A Hands-on Cross-Platform Mobile Programming Approach to Teaching OOP Concepts and Design Patterns

Pınar Muyan-Özçelik
Department of Computer Science
California State University, Sacramento
Sacramento, CA
Email: pmuyan@csus.edu

***Abstract*—This study explores the learning outcomes of utilizing a hands-on cross-platform mobile programming approach for introducing two important software engineering topics, namely, object-oriented programming (OOP) concepts and design patterns. This approach presents an innovative teaching methodology that aims to help addressing unique needs and expectations of Millennials and their prospective employers with regards to software engineering education. To conduct this project, a widely adopted cross-platform mobile application development framework in the industry called Codename One (CN1) is used. This paper first presents an extensive review of literature relevant to the use of mobile programming in computer science curricula. Then, based on experience on using CN1 as a novel teaching tool, insights are provided as to why CN1 is a suitable framework for teaching OOP concepts and design patterns, and how CN1 can be utilized to effectively teach these topics using hands-on learning techniques. Recommendations for the educators who would like to adapt CN1 as a teaching tool are also provided. Reflecting Millennials' intrinsic tendency to explore emerging mobile platforms, the learning outcomes of this study show that this programming approach increases Millennials' interest towards these software engineering concepts and improves student success. In addition, feedback from students suggests that this approach provides a competitive advantage to Millennials in the job market, which increasingly demands mobile application development skills.**

***Keywords*-software engineering education; cross-platform mobile programming; object-oriented programming concepts; design patterns; hands-on learning**

I. Introduction

Providing a thorough understanding of object-oriented programming (OOP) concepts and elements of reusable object-oriented software, i.e., design patterns [1], constitutes a fundamental goal of software engineering curricula. OOP concepts and design patterns were first introduced in the mid-1960s [2] and the end of 1980s [3], respectively. Although these are established topics in the field of software engineering, they are directly applicable to contemporary and emerging paradigms such as mobile software development. This opens up exciting opportunities for making innovations in the way we introduce these topics to Millennials [4] who are considered tech-savvy and have a keen interest in new technologies. This study presents an example of one such innovative pedagogical method by introducing OOP concepts and design patterns using a hands-on cross-platform mobile programming approach.

To conduct this project, a programming environment called Codename One (CN1) [5], which is a widely adopted framework in the industry for developing cross-platform mobile applications, is used. To the best of the current author's knowledge, with the exception of one massive open online course initiative [6], this framework has not previously been used as a teaching tool in higher education. Hence, this paper presents novel insights and lessons learned about utilizing CN1 as a teaching tool in the undergraduate curriculum, specifically in software engineering education.

With a focus on improving software engineering education that we provide to Millennials, this study is motivated by three primary aspects that might be considered contributions to this field: (1) Since Millennials have a natural curiosity about mobile platforms, this project offers an opportunity to increase the students' interest and success rates in software engineering course topics, (2) Considering the growing need of prospective employers for skilled mobile software developers, the approach suggested in this study can help providing Millennials a competitive advantage in the job market, (3) By requiring students to implement large scale mobile applications, this project aims to provide experiential and hands-on learning methods for Millennials, which allows them to apply the theory they learn to practice.

This study also contributes to the developing research field that focuses on the use of mobile technology in computer science education by sharing insights about incorporating mobile programming to a software engineering course and providing an extensive survey of related studies in this area.

II. Background and Motivation

Integrating mobile technology to computer science curricula, specifically to software engineering education, has several benefits for Millennials with regards to their learning interests, diverse backgrounds, and employment prospects. Previous research shows that students are more motivated when the content of their courses relates to their interests [7]. Mobile applications attract users from a wide range of age groups; and Millennials especially enjoy and heavily utilize these applications [8]. Hence, Millennials are intrinsically motivated to learn about emerging mobile technologies, which they

978-1-5386-2795-2/17 $31.00 © 2017 IEEE
DOI 10.1109/SECM.2017.12

widely utilize in their daily lives. Students are also aware of the fact that, due to the increasing market demand for mobile applications, job openings in the mobile software development area are booming [9]. Thus, in addition to having a natural curiosity towards mobile computing topics, Millennials that aspire to be software engineers are also eager to develop skills in this area to find better jobs. Hence, they need and expect higher education institutions to train them in mobile software development area. Given these motives and needs, one can expect that this new generation of students would spend effort in mobile-programming-related projects with a drive that extends beyond fulfilling the basic requirements of a software engineering project.

