Volume 13, No. 5
September 2015
ISSN: 1545-679X

INFORMATION SYSTEMS EDUCATION JOURNAL

In this issue:

The **Information Systems Education Journal** (ISEDJ) is a double-blind peer-reviewed academic journal published by **EDSIG**, the Education Special Interest Group of AITP, the Association of Information Technology Professionals (Chicago, Illinois). Publishing frequency is six times per year. The first year of publication is 2003.

ISEDJ is published online (http://isedjorg). Our sister publication, the Proceedings of EDSIG (http://www.edsigcon.org) features all papers, panels, workshops, and presentations from the conference.

The journal acceptance review process involves a minimum of three double-blind peer reviews, where both the reviewer is not aware of the identities of the authors and the authors are not aware of the identities of the reviewers. The initial reviews happen before the conference. At that point papers are divided into award papers (top 15%), other journal papers (top 30%), unsettled papers, and non-journal papers. The unsettled papers are subjected to a second round of blind peer review to establish whether they will be accepted to the journal or not. Those papers that are deemed of sufficient quality are accepted for publication in the ISEDJ journal. Currently the target acceptance rate for the journal is under 40%.

Information Systems Education Journal is pleased to be listed in the 1st Edition of Cabell's Directory of Publishing Opportunities in Educational Technology and Library Science, in both the electronic and printed editions. Questions should be addressed to the editor at editor@isedj.org or the publisher at publisher@isedj.org.

INFORMATION SYSTEMS EDUCATION JOURNAL

Teaching Non-Beginner Programmers with App Inventor: Survey Results and Implications

Andrey Soares
asoares@siu.edu

Nancy L. Martin
nlmartin@siu.edu

Information Systems Technologies
Southern Illinois University
Carbondale, IL 62901, USA

Abstract

This paper reports the results of a survey with 40 students enrolled in an Android Application Development course offered during the spring semester of 2013 and 2014. The course used App Inventor to build the apps and required students to have an introduction to programming course as a prerequisite. The survey asked for demographic information and students' opinions about prerequisites, App Inventor, previous programming skills, new concepts learned, teamwork, and more. The positive responses support the practice of using App Inventor to teach not only beginner programmers, but also more experienced programmers. The paper also shows that App Inventor can be used to support the teaching of more advanced computing concepts.

Keywords: App Inventor, Mobile Applications, Non-Beginner programmers, Survey.

1. INTRODUCTION

App Inventor is a visual programming language developed by Google in 2010 and currently hosted and maintained by the MIT Center for Mobile Learning. App Inventor has been successfully used to teach introductory programming concepts to beginners in both secondary and higher education courses (Abelson, 2012; Haungs, Clark, Clements, & Janzen, 2012; Robertson, 2014).

App Inventor can also be used to teach programming and other computing concepts for students that already have some programming experience (Gestwicki & Ahmad, 2011; Soares, 2014). For example, Gestwicki and Ahmad (2011) suggest that App Inventor and their Studio-Based Learning approach can be used not only to "introduce non-CS majors to concepts of Computer Science-not just programming, but also ideas that tend not to be covered in conventional CS1 courses such as human-computer interaction, incremental and iterative design processes, collaboration, evaluation, and quality assurance" (p. 55). Soares (2014) discusses issues, challenges and opportunities that instructors should be aware of when designing a course in mobile application development with an introductory programming course as prerequisite.

This paper presents the results of a survey with students enrolled in a mobile application development course that used App Inventor as the tool for teaching and building applications, and required an introduction to programming course as prerequisite.

The following section of the paper describes the methods used for data collection. The results and discussion of the data analysis are combined

into the next section, and the paper closes with conclusions and recommendations.

2. METHODS

The course, Android Application Development, was offered during spring semester in 2013 and 2014, and had an introductory programming course as a prerequisite.

