A Case-Study on Teaching Software Engineering Concepts using a Case-Based Learning Environment

Kirti Garg
IIIT-Hyderabad (India)
kirti@iiit.ac.in

Ashish Sureka
ABB (India)
ashish.sureka@in.abb.com

Vasudeva Varma
IIIT-Hyderabad (India))
vv@iiit.ac.in

***Abstract*—Case-based teaching is a well-known teaching methodology consisting of learning by reading, discussing and analyzing real-life cases and scenarios. We present a Case-Oriented Learning Environment (COSEEd) for teaching Software Engineering concepts to undergraduate and graduate students in a first course of Software Engineering. The novelty of the proposed model lies in being a complete learning environment framework, consisting of pedagogy, broad level learning objectives, assessment, resources and management details, all designed specifically for Software Engineering. Learning and teaching is centered around well-designed SE case studies from authentic software development instances. We describe the COSEEd model, a sample case-study and share out insights as well as lessons learnt while applying the proposed model in practice. We implement and evaluate the proposed model in Software Engineering courses at a University in India focused on the core areas of Information Technology. We use empirical studies on student perception and actual performance to determine the effectiveness of COSEEd towards achieving various learning goals of SE.**

I. Introduction

We present a Case-Oriented Learning Environment (COSEEd) for teaching Software Architecture and Design at University (undergraduate and graduate programs in Computer Science). The proposed learning environment has been implemented in several course offerings at IIIT-Hyderabad[1] (University in India). We conduct empirical studies to validate our framework and experimental results in the classroom demonstrate that the proposed learning environment framework is effective and a suitable teaching approach in Software Engineering (SE) courses. Case studies are known to promote *think forward from first principles*. Thus as an instructional method, use of cases can bring both theory and practice to learning by engaging students in contextualized and real(isitic) learning [1], [2], [3]. Irby et al. suggested that cases are suitable for professional education as they can be used to create a collaborative learning environment for experiential learning through contextualized instruction that could actively involve the learners; could model professional action and thinking, and provide feedback [4]. These qualities resonate with the requirements of Software Engineering education [5]. Case-Studies are versatile tools of learning and can be used in didactic as well as an active learning environment. The utility is considered multifold:

1) Help students gain deeper understanding of concepts by seeing their application in real world complex situations [1], [2], [3], [4]
2) Involve students in active learning
3) Nurture analytic skills by solving of case
4) Encourage discussion [4].
5) Nurture interpersonal or communication skills

There have been a few recent uses of case studies in SE education[6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. These studies used some variation of the typical case study based approach used in management education. Razali and Chitsaz [16] even prescribed a process of writing case studies for teaching SE. However, most reports in the literature are from experiential studies that suggest considerable potential of case studies for teaching SE. Further rigorous research based investigation is required to understand and recommend useful customizations to suit the nature of the SE education. Our work bridges this gap. In context to existing work, the study presented in this paper makes several novel and unique contributions to the field of Software Engineering Education. The novelty of COSEEd lies in being a complete framework i.e. the learning environment that uses well-designed case studies for teaching and learning. COSEEd is a learning environment, not just a pedagogical model. A learning environment consists of learning objectives, pedagogy and assessment in tight integration to help students learn a particular set of knowledge. COSEEd also prescribes ways of using it in different academic settings. COSEEd is flexible, highly reusable and a refined blend of multiple learning theories (case based reasoning/learning being just one of them), though it has case studies as primary teaching instruments. A problem solving model lies at its heart. It uses well-designed case studies, crafted from authentic problems seen in software industry. Cases can be used as a context of problem solving and as an authentic record of the real-world problems, allowing student to experience the same unstructured, multi-perspective, complex nature of real world problems and simulate on-the-job learning. In this paper, we will present the COSEEd model and an empirical study to show its suitability for learning software architecture and design.

II. COSEEd Model

COSEEd is a case study oriented learning environment specifically designed for Software Engineering. In this section,

[1]https://www.iiit.ac.in/

we present the design of our learning environment.

A. Basic Design

Basic design decisions of the learning environment are as follows: The prosed model primarily utilizes a blend of Case Based Reasoning (CBR)[20], [3], and traditional didactic methods as the base pedagogical models. It also consists of attributes from several supporting pedagogical models namely Cognitive Apprenticeship [21], Learning by Doing [22], [23], Learning by Reflection [24]. COSEEd uses enquiry of case studies of a certain nature as the central objects to learning Software Engineering. We call the model as COSEEd i.e. Case Oriented Software Engineering Education model. This is a modified version of the traditional case study teaching method [25], [26]. Here, the case i.e. a hypothetical or real context already exists and student works on a problem (also called as a *challenge*) situated in this case. The task of solving case challenges is equivalent to informed decision-making as required in software engineering. It involves analysis of problems, and devising a solution by application of knowledge and skills, evaluation of alternatives and thus informed decision making. As a side effect of case solving, the domain and discipline knowledge (both conceptual and procedural) are reinforced and problem-solving skills get practiced. We find that case studies can be a useful tool for SE education on many accounts, given as follows.