In addition to having innate interest in emerging technologies, Millennials are also known to have diverse interests, different learning styles, and come from a variety of cultural and social backgrounds. Another benefit of using mobile technology is to serve these diverse needs of Millennials. Developing teaching strategies that answer the needs of this unique blend of students allows us to take advantage of the blessing of diversity in higher education institutions and create a vibrant learning environment. Utilizing mobile technology employs one such teaching strategy that aims to enhance the learning experience of this diverse student population. This is because mobile programming is an attractive topic for students from a variety of different backgrounds. With the plethora of mobile applications available, each student typically has several favorite applications that (s)he uses daily. Thus, integrating mobile programming topics to software engineering lectures allows grasping the attention of students with various backgrounds on a common platform. Hence, another benefit of utilizing mobile technology is that it enhances the learning of a diverse student population. As explained in Section IV-B, to especially address the needs of those students who learn better through experiential learning [10], the mobile programming approach can be used in combination with a pedagogical method that is based on "active doing". Active doing can be realized by letting the students gain hands-on experience with projects that require them to apply taught software engineering techniques to mobile application development.

Because of the increasing popularity of the use of mobile applications for marketing and selling business products, prospective employers also need and expect teaching institutions to graduate Millennials with a proper background in mobile software development. Hence, to address the unique needs of both the new generation of students and the their prospective employers, there is an increasing trend to integrate mobile application development techniques into computer science curriculum. These needs are also acknowledged by Accreditation Board for Engineering and Technology (ABET) which now demands this integration as a criteria for accrediting computing programs [11]. Thus other benefits of using mobile computing include providing Millennials with a competitive advantage in the job market and maintaining currency with accreditation standards.

III. Previous Work

Despite the significant benefits of utilizing mobile technology in computer science curricula, as described above, relatively little is known about the philosophy, implementation, and results of upper-division computer science courses that cover subjects related to mobile technologies and application development [12]. Previous work conducted in this area has mostly developed in the last five years and can be grouped into three main categories:

1) Studies that discuss integration of mobile computing to computer science curricula: In an extensive survey of courses which incorporate mobile computing, Burd et al. [13] explain why mobile computing belongs in the computer science curricula, and discuss how educators might adopt mobile computing in their courses. Likewise, Minaie et al. [14] discuss various approaches that are used by different colleges and universities around the world to integrate mobile computing into their computer science and engineering curricula. As an example of such curricula, Suo [15] presents a mobile computing curriculum that focuses on practical experiences including design, development, and testing of mobile applications on mobile devices. In another related study, Mahmoud [16] presents best practices for teaching mobile application development across the computing curricula by utilizing BlackBerry smartphone and CMER Academic Kit. With a different focus, Gordon [17] discusses underlying concepts that should be addressed in mobile programming courses, i.e., user interface, device cooperation, hardware issues, data handling, application interaction, and programming issues. In another domain, Kurkovsky [18] illustrates the reasons and the possibility of using mobile devices and mobile game development as a learning context and a motivational tool in the computing curriculum.

2) Studies that use cross-platform mobile programming approach in particular courses: For teaching application development in introductory and upper-level courses that focus on application design, Dickson [19] suggests utilizing a cross-platform mobile development system called Cabana. Likewise, Lutes [20] suggests an approach that uses cross-platform development tools and describes the pedagogy utilized in a course that focuses on software development for mobile devices.

3) Studies that use platform-specific mobile programming approach in particular courses: Small number of the studies in this category target iOS or Windows development. For instance, England [21] shares experience teaching a software development course featuring iPad app development. Likewise, Thomas [22] shares experience teaching an iOS app programming course using only available on-line resources and utilzing a flipped classroom style instruction. On the other hand, Sung and Samuel [12] discuss the merits of implementation versus design-based mobile application development courses that utilize Windows phone and the C# programming language.

Larger number of the studies in this category target Android development. For instance, Hollingsworth [23] describes two adjustments done to the core computer science curriculum

at one institution that require students to take a course on web programming using the Google App Engine cloud, and a mobile computing course based on Android. Likewise, Kang and Cho [24] propose a model that uses a combination of Eclipse with Android SDK and App Inventor for effectively teaching Android application development. In another study in this sub-category, Margulieux et al. [25] present how subgoal-labeled instructional materials can be employed to promote the creation of mental models when teaching novices to program in Android App Inventor. To teach Java programming and other advanced computer science topics, Riley [26] provides an approach that utilizes the Android mobile phone platform. Similarly, Yoo et al. [27] proposes ways to systematically introduce object-oriented design concepts in the "Java: Android Application Development" class.

