A Novel Interdepartmental Approach to Teach Cross-Functional Collaboration in Software Engineering

Dr. Lynn Roy Thackeray, Utah Valley University

I hold a doctorate degree from Northeastern University in technical curriculum development, teaching and leadership. The focus of my research was on leadership, the learning sciences in the Science, Technology, Engineering and Math (STEM) fields. The title of my dissertation is Women in Computer Science_ Phenomenological Analysis that explores common factors that contribute to women's selection and persistence in Computer Science as an academic major.

My professional background includes twenty-five years of progressively responsible positions in software and systems development, information technology management, and technical leadership. I am a proven technical leader with verifiable results. I have built and led several distributed and off shore technical organizations. My academic experience includes over ten years teaching both in-class and on-line university level computer science courses, curriculum development and course mentorship. I am currently the C# faculty course mentor.

As an educator, a priority for me is to provide a supportive learning environment that promotes student persistence and success. I am safe zoned trained, and I have deep mentoring experience with first year and at-risk students.

Dr. Susan L. Thackeray, Utah Valley University

Dr. Susan L. Thackeray is an Associate Professor and Department Chair at the Scott M Smith College of Engineering and Technology at Utah Valley University. She has over twenty-five years of demonstrated administrative leadership in industry and education that includes international and domestic higher education instructional design, distance learning development, usability testing, workforce development, and team organization/training. Dr. Thackeray is noted for her expertise in career pathways to align with the workforce and has received multiple awards for STEM education innovation. Susan holds a Bachelor of Science in Digital Media from Utah Valley University, a Master of Education in Instructional Technology from Utah State University, a Doctor of Education from Northeastern University, Boston, and a fellowship with Stanford University. Dr. Thackeray has a research focus on credit for prior learning, competency-based education, and underserved populations in science, technology, engineering, and math (STEM).

A Novel Interdepartmental Approach to Teaching Cross-Functional Collaboration in Software Engineering

Authors

Lynn Roy Thackeray, Computer Science Department, Utah Valley University

Susan L. Thackeray, Technology Management Department, Utah Valley University

Abstract

In recent years, the development of software products has become increasingly complex and involves a variety of professionals from different disciplines. Software engineers need to be able to communicate and collaborate across teams, departments, and organizations. Although interdepartmental course collaborations are not a new pedagogical approach, linking concepts from different subject areas creates a holistic learning experience that is often lacking in software engineering courses and is needed to effectively mirror industry software development. The collaborative approach to course delivery has the advantage of allowing software engineers to work together with less technical project managers to gain a broader understanding of the software industry. This experiential paper will describe two approaches implemented in technology management and software engineering courses: a novel interdepartmental active learning environment for undergraduate and graduate students and a discipline-specific application of an Agile Scrum project framework. The undergraduate course Introduction to Technology Management is a three-hour per week project-based class with the goal of introducing students to the challenges and rewards of managing complex technical projects with budget and time constraints. The graduate course Software Engineering Leadership is a three- hour per week project-based class designed for computer science graduate students to identify important roles and success in software project deliverables. The primary goals of both courses are to provide technical students with engaging activities related to developing skills like teamwork, communication, and following a development framework that involves both synchronous and asynchronous communication and collaboration with non-technical teams in a distributed environment.

The traditional classroom instructional approach to teaching management and software engineering typically includes lectures, discussions, and group activities. For this study, the interdepartmental collaboration experimentation identified strategies essential for the novel coordination of efforts to align course content between two skilled disciplines. The research was designed with the intention that the course should prepare, motivate, and engage students in collaborative project outcomes that focus on project frameworks, documentation, and intentional outcomes. Course design commenced with a volunteer interdepartmental partnership between two professors within the College of Engineering and Technology and was delivered in one semester during separate times and class locations. Undergraduate and graduate students were compelled to interact for project success. The advantage of practical experience through collaboration provides students with insights into interpersonal relationships. This study focuses on the effectiveness of interdisciplinary teaching on engineering student project outcomes.

Key Words: Software Engineering, Agile Software Development, User documentation, Active Learning, Real-world project, Technical Communication.

