Journal of Engineering Science and Technology
Vol. 10, No. 3 (2015) 269 - 281

NECESSITY OF LEARNING ANALYTICS IN SOFTWARE ENGINEERING EDUCATION

N. PRATHEESH*, T. DEVI

Department of Computer Applications, School of Computer Science and Engineering, Bharathiar University, Coimbatore – 641 046, India
*Corresponding Author: pratheesh_n@hotmail.com

Abstract

Computers are necessary and unavoidable part of the modern lifestyle. Software is the key factor to satisfy all the desires and keep up this rock status. Software engineers are the people who develop the software to satisfy the user needs and make their work easier. Software engineers are academically mold up to the industry requirements through the proper software engineering education. Hence software engineering education plays an imperative role in computer education, but it falls short to fabricate the genius software engineers to satisfy the industries need. To overcome these issues researchers proposed number of software engineering learning/teaching methods to egg-on students to reap their depth knowledge in software engineering, albeit these allusions does not utterly conquer this decisive issue since the suggested approaches does not meet with the student's learning style. Learning analytics plays a vital role to improve the students learning activities. This paper describes the software engineering students' requirements through learning analytics and proposes a teaching/learning tool to engage the students learning activities to overcome such issues and inspire them in gathering software engineering knowledge.

Keywords: Software engineering education, Learning analytics, Social learning analytics, Learning style, Learning engagement, Software engineering, Learning/Teaching methods.

1. Introduction

While the software production has had amazing triumph in emergent software that is of mounting degree and intricacy, it has also practiced a stable and noteworthy flow of collapses. The majority of are well-known with open tragedy such as

failed Mars landings, rockets carrying satellites needing to be destroyed shortly after takeoff or unavailable telephone networks and many more "private" tribulations crop up that can be similarly disastrous or at least, problematic and infuriating to those occupied. Exploratory, one of the major forums documenting these failures, the risk forum, supplies an enlightening insight: a considerable section of documented failures can be credited to software engineering process breakdowns [1]. This collapses range from individuals not following an approved procedure such as not performing all required tests, not informing a colleague of a changed module interface, to group coordination problems, such as not using a configuration management system to coordinate mutual tasks, not being able to deliver a subsystem in time, to organisations making strategic mistakes such as choosing to follow the waterfall process model where an incremental approach would be more appropriate, not accounting for the complexity of the software in budget estimate. As a result, it is estimated that billions of dollars are wasted each year due to ineffective processes and subsequent faulty software being delivered [2].

The root cause of the above said problems is the fabrication in the software engineering education. Present software engineering education typically pays poor concentration to students being able to prepare the crises encircle the software engineering. The archetypal software engineering course consists of a series of lectures in which theories and concepts are discussed and make an effort to learn this knowledge into practice. For this a small piece of software engineering project must be developed by the students. Even though in cooperation of these mechanisms is necessary because lectures as a source to feed the basic knowledge of software engineering and the projects are the ways to acquire hands on experience with some of the techniques of software engineering, but this tactic not sufficient to satisfactorily teaches the complete software engineering education [2-4].

2. Issue in Software Engineering Education

Software development companies need the talented software engineers to develop the software. Studies highlighted that the quality of software engineering workplace is a direct function of the quality of software engineering education even though other factors may also play a role. Practical issues in software engineering are a consequence of the insufficient software engineering knowledge [5, 6]. Software engineering courses or training programs emphasise on deeper understanding of the topic and highlights on cognitive learning goals of knowledge and understanding. Higher order cognitive ambitions especially application and analysis, followed by evaluation and synthesis, since these skills are highly used and valued by the industry [4]. Some of the highlighted issues are, substantial numbers of development projects are never completed, many of the completed projects do not meet the user's needs, poor quality and failure occurs because of misunderstanding of requirements, mismatches in system design and implementation, unrealistic expectations and bad project planning [6, 7].