1) *As a Recording Instrument*: Literature suggests that cases can be used to record knowledge as well as the experience [20], [27]. Cases can be particularly useful for recording the heuristic based, practical knowledge and skills of SE that otherwise remain tacit and stay with the professionals. Cases can adequately record the complex, semi-structure, ill-structured nature [2] of software development and hence can provide the glimpse of the real world SE. Cases can describe the roles and responsibilities of various people in software development. Thus it will help to cover the breadth of the knowledge base, which may be otherwise difficult in a didactic classroom setting. This recorded experience will provide the necessary concretization of the SE concepts that students find abstract otherwise.
2) *As a Context for Problem Solving:* Cases can be the units of problems that require application of SE concepts and skills to solve them systematically. The case-study will form the context and students will understand, reason and act based on it [20]. Student can use the context to understand the way the industry is working and the associated issues and challenges, use the given challenges as the problems to be solved. They will be asked to reason about their solution based on its suitability to address the problem. These problems, since representing the software development activities, may not always be well-structured. This will create opportunities for students to employ techniques suitable for solving semi-structured and ill-structured problems as well [28].
3) *As a Prescribed Way to Think:* The case solving process where the context is not very structured, will prescribe a particular approach to think through structured learning activities so that the thinking process will get nurtured without being very explicit about it.
4) *As Unit of Practice:* A case study will allow students to practice SE and general engineering skills of communication, collaborative work, problem solving, etc. These contexts will provide problems that require systematic application of SE knowledge and skills, are sufficiently complex and can be addressed within time and resource constraints. The case context can be used to demonstrate usage of authentic skills for an authentic problem and case challenges will provide opportunities to practice those skills. Thus each case will act as a unit of practice.
5) *As a Medium to Include Higher Order Cognitive Skills:* Case challenges will be structured such that they require cognition at various levels (understanding to synthesis/create) [29]. Some challenges may be application oriented, while others may require students to analyze the events, decisions taken in a scenario and their result. Other challenges may require students to evaluate two given solutions and choose the better alternative. Designing Software systems are synthesis activities. Hence all cognitive levels can be addressed through well designed SE cases, unlike typical case studies that limit them to Analysis.
6) *As a Unit for Reflection:* The case study description will allow students to reflect on a coherent unit. Each case is based on an authentic software development example. Hence students can reflect on the events to understand the way industry works. Multiple solutions to same case challenges will help students to see the problem solving, heuristic, not so structured nature of SE.
7) *As an Organization Medium:* Use of case studies will allow us to organize the course in units. Each case can be situated in a) individual topics in SE, or b) Essential skills.

COSEEd framework is not bound to a specific curriculum. It is flexible to adapt to curriculum changes with little effort. The flexibility is achieved through configurations of learning objects (case studies, lecture units, assignments) and learning activities according to the learning objectives. We find that Case studies allow us the flexibility. Same case context can be used with different case challenges.The challenges can be modified for every course offering or set of learning objectives. This quality makes COSEEd in line with the evolving, dynamic nature of SE. CoSEEd supports Solving of Ill and Semi-structured problems, as commonly seen in Software Engineering in general, and software architecture and design in particular. Studies show that learning from well-structured problem solving do not always transfer to ill-structured problems, since they relate to different sets of cognitive, meta-cognitive processes, communication processes and even epistemological beliefs[30]. COSEEd requires reflection

of learning [24], [31]. Reflection is important in a Case Based Reasoning environment since a single case is not representative of all problems and hence reflection can help students in seeing patterns, or form structures that will aid in far transfer. The reflection is incorporated in COSEEd through Reflection reports that require a student to reflect on their learning and solve select exercises, leading to far transfer. Instruction and assessment in COSEEd use common techniques or activities [32]. This will reduce the number of activities, leading to lesser administrative overhead and complexity.

B. The Learning Cycle

A complete COSEEd learning cycle, i.e. the pedagogical element of the learning environment, is illustrated in Figure 1. The concepts are taught and learned in a learning cycle. A cycle includes learning activities of lectures, case solving, case discussions and reporting (reflection or detailed case solution). A case is assigned to select teams. Rest of the class is in an Active Listener mode and focuses on understanding the case study, SE concepts, case discussion and submission of a Reflections report. Selected teams are solvers and focus on addressing case challenges through use of SE concepts learned in class or labs. There are certain optional elements like quizzes, reading assignments, and role-plays. Every activity has specific purpose and contributes towards one or more learning objectives. Activities require inputs and many produce artifacts that can be used to assess a students competency of certain skills. The cycle is specific to a topic and will commence with lectures. Lectures will be used to impart theoretical knowledge, mainly conceptual and procedural, from SE and problem solving domains. The lectures can be didactic or contain elements of active learning. Assignments and tutorials may accompany lectures depending on the nature of the topic, desired levels of competencies or cognitive goals. Students engage in case solving. Each case represents a real world software development project, and forms the learning context. Challenges given in the case need to be solved. Some of the possible forms of challenges for a given scenario in software development are:

- Find a promising software architecture and justify it using SE principles
- Given two or more alternative architecture or design, pick the more suitable one
- Analyze and describe why a particular problem occurred in first place (what went wrong)
- Given outline of a technical solution, implement the solution
- Use a particular tool to model the system
- Identify good and bad practices as observed in the design process. Justify using SE concepts.

