Investigating Effects of Scrum Practices on Student Performance

Dr. Igor Stanojev, University of Wisconsin - Platteville

Dr. Igor Stanojev is an Associate Professor at the Electrical and Computer Engineering Department at UW-Platteville. His interests are in networking, wireless communications, and digital design.

Dr. Xiaoguang Ma, University of Wisconsin - Platteville

Xiaoguang Ma, Ph.D. is an Associate Professor in Electrical & Computer Engineering at the University of Wisconsin-Platteville. Dr. Ma has over a decade of experience in teaching, research, and industry. Specializing in agile methodologies, embedded systems, and industrial communication networks, he brings a unique blend of academic and practical expertise to his research and teaching. Dr. Ma is a Certified ScrumMaster® and has pioneered the use of Scrum practices in engineering education, creating innovative curriculum models such as the ”Tiered Educational Scrum Model” and ”Mini Scrum” for student-centered project-based learning. His work aims to cultivate an entrepreneurial mindset among engineering students through active learning approaches. Dr. Ma has also authored multiple publications on integrating agile practices into engineering education, presented at major conferences such as IEEE Frontiers in Education and ASEE Annual Exposition.

Dr. Hynek Boril, University of Wisconsin-Platteville

Dr. Hynek Boril is an Associate Professor at the Electrical and Computer Engineering Department at UW-Platteville and teaches courses in Computer Engineering, Electrical Engineering, and Computer Science programs. His research interests are in signal processing and machine learning for speech technologies and natural language processing, and in improvement of learning in engineering programs.

Investigating Effects of Scrum Practices on Student Performance

Abstract

In this paper, we investigate the effects of applying Scrum practices–a lightweight Agile framework that emphasizes iterative development, collaboration, and adaptability–on student performance in a Computer Engineering course. Two student cohorts are compared: one engaged with Scrum techniques (Scrum cohort) and another one without this experience (Vanilla cohort). Performance metrics include assignment grades, the quality and quantity of the incremental product improvements completed during the final project, which was central to mastering most of the Scrum techniques, and the Expected Learning Outcome (ELO) survey results completed by the students at the end of the semester. Our findings indicate that, in addition to the known benefits of learning and practicing Scrum, involvement in these techniques led to improved student performance. Notably, we find that i) the Scrum cohort was much more inclined to implementing additional high-quality product increments for extra credit, even though they had less need for extra credit, grade-wise, compared to the Vanilla cohort, and ii) the Scrum cohort outperformed the Vanilla cohort in test topics practiced in the Scrum-based activities.

The observed positive outcomes align well with and extend the numerous well-documented benefits of Scrum. These benefits–including self-organization, iterative development, and flexibility–typically lead to more incremental and faster product delivery. In our study, Scrum principles were first introduced through lectures, contextualizing the principles for an academic setting. This was followed by a laboratory project, where students focused on creating *user stories*–short and simple product feature descriptions written from the user's perspective–for the product they would later develop in the final project. The final phase of the study leverages the final project, conducted within the Scrum framework, where user stories are implemented as product increments.

Introduction

Agile methodologies are transforming how today's products are developed, delivered and updated, particularly in dynamic and innovation-driven industries [1, 2]. In contrast to Waterfall methodologies, a traditional sequential project management approach where each phase must be completed before moving on to the next, Agile methodologies prioritize incremental and iterative development, flexibility, and collaboration, allowing for adaptive responses to change [3]. Advantages intrinsic in Agile technologies include fast product delivery, enhanced satisfaction of both customers and employees, and more efficient resource allocation [4], to name just a few. The needs for the heavy documenting and locking-in requirements at a very early-stage time, and the

consequent resource demanding tasks such as product definition, schedule planning, and resource allocation, typical for conventional Waterfall methodologies, become obsolete with Agile.

Early after its inception in early 1990s, Scrum has become by far the most frequently applied Agile methodology [5]. In Scrum, product development is implemented through time-constrained cycles called *sprints*, where the lessons learned by the team in one sprint are used to determine and plan the next sprint. It is a lightweight framework that emphasizes collaboration, adaptability, and continuous improvement through regular feedback and reflection.

Scrum's principles of teamwork and iterative learning can be effectively applied not only in industry but also in educational contexts [6]. Recognizing this, the Computer Engineering program offered by the Department of Electrical and Computer Engineering at the University of Wisconsin-Platteville, piloted the implementation of Scrum methodologies in an entry-level course. The pilot was carried out within the framework of a larger educational research project funded by KEEN (Kern Entrepreneurial Engineering Network) [7]. The Scrum principles were addressed through i) dedicated lectures, where key Scrum concepts were introduced and contextualized for an academic setting; ii) a laboratory project, focused on designing the sprint backlog, which is made of a list of *user stories* for the final project; and iii) the final project performed within the Scrum framework.

