International Journal of Scientific and Innovative Research 2014; 2(1): 107-114,
P-ISSN 2347-2189, E- ISSN 2347-4971

DESIGN AND DEVELOPMENT OF AN ALGORITHM FOR ASSESSMENT OF THE LEARNING STYLE OF SOFTWARE ENGINEERING STUDENTS

***ANKITA[1], DR. K P YADAV[2]**
[1]Research Scholar, Department of Computer Science & Engineering,
Sai Nath University, Ranchi, India
[2] Director, MIET, Greater Noida, Uttar Pradesh, India

*Address for correspondence: Ankita, Research Scholar,
Department of Computer Science & Engineering,
Sai Nath University, Ranchi, India

ABSTRACT

Software engineering courses are core elements of the computer science curricula. While the main aim of such courses is to give students practical industry-relevant "software engineering" in the large experience, often such courses fall short of this important target due to lack of industrial experience and support infrastructure. Therefore, it is necessary to design and develop better infrastructure support for teaching or learning such courses and this would benefit instructors and students world over. This aims to create a knowledge and tool infrastructure for the benefit of instructors and students in software engineering courses. Thus, the system embodies a teaching and learning environment that blends collaboration, simulation, knowledge sharing, and industrial software projects. This paper discusses the need for the learning style in software engineering education and proposes an algorithm for dynamically assessing the learning style of the learner and channels the learners to choose appropriate learning materials based on their learning style.

Keywords: *Software Engineering, Software Engineering Education, Learning Style, Learning Theory, Collaborative Learning, Teaching/Leaning Methods*

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 these failures 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 such as a considerable section of documented failures can be credited to software engineering process breakdowns [16]. 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 organizations 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 [1].

The root cause of the above said problems is fabrication in the software engineering education. Present software engineering education typically pays poor concentration to

students being able to preparation 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 cooperation of these mechanisms is necessary because lectures are 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 is not succeeded to satisfactorily teach the complete software engineering education [1, 15, 17].

SOFTWARE ENGINEERING EDUCAION

IT industry is an emerging industry, software development tools and technology bring up to date rapidly. Even now, the majority of the software educating organizations has been coaching graduates comparatively old knowledge systems. It is complicated to become accustomed to the speedy advancement of information technology requirements. However, there is continuously a ferocious loop in computer science graduates, that is, in one hand employers believe that the software talent deficiency, on the other hand a significant number of graduates cannot find a suitable position [16]. Principally the quality of software professionals cannot come across social necessities.

Software engineering education consists of requirements: engineering, software design, validation and verification, others focused on the management of the complexity of products and processes. Frequency of project failure due to problems in software is significantly greater than that due to problems in other engineering disciplines. In too many cases the causes of failure originate in misunderstanding of requirements, mismatches in system design and implementation, unrealistic expectations, and bad project planning [28]. Aligning software engineering education with industry needs is a substantial defy because the relevance and depth of the knowledge that software professionals had received as part of their graduate education and a significant mismatch between software engineering education and industry in terms of the knowledge needed by software engineers to perform the tasks required by industry. Stop up this gap is one of the majority vital errands to be focused in software engineering education. This chore is knotty by a number of open queries; contain "What industrial practices are currently not being taught? How effective are these practices? Which practices should be taught to undergraduates?" [16, 18, 32].

Various tactics are being occupied in an effort to response the above questions. A few higher education institutions perform with group of professionals composed of industry and education legislative body whose target is to assess the sharpness of their software engineering graduate or undergraduate programmes. To efficiently satiate this space, it would be essential, on one hand, to promise that the educational programmes provide the knowledge necessitate for the job profiles recommended by industry and also guarantee that this knowledge is educated in a manner facilitating future professionals to accurately gear the tribulations that they will face throughout their professional career [19, 30].

SOFTWARE ENGINEERING TEACHING/ LEARNING ISSUES

Software engineering education has a number of issues and the underlying ones are the constraints of the academic environment. While relevant process theory can be typically presented in lectures, the opportunities for students to practically and comprehensively experience the presented concepts are limited [18, 30, 42]. Students assume that software engineering is a theoretical subject and it is humdrum and very little use in future [37]. Hence, they are not actively occupied in software engineering learning activities. Traditional based teaching software engineering and knowledge-gaining process turns out to be monotonous and less interactive and instructor enlightens the concepts and the students grasp the facts, which they memorize and reproduce it in the examination [42]. In this method, students have been enforced to passively provide solutions instead of taking actions to cause changes to

the solutions [42]. This method of education never produces the industry relevant software engineering knowledge to the students rather than it increases the gap between the software engineering education and the industry needs.

