Teaching Software Development for Real-World Problems using a Microservice-Based Collaborative Problem-Solving Approach

Yi Meng Lau
School of Computing and Information Systems
Singapore Management University
Singapore
ymlau@smu.edu.sg

Christian Michael Koh
School of Computing and Information Systems
Singapore Management University
Singapore
ckoh.2021@scis.smu.edu.sg

Lingxiao Jiang
School of Computing and Information Systems
Singapore Management University
Singapore
lxjiang@smu.edu.sg

ABSTRACT

Experienced and skillful software developers are needed in organizations to develop software products effective for their business with shortened time-to-market. Such developers will not only need to code but also be able to work in teams and collaboratively solve real-world problems that organizations are facing. It is challenging for educators to nurture students to become such developers with strong technical, social, and cognitive skills.

Towards addressing the challenge, this study presents a Collaborative Software Development Project Framework for a course that focuses on learning microservices architectures and developing a software application for a real-world business. Students get to work in teams to solve a real-world problem of their own choice. They are given opportunities to recognize that the software development process goes beyond writing code and that social and cognitive skills in engaging with each other are also essential. By adopting microservices architectures in the course, students learn to break down the functionalities of their applications into smaller pieces of code with standardized interfaces that can be developed, tested, and deployed independently. This not only helps students to learn various technical skills needed for developing and implementing the functionalities needed by the application in the form of microservices but also facilitates task allocation and coordination among their team members and provides a platform for them to solve problems collaboratively. Upon completion of their projects, students are also asked to reflect on their development process and encouraged to think beyond the basics for better software design and development approaches.

The course curriculum incorporates the framework, especially for the student team projects. The earlier teaching weeks introduce a combination of concepts and lab exercises to students as the building blocks. The survey studies show that the framework is effective in enhancing the students' learning of technical, social, and cognitive skills, while further improvements, such as closer collaboration with other courses, can be done to improve a holistic learning curriculum.

ICSE-SEET'24, April 14-20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0498-7/24/04
https://doi.org/10.1145/3639474.3640064

CCS CONCEPTS

• Software and its engineering → Software creation and management → Collaboration in software development; • Software and its engineering → Software notations and tools → Development frameworks and environments → Software as a service orchestration system; • Applied computing → Enterprise computing → Service-oriented architectures; • Information systems → World Wide Web → Web services

KEYWORDS

software development, collaborative problem-solving, real-world solutions, microservices architectures

ACM Reference format:

Yi Meng Lau, Christian Michael Koh and Lingxiao Jiang. 2024. Teaching Software Development for Real-World Problems using a Microservice-Based Collaborative Problem-Solving Approach. In *Proceedings of IEEE/ACM 46th International Conference on Software Engineering Education and Training (ICSE-SEET'24), Lisbon, Portugal, 12 pages.* https://doi.org/10.1145/3639474.3640064

1 Introduction

With the rapid advancement of technology where digitalization has become the norm as organizations transform, there is an increased need for software developers. It is noted that the U.S. Bureau of Labor Statistics predicts a 26% increase in employment for software developers over the next ten years [1]. As the demand for highly skilled software engineers continues to rise, the education system must be aligned with current market needs [2]. While there are emerging tools that can speed up software development, the fundamental processes in software development still require experienced individuals who understand the underlying principles and have the skills to produce effective software products, minimizing loss due to failed software [3].

Software development has been defined by IEEE as "the application of a systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software" [4].

The role of a software developer is no longer just writing a set of codes that are working but requires the ability to understand real-world problems within the industry domain to design and develop effective solutions [3] [5] [6] [7]. A typical software development process in the industry requires software developers to work as a team to analyze and understand the needs of real-world issues before addressing the needs [2]. This continues even after the software implementation for future enhancements and bug fixing [4] [7] [8]. Software developers must possess both soft and technical skills for effective problem-solving [9] [10].

Organizations are often under tremendous pressure for quicker releases to stay competitive. This indirectly asks for software development turnaround time to be shorter, with reduced bugs, failure risks, and shorter development life cycles [11]. This was significant during the COVID-19 pandemic when organizations faced overwhelming changes where technological solutions had to be implemented in a short time [12] [13]. The responsibility essentially lies with software developers who need to be highly responsive to unforeseen changes and flexible for quicker deployment.

A microservices architecture (MSA) is a software architectural pattern where functions of an application are broken down into small, independent pieces of code with language- and platform-agnostic interfaces that can be developed, tested, and deployed independently [14]. They can be reused by different applications, making software development flexible and agile [15]. Adopting a microservices architecture will enable organizations to develop applications as a set of loosely coupled services [16]. Software developers can choose to assemble different services based on their needs, and software solutions can be delivered more efficiently, reducing time to market [17]. With time pressure for software to be rolled out quickly to adapt to new changes, adopting microservices architectures in system designs can play a critical role [18].

To learn about microservices architecture design, students can either go through formal degree programs that provide a foundation in computer science or explore alternative paths, such as boot camps, coding schools, self-studying with online resources, and MOOCs (Massive Open Online Courses) [7] [10]. In most programming courses, the focus is on coding skill development and mastering programming language features before exploring development processes and software architectural designs [19]. Computing courses emphasize technical skills, neglecting social aspects in professional growth [20].

This study presents a Collaborative Software Development Project Framework for a Year-2 university software development course that focuses on microservices architectures. The course is conducted in person, emphasizing team projects, which are frequently used in undergraduate software development courses [21]. Students work collaboratively in teams to solve real-world problems using microservices architectures. This approach provides students with valuable practical experience and fosters teamwork in collaborative problem-solving, on top of learning technical skills. The framework aims to achieve the following:

- Students work in teams to collaboratively solve a multifaceted real-world problem of their choice using microservices architectures.
- Students recognize that software development goes beyond writing functional codes [5] [22]. Their collaborative efforts are essential, requiring the development of their social and cognitive skills as they engage in task planning, interaction, and learning from each other, they are better equipped to address real-world problems.
- Students get to reflect on their software development processes and are encouraged to think beyond the basics for better software designs and approaches.

The study seeks to assess the effectiveness of this framework applied to a software development course, by addressing the following research questions:

- RQ1: Are students able to identify a real-world problem for their project and relate it to a possible solution taught in class?
- RQ2: Are students able to acquire technical competencies learned in the course and applied to their project?
- RQ3: Have students acquired social and cognitive skills working on the project?
- RQ4: Are students able to reflect on their piece of work, look beyond the project, and identify areas for improvement?

The subsequent sections of this paper are organized as follows. Section 2 shares related works in the field. Section 3 describes the proposed Collaborative Software Development Project Framework in comparison to related work. Section 4 applies the framework to a second-year undergraduate software development course. Section 5 presents the results and findings of this study. Section 6 addresses the lessons learnt and the limitations of this study while suggesting potential areas for future work. Section 7 concludes.