In order to explore students' perceptions about the course, the instructor developed a brief exploratory survey, and students were asked to complete it during the last week of the semester. The instructor explained that survey completion was voluntary and anonymous and was not in any way related to course grades. A total of 40 students enrolled in the course completed the survey, 16 in spring 2013 and 24 in spring 2014. The survey questions contained 37 unique data points that were designed to gather students' feedback on a variety of topics related to the course. Questions asked for demographic information and perceptions about course prerequisites, the App Inventor tool, the reinforcement of programming fundamentals, concepts learned, teamwork, and an interest in learning more about mobile application development.

All questions, other than demographic related ones, were answered on a 5-point Likert scale ranging from strongly agree to strongly disagree. Since the survey was exploratory in nature with no predicted outcomes, the data was analyzed using descriptive statistics, primarily frequency analysis.

In the next section, the questions and their data analysis are presented, along with a relevant discussion of each topic.

3. RESULTS AND DISCUSSION

Demographics
Forty students completed the survey during the spring semester in 2013 and 2014. Table 1 displays the breakdown by gender and class. Males comprised 88 percent of the sample; females comprised 10 percent, and one person did not report gender. The students were either juniors (22.5 percent) or seniors (77.5 percent). As the course has the Introduction to Programming course as a prerequisite and is currently not required as prerequisite for any other course in the program, students generally take it in their third or fourth year.

	Class		
Gender	Junior	Senior	Total
Female	0	4	4
Male	9	26	35
Missing	0	1	1
Total	9	31	40

Table 1: Sample demographics

Prerequisites
Although an introductory programming course was a prerequisite for the Android course, students were asked whether they agreed or disagreed with the reference text's statement that no programming experienced is required. As Figure 1 reflects, over 46 percent agreed that no programming experience was required, while only 18 percent disagreed and 36 percent were neutral.

Students were also asked whether they believed certain courses should be prerequisites for the Android course. The results of that question are shown in Figure 2. Interestingly, even though about 46 percent of students agreed that no programming experience was required for the course, about 43 percent agreed that Programming II should be required. A database course was also identified by about 43 percent of students as a recommended prerequisite.

Because some assignments were completed in groups requiring close collaboration, some students even considered Project Management (21%) and Software Engineering (15%) as prerequisites for the course. Almost all assignments resulted in a new app created from scratch, with some exceptions where students improved their existing apps. That means, for each app they were supposed to plan, design, implement and test their apps. When working in groups, students will most likely deal with scope definition, scheduling, task management, communication, human resources and other activities needed to complete their apps.

App Inventor Tool for Beginners and for Experienced Programmers
Respondents overwhelmingly agreed that App Inventor is a great tool for teaching both beginners and more experienced programming students. Figure 3 shows that 87.5 percent agreed or strongly agreed the tool was good for beginners, and 85 percent reported the same opinion related to those with some programming experience.

App Inventor provides developers with the ability to quickly design and implement an app using a variety of features from a mobile device. The use of a visual tool reduces code-related distractions (e.g., missing semi-colon, braces or misspelled code) as students create the application with blocks of code (Soares, 2014). The tool hides some of the complexity of the code by providing predefined blocks for specific functions. Figure 4 shows examples of the event Click for the component Button1 both as a visual code and as a textual code. When the component Button1 is created, the block "When-Button1.Click-do" and several others are automatically created and are available for students to just drag-and-drop it into the blocks editor.

Because of the blocks, students do not have to worry about the syntax of the code, and they can focus "more on the functionalities of the application and what can be done with the phone" (Soares, 2014, p. 59). Nonetheless, they still need to know about the logic of programming to complete the application, especially the event-driven programming approach. Ninety-five percent of the respondents agreed/strongly agreed that App Inventor helped them learn about developing mobile applications, and 85 percent demonstrated interest in learning more about developing mobile applications for smartphones and tablets. When asked if writing the code in Java to create applications would be preferred over the App Inventor, 47.5 percent of students disagreed/strongly disagreed, 27.5 percent were neutral, and 25 percent agreed/strongly agreed.

Similar to others' experience (Robertson, 2014) it appears that overall, both more and less experienced students found value in the use of App Inventor.

Reinforcing Fundamentals of Programming

Most students agreed that it was easy to apply previous programming concepts to the App Inventor environment. Figure 5 shows that 85 percent of the respondents agreed or strongly agreed.