There are also studies in this category that target different platform-specific frameworks. For instance, Goadrich and Rogers [28] provide a discussion on whether faculty should teach smartphone development in iOS or Android. On the other hand, Payne [29] presents an approach to teach both Android and iOS development in native Java and Objective-C in a single semester by building parallel example projects to shorten the students' time-to-proficiency in both environments. In an other related study, Khmelevsky and Voytenko [30] present a new paradigm for teaching mobile application development, focusing on software development and software engineering capstone projects with industrial sponsors.

This study differs from these three streams of prior work in the following ways: (1) It is one of very few studies that explore the use of a cross-platform mobile programming approach and utilizes CN1 as a teaching tool, which has not been previously explored by prior studies, (2) It focuses on teaching OOP concepts and design patterns by utilizing hands-on mobile programming techniques, a combination of specific topics and methods that has received less attention in the software engineering education literature, (3) It explicates how it aims to help addressing the needs and expectations of Millennials and their prospective employers.

IV. Methodology

This section explains why CN1 is a suitable framework for teaching OOP concepts and design patterns and how CN1 can be utilized to effectively teach these topics using hands-on learning techniques. In addition, recommendations are provided for educators who wish to adapt CN1 as a teaching tool.

A. Why CN1?

There are several different mobile application development frameworks that are popularly used in industry to develop mobile software. We can divide these frameworks into two main groups: (1) platform specific frameworks such as Android Studio [31] and iOS SDK [32], (2) cross platform frameworks such as CN1, PhoneGap [33], and Xamarin [34].

Applications developed in a platform specific framework can only run on its target devices (e.g., Android or iOS devices). On the other hand, applications developed in cross-platform frameworks can run across devices that have different operating systems (e.g., Android, iOS, and Windows).

In this study, the main reason why CN1 is selected as a teaching tool among all the available mobile programming environments is that CN1 is a cross-platform and Java-based tool. Working with a cross-platform tool allows students to get familiar with different mobile operating systems. This is important academically since the educators should give students an opportunity to explore as many alternatives as possible, rather than only focusing on one type of technology. In addition, having these alternatives further motivates Millennials towards learning course topics since they are known to be technically diverse and keenly interested in learning different technologies.

Utilizing cross-platform mobile programming has advantages also for prospective employers since companies usually need to release their mobile applications in many popular marketplaces (e.g., Apple's App Store, Google's Play Store) to expand their customer base. Hence, recruiting employees that are experienced in cross-platform mobile application development might help reducing a company's development time and cost.

Among the popular cross-platform mobile programming environments, CN1 is chosen at the institution this study is conducted, i.e., at California State University, Sacramento (CSUS), since it is the only cross-platform framework which is Java-based [35]. Since CN1 is Java-based and Java is a one of the most prominent OOP languages, CN1 is very suitable for teaching OOP concepts and the elements of reusable object-oriented software, i.e., design patterns. Some of the other cross-platform mobile programming frameworks are also based on object-oriented languages (e.g., Xamarin is based on C#). However, CN1 is the best option for CSUS, since prerequisites of the software engineering course that introduces OOP concepts and design patterns are offered in Java. In addition, the electives for which this software engineering course is a prerequisite are also offered in Java. Hence, CN1 provides a smooth transition for the students throughout the CSUS undergraduate computer science curriculum.

The above-mentioned platform-specific mobile programming frameworks are also based on OOP languages (i.e., Android Studio uses Java and iOS SDK uses Objective-C). Hence, the teaching techniques that are discussed in this study are transferrable to alternative OOP-language-based mobile application development frameworks, which may be either platform-specific or cross-platform. For this reason, other higher education institutions are not required to use CN1 to utilize the techniques presented in this paper. If some other OOP-language-based mobile programming environment is more suitable for a particular teaching institution, the teaching methodology presented in this project can be adapted to the needs of that institution.

Like other popular mobile programming frameworks CN1 is bundled with a device simulator that allows developers to implement and test their applications without being required

to have a physical device. The device simulator of CN1 is very fast and allows developers to quickly switch between, and do testing on, various different skins (e.g., iPad, iPhone, Nexus, Samsung, HTC, Nokia Lumia, Xoom, Blackberry, etc.).