2 Related Work

2.1 Collaborative Problem-Solving and Software Development

Organizations constantly face increasingly complex problems [23], resulting in the need to rely on team projects to solve problems [24] [25] [26]. A team's effort is required to make sense of the challenges and perform deeper analysis and understanding of the subject matter formulating, planning, executing, monitoring, and reflecting before reaching common ground [27] [28]. Such problem-solving requires collaborative work and effort.

Collaborative problem-solving can be practiced in universities [24] as a structured process in which students take time to understand the problem statement identify a problem, analyze the problem, evaluate it, and derive possible solutions to develop the solution [29]. Studies have found that STEM problem-solving often requires a substantial amount of domain knowledge, but successful teams collaborate and build their knowledge on top of their teammates' establishing a knowledge support structure [30]. This can also be applied to the discipline of software engineering.

It is rare to have software development done by a single developer but more as a team effort [10]. Teamwork is critical in software development, and all members need to work together to achieve optimal solutions [31]. Software quality has also become a keen interest to organizations [32] as software developments which are often constrained by scope, cost, and time will need a concerted team effort to accomplish the tasks [33] and every software development will demand a certain amount of effort to complete by a team [34].

Traditional software development methodologies like the waterfall model and the Rational Unified Process (RUP) model have distinct phases for individuals working as a team towards project completion [32]. The later ones such as Scrum and extreme programming emphasize more on teamwork and a collaborative mindset [35] [36] [37]. Scrum fosters collaboration within teams and in extreme programming, all team members are equals within the collaborative team [35]. These reiterate the need for collaborative problem-solving in software development to achieve software quality and team effectiveness [31].

2.2 Microservices as a Software Solution

Microservices architectures are widely adopted in software solutions for compelling reasons [38]. A microservices architecture is one where applications are designed as small applications that provide flexibility and modularity on a network or platform and can be independently deployed [16]. The agility of microservices architecture offers flexibility in development, deployment, operations, versioning, and scaling [39]. Microservices are platform-agnostic and can be integrated with DevOps practices for continuous integration and continuous delivery [18] [40] [41]. Continuous integration and delivery will allow organizations to have more frequent deployment and releases of software, reducing risks and yet providing up-to-date features [42]. In contrast, the conventional approach of designing and developing software as a monolith, where the whole application is packed into a single artifact and running as one deployable unit, has often raised difficulties in the maintainability and the introduction of new features to the application [39] [43].

Organizations with software that is designed on microservices architecture are benefiting from it with its flexibility and agility to pull together a solution quickly to achieve sustained competitive advantage [41] and speed up delivery to cater to market competitions [39] and provide functionalities that are aligned with organizations objectives [15].

While organizations such as Amazon and Guardian have benefited from microservices architectures [40] [41] [44], to effectively implement and manage large-scale applications using microservices architecture will require skills in distributed systems development and DevOps [41]. The development team needs to be equipped with the necessary skills and knowledge in using microservices architecture and the designing of the system, taking into consideration different aspects from network reliability and bandwidth [41]. The alignment of the real-world context to a technical solution will require domain expertise from the development team as there are no clear guidelines for which microservices will be created for the application [45].

Although there are challenges in microservices architecture as it is deemed more complex in setup and may be more expensive given the number of resources it needed [40] if adopted successfully, the microservices architectures would also bring cultural change concerning quality awareness and responsibilities [18] to the developer's team.

In light of this perspective, the forthcoming proposed framework aims to provide students with the chance to experience team-based software development processes where they collaboratively solve real-world problems by implementing microservice solutions that resonate with the real-world scenario.

3 Proposed Framework

3.1 Learning Microservices-Based Development

A microservices architecture distinguishes itself with its potential benefits from a monolithic design [40] [41] [44]. Software developers having experience and knowledge in microservices architectures will be highly sought after in the industry [46].

Software developers acquire expertise through practical, hands-on experience across diverse domains. The inherent complexity of software development adds another layer of uncertainty [3]. Effectively developing software solutions for real-world problems demands thorough analysis from various perspectives, which necessitates effective teamwork to comprehend the problem before coming up with solutions [33] [47] [48]. Research has also shown that software development is a complex socio-technical activity where strong communication skills play a crucial role on top of the need for good technical skills [4] [9]. A holistic learning curriculum in software development will need to encompass all these aspects.

3.2 Collaborative Software Development Project Framework

Collaboration stands as a fundamental mode of human interaction and problem-solving enabling individuals to work collectively. Collaborative problem-solving is fundamentally socio-cognitive [49]. Engaging with peers fosters interactions that can lead to significant achievement gains among the participants [50]. The proposed Collaborative Software Development Project Framework enables students to collaboratively problem-solve and develop a solution for a real-world problem using microservices architectures, bridging the gap between theory and practice with real-world problems [51].

The proposed framework consists of four stages: Ideation will allow students to identify a real-world problem and propose an initial design, Development is where students develop a solution using a microservices architecture, ongoing Testing is carried out to improve their solutions, resolving any bugs. Through a Presentation, students showcase their prototypes and receive feedback. In Post Learning, students encapsulate and reflect on their learning journey for the project through a set of survey questions. The proposed framework's stages are coordinated with

Decision Points in an Established Collaborative Problem-Solving Model	Proposed Collaborative Software Development Project Framework
Identify a Problem	Ideation - Identifying a real-world problem
Gather Information	
Analyze the Situation	Ideation - Proposed an initial design
Generate Possible Options	
Evaluate the Feasibility of Ideas	
Reach Consensus	
Develop a Plan and Implementation	Development
	Testing
--	Presentation
	Post Learning

Table 1: Mapping of Collaborative Software Development against Decision Points in an Established Collaborative Problem-Solving Model

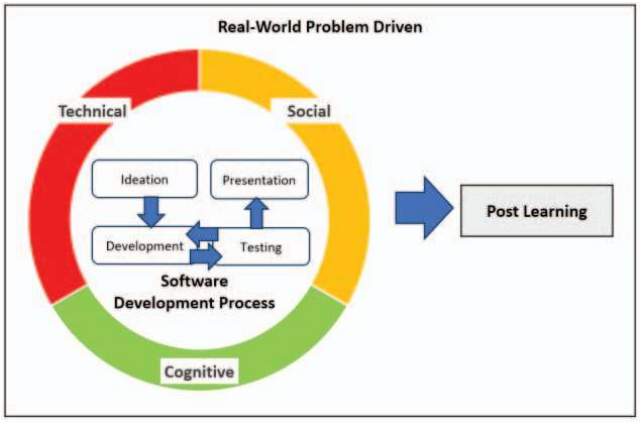


Figure 1: Proposed Collaborative Software Development Project Framework

decision points sourced from an established collaborative problem-solving model [53], as presented in Table 1.

It is observed that many software development courses predominantly emphasize the imparting of technical skills while often overlooking equally important soft skills. This deficiency in soft skills training can lead to challenges for graduates [54]. Hesse et al. illustrated that students who are engaged in collaborative problem-solving can simultaneously develop their social and cognitive skills [22] [26]. These skills are widely recognized as crucial competencies in the 21st century. Software developers need to possess commendable communication, analytical, problem-solving, and interpersonal skills [10], which students can acquire through collaborative problem-solving is detailed in Table 2.