Software engineering students are generally passive listener throughout the lectures and do not enthusiastically involved in the learning process [18]. Learning scientists and cognitive psychologists has proven that these kinds of learning is not effective and never achieve the goal of study [8, 13, 34]. Students could learn more effectively when they get actively engaged their learning process and they have more awareness and interest in programming languages than software engineering [18].

Most of the industry professionals are not educated in key portions of the software engineering body of knowledge such as requirements, architecture, testing, human factors and project management. They are merely skilled at programming in a few popular languages or at using specific technology products, such as database management and web development tools [23]. Software industry's bigger problem is lack of talented software engineers, when industry employs the fresh graduates, they do not satisfy the industry's needs because they are good in following the syntax, semantics logic and process but they are not well versed with software engineering concepts [19].

Higher educational institutions have difficulties in educating professional software engineers to satisfy the industry's requirements. Software engineers must have the skills such as blended formal knowledge, problem solving, self-learning, professional communication, good judgment, experience, ability to work together and understand the client's needs [23, 40]. But it is not easy to teach all of this within one or two courses. The contemporary software engineering education usually pays little or no attention to students being able to practice issues surrounding the software engineering. The curriculum mostly focuses on the phase of software development process such as requirements, analysis, specification, design and testing. Hence, most of the root problems in software engineering occurred in the professional education [20, 23, 30].

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, design patterns and other fields, but current materials and teaching content, knowledge structure and practice have so serious shortcomings, which restrict the effect of the teaching of software engineering [31, 33]. Software engineering researchers proposed numerous teaching/learning methods such as group project, case tools, educational game and web based learning to overcome these challenges even though still there is a lack in producing endowed software engineers to satisfy the industry's needs using these methods.

A. Group Project

Most of the software engineering teaching model highlighted the magnitude of projects and wished for prototypes such as "Small Group Project" and "Large Project Team". Students necessitate to work on projects supported by an external organization, 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 [5, 8, 12, 13, 21, 28, 29].

B. 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 in use of a CASE tool which will smooth the progress of the disciplined and standardized 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 [2, 11, 22, 23, 25].

C. Educational Game

An educational game which is used in software engineering education to simulate 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, but it also encourages collaborative learning, makes sure that all of the fundamental technicalities of the software engineering process being simulated are able to be seen and it has a fun and engaging nature, and quality that is known to be highly conducive to learning [3, 10].

D. Web-Based Learning

Looking for mainly a complement and not a replacement to traditional education, a set of learning resources is particularly designed for the world-wide-web. As a complement to the lectures and printed material, the students had 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 computerized by software systems and this kind of automation is precious, since such procedures are monotonous, tiring and 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 teacher plays the function of a facilitator and helps out the learning procedure [20, 24, 26].

IMPORTANT TO ASSESS THE LEARNING STYLE

Individual learners perform a most important part in traditional as well as technology enhanced learning. Each learner has individual needs and personality such as various preceding knowledge, cognitive skills, learning styles and motivation. These personality variances influence the learning process and are the cause why some learners find it easy to learn in a specific module, while others discover the same module tough. Prior knowledge is one of the greatest and reliable personal variance forecasters of triumph [14]. While previous knowledge give the impressions to account for further variance in learning than other individual differences, more recently educational researchers have paying attention on features of individual characteristics such as learning styles, their effect on learning and also how they can be integrated in technology improved learning.

Considering learning styles, researches are provoked by educational and psychological theories, which claim that learners have dissimilar ways in which they wish to gain the knowledge. Learners with a strong preference for a specific learning style may have difficulties

in learning if the teaching style does not match with their learning style [9]. From theoretical point of view, inference can be drawn that assimilate learning styles of students in the learning environment formulate learning easier for them and improve their learning effectiveness. On the other hand, learners whose learning styles are not supported by the learning environment may experience problems in the learning process.

Adaptive educational methods address exactly this issue and intention to offering learners with program that fit their individual needs and characteristics such as their learning styles. While supporting adaptively is a big advantage of these systems, they also have strict limitations such as adaptive systems lack integration, supportingonly few functions of web-enhanced education, and the content of courses is not available for reuse [4]. Although educational and psychological theories suggest incorporating individual differences of learners, different learning methods provide only little or, in most cases, no adaptive for them. However, the learning style plays a vital role in software engineering education too. Previous research shows that software engineering class contains different types of learner nearly equal strength and also suggests that if the knowledge delivery satisfies all the learning groups then it improves the knowledge gathering capabilities of the students [27].