Students also generally agreed that developing mobile applications with App Inventor helped reinforce fundamentals of their programming knowledge. Figure 6 displays the responses related to several programming fundamentals. The first three are the use of variables, conditions, and loops. Eighty percent of students agreed or strongly agreed that their knowledge of variable use improved, and 82.5 felt the same about the use of conditions. Fifty-five percent reported knowledge reinforcement of loops.

Another area of previous programming knowledge reinforced in class is the use of procedures. Figure 6 also displays student opinions about this topic. Regarding the use of procedures with a return value, 60 percent of students agreed/strongly agreed that App Inventor helped them to reinforce the knowledge on how to use procedures. Thirty percent were neutral and 10 percent disagreed/strongly disagreed. Regarding the use of procedures that do not return any value, 77.5 percent of students agreed/strongly, 17.5 were neutral and 5 percent disagreed/strongly disagreed that App Inventor was a useful tool.

In some cases, students would create a global variable and would use a procedure to change its value, rather than making the procedure to return a value. Creating input arguments to allow data to be passed on to the procedures was also challenging, and often students would use global variables or read data from existing components as a workaround. When the data to be passed was related to a component, some students explored the use of the advanced blocks, which allows applications to work with components dynamically.

Working with Lists (Arrays)

More than 80 percent of students reported that mobile application development with App Inventor strengthened their knowledge of both single lists (i.e., arrays) and multidimensional lists (i.e., multidimensional arrays). Specific response percentages are shown in Figure 7.

Arrays are considered a difficult topic for students to learn (Dale, 2006; Lahtinen, Ala-Mutka, & Jarvinen, 2005). Considering the prerequisite of an introduction to programming course, "even students that are already familiar with the concepts of arrays may need a period of adjustment to translate and adapt their prior knowledge with arrays into the new environment" (Soares, 2014, p. 61). Nonetheless, students taking the Android Application Development course may have different levels of programming experience. For example, arrays may not have been covered in the introduction to programming course and thus will be a new topic, or students may have taken other programming courses that cover the topic of arrays and be fairly familiar with the topic.

The concept of lists is introduced early in the course because it is required for several assignments, either as the main focus of an assignment or as part of a larger application. For example, an extended version of the App Inventor's "Map It" tutorial uses lists to store information about the location (e.g., name, latitude and longitude) of points of interest on campus and around town. The app starts with a pre-defined list of 10 locations selected by the students (i.e., static list), but users of the app can also add new items to the list or delete existing ones (i.e., dynamic list).

App Inventor has a variety of functions built-in to work with lists, but these functions generally relate to single lists. To work with multidimensional lists, students must write additional code or create specific functions to work with them.

Web Services

Besides learning how to build applications that work with the features of the Android mobile devices, students are also interested in creating apps that can interact with other web applications (Soares, 2014). Not surprising, "the Web is evolving into a dynamic repository of information on virtually every topic, including people and their connections to one another as well as to content" (Ramakrishnan & Tomkins, 2007, p. 63). And, many people, students included, seek opportunities to be content producers (e.g., share their data) and content consumers (e.g., retrieve data from other sources on the Web).

App Inventor has several components (i.e., TinyWebDB, ActivityStarter, FusionTables, Twitter, and Web) that permit developers to incorporate Web Services and APIs into the applications, and permit instructors to use them to teach a variety of topics. For example, the Google Maps API is explored in combination with the component Location Sensor, which reads global positioning system (GPS) coordinates from the device. Note that a developer can interact with the Google Map app installed on the phone, but can also interact with the Google Map API available on the Web. Lim, Jong, and Mahatanankoon (2005) discuss the potential of integrating Web Services earlier into the curriculum to make the course more interesting and to expose students to Web Services and its potential "to speed up application development and reduce costs to access data on disparate systems" (p.241).