Given that the framework is developed for software development courses in microservices architectures (MSA), students are expected to also acquire many technical skills, such as the design and implementation of microservices providing needed functionalities for real-world problems, Service-Oriented Architecture (SOA) layers, communication patterns using messaging queues, API gateways, Docker containers, and more. The complete Collaborative Software Development Project Framework is illustrated in Figure 1.

4 Framework Applied to a Development Course

The Collaborative Software Development Project Framework is piloted in a Year-2 undergraduate Information Systems course on developing enterprise applications to assess its effectiveness.

4.1 Enterprise Solution Development Course

The Enterprise Solution Development course studies the evolution of both emerging technologies and existing ones, exploring how enterprises adapt their applications to meet the heightened expectations of customers. Traditionally, enterprises opted for building monolithic applications due to simplicity. However, with the need for shorter time-to-market, more frequent releases, and rapidly evolving ecosystems, the focus has shifted towards constructing application solutions by harnessing existing functionality exposed as services and developing new functionality as microservices. The shift aligns with the principles of SOA and MSA, where applications are constructed as assemblies of loosely connected services and/or microservices.

In this course, students will gain experience in designing and implementing enterprise applications using SOA and MSA, utilizing appropriate techniques and tools. The curriculum encompasses a range of topics, including the basic concepts and principles of SOA and MSA and their relations to business

Skills from Hesse's Collaborative Problem-Solving Framework		
Social	Participation	Students' willingness and readiness to share information and thoughts with others. Their openness to engaging and interacting with others may result in being involved which will foster collaboration and accomplishing tasks.
	Cooperation	Students adeptly reconcile differences in team members' viewpoints, recognizing each other's strengths and weaknesses. They collectively undertake the responsibility to complete the group's required task.
	Perspective taking	Students can see the problem from the view of other collaborators.
Cognitive	Organizes	Students can divide the problem into subtasks such as to meet the project schedule.
	Adaptable	An individual being able to adjust changes in plans and being flexible to accommodate changes
	Be Flexible	Being creative in problem-solving particularly, the challenging ones.
	Learn new skills	Students learn from each other through the group's interactions.

Table 2: Social and Cognitive Skills from Hesse's Collaborative Problem-Solving Framework

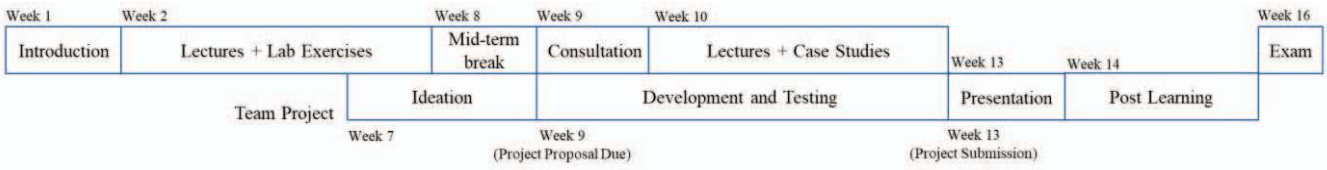


Figure 2: Course Curriculum and Project Timeline

processes, Python programming skills, and commonly used Python packages (e.g., Flask and Flask-SQLAlchemy) for supporting the development of microservices with databases, HTTP-based or AMQP-based communication patterns and technologies for interactions among services, JSON and XML-based data transformation and service interface definitions, web user interfaces based on HTML/CSS/JavaScript, Docker-based deployment of the services and applications, and API Gateway (e.g., Kong) for managing accesses to services. This course is examinable. Other assessment components include class participation, quizzes, and an important team project.

The upper half of Figure 2 gives an overview of the timeline of the course curriculum over the university term. Week 1 introduces the basic concepts and principles. Weeks 2-7 teach the commonly used techniques and tools for developing and deploying microservices. This continues in Weeks 10-12 where more advanced topics and case studies are shared. The students are informed of the expected team project in Week 1 and are encouraged to start to form a team of 5-6 students and look for real-world problems that they would like to address. The students do not have differentiated roles, and there should be equal involvement from all members. The project requirements (more in Section 4.2) are also officially released to students at the beginning of Week 7.

4.2 Applying the Collaborative Software Development Project Framework to Project

The course project requires students to work in their teams, to design and implement an enterprise application for a real-world problem using a microservices architecture. Students form their own teams and choose their real-world problems. To facilitate students' learning, sample real-world problems with simplified solutions are shown in the course via lectures, lab exercises, and case studies. For example, the course incorporates simplified business processes and functionalities (e.g., browsing books, placing an order for books) of an online bookstore into the lectures and lab exercises, providing concrete scenarios for students to learn how to identify suitable microservices in a scenario (e.g., a Book microservice, an Order microservice, etc.) and choose appropriate techniques and tools to implement and assemble the microservices and user interfaces into mini web applications. Another case study on a simplified taxi-booking application is used to explain how real-world problems involve more microservices (e.g., a Passenger microservice, a Driver microservice, a Trip microservice, a Billing microservice, a Payment microservice, a Notification microservice) and how their complex interactions can follow certain business processes and usage scenarios.

Students also learn to draw various kinds of architectural diagrams, such as Microservice Interaction Diagrams and Technical Overview Diagrams (see samples in Figure 5), to express their designs and use them to guide their implementations. A Microservices Interaction Diagram illustrates the interaction flow between different microservices in an MSA. It provides a visual representation of how individual microservices communicate with each other to collectively deliver a specific function. The Technical Overview Diagram illustrates the relationship between complex and simple microservices with information on the API endpoints.

The project requirements are defined using the concepts and technologies taught in the course as the baseline:

- Include a minimum of three user scenarios, with two of them requiring the use of complex microservices to coordinate the simple ones.
- Incorporate a minimum of three simple microservices, each having its data entities and independent datastore. Additionally, one of these microservices is utilized across multiple user scenarios.
- Implement the capability to call external services provided by third parties (e.g., Google, PayPal).
- Establish either HTTP-based or AMQP-based communication between select microservices within the application.
- Develop a web-based user interface to simplify the demonstration of the prototype during presentations.
- Implement a Docker solution that is tailored to the specific containerization needs of the application.
- Incorporate at least one technology not taught in the course into the project solution; it could be a different programming language or framework for implementing microservices, a different type of database, a different containerization tool, a different API gateway, etc.

The bottom half of Figure 2 shows the timeline of the team project aligned with the Collaborative Software Development Project Framework. The project spans over six weeks from ideation to implementation and presentation. Students receive the official project requirements at the beginning of Week 7 when they should have formed their project teams. In Week 9, students have dedicated time slots to review their project proposal and initial designs with the course instructors to receive feedback and suggestions on their microservices architectures, while they can seek additional feedback throughout the whole duration. In Week 13, students will deliver a presentation, including a demonstration, to showcase how their solutions can address the identified problem. Students will describe the technology stack they have used and explain in detail the integration of their microservices, emphasizing their advantages within the context of the various user scenarios. Each team's final submission includes presentation slides, a comprehensive project report, a functional codebase, and a video

walkthrough of their application. After their project presentation and report submission, students have a couple of weeks to reflect on their learning and revise the course materials before the final closed-book examination.