Introduction

"Complexity kills," Microsoft executive Ray Ozzie famously wrote in a 2005 internal memo [1]. "It sucks the life out of developers; it makes products difficult to plan, build, and test; it introduces security challenges; and it causes user and administrator frustration." If Ozzie thought things were complicated back then, one might wonder what he would make of the complexity software developers face today with software users that expect flexibility from software in many the areas of features, connectivity options, high performance, multiple platforms, including the Internet of Things (IoT), context-aware mobile apps, web and service-based deployments and cloud-based systems.

Moreover, the current dynamic business environment requires organizations to develop and evolve software systems at Internet speed. As a consequence of these trends, software development organizations have embraced the agile and distributed development approaches to software development [2]. This makes the ability to communicate and collaborate effectively an essential ingredient for successful software engineering teams. Cross-functional collaboration involves teams from different groups, departments, or even different organizations that work together to achieve a common goal with the trend toward a distributed organization, with product teams working remote, often in different time zones and countries, these collaboration and communication skills have been elevated for the nice to have to the essential category.

Additionally, the ability to collaborate effectively is essential for successful software engineering teams. Cross-functional collaboration involves teams from different departments, functions, and organizations working together to achieve a common goal. It requires teams to have a strong understanding of each other's roles and responsibilities in order to work together effectively. This understanding can be built through an interdepartmental approach to teaching cross-functional collaboration in software engineering. An interdepartmental approach to teaching cross-functional collaboration in software engineering involves teams from different departments and functions coming together to learn and practice effective collaboration. This approach can be beneficial for teams because it allows them to learn from each other and build a better understanding of the roles and responsibilities of each team member. It also allows teams to learn how to effectively communicate with each other and build a shared understanding of the project's goals and objectives.

The problem this paper attempts to address is how to give students practical classroom experience working and communicating with team members who are separated not only by time and distance. In other words, a collaborative distributed work environment.

The approach described in this paper outlines an interdepartmental approach to teaching students to practical cross functional collaboration and communication in

simulated distributed organization involving teams of students from different college departments who come together to learn and practice effective collaboration. This approach can be beneficial for teams because it allows them to learn from each other and build a better understanding of the roles and responsibilities of each team member. It also allows teams to learn how to effectively communicate with each other and build a shared understanding of the project's goals and objectives.

Agile Development

Agile software development is an umbrella team for frameworks of Scrum, Extreme Programming and Feature-Driven development. Agile focuses on small, multidisciplinary (read: diverse) groups with the skills and backgrounds to design, build, test, and deliver products. The Agile manifesto for software development emphasizes direct and open communication and values individuals and interactions over processes and tools and responding to change over following a prescribed plan. A key feature of Agile is that communication is about reducing the steps required to get the information across.

Agile development relies on simplicity, flexibility, and constant iteration [3]. Because Agile teams learn as they go, it’s difficult to anticipate exactly where a project will be and what the communication needs will be on any given day. Which makes traditional documentation-heavy communication cumbersome and counterproductive. In Agile, even the traditional weekly software development department meeting is replaced by short, daily stand-up meetings. This type of communication is highly interactive. Open and direct communication is key to success in an Agile development environment. However, promoting a straightforward communication model has received challenges with the advent of distributed software development organizational models [4].

Distributed Development

Distributed software development refers to planning, designing, building, testing, and managing software with decentralized teams located across different physical workspaces, which often span not only time zones, but countries and even continents [5]. Like Agile, distributed software development is team based. However, the distributed development approaches differ significantly from Agile in their key tenets. For example, while agile methods mainly rely on informal synchronous communication processes, a distributed software development organization relies on more formal mechanisms. This is understandable as working in decentralized locations requires teams to collaborate in asynchronous fashion which is time and location agnostic.

Neither Agile development framework nor distributed development environments are new to industry. Both have been thoroughly vetted [6] [7], and there are several workable solutions. All require increased communication and team working skills. These skills are no longer "nice to have" soft skills. Most organizations now consider communication, both verbal and written, and team working skills a requirement [8]. Recent college graduates can expect employment

interviews that will include probes on collaboration and communication skill sets that go beyond small classroom group efforts.

Curriculum Challenges

While this demand is apparent, most academia still operates within the constraints of the functional silo of the classroom. Project design and development are often products of a formalized course assignment. The constraints are time based (Semester) and often scoped to the members of the class. Functional specifications gathering, operations management, logistics cross department (Marketing, Sales, Customer Support) if not ignored are often simulated.
These functional silos are influenced by a combination of departmental structure and performance measures (student credit hours), which in turn drive faculty lines. Furthermore, separate programs prohibit students from understanding critical elements of communication in favor of in-depth knowledge in one area, which in a Computer Science or Software Engineering department are using the technical aspects of designing and developing the product.