This course included several assignments using Web Services and APIs within the apps such as displaying driving directions on a map or a pie chart with data from an online survey app, returning information about a product after scanning a UPC code or an ISBN code, getting the weather forecast or the list of businesses for a given zip code, displaying Bible passages or displaying products for sale from Craigslist. A great source of Web Services and APIs is the website www.programmableweb.com with a list of over 11,500 APIs (as of July, 2014) that can be used to build applications. However, Soares (2014) cautions about the different formats of responses returning from the APIs (e.g., XML, JSON, etc.) and the need to teach students how to parse the responses in order to use the information needed for their apps.

Despite the issue with parsing Web Services responses, when asked if developing mobile applications with App Inventor helped in learning about Web Services, over 76 percent of students agreed or strongly agreed (see Figure 8).

Animation and Sensors

Designing games is a great approach to learn about mobile application development, especially about what the phone can do in terms of interacting with users. Our results show that more than 85 percent of students agreed that they learned about animation (see Figure 8).

The basic tutorials such as PaintPot, MoleMash and Ladybug Chase (see www.appinventor.org) are great introductions to drawing and animation components as they expose students to several functionalities of the phone such as touching, dragging, and tilting. Because of the relatively small amount of code needed to work on the tutorials, students usually get excited about creating their own games. Of course, some game ideas are too complex and will require students to combine several concepts learned throughout the course. Since this course started being offered in spring 2012, many of the students' final projects utilized some of the drawing and animation components; for example, a chess game played over Bluetooth, a flight combat game, a breakout game, and more.

The component Clock is also used to implement games and other applications that require control of time (e.g., time left to play) or need to take actions repeatedly (e.g., move an object every tick of the clock). In addition, the Accelerometer sensor, Location sensor, and

Orientation sensor can be added to the applications to improve the users' experience.

Table 2 presents a list of events and parameters for the sensor components. The location sensor works with the GPS and provides information on latitude, longitude, altitude, and address; the orientation sensor provides information about the phone's orientation (i.e., tilt and direction pointing to); and the accelerometer sensor detects acceleration using the X, Y, and Z dimensions as well as the shaking of the phone.

Accelerometer	AccelerationChanged: xAccel yAccel zAccel Shaking
Location	LocationChanged: latitude longitude altitude StatusChanged: provider status
Orientation	OrientationChanged: azimuth pitch roll

Table 2: Events and parameters for the sensor components

Participants were asked if developing mobile applications with App Inventor helped them to learn about sensors. About 88 percent agreed/strongly agreed they learned about Location sensor, 85 percent agreed/strongly agreed they learned about Orientation sensor, and 87.5 percent agreed/strongly agreed they learned about Accelerometer sensor.

Event-Driven Programming

Even though events are considered an important programming concept, they "are typically taught late in the CS curriculum" (Turbak, Sherman, Martin, Wolber, & Pokress, 2014, p. 81). Because the prerequisite for this course is an introduction to programming course, students may not be familiar with the concept of events or event-driven programming as it may not have been covered in the prerequisite course.

Wolber, Abelson, Spertus, and Looney (2011) explain that "with App Inventor, you design how an app looks and then you design its behavior—the set of event handlers that make an app behave as you want" (p.227). They describe that an app responds to user-initiated events, initialization events, timer events, animation events or external events. Students, especially novice programmers may find it difficult to identify all the events of a behavior (Soares, 2014; Wolber et al., 2011).

In this course, events are introduced in the first week of class and then reinforced throughout the course with examples, in-class discussions, and lab assignments. In addition, for some assignments, students are required to design mockup screens of the applications to be created, which helps them "to think not only about the components but also about the underlying events, functions and blocks that need to be used to achieve the desired results" (Soares, 2014, p. 59). In our survey, 82.5 percent of students agreed/strongly agreed that developing mobile applications with App Inventor helped them to learn about event-driven programming (see Figure 8).