A case contains on or more such challenges embedded in a context. Each case is assigned to multiple pre-formed teams (solvers). Rest of the students, termed as Case Listeners, are instructed to read the case, understand it, actively participate in case discussion and submit a Reflection report. Reflection is the instrument to make students reflect and record the learning from the learning cycle. It may include a few practice-oriented questions. A case solution evaluation guideline is posted for Case Solvers reference. Solution to these challenges will require different types of efforts. Students will have to analyze the case in order to understand the context and the challenges. A thorough analysis of the case will require the following:

- Understand the project in context using domain and SE knowledge
- Identify the constraints,
- Identify implicit problems and factors impacting the explicit challenges
- Identify various stakeholders, their roles and consider different stakeholder perspectives.

In light of the acquired understanding, students apply SE knowledge and skills to develop solutions for the given challenges. Most of the required knowledge and skills will come from the lectures and self-learning. Contextual application reenforces the learning. Typically students develop many alternate solutions. Further, students may implement the proposed solutions, which is application of SE knowledge and skills. This is followed by a self-evaluation of the work by exercising the evaluation and meta-cognitive skills. Thus students use the problem solving techniques and skills to solve contextualized SE problems i.e. case studies. The Learning Cycle consists of Pre-evaluation sessions. A session is conducted for a team 1-2 days before the presentation and discussion of solutions with class. Primarily these are scaffolding and coaching sessions. Scaffolding [1] is the support that the instructors and support staff provides to the solvers. These sessions contain typical elements of coaching i.e. observing students, offering hints and reminders, providing feedback and encouragement, scaffolding, modeling of problem solving and software design. Students revert to the case solving process after the pre-evaluations. They may be asked to prepare a formal presentation for class. Depending on available time for presentations, students may be asked to present solutions to only a limited number of challenges. The major learning activity is case presentation and discussion in class. Once a specific solution is obtained, students present their problem understanding and solution in form of a presentation, a role-play and a detailed report to the class. The solution will be discussed with peers. Role-play, a small dramatic enactment, can be used to present any aspect of the problem or the general understanding of the topic. Such role-plays serve many purposes. First is to add to the interestingness of the session. Students get the opportunity to see the relationships/similarities of SE concepts and problems with other aspects of life. Second, role-play can be used to emphasize a certain message deemed important by the solvers. Third, it makes the instruction interesting. Discussion is another important aspect of proposed methodology. It will not be exaggeration to say that Discussions are the core of case study method and will be important in our proposed variant as well. Discussions make the instruction learner centric and focus on airing different view, may be opposing