While motivated by the potential benefits of exposing students to Scrum principles, we were equally concerned that dedicating time to these methodologies may detract from the coverage of already dense technical material in this foundational course. The study presented in this paper builds on an analysis conducted within our program at the completion of the pilot, to evaluate the above tradeoff and the feasibility of integrating Scrum into our curriculum.

In this study, two student cohorts are compared, one that was involved with Scrum techniques, here called the Scrum cohort, and another without this experience, the Vanilla cohort. As primary performance metrics, we use the assignment grades and quantity and quality of the incremental product improvements completed in the final project of the course, where most of the Scrum techniques were mastered. We find that in addition to the known benefits of learning and practicing these techniques, the involvement also brought an improved student performance. Most notably, we find that i) Scrum cohort was much more inclined to implement additional high-quality product increments for extra credit, despite being less reliant on such credit grade-wise compared to the Vanilla cohort, and ii) Scrum cohort performed markedly better compared to the Vanilla exactly in the test topics practiced using the above techniques.

Interestingly, despite the objective gains, the Expected Learning Outcome (ELO) survey results indicate that the Scrum cohort was more critical about their abilities, including those achieved while practicing Scrum methodologies. While we do not have a conclusive explanation for this phenomenon, we provide several plausible hypotheses for the underlying rationales for such results. These include a heightened students' awareness of the challenges and their knowledge gaps due to Scrum's reflective practices, cognitive load and time constraints, and reduced coverage of some technical topics to introduce Scrum principles, echoing our earlier concern about balancing Scrum methodologies with the technical content.

Through this work, we highlight the benefits and possible trade-offs of integrating Scrum into engineering education. By leveraging the iterative and reflective nature of Scrum, educators can

equip students with not only technical but also the project management skills, critical thinking, and adaptability essential for lifelong learning.

Scrum Overview

Scrum method is illustrated in Figure 1 [5]. The process relies on three principles – roles, items, and events, to support iterative product development over a series of time limited episodes–*sprints*.

The Scrum team roles are *product owner*, *Scrum master*, and the development team [5]. The product owner is the interface between the customer and the Scrum team (and the developing organization, in general) responsible for maximizing the product value. The Scrum master is responsible for the Scrum process, guiding the development team and product owner through the process. The Scrum master can be considered as an interface between the product owner, the development team, and the developing organization. The development team works collaboratively to deliver a potentially releasable product increment during a sprint [5].

The Scrum items include the *product backlog*, the *sprint backlog*, and results of the sprint work. The product backlog contains all project specifications broken to *user stories*, the concise, non-technical descriptions of a desired functionality. User stories are atomic in nature, representing indivisible units of product functionality. The user stories are sorted in the product backlog according to their priority. The sprint backlog is one or a collection of user stories taken during *sprint planning* meeting from the top of the product backlog, to be realized as a product increment during the next sprint. The sprint ends with a *sprint review* and *retrospective* meetings, during which the team evaluates the completion of the sprint backlog and scrutinizes and seeks to improve the Scrum management practices, respectively. Throughout the sprint the Scrum master keeps the team focused on the user stories and aligned with the Scrum practices. This process ends when the product backlog is empty, or the project development time expires.

User stories present accurate user's needs and should be constructed as concise definitions of a functionality to implement. They are a powerful yet simple tool that breaks down the product specifications into simple tasks, promotes team collaboration, and streamlines testing by providing clear criteria for each feature. The collection of all user stories in the product backlog completely describes the product. Constructing user stories contributes to the project planning, as the team discusses which functions to prioritize or add. User stories are also a great token for a conversation about the product, translating user requirements to the development goals.

User stories must satisfy the Scrum required format and follow the 3C principle (Card, Conversation, Confirmation). They are written on Cards. They are written as a Conversation answering three Ws: who, what, and (optional) why. Finally, the back side of the card contains a Confirmation part, the tests needed to confirm that the user story is satisfied. User stories do not contain any parts related to the development (sprint) phase.

