer the main research question driving this mapping study, the authors of this study stated the following specific research questions:

RQ1. What are the general characteristics of the software process education and training initiatives?

RQ2. What lessons did researchers learned from previous research on software process education?

RQ3. What future works did authors propose in previous research on software process education?

The answer to RQ1 will help determine different aspects of the software process education such as which stage of software engineers' career does this education usually focus on, the educational methods that are typically followed, how this education is usually delivered, or which parts of the software process have not received much attention yet with regard to software process education. The aim of RQ2

is to identify best practices on the field. RQ3 gathers challenges identified in the field of software process education.

### 2.2 Search Strategy

The search strategy is key to ensure a good starting point for the identification of studies and ultimately for the actual outcome of the study. An extensive and broad set of primary studies was needed to answer the research questions. The most popular academic databases in the domain of software engineering were selected to be used in this systematic mapping to search for potentially relevant papers:

- ACM Digital Library (http://dl.acm.org)
- IEEE Xplore Digital Library (http://ieeexplore.ieee.org)
- ScienceDirect (http://www.sciencedirect.com)
- Springer Link (http://link.springer.com)

Regarding the keywords for the search, after some exploratory searches using different combination of keywords, the researchers jointly established the final string to be used in the search for papers in the databases:

"software process" AND (education OR training)

The search was performed at the beginning of 2015. The search string was applied to title, abstract and keywords, and limited to journal papers written in English in the area of Computer Science and published between the years 2000 and 2014. A total of 1450 papers were retrieved from the different databases. Unfortunately, despite using the advanced search, only IEEE's database seems to properly retrieve exact phrases in title, abstract and keywords, so this set had to be revised and only 253 unique papers were finally considered for the study selection (Figure 1).

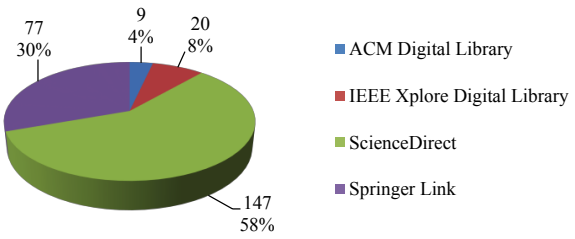


Figure 1: Selected databases and retrieved papers

### 2.3 Study Selection

The main guiding criterion to include a paper in the study or not was its focus on software process education. To reduce the possibility of researcher bias, the authors jointly agreed the exclusion criteria to be used in the following order:

- Based on title: the title does not suggest that there is any relation to software process education.
- Based on abstract: the abstract shows the paper is not focused on software process education.
- Based on full text: the paper is definitely not related to software process education.

In those cases where there was disagreement between researchers regarding the relevancy of a paper, the paper was not finally excluded.

The authors of this study must point out that the revision of the full text of the primary studies allowed to assure that all of them were relevant for structuring and characterizing the state of the practice of the education on software process. This revision is also important because this study does not contain a formal quality evaluation of the primary studies, which indeed is not essential in mapping studies and could not be properly achieved due to the inclusive nature of the search that includes theoretical studies as well as empirical studies of all types [KiBP11].

After the exclusion of irrelevant papers, the researchers finally agreed on 33 primary studies to be included in the systematic mapping study (Table 1). The full list of primary studies is listed in the appendix.

Table 1: Study selection reading detail

| Reading detail | # of studies |
|---|---|
| **Search** | 253 |
| **Title** | 95 |
| **Abstract** | 53 |
| **Full-text** | 33 |

### 2.4 Study classification

A data extraction form was designed to collect relevant information from each one of the selected primary studies. It included the following properties: title, authors, year, journal, number of citations in the ISI Web of Knowledge, type of participants in the educational initiative, educational method, mode of delivery, focus of the initiative, lessons learned in the initiative, and future work proposed.

The authors agreed in classifying primary studies depending on three different types of participants in the educational initiative: undergraduates, graduates and industry professionals.

Regarding the different educational methods and attending to the background of this research described previously, the authors decided to classify the primary studies in these groups: lectures, exercises, project, teaching a missing subject, adding realism to a project, inclusion of simulation in practical classes, and gamification.