Database

The topic of database is covered in the course in two ways. First, we discuss the phone's internal database and we use the component TinyDB to store and retrieve data from the local database. Second, we discuss the use of web databases, starting with the component TinyWebDB and later exploring the component Fusiontables. Both TinyDB and TinyWebDB are fairly straightforward since the developer simply uses tags to store and retrieve data from the databases. Fusiontables, on the other hand, has more complexities and requires developers to work with Google Drive and Google Fusiontables API in order to create a table and make it available to integrate with an app. During implementation with App Inventor, developers must understand the basics of database design and Structured Query Language (SQL) to query the tables. With Fusiontables, developers can use commands to insert, update, delete and select data from the tables. As Soares (2014) describes, "the query results are in CSV or JSON formats and can be transformed into lists with the appropriate blocks in App Inventor" (p.61).

When asked if they would recommend a database course as a prerequisite for the Android Application Development course, 43 percent of students agreed/strongly agreed, 18 percent were neutral, and 41 percent disagreed/strongly disagreed (see Figure 2). Considering that the majority of students are seniors and have likely taken a database course, these answers are rather surprising, especially since 80 percent of students agreed/strongly agreed that they learned about web databases in

the course. On one hand, students may have previous knowledge of database design and SQL, and they considered the learning of how to work with TinyWebDB and Fusiontables during the course. On the other hand, students may be new to the concept of database, and they considered learning about the concept during the course.

User Interface and Input/Output

One of the main reasons students take the Android application development course is the excitement of building their own apps. It is definitely fun, however some students feel overwhelmed by the process of creating an app and focusing on the user interaction with the app and all the necessary validations and tests involved.

As discussed earlier, the visual programming approach of App Inventor helps to hide some of the complexities of programming, providing students with opportunities to concentrate on the design of the application, its features, and how users will interact with it. That means, students should learn and practice the design of user interfaces, user input and output, and input validation as they play an important role in the user's experience with mobile applications.

More often than not, students will detect some problems with their apps that can be a result of poor user interface design. In particular, user inputs are overlooked which will make apps misbehave or crash when users enter unexpected data or do not provide any data. Of course, App Inventor provides several properties to the components to allow developers to set up the application as needed. For example, the component TextBox can be set to number only in order to restrict the type of data entered. However, it is the developer's responsibility to define a range of acceptable numbers and to create the appropriate code to validate it. Other properties such as enable/disable and visible/hidden provide ways for developers to customize their apps. Soares (2014) suggests the use of mockup screens during the planning of applications to help define the apps' user interfaces and behaviors.

Students were asked whether they believed that developing mobile applications with App Inventor helped them to learn about user interface design, user input and output, and input validation. Ninety percent of students answered that they agreed/strongly agreed to have learned about user interfaces, and 77.5 percent of students agreed/strongly agreed to have learned about user input validation (see Figure 8). In addition, 87.5 percent of students agreed/strongly agreed that they reinforced their knowledge about handling user input, and 85 percent of students agreed/strongly agreed their knowledge of user output was strengthened.

Connections and Data Communication

According to a survey by the Pew Research Center's Internet & American Life Project (Duggan & Smith, 2013), "six in ten cell phone owners (63%) now go online using their mobile phones, an eight-point increase from the 55% of cell owners who did so at a similar point in 2012" (p.4). It is not surprising that people are spending more time on their phones and using it mainly for some sort of communications (e.g., with another person or a web/mobile application). In fact, besides making phone calls, the most popular cell phone activities are (Duggan, 2013, p. 2):

- Send or receive text messages (81%)
- Access the internet (60%)
- Send or receive email (52%)
- Download apps (50%)
- Get directions, recommendations, or other location-based information (49%)
- Listen to music (48%)
- Participate in a video call or video chat (21%)
- Check in or share your location (8%)

Students in this course have demonstrated great interest in creating applications that go beyond the capabilities of the mobile device and explore approaches to connect and communicate with other people, devices and applications. During the course, students had the opportunity to use some of the App Inventor components that support connectivity and communication. For example, the Image component can link directly to an image using its URL; the component WebViewer permits the display of a webpage for a specific URL; the component PhoneCall makes a call to the phone number specified; the component Texting permits users to send and receive text messages from other devices; the component FusionTablesControl permits the app to interact with tables stored on Google Drive; the component TinyWebDB connects with a web service that provides database services; the component ActivityStarter permits an app to open an application such as a browser or a map; the components BluetoothClient and BluetoothServer support communication of paired mobile devices; and the Web component supports the use of HTTP methods (e.g., POST and GET) for request-response connections.