Other advantages of CN1 are that it is free and open source. The base version of CN1, which is free to use for all developers, including students and professional programmers, is designed so that it is sufficient for the needs of academic work and there is no need for obtaining academic licenses. CN1 includes some pro-only features that require licensing. However, these are optional and teaching OOP concepts and design patterns does not require the use of these features. CN1 is also an open source framework. 90% of the source code of CN1 framework is available online and CN1 developers claim that this is one of the biggest advantages of CN1 over other mobile solutions [36]. Hence, developers can examine, change, and contribute to the actual source code of CN1. This is advantageous for enhancing the learning of Millennials that aspire to be software engineers. Interested students can access this software repository and examine how different software engineering concepts are applied to a large-scale real-world software project.

CN1 is installed as a plugin to the three popular Integrated Development Environments (IDEs). Hence, while catering to the unique needs of the Millennials, this approach also capitalizes on the strengths of Millennials. This is because learning OOP concepts and design patterns utilizing this mobile development environment requires students to use new programming tools such as an IDE that they might not have utilized before, as well as the CN1 framework, which is installed as a plugin to this IDE. However, since Millenials are also known to be tech-savvy and technically diverse, the requirement of utilizing these new tools does not provide a drawback for their success. On the contrary, it would motivate them further towards course topics by allowing them to further diversify their technical background.

B. Hands-on Learning Techniques

At CSUS, hands-on learning techniques are used to teach OOP concepts and design patterns. As mentioned in Section II, a pedagogical method based on active doing is used which especially addresses the needs of those students who learn better through experiential learning. Active doing is realized by letting the students gain hands-on experience with projects that require them to apply the theory that they learn in class to practice. Specifically, students are required to prepare large-scale projects, which ask them to apply the learned OOP concepts and design patterns to the implementation of mobile applications.

Prior to the introduction of this study, OOP concepts and design patterns were taught using Java Standard Edition (SE), where students were asked to develop desktop applications in their projects. To conduct this study, the course is redesigned by updating the course content and modifying projects that were originally developed mainly by colleagues who previously taught these topics [37]. The updated projects ask students to use CN1 in their solutions and provide tips on the usage of CN1 Application Programming Interface (API) for solving particular problems included in their projects.

The projects are cumulative in nature and ask students to gradually develop a two-dimensional mobile game (e.g. snake, breakout, xonix, etc.). In the first project the game has a primitive text-based interface, whereas, in the last project, the application becomes a fully functioning game with graphical interactive interface where game objects are being animated on screen. In total, four projects are given. Table I provides detailed information about each project. This paper focuses on sharing experiences regarding teaching the topics of the first two projects utilizing CN1. These topics include OOP concepts, design patterns, and their related topics such as class associations, Graphical User Interface (GUI) design, and event-driven programming. The discussion of insights regarding utilizing CN1 for teaching the topics of the last two projects, which focus on object-oriented implementations of interactive graphics, animation, two-dimensional graphical transformations, and Bezier curves, is left as a future work.

PROJECT#	TOPICS AND GAME STAGES
1	*TOPICS:* Using OOP concepts and class associations, students design a class organization for the entire game utilizing Unified Modelling Language (UML). They implement this design in CN1. *GAME STAGE:* The game has a text-based interface. The player enters the single character game commands through a text-box on screen. The map of the game world is printed as text to the console. The player needs to advance the time. The game pretends that collisions between game objects happen.
2	*TOPICS:* Students add a GUI to the game which replaces the single-character game commands. They implement four to six different design patterns (e.g., they use iterator design pattern to maintain a collection of game objects), and update the UML diagram of the game. *GAME STAGE:* The game commands are entered through the GUI. The map of the game world is still printed to the console. The player still needs to advance the time. The game still pretends that collisions happen.
3	*TOPICS:* Students use interactive graphics techniques such as component repainting (to draw game objects on screen) and on-screen selection to allow the player to interact with the game objects. They add animation via use of the automatic timers. They also add collision detection and sound. *GAME STAGE:* The map of the game world is displayed graphically on screen. Time is advanced automatically and the game objects are animated on screen. The player can interact with game objects via making on-screen selections using a pointer (i.e., a finger). The collisions are automatically detected and collision sounds are played. The game world is displayed upside down and game objects are not properly rotated.
4	*TOPICS:* Students use two-dimensional graphics techniques to make game objects transformable (i.e., objects can be rotated, translated, and scaled), to convert between local/world/screen coordinates systems, to add panning and zooming capability to the game, and to draw Bezier curves. *GAME STAGE:* The game world is displayed right side up and the game objects are properly rotated. The player can zoom and pan.