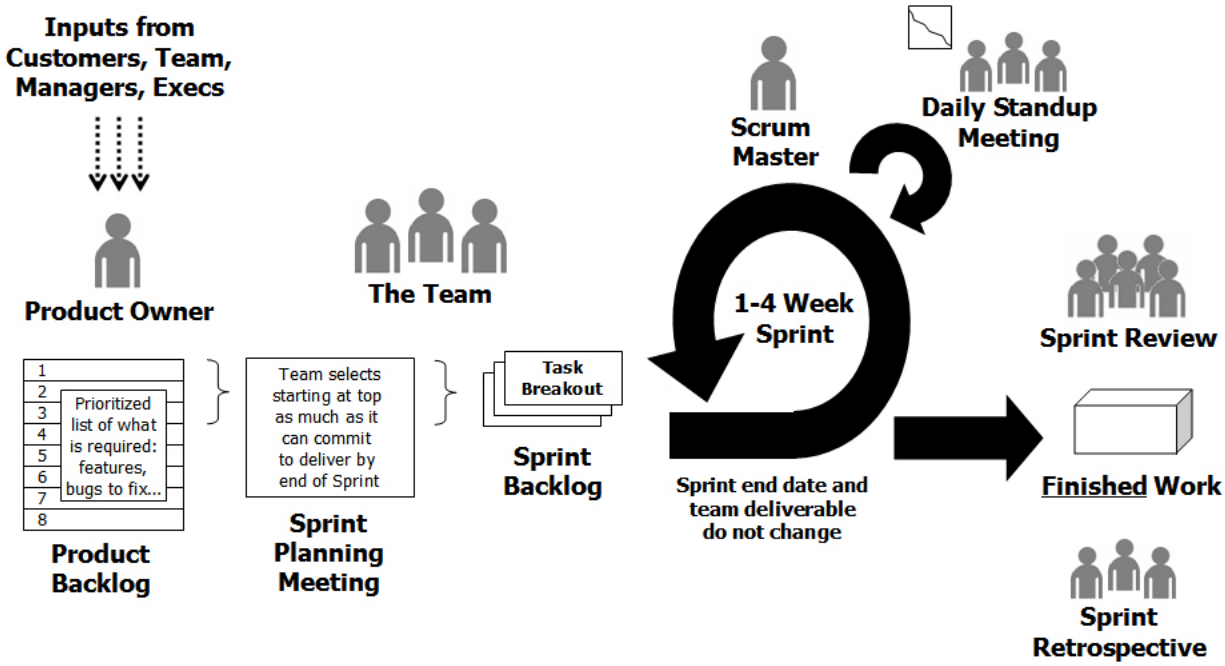

Figure 1: Scrum Process (*from Sutherland & Schwaber [5]*).

Methodology

Building on an ongoing Scrum-related KEEN project [7] in the program, the Computer Engineering program at University of Wisconsin-Platteville, within Electrical and Computer Engineering department, decided to pilot the implementation of Scrum methodologies to assess the feasibility of their implementation into our curriculum. A sophomore entry course in the program curriculum, Logic and Digital Design, is chosen for the implementation. The course instructors were trained on appropriate Scrum methodologies by a Scrum master from the program. The final project of the course was chosen as the focus of Scrum activities.

The Scrum principles were addressed through i) dedicated lectures, where key Scrum concepts were introduced and contextualized for an academic setting; ii) a laboratory project focused on designing the sprint backlog with *user stories* for the product that students will be developing in the final project; and iii) the final project performed within the Scrum framework.

Part I–Lectures

Approximately two weeks prior to the final project, a 70-minutes lecture was given on Scrum.

In the first 45 minutes, the lecture introduced Agile methodologies in a project management context, an overview of Scrum and its benefits, as well as a dive into its roles, items, events, and the Scrum process. Particular attention was given to Scrum benefits, to motivate the students to embrace the process, and the modifications needed to implement the Scrum in our academic setting. For the latter, the modifications included significantly shorter *sprints* (measured in hours) and other Scrum events. It was also stressed that the primary resource in their project will be the

available time.

The last 25 minutes of the lecture were focused on user stories, exploring the concept and benefits of breaking down product requirements into smaller, manageable tasks. A case study from a company specializing in Agile training [8] was used as a guideline for user stories' design, a task that the students would soon undertake in the upcoming laboratory project.

Part II–Laboratory Project on User Stories

A week after the lecture and a week before the final project, a laboratory project was dedicated to practicing creating the user stories. Teams of 3 to 4 students were formed. The product of interest is also the product the team will design for final project.

