EasyChair Preprint

№ 4665

Teaching Strategies in Software Engineering Towards Industry Interview Preparedness

William Gregory Johnson, Raj Sunderraman and Anu G. Bourgeois

November 27, 2020

Teaching Strategies in Software Engineering Towards Industry Interview Preparedness

William Gregory Johnson
wjohnson6@gsu.edu
Georgia State University
Atlanta, GA

Raj Sunderraman
rsunderraman@gsu.edu
Georgia State University
Atlanta, GA

Anu G. Bourgeois
abourgeois@gsu.edu
Georgia State University
Atlanta, GA

Abstract

The Software Engineering (SE) curriculum in undergraduate computer science (CS) education is designed to train students in the process of software and systems development. Traditionally, topics such as software development methodologies, industry nomenclatures, and solution analysis are delivered through lectures and group projects. We propose a novel approach in teaching SE that we call MACROVR: MAchine learning to select project team members; Cloud technologies required for project control, code versioning, and team communications; ROtational schedules in Agile/Scrum roles; an individual Video of the team project story board; and Rubrics for all presentations. Our teaching strategy with this approach utilizes the latest technologies currently employed in industry and corresponds to soft skills commonly assessed in interviews.

The goal of our study is to measure if using the MACROVR approach contributes to preparedness for a computing job interview. Most often, this course is taken towards the end of a four–year CS degree program while students are job hunting or seeking an internship in the computing industry. We use an anonymous, fifteen question survey instrument sent to volunteers that indicated they are seeking a computing job and have successfully completed the SE course. The sample is comprised of three sections of the SE course using the MACROVR approach (135 students) and four sections that did not use all of the required strategies and technologies, which we call MACROVR–lite (184 students). Our two cohorts, MACROVR and MACROVR–lite, are each given the same survey questions. We analyze their Likert scale data responses using non–parametric methods. Our findings indicate the MACROVR approach better prepares students with the skills and highly valued qualities for success in computing industry interviews.

CCS Concepts

• **Social and professional topics** → *Software engineering education.*

Keywords

Software engineering, CS education, education data mining, agile, scrum, career readiness

ACM Reference Format:
William Gregory Johnson, Raj Sunderraman, and Anu G. Bourgeois. 2020. Teaching Strategies in Software Engineering Towards Industry Interview Preparedness. In *Proceedings of Conference (CSERC).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

CSERC, October 2020, Utrecht, The Netherlands
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

According to a Collabnet/VersionOne[1] 2019 report, when developing software systems, 97% of industry uses the Agile methodology and of those, 54% employ the Scrum technique. This has been a shift from the previous waterfall model methodology. Recognizing this change, we see that many Software Engineering (SE) courses have adapted their course content focus on a more industry–relevant Agile methodology. We posit a CS student is better positioned to enter the workforce with a career in computing when the latest technologies are utilized in the classroom, in conjunction with training students to efficiently work in a team environment. In the case of a SE course, this would be Agile with Scrum, along with project based learning, using a version control system for team code sharing, managing the system through a cloud–based project management system, and incorporating key areas of team experiences in collaboration and communication. Searching through several career opportunity resources seeking SE positions, such as GlassDoor[2], Indeed[3], and ACM Career and Job Center[4], we find key areas like Git, Agile/Scrum, team problem solving, and web services being advertised as desirable skill sets and knowledge for new hires in industry. In 2017, Ford et al., concluded that technical interviewers care greatly if candidates can demonstrate CS technical skills with verbal and written clarity [14]. Our approach teaches beyond technical skills to include team presentations among peers, team collaboration and communication, and working through group consensus to deliver a software system, thus preparing them for the soft skills interview areas. Much research has looked at incorporating one or two of these components in SE courses, but do not analyze any impact upon a student's industry readiness [3, 5, 7, 10–12]. We choose to integrate Agile/Scrum components in combination with the complementary skills mentioned above and then evaluate if this approach helps prepare students for a computing job interview.

Undeniably, knowledge and experiences CS students obtain from courses in their degree program is critical to articulate their skill and value during a computing job interview. Our teaching strategy incorporates additional techniques in the SE course to aid in SE career readiness. In our course, we employ machine learning (ML) to form teams in project–based learning (PBL), require cloud technologies for project management, rotational Scrum roles, detailed rubrics, and individual and student team presentations of their project. We refer to this new approach as **MA**chine learning to select project team members, **C**loud technologies required for project,

[1] https://explore.versionone.com/state-of-agile/13th-annual-state-of-agile-report
[2] https://www.glassdoor.com/Job/jobs.htm?sc.keyword=software+engineer
[3] https://www.indeed.com/q-software-engineer-jobs.html
[4] https://jobs.acm.org/jobs/results/keyword/Entry+Level+Software+Engineer

control management, and team communications, **RO**tational schedule in Agile/Scrum roles, a final **V**ideo of the project story board, and **R**ubrics for all presentations; –MACROVR[5]. By integrating these teaching strategies and technology requirements into our SE course, students are exposed to a wide view of current practices of the computing industry.

Components in the MACROVR approach are chosen based upon one of our author's past twenty–three year industry experience as a software engineer in a major Fortune 500 US corporation and several US federal government agencies. They used many Agile/Scrum techniques in their job, managed software development teams, and conducted numerous technical and soft skill interviews for new CS graduates seeking a computing position. We concur and include researched areas and attributes as being important to computing job interviews and industry skills as found in [1, 12]. Recent, seminal work from Oguz et al., find the perspectives of industry identify recent graduates with a lack of soft skills, limited use of current tools, and little exposure to real life projects as important gaps between SE education and workforce [18]. Again, we use countermeasures in these areas with our approach through team presentations among peers, required technology, and instructor vetted projects. The CS technical skills students acquire from CS program courses like algorithmic computations, object–oriented coding principles, and system architectures are considered to be utilized by all students in any SE course, so they are not studied specifically in the MACROVR approach.

Our motivation is to evaluate if teaching real–world SE experiences through MACROVR contributes to a CS student's career readiness. We measure the effect of our strategy through a fifteen question survey[6] using a Likert scale [17]. To guide our research, we develop the following research question:

RQ1: Does teaching SE using the MACROVR approach better prepare students for a computing job interview?