The learning style models that exist in literature are majorly classified into five families which are based on some overarching ideas behind the models, attempting to reflect the views of the main theorists of learning styles [6]. The first family relies on the idea that learning styles and preferences are largely constitutionally based including the modalities: visual, auditory and kin aesthetic. The second family deals with the idea that learning styles reflect deep-seated features of the cognitive structure, including patterns of abilities. A third category refers to learning styles as one component of a relatively stable personality type and fourth family learning styles are seen as flexibly stable learning preferences. The last category moves on from learning styles to learning approaches, strategies, orientations and conceptions of learning [6, 7].

First family of learning style has been chosen for this research because it relies on the idea that learning styles and preferences are largely constitution based including the modalities but other families have their own limitations. Visual learners have a preference for seen or observed things, including pictures, diagrams, demonstrations, displays, handouts, films and flip-chart. Auditory learners have a preference for the transfer of information through listening: to the spoken word, of self or others, of sounds and noises. Kin aesthetic learners have a preference for physical experience - touching, feeling, holding, doing, and practical hands-on experiences [6].

AN ALGORITHM TO DYNAMICALLY ASSESS THE LEARNING STYLE

The proposed learning style algorithm uses the structured questionnaire developed by V. Chislett and A. Chapman (2005) under the first family of learning style. A structured questionnaire has been used to identify the type of learners such as visual, auditory and kin aesthetic using thirty questions.

Fig. 1. Algorithm to Find the Learning Style

Fig.1 shows an algorithm to dynamically find the learning style of the learner. Learner has to login the system with their username and password. Once the learner logs in, he/she must answer the thirty questions that help to measure the style. Each question consists of three choices which represent the learning style and is uploaded randomly from the database. Among the answers, the system calculates the statistical mode, finds the learning style of the respective user and shows his/her learning style

Step 1: Upload the learning style questions from database in random order.
Step 2: Assign the three numeric variables for each type of learning style statements.
Step 3: User compulsorily answers learning style questions and creates an array to store the selected value.
Step 4: Find the statistical mode for user selected answers and display their learning style accordingly and guide them to appropriate lessons in line with their learning style.
Step 5: Individual learning styles are ascertained and stored in the database with the respective user name.

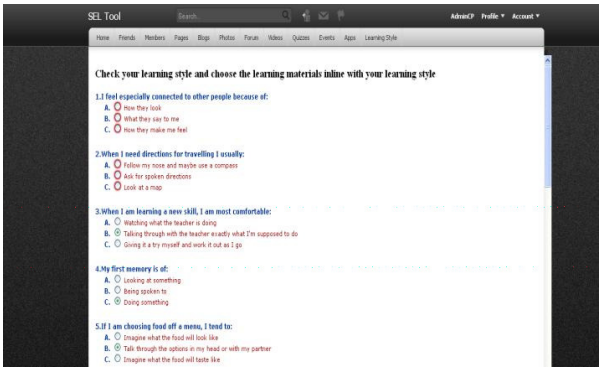
and guides them to the appropriate lessons, in line with their learning style. Even when the student gets low grade in individual online examinations of various modules, the system guides them to make sure about their learning style. Learning style is a psychological factor and it differs from person to person, time to time and environment to environment. Therefore individual learning styles are ascertained and stored in the database with respect to the user and it can be used for future analysis.

DEVELOPMENT OF DYNAMICALLY ASSESS THE SOFTWARE ENGINEERING STUDENTS'S LEARNER LEARNING

This has been originated using the algorithm, which is mentioned in Fig.1 and learning analytics concepts. "Keller ARCS Theories", "Learning Through Dialog

Fig. 2: Screenshot for Measure the Learning Style

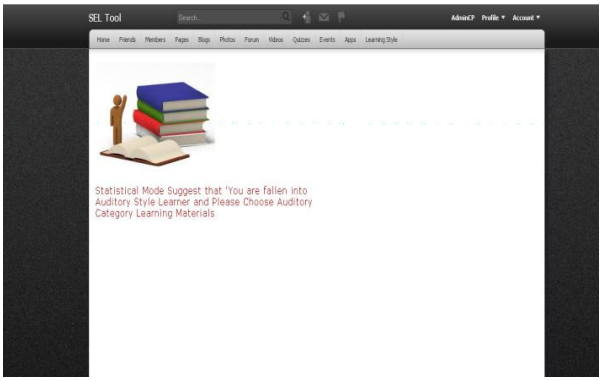
Theory" and "Aptitude–Treatment Interaction



Theory" learning theories are assumed for the formulation of this module.