Contemporary software engineering education has been lacking to produce the knowledgeable software engineers to meet the industry's need. University graduates entering into the software engineering professionals are generally unsatisfied with the level of real-world preparedness [8, 9]. Investigations show that software professionals received the knowledge, as part of their graduate

education, which is not sufficient to analysed the industrial software engineering problems. This atmosphere focuses on the significant mismatches between software engineering education and the knowledge that is needed by the industry to perform the required task [1, 10].

In the case of India, software engineering education pattern is divided into modules and each module converses the individual sections and are covered through a series of lectures. Assessments are done at the completion of each module and there may be a final cumulative assessment at the end of the training program [11]. Most of the training programs follow the traditional approaches of lecture based training and performance-based assessments. These approaches are not only less effective, but are also labour intensive on part of training [11].This model of learning might not satisfies the industry's needs and expectations of the clients and changing market conditions and neglects to impart the constantly incorporating newer technologies, techniques, tools, methods and standards [11].

In general, graduates enter to the industry are willingly or unwillingly good in following the syntax, semantics, logic and process. But they are not well in software engineering concepts because, software engineering is offered as just one of the subject in a computer science course. Most of the computer science graduates study software engineering for at most one semester. For some students this is the only opportunity to get familiar with software engineering concepts before starting their career as software engineers [11].

Employers carp about the communication inability of fresh graduates' as they fail to properly communicate with customers and within team members. Most of the new recruits have insufficient experience in working as part of a team, inability to manage their individual work in an efficient and productive manner and do not understand or appreciate organisational structures or business practices [7]. Computer education too often focuses on individual contributions rather than on managed group efforts that depend on defined standards, methodologies and processes [7]. However, such group efforts are the norm in the software industry. Principles and theories of software engineering may not be directly applicable, and students should be motivated to learn. This practice shapeup the student's mentality and improve their approach to solving practical problems [12]. Industry's aggravation is explicable in order that fresh graduates to be prolific in an industrial setting. However, the industries need the resources, hence, these organisations that employ the fresh graduates, provide comprehensive on-the-job training other than their university education and smarten them up with the skills and knowledge, they lack [8].

3. Software Engineering Teaching/Learning Methods

Traditional teaching of software engineering is short of the relationship between theoretical mastery and practice skills development. In addition, software engineering is an important field, especially in the programming language, software development and design tools, software reuse technology and design patterns. However the current software engineering materials and teaching content, knowledge structure and practice have so serious shortcomings, which restrict the effect of the teaching of software engineering [13, 14]. Software engineering researchers proposed numerous teaching/learning methods such as group project, case tools, educational game and web based learning to overcome these challenges.

3.1. Group project

Most of the software engineering teaching model highlight the magnitude of projects and wish for prototypes such as "Small Group Project" and "Large Project Team". Students necessitate to work on projects supported by an external organisation, deliberately employing real-world difficulties during the class project, such as changing requirements while the design is in progress, fit in multiple universities and branch of learning into the project, sustaining a major continuing project in that dissimilar cluster of students work on from semester to semester, and many others. All of these come up to share the identical objective that is to bridge the association between theories and practice. Moreover, students are set in an environment that highly simulates the real software development world and are assigned with jobs such as principle architect, project administrator and configuration manager. The advantages of this method are its intensive simulation of real projects and students are propelled to learn and do more than they would in traditional courses [15-21].

3.2. CASE tools

An extensive assortment of positive hope has been credited to the professional use of CASE tools in software engineering education. This incorporates the thinking of the students use of a CASE tool will smooth the progress of the discipline. This standardise the development process, enhance stability and fullness of the models that are developed, amplify the capability for quality assurance, transform the concentration of assessing away from mere correction of minor errors, make better project planning and management by providing general idea of the development process, cheer on reverse engineering, expand the capability to fabricate high-quality documentation, and bridge the gap between design and implementation [4, 22-25].

3.3. Educational game

An educational game is used in software engineering education to simulates the software engineering process from requirements specification to product delivery. This game provides students with an overall, high-level, practical experience of the software engineering process in a speedy enough method to be used continually in a limited amount of time. Educational game has a number of other traits that contribute to its learning efficiencies. Competition motivates students to play the game and encourages collaborative learning. This atmosphere makes sure that all of the fundamental technicalities of the software engineering process being simulated are able to be seen. Game has a fun, engaging nature and quality that is known to be highly conducive for learning [26, 27].