To answer our research question, our survey asks CS students that finish the SE course if they feel our teaching strategies help with career readiness by preparing them for computing job interviews. This voluntary, anonymous online survey contains questions relating to specific MACROVR course components and some that are common in all SE courses taught at our university. We form two cohorts where the first is students that successfully complete the SE course requiring all components of the MACROVR approach be used (three sections, 135 students). The second cohort is students that successfully complete the SE course where the instructors may or may not use all the MACROVR approach components (four sections, 184 students). We call these cohorts MACROVR and MACROVR–lite, respectively. All survey responses result from semesters of fall 2017 through fall 2019 in the SE courses. We do not include the summer semesters due to the shorter time for delivery of the course content, only one section being offered, and typically being taught by a student, graduate teaching assistant. The semesters being surveyed in this work are taught, face–to–face by faculty instructors, usually three each fall and spring semester. All volunteers taking the survey either graduated with a four–year CS degree and were actively looking for a full–time computing job through interviews or remained enrolled while actively interviewing to secure an internship in computing. The same survey is given to both cohorts to preserve our experimental group (MACROVR) and the control group (MACROVR–lite). Since the control group is taught without requirements of all MACROVR components, the data results indicate which ones may have been used when comparing to our experimental groups' results. Our results show when the experimental group is taught with all components of the MACROVR approach, students are better prepared for a computing job interview.

The remainder of this paper is organized as follows: Section 2 discusses the background and relevant work related to our research question. Section 3 provides the MACROVR course structure, a detailed description of its system, the implementation components, and survey compositions. Section 4 interprets the response data of the survey using non–parametric methods and discusses the threats to validity. Lastly, in Section 5, we conclude the paper and discuss future work.

2 Background and Relevant Work

Much research in SE education relates to learning strategies, active teaching techniques [19], investigation on gaps between academia and the software industry [18], and authenticity of the SE course content [1]. Many report research on the effectiveness when using Project–based learning (PBL) in SE education [1, 13, 20]. Garcìa-Peñalvo et al. in 2019, research the broader area of active learning that includes PBL and concludes SE courses using PBL give students a better understanding of the concepts involved when used in the classroom [9]. Additionally in 2018, Garcìa-Holgado et al., found using similar teaching strategies increased student's performance in SE courses by 20% [15]. We adopt a similar pedagogical strategy in our research and introduce several new techniques within PBL and then measure their effect related to our goal: –CS student preparedness for computing job interviews.

One strategy researched is collaboration in SE teams [8]. Chowdhury et al., use the IBM Watson™ Personality Insights service to analyze their student teams use of a collaboration platform from Slack[7]. They find the participation of a teams' communications highly correlate to their self and team evaluations, both good and bad, as well as their team grade for each Agile/Sprint cycle in the SE course. We require the use of Slack's team collaboration platform for our SE course in the MACROVR approach.

Teaching SE while using a software version control system such as GitHub[8] has been shown to benefit students [12]. Feliciano et al., find there are numerous challenges with using GitHub as a learning platform and its wide use in industry makes it an important part of the CS student's SE education. They also expand its use beyond the team project work, utilizing it to disseminate course assignment materials and lab content with related discussions. They conclude the students benefit from using this platform and incorporate their team content as part of their personal, online presence. Rraslan et al., find many problems using Git based version control systems in

[5] /mæk 'roʊvər/

[6] IRB approval H19266.

[7] https://slack.com/features

[8] https://github.com/features

teaching SE [11]. They identify seven categories of errors and poor practices from students using Git. Their research is the most recent (2020) and comprehensive list of Git usage artifacts in the literature and gives us a new basis to consider Git as an integral part of SE education along with measurable instruments. By contrast in the MACROVR approach, we require Git usage in all project teams and require the instructor be a member of each to monitor submissions of a team's code and documentation, indicating levels of progress.

We find formation of student teams a significant factor for SE project success as reported by [2, 4, 10, 21]. Dzvonyar et al., develop and use a hierarchical criteria for team composition and through a manual selection process by instructors, form the student teams [10]. Bosnic et al., conducted a 10–year study of matching student teams to projects and find the 'first come, first serve' method or teaching staff directed were not effective [6]. They realize a system of student pre–course questionnaires and student's proposals of projects based on platform groupings (standalone, web, or mobile) as a better solution. Conversely, we form teams using several ML algorithms basing on student skills, CS aptitudes, and semester course load. We design projects requiring specific component technologies that all students enrolled in the course should possess, and teams choose a project using a lottery system. We offer details of these component technologies in Section 3.2, and sample project descriptions in Section 3.54.

Soft skills in SE education is considered important for student preparation and readiness for industry. Abad et al., analyze data gathered from SE students over three semesters and investigate the amount of authenticity that can be achieved in PBL courses in SE education [1]. Their work investigates seven 'authentic' activities with *Why?*, *What?*, and *How?* explored in each activity. Interestingly, their work indicates 'soft–skills' like *planning*, *problem understanding*, *negotiation*, and *organizational* as the most statistically significant for CS students to be prepared for industry.

Hogan et al., find CS students presenting their work in front of peers improve soft skills like communications [16]. We posit our MACROVR approach supports soft skills and amplifies them through teaching elements requiring a rotation of Scrum roles: –Project Manager, Scrum Master, Senior Programmer, Junior Programmer, QA/Testing, in each Sprint cycle. This encourages students to work on different aspects of their project for a period of time and broaden their understanding of software engineering principles. We discuss details of these in Sections 3.2 and 3.3.

Through our literature review, we are not able to identify any other research studies that consider a holistic approach in delivering the SE course, to better prepare students for the workforce. We find most research focuses on just one or a few strategies. We incorporate into our MACROVR approach numerous teaching strategies, technical requirements, and communication skills to ready a CS student for their first real–world experience, namely, a computing job interview.