Instructors can benefit from these components, when designing course content and assignments, and should encourage students to explore the features of the phone that support connections and data communications. Most students agreed/strongly agreed that developing mobile applications with App Inventor helped them to learn about Bluetooth communication (90%), Web Databases (80%), Web Services (72.5%), and Text Messaging (65%).

Teamwork

Working in groups with App Inventor is not an easy task. Students can work together to plan their apps and share ideas, but when it is time to design and implement their apps, the tool has some limitations that make group work challenging. For example, blocks cannot be copied from one project to another, and two or more people cannot work on the same project at the same time. Even with the limitations of App Inventor to support collaborative work, 68 percent of the respondents agreed/strongly agreed they enjoyed working in teams to develop mobile apps. The course included several group assignments with group sizes of 2 to 5 students.

Besides teamwork, many assignments also included time and scope constraints to challenge and persuade students to manage their tasks and progress. Students had to come up with their own approach to make a group assignment work. For example, groups spent more time planning the app and creating mockup screens to define the components and functionalities needed before starting any code. On some occasions, they even discussed how to name the components so that other group members could easily find them.

Some groups decided to separate their activities and each member would work on their own to complete their respective tasks. After that, they would create a shared account to access App Inventor and then each member would take turns implementing their part of the project. One group tried to get all members logged in on App Inventor at the same time to work on the same project using the same user account. However, they quickly learned that the current version of App Inventor does not support synchronous collaboration. For some groups, each member would work individually on their tasks and send their work to a member that was responsible to combine all parts into one project. Finally, for other groups, the approach was one member working on the computer and the other members around him or her discussing the project and providing support during the implementation.

Learning More about Mobile Application Development

The Android application development course can be considered a success, with great feedback from students, great student evaluations, and students showing interest in learning more about mobile application development. Two students that took the course created their apps, alone or in teams, to enter in the university's App competition.

When asked if, after the course, students would be interested in learning more about developing mobile applications for smartphones and tablets, 85 percent of participants answered that they agreed/strongly agreed and 12.5 percent were neutral. However, some students would prefer to learn how to develop apps using Java (25%).

4. CONCLUSION AND RECOMMENDATIONS

This paper presented the results of a survey with students enrolled in an Android Application Development course with an introduction to programming course as prerequisite. The results show positive feedback from the students about course prerequisites, App Inventor, reinforcing fundamentals of programming, learning new concepts, teamwork, building mobile apps, and more. The paper presented a discussion of the survey results and some recommendations related to the use of App Inventor to teach beginners and more experienced programmers as well as to teach other advanced computing concepts.

App Inventor has been used successfully for teaching beginner programmers from elementary school to higher education and from CS/IS to other majors (Gray, Abelson, Wolber, & Friend, 2012; MacKellar, 2012; Wolber, 2011). In fact, the authors of this paper have experienced firsthand the potential of using App Inventor to introduce programming skills to beginner programmers by offering a summer camp to middle school girls on mobile app development.

The visual programming approach of App Inventor helps students to learn about programming concepts and to apply their existing skills to build mobile apps. As the survey shows, 87.5 percent agreed or strongly agreed the tool was good for beginners and 85 percent agreed or strongly agreed that it was easy to apply previous programming knowledge

to the App Inventor environment. Some students even mentioned during the course that they would prefer to have the Android app development course as the first introduction to programming course rather than a Java course. The reward of seeing quick results and not dealing with the code behind the blocks seems engaging to beginners, but a possible disadvantage for more skilled programmers.

Students who took an introductory programming course prior to this course had the opportunity to apply and reinforce their knowledge on the fundamentals of programming, and also to learn new programming approaches and computing concepts. According to Soares (2014), "the time used for teaching logic and the fundamentals of programming could be used to explore more features of the phone and the App Inventor tool" (p.58). It is not surprising to see that students agreed/strongly agreed that App Inventor is a great tool for teaching more experienced programmers (85%). Because of the assortment of features available on mobile devices and the relatively easy way to handle them with App Inventor, instructors can design course assignments that use basic programming skills but also require the application of more advanced skills to build the apps. Each app created with App Inventor provides students with opportunities to practice different phases of the development process, apply different programming skills, and use different features of the mobile device.