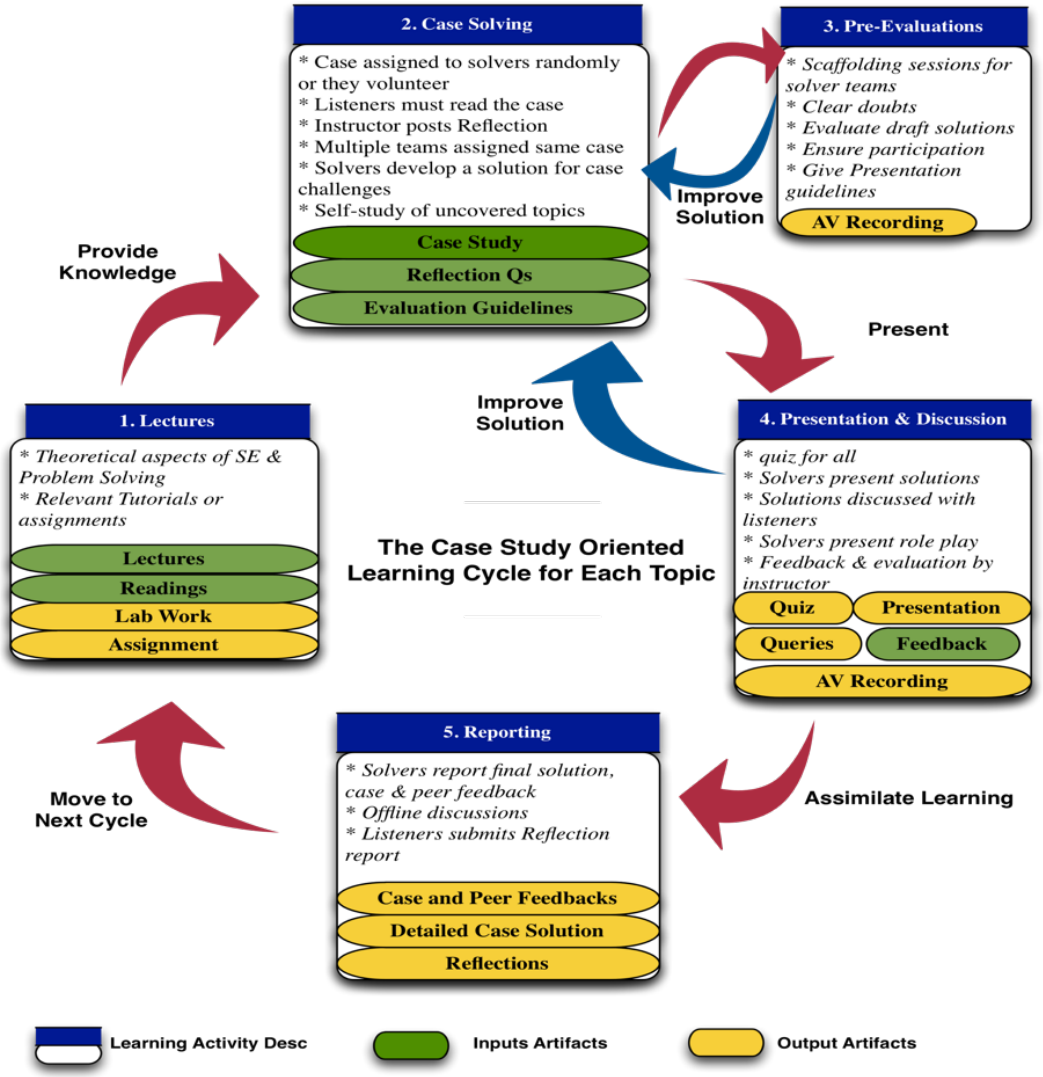

Fig. 1. COSEEd Learning Cycle showing Learning Activities, Input and Output Artifacts

[33]. Discussion provides listeners the opportunity to seek clarifications, raise concerns, highlight different perspectives, present different viewpoints or personal insights, and to know multiple perspectives. Useful discussion are hence entrusting personalized learning as it will allow learners to form informed opinions of their own. The solvers can improve their solutions in light of the discussion. This discussion can continue after class as well. Instructor can use a guided discussion to emphasize on certain learning objectives by highlighting the connections between the case and the objective during the discussion. Active participation by students is a must for this effective discussion [33]. Students must read and analyze the case beforehand for useful discussion and should feel confident to raise their concerns, seek clarifications or present alternates. Once discussions are over, the instructor will provide feedback to the solvers. The last activity in the learning cycle (refer to Figure 1) is submission of detailed case solution report and all other artifacts (the presentations, role-play scripts, recorded AV sessions). It formally records the detailed solution to case challenges, the problem solving process and the overall learning. A good report is the assimilation of learning from case solving, discussion and feedback. It can be an improvement over the solution presented even. It requires students to explicitly state their goals and then validate the techno-managerial solutions if it fulfilled the said goals. The process of writing the detailed report will force the solvers to introspect on their problem solving process. Solvers are asked to list their individual contributions in the case solving process for accountability. This information, along with the peer reviews and pre-evaluation is used in grading. Thus ends a Case Oriented Learning cycle.

C. A Sample Case Study

In this section, we will present summary of one of the case-studies that was designed and used by us to teach the topic of Software architecture. *Twitter Refactoring* case is one of our most popular cases due to its technical appeal. The case can be easily modified to suit very specific learning outcomes or student profiles. Even if students search for solution on the web, no complete solution is found. Students need to understand the Twitter requirements, specially the non-functional ones. They can see the impact of architectural choices of achievement of the non-functional requirements. The relation between computer science (CS) and SE becomes evident. For advanced case, we invite students to simulate the network architecture or build a prototype of the system. We also list case challenges used for the same. We modify the case context every year to reflect the current state of Twitter and also to avoid plagiarism. We also modify our case challenges to meet specific learning objectives of a course offering. We provide a brief description of the case in the following Section. The entire details of the case cannot be covered in this paper due to the page-limit and hence we present some of the key points and ideas.

1) Twitter Case Summary: Twitter started as a side project of some of the employees of Odeon Inc. in 2006. It had immense growth nearly 1000 % growth/year and soon became the micro blogging platform of choice for majority of Internet users. It had 400,000 tweets per quarter in 2007. This grew to 100 million tweets per quarter in 2008. By 2010 there were around 175 million users, 90 million tweets per day and around 500 million searches per day. Twitter was initially built with time to market in mind. So the architecture and technology to build twitter was chosen such that they can build the site in a very short time. Twitter was not designed with this kind of growth in mind. So Twitter had lot of outages especially during popular events such as 2008 Macworld conference keynote address. The main task assigned to our team is to come up with a new architecture that addresses the scalability problems of twitter. We have to identify the most important goals and architectural drivers for twitter and we have to redesign the twitter architecture based on these architectural drivers. We have to identify what COTS components we can use in this architecture and come up with a deployment architecture so that twitter can meet its rapidly growing demand and does not encounter any outages or availability problems.