Finally, for classifying the main mode of delivery used by the initiative the authors agreed in the following ones: traditional, experiential (learning by doing), anchored instruction, aptitude-treatment interaction, situated learning, lateral thinking, and motivation.

### 2.5 Data extraction and synthesis of results

This section synthesizes the results produced by the extraction of data from the primary studies according to the protocol described above.

The distribution of primary studies does not vary much throughout the years considered in this mapping study. Number of publications fluctuates mainly between 1 and 3, being 2002 and 2008 the most productive years with 5 publications.

Data extracted from primary studies revealed that a total of 77 different authors published papers on the topic of software process education. It is not a surprise to find W.S. Humphrey is the most prolific author among the primary studies with 3 papers as he created the Personal Software Process (PSP), which is one of the processes often used in software process education.

Regarding the journals that published the primary studies, IEEE Software is the journal that accepted the most publications (7) related to software process education, closely followed by the Journal of Computing Sciences in Colleges, the Journal of Systems and Software, Information and Software Technology, and the Software Quality Journal.

Similarly, papers published in IEEE Software sum the largest amount of citations in ISI (64), given that the Journal of Computing Sciences in Colleges (79 citations in Google Scholar) is not indexed in ISI. Taking into account the number of papers, journals such as IEEE Transactions on Engineering Management, IEEE Transactions on Software Engineering and Annals of Software Engineering have a better average of citations in ISI as they published only one of the primary studies that, however, received a significant amount of citations.

To provide a better overview of the field, Figure 2 depicts the types of students involved in the initiatives described in the primary studies, Figure 3 shows the educational methods followed in the initiatives described in the primary studies, and Figure 4 illustrates the modes used for delivering education in the initiatives described in the primary studies. The authors must point out that some primary studies involved more than one type of students, followed more than one method and/or used more than one mode of delivery in their educational initiatives.

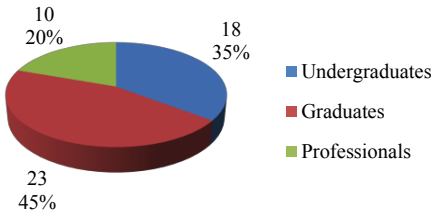


Figure 2: Participants in the primary studies

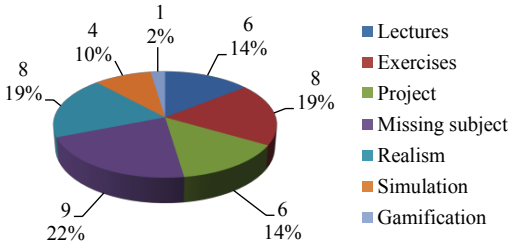


Figure 3: Educational methods in the primary studies

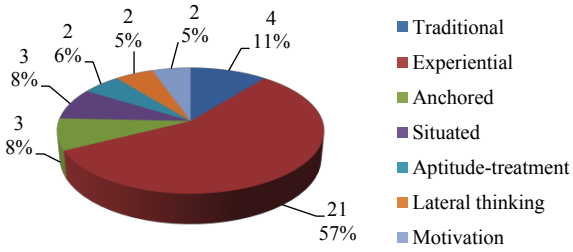


Figure 4: Modes of delivery in the primary studies

## 3 Analysis and discussion of results

In the following sub-sections the authors analyze and discuss the results produced by conducting the systematic mapping study according to the steps described in the previous section in order to find answers to the research questions of this study.

### 3.1 What are the general characteristics of the software process education and training initiatives? (RQ1)

According to the results shown in Figure 2, the majority of studies related to software process education focus research on early stages of software engineers' career, graduates (23) and undergraduates (18), while few of them focus on education for industry professionals (10).

Some of the studies represented in Figure 3 follow the most traditional method for educating future software engineers (6) consisting in a series of lectures combined with some exercises and/or a small project to put acquired knowledge into practice; these studies are usually oriented to undergraduates. Most of the studies, however, describe initiatives to cover a subject that is usually missing in software process education curricula (9); these initiatives aimed at completing the education on software process are usually based also in the combination of lectures with exercises and/or a small project. Another method which is broadly used (8) is making students' project experience closer to the real world (e.g. using an external customer [S23]). Several experiences with simulations (4) to improve software process education have been also reported in the last years; these simulations are often oriented to graduates, and especially to industry professionals, in conjunction with other initiatives based on task assignments, tutorials and workshops. Finally, only one primary study is related to gamification [S17].