Fig.2 exhibits dynamically measure of the learning style of the user. This consists of thirty learning style questions and the user compulsorily answers all the questions. Once the user submit the answer for the questions, it finds the learning style of the respective user and it has been stored in database with the user name and displays the learning style and guides the user to choose the right learning materials according to their learning style. Fig.3 shows the sample screenshot for the suggestion of the learning materials. This helps the teacher to provide the learning materials inline with the student's learning style which motivates the students to study software engineering.

Fig 3.Screenshot for Suggestion of Learning Materials



CONCLUSION

Software engineering education has been practiced through several teaching and learning methods but these methods do not produce enough talented software engineers to the industry's expectations since these methods satisfy a specific learning style. Previous research provides evidence that software engineering class contains all types of learner categories such as visual, auditory and kin aesthetic, in nearly equal in strength. Therefore, it is important to check the learning style of the learner and provides the learning materials according to his/her learning style that would motivate the knowledge gathering process. The proposed algorithm discovers the leaner style and leads the student to easily pick the learning materials in line with their learning style. Therefore, students are motivated to learn software engineering concepts enjoyably than the other teaching and learning methods and this model also stimulates collaborative learning environment. This environment encourages the students to study software engineering in depth and become a knowledgeable software engineer.

REFERENCES

1. Aasheim, C.L., Li, L., Williams, S, "Knowledge and skills requirements for entry level information technology workers: a comparison of industry and academia", Journal of Information Systems Education, 20, 2009, pp.349–356.
2. Bromell. J, Preston. J, "Will CASE help me develop more reliable software?", IEEE Colloquium on „The Application of Computer Aided Software Engineering

Tools , Digest No. 24(5), 1989.

3. Bruffee, K.A, "Collaborative Learning: Higher Education, Interdependence, and the Authority of Knowledge", John Hopkins University Press, 1983.
4. Brusilovsky P, "Knowledge Tree: A Distributed Architecture for Adaptive E-Learning", Proceedings of the International Conference on World Wide Web. New York, USA, ACM Press, 2004, pp. 104– 113.
5. Burnell, L.J., Priest, J.W., Durrett, J.R, "Teaching distributed multi disciplinary software development", IEEE Software 19(5), 2002, pp.86– 93.
6. Coffield F, Moseley, D., Hall, E., and Ecclestone K, "Should We Be Using Learning Styles? What Research Has to Say to Practice", Learning and Skills Research Centre / University of Newcastle upon Tyne, London, 2004.
7. Coffield F., Moseley D., Hall E., and Ecclestone K, "Learning Styles and Pedagogy in Post-16 Learning: A Systematic and Critical Review", Learning and Skills Research Centre/University of Newcastle upon Tyne, London, 2004.
8. David H. Jonassen and Barbara L. Grabowski, "Handbook of Individual Differences, Learning and Instruction", Lawrence Erlbaum Associates, New Jersey, 1993.
9. Dawson.R, "Twenty dirty tricks to train software engineers", Proceedings of the 22nd International Conference on Software Engineering, ACM, 2000, pp.209–218.
10. Felder. R.M, and Silverman, L. K, "Learning and Teaching Styles in Engineering Education", Engineering Education, 78 (7), 1988, pp.674–681.
11. Ferrari, M., Taylor, R., VanLehn, K, "Adapting work simulations for schools", The Journal of Educational Computing Research 21 (1), 1999, pp.25–53.
12. Gane. C, "Computer-Aided Software Engineering-The Methodologies, the Product, and the Future", Prentice-Hall, 1990.
13. George L. Geis, "Comparing Instructional Methods: Some Basic Research Problems", The Canadian Journal of Higher Education, 24(2), 1984, pp.91-98.
14. Hayes.J.H, "Energizing software engineering education through real-world projects as experimental studies", Proceedings of the 15th Conference on Software Engineering Education and Training. IEEE, 2002, pp.192–206.
15. Jeffrey Carver, Letizia Jaccheri, Sandro Morasca, Forrest Shull, "Issues in Using Students in Empirical Studies in Software Engineering Education", Proceedings of the Ninth International Software Metrics Symposium (METRICS 03), IEEE, 2003.
16. Jonassen, D. H, and Grabowski, B. L, "Handbook of Individual Differences, Learning, and Instruction", Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1993.
17. Kim, Y., Hsu, J., Stern, M, "An update on the IS/IT skills gap", Journal of Information Systems Education, 17, 2006, pp.395–402.
18. Kirti Garg and Vasudeva Varma, "A Study of the Effectiveness of Case Study approach in Software Engineering Education", Proceeding of the 20th Conference on Software Engineering Education & Training (CSEET'07), IEEE Computer Society, IEEE, 2007.
19. Kirti Garg and Vasudeva Varma, "People Issues Relating to Software Engineering Education and Training in India", Proceeding of the ISEC 08, ACM, 2008, pp.121-127.
20. Kirti Garg and Vasudeva Varma, "Software Engineering Education in India: Issues and Challenges", Proceedings of the 21st Conference on Software Engineering Education and Training, IEEE Computer Society, IEEE, 2008, pp.110-117.
21. Kitchenham, B., Budgen, D., Brereton, P., Woodall, P, "An investigation of software engineering curricula", Journal of Systems and Software, 74, 2005, pp.325–335.
22. Lee, C.K., Han, H.J, "Analysis of skills requirement for entry-level programmer/analysis in fortune 500 companies",Journal of Information Systems Education, 19, 2008, pp.17–27.
23. Lethbridge, T.C, "What knowledge is important to a software professional?", Computer 33, 2000, pp.44–50.
24. Loftus, C., Thomas, L., Zander, C, "Can graduating students design: revisited", Proceedings of the 42nd SIGCSE Technical Symposium on Computer Science Education, 2011, pp. 105–110.
25. Marshall AD, "Developing hypertext courseware on the World Wide Web", Proceedings of World Conference on Educational Multimedia and Hypermedia (ED-MEDIA 95). Graz, Austria, July 1995.
26. Mayr.H, "Teaching software engineering by means of a virtual enterprise", Proceedings of the 10th Conference on Software Engineering. IEEE Computer Society, 1997.
27. McClure.C," The CASE for structured development", PC Tech Journal6(8),1989,pp.51-67.
28. McClure. C, "CASE in Software Automation", Prentice-Hall, 1989.
29. McCormack C, Jones JD, "Building a Web-based education system", New York: Wiley, 1997.
30. Mehdi Jazayeri, The Education of a Software Engineer. Proceedings of the 19th International Conference on Automated Software Engineering (ASE 04), IEEE Computer Society, IEEE, 2004.
31. Orlikowski.W," CASE Tools and the IS Workplace: Some findings from empirical research", Proceedings of the ACM SIGCPR Conference on the Management of Information Systems Personnel, 1988.
32. Pratheesh N, Devi T, Prithvi Vidhya P, "Web Enabled Learning Tool for Software Requirements Analysis", International Journal of Software Engineering Research & Practices, 2(1), 2012, pp.6– 11.