TABLE I
TOPICS OF THE PROJECTS AND STAGES OF THE GAME DEVELOPMENT

To prepare their projects students use lecture notes as a reference. In the lectures, OOP concepts and design patterns are introduced by providing code snippets written in CN1. Before the first project is released, particular OOP concepts and their related topics (e.g., class associations such as aggregation, composition, and dependency) are discussed. Code snippets of concrete examples that involve CN1 implementation of

OOP concepts such as abstraction, encapsulation, inheritance, polymorphism, and interfaces are provided. Likewise, before the second project is released, design patterns and their related topics are discussed. For instance, to introduce the command design pattern, first its related topics, which are GUI design and event-driven programming, are introduced. The design patterns that are covered include iterator, composite, singleton, observer, command, strategy, proxy, and factory. Code snippets that highlight important parts of the CN1 implementations of these design patterns are provided in class lectures. Students are asked to implement four to six different design patterns in their second project.

C. Teaching with CN1

CN1 is a Java-based framework and uses Java language features (i.e., syntactic constructs). Hence, most of OOP concepts, design patterns, and their related topics can be introduced to the students in CN1 in the same way that they are introduced in Java (e.g., in the way presented by Horstmann [38]). However, there is an important difference between CN1 and Java that requires a slightly different approach to teach some of these topics in CN1 as compared to that used in Java. The difference is that CN1 does not support many of the Java APIs since they were primarily designed for desktop environments and were not appropriate for mobile devices. To remedy for unsupported APIs, CN1 usually provides its own alternatives. For instance, CN1 does not support the AWT/Swing APIs which are needed in the GUI design; it instead provides the UI API, which is a redesigned version of the AWT/Swing APIs for mobile devices.

As mentioned above, to introduce the command design pattern, first GUI development and event-driven programming concepts are introduced. Hence, to teach the command design pattern and its related concepts, the UI API should be utilized in CN1 whereas the AWT/Swing APIs would have been utilized in Java. However, the UI API provides classes and interfaces which are similar to the ones provided by the AWT/Swing APIs. Table II presents sample classes and interfaces that show the correspondence between the AWT/Swing and UI packages. Hence, the approach used for teaching the command design pattern and its related concepts in Java can easily be adapted for teaching these topics with CN1 instead.

AWT/SWING (JAVA)	UI (CN1)
Component	Component
JFrame	Form
JPanel	Container
JButton/JLabel/JCheckBox	Button/Label/CheckBox
JMenuBarJMenu/JMenuItem	Toolbar
BorderLayout/BoxLayout/FlowLayout	BorderLayout/BoxLayout/FlowLayout
ActionEvent/MouseEvent/KeyEvent	ActionEvent
ActionListener	*ActionListener*
AbstractAction	Command

TABLE II
SAMPLES OF THE CORRESPONDENCE BETWEEN JAVA AWT/SWING AND CN1 UI CLASSES AND *INTERFACES*

D. Recommended Teaching Strategies

Based on positive experience, CN1 is recommended for use in teaching OOP concepts, design patterns, and possibly other software engineering topics. Hence, this section provides tips and recommendations for educators wishing to adapt CN1 as a teaching tool. The following teaching strategies were observed to have helped enhance students' learning:

1) Giving a demo of the installation of CN1: Even if labs and remote servers with CN1 are accessible to students, most students prefer to install CN1 on their personal computers. Hence, giving a demo of the installation steps at the beginning of the course is recommended.

As mentioned above CN1 is installed as a plugin to the three popular IDEs, namely, NetBeans, Eclipse, and IntelliJ IDEA. The latest version of CN1 usually requires the latest version of the IDE to be installed. For instance, as of this writing, to install latest version of CN1 (i.e., CN1 3.6), the latest version of Eclipse for Java Developers called Neon.2 is required. In addition, to install these IDEs, the Java SE Development Kit (JDK) should first be installed. Hence, a demo should present links to the proper versions of JDK/IDEs and explain how to download and install them. Then it can show how to install the CN1 plugin. Finally, the development of "CN1 Hello World" project can be presented [39].