4.3 Evaluating the Collaborative Software Development Project Framework

The Collaborative Software Development Project Framework is evaluated through two optional surveys and students' final reports. The project is one of the assessment components of the course that may not be relevant to consider with the final distribution of grades. The survey studies have been approved by the university's Institutional Review Board (IRB). The first survey happens in Week 9, and the second in Week 14, which includes Post Learning. The Likert scale from 1 (least favorable) to 10 (most favorable), with open-ended questions, is used. Response rates are 37.2% for the first survey and 18.9% for the second from a cohort of 390 students, across about 80 project teams. Evaluation of cognitive and social skills assessed through final project reports alongside the surveys. Survey questions and students' responses are available in the anonymized supplementary material.

5 Results and Findings

5.1 Real-World Problems

- RQ1: Are students able to identify a real-world problem for their project and relate it to a possible solution taught in class?

The Week 9 survey findings (in Table 3) suggested that students exhibit the ability to identify real-world problems for their projects, relating them to technical solutions covered in the course. On the ease of identifying a real-world problem, students scored an average of 6.37 out of 10, with a standard deviation of 1.866. Most students who responded found it easy to correlate the real-world problem with a potential technical solution, scoring an average of 6.94 with a standard deviation of 1.514.

Students further demonstrate their ability to define problem statements by approaching them from various perspectives. They consider factors like industry demands, persona's pain points, business unit challenges, immediate and future benefits, as well as data availability through APIs. This multifaceted approach highlights their proficiency in identifying real-world issues suitable for solutions using microservices architectures. The statistics are presented in Figure 3.

5.2 Technical Skills

- RQ2: Are students able to acquire technical competencies learned in the course and applied to their project?

From the technical perspective, the students' responses from the Week 13 survey are positive overall. It showed that most students have gained proficiency in designing and developing microservices

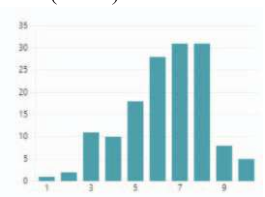
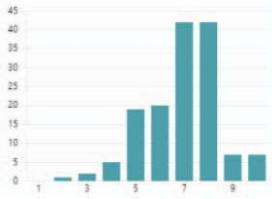
Survey Question	Average (Std Deviation) Distribution of Survey Results
It was easy to identify a real-world problem to solve for the group project.	6.37 (1.866) 
It was easy to relate the real-world problem to a possible solution design taught in the course.	6.94 (1.514) 

Table 3: Survey Results on the Identification of Business Problem and Relating it to the Course

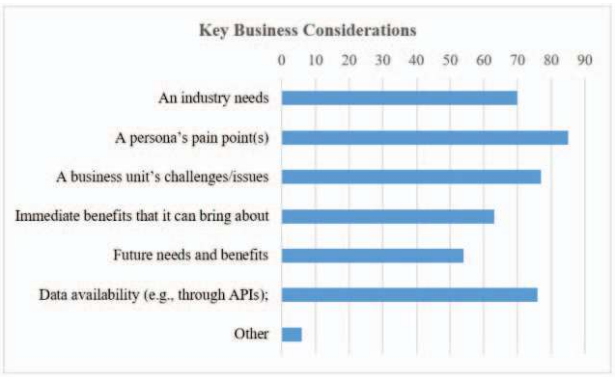


Figure 3: Survey Results - Key Business Considerations

(7.53 and 7.42 out of 10), designing SOA, and using Docker (7.46 and 7.31 out of 10). However, they exhibit less confidence in the development of Service Oriented Architecture and using API gateways, Docker, and message queues, suggesting a lower level of competence in these domains. 4 illustrates the results.

When asked by the open-ended survey questions about alternative tools that they could use in their projects in the future, students presented a variety such as different programming languages, Kubernetes, Express.js, React.js, AWS services, Vue.js, Tailwind, Kafka, cloud resources, Java/Spring Boot for microservices, Django, Swagger, and Telegram bot. This reflects the students' capacity to broaden software development skills and willingness to explore diverse tools.

The students also provided a comprehensive list of proposed technical improvements they would like to make in the future, such as adopting object-oriented design principles, emphasizing thorough initial planning, diversifying communication technologies besides AMQP, alternatives to MySQL databases, implementing Kubernetes, prioritizing event-driven service development, exploring choreography design of microservice interactions versus orchestration design, employing various API

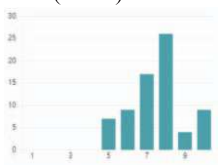
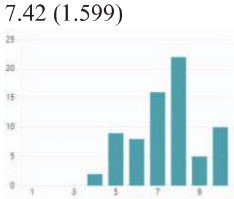
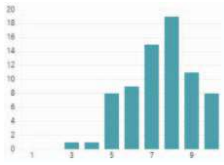
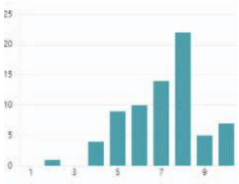
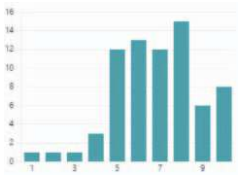
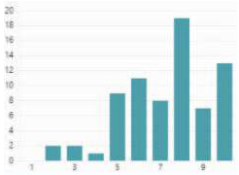
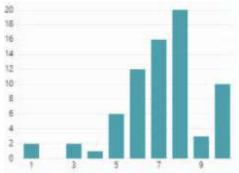
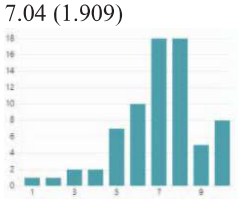
Survey Question	Average (Std Deviation) Distribution of Survey Results
Design of microservices	7.53 (1.404) 
Development of microservices	7.42 (1.599) 
Design of Service Oriented Architecture (SOA) layers	7.46 (1.618) 
Development of Service Oriented Architecture (SOA) layers	7.11 (1.724) 
Using API gateways	6.86 (1.967) 
Using Docker	7.31 (2.039) 
Design of communication patterns with the application of messaging queues	7.13 (1.950) 
Use of communication patterns with the application of messaging queues	7.04 (1.909) 

Table 4: Survey Results on Mastering the Key Technical skills

gateways for enhanced security functionality, considering a more loosely coupled SOA, and leveraging cloud services.

Additionally, they suggest a different application design approach by beginning with a front-end design to have a clearer understanding of the users' needs and inform better microservice parameter decisions; and they recognize the need to potentially implement intermediary management services to decouple microservices and services further from each other when business requirements would go through frequent changes.

In summary, it indicates that students have gained a significant level of technical competency through their project experiences.