3 Course Structure

The SE course using our MACROVR approach has several foundations to enable the project team formation and introduce project technology requirements. Using a variety of strategies, we organize the course to simulate a real–world environment where software engineers face many abstract ideas, must use critical thinking to understand project team goals, and must integrate/communicate with a group of unfamiliar individuals. Technology requirements in our approach include a cloud–based version control system with bug reporting, an Agile/Scrum management system, and integrated communication with an instant messaging system.

In the first two weeks using MACROVR, we introduce concepts of Agile/Scrum in lectures and give three short assessments to build the ML models for project teams formation. We discuss the technical requirements for projects like web services, a database element, and using an object oriented programming (OOP) language. The second week has team formation and team meeting time given at the end of each class session (usually fifteen or twenty minutes) for questions with the instructor, to establish meeting times for each group to interact face–to–face, and give the instructor interactions with the teams to check their project's current sprint progress. At the beginning of each sprint cycle, the instructor gives a rubric for appropriate preparation time and provides a partial structure to complete their tasks. Examples of a team end of sprint presentation rubric is found in Table 4 and an end of semester individual video of the project story board rubric is found in Table 5.

3.1 Project Team Formation

With the MACROVR approach, we form project teams by ranking the students using several ML models. Based upon a student's current course workload and three individual assessment scores we develop a heuristic for choosing students for a team as shown in Figure 1. Most class populations are close to a multiple of five, allowing five or six students per team. Typically, a MACROVR class size is fifty and we compute the total number of combinations (*n=50, r=5*), to form five–person teams as 2,118,760 choices. This computation formula is:

$$C(n,r) = \frac{n!}{r!(n-r)!}$$

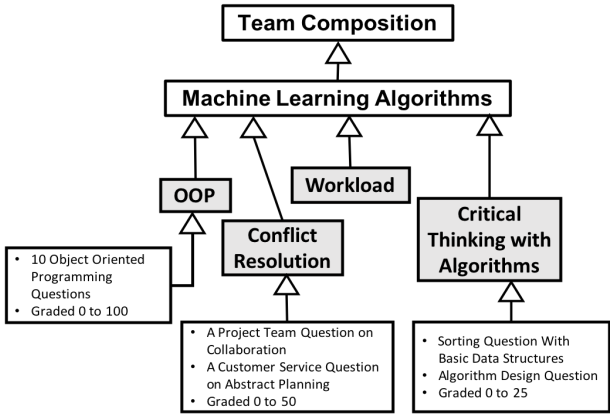

Figure 1: Assessment Structure for Team Composition

Using ML with these four attributes, we create supervised training data through ranking some students as one (novice) to five (very knowledgeable) based on their scores and course workload in contrast to [6, 10]. Using five ML classifiers: 1) Gaussian Naive Bayes,

2) K–Nearest Neighbor, 3) Decision Tree, 4) Nearest Centroid, and 5) Linear Discriminant Analysis, we take the arithmetic mean of these five rankings for grouping into classifications, one to five. We cross–validate our ML algorithms using the holdout method with a 20 / 80 split. After teams selection, we make a schedule of rotations with respect to all Scrum roles (project manager, Scrum master, senior programmer, junior programmer, QA/tester) and give teams the rubric for their first project presentation. We show a sample of one team's rotation schedule in Table 1.

Table 1: Sample Team Rotation Schedule with Scrum Roles

Team*X*	**Sprint1**	**Sprint2**	**Sprint3**	**Sprint4**	**Sprint5**
Person1	Project Mgr*	Scrum Master*	Senior Prog.	Junior Prog.	Testing and QA
Person2	Scrum Master*	Senior Prog.	Junior Prog.	Testing and QA	Project Mgr*
Person3	Senior Prog.	Junior Prog.	Testing and QA	Project Mgr*	Scrum Master*
Person4	Junior Prog.	Testing and QA	Project Mgr*	Scrum Master*	Senior Prog.
Person5	Testing and QA	Project Mgr*	Scrum Master*	Senior Prog.	Junior Prog.

* indicates a team presenter at end of sprint

3.2 Team Project Requirements

In the second week, the class instructor presents a pool of team projects with descriptions and recommendations of platforms for implementation (samples shown in Section 3.4). Each team randomly selects a sealed, envelop with a number forming a lottery selection from the list of projects. Most are web, but several are desktop or can be mobile app centric. Each must incorporate at least one web service API, and a database (SQL or No SQL) to maintain application and/or user state. The instructor gives a brief project description, possible associated web service API, and other requirements like GUI based interfaces, and recommended OOP languages. Some script languages like JavaScript are allowed, but cannot not comprise the majority of the code base.

Technology requirements for projects in our MACROVR approach include cloud–based version control management (GitHub[8]), a group communication systems (Slack[7]), and the project's progress and activity management using Scrum (ZenHub[9]). End of sprint cycle's team presentation incorporates SE course content timed with teaching progress like: conceptual models, test case scenarios with test cases, UML activity diagrams, object class diagrams, and UML state diagrams specific to their project as well as metric reporting (burn down and velocity) of their project progress from the ZenHub system.

The first sprint involves building a Scrum backlog with user stories mostly comprised of investigation and research for the project's technology requirements and setup of the team members in GitHub, ZenHub, and Slack. Additionally, the first sprint requires a global system concept model review by the instructor before their first presentation (end of sprint one) to allow constructive feedback. This is important to start each team with a realistic and correct direction for their system development and implementation in the lifetime of the course.

Grading from the rubric requires each team to upload their presentation files into our learning management system (LMS) for feedback and suggestions to improve soft skills areas for the next sprint presentation. A rubric for the team presentation is shown in Table 4. Some research indicates that collaboration, fair cooperation, and communication are the most important student skills and is a problem in teams of PBL, especially in SE education [8]. In order to mitigate some of these negative effects, the MACROVR strategy uses a sliding scale of grade impact from each end of sprint presentation. The more active roles (project manager and Scrum master) carry more responsibility, thus have the majority of earned points for the grade of a presentation. We weight the project manager at 45%, the Scrum master at 25% and other members equally at 10% each. Grading with this style of scaled points puts emphasis on active and passive roles during a sprint. It encourages team leadership and strong communications from the active roles. Referring back to Table 1, we ensure each team member will have a chance to experience all team roles in the Scrum methodology over the semester time period. The total course grading for a student is weighted as 50% for presentations with the rotating Agile roles and 50% equally split among exams.