3.4. Web-based learning

Looking for mainly a complement and not a replacement to traditional education, a set of learning resources particularly designed for the world-wide-web. As a complement to the lectures and printed material, the students have right of entry to the web-based courseware which contained an improved adaptation of the lessons material in electronic form and useful links to pertinent material on the

internet. Furthermore, asynchronous communication amenities were presented via a web-based discussion forum and class management was included in the web-based software engineering learning environment. Information procedures are computerised by software systems and this kind of automation is precious, since such procedures are monotonous, tiring, disagreeable and time consuming. Software engineering is the technological branch anxious with the creation of software systems, which can always be thought of as gears of larger artificial systems. The enriched instructional delivery mode has several advantages over the traditional mode such as students can progress at their own pace and study the instructional material in the order that best look good on their skills or preferences. The learning material is stored online and the course is ''open'' at any time and from anywhere for the students registered in it and the lecturer plays the function as a facilitator and helps out the learning procedure [28-30].

4. Need of Learning Analytics in Software Engineering Education

Learning analytics is one of the fastest mounting fields of technology enhanced learning (TEL); research that has emerged during the last decade. Growth of this field offered in a broadly sequential structure, demonstrating the increasingly rapid pattern of development as new drivers emerge, new fields are appropriated and new tools developed. Tracing the development of learning analytics over time highlights a gradual shift away from a technological focus towards an educational focus, and the introduction of tools, initiatives and methods that are significant in the field today [31]. Learning analytics consists of 'socialised' approaches, which can readily be applied in social settings. These include content analytics – a broad heading for the variety of automated methods that can be used to examine, index and filter online media assets, with the intention of guiding learners through the ocean of potential resources available to them [32]. These analytics take on a social aspect when they draw upon the tags ratings and metadata supplied by learners [33]. A second group of socialised analytics focuses on the learning dispositions that can be used to render visible the mixture of experience, motivation and intelligences that influences responses to learning opportunities. Dispositions analytics can be regarded as socialised learning analytics when the emphasis is on the learner in a social setting, engaged in a mentoring or learning relationship [28].

Social Learning Analytics (SLA) are strongly grounded in learning theories and focus attention on elements of learning that are relevant when learning in a participatory online culture [34]. Approaches to analytics that can be classified in this way include intrinsically the social forms of analytic: social network analytics and discourse analytics [35]. Social learning analytics also includes 'socialised' approaches, which can readily be applied in social settings. These include content analytics – a broad heading for the variety of automated methods that can be used to examine, index and filter online media assets, with the intention of guiding learners through the ocean of potential resources available to them [32, 36].

Software engineering education composed of enormous areas of knowledge and every area contains more information. It is very intricate to students and teachers to find the proper information for their needs and the retrieved information may not motivate them to study the software engineering concepts. Learning analytics is a new thought which helps to measure and improve the

learning. Learning style and learning engagement influence the acquisition of knowledge, since these are considered as influential factors of learning analytics. Previous study proves that learning style plays a vital role in acquire knowledge in software engineering education and suggested to think about students learning style when deliver the knowledge in the software engineering class [37]. This supports to improve and inspire the software engineering education and fabricate knowledgeable software engineer to fulfil the industry needs.

5. Results and Discussions

Data for the study were gathered from self-administrated questionnaire for student's learning engagement with the variable of active participation, emotional engagement, avoidance of text book dominated instruction, reflective thinking, student decision making and problem solving choice, behavioural engagement and relevance. Active participation was measured from the statements like "free to share ideas in class", "discouraged to make judgments on issue within classroom", etc., emotional engagement was measured through "feel happy in the software engineering technology-based class", "feel happy in software engineering traditional-based class", etc., avoidance of text book was identified using "do not use a textbook for software engineering", "teachers lecture from the textbook and we take notes", etc., reflective thinking was measured via "memorizing is the best way to get a good mark", "encouraged to make arguments supporting our own opinions", etc., student decision making and problem solving choice skill was identified through the statements "encouraged to take our own initiative", "learn difficult study concepts more easily when I am able to picture them", etc., behavioural engagement was identified using "prefer independent learning opportunity", "I can illustrate what was learned in software engineering", etc., and relevance was measured via "I can apply the software engineering learning in the real world", "I can work with challenging real world issues in software engineering", etc.