These shortcomings are not unknown to university instructors and curriculum designers. The main complaint is that there are not enough resources to permit a detailed investigation into distributed and cross functional development in sufficient detail. Computer Science and Software Engineering departments are made up of students who are focused on the technical aspects of software project creation and deployment.

Faculty of these departments likewise have focused their careers on teaching these skillsets and are uncomfortable about teaching in areas beyond their expertise. Classroom materials in these technical areas generally are not integration oriented. The result is a strong impetus for curriculum to remain functionally focused, even as industry is looking for individuals with broad communication and collaborative skillsets.

Literature Review

Recent research has focused on the effectiveness of an interdepartmental approach to teaching cross-functional collaboration in software engineering. In a study conducted by Montero et al. [9], the authors assessed the impact of an interdepartmental approach on teams' collaboration and software development performance. The study found that teams who used an interdepartmental approach to teaching cross-functional collaboration had higher levels of collaboration and a better understanding of the roles and responsibilities of each team member. The study also found that teams who used an interdepartmental approach had higher levels of software development performance, as measured by code quality and bug reports.

In another study, Dutta et al. [10] evaluated the effectiveness of an interdepartmental approach to teaching cross-functional collaboration in software engineering. The study found that teams who used an interdepartmental approach had higher levels of collaboration, a better understanding of the roles and responsibilities of each team member, and improved software development performance as measured by code quality. The study also found that teams who used an interdepartmental approach had higher levels of task completion and fewer bugs reported.

A third study, by Gudiseva et al. [11], assessed the effectiveness of an interdepartmental approach to teaching cross-functional collaboration in software engineering. The study found

that teams who used an interdepartmental approach had higher levels of collaboration and a better understanding of the roles and responsibilities of each team member. The study also found that teams who used an interdepartmental approach had higher levels of software development performance, as measured by code quality and bug reports.

Discussion

The findings of the research discussed in this literature review suggest that an interdepartmental approach to teaching cross-functional collaboration in software engineering can be beneficial for teams. Teams who use an interdepartmental approach have higher levels of collaboration and a better understanding of the roles and responsibilities of each team member. This leads to improved software development performance, as measured by code quality and fewer reported bugs. The studies also suggest that an interdepartmental approach can help teams learn how to effectively communicate with each other and build a shared understanding of the project's goals and objectives. This can be beneficial for teams as it helps them to work together more effectively and efficiently.

Motivations

With this deficit gap in mind, the authors have taken an interdepartmental cross-functional approach teach cross functional communication and collaboration to Software Engineering, with the focus being on communication with teams that are not within the software development collective, and importantly are part of a distributed (not on site) organization. While cross- functional collaboration and communication between departments is not new to the teaching of software engineering. The approach outlined in this paper is unique in that the focus is on distributed team-based learning and communication approach. This approach developed by Michaelsen and Sweet [12] and is based on the idea that teams of learners working together to solve complex problems.

On a personal level, the motivation of this course was based on the observation that university students are often in a degree silo. As mentioned above, Computer Science and Computer Engineering critique is focused on the "how" of software development and management. Students do not always have professional experience. Additionally, students are often nervous about course group work. Technical projects, which require not only teamwork, but promote innovation require a diverse team of subject matter experts. A diverse team which scope extends beyond the average Computer Science classroom. A pluralist system of project development allows technical teams to leverage the strength of expertise to collaborate effectively.

As this paper documents, such a pluralistic collaboration was successfully demonstrated at UVU between a Computer Science graduate course and a Technology Management Introduction course. The distributed simulation comes for the collaboration of two departments with two different courses that meet in separate locations and at different times. The schedule of the two courses (time) and their physical locations (different buildings) along with the requirements that each class is responsible for the design, creation and implementation of a working product simulates a typical industry distributed project devilment environment. It should be noted that it

was required that for a complete and finished product, both groups would be integrated into project teams with direct and constant communication being a requirement.