After choosing their product, the team's task was to break its specifications into prioritized user stories, applying the 3C principle. They also needed to propose achievable (with time as a major constraint) product improvements, i.e., new functionalities, and create the corresponding user stories. For this second part of the laboratory project, they were required to research on Internet the user reviews for the comparable products available on market and create user stories that address the pain points. Collected stories were prioritized based on their value to the product and the available resources (here, mostly time), forming the product backlog.

Part III–Final Project

Three weeks were dedicated to the final project. Students would typically meet around three to six times, including three two-hours in-laboratory sessions. The focus of the final project was on the state machine design [9], performed in Altera (Intel) Quartus CAD programmable logic device design software [10], using its Block Diagram/Schematic tool for inputting the design. The topics included a kitchen timer, a 24-hour time display, a table-tennis scoreboard, a metronome, or student-proposed topics approved by instructors. Students were allowed to use any Small-Scale Integration (SSI) and Medium-Scale Integration (MSI) devices available on market and in Quartus' libraries. Typically included devices are the ones designed in the course, like logic gates, latches and flip-flops, multiplexers, line decoders/encoders, (shift) registers, and binary counters [9, 10].

Students were expected to follow the Scrum approach, exploiting the user stories created in the preceding laboratory project, with short sprints to address them. Additional functionalities produced in the preceding laboratory project yielded extra credit if implemented as product increments. The same teams as in the laboratory project were used. All team members had to participate in the development team all times. While the Scrum roles like Scrum Master and Product Owner were not explicitly assigned, students were asked to adopt these mindsets and rotate them in the team. Students were required to reflect within a group between two sprints, mimicking backlog refining, and sprint scheduling, review and reflection events. The instructor of the course occasionally assumed one or Scrum roles, depending on the progress of the team and the direction of the adopted design.

Examples of User Stories and Product Increments

The creation of *user stories* and working in the Scrum framework generally motivated the students to complete more *increments* of the product in the final project (this is also indicated by the final project grades, as detailed in the next section). In the following, we provide user stories and product increments for the table tennis scoreboard, one of the most challenging final project products, implemented by one of the Scrum teams. Furthermore, we provide an overview of innovative user stories and those implemented as product increments by the Scrum cohort, and compare these product increments with the achievements of the Vanilla cohort.

In summary, the table tennis scoreboard should display the score in a four-digit format, with two digits for each player. The referee must be able to increment, decrease, and clear a player's score. The score board should automatically detect when one of the players wins the game (games of at least 11 points where the winner needs to overcome the opponent by a margin of at least two points) and turn on the corresponding LED to denote this player's victory.

These specifications are converted to the following user stories by the mentioned Scrum team:

1. (Card front) As a judge, I want to press a button to increase Player's score, so that Player can win the round. (Card back) When Player's increase button is pressed, Player's score increases by 1.
2. (Card front) As a judge, I want to press a button to decrease Player's score, so that Player's score can be corrected. (Card back) When Player's decrease button is pressed, Player's score decreases by 1.
3. (Card front) As a judge, I want to be able to clear points so that I can start a new match. (Card back) When clear button is pressed, both Player 1's and Player 2's scores are set back to 0.
4. (Card front) When a user has 2 or more points than opponent and has 11 or more points, then the user's win LED turns on. (Card back) As a player I want to see my win LED turn on so that I can when I have won.
5. (Card front) As a player, I want to be able to see my score so that I can see how close I am to winning. (Card back) My score display is on and showing my current score.
6. (Card front) As a player, I want to be able to see my score, so that I can see how many times I have won. (Card back) My win counter display is always on and showing my current score.

Notice how project specifications are broken into several smaller implementation tasks, each adding another functionality to the product. All tasks address the three Ws: who, what, and why. The atomicity of a task enables simple and well-defined functionality tests, as displayed in the back of the cards. For illustration, the cards for user stories 1 and 6, as created by this team in the preceding laboratory project, are shown in Figure 2.

Table 1 summarizes innovations implemented as product increments or only proposed by the Scrum teams through *user stories*. In comparison, seven teams in the Vanilla cohort implemented