Survey was conducted among Master of Computer Applications (M.C.A.) students to identify the students learning engagement in traditional based teaching software engineering class from equal number of arts and science and engineering and technology streams. Two hundred questionnaires were distributed to M.C.A. students of various institutions in Coimbatore such as Bharathiar University, Sri Ramakrishna Mission Vidyalaya College of Arts and Science, Dr. N.G.P. Institute of Technology and Sri Krishna College of Engineering and Technology. These institutions were selected based on lottery method. Bharathiar University and Sri Ramakrishna Mission Vidyalaya College of Arts and Science are under arts and science stream and offer the courses such as M.Sc. in Mathematics, M.Sc. in Computer Science, M.Sc. in Information Technology, M.C.A., M.B.A. and M.Sc. in Statistics. Dr. N.G.P. Institute of Technology and Sri Krishna College of Engineering and Technology are offering the programme under engineering and technology stream. These institutions offer courses such as M.E. in Computer Science & Engineering, M.E. Embedded Systems, M.E. Engineering Design, M.B.A., M.C.A. and M.E. Power Electronics & Drives. However, this survey considers M.C.A. degree programme, because M.C.A. is one of the important course offered by both the streams and it contains software engineering course in the second year programme. After the screening, one hundred sixty eight questionnaires were fully completed and useable, yielded a response rate of 84%.

SPSS version 17.0 was used to analyse the collected data. Table 1 shows the summary of descriptive statistic for M.C.A. software engineering students' learning engagement.

Table 1. Descriptive statistic for student learning engagement.

Variables	Mean	Median	Mode	SD
Active Participation	27.0	26.0	22	5.2
Emotional Engagement	30.0	29.5	26	5.4
Avoidance of Text book Dominated Instruction	28.9	29.0	30	4.2
Reflective Thinking	30.1	29.0	28	6.0
Student Decision Making and Problem Solving Choice	30.0	29.0	27	6.1
Behavioural Engagement	30.8	30.0	29	5.1
Relevance	28.1	27.5	24	5.4

5.1. Active participation

Figure 1 attests that the students distributed with the mean score of 27, median of 26 and mode of 22 for the variable active participation among traditional based learning software engineering students. The standard deviation for the factor active participation is 5.2. It revealed that 68% of students are clustered closely around the mean and they fall in the range of 21.8-32.2. This value is above the factor average 20 (10×4). It renders that the distribution is skewed positively since the mean score is higher than the median and mode. It confirms that traditional based learning software engineering students are not incited to reveal their thoughts, disagree with views, raise technical issues, and make judgment for the problem. Therefore students are not active in traditional based teaching software engineering classes.

5.2. Emotional engagement

Figure 2 authenticates that the students distributed with the mean score of 30, median of 29.5, mode of 26 and standard deviation of 5.4 for the variable emotional engagement among traditional based learning software engineering students. It exemplifies that the distribution is skewed positively while the mean score is more than the median and mode. The score of the standard deviation expound that 68% of students in traditional based software engineering teaching class-room are assemblage near to the mean and they fall in the range of 21.8-32.2. It points out that most of the students prefer the technology based collaborative learning than the pure traditional based system, as they felt that existing teaching method is tediousness and do not invent much interest in learning process.