2) Twitter Case Challenges:

- Challenge 1: What are the architectural drivers, assumptions and major constraints? Give details of at least 5 decisions related to major architectural strategies.
- Challenge 2: Give the architecture in terms of system decomposition (as a diagram and text), structure, connector and component responsibilities.
- Challenge 3: Give examples from your architecture to exhibit various basic design principles.

III. COSEEd Framework Evaluation

A. Experimental Setup

We conduct a post-course survey recording students perception about their competencies and learnings from various topics taught during the course. The course offerings were a first course in software engineering at one of Indias premier research institute. The SE course is offered once every year as part of undergraduate (B.Tech and B.Tech Dual Degree program) and graduate engineering degree program (M.Tech, MS and PhD) in CSE. Undergraduate, graduate students and industry participants (Post-Graduate Student Status Programme (PGSSP[2]) students) attend the SE course. Most graduate students possess undergraduate degree in Computer science, and may have attended Software Engineering course. All students have some project experience, either academic or industrial, but not necessarily team-work. Graduate students may have some work experience or experience of a large project, usually system development done as part of their undergraduate degree program. This system building may not be in accordance with SE principles and practices. All undergraduate students take the SE course after taking basic CS courses like programming, algorithms, database systems, operating systems, and principles of programming languages. This profile is typical of almost all SE students in India.

B. Experimental Results

We asked students to rate their confidence-level in performing certain tasks that directly reflect the learning outcomes of the course in a post-course survey. We received 320 valid responses from 4 course offerings of COSEEd. A Cronbachs Alpha (reliability coefficient) of 0.9 makes our data highly reliable.Less than 2 % of valid responses had missing values, which were replaced using mean value imputation, computed for every item. The results for the topics of Architecture and Design are presented in Figure 2. The evaluation is done for five learning objectives: identification of architecture drivers, selection of architecture style, object-oriented design, deciding architecture strategy and realizing classes. The evaluation consists of five choices: cannot do at all, lots of help required, some help, very little help and without any help (refer to Figure 2). Being ordinal data, we use simple descriptive statistics to analyze the distribution of students perception of their competency. Lower values indicate higher perception of competency and hence the learning. Frequencies and cumulative frequencies show positive perception of individual competencies.

We found that responses were heavily skewed towards confidence scores of 1 and 2. At least 60% students felt that they can accomplish the tasks, across all categories, with little help. Deciding architectural strategy i.e. taking all major decisions regarding the architecture of a given system was one of the most difficult for students. We believe that this lack of confidence is more related to the nature of the task and its complexity. It requires undergraduate students,