2) Requiring students to use one IDE: As mentioned above, CN1 can be installed in three different IDEs. However, some of the IDEs might not have the latest CN1 plugins. For instance, before it received a major update in Spring 2016, IntelliJ IDEA had an outdated CN1 plugin [40]. Hence, it is recommended to test all CN1 snippets listed in lecture notes and sample project solutions in one IDE and then require students to use this particular IDE. This minimizes any discrepancy that students might face when reading the notes and coding their projects.

3) Grading the projects from command-line: Mobile applications developed with CN1 can run from the command-line [41], and it is recommended to make use of this functionality for simplifying the grading process. When an application is built, CN1 creates a jar file that includes all the class files of the application. Then this jar file can be utilized to run the application from the command line. Hence, instead of submitting the whole application as developed in the IDE, students would only be required to submit this jar file for the grader to run and test their applications. Along with this jar file, it is also recommended to require students submit the source files so that the grader can check the coding style and adherence to design patterns. Since the grader would not need to build and run the submissions using an IDE, the grading process would be faster and thus, the students can receive their feedback more quickly. In addition, if students are allowed to use different IDEs, running projects from the command-line provides a standard grading process for all submissions.

4) Emphasizing the differences between CN1 and Java: As mentioned in Section IV-C, there is an important difference

between CN1 and Java in that CN1 does not support many of the Java APIs. It is recommended to make this difference clear to the students in the beginning of the course. The instructor can show that the Javadocs of Java SE [42] and CN1 [43] are different and explain that students should refer to CN1 Javadocs to develop their projects. This practice will prevent students from mistakenly refering to Javadocs of Java later in the semester.

5) Avoiding the use of automatic GUI builder: Like most other programming environments, CN1 provides an automatic GUI builder. However, to help students correctly implement some of the design patterns, it is recommended to not use this functionality, and to avoide automatically generated code. In addition, letting students build the GUI through the use of programming helps them better understand how GUIs are constructed and how GUI builders work. To avoid the use of automatic GUI builder, use the CN1 "Hello World (Bare Bones)" template while creating the application.

6) Explaining the build servers, device installation, code signing, and distribution: As mentioned above, CN1 comes with a device simulator; developers are not required to have an access to physical mobile devices to develop and test their applications. However, almost all tech-savvy Millennials have mobile devices where they can install and test the mobile applications that they develop in the course. Although device installation is not a requirement of the course as taught, it is observed that most students enjoy learning to test and play the game that they developed throughout the semester on their personal mobile devices. Since CN1 is a cross-platform environment, students are not restricted to use mobile devices that have certain operating system, so they can utilize various kinds of smartphones or tablets they might have to test their games. In addition to device installation, some students are also interested in mobile application signing and distribution and would like to know how they can publish their applications in mobile marketplaces such as App Store and Play Store.

Hence, it is recommended to introduce students to these relevant concepts, even though they are not part of the course requirements. To explain to students how device installation can be accomplished, first the concept of CN1 build servers should be introduced. After an application is developed and tested on the device simulator, the developer sends it to CN1 build servers which convert the CN1 code of the application to native Android, iOS, or Windows code. Then this converted code is used to install the application on a physical device. Device installation process for Android devices is simpler than for iOS devices since at this stage Android does not require the code to be signed. Android only requires the code signing process when the code is distributed (e.g., published in Play Store), whereas Apple requires the code signing process both in device installation and distribution stages (also called "debug" and "release" stages, respectively). Students should also be introduced to concepts like "certificates" and "provisioning" to explain how device installation, code signing, and distribution works. More information on all of these concepts and build servers can be found in the CN1 Developer Guide [44].

V. Results

With the support of grant programs funded by CSU Course Redesign with Technology Office and CSUS Center for Teaching and Learning, the related software engineering course was redesigned to use the mobile programming approach. The first section of the redesigned version of the course was offered in Spring 2016. The ratios of repeatable grades (D, F, and W grades) that students received before and after the redesign of the course were compared. The statistics show that the percentage of the repeatable grades is significantly reduced (by 11.6%) in the redesigned version of the course.

In addition to these quantitative findings, it was observed in the redesigned version of the course that students appeared to be more engaged in class activities and were asking more questions. A discussion board set up for the course was more actively used. It was also observed that students were frequently commenting on how they can apply the skills learned in lectures to outside the classroom. For instance, several different students said that they plan to use CN1 to develop applications in their jobs. Some of the students mentioned that having the knowledge of CN1 helped them get internships. In addition, several other students mentioned that they plan to publish the mobile game they have developed in the projects to popular mobile marketplaces to strengthen their resume, since many potential employers like candidates to be able demonstrate that they can develop mobile applications.