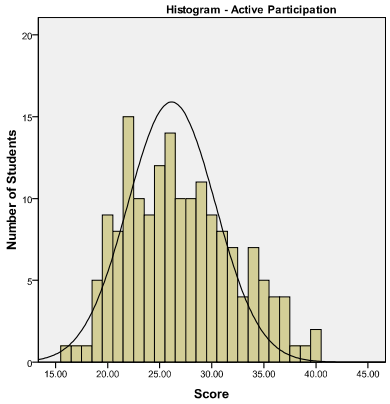

Fig. 1. Distribution of active participation.

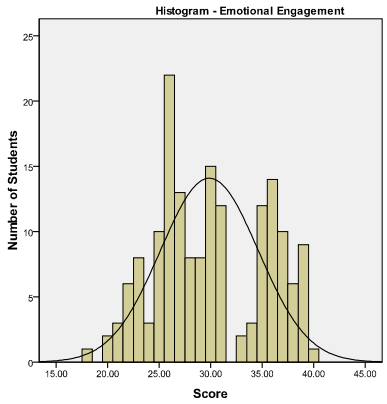

Fig. 2. Distribution of emotional engagement.

5.3. Avoidance of text book dominated instruction

Figure 3 bear outs that the students distributed with the mean score of 28.9, median of 29 and mode of 30 for the variable avoidance of text book dominated instruction among traditional based learning software engineering students. It directs that the distribution is skewed negatively while the mode value is higher than the median and mean score. Further the standard deviation for the factor avoidance of text book instruction is 4.2. It revealed that 68% of students are clustered closely around the mean and they fall in the range of 24.7-33.1. This value is above the factor average 20 (10×4). It rendering that most of the students keep up to date their knowledge and come across the solution for software engineering problems from watching software engineering lecture videos, browsing information from internet, reading magazines, newsletters, books and articles, chatting with colleagues, discussion forum, etc., than the traditional dictation method of teaching.

5.4. Reflective thinking

Figure 4 substantiates that the students distributed with the mean score of 30.1, median of 29 and mode of 28 for the variable reflective thinking among traditional based learning software engineering students. It represents that the distribution is skewed positively since the mean score is exceeding the median and mode. The standard deviation for the factor reflective thinking is 6.0. It revealed that 68% of students are clustered closely around the mean and they fall in the range of 24.1-36.1. This value is above the factor average 20 (10×4). It exposes that a large amount of traditional based learning software engineering students expect encouragement from teachers to counter different opinions and gathering knowledge through discussion forum rather than focus on facts and memorisation. Students prefer to have the discussion in practical issues and the use of technology in software engineering concepts.

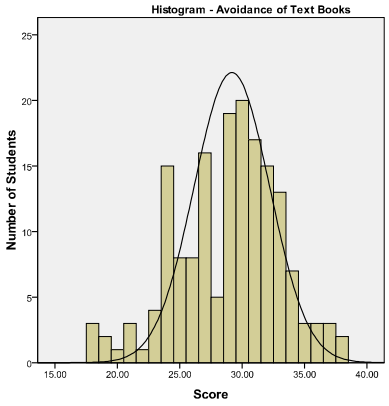

Fig. 3. Distribution of avoidance of text book dominated instruction.

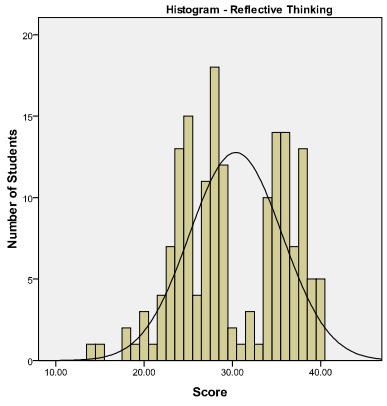

Fig. 4. Distribution of reflective thinking.

5.5. Student decision making and problem solving choice

Figure 5 confirms that the students distributed with the mean score of 30, median of 29 and mode of 27 for the variable student decision making and problem solving choice among traditional based learning software engineering students. It depicts that the distribution is skewed positively though the mean score is higher than the median and mode. Further the standard deviation for the factor student decision making and problem solving choice is 6.1. It revealed that 68% of students are clustered closely around the mean and they fall in the range of 23.9-36.1. This value is also above the factor average 20 (10×4). It discloses that mass number of the students perceive they do not have enough freedom to participate and influence the decision making in the traditional based software engineering classes. Further they perceive they don't have abundant opportunity to make up their own minds about issues related to software engineering concepts, teachers determine the class activities and force them to enrol the activities.