[2]https://www.iiit.ac.in/academics/programmes/pgssp

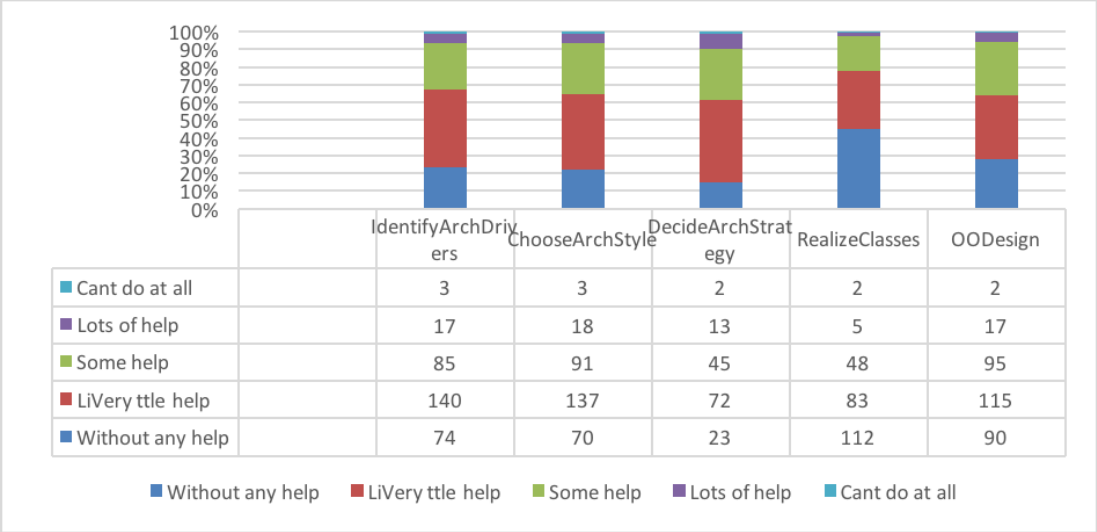

Fig. 2. Evaluation Results based on Course Survey by Students

who have limited exposure to technology, multi-disciplinary knowledge and abstract thinking, to involve at a very high level of cognition resulting into a cohesive architecture. Students were comfortable with the design related tasks. We believe that a contributing factor towards this ease was multiple opportunities to practice, conduct of tutorials and highly modular nature of Object-Oriented design activities. Students conduct Object-Oriented realization one use-case at a time and in guidance of architecture and high level design decisions. Such a guided design was absent when dealing with software architecture. For next analysis, we computed actual scores of student on the respective competencies. The actual scores were calculated by considering student grades of the respective artifacts generated by them during course. The scores were added, scaled to 5 and then binned to make them comparable with confidence perception scores. The perception scores were also recoded such that now a score of 5 represents highest level of confidence (I can do the Task without any help), and 1 represents the lowest confidence. Now we can compare the actual and perception scores.

It is evident that students feel more confident about their competencies than their actual abilities. About 50% students actually did the competency tasks with no or very little help (highest score, or minor problems with their solutions), as compared to about 60% who showed similar confidence in perception scores. The gap is more visible in architectural topics as compared to the design topics. This is inline with our observations of the Perception data.

We further drill to understand if the nature of learning activity impacted actual competencies (refer to Table I). An examination of the cumulative frequencies of actual competency scores tabulated against learning mode (refer to Table I) clearly shows that students that actually solved a case scored higher than the students who were active Listeners and worked on smaller exercises (like assignments).

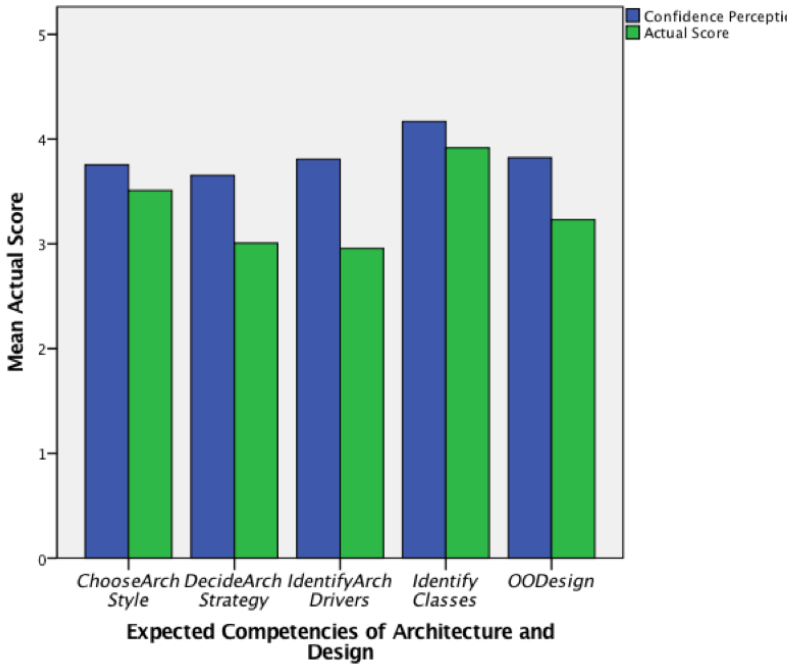

Fig. 3. Confidence and Actual Scores for Different Cognitive Levels

We further examine student perception of their communication skills, another important learning objective of students. COSEEd learning environment is designed to nurture SE relevant aspects of communication skills. We evaluated student communication skills through detailed report and class presentations. Students also self-reported their improvements in communication skills and attributed them to different learning activities. Evidently, case solving and case listening activities provided opportunities to work on these skills. (see Figure 4: Perception Score for Communication Skills). A Wilcoxons

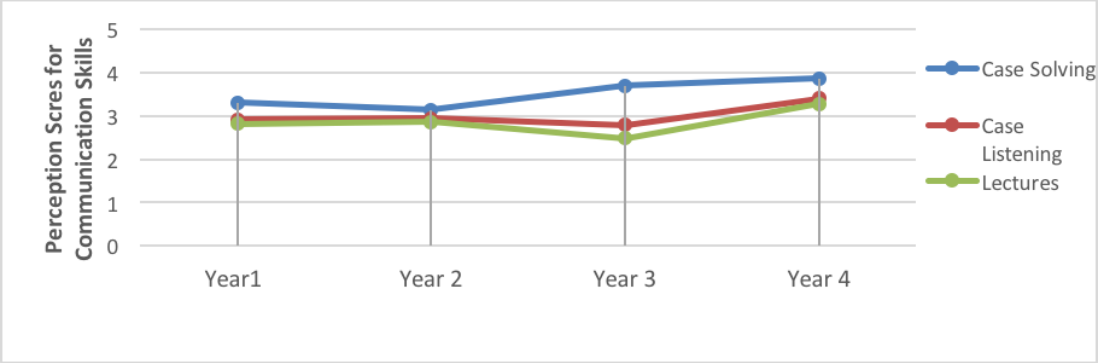

Fig. 4. Evaluation Results (on Communication Skills) based on Course Survey by Students