Given that many of the primary studies report initiatives using exercises, it is not strange finding experiential learning is the most used delivery method by far (21), see Figure 4; as Albert Einstein once said: "Learning is experience, everything else is information". Results also show that modes of delivery such as situational learning or motivation are generally used when adding realism to a project; while the former is basically used with undergraduates, the latter is mainly used when education is oriented to industry professionals.

To conclude this section, the authors found several of these approaches focus on teaching a specific software process such as the PSP (7) or the TSP (5), while others train students in iterative and agile software development methods (10). Still many of them put emphasis on process improvement training (8), mainly related to CMM and CMMI. Only 3 of the primary studies deal with software process education from the point of view of Project Management. Finally, the remainder focuses on specific parts of the software process such as design, programming or document inspection.

### 3.2 What lessons did researchers learned from previous research on software process education? (RQ2)

Previous research on software process education has provided numerous and various lessons learned. In the following paragraphs the authors cover the most relevant ones found in the scope of this mapping study.

In general, introducing processes into the classroom environment is not easier than injecting them into the workplace, so future researchers should take some considerations into account. Matching the software process weight to the students' abilities, expectations and tolerance is vital for success [S25]. Furthermore, although the use of model representations eases the understanding of the process and increases visibility, giving the students a written process description is not enough; instructors must also provide guidance in the form of mentoring to have a major impact [S6]. Motivation is also essential as engaged and motivated students are more likely to accept the software process [S29].

In addition, if **tools** are used to support process activities, they should be easy to learn and use to create a positive attitude towards their adoption [S2]; despite their learning curve, software process tools have proven to be important to the successful development of projects. In some cases, using a knowledge repository can facilitate the learning process and the transfer of knowledge among students [S20]; low-experienced software engineers can gain experience from more experienced ones and giving them more autonomy [HGAS13].

With regard to software development methods, results point out that **Agile** works well for student projects in an introductory software engineering course

[S26]. Such incremental and iterative approaches allow students to learn from preceding iterations and incorporate previous experience and feedback into the next iteration [S27].

**PSP and TSP** are also good means to introduce discipline concepts and software process to potential engineers because they show students how to define processes, how to use a defined process, how to plan, measure and track their work, and how to measure and manage quality [S31]. Results gathered from the primary studies confirm the benefits of training students on the PSP. It enhances predictability and reduces the number of (trivial) defects in the code, although students may require more time for finishing tasks because of the error checking that leads to the improved robustness [S8].

The authors in [S28] and [S29] provide some recommendations for using PSP and TSP as discipline drivers in software process education: 1) Customize PSP and TSP courses to the context and the needs of the students; 2) Integrate PSP as part of TSP in order for students to first master PSP techniques before assuming a role in a software development group; 3) Arrange PSP training regularly and continuously to ensure that a student can meet both essential and accidental software challenges (actually, some authors state that learners should apply PSP practices not just in a single course, but as a regular part of their studies for instilling good habits and professional attitudes); 4) Motivate students about the benefits of PSP and TSP; and 5) Let students see their progress through the data collected, but these data should not be used for grading purposes in order to reduce the likelihood of students manipulating the values in an attempt to gain better grades.

Another interesting recommendation found in the primary studies is tailoring the assignments the course to imitate the real-world software projects [S12]. **Realism** has to be seek so that when a process-related problem arises, the process should be improved in order to not repeat the same problem in subsequent projects. To increase reality, instructors can promote collaboration between students and external customers [S33], provided that customers' involvement may help to produce software better adapted to real expectations. However, there is a risk of students giving more attention to the product than the process as customers are interested in the product [S23]. On the other hand, facilities such as studios [S24] not only bring home a great opportunity to take theory into practice, but also provide students with environments and experiences they will encounter or maybe even bring to their future jobs.