5.3 Social Skills

- RQ3: Have students acquired social and cognitive skills from the project?

To prepare students for the challenges of future real-world problems, social and cognitive skills are important. This study looks at how students can learn three aspects of social skills and four cognitive skills through their project report.

The three aspects of social skills that students will acquire are participation, cooperation, and perspective-taking. Students are engaged in a variety of tasks besides coding.

Each project team will summarize each student's contribution. An example is shown in Figure 4. Their collaborative effort is needed to work on their Microservice Interaction Diagrams and Technical Overview Diagrams in Figure 5. It is an example of the microservices architecture design of a Pet Sitter application. In each scenario, the diagram on the left shows interactions steps between microservices. The diagram on the right provides a technical overview of the microservices architecture. Scenario 1 shows a Pet Owner seeking assistance with pet sitting, generating a job opportunity (Create a Job) and inviting interested Pet Sitters to apply (Send Job request out to Preferred Sitters). In Scenario 2, the Pet Owner reviews a list of applications from Pet Sitter before selecting a suitable candidate (Accept Job Application). Each student in the team will take on different tasks to design and develop the solution. This shows **participation** in the project.

While simple microservices can be designed and coded independently, there are interactions between the complex and simple microservices. Besides working individually on their assigned task, students will have to work with each other to ensure that the microservices can be integrated as one. The need for **cooperation** and collaboration between the students who are developing the microservices is necessary, as they will need to be

Name	Contributions
Student 1	Responsible for the "Owner Accepting Sitter" and "Payment" process. Ensure all associated microservices are integrated and handle AMQP setup and external services integration.
Student 2	Responsible for the "Job Creation" process. Ensure all associated microservices are integrated and handle AMQP setup and external services integration.
Student 3	Responsible for the "Sitter or Owner Pulling Out" process. Ensure all associated microservices are integrated and handle AMQP setup and external services integration.
Student 4	Responsible for the "Penalty notification" processes and all its associated microservices. Database configuration for all tables.
Student 5	Responsible for database configuration for Sitter and the related microservices.

Figure 4: Students' Contributions to a Project Team

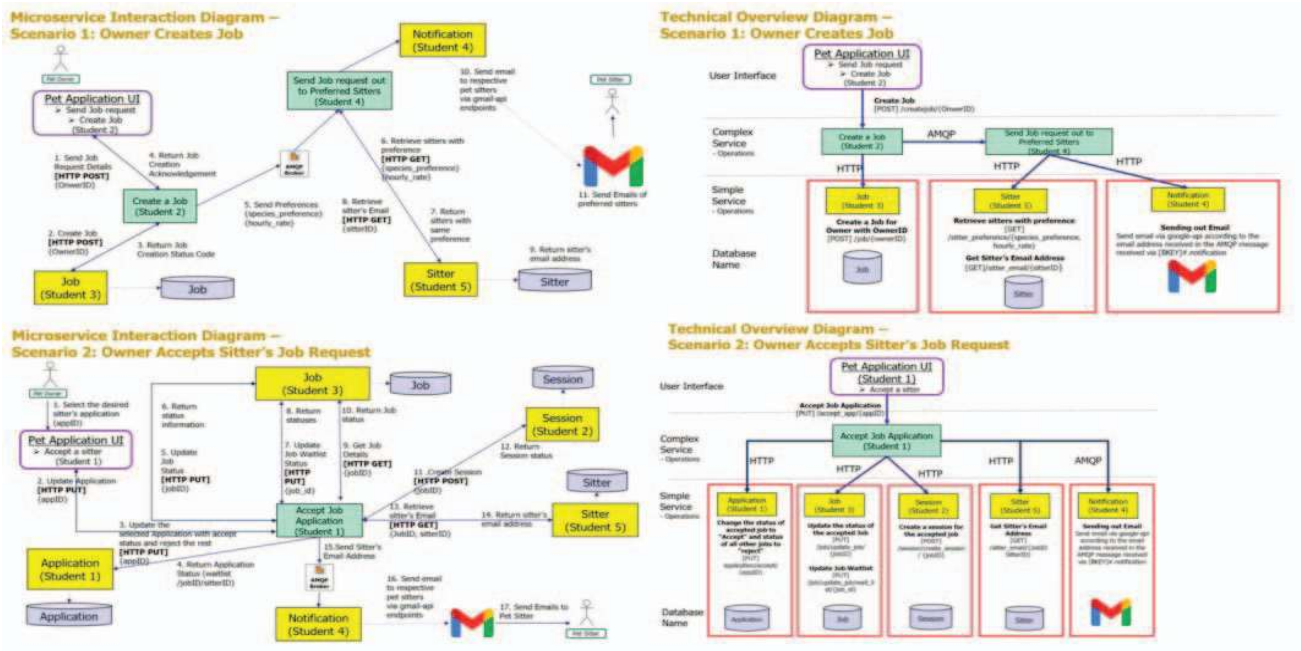


Figure 5: Example of Microservice Interaction Diagram and Technical Overview Diagram from a Project Team

familiar with how each other's microservices function and the format of the data on the receiving end.

Students while working with teammates' microservices, may offer alternative implementations and seek insights into the reasons behind specific choices. This promotes perspective-taking, encouraging students to suggest improvements and understand the rationale behind each other's microservice design and development.

The survey results highlighted in Table 5 show students' ability to consider problems from multiple angles and appreciate the feedback from the teaching team. Students are confident that their application can solve the real-world problem with MSA.

The cognitive skills that students can acquire are the ability to organize themselves, be adaptable, be flexible, and able to learn new skills through interaction with one another. Figure 2 outlines a timeline aligned with the course curriculum, allowing minimal flexibility for deadline extensions. To complete the project, students need to **organize**, divide up the tasks, and coordinate to ensure smooth integration of their microservices.

As in all software development projects, adjustments will be required to accommodate changes. They are new to technologies such as API gateways, AMQP messaging queues, and Docker technology. Being **adaptable** and **flexible** is a skill to tackle any unforeseen technological challenges. In some cases, they may need to think creatively to evaluate alternative ways to develop their solution should the technology not work for them. These are imperative to their success in the project.

In this context, students acquired knowledge not solely from the course delivery but also from active project participation and mutual interactions. This collaborative problem-solving approach allows them to **learn new skills** from each other. It is essential to highlight that constructing a microservices architecture is inherently a collective effort, requiring teamwork and robust communication rather than an isolated endeavor.