The Course

Two groups from two very different classes were involved with the project. The UVU's Master of Computer Science, Software Engineering Leadership course (CS6300) was pared with groups from the undergraduate course, Introduction to Technology Management (TECH 3000). Which is within the department of Technology Management department. This pairing was meant to give group diversity. The Tech Management focus is on general management of projects, such as manufacturing, food services, and construction. Currently, software project management is not an area of focus for the Tech Management program. Most students have had little or no contact with technology groups including Software Engineering or Computer Science. The Software Engineering (CS6300) course is focused exclusively on Software Development management, with current students coming from coming from computer science backgrounds.

Integration of the two groups was accomplished by utilizing the Scrum project management framework for the project development. This iterative teams-based approach offered several group integration points, which will be discussed below.

The Course Project

The project was designed to mirror as closely as possible an industry type project. The project scenario used is that students work for a fictitious company, Pegasus Inc, which designs, builds, and manufactures lawn care equipment. Pegasus is a medium sized company of approximately 5000 employees. They have a dispersed organization, with corporate headquarters, which includes Marketing, Sales, and Project Management departments, located in Westport, Connecticut. Their manufacturing facilities are in Athens, Georgia, and the Engineering division, which includes all software development located in Orem, Utah. Students in the Technology Management (TECH 3000) class would make up the Project Management department, located in Connecticut, while students from the Software Engineering Leadership course (CS6300) would serve as the software development department located in Utah. In addition to providing oversite, the two course instructors would serve as the project stakeholders.

The scenario continues: Recently, the Software Development department has adopted the Agile process of Scrum. However, other departments, such as Engineering and Manufacturing continue to work in a Waterfall mode. Because of this, all projects, Scrum and otherwise are managed by the Project Management department. Project Managers are expected to work closely with all teams, including Software Development. Project Managers are expected to be the main contact point with project stakeholders (Marketing and Corporate Departments). Project Managers will also serve as the liaison between stakeholders and the Software Development department. In the Agile process, a Project Manager can be considered the Project Owner.

Proposed Project

Pegasus has a successful line of lawnmowers, both electric and gas powered. Currently, marketing research is exploring the possibility of expanding their lawnmower line with the addition of an electric robotic lawnmower. It is envisioned that the lawnmower would be similar to the automated vacuum cleaners that have become popular recently. The difference is that robotic lawnmowers will be designed for outdoor instead of indoor use, will learn the layout of the lawn instead of the layout of the floors of a home, and will cut the grass instead of vacuuming floors.

Before the company invests millions in the design and manufacturing of a new product, Marketing would like to have a software simulation created that would demonstrate the abilities of the proposed robotic lawnmower. Marketing and Sales would use this simulation as a sales tool to test the market and help determine what level such a product would be accepted. It is also desired by corporate that simulation would not only serve as a sales tool, but as a software prototype that could possibly be used in the production of the product, which would require not only a working prototype, but extensive documentation.

Project Requirements

Marketing Research along with corporate planners have come up with some initial requirements for the simulation system.

- The System must be able to read in a file that will describe the lawn to be cut, and then track the progress of the lawnmower as it moves around the lawn.
- The lawnmower needs to avoid obstacles, and to stay in-bounds (on the lawn) and cuts only the grass.
- The lawnmower needs to keep track of grass that has been cut so as not to re-do any areas. The lawnmower will need to sense when the entire lawn has been cut and turn itself off.
- Other requirements are being researched now, and several are under discussion. But Marketing feels that input from Project Management and Software Development should be solicited before final decisions are made.

Project Schedule & Deliverables

Marketing believes this is an 8-week window of opportunity to complete the project, and Corporate has placed a high priority on this effort. This means that project teams will need to be formed immediately. Available Project Managers along with Development teams will be identified immediately (start of semester), with an official project kick off within a week. It is expected that a workable design with as much functionality as possible be ready to demonstrate to Corporate by week five. This demonstration will serve as a go, no-go decision point for the Simulation project (mid semester). If given the green light, a working simulator needs to be delivered to Marking no later than week fifteen (Semester end). During this week, the Software teams will be performing demos and training for the Marketing and Sales departments.

These milestones and requirements are at a high level. Corporate and Marketing (course instructors) will be receptive to detailed modifications, but because of business considerations, the due date is non-negotiable. A project post-mortem (final course review) review will also be held by the stakeholders, and if successful, this project could be moved into phase II for further development.