33. Pratheesh N, Devi T, "Influence of Learning Analytics in Software Engineering Education", Proceedings of the IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICECCN 2013), IEEE, 2013, pp.712-716.

34. Pfleeger S. L,"Software Engineering, Theory and Practice", Prentice-Hall, Inc., 1998.

35. Richard E. Mayer and Patricia A. Alexander (Eds.), "Handbook of Research on Learning and Instruction". Routledge, Taylor & Francis, New York, 2011.

36. Sebern, M.J, "The software development laboratory: incorporating industrial practice in an academic environment", Proceedings of the 15th Conference on Software Engineering and Trainin, IEEE, 2002, pp.118–127.

37. Shaw, M., "Software Engineering Education: A Roadmap", Proceedings of the conference on the Future of Software Engineering, ACM, 2000, pp.371-

38. Shi Zhi-cai, Han Yan-duo, Ge Bin, Zhao Zhi-wu, "Reformation and innovation of the personal cultivation pattern of software engineering specialty", Journal of Dalian University, 6(26), 2005, pp.26-28.

39. Song Hai-yu, LI Xi-zuo, Zheng Hai-xu, et all, "Investigation and Practice of Cultivating Software Engineering Specialty Talents", Journal of Dalian Nationalities University,5(10), 2008, pp.473-476.

40. Surakka, S, "What subjects and skills are important for software developers?", Communications of the ACM 50, 2007, pp.73–78.

41. Tockey, S. R., "Recommended Skills and Knowledge for Software Engineers", Proceedings of the International Conference on 12th Software Engineering Education & Training, 1999, pp.168-176.

42. Varma, V., and Garg, K., "Case Studies: The Potential Teaching Instruments for Software Engineering Education", Proceedings of the 5th International Conference on Quality Software, Melbourne, 2005, pp.279-284.

43. Wang Zhi-he, Yuan Fei-yong, "Study on Teaching Reform of Course Software Engineering", Computer knowledge and technology, 23, 2006, pp.219-220.