5.6. Behavioral engagement

Figure 6 corroborates that the students distributed with the mean score of 30.8, median of 30 and mode of 29 with the standard deviation of 5.1 for the variable behavioural engagement between the traditional based learning software engineering students. It symbolizes the distribution is skewed positively seeing as the mean score is higher than the median and mode. The score of the standard deviation divulge that 68% of students in traditional based software engineering class-room are congregation around the mean and they fall in the range of 25.7-35.9 for behavioural engagement. It demonstrates that nearly everyone in the traditional based teaching software engineering class tend to have disruptive behaviour such as skipping lectures and getting in trouble in traditional learning method. This makes students to have low involvement in learning effort, persistence, concentration, attention, asking questions and contribution to class discussions and classroom activities.

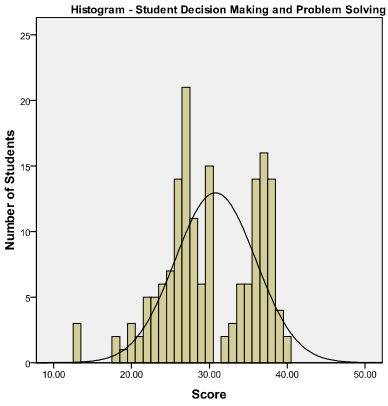

Fig. 5. Distribution of student decision making and problem solving choice.

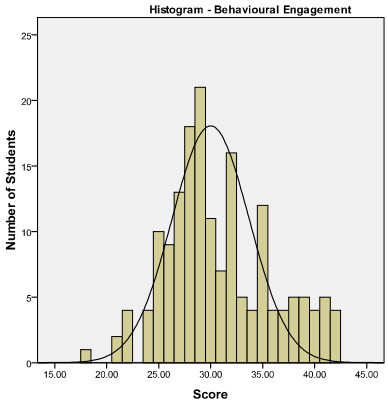

Fig. 6. Distribution of behavioural engagement.

5.7. Relevance

Figure 7 illustrates that the students distributed with the mean score of 28.1, median of 27.5 and mode of 24 for the variable relevance among traditional based learning software engineering students. It makes clear that the distribution is skewed positively whereas the mean score is exceeding the median and mode. Further the standard deviation for the factor relevance is 5.4. It revealed that 68% of students are clustered closely around the mean and they fall in the range of 22.7-33.5. This value is above the factor average 20 (10x4). It provides an evidence that, traditional based learning software engineering students are of the opinion that technology based course make them more marketable in their chosen field, since it coincide with the challenging real world issues compare to traditional-based learning environment.

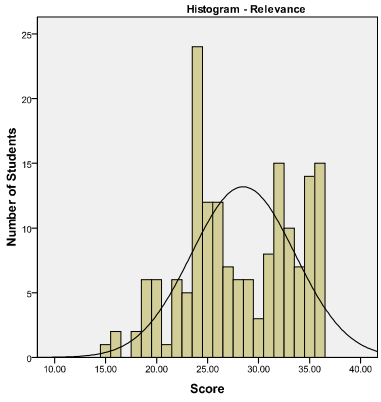

Fig. 7. Distribution of relevance.

6. Conclusion

Software engineering education commonly practices the traditional method teaching. Research findings focus that, students are not active in traditional based teaching software engineering classes. They felt that existing teaching method is tediousness and do not invent much interest in the learning process. Students prefer to watch software engineering lecture videos, browsing information from internet, reading magazines, newsletters, books and articles, chatting with colleagues, discussion forum, etc., to gather their knowledge than the traditional

dictation method of teaching. Students expect the encouragement from teachers to counter different opinions and gathering knowledge through discussion forum rather than focus on facts and memorisation. They prefer to have the discussion in practical issues and the use of technology in software engineering concepts. Students don't have abundant opportunity to make up their own minds about issues related to software engineering concepts, teachers determine the class activities and force them to enrol the activities. They have low involvement in learning effort, persistence, concentration, attention, asking questions and contribution to class discussions and classroom activities. Students have the opinion that technology based course make them more marketable in their chosen field, since it coincide with the challenging real world issues compare to traditional-based learning environment.