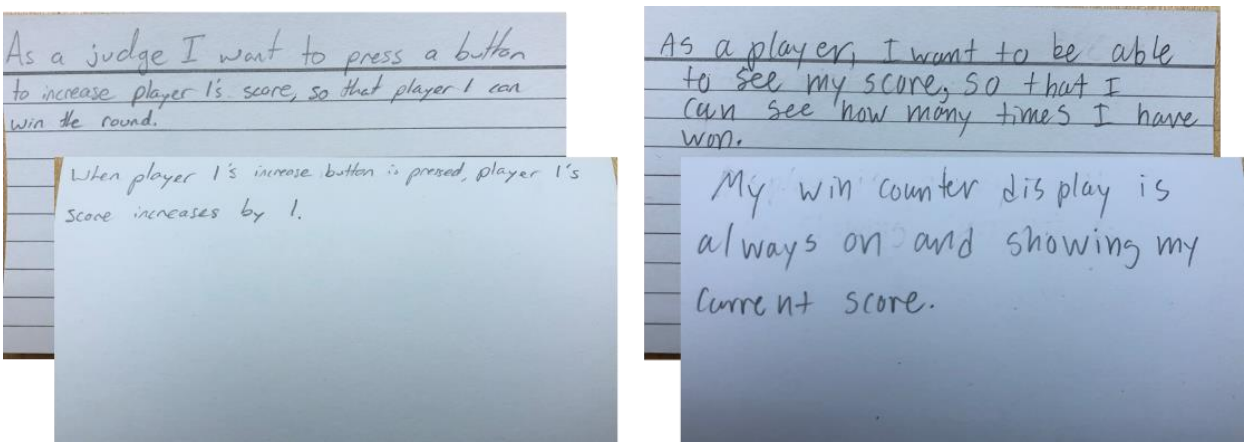

Figure 2: Examples of *user stories* (card front and back) for the table tennis scoreboard product.

only two product increments - stop alarm sound (Kitchen Timer) and reset clock (Digital 24-hour Clock).

Table 1: Implemented product increments and those that were only proposed as a *user story*. Unless stated otherwise in parentheses, a single team has implemented a particular increment.

Final Project Topic	Implemented Product Increments	User Stories Only
Kitchen Timer (2 Teams)	*Pause timer.* *Clear timer.* *Stop alarm sound.*	*Snooze.* *User sets snoozing time.*
Digital 24-Hour Clock (3 Teams)	*Pause clock (2 teams).* *Reset clock (2 teams).* *Button to decrement each digit.*	*Reset by pressing increment button long time.* *12-hour and 24-hour display modes.*
Metronome (2 Teams)	*Pause.*	*Buttons for max/min tempo.* *Preset tempo.*
Table Tennis Scoreboard (1 Team)	*11- and 21-points game mode.* *Count players' victories.* *Clear win counters.*	*N/A*

Assessment Results

To assess the success of the implementation of Scrum methods, we investigate their impact on student performance, in particular the assignment grades and quantity and quality of the incremental product improvements completed in the final project of the course, where most of the Scrum techniques were mastered. We compare the cohort involved with Scrum methods as described, herein called the Scrum cohort, with another without such opportunity, the Vanilla cohort. The Scrum and Vanilla cohorts consisted of 31 and 26 students, respectively.

In Figure 3, we show the average grades along with 95% confidence intervals for the two cohorts