When using a **project** to educate on software process or in senior **capstone courses**, students can use everything they already learned [S14]. The use of **dynamic teams** [S27] in these projects is a good experience because it challenges students to adapt to multiple personalities and skill sets; they can learn from one another, they feel more comfortable in rating peers honestly, and it leads to fewer group breakdowns when team members underperform. Other practices such as **pair designing** [S18] may slow down the project, but it is more predictable than individual designing with regards to quality.

To improve the likelihood of successfully design and implement a software project course, researchers should follow several guidelines [S30]: 1) Clearly identify course goals; 2) If the course is time-restricted or represents students' first team project experience, use a modest and well-defined problem; 3) Use a defined team process for the project work; 4) Enforce process discipline; and 5) Instructors should move their role from lecturer to coach.

Concerning **process improvement training**, CMMI-recommended practices are accepted across much of the industry and thus they are a good reference for software process education efforts. Results of previous research [S22] revealed some hot spots that require more training in software process programs (e.g. organizational practices). It may be beneficial to dedicate significant time to provide details about process models such as CMMI at the graduate level, but it may not be appropriate at the undergraduate level [S3]. Therefore, researchers recommend addressing individual skills at the undergraduate level and management skills at the graduate level.

With regard to **gamification**, it proved to have potential to support education [S17], although further research is needed. In this sense, [HCAY14] presents a Gamification approach for software process, but not linked to education or training. Likewise, **simulation** seems to be very useful because allows students to change process settings and helps decide if a process is suitable for a certain context [S10]. When researching on the benefits of simulation for educating on the software process, researchers must take into account that it is not an inexpensive undertaking and students

need time for the familiarization with the simulator [S1]. Yet, there is little evidence that process simulation has become an accepted and regularly used tool in industry [S4]. Moreover, the use of simulation techniques like, for instance System Dynamics is well grounded in software engineering education [GCGP08].

To conclude this section, researches should consider learning from practitioners of other engineering disciplines [S19], as their lessons learned can be useful for software engineering too.

### 3.3 What future works did authors propose in previous research on software process education? (RQ3)

In spite of the large amount of lesson learned gathered from previous research on software process education, not many of the primary studies propose future works. The most common ones proposed exporting described initiatives to other universities [S16] or to the industry [S15].

More interesting proposals, especially those focused on simulation and gamification, suggested enhancing complexity and variability to allow a more dynamic learning experience [S17]. In addition, future research could consider the extension of the single-learner model towards a collaborative learning environment [S1]. Nevertheless, there is still a need for providing evidence of the usefulness of simulation in the real-world and additional studies of long-term evolution from a product and organizational perspective [S4].

Primary studies related to the PSP raise several questions to address with further studies regarding the degree to which PSP students make more balanced estimates, the relationship between productivity and effort estimation accuracy, whether planning time and postmortem time are dependent on project size or whether they are more or less constant and could be viewed as overhead [S28]. Other additional important questions could be: How will defect estimation behave in further studies? How could we prepare a set of exercises that allows us to separate the complexity of exercises from the PSP levels? To what extent is a virtual environment the most appropriate tool for teaching discipline teamwork? What kind of feedback is received best as motivation by the students: defects, size estimation or effort estimation?

To conclude the answer to this research question, studies considering issues related to human capital suggest incorporating ethical and social aspects of ICTs in computer science programs and developing awareness of potential threats posed by new ICTs among today's students [S33]. Others propose analyses of the impact of outdated technology skills or about attitudes toward software process innovations [S5].

## 4 Limitations

The objective of this study was to structure and characterize the state of the practice on software process education, analyzing previous works published in the literature to provide an overview of the topic and to help discover potential gaps for future research. For that purpose, the authors decided to use a general search string to not bias the study towards any specific educational method or mode of delivery. However, other searches using keywords related to specific educational method, such as realism or simulation, or mode of delivery, such as lateral thinking or situated learning, could provide more primary studies. This limitation makes this study to be a first step towards a future research that could include a systematic literature review centered on new approaches for the education on software process based on trending modes of delivery such as flipped learning or Massive Open Online Courses (MOOCs).