Hence it concluded that majority of the traditional based learning software engineering students prefer technology based collaborative learning environment than the pure traditional based teaching method. Students feel that this atmosphere swells their learning engagement and motivates them to study software engineering in depth. This study proposes a simulated web based software engineering teaching tool. This tool has the features such as dynamically detects the learning style of the learner and evokes the learning materials in line with their learning style, discussion forum, update the activities and events to the students and leisure activities using social software concepts. The propose tool also guides the students to select the appropriate learning materials and lecturer to find the student's needs using learning analytics concepts. This mounts the learning engagement of the students and impels their learning activity enjoyably in software engineering. This would triumph over the issues and bring forth the clued-up software engineers to the industry requirements.

References

1. Kitchenham, B.; Budgen, D.; Brereton, P.; and Woodall, P. (2005). An investigation of software engineering curricula. *Journal of Systems and Software*, 74(3), 325-335.
2. Aasheim, C.L.; Li, L.; and Williams, S. (2009). Knowledge and skills requirements for entry level information technology workers: a comparison of industry and academia. *Journal of Information Systems Education*, 20(3), 349-356.
3. Kim, Y.; Hsu, J.; and Stern, M. (2006). An update on the IS/IT skills gap. *Journal of Information Systems Education*, 17(4), 395-402.

4. Lee, C.K.; and Han, H.-J. (2008). Analysis of skills requirement for entry-level programmer/analysis in fortune 500 companies. *Journal of Information Systems Education*, 19(1), 17-27.
5. Beckman, H.; Coulter, N.; Khajenoori, S.; and Mead, N.R. (1997). Collaborations: Closing the industry-academia gap. *IEEE Software*, 14(6), 49-57.
6. Gibbs, W.W. (1994). Software's chronic crisis. *Scientific American*, 271(3), 86- 95.
7. Denning, P.J. (1999). Educating a new engineer. *Communications of the ACM*, 35(12), 83-97.
8. Richard, Conn. (2002). Developing software engineers at the C-130J Software Factory. *IEEE Software,* 19(5), 25-29.
9. Callahan, D.; and Pedigo, B. (2002). Educating experienced IT professionals by addressing industry's needs. *IEEE Software,* 19(5), 57-62.
10. Surakka, S. (2007). What subjects and skills are important for software developers? *Communications of the ACM*, 50(1), 73-78.
11. Kirti, G.; and Vasudeva, V. (2008). Software engineering education in India: issues and challenges. *Proceedings of the 21st Conference on Software Engineering Education in India: Issues and Challenges, IEEE Computer Society, IEEE*, 110-117.
12. Carlo, G.; and Dino, M. (2005). The challenges of software engineering education. *Proceedings* of the *ICSE 05, St. Louis, Missouri, USA. ACM,* 637-638.
13. Song, H.Y.; Li, X.Z.; and Zheng, H.X. (2008). Investigation and practice of cultivating software engineering specialty talents. *Journal of Dalian Nationalities University*, 5(10), 473-476.
14. Wang, Z.-H.; and Yuan, F.-Y. (2006). Study on teaching reform of course software engineering. *Computer Knowledge and Technology*, 23, 219-220.
15. Burnell, L.J.; Priest, J.W.; and Durrett, J.R. (2002). Teaching distributed multi disciplinary software development. *IEEE Software*, 19(5), 86- 93.
16. Dawson, R. (2000). Twenty dirty tricks to train software engineers. *Proceedings of the 22nd International Conference on Software Engineering, ACM*, 209-218.
17. Hayes, J.H. (2002). Energizing software engineering education through real-world projects as experimental studies. *Proceedings of the 15th Conference on Software Engineering Education and Training*, *IEEE*, 192-206.
18. Jeffrey, C.; Letizia, J.; Sandro, M.; and Forrest, S. (2003). Issues in using students in empirical studies in software engineering education. *Proceedings of the Ninth International Software Metrics Symposium (METRICS'03), IEEE*, 239-249.
19. Mayr, H. (1997). Teaching software engineering by means of a virtual enterprise. *Proceedings of the 10th Conference on Software Engineering, IEEE Computer Society, IEEE*, 176-184.
20. Pfleeger, S.L. (1998). *Software engineering, theory and practice.* Prentice-Hall, Inc.
21. Sebern, M.J. (2002). The software development laboratory: incorporating industrial practice in an academic environment. *Proceedings of the 15th Conference on Software Engineering and Trainin, IEEE*, 118-127.