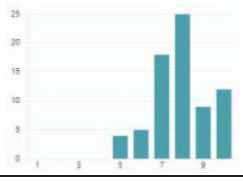
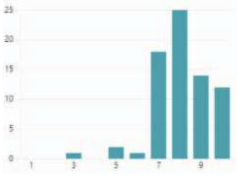
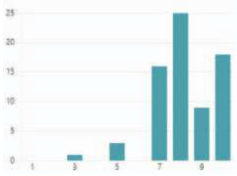
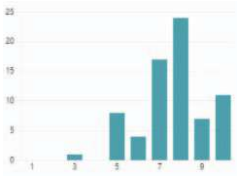
Survey Question	**Average (Std Deviation) Distribution of Survey Results**
Was the teaching team's feedback useful for you to look at different business angles for your team's project?	7.90 (1.345) 
How would you grade your application to solve the business needs?	8.10 (1.325) 
Was the teaching team's feedback useful for you to look at different technical angles for your team's project	8.21 (1.433) 
You are confident in designing the technical elements of this project e.g., microservices, SOA, etc. using the business processes and use cases of your problem.	7.65 (1.557) 

Table 5: Survey Results on Students' Assessment of Their Team Projects

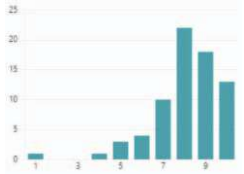
Survey Question	Average (Std Deviation) Distribution of Survey Results
How intuitive is the usability of this solution?	8.08 (1.645) 

Table 6: Survey Result on Usability of the Solution

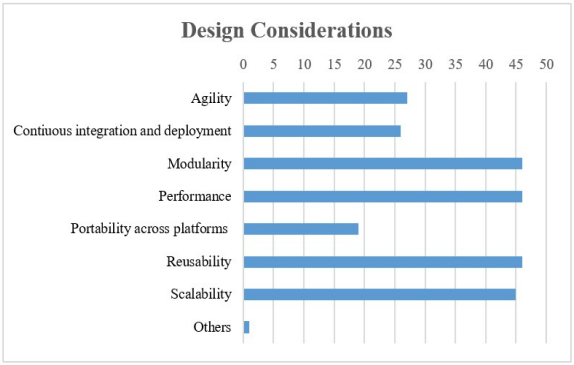


Figure 7: Survey Results on Design Considerations

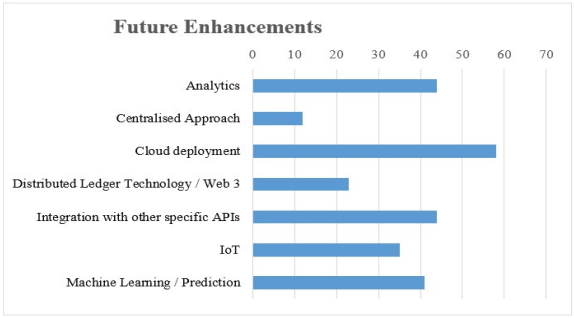


Figure 6: Survey Results on Possible Future Enhancements of Team Project

5.4 Post Learning

- RQ4: Are students able to reflect on their piece of work, look beyond the project, and identify areas for improvement?

In Post Learning, students reflect on their team project through questions provided in the survey, spanning technical and business perspectives.

On the business front, students are asked to suggest additional features to incorporate into their solution if given an opportunity. They suggest innovative ideas such as the use of IoT, chat features, user authentication, customers' preferences dashboards, and notification services. Students assessed the usability and user experience of their solution, assessing its effectiveness in addressing identified personas' pain points and potential expansion to new target audiences. A positive average score of 8.08 is documented in Table 6.

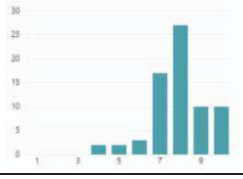
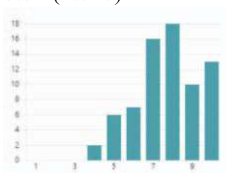
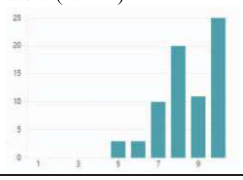
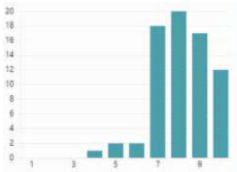
Survey Question	Average (Std Deviation) Distribution of Survey Results
What is your perspective of "Business drives Technology"?	7.90 (1.354) 
What is your perspective of "Technology drives Business"?	7.72 (1.620) 
What is your perspective of "Both technology and business play a critical role in solving real-world problems."	8.50 (1.414) 
I am able to see a good linkage between technology and business to solve a real-world problem.	8.13 (1.321) 

Table 7: Survey Results on linkage between technology and business

On the technical front, survey questions cover technical design considerations, alternative tools, software architecture, security, and future enhancements. The survey prompts students to reflect on their project design decisions and explore alternative solutions. Results from students on design considerations and future technologies are presented in Figure 6 and Figure 7.

For software architecture, students suggested designing the databases differently or aligning with industry standards. However, most still favored microservices architectures, dismissing other alternatives. In security, students suggested authentication and authorization using API gateways and login credentials from Google or equivalent to make the application more secure. Some suggested the use of tokens for better encryption capabilities.

The final set of questions assesses if students see the practical connection between technology and business. The results are detailed in Table 7. High scores of 8.50 and 8.13 indicate students recognize the crucial roles both play in solving business problems, perceiving a strong link between technology and business needs.

6 Lessons Learnt

This section shares the experience of the course execution from two perspectives. First, it examines if the coursework adequately prepares students for the team project. Second, it assesses if the proposed Collaborative Software Development Framework has achieved its stated objectives and explores the limitations identified in this study. Third, we explore potential future works that can be undertaken in this area.

6.1 Coursework Prepares Student for Project

The curriculum prepares students for their projects by covering the essential concepts of SOA and MSA in the early weeks. This foundation equips them to develop Microservice Interaction and Technical Overview Diagrams to guide them in their project design.

In the subsequent weeks from Weeks 2 to 9, the curriculum shifts its focus to the hands-on application of Python programming skills and key Python packages like Flask and Flask-SQL Alchemy. This phase emphasizes the use of these tools to support the development of microservices, covering aspects such as databases, communication patterns, and technologies for service interactions. Students also explore how their microservices can engage with web interfaces using JSON and XML-based data formats. The learning experience includes engaging lectures and practical lab exercises, providing students with small-scale, hands-on activities. This provides a good starting point for their project.

The curriculum introduces Docker and API Gateways in later stages, recognizing them as additional tools that students can leverage, typically in the later phases of their development cycle.

6.2 Limitations and Future Works

The evaluation of the Collaborative Software Development Project Framework is primarily conducted via two surveys and observations from students' project reports. Incentives can be introduced to encourage more participations.

The team could also make use of the Week 9 survey to pinpoint areas where students needed help, allowing for timely interventions. The more advanced students may want to incorporate features beyond the project's requirements and may find it advantageous to learn about them in the course. However, these topics are covered towards the end of the course which may leave them little time to integrate these features into their application. A possible solution could be to provide students with self-learning materials.

The post learning took place within the course. Survey questions are meant to encourage students to explore future alternatives and are currently not verified by the course instructors. The future utilization of the data remains unspecified. Additionally, the non-compulsory nature of the survey may lead to students not giving earnest consideration to the inputs, although the results from these insights for students to advance their projects.

Fourth, the existing framework is not able to recognize if any student requires assistance. It relies solely on students initiating contact with instructors. Establishing a process or system to identify and support at-risk students could be advantageous.