Similarly, due to the specific focus of this 1st International Workshop on Software Process Education, Training and Professionalism, the authors decided to include just the term "software process" and not the term "software engineering" in the search string. Broadening the scope of this research to software engineering education and not focusing only in the software process would have provided a richer set of primary studies and should be considered for a future work.

The exclusion of conference papers and books represent another limitation of this study. This publication bias is based mainly on practical concerns; the amount of primary studies to be included could have been unmanageable and a lot of analysis would be needed to handle the fact that many journal papers are improvements of previously published conference papers. Nevertheless, the inclusion of journal papers guarantees a high scientific quality of the primary studies. However, and in spite of the inclusion of

journals, given the composition of databases for the study, some papers published in journals not listed in the databases can also be biased in this study.

Finally, another threat for this study is researcher bias that could have affected the selection of primary studies, their classification and the accuracy in data extraction. To reduce the subjective component of this study, two researchers participated in the selection and classification of primary studies following a multi-staged protocol for the inclusion and exclusion criteria and resolving disagreements by discussion.

## 5 Conclusions and future work

Software process improvement is considered one of the most important fields in the software engineering discipline. However, and in spite of its importance, increasing its coverage in educational settings is still challenging. The complexity of the subject together with the need of a good background of the discipline is normally pushing subjects into master programs, while PSP and TSP approaches are mostly present in bachelor curricula. This paper is a first effort towards understanding the subject and interpreting its needs and implementation in the academia.

Future works will be twofold. Firstly, it is intended to investigate the use of MOOCs in software process improvement settings and secondly, it is aimed to develop specific gamification strategies and tools for software process improvement education and training.

## References


[Acmc15] ACM COMMITTEE ON COMPUTERS AND PUBLIC POLICY: *The Risks Digest*. URL http://catless.ncl.ac.uk/risks. - abgerufen am 2015-02-27

[AlUn14] BIN ALI, NAUMAN ; UNTERKALMSTEINER, MICHAEL: Use and evaluation of simulation for software process education: a case study. In: . Seeon Monastery, Germany : Shaker Verlag, 2014, S. 59–73

[AnRS96] ANDERSON, JOHN R. ; REDER, LYNNE M. ; SIMON, HERBERT A.: Situated Learning and Education. In: *Educational Researcher* Bd. 25 (1996), Nr. 4, S. 5–11

[BaOH05] BAKER, ALEX ; OH NAVARRO, EMILY ; VAN DER HOEK, ANDRÉ: An experimental card game for teaching software engineering processes. In: *Journal of Systems and Software*, *Software Engineering Education and Training*. Bd. 75 (2005), Nr. 1–2, S. 3–16

[Bono09] BONO, EDWARD DE: *Lateral Thinking: A Textbook of Creativity* : Penguin UK, 2009 — ISBN 9780141938318

[BSHK90] BRANSFORD, JOHN D ; SHERWOOD, ROBERT D ; HASSELBRING, TED S ; KINZER, CHARLES K ; WILLIAMS, SUSAN M: Anchored instruction: Why we need it and how technology can help. In: NIX, D. ; SPIRO, R. (Hrsg.): *Cognition, education, and multimedia: Exploring ideas in high technology*. Hillsdale, NJ. : Lawrence Erlbaum, 1990, S. 115–141

[CCGG09] CASADO-LUMBRERAS, C. ; COLOMO-PALACIOS, R. ; GOMEZ-BERBIS, J.M. ; GARCIA-CRESPO, A.: Mentoring programmes: a study of the Spanish software industry. In: *International Journal of Learning and Intellectual Capital* Bd. 6 (2009), Nr. 3, S. 293–302

[CCMS14] COLOMO-PALACIOS, RICARDO ; CASADO-LUMBRERAS, CRISTINA ; MISRA, SANJAY ; SOTO-ACOSTA, PEDRO: Career Abandonment Intentions among Software Workers. In: *Human Factors and Ergonomics in Manufacturing & Service Industries* Bd. 24 (2014), Nr. 6, S. 641–655