3.3 Survey Questions

Each Likert scale question gives the participant a choice: *Strongly Agree*, *Agree*, *Neutral*, *Disagree*, *Strongly Disagree*, or *N/A*. We separate the responses for analysis in two cohorts from students in MACROVR or MACROVR–lite SE courses taught in Spring 2017, through Fall 2019. Both cohorts receive the same set of questions to allow us a statistical comparison in their opinions, interests, or perceived efficacy as it relates to the **RQ1**.

From the fifteen survey questions, we show in Table 3, there is a threat of Type 2 errors in our analysis to use a parametric testing model with the distribution of student responses being unevenly distributed. We observe several histogram graphs to verify this non–normal distribution. We choose a two non–parametric test, Kruskal–Wallis[10] to analyze our data. The general analysis of our n=319 is shown in Table 2, and in section 4 we interpret the results of our tests.

Table 2: Survey Data Composition

	MACROVR	**MACROVR–lite**
Sent	135	184
Complete	87	29
Incomplete	18	1
Refusal	5	3
Participation	64.4%	15.8%

The percentage of participation from our cohorts, shown in Table 2, indicate instructors of MACROVR–lite sections of the course may not have encouraged their students to respond to the survey

[9] https://www.zenhub.com/product

[10] https://www.statisticshowto.datasciencecentral.com/kruskal-wallis/

with similar enthusiasm as instructors of the MACROVR sections. We present the data separately as percentages based upon each respective cohort's participation shown in Figures 2 and 3.

Table 3: Survey Questions

No.	Survey Question
(1)	The use of team projects better prepared me for interviews.
(2)	The use of Agile and Scrum methodologies in class better prepared me for interviews.
(3)	The single and final project presentation in class better prepared me for interviews.
(4)	The topics covered in class better prepared me for interviews.
(5)	The single and final project document better prepared me for interviews.
(6) †	The Agile SCRUM team roles being rotated better prepared me for interviews.
(7) †	Having end–of–sprint presentations better prepared me for interviews.
(8) †	Using cloud based project and control management tools better prepared me for interviews.
(9) †	The in–class team activities related to Agile and Scrum better prepared me for interviews.
(10)	The in–class team activities related to course topics (UML, test cases, state models) better prepared me for interviews.
(11) †	Using a web API (remote) better prepared me for interviews.
(12) †	Incorporating a database system into my project better prepared me for interviews.
(13) †	Incorporating a cloud based system for project team communications better prepared me for interviews.
(14) †	Incorporating a GIT / repository system better prepared me for interviews.
(15) †	Creating a video of the project storyboard better prepared me for interviews.

† MACROVR specific survey question.

3.4 Sample Projects

Several successfully completed applications by student project teams are given from the instructor list below. We use these and other projects in the MACROVR approach. Note that some require multiple teams communicating and coordinating project resources among two or three teams. This requires a small amount of product management from the instructor to help with setup and configuration as a 'monitoring' member of their Slack, GitHub, and ZenHub systems. Additionally, the instructor helps the multiple team projects with the abstract concept of a 'test harness interface' to initially connect different applications before full implementation.

Our sample list of team projects include:

- **Attend-In:** Mobile (Android) application to use geo–location from student's mobile device's IP address for time and attendance in college classrooms. It is designed so an instructor can setup their classes from a web browser or desktop application, class times, and campus buildings. The application will convert building address to geo–location. The web services recommended is IPStack[11]. This project has two teams requiring combined work. One for the Android application built using Android Studio, one for an instructor application with back end database in a web hosted system like Google Firebase[12].
- **Client Care:** A desktop or web browser application to help a non-profit organization with client intake and management for providing services. One team to build a system to register a client's information and show if already registered for an available service. A second team for back end database and GUI to manage the services. This non–profit might offer services to include a food pantry, educational class for: 1) personal finance, 2)resume writing, 3) job interviewing and 4) MS Office training, and a clothing closet for professional and business clothing. Recommended web service would be screen scraping for food banks data[13] or interview skills on Udemy[14] to give case workers a resource in the application.
- **Baby Buddy:** Mobile (Android) application to use near field communications, (NFC) with a RFID tag to log your baby or pet as an occupant of your vehicle. It detects movement with accelerometers once baby registers as 'checked in' and when stopped. If Baby–Buddy is not checked out through the NFC tag and vehicle is stopped after a time limit, alarms are sent to registered contacts and 911 at the central system. This is all potentially hosted from Amazon Web Services. The same database will be used for check–in and check–out of baby's NFC tag. Recommended web services are Google GPS on the mobile application and Amazon web services for texting and database. This project has two teams requiring combined work. One for the Android application using Android Studio and a second for the back end database and communication to the web services.
- **Food Oasis:** Mobile (Android) application to find small popup or unknown food sources offering quality items in food deserts and swamps. A good example is at Georgia Food Oasis in FaceBook, but it requires a FaceBook account and uses your personal data and computer GPS location data in undisclosed ways. The design is a small footprint of user information for finding good, fresh food sources with web services of Google GPS and Firebase. This project has two teams requiring combined work. One for mobile app and a second for the back end database and communication to the food sources. The back end system will need a GUI to register and maintain food sources (vendor data).