These learning outcomes and observations suggest that students are eager to go beyond class requirements and are motivated to apply the knowledge they acquire from this redesigned course outside the classroom. They are more inclined to spend effort learning and practicing software engineering concepts when applied to mobile technologies, towards which they are intrinsically motivated, with a drive far beyond fulfilling the basic course requirements. In addition, the feedback received from students suggests that building a skill-set in mobile software development can help them find better jobs or perform better in their current positions.

VI. Conclusion

Innovative teaching approaches are beneficial in addressing the expectations of Millennials and their prospective employers with respect to the software engineering curricula. This paper introduces one such approach and proposes the use of a hands-on cross-platform mobile programming environment for teaching two important topics in software engineering curricula: OOP concepts and design patterns. The results of this study indicate that this approach increases the engagement of Millennials who have diverse backgrounds, improves the students' success rate, and might help them find better jobs.

This study utilizes a mobile programming environment called Codename One (CN1) as a novel hands-on learning tool. Since CN1 is a Java-based framework, it is a very suitable method for teaching OOP concepts and design patterns. In

addition, since CN1 allows developing cross-platform mobile applications, it motivates technically diverse Millennials further in course topics and is attractive to prospective employers. Based on positive experience from this study, CN1 is recommended as a teaching tool to educators in the software engineering field. In line with this recommendation, this study also suggests teaching strategies that give tips and observations regarding the effective use of CN1 as a teaching tool. Although the hands-on learning techniques introduced in this study utilize CN1, they are transferable to other mobile application development environments which use object-oriented programming languages such as Android Studio, iOS SDK, and Xamarin.

In addition, this study contributes to the developing area of research that focuses on integrating mobile technology to computer science education by presenting an approach that aims to incorporate mobile programming to software engineering curriculum and providing an extensive survey of related pedagogical studies in this field.

ACKNOWLEDGMENT

This work was supported by Promising Practices in Course Redesign and Sustaining Success in Course Redesign grant programs of CSU Course Redesign with Technology Office and Pedagogy Enhancement Award of CSUS Center for Teaching and Learning. The author would like to thank colleagues Dr. John Clevenger and Dr. Scott Gordon for providing valuable feedback in the initial design of this project and comments on a prior draft of this paper. In addition, the author would like to thank CN1 team members Vered Katch and Shai Almog for providing important insights concerning the technical details of CN1.

REFERENCES

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Prentice-Hall Inc., 1994.

[2] O. J. Dahl, "The birth of object orientation: the simula languages," in *From Object-Orientation to Formal Methods*, 2004, vol. 2635.

[3] K. Beck and W. Cunningham, "Using pattern languages for object-oriented program," Apple Computer, Inc. and Tektronix, Inc., Tech. Rep. CR-87-43, 1987.

[4] N. Howe and W. Strauss, *Millennials Rising: The Next Great Generation*. Vintage Books, 2000.

[5] Codename One, "Java-based cross-platform mobile application development framework," Accessed: Feb, 2017, http://www.codenameone.com.

[6] EMLYON Business School, "Codapps: Coding mobile apps for entrepreneurs," Accessed: Feb, 2017, https://www.coursera.org/learn/codapps.

[7] B. G. Davis, *Tools For Teaching*, 2nd ed. John Wiley & Sons, 2009.

[8] Nielson, "Mobile millennials: Over 85% of generation Y owns smartphones," Accessed: Feb, 2017, http://www.nielsen.com/us/en/insights/news/2014/mobile-millennials-over-85-percent-of-generation-y-owns-smartphones.html.

[9] B. Stackpole, "Your next job: Mobile app developer?" Accessed: Feb, 2017, http://www.computerworld.com/article/2509463/app-development/your-next-job--mobile-app-developer-.html.

[10] D. A. Kolb, *Experiential learning: Experience as the Source of Learning and Development*. Prentice-Hall Inc., 1984.

[11] ABET, "Criteria for accrediting computing programs, 2016-2017," Accessed: Feb, 2017, http://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2016-2017/.

[12] K. Sung and A. Samuel, "Mobile application development classes for the mobile era," in *Proc. 19th ITiCSE*, 2014.

[13] B. Burd, J. P. Barros, C. Johnson, S. Kurkovsky, A. Rosenbloom, and N. Tillman, "Educating for mobile computing: Addressing the new challenges," in *Proc. ITiCSE-WGR*, 2012.