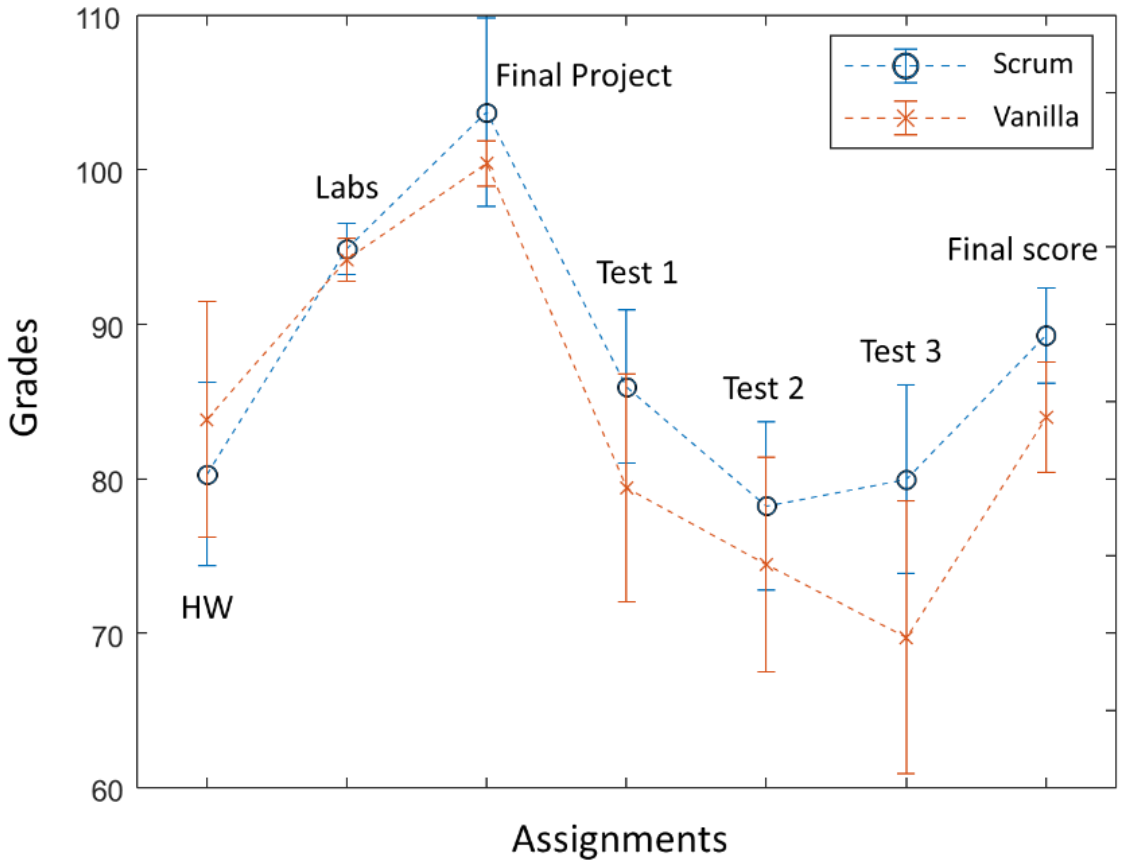

Figure 3: Average assignment grades for *Scrum* and *Vanilla* cohorts.

and the following assignments: homework (10 % of the grade), laboratory projects (15 %), final project (15 %), tests 1–3 (20 % each), and final score. Focusing on final project, notice that the Scrum cohort achieved higher grades than the Vanilla. This is due to the Scrum cohort implementing more of the innovative *user stories* as *product increments* for extra credit. In particular, eight Scrum teams (31 students) implemented 12 innovations, compared to the seven Vanilla teams (26 students) completing only 2 (see previous section for details). Interestingly, Scrum cohort was less reliant on this extra credit compared to the Vanilla cohort, as their final score grades were already higher (mainly due to the tests performance). Thus, the Scrum cohort in average worked more efficiently than Vanilla cohort, or the fact that they were involved in designing the user stories steered them to implement the stories as product increments. It is also noted that better performance of the Scrum cohort in the final project could have been expected regardless of the Scrum methodologies, given that this cohort in general outperforms the Vanilla cohort (see test assignments grades or final score in Figure 3), but perhaps not to this extent (see details in the previous section).

In addition to the final project, it is interesting to observe the test 3 average grade, as it is comprised of topics fundamental for final project. Here we are not interested in the absolute value, but rather the trends between two cohorts. After learning and practicing this material through Scrum techniques, students seem to have mastered it well enough to bounce back after a