[11] https://www.gps-coordinates.net/gps-coordinates-converter
[12] https://firebase.google.com/docs/hosting
[13] https://www.feedingamerica.org
[14] https://www.udemy.com

Table 4: End of Sprint Team Presentation Rubric Used in Our MACROVR Approach

Points	Timing	Content	Demonstrate Product Knowledge	Architecture and Interfaces	Skill and Look of Presentation
Level 4 20 points	Effectively use between 6 and 8 minutes. PM and SM are equally presenting content.	**Discuss These:** • Burn down chart. • Velocity chart. • Completed Scrum stories. (Not necessary for support tasks.) • Backlog stories and tasks (If none, why?).	PM can freely describe the overall purpose of the system and SM can do the same for technology and/or integration of the system.	**Architecture Pattern:** • Declare why you choose a particular pattern. • Show a clear and detail drawing (by hand) of your pattern. **Interface Model(s):** • Clear and readable interface objects (drawn by hand). • Brief explanation of where used in the system.	• Clear, concise with an appropriate amount of text. • Ease of answering questions. • Honesty of answering questions or saying you do not know the answer. (If you do not know, be honest and say so.) • You do not look as the screen when talking. (Talk to audience.) • No use of animation. • No blue or green text. • Readable photo / screen shots (Big enough and clear if hand drawn.)
Level 3 17 points	Use between 6 and 8 minutes. PM and SM are **NOT** equally presenting content.	**Discuss These:** • Burn down chart. (If none, why?) • Velocity chart. (If none, why?) • Completed Scrum stories. (Not necessary for support tasks.)	You can describe the overall product and challenges without reading the screen text verbatim.	**Architecture Pattern:** • Show a clear and detail drawing (by hand) of your pattern. **Interface Model(s):** • Clear and readable interface objects (drawn by hand).	• Clear, concise with an appropriate amount of text. • Ease of answering questions. • Honesty of answering questions or saying you do not know the answer. (If you do not know, then be honest and say so.) • You read text verbatim from the slides. (Talking to audience.) • No use of animation. • No blue or green text. • Unreadable photo / screen shots.
Level 2 10 points	Use between 4 and 6 minutes. PM and SM **ARE** equally presenting content.	**Discuss These:** • What is the product? • What does it do? • Is it mobile app or other? • Completed Scrum stories. (Not necessary for support tasks.)	You are reading the screen text while facing away from the audience.	**Architecture Pattern:** • Declare why you choose a particular pattern. • Show a clear and detailed drawing (by hand) of your pattern.	• Slides have too much text. • Ease of answering questions. • Looking at the screen when talking. (Does not apply to writing on the white board.) • You read text verbatim from the slides. (Talking to audience.) • Use of animation. • Use of blue or green text. • Unreadable photo / screen shots.
Level 1 6 points	Use between 4 and 6 minutes. PM and SM **ARE NOT** equally presenting content.	**Discuss These:** • What is the product? • Completed Scrum stories. (Not necessary for support tasks.)	You appear lost and not knowing what is going on in your project.	**InterfaceModel(s):** • Clear and readable interface objects (by hand). • Brief explanation of where used in the system.	• Slides have too much text. • Looking at the screen when talking and reading the slides. (Does not apply to writing on the white board.) • Use of animation. • Using many different colors or hard to read like blue or green. • Unreadable photo / screen shots.

Table 5: End of Semester Video Presentation Rubric for MACROVR Approach

Points	Timing	Content	Demonstrate Product Clearly and Coherently	Skill and Look of Demonstration Video
Level 4 25 points	Effectively use between 10 and 15 minutes presenting your project and product.	• What is the product? • What is the audience for the product? (Targeted users.) • What does it do? (Or supposed to do?) • What are the technologies you used to build the product? • What makes it worthy of being built? (Market importance.)	• You can freely show all the proposed and completed features of your product. (Think test scenarios of all the stories that are completed.) • The demonstration is uniquely yours. (Not a voice over with showing someone else doing the demonstration.) • The demonstration matches (actions, reactions) what you are saying as it is recorded. • The quality is such that your picture is always shown and sized to not block the screen actions or slides as it is recorded. • The demonstration is well done where there are no technical problems with the product, you don't REPEATEDLY say filler words (umh, like you know, as you see here).	• Clear speaking with showing high comfort levels using the software. • No pauses and no bugs recorded in the demonstration. • One continuous recording, not segments put together. • Sound is understandable and lighting is good. • Your skill of demonstration is well rehearsed and you show confidence in the video.
Level 3 18 points	Effectively use between 8 and 10 minutes presenting your project and product.	• What is the product? • What is the audience for the product? (Targeted users.) • What does it do? (Or supposed to do?) • What are the technologies you used to build the product?	• You can freely show all the proposed and completed features of your product. (Think test scenarios of all the stories that are completed.) • The demonstration is uniquely yours. (Not a voice over with showing someone else doing the demonstration.) • The demonstration matches (actions, reactions) what you are saying as it is recorded. • The quality is such that your picture is always shown and sized to not block the screen actions or slides as it is recorded.	• Clear speaking with showing high comfort levels using the software. • No pauses and no bugs recorded in the demonstration. • One continuous recording, not segments put together. • Sound is understandable and lighting is good.
Level 2 10 points	Effectively use between 6 and 8 minutes presenting your project and product.	• What is the product? • What is the audience for the product? (Targeted users.) • What does it do? (Or supposed to do?)	• You can freely show all the proposed and completed features of your product. (Think test scenarios of all the stories that are completed.) • The demonstration is uniquely yours. (Not a voice over with showing someone else doing the demonstration.) • The demonstration matches (actions, reactions) what you are saying as it is recorded.	• Clear speaking with showing high comfort levels using the software. • No pauses and no bugs recorded in the demonstration. • Sound is understandable and lighting is good.
Level 1 6 points	Effectively use between 6.5 and 4.5 minutes presenting your project and product.	• What is the product? • What is the audience for the product? (Targeted users.)	• You can freely show all the proposed and completed features of your product. (Think test scenarios of all the stories that are completed.) • The demonstration is uniquely yours. (Not a voice over with showing someone else doing the demonstration.)	• Clear speaking with showing high comfort levels using the software. • No pauses and no bugs recorded in the demonstration.

4 Results

In this section, we present our statistical analysis, show our descriptive and inferential data, and discuss threats to validity. The composition of data from our cohorts' responses are shown in Table 2. In Table 6, we present the descriptive analysis of our survey responses. In Table 7, we present the inferential analysis, indicate the survey question's correlation to our **RQ1** and show the most significant indicators. Lastly, we discuss threats to the validity of our findings, both internal and external.