[CCSG13a] COLOMO-PALACIOS, RICARDO ; CASADO-LUMBRERAS, CRISTINA ; SOTO-ACOSTA, PEDRO ; GARCÍA-CRESPO, ÁNGEL: Decisions in software development projects management. An exploratory study. In: *Behaviour & Information Technology* Bd. 32 (2013), Nr. 11, S. 1077–1085

[CCSG13b] COLOMO-PALACIOS, R. ; CASADO-LUMBRERAS, CRISTINA ; SOTO-ACOSTA, PEDRO ; GARCÍA-PEÑALVO, FRANCISCO J. ; TOVAR-CARO, EDMUNDO: Competence gaps in software personnel: A multi-organizational study. In: *Computers in Human Behavior*, *Advanced Human-Computer Interaction*. Bd. 29 (2013), Nr. 2, S. 456–461

[ChCh11] CHEN, CHUNG-YANG ; CHONG, P. PETE: Software engineering education: A study on conducting collaborative senior project

development. In: *Journal of Systems and Software* Bd. 84 (2011), Nr. 3, S. 479–491

[Daws00] DAWSON, RAY: Twenty Dirty Tricks to Train Software Engineers. In: *Proceedings of the 22Nd International Conference on Software Engineering*, *ICSE '00*. New York, NY, USA : ACM, 2000 — ISBN 1-58113-206-9, S. 209–218

[Exte14] EXTER, MARISA: Comparing Educational Experiences and On-the-job Needs of Educational Software Designers. In: *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, *SIGCSE '14*. New York, NY, USA : ACM, 2014 — ISBN 978-1-4503-2605-6, S. 355–360

[Fugg00] FUGGETTA, ALFONSO: Software Process: A Roadmap. In: *Proceedings of the Conference on The Future of Software Engineering*, *ICSE '00*. New York, NY, USA : ACM, 2000 — ISBN 1-58113-253-0, S. 25–34

[FuNi14] FUGGETTA, ALFONSO ; DI NITTO, ELISABETTA: Software Process. In: *Proceedings of the on Future of Software Engineering*, *FOSE 2014*. New York, NY, USA : ACM, 2014 — ISBN 978-1-4503-2865-4, S. 1–12

[GCGP08] GARCÍA-CRESPO, ÁNGEL ; COLOMO-PALACIOS, R ; GOMEZ-BERBIS, MJ ; PANIAGUA-MARTIN, F: A Case of System Dynamics Education in Software Engineering Courses. In: *IEEE Multidisciplinary EngineeringEducation Magazine* Bd. 32 (2008), S. 52–59

[HCAY14] HERRANZ, EDUARDO ; COLOMO-PALACIOS, RICARDO ; DE AMESCUA SECO, ANTONIO ; YILMAZ, MURAT: Gamification as a Disruptive Factor in Software Process Improvement Initiatives. In: *j-jucs* Bd. 20 (2014), Nr. 6, S. 885–906. — |http://www.jucs.org/jucs_20_6/gamification_as_a_disruptive|

[HeCG13] HERNÁNDEZ-LÓPEZ, ADRIÁN ; COLOMO-PALACIOS, RICARDO ; GARCÍA-CRESPO, ÁNGEL: Software engineering job productivity — a systematic review. In: *International Journal of Software Engineering and Knowledge Engineering* Bd. 23 (2013), Nr. 03, S. 387–406

[HGAS13] HEREDIA, ALBERTO ; GUZMÁN, JAVIER GARCÍA ; AMESCUA, ANTONIO ; SEGURA, MARIA ISABEL SÁNCHEZ: Interactive Knowledge Asset Management: Acquiring and Disseminating Tacit Knowledge. In: *Journal of Information Science and Engineering* Bd. 29 (2013), Nr. 1, S. 133–147

[Hump95] HUMPHREY, WATTS S.: *A Discipline for Software Engineering*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1995 — ISBN 0201847485

[Kell87] KELLER, JOHN M.: Development and use of the ARCS model of instructional design. In: *Journal of instructional development* Bd. 10 (1987), Nr. 3, S. 2–10

[KiBP11] KITCHENHAM, BARBARA A. ; BUDGEN, DAVID ; PEARL BRERETON, O.: Using mapping studies as the basis for further research – A participant-observer case study. In: *Information and Software Technology*, *Special Section: Best papers from the APSEC Best papers from the APSEC*. Bd. 53 (2011), Nr. 6, S. 638–651