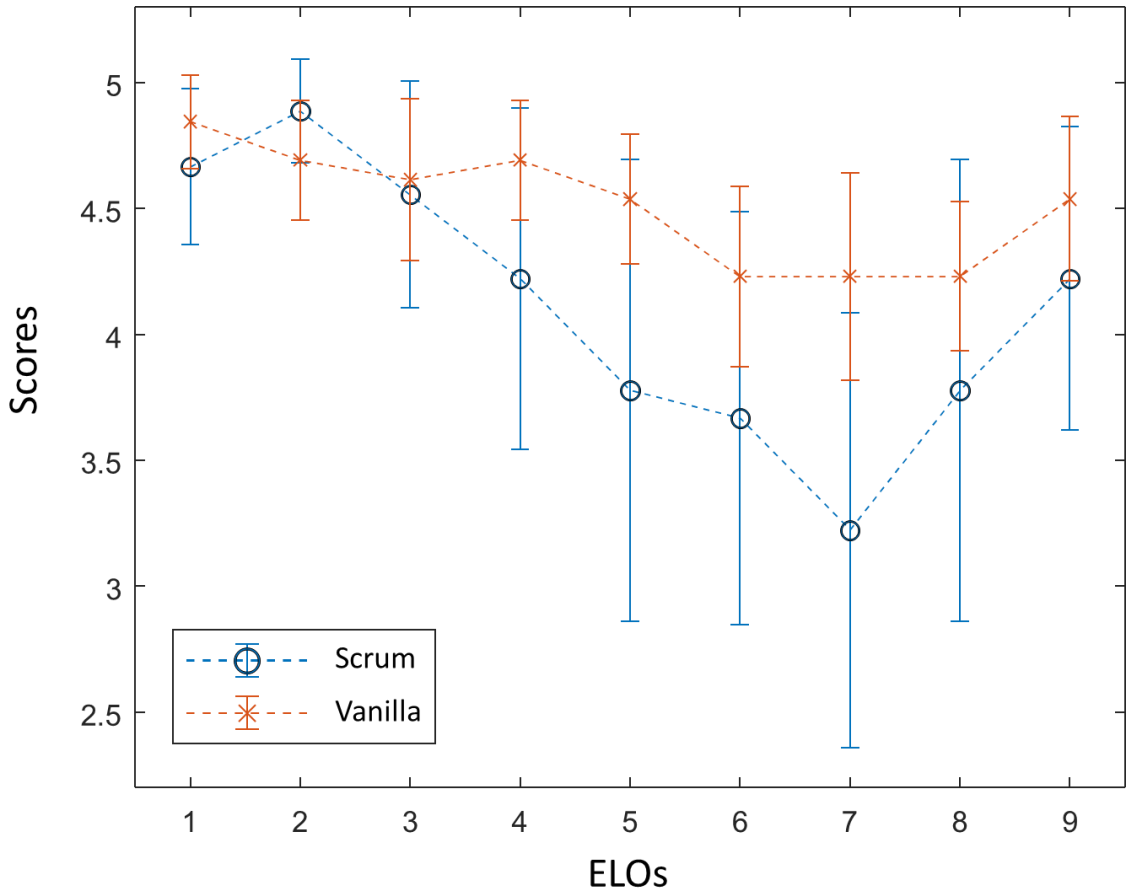

Figure 4: Average ELO scores for *Scrum* and *Vanilla* cohorts.

mediocre test 2 result. This trend was not observed in the Vanilla cohort. It is noteworthy that results on test 1 are typically high, as a large part of a cohort is comprised of Computer Science and Software Engineering majors, who already had the experience with test 1 topics, namely Boolean Algebra.

The significant score differences observed in Figure 3, particularly in the final project grades, reflect the impact of Scrum practices on student-driven learning and problem solving. By encouraging students to propose and address their own issues within the project framework, Scrum fosters a deep sense of ownership and motivation. This approach often leads to "aha" moments, such as realizing how state machines can be applied to build functionalities like clock counting, sparking a genuine eagerness to learn. Additionally, as students implement more features through hands-on experimentation, they naturally acquire a deeper understanding of the concepts. This cycle of learning by doing not only enhances their technical skills but also builds their confidence in applying knowledge to solve complex problems, resulting in higher performance compared to the Vanilla cohort.

Towards the end of each semester, our students are required to complete the Expected Learning Outcomes (ELO) survey for each course. For the course in the project, there are nine outcomes, listed in Table 2. The survey has 5-points Likert-type questions that allow students to evaluate the

Table 2: Expected Learning Outcomes (ELOs) for the course Logic and Digital Design.

Number	Expected Learning Outcome
1	*Understanding of binary and hexadecimal number systems and two's complement arithmetic.*
2	*Understanding of Boolean algebra and proficiency in the use of theorems and laws to manipulate Boolean expressions.*
3	*Understanding of digital systems, logic gates, truth tables, and combinational circuit design.*
4	*Ability to design, simplify, build, and test combinational circuits.*
5	*Ability to design and build circuits using medium-scale integration components such as multiplexer, decoder, and adder.*
6	*Understanding of flip-flops.* *Ability to derive state table and state diagram.*
7	*Ability to implement a state machine using CAD tools for schematic capture and simulation.*
8	*Ability to design a simple state machine.*
9	*Ability to write proposals, progress reports, and test reports.*

ELO accomplishment and records their answers as a grade from 1 (lowest) to 5 (highest). The results of the survey for the two cohorts, namely the average grades along with 95% confidence intervals for the nine outcomes and the two cohorts, are presented in Figure 4. Notice that in ELOs 5–8, corresponding roughly to test 2 and 3 topics, the Scrum cohort students assessed their abilities lower than the Vanilla cohort in spite of their objective performance being higher (see the scores on test 2 and 3 for Scrum versus Vanilla cohorts in Figure 3). This is an interesting phenomenon for which we do not have a conclusive explanation, but the following paragraphs present several hypotheses for what might have been the underlying causes. We note that the substantial drop for Scrum cohort in ELO 7, which refers to designing in CAD, is likely due to instructors not emphasizing that Quartus is a CAD tool (the whole design is performed in Quartus).