[14] A. Minaie, P. Sanati-Mehrizy, A. Sanati-Mehrizy, and R. Sanati-Mehrizy, "Integration of mobile devices into computer science and engineering curriculum," in *Proc. ASEE Annual Conference & Exposition*, 2011.

[15] X. Suo, "Teaching mobile computing: Curriculum design and strategies applied," in *Proc. FEC*, 2013.

[16] Q. H. Mahmoud, "Best practices in teaching mobile application development," in *Proc. 16th ITiCSE*, 2011.

[17] A. J. Gordon, "Concepts for mobile programming," in *Proc. 18th ITiCSE*, 2013.

[18] Y. Khmelevsky and V. Voytenko, "Integrating mobile culture into computing education," in *Proc. 2nd ISEC*, 2012.

[19] P. E. Dickson, "Cabana: A cross-platform mobile development system," in *Proc. 43rd SIGCSE*, 2012.

[20] K. Lutes, "Cross-platform mobile app software development in the curriculum," *Issues in Informing Science and Information Technology*, vol. 9, 2012.

[21] R. England, "A software development course featuring iPad app construction," *J. Comput. Sci. Coll.*, vol. 27, no. 5, 2012.

[22] M. Thomas, "iOS app programming using an inverted classroom in a small department," *J. Comput. Sci. Coll.*, vol. 29, no. 5, 2014.

[23] J. Hollingsworth and D. J. Powell, "Requiring web-based cloud and mobile computing in a computer science undergraduate curriculum," in *Proc. 49th ACM-SE*, 2011.

[24] H. Kang and J. Cho, "Case study on efficient Android programming education using multi Android development tools," *Indian Journal of Science and Technology*, vol. 8, no. 19, 2015.

[25] L. E. Margulieux, M. Guzdial, and R. Catrambone, "Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications," in *Proc. 9th ICER*, 2012.

[26] D. Riley, "Using mobile phone programming to teach Java and advanced programming to computer scientists," in *Proc. 43rd SIGCSE*, 2012.

[27] J. Yoo, S. Yoo, and Z. Dong, "Mobile app development course for teaching OOD," in *Proc. EdMedia*, 2013.

[28] M. H. Goadrich and M. P. Rogers, "Smart smartphone development: iOS versus Android," in *Proc. 42nd SIGCSE*, 2011.

[29] B. R. Payne, "Teaching Android and iOS native mobile app development in a single semester course," *J. Comput. Sci. Coll.*, vol. 30, no. 2, 2014.

[30] Y. Khmelevsky and V. Voytenko, "A new paradigm for teaching mobile application development," in *Proc. 21st WCCCE*, 2016.

[31] Google, "Android studio," Accessed: Feb, 2017, https://developer.android.com/studio/index.html.

[32] Apple, "iOS SDK," Accessed: Feb, 2017, https://developer.apple.com/ios/.

[33] Adobe, "PhoneGap," Accessed: Feb, 2017, http://phonegap.com/.

[34] Microsoft, "Xamarin: Mobile app development & app creation software," Accessed: Feb, 2017, https://www.xamarin.com/.

[35] B. Nielson, "Cross platform mobile development," Master's thesis, Dept. of Computer Science, University of Aarhus, Denmark, 2015.

[36] Codename One, "Use the source," Accessed: Feb, 2017, http://www.codenameone.com/blog/use-the-source.html.

[37] J. Clevenger and S. Gordon, "Object-oriented computer graphics programming lecture notes and assignments," Spring, 2014, California State University, Sacramento.

[38] C. S. Horstmann, *Object-oriented design & patterns*, 2nd ed. John Wiley & Sons, 2006.

[39] Codename One, "Building Hello World using Codename One," Accessed: Feb, 2017, https://www.codenameone.com/hello-world.html.

[40] ——, "A new idea," Accessed: Feb, 2017, https://www.codenameone.com/blog/a-new-idea.html.

[41] ——, "Simulator command line arguments," Accessed: Feb, 2017, https://groups.google.com/forum/#!topic/codenameone-discussions/KWpBdwj8RuI.

[42] Oracle, "Java platform, standard edition 8, API specification," Accessed: Feb, 2017, https://docs.oracle.com/javase/8/docs/api/.

[43] Codename One, "Codename One API," Accessed: Feb, 2017, https://www.codenameone.com/javadoc/index.html.

[44] ——, "Developer guide," Accessed: Feb, 2017, https://www.codenameone.com/files/developer-guide.pdf.