4.1 Interpretation of Results

The highest mean score, and closest to *Strongly Agree* response, are questions 2, 12, and 14; –two requirements in the MACROVR approach. Question 2 indicates a high mean score and if we look at its Likert scale scores in Figure 2, the MACROVR cohort shows there are 0% *NA* responses and the third highest, 55.4% *Strongly Agree* in the entire corpus. The highest *Strongly Agree* response, 59.3% is in the MACROVR cohort indicating use of a database component in the MACROVR approach realizes a benefit in computing job interviews. This high mean value for question 12, could result from students taking a database fundamentals course prior to the SE course being studied. Inferential analysis of question 11, using a Web API, in both cohorts indicate highly significant (Table 7), but descriptively, the MACROVR approach shows as a much higher correlation; –*Strongly Agree*, 36.4% and *Agree,* 31.8% in contrast to MACROVR–lite *Strongly Agree*, 14.8% and *Agree*, 22.2% respectively. Simple descriptive analysis supports our **RQ1** showing the MACROVR approach responses (x=87) is three times the amount of the MACROVR–lite approach responses (x=29).

Table 6: Survey Question Descriptive Statistics

No.	Obs	Mean	Std.Dev.	Min	Max
(1)	111	3.982	1.206	1	5
(2)	107	4.093	1.178	1	5
(3)	109	3.817	1.270	1	5
(4)	111	3.901	1.206	1	5
(5)	108	3.593	1.215	1	5
(6) †	108	3.796	1.372	1	5
(7) †	108	3.935	1.162	1	5
(8) †	99	3.929	1.197	1	5
(9) †	98	3.755	1.293	1	5
(10)	108	3.954	1.171	1	5
(11) †	98	3.878	1.204	1	5
(12) †	109	4.312	0.959	1	5
(13) †	100	3.890	1.180	1	5
(14) †	109	4.156	1.148	1	5
(15) †	93	3.419	1.362	1	5

† MACROVR specific survey question.

An indicator supporting the use of the video component might simply be an outlier in the MACROVR–lite cohort. Looking at the results of survey question 15, as shown in Figure 3, we see this cohort gives 25.9% of *NA* choice, the highest over all other questions. We also see they give 14.8% *Disagree* and 22.2% *Strongly Disagree* for the same question. Compared to the MACROVR cohort for the video component question, the *NA* is 0%, *Disagree* is 8.7%, and *Strongly Disagree* is 4.3%. These responses indicate the individual video component has a stronger connection to students in the MACROVR approach. Another indicator supporting the MACROVR approach versus MACROVR–lite is question 14. The support for GIT/repository in both cohorts show the MACROVR cohort has a combined support (*Strongly Agree* with *Agree*) of 72% versus 55.5% of similar support in the MACROVR–lite cohort. This indicates both cohorts use GIT/repository, but our control group, the MACROVR cohort verify it better prepares them for computing job interviews. Our holistic approach, combining all MACROVR components, shows advantage over using only some of them. We normalize all percentages shown in Figures 2 and 3 relative to the respective cohort's response sizes.

The Wilcoxon rank–sum (Mann-Whitney) test indicates several of our survey questions reject the H_0 hypothesis. In Table 7, we see the same questions: 1, 3, 4, 5 ,6, 7, 8, 9, 10, 11, and 14, reject the H_0 hypothesis. This indicates the findings in our survey responses for these questions show significant acceptance of the alternative hypothesis, H_a, and their Likert data substantiate our **RQ1** that the MACROVR approach is reporting as more effective in preparing students for computing job interviews.

Table 7: Wilcoxon Rank–Sum (Mann-Whitney) Test

No.	MACROVR Obs	MACROVR–lite Obs	p-value	z score
(1)	85	26	0.0009^{***}	-3.309
(2)	83	24	0.0133^{*}	-2.476
(3)	82	27	0.0070^{**}	-2.695
(4)	84	27	0.0039^{**}	-2.885
(5)	81	27	0.0156^{*}	-2.418
(6) †	82	26	0.0001^{***}	-3.824
(7) †	83	25	0.0038^{**}	-2.891
(8) †	75	24	0.0002^{***}	-3.782
(9) †	74	24	0.0004^{***}	-3.565
(10)	82	26	0.0356^{*}	-2.102
(11) †	74	24	0.0006^{***}	-3.451
(12) †	84	25	0.3305	-0.973
(13) †	76	24	0.0627	-1.862
(14) †	82	27	0.0032^{**}	-2.948
(15) †	73	20	0.1090	-1.603

$^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$
† MACROVR specific survey question.

4.2 Threats to Validity

Although our data collection is sourced from multiple semesters, our research treats them as one group. We acknowledge the validity of new discovery in the MACROVR approach requires maturation and a longitudinal study as is the case in most research. Revisiting this survey in future CS student cohorts, refining the components of MACROVR, without losing the intention and relevance in the computing industry can mitigate this threat. However, our study is