22. Bromell, J; and Preston, J. (1989). Will CASE help me develop more reliable software? *Proceedings of the IEEE Colloquium on The Application of Computer Aided Software Engineering Tools*, *IEEE*, 1-4.
23. McClure, C. (1989). The CASE for structured development. *PC Tech Journal*, 6(8), 51-67.
24. McClure, C. (1989). *CASE in software automation*. Prentice-Hall.
25. Orlikowski, W. (1988). CASE Tools and the IS Workplace: Some findings from empirical research. *Proceedings of the ACM SIGCPR Conference on the Management of Information Systems Personnel, ACM*, 88-97.
26. Bruffee, K.A. (1983). *Collaborative learning: higher education, interdependence, and the authority of knowledge*. John Hopkins University Press.
27. Ferrari, M.; Taylor, R.; and VanLehn, K. (1999). Adapting work simulations for schools. *The Journal of Educational Computing Research*, 21(1), 25-53.
28. Deakin, C.R.; Broadfoot, P.; and Claxton, G. (2004). Developing an effective lifelong learning inventory: the ELLI project. *Assessment in Education: Principles, Policy & Practice*, 11(3), 247-272.
29. McCormack, C.; and Jones, J.D. (1997). *Building a web-based education system*. Wiley, New York.
30. Pratheesh, N.; Devi, T; and Prithvi Vidhya, P. (2012). Web enabled learning tool for software requirements analysis. *International Journal of Software Engineering Research & Practices*, 2(1), 6-11.
31. Rebecca, F. (2012). The state of learning analytics in 2012: a review and future challenges. *Technical Report KMI-12-01, Knowledge Media Institute, The Open University, UK*.
32. Verbert, K.; Drachsler, H.; Manouselis, N.; Wolpers, M.; Vuorikari, R.; and Duval, E. (2011). Dataset-driven research for improving recommender systems for learning. *Proceedings of the LAK11: 1st International Conference on Learning Analytics and Knowledge, ACM*, 44-53.
33. Clow, D.; and Makriyannis, E. (2011). iSpot analysed: participatory learning and reputation. *Proceeding of the 1st International Conference on Learning Analytics and Knowledge, ACM*, 34-43.
34. Rebecca, F.; and Simon, B.S. (2012). Social learning analytics: five approaches. *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge, ACM,* 23-33.
35. De Liddo, A.; Simon, B.S.; Quinto, I.; Bachler, M.; and Cannavacciuolo, L. (2011). Discourse-centric learning analytics. *Proceedings of the LAK11: 1st International Conference on Learning Analytics and Knowledge, ACM*, 23-33.
36. Drachsler, H.; Toine, B.; Riina, Vuorikaric.; Katrien, V.; Erik, D.; Nikos, M.; Guenter, B.; Stephanie, L.; Hermann, S.; Martin, F.; Martin, W. (2010). Issues and considerations regarding sharable data sets for recommender systems in technology enhanced learning. *Procedia Computer*, 1(2), 2849-2858.
37. Pratheesh, N.; and Devi, T. (2013). Influence of learning analytics in software engineering education. *Proceeding of the IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICECCN 2013), IEEE*, 712-716.