TABLE I
ACTUAL SCORE (BINNED) - LEARNING MODE CROSS-TABULATION (CS: CASE SOLVING, CL: CASE LISTENING)

	CL	CS	Total
I can't do the task at all	140 (14.2%)	12 (6.7%)	152 (13.1%)
I will need a lot of help	143 (14.5%)	1 (0.6%)	144 (12.4%)
I will need some help	232 (23.6%)	48 (27.0%)	280 (24.1%)
I will need very little help	304 (30.9%)	48 (27.0%)	352 (30.3%)
I will not require any help Count	164 (16.7%)	69 (38.8%)	233 (20.1%)
	983	178	1161

signed rank test to compare the median scores assigned by students suggest that students believe that COSEEd significantly contributed towards their communication skills as compared to lectures ($Z = -8.109$, $p = 0.00$).

C. Other Observations and Suggestions

- Practice using authentic systems: We used a live system (Twitter) as a well-designed teaching case study. The authenticity associated with such systems makes the learning activity realistic, interesting and hence motivating for students.
- Need for Scaffolding: Scaffolding through timely feedback, skill demonstration and discussion were very important for architecture and design. Scaffolding helped to build student skills and confidence. Pre-presentation discussions allowed students to design well reasoned quality solutions and avoid tangents.
- Encouraging Collaborative Work: We believe that collaboration, apart from a prevalent software development practice, is necessary for designing useful systems. We found collaborative work particularly useful since students were very new to the process and architecting. Teamwork helped to bring multiple perspectives and knowledge sets, thus strengthening the process of design. We felt that the quality of designs improved due to collaborative knowledge and design.
- Set expectations realistically: Experience with Software architecture and design clearly suggests that instructors should set their expectations after due consideration of student profile, nature of topic and time available. Software architecture is a difficult topic for 20 something undergraduate students. The solutions to Twitter engineering were reasonable, but not matching to the level of sophistication seen in actual system. Students explored newer technologies, but had limited time to actually experiment and implement their solutions.
- Clearly defined grading criteria: The grading criteria was known to students in advance. They know that the evaluation will include their reasoning, detailing of architecture and design and suitability to the requirements/challenge. Clear definitions made the evaluation task easy for instructors and teaching assistants. It also helped students to decide when to stop with detailing, since the topics are kind of endless, but we had limited time.
- Openness to multiple right solutions: A very important value adopted by COSEEd is the acceptance of all right solutions, in accordance with the very ill-structured nature of SE problems, where multiple right answers exist. Different teams prescribed different solutions, most of which were right (suitable for the purpose, satisfied functional and non-functional requirements, followed some chosen design principle, technically correct). We felt that not looking for one specific solution also made students competency oriented instead of performance oriented. The Listeners also got an insight into nature of the topics and provided pointers for deeper analysis like "Why solution by team A is better than that of team B?" Collins also says "By allowing students to generate their own solution paths, it helps make them conscious, creative members of the culture of problem-solving mathematicians. And, in enculturating through this activity, they acquire some of the culture's tools–a shared vocabulary and the means to discuss, reflect upon, evaluate, and validate community

procedures in a collaborative process" [1].

IV. Conclusion

Case-oriented teaching pedagogy creates an active environment by encouraging learning, discussion, team working and higher order thinking. COSEEd is a learning environment for SE with contextualized case enquiry at its heart. It consists of learning objectives, pedagogy and assessment in tight integration to help students learn SE and engineering skills expected from a software professional. COSEEd is designed in accordance with the, evolving, dynamic, problem solving, heuristic nature of SE as well as best practices from the learning sciences. The evaluation of COSEEd in 4 Software Engineering courses (on Software Architecture and Design module) by 320 student reveals that the prosed model improves student competencies. Students also felt that COSEEd contributed towards their communication skills, with Case solving activities contributing more than case listening activities. Over all, we believe that a case study oriented approach, with well-designed case studies is suitable for teaching and learning of software engineering.

References

[1] J. S. Brown, A. Collins, and P. Duguid, "Situated cognition and the culture of learning," *Educational Researcher*, vol. 18, no. 1, pp. 32–42, 1989.

[2] M. J. Hannafin, "Grounded design of web-enhanced case-based activity," *Educational Technology Research and Development*, vol. 56, pp. 161–179, Apr. 2008.

[3] J. L. Kolodner, "Instructional design: Case-based reasoning," *Retrieved July*, vol. 3, p. 2004, 2003.

[4] D. Irby, "Three exemplary models of case-based teaching," *Academic Medicine*, vol. 69, no. 12, pp. 947–953, 1994.

[5] P. Freeman, "Essential elements of software engineering education Revisited," *IEEE Transactions on Software Engineering*, vol. 13, no. 11, pp. 1143–1148, 1987.