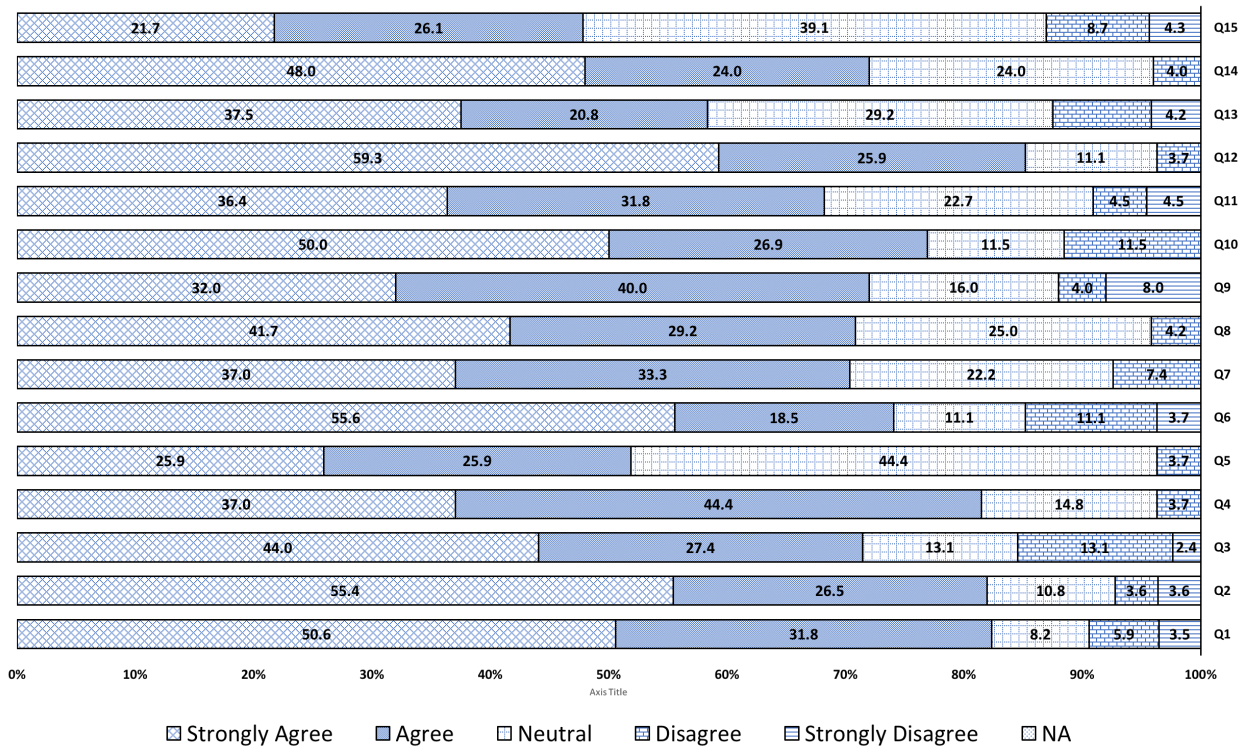

Figure 2: Percentages of MACROVR Survey Responses by Likert Value

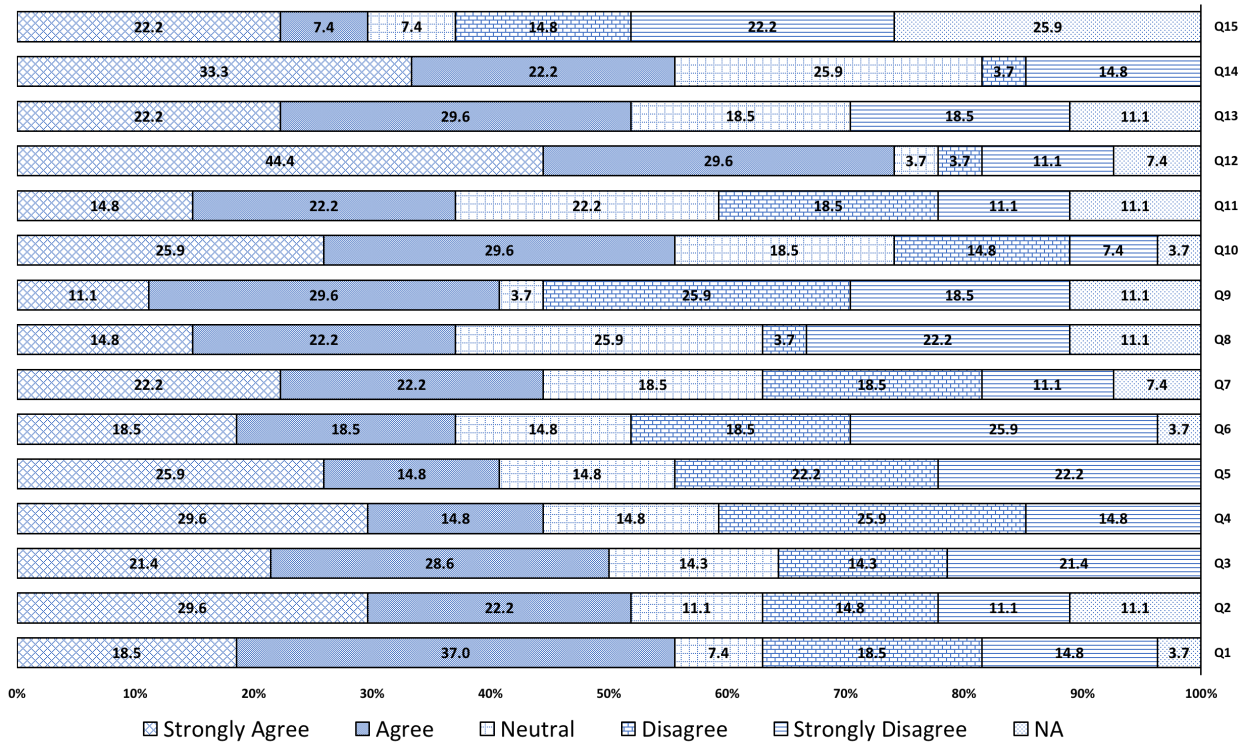

Figure 3: Percentages of MACROVR–lite Survey Responses by Likert Value

to determine if using the MACROVR approach better prepares students for interviews, as compared to a MACROVR-lite approach. Establishing benefits of the MACROVR approach opens investigation into new methods to optimize the method and delivery. Another threat is pre–test and post–test analysis. This research conducts only post–test, as students require exposure to the MACROVR approach in order to form opinion of its efficacy in preparedness in a computing job interview. One mitigation 'pre–test' would involve mock interview questions with specific relations to components of the MACROVR approach, collection of performance data in the course, and inferential analysis.

Extraneous variables such as the details of the computing job focus as well as the practical industry conducting the interview is not in this research. We do not ask specifically in the survey what are the details and duties of the SE job like building cybersecurity components versus web UI components. Also we do not ask from what industry the interview originates like healthcare versus communications versus gaming. Our assumptions that all computing job interviews relate to SE positions in similar ways threaten the external validity or our findings. Mitigation of this area would involve future surveys to include groupings of computing job details and industry as demographic data to be collected and thus, creating an internal threat to longitudinal research with this original data.

The study of SE education effectiveness in industry is a vast challenge, but analysis of teaching specifics in the application of the MACROVR approach shows help in minimizing the scope in this challenge. This case study supports preparing CS students for industry by using several techniques and technologies from the MACROVR approach when teaching SE. We see promise in our measures to strengthen the validity of our findings by continuing to refine and compare SE education practice and computing industry readiness. The computing industry and workforce readiness is a moving target, but with identifying threats to validity, we increase awareness of this change and the need to constantly examine, collect, and research SE education.