Project Structure

This was a semester long project with the requirement that the Agile Management of scrum would be used. Scrum teams consisting of 3 to 5 members would be made up of groups from both courses. These teams are meant to be cross-functional, meaning the members have all the skills necessary to create value for each development iteration (Sprint). They are also self- managing, meaning they internally decide on team member roles and tasks, and scheduling.
Project Managers would be selected from the Management course (TECH 3000), and software developers would be made up of students from the Software Engineering (CS6300) class.

For each team, a Scrum Master (SM) would be identified and chosen by Scrum Teams. A lead Project Manager (PM) would also be identified. These positions would be rotating each week (sprint) so to give all group members an opportunity. The project schedule was set up to allow 10-week long sprints.

Collaboration & Communication Requirements

For each one-week sprint, students would be required to:

- Plan and hold an initial planning meeting to identify, prioritize and assign development tasks for the upcoming sprint.
- Schedule and hold at least three daily "stand-up" meetings in which all team members would report on their progress and identify any issues.
- Plan and hold an end of sprint review, where new functionality would be demonstrated for the stakeholders and other sprint groups (both classes attending).
- A mid-sprint update report to the stakeholders, to be completed jointly by the SM and PM.
- An end-sprint retrospective report to stakeholders to be created by the Scrum teams and PM department.

The Scrum Master and Project Manager are required to generate weekly reports on the above meetings.

Project Deliverables:

- A working Final project: Approximately 150-200 lines of code per developer
- A Functional Specifications document.
- A Complete product design document using Unified Modeling Language (UML)
- Technical Documentation (how to set up and run simulation)

- An end of project (semester) Product demonstration to stakeholders

All documentation and product demonstrations were to be completed jointly as a collaboration between CS6300 and TECH 3000 students.

Because of the distributed nature of the two groups (both courses met in different buildings and at different times), Scrum teams were responsible to formulate and implement a plan on how and what type of technology they would use to collaborate and communicate.

Method of Instruction and Evaluation

The scrum requirements listed above necessitated close communication and collaboration between both TECH 3000 and CS6300 students. To support and guide these efforts, an active learning framework combined with a scaffolding teaching strategy was employed. Instructors delivered lessons, guidance, and instructions in distinct segments, providing less and less support as students began to master project and course concepts.

Observations and Evaluations

The objective of this paper was to describe the efforts implement an interdepartmental effort to teach cross-functional Collaboration. It was not the author's intent to produce research or findings. With that said, it was observed that students began to recognize their own subject matter expertise and value to the deliverable effort. Students evolved their understanding to know that there is no one correct logic, or alternatively, that there is more than one correct logic (pluralism, unity) where multiple groups share power. Each group played a critical role in the process of the product delivery.

As mentioned above, the role of Product Manager (PM) with overall project management responsibility was assigned to the undergraduate TECH 3000 students. The technical development effort which included the architecture, design and development of the project was the responsibility of the CS6300 class which was made up of software engineers. By design, the TECH 3000 students were exposed to the details of these efforts, which was meant to given them a greater understanding of the complexities of software development. At the same time, the CS6300 students having to work closely with a largely non-technical group found that they needed to craft their communication to have less technical jargon and to be more business professional.

Students entered the class with various levels of experience with Agile software development practices, and in particular, Scrum. While the Agile process of Scrum was new to most Technology Management undergraduate students, many of the Software Engineering graduate students were familiar with Scrum or similar Agile processes.

Implementing a project management framework such as Scrum methodology in an educational context does put a high demand on instructors. However, the experiences of the participating instructors in this cross-Functional Collaboration reveal that a classroom climate in which students are required to work together in a consistent manner on a well-defined project is both beneficial for

the implementation of a real-world Agile process. The simulated distributed environment also provided opportunities for students from different disciplines to communicate and work together.

With clearly defined project deliverables and milestones. Students collaborated and carefully prepared Scrum ceremonies (stand-up, review, retrospective), and set up the required artifacts (burn down charts, product backlog and end of sprint reports.).

The implementation of Scrum methodology did initially increase the complexity of the learning environment, as for a Scrum team to be effective, it must do more than apply the practices superficially. An understanding of the principles of systems and human interaction that provide the foundation of Scrum is necessary and needs to be part of the course learning objectives.

Project Evaluation

The project evaluation comprised three sections:

Student collaboration and communication.

- Daily Scrum stand up meetings
- Team management (Scrum Master)
- Project progress and reporting (Project Manager)
- Development progress (Scrum Team)
- Weekly status presentation to stake holders (Scrum Team, Project Manager)

Project Deliverable.