[6] J. Burge and D. Troy, "Rising to the Challenge: Using Business-Oriented Case Studies in Software Engineering Education," *Software Engineering Education and Training, 2006. Proceedings. 19th Conference on*, pp. 43–50, 2006.

[7] S. A. Butler, "A client/server case study for software engineering students," pp. 156–165, 1999.

[8] T. B. Hilburn, M. Towhidnejad, S. Nangia, and S. Li, "A Case Study Project for Software Engineering Education," in *36th Annual Frontiers in Education Conference*, pp. 1–5, 2006.

[9] J. Krone, D. Juedes, and M. Sitharam, "When theory meets practice: Enriching the CS curriculum through industrial case studies," *Proceedings 15th Conference on Software Engineering Education and Training (CSEE&T2002)*, pp. 207–214, 2002.

[10] N. R. Mead and E. Hough, "Security Requirements Engineering for Software Systems: Case Studies in Support of Software Engineering Education," *Software Engineering Education and Training, 2006. Proceedings. 19th Conference on*, pp. 149–158, 2006.

[11] P. K. Raju, C. S. Sankar, G. Halpin, and G. Halpin, "Bringing theory and practice together in engineering classrooms," in *29th Annual Frontiers in Education Conference*, p. 11, 1999.

[12] M. B. Rosson, J. M. Carroll, C. M. Rodi, M. B. Rosson, J. M. Carroll, and C. M. Rodi, *Case studies for teaching usability engineering*, vol. 36. ACM, Mar. 2004.

[13] Y. Jia and Y. Tao, "Teaching Software Design Using a Case Study on Model Transformation," in *6th International Conference on Information Technology: New Generations*, pp. 702–706, 2009.

[14] J. Zhang and J. Li, "Teaching Software Engineering Using Case Study," *Biomedical Engineering and Computer Science (ICBECS), 2010 International Conference on*, pp. 1–4, 2010.

[15] R. Razali and D. A. P. Zainal, "Success Factors for Using Case Method in Teaching and Learning Software Engineering," *International Education Studies*, vol. 6, May 2013.

[16] R. Razali and M. Chitsaz, "Cases development for teaching software engineering," *2nd International Conference on Education Technology and Computer 2010*, vol. 2, pp. V2–121 – V2–125, 2010.

[17] A. Fuller, P. Croll, and L. Di, "A new approach to teaching software risk management with case studies," in *15th Conference on Software Engineering Education and Training*, pp. 215–222, 2002.

[18] *A Case Study Initiative for Software Engineering Education*, 2005.

[19] k. Garg and V. Varma, "A Study of the Effectiveness of Case Study Approach in Software Engineering Education," in *20th Conference on Software Engineering Education & Training,*, pp. 309–316, IEEE, 2007.

[20] A. Aamodt and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," *AI Communications*, vol. 7, no. 1, pp. 39–59, 1994.

[21] A. Collins, J. S. Brown, and A. Holum, "Cognitive apprenticeship: Making thinking visible," *American Educator*, 1991.

[22] R. C. Schank and C. Cleary, "Engines for Education - Roger Carl Schank, Chip Cleary - Google Books," *Lawrence Erlbaum Associates, Inc*, 1995.

[23] R. Bareiss, M. Griss, R. Bareiss, and M. Griss, "A story-centered, learn-by-doing approach to software engineering education," *ACM SIGCSE Bulletin*, vol. 40, pp. 221–225, Mar. 2008.

[24] D. A. Schön, "Educating the reflective practitioner," *San Francisco: Jossey-Bass*, 1987.

[25] R. K. Yin, "Case Study Research: Design and Methods - Robert K. Yin - Google Books," 2009.

[26] W. Tellis, "Introduction to Case Study," *The Qualitative Report*, 1997.

[27] K. Hyeonjin, *Situated learning with cases: Web-enhanced case-based reasoning in teacher education*. PhD thesis, University of Georgia, University of Georgia, Athens, Georgia, 2005.

[28] k. Garg and V. Varma, "An effective learning environment for teaching problem solving in software architecture," in *2nd India Software Engineering Conference*, p. 139, ACM Press, 2009.

[29] D. R. Krathwohl, "A Revision of Bloom's Taxonomy: An Overview," *Theory into practice*, vol. 41, no. 4, pp. 212–218, 2002.

[30] D. H. Jonassen, "Instructional design models for well-structured and III-structured problem-solving learning outcomes," *Educational Technology Research and Development*, vol. 45, pp. 65–94, Mar. 1997.

[31] J. Armarego, "Learning from Reflection: Practitioners as Adult Learners," in *20th Conference on Software Engineering Education & Training,*, pp. 55–63, IEEE, 2007.

[32] J. D. Bransford, A. L. Brown, and R. R. Cocking, *How People Learn: Brain, Mind, Experience, and School.* . National Academy Press Washington, DC, 2000.

[33] B. P. Shapiro, "Hints for Case Teaching," *Harvard Business Publishing*, 2005.