5 Conclusions and Future Work

In this study we analyze a novel approach to teaching Software Engineering (SE) that integrates numerous researched techniques for industry readiness in SE education. Specifically, we incorporate Agile/Scrum, team formation in project–based learning (PBL), Git version control, and team communication and collaboration. Combining these researched and proven strategies as effective in the computing workforce with our techniques of AI team formation, rotation of team roles, rubrics, and video reporting, we present the MACROVR approach. We report upon Likert scale data responses from a fifteen question survey and with descriptive and inferential analysis, positively answer our research question in that the MACROVR approach better prepares a CS student for a computing job interview. Our analysis validates this research in eleven out of fifteen survey questions supporting Agile/Scrum, using machine learning to form student teams for PBL, project technologies integrated in the cloud, rotational Scrum roles, rubrics, and student video recordings; –the MACROVR approach. Within the CS student population, we identified two cohorts: 1) students being taught SE topics and concepts requiring use of all components in our MACROVR approach and, 2) students being taught SE utilizing some, but not all components, we call MACROVR–lite. From the total population of n=316, the responses give us an analysis sample size of x=116. From this sample size, we investigate the effect of using real–world SE experiences through CS technical skills and soft skills. Using teaching strategies from the MACROVR approach is improving preparedness in students for computing job interviews.

The knowledge gain from our study can translate into actionable changes to teaching strategies in SE education and other courses/programs utilizing PBL, forming student teams with machine learning, and exposing team members to a variety of roles through rotations. The data shows strong influence of some strategies in the MACROVR approach and it also shows that some strategies need improvement and evaluation to refine the teaching techniques discussed in this paper. We leave these as future work to improve the SE education for CS programs.

References

[1] Zahra Shakeri Hossein Abad, Muneera Bano, and Didar Zowghi. 2019. 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)*. IEEE, 208–219.

[2] Shoaib Akbar, Edward Gehringer, and Zhewei Hu. 2018. Poster: Improving Formation of Student Teams: A Clustering Approach. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*. IEEE, 147–148.

[3] Craig Anslow and Frank Maurer. 2015. An experience report at teaching a group based agile software development project course. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. 500–505.

[4] Georgine Beranek, Wolfgang Zuser, and Thomas Grechenig. 2005. Functional group roles in software engineering teams. In *Proceedings of the 2005 workshop on Human and social factors of software engineering*. 1–7.

[5] Jürgen Börstler and Thomas B Hilburn. 2015. Team projects in computing education. *ACM Transactions on Computing Education (TOCE)* 15, 4 (2015), 1–5.

[6] Ivana Bosnić, Igor Čavrak, Marin Orlić, and Mario Žagar. 2013. Picking the right project: Assigning student teams in a GSD course. In *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 149–158.

[7] Rafael Chanin, Jorge Melegati, Afonso Sales, Mariana Detoni, Xiaofeng Wang, and Rafael Prikladnicki. 2019. Incorporating real projects into a software engineering undergraduate curriculum. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 250–251.

[8] Shuddha Chowdhury, Charles Walter, and Rose Gamble. 2018. Toward Increasing Collaboration Awareness in Software Engineering Teams. In *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[9] A Dominguez, Hugo Alarcón, and F J Garcia-Peñalvo. 2019. Active learning experiences in Engineering Education. (2019).

[10] Dora Dzvonyar, Lukas Alperowitz, Dominic Henze, and Bernd Bruegge. 2018. Team composition in software engineering project courses. In *2018 IEEE/ACM International Workshop on Software Engineering Education for Millennials (SEEM)*. IEEE, 16–23.

[11] Sukru Eraslan, Julio César Cortés Rios, Kamilla Kopec-Harding, Suzanne M Embury, Caroline Jay, Christopher Page, and Robert Haines. 2020. Errors and Poor Practices of Software Engineering Students in Using Git. In *Proceedings of the 4th Conference on Computing Education Practice 2020*. 1–4.

[12] Joseph Feliciano, Margaret-Anne Storey, and Alexey Zagalsky. 2016. Student experiences using GitHub in software engineering courses: a case study. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 422–431.

[13] Maria Lydia Fioravanti, Bruno Sena, Leo Natan Paschoal, La´ R Silva, Ana P Allian, Elisa Y Nakagawa, Simone R S Souza, Seiji Isotani, and Ellen F Barbosa. 2018. Integrating project based learning and project management for software engineering teaching: An experience report. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 806–811.

[14] Denae Ford, Titus Barik, Leslie Rand-Pickett, and Chris Parnin. 2017. The tech-talk balance: what technical interviewers expect from technical candidates. In *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 43–48.

[15] Alicia Garcia-Holgado, Francisco J Garcia-Peñalvo, and Maria José Rodriguez-Conde. 2018. Pilot experience applying an active learning methodology in a Software Engineering classroom. In *2018 IEEE Global Engineering Education*

Conference (EDUCON). IEEE, 940–947.
[16] James M Hogan and Richard Thomas. 2005. Developing the software engineering team. In *Proceedings of the 7th Australasian conference on Computing education-Volume 42*. Australian Computer Society, Inc., 203–210.
[17] Saul McLeod. 2008. Likert Scale. https://www.simplypsychology.org/likert-scale.html
[18] Damla Oguz and Kaya Oguz. 2019. Perspectives on the Gap Between the Software Industry and the Software Engineering Education. *IEEE Access* 7 (2019), 117527–117543.
[19] Williamson Silva, Igor Steinmacher, and Tayana Conte. 2019. Students' and instructors' perceptions of five different active learning strategies used to teach software modeling. *IEEE Access* 7 (2019), 184063–184077.
[20] Maur´Souza, Renata Moreira, and Eduardo Figueiredo. 2019. Students Perception on the use of Project-Based Learning in Software Engineering Education. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*. 537–546.
[21] Roderick van Cann. 2012. Optimal Team Composition in Distributed Software Development. In *Collaboration in Outsourcing*. Springer, 160–182.