Scrum's Emphasis on Iterative Learning: Scrum methodologies encourage continuous improvement and iterative development. While this approach helps students achieve better results by learning through practice and implementation, it also makes them acutely aware of the challenges and gaps in their knowledge as they reflect on their work during sprint reviews and retrospectives. This heightened self-awareness may lead them to evaluate their learning outcomes more critically, resulting in lower self-evaluation scores.

Cognitive Load and Time Constraints: Introducing Scrum methodologies likely added cognitive load as students adapted to the framework while simultaneously tackling complex content. The time-intensive nature of Scrum practices and events might have led students to feel that they were unable to master all course topics, further influencing their self-evaluation. Therefore, the Scrum cohort, while achieving better project and test scores, may have felt they missed out on in-depth theoretical instruction due to the hands-on focus of their curriculum.

Reduced Coverage of Technical Topics: The time required for introducing Scrum methodologies was mostly on account of reducing the coverage of the technical topics for tests 2 and 3, and Scrum cohort were made aware of it. It is possible that this awareness and lower grades in tests 2 and 3 comparing to test 1, made them believe they missed out on some key aspects of the course. In any scenario, this is an issue that warrants careful refining of the balance between covering technical and Scrum methods.

Dunning-Kruger Effect and Student Perceptions: The Dunning-Kruger effect [11] offers an additional explanation. Rather than being on the left side of the curve (where individuals with limited knowledge overestimate their abilities), the Scrum cohort's high test scores suggest they are transitioning to the right side of the curve. Through Scrum's iterative and reflective practices, students developed a heightened awareness of their knowledge gaps and the complexities of the material. This awareness likely contributed to their lower self-evaluation scores, as they realized the depth of the subject and their ongoing learning needs. In contrast, the Vanilla cohort, without such reflective practices, may have overestimated their understanding, resulting in higher self-evaluation scores despite lower performance.

Concluding Remarks

In this paper we have described methods used in the implementation of Scrum methodologies in our entry Computer Engineering course, Logic and Digital Design. We have assessed the implementation comparing the consequent student performance with that of a cohort that was not exposed to such methods. We found that in addition to the known benefits of learning and practicing these techniques, the involvement also brought an improved student performance. In particular, the Scrum cohort was much more inclined to implement additional high-quality product increments in final project for extra credit, despite being less reliant on such credit grade-wise compared to the Vanilla cohort. We also found that Scrum cohort performed markedly better compared to the Vanilla exactly in the test topics practiced using the above techniques. Hence, our results suggest that the collaborative engagement and active learning fostered by Scrum have helped students to excel in the class.

References

[1] A. O. Adegbite, A. Adefemi, E. A. Ukpoju, A. Abatan, O. Adekoya, and B. O. Obaedo, "Innovations in project management: Trends and best practices," *Engineering Science and Technology Journal*, vol. 4, no. 6, pp. 509–532, Dec 2023.

[2] S. Al-Saqqa, S. Sawalha, and H. Abdelnabi, "Agile software development: Methodologies and trends," *International Journal of Interactive Mobile Technologies*, vol. 14, pp. 246–270, 2020.

[3] "Agile manifesto," http://agilemanifesto.org/principles.html.

[4] A. Kumar and B. Goel, "Factors influencing agile practices: A survey," *International Journal of Engineering Research and Applications*, vol. 2, no. 4, pp. 1347–1352, 2012.

[5] J. Sutherland and K. Schwaber, “The scrum papers: Nuts, bolts, and origins of an agile process,” Boston: Scrum, Inc., 2007.

[6] D. Lee, C. E. Wick, and H. Figueroa, “Applying scrum project management methods in biomedical and electrical and computer engineering capstone design courses,” in *2018 ASEE Mid-Atlantic Section Spring Conference*, no. 10.18260/1-2–29456, April 2018, https://peer.asee.org/29456.

[7] X. Ma, “Mini scrum: An innovative project for an introductory digital logic design course,” https://engineeringunleashed.com/card/2318, 2023.

[8] M. G. Software, “User stories,” https://www.mountaingoatsoftware.com/agile/user-stories.

[9] C. H. Roth and L. Kinney, *Fundamentals of Logic Design*, 2013.

[10] “Quartus prime design software,” https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/.

[11] D. Dunning and J. Kruger, “Unskilled and unaware of it: How difficulties in recognizing one’s own incompetence lead to inflated self-assessments,” *Journal of Personality and Social Psychology*, vol. 77, no. 6, pp. 1121–1134, 1999.