With a short project timeline and with an emphasis on learning microservices architectures, a traditional software development methodology is adopted. Project management tools like Trello or Jira would be beneficial for larger-scope projects with longer timeframes. They are used in subsequent courses.

Managing documentation for a project with numerous APIs can be more efficient using tools like swagger.io, which specializes in handling API documentation, rather than traditional word processing methods. This can be included as a suggestion for students to consider.

The Microservice Interaction Diagrams and Technical Overview Diagrams serve as blueprints illustrating the application's functionality. A potential next step could involve transforming these diagrams into detailed documentation as part of their project report. This approach would prove valuable, particularly considering the time constraints students have for their projects.

Lastly, the framework lacks more advanced objectives tools, and techniques for ensuring equitable team participation and measuring collaborative engagement between students. While some teams use GitHub, studies show no significant correlation between project quality and the number of commits [21] [55], highlighting an area for framework enhancement.

6.3 Suggested Adaption

The evaluation indicates that the framework performs effectively in this course and has the potential for adaption in similar software development courses beyond microservices architectures. However, adjustments may be required to align with the duration and content of the curriculum. The project timeline depicted in Figure 2 can be tailored to fit the course duration, and the scope of the project can be modified by varying requirements for the number of user scenarios. Other software development courses may potentially adopt this framework with a similar project scope, but the implementation may involve different technologies.

7 Conclusion

Organizations today depend on technology, and microservices-based solutions offer agility and flexibility. Developing solutions using microservices architectures requires hands-on experience and teamwork. This experience report shares how a Collaborative Software Development Project framework is implemented and aligned with the course curriculum of a Year 2 university course. Students work in teams and solve real-world problems with MSA, and at the same time practice to become developers with strong technical, social, and cognitive skills [22] [52] thereby meeting the demand of the increasing need for software developers [2]. The post learning process allows students to think about how they can improve and do better.

The framework has its limitations which provide the opportunity to refine the learning experiences, catering to students of different learning needs. This study will serve as a foundation for future works to improve the framework for application in similar courses.

ACKNOWLEDGEMENTS

The authors would like to express gratitude to the students and instructors involved in this course for making this study possible.

REFERENCES

[1] "The Software Developer Is Dead: Long Live The Software Developer," 29 3 2023. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2023/03/29/the-software-developer-is-dead-long-live-the-software-developer/?sh=abf250a25afc.

[2] K. Kuusinen and S. Albertsen , "Industry-Academy Collaboration in Teaching DevOps and Continuous Delivery to Software Engineering Students: Towards Improved Industrial Relevance in Higher Education," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, Montreal, QC, Canada, 2019.

[3] U. Gulec, M. Yilmaz, V. Isler and P. M. Clarke, "Applying virtual reality to teach the software development process to novice software engineers," *IEEE Software,* vol. 15, no. 6, pp. 464-483, 2021.

[4] E. Meade, N. Lyons, E. O'Keeffe, D. Lynch, M. Yilmaz, U. Gulec, R. V. O'Connor and P. M. Clarke, "The Changing Role of the Software Engineer," in *European Conference on Software Process Improvement*, 2019.

[5] S. Hamer, C. Quesada-López, A. Martínez and M. Jenkins, "Using git metrics to measure students' and teams' code contributions in software development projects," *CLEI Electronic Journal,* vol. 24, no. 2, 2021.

[6] S. McConnell, "Who needs software engineering?," *IEEE Software,* vol. 18, no. 1, pp. 5-8, 2001.

[7] B. Dasarathy, K. Sullivan, D. C. Schmidt, D. H. Fisher and A. Porter, "The past, present, and future of MOOCs and their relevance to software engineering," in *FOSE 2014: Future of Software Engineering Proceedings*, 2014.

[8] F. A. A. Rub and A. A. Issa, "A business process modeling-based approach to investigate complex processes: Software development case study," *Business Process Management Journal,* vol. 18, no. 1, pp. 122-137, 2012.

[9] D. Rabelo, A. Lopes, W. Mendes, C. de Souza, K. Gama, D. Moteriro and G. Pinto , "The Role of Non-Technical Skills in the Software Development Market," in *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*, 2022.

[10] F. Ahmed, "Software requirements engineer: An empirical study about non-technical skills," *Journal of Software,* vol. 7, no. 2, 2012.

[11] A. B. Farid, Y. M. Helmy and M. M. Bahloul, "Enhancing lean software development by using devops practices," *International Journal of Advanced Computer Science and Applications,* vol. 8, no. 7, pp. 267-277, 2017.

[12] F. Almeida, J. D. Santos and J. A. Monteiro, "The challenges and opportunities in the digitalization of companies in a post-COVID-19 world,," *IEEE Engineering Management Review,* vol. 48, no. 3, pp. 97-103, 2020.

[13] S. Li, H. Zhang, Z. Jia, C. Zhong, C. Zhang, Z. Shan, J. Shen and M. A. Babar, "Understanding and addressing quality attributes of microservices architecture: A Systematic literature review," *Information and Software Technology,* vol. 131, no. 106449,, 2021.

[14] J. Thones, "Microservices," *IEEE Software,* vol. 32, no. 1, p. 116, 2015.

[15] B. Antonio, N. Dragoni, S. Dustdar, S. T. Larsen and M. Mazzara, "From Monolithic to Microservices An Experience Report from the Banking Domain," *IEEE Software,* vol. 35, no. 3, pp. 50-55, 2018.

[16] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, R. Mustafin and L. Safina, "Microservices: Yesterday, Today and Tomorrow," Springer, Cham, 2017.

[17] A. Balalaie, A. Heydarnoori and P. Jamshidi, "Microservices Architecture Enables DevOps: An Experience Report on Migration to a Cloud-Native Architecture," *IEEE Software,* vol. 33, no. 3, pp. 42-52, 2016.

[18] J. Bogner, J. Fritzsch, S. Wagner and A. Zimmermann, "Industry practices and challenges for the evolvability assurance of microservices: An interview study and systematic grey literature review," *Empirical Software Engineering,* vol. 26, no. 5, pp. 2-39, 2021.

[19] A. C. M. Moskal and R. Wass, "Interpersonal process recall: a novel approach," *Computer,* vol. 29, no. 1, pp. 5-22, 2019.

[20] L. Tubino, K. Morgan, G. Wood-Bradley and A. Cain, "Shaping a Tool for Developing Computing Students' Professional Identity - Industry Perspectives," in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, Melbourne, Australia, 2023.

[21] C. Hundhausen, P. Conrad, O. Adesope and A. Tariq, "Combining GitHub, Chat, and Peer Evaluation Data to Assess Individual Contributions to Team Software Development Projects," *ACM Transactions on Computing Education,* vol. 23, no. 3, 2023.

[22] O. Hazzan, "Exponential Competence of Computer Science and Software Engineering Undergraduate Students," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, Madrid, ES, 2021.