[KoCM14] KOHWALTER, T.C. ; CLUA, E.W.G. ; MURTA, L.G.P.: Reinforcing Software Engineering Learning through Provenance. In: *2014 Brazilian Symposium on Software Engineering (SBES)*, 2014, S. 131–140

[KuFM13] KUHRMANN, MARCO ; FERNÁNDEZ, DANIEL MÉNDEZ ; MÜNCH, JÜRGEN: Teaching Software Process Modeling. In: *Proceedings of the 2013 International Conference on Software Engineering*, *ICSE '13*. Piscataway, NJ, USA : IEEE Press, 2013 — ISBN 978-1-4673-3076-3, S. 1138–1147

[PFMM08] PETERSEN, KAI ; FELDT, ROBERT ; MUJTABA, SHAHID ; MATTSSON, MICHAEL: Systematic Mapping Studies in Software Engineering. In: *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, *EASE'08*. Swinton, UK, UK : British Computer Society, 2008, S. 68–77

[PGBP15] PEDREIRA, OSCAR ; GARCÍA, FÉLIX ; BRISABOA, NIEVES ; PIATTINI, MARIO: Gamification in software engineering – A systematic mapping. In: *Information and Software Technology* Bd. 57 (2015), S. 157–168

[RoZS14] RONG, GUOPING ; ZHANG, HE ; SHAO, DONG: Where does experience matter in software process education? An experience report. In: *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE T)*, 2014, S. 129–138

[WaSB12] VON WANGENHEIM, CHRISTIANE GRESSE ; SAVI, RAFAEL ; BORGATTO, ADRIANO FERRETI: DELIVER! – An educational game for teaching Earned Value Management in computing courses. In: *Information and Software Technology* Bd. 54 (2012), Nr. 3, S. 286–298

[Yeh12] YEH, YU-CHU: Aptitude-Treatment Interaction. In: SEEL, P. D. N. M. (Hrsg.): *Encyclopedia of the Sciences of Learning* : Springer US, 2012 — ISBN 978-1-4419-1427-9, 978-1-4419-1428-6, S. 295–298

## Appendix: Primary studies selected for the systematic mapping study

[S1] Pfahl, D., Klemm, M., Ruhe, G.: A CBT module with integrated simulation component for software project management education and training. Journal of Systems and Software. 59, 283–298 (2001).

[S2] Agarwal, R., Prasad, J.: A field study of the adoption of software process innovations by information systems professionals. IEEE Transactions on Engineering Management. 47, 295–308 (2000).

[S3] Biberoglu, E., Haddad, H.: A Survey of Industrial Experiences with CMM and the Teaching of CMM Practices. Journal of Computing Sciences in Colleges. 18, 143–152 (2002).

[S4] Ali, N.B., Petersen, K., Wohlin, C.: A systematic literature review on the industrial use of software process simulation. Journal of Systems and Software. 97, 65–85 (2014).

[S5] Matalonga, S., Solari, M., Feliu, T.S.: An empirically validated simulation for understanding the relationship between process conformance and technology skills. Software Qual J. 22, 593–609 (2013).

[S6] Elliott, M., Dawson, R., Edwards, J.: An evolutionary cultural-change approach to successful software process improvement. Software Qual J. 17, 189–202 (2009).

[S7] Kamatar, J., Hayes, W.: An experience report on the personal software process. IEEE Software. 17, 85–89 (2000).

[S8] Prechelt, L., Unger, B.: An experiment measuring the effects of personal software process (PSP) training. IEEE Transactions on Software Engineering. 27, 465–472 (2001).

[S9] Morisio, M.: Applying the PSP in industry. IEEE Software. 17, 90–95 (2000).

[S10] Hsueh, N.-L., Shen, W.-H., Yang, Z.-W., Yang, D.-L.: Applying UML and software simulation for process definition, verification, and validation. Information and Software Technology. 50, 897–911 (2008).

[S11] Sampaio, A., Vasconcelos, A., Sampaio, P.R.F.: Assessing agile methods: An empirical study. J Braz Comp Soc. 10, 21–48 (2004).