- Running, workable solutions with required documentation (see Project Deliverables section)

Final Presentation.

- Both classes were required to collaboration in presenting a formal final project demonstration and project recap to state holders (Instructors).

The students did meet the assignment requirements. The project submission was a multi-media solution that included a robust graphical user interface (GUI), robotic lawn mower animation, sound, a user dashboard that includes metrics on run time, gasoline consumption, percentages of lawn cut, etc. Additionally, there was file upload and save option for different lawn scenarios.

The final project presentation was a collaborative effort between both classes. This was a formal scripted presentation where students demonstrated their project to stakeholders (course instructors).

Conclusions and Next Steps

This literature review has explored the current research on the use of an interdepartmental approach to teaching cross-functional collaboration in software engineering. The findings of the research suggest that an interdepartmental approach can be beneficial for teams, as it leads to higher levels of collaboration, a better understanding of each other's roles and responsibilities, and improved software development performance. Additionally, an effort to effectively

communicate was observed. In the opinion of the authors, this process fostered an environment that encouraged teams to discover how to effectively communicate with each other and build a shared understanding of the project's goals and objectives. The observations of this course support the current research.

The goal of this interdepartmental approach was to create a simulated environment where diverse groups, separated by not on time and space, but by professional culture could be exposed to and overcome common hurdles found in most professional management and development environments today. It is the author's opinion that this goal was met. Requiring a joint project deliverable from two very different groups of students, meeting at different times and in different buildings with required weekly collaboration intersections did present students with "real word" challenges that needed to be solved for the project to be successful.

The instructors plan to continue to use this module in their instruction, and to expand their effort to include formalize research using this cross-functional collaboration with the discipline specific version to gather more data on the student engagement responses.

fReferences

[1] Lohr, S., & Markoff, J. (2006). Windows is so slow, but why. Te New York Times, Mar..(Referenced on page.).
[2] Rogers, Y. (1992, December). Ghosts in the network: distributed troubleshooting in a shared working environment. In Proceedings of the 1992 ACM conference on Computer-supported cooperative work (pp. 346-355).

[3] Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2009). A framework for adapting agile development methodologies. European Journal of Information Systems, 18(4), 332-343.

[4] Rahman, N.,Wittman, A., & Thabet, S. (2016). Managing an engineering project. International Journal of Information Technology Project Management (IJITPM), 7(1), 1-17.

[5] Korkala, M.,& Abrahamsson, P. (2007, August). Communication in distributed agile development: A case study. In 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROM/CRO 2007) (pp. 203-210). IEEE.

[6] Sarker, S., Ahuja, M., & Sark.er, S. (2018). Work-life conflict of globally distributed software development personnel: An empirical investigation using border theory. Information Systems Research, 29(1), 103-126.

[7] Sengupta, B., Chandra, S.,& Sinha, V. (2006, May). A research agenda for distributed software development. In Proceedings of the 28th international conference on Software engineering (pp. 731-740).

[8] Moniruzzaman, A B. M.,& Hossain, D.S. A (2013). Comparative Study on Agile software development methodologies. arXiv preprint arXiv:1307.3356.

[9] Ahmed, F.,Capretz, L. F., & Campbell, P. (2012). Evaluating the demand for soft skills in software development. It Professional, 14(1), 44-49.

[10] Montero, J. G., Lopez, J. L., Garcia, M. I., & Garcia, M. (2019). Impact of interdepartmental collaboration on software engineering teams: Evidence from Spanish software companies. Journal of Software Engineering Research and Development, 7(1), 1-17.

[11] Dutta, P., Hira, M., & Kumar, A. (2018). Impact of inter-departmental collaboration on software engineering. International Journal of Computer Science and Information Security, 16(2), 30-36.

[12] Gudiseva, K., Goel, A., Kumar, A., Chaudhary, S., & Chawla, V. (2017). An interdepartmental approach for effective cross-functional collaboration in software engineering. International Journal of Computer Science and Information Security, 15(3), 59---{)5.

[13] Michaelsen, L. K.,& Sweet, M. (2008). The essential elements of team-based learning. New directions for teaching and teaming, 2008(116), 7-27.

[14] Smart, K. L., & Csapo, N. (2007). Learning by doing: Engaging students through learner-centered activities. Business Communication Quarterly, 70(4), 451-457.