[23] T. Nelson and V. Squires, "Addressing Complex Challenges through Adaptive Leadership: A Promising Approach to Collaborative Problem Solving," *Journal of Leadership Education,* vol. 16, no. 4, pp. 111-123, 2017.

[24] B. Barron, "Achieving Coordination in Collaborative Problem-Solving Groups," *The Journal of the Learning Sciences,* vol. 9, no. 4, pp. 403-436, 2000.

[25] N. M. M. Dowell, Y. Lin, A. Godfrey and C. Brookes, "Exploring the Relationship Between Emergent Sociocognitive Roles, Collaborative Problem-Solving Skills, and Outcomes: A Group Communication Analysis," *Journal of Learning Analytics,* vol. 7, no. 1, pp. 38-57, 2020.

[26] F. Hesse, E. Care, J. Buder, K. Sassenberg and P. Griffin, "A Framework for Teachable Collaborative Problem Solving Skills," in *Assessment and Teaching of 21st Century Skills Methods and Approach*, Springer Science+Business Media Dordrecht, 2015, pp. 37-56.

[27] A. C. Graesser, S. Greiff, S. M. Fiore and J. Andrews-Todd, "Advancing the Science of Collaborative Problem Solving," *Psychological Science in the Public Interest,* 2018.

[28] A. Graesser, B.-C. Kuo and C.-H. Liao, "Complex Problem Solving in Assessments of Collaborative Problem Solving," *Journal of Intelligence,* vol. 5, no. 10, 2017.

[29] L. R. Albert and R. Kim, "Developing Creativity Through Collaborative Problem Solving," *Journal of Mathematics Education at Teachers College,* vol. 4, 2013.

[30] K. Kim and A. A. Tawfik, "Different approaches to collaborative problem solving between successful versus less successful problem solvers: Tracking changes of knowledge structure," *Journal of Research on Technology in Education,* vol. 55, no. 4, pp. 628-645, 2023.

[31] D. Strode, T. Dingsøyr and Y. Lindsjorn, "A teamwork effectiveness model for agile software," *Empirical Software Engineering,* vol. 27, no. 2, 2022.

[32] M. A. Akbar, J. Sang, A. A. Khan, Fazal-E-Amin, Nasrullah, M. Shafiq, S. Hussain, H. Hu, M. Elahi and H. Xiang, "Improving the Quality of Software Development Process by Introducing a New Methodology-AZ-Model," *IEEE Access,* vol. 6, pp. 4811-4823, 2017.

[33] E. R. Sánchez, E. F. V. Santacruz and H. C. Maceda, "Effort and Cost Estimation Using Decision Tree Techniques and Story Points in Agile Software Development," *Mathematics,* vol. 11, no. 1477, 2023.

[34] M. Hericko, A. Živkovic and I. Rozman, "An approach to optimizing software development team size," *Information Processing Letters,* vol. 108, 2008.

[35] "Extreme Programming (XP)," [Online]. Available: https://www.strategic.tech/xp-extreme-programming. [Accessed 11 October 2023].

[36] M. Iqbal, "Collaboration with Structure," 22 May 2023. [Online]. Available: https://www.scrum.org/resources/blog/collaboration-structure. [Accessed 11 October 2023].

[37] A. Meier, M. Kropp and G. Perellano, "Experience Report of Teaching Agile Collaboration and Values: Agile Software Development in Large Student Teams," in *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, Dallas, TX, USA, 2016.

[38] T. Černý, M. J. Donahoo and M. Trnka, "Contextual understanding of microservice architecture: current and future directions," *ACM SIGAPP Applied Computing Review,* vol. 17, no. 4, pp. 29-45, 2018.

[39] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis and S. Tilkov, "Microservices The Journey So Far and Challenges Ahead," *IEEE Software,* vol. 35, pp. 24-35, 2018.

[40] I. Asrowardi, S. D. Putra and E. Subyantoro, "Designing microservice architectures for scalability and," *Journal of Physics: Conference Series,* vol. 1450, no. 1, 2020.

[41] S. Baskarada, V. Nguyen and A. Koronios, "Architecting Microservices: Practical Opportunities and Challenges," *Journal of Computer Information Systems,* vol. 60, no. 5, pp. 1-9, 2018.

[42] R. V. O'Connor, P. Elger and P. M. Clarke, "Continuous Software Engineering – A Microservices Architecture Perspective," *Journal of Software: Evolution and Process,* vol. 29, no. 11, 2017.

[43] "Towards Decomposing Monolithic Applications into Microservice," in *IEEE 14th International Conference on Application of Information and Communication Technologies (AICT)*, Tashkent, Uzbekistan, 2020.

[44] J. Ghofrani and D. Lübke, *Challenges of Microservices Architecture: A Survey on the State of the Practice,* Dresden, Germany, 2018.

[45] B. Hippchen,, P. Giessler, R. H. Steinegger, M. Schneider and S. Abeck, "Designing Microservice-Based Applications by Using a Domain-Driven Design Approach," *International Journal on Advances in Software,* vol. 10, no. 3 & 4, pp. 432-445, 2017.

[46] M. Waseem , Liang Peng and M. Shahin, "Waseem," *A Systematic Mapping Study on Microservices Architecture in DevOps,* vol. 170, p. Journal of Systems and Software, 2020.

[47] [Online]. Available: https://www.browserstack.com/guide/learn-software-development-process. [Accessed 11 October 2023].

[48] M. Shuto, H. Washizaki, K. Kakehi, Y. Fukazawa, S. Yamato and M. Okubo, "Learning Effectiveness of Team Discussions in Various Software Engineering Education Courses," in *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, Dallas, TX, USA, 2016.

[49] Z. Swiecki, A. R. Ruis, C. Farrell and D. W. Shaffer, "Assessing Individual Contributions to Collaborative Problem Solving: A Network Analysis Approach," *Computer in Human Behavior,* vol. 104, no. 105876, 2020.

[50] W. Damon and E. Phelps, "Critical distinctions among three approaches to peer education," *International Journal of Educational Research,* vol. 13, no. 1, pp. 9-19, 1989.

[51] R. Podeschi, "Building I.S. Professionals through a Real-World Client Project in a Database Application Development Course," *Information Systems Education Journal,* vol. 14, no. 6, pp. 34-40, 2016.

[52] Z. S. H. Abad, M. Bano and D. Zowghi, "How Much Authenticity can be Achieved in Software Engineering Project Based Courses?," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, Montreal, QC, Canada, 2019.

[53] "Collaborative problem-solving steps," Ripon College, 20 September 2023. [Online]. Available: https://ripon.edu/collaborative-problem-solving/.

[54] N. Assyne, H. Ghanbari and M. Pulkkinen, "The state of research on software engineering competencies: A systematic mapping study," *The Journal of Systems & Software,* vol. 185, 2022.

[55] M. Tushev, G. Williams and A. Mahmoud, "Using GitHub in large software engineering classes. An exploratory case study," *Computer Science Education,* vol. 30, no. 2, pp. 155-186, 2020.