[S12] Shen, W.-H., Hsueh, N.-L., Lee, W.-M.: Assessing PSP effect in training disciplined software development: A Plan–Track–Review model. Information and Software Technology. 53, 137–148 (2011).

[S13] Carver, J., Shull, F., Basili, V.: Can observational techniques help novices overcome the software inspection learning curve? An empirical investigation. Empir Software Eng. 11, 523–539 (2006).

[S14] Beasley, R.E.: Conducting a Successful Senior Capstone Course in Computing. Journal of Computing Sciences in Colleges. 19, 122–131 (2003).

[S15] García, J., Amescua, A., Sánchez, M.-I., Bermón, L.: Design guidelines for software processes knowledge repository development. Information and Software Technology. 53, 834–850 (2011).

[S16] Bagert, D.J., Mengel, S.A.: Developing and using a web-based project process throughout

the software engineering curriculum. Journal of Systems and Software. 74, 113–120 (2005).
[S17] Wangenheim, C.G. von, Thiry, M., Kochanski, D.: Empirical evaluation of an educational game on software measurement. Empir Software Eng. 14, 418–452 (2008).
[S18] Canfora, G., Cimitile, A., Garcia, F., Piattini, M., Visaggio, C.A.: Evaluating performances of pair designing in industry. Journal of Systems and Software. 80, 1317–1327 (2007).
[S19] Hantos, P., Gisbert, M.: Identifying software productivity improvement approaches and risks: construction industry case study. IEEE Software. 17, 48–56 (2000).
[S20] Amescua, A., Bermón, L., García, J., Sánchez-Segura, M.-I.: Knowledge repository to improve agile development processes learning. IET Software. 4, 434–444 (2010).
[S21] Guzmán, J.G., Martín, D., Urbano, J., Amescua, A. de: Practical experiences in modelling software engineering practices: The project patterns approach. Software Qual J. 21, 325–354 (2012).
[S22] Moreno, A.M., Sánchez-Segura, M.-I., Medina-Dominguez, F., Cuevas, G.: Process Improvement from an Academic Perspective: How Could Software Engineering Education Contribute to CMMI Practices? IEEE Software. 31, 91–97 (2014).
[S23] Tadayon, N.: Software Engineering Based on the Team Software Process with a Real World Project. Journal of Computing Sciences in Colleges. 19, 133–142 (2004).
[S24] Niño, J.: Software Engineering Team Studios. Journal of Computing Sciences in Colleges. 23, 59–65 (2008).
[S25] Umphress, D.A., Hendrix, T.D., Cross, J.H.: Software process in the classroom: the Capstone project experience. IEEE Software. 19, 78–81 (2002).
[S26] Hart, D.: Supporting Agile Processes in Software Engineering Courses. Journal of Computing Sciences in Colleges. 25, 136–143 (2010).
[S27] Anewalt, K., Polack-Wahl, J.A.: Teaching an Iterative Approach with Rotating Groups in an Undergraduate Software Engineering Course. Journal of Computing Sciences in Colleges. 25, 144–151 (2010).
[S28] Rombach, D., Münch, J., Ocampo, A., Humphrey, W.S., Burton, D.: Teaching disciplined software development. Journal of Systems and Software. 81, 747–763 (2008).
[S29] Borstler, J., Carrington, D., Hislop, G.W., Lisack, S., Olson, K., Williams, L.: Teaching PSP: challenges and lessons learned. IEEE Software. 19, 42–48 (2002).
[S30] Hilburn, T.B., Humphrey, W.S.: Teaching teamwork. IEEE Software. 19, 72–77 (2002).
[S31] Humphrey, W.S.: Three Process Perspectives: Organizations, Teams, and People. Annals of Software Engineering. 14, 39–72 (2002).
[S32] Germain, É., Robillard, P.N.: Towards software process patterns: An empirical analysis of the behavior of student teams. Information and Software Technology. 50, 1088–1097 (2008).
[S33] Begier, B.: Users' involvement may help respect social and ethical values and improve software quality. Inf Syst Front. 12, 389–397 (2009).