With the help of App Inventor, a mobile development course should be fun and packed with several computing concepts besides programming, such as database, data communication, software development, project management, mobile applications development, web services and more. Now that students have built a background on app development, instructors teaching more advanced courses can illustrate the concepts of their specific courses with the support of App Inventor. For example, in a database course, students could create forms to insert data into tables or display data from the tables using both static and dynamic queries. In a software engineering or systems analysis and design course, students could benefit from App Inventor's support for rapid development to plan, design, implement and test mobile apps as part of course assignments or projects. In particular, instructors could explore principles and techniques for user interface design. Also, it should not take long to find units on campus or other organizations that could use apps for their business and would be interested in collaborating with students through class projects. The Bluetooth communication could be used, for example, in a biomedical or heath information technology course, where students could read data from medical equipment to display the status of the machines or the patients connected to the machines. For a network and security course, the Web capability of App Inventor could be used to monitor and communicate the status of network devices.

5. REFERENCES

Abelson, H. (2012). *From computational thinking to computational values*. Paper presented at the Proceedings of the 43rd ACM technical symposium on Computer Science Education, Raleigh, North Carolina, USA.

Dale, N. B. (2006). Most difficult topics in CS1: results of an online survey of educators. *SIGCSE Bulletin, 38*(2), 49-53. doi: 10.1145/1138403.1138432

Duggan, M. (2013). Cell Phone Activities 2013. Washington, D.C.: Pew Research Center's Internet & American Life Project - available from http://www.pewresearch.org.

Duggan, M., & Smith, A. (2013). Cell Internet Use 2013. Washington, D.C.: Pew Research Center's Internet & American Life Project - available from http://www.pewresearch.org.

Gestwicki, P., & Ahmad, K. (2011). App inventor for Android with studio-based learning. *Journal of Computer Science in Colleges, 27*(1), 55-63.

Gray, J., Abelson, H., Wolber, D., & Friend, M. (2012). *Teaching CS principles with app inventor*. Paper presented at the Proceedings of the 50th Annual Southeast Regional Conference, Tuscaloosa, Alabama.

Haungs, M., Clark, C., Clements, J., & Janzen, D. (2012). *Improving first-year success and retention through interest-based CS0 courses*. Paper presented at the Proceedings of the 43rd ACM technical symposium on Computer Science Education, Raleigh, North Carolina, USA.

Lahtinen, E., Ala-Mutka, K., & Jarvinen, H.-M. (2005). *A study of the difficulties of novice programmers*. Paper presented at the Proceedings of the 10th Annual Conference on Innovation and Technology in Computer

Science Education (SIGCSE), Caparica, Portugal.

Lim, B. B. L., Jong, C., & Mahatanankoon, P. (2005). *On integrating Web services from the ground up into CS1/CS2*. Paper presented at the Proceedings of the 36th Technical Symposium on Computer Science Education (SIGCSE), St. Louis, Missouri, USA.

MacKellar, B. (2012). *App Inventor for Android in a Healthcare IT course*. Paper presented at the Proceedings of the 13th Annual Conference on Information Technology Education, Calgary, Alberta, Canada.

Ramakrishnan, R., & Tomkins, A. (2007). Toward a PeopleWeb. *Computer, 40*(8), 63-72. doi: 10.1109/mc.2007.294

Robertson, J. (2014). Rethinking how to teach programming to newcomers. *Communication of the ACM, 57*(5), 18-19. doi: 10.1145/2591203

Soares, A. (2014). Reflections on teaching App Inventor for non-beginner programmers: Issues, challenges and opportunities. *Information Systems Education Journal, 12*(4), 56-65. (A preliminary version appears in The Proceedings of ISECON 2013).

Turbak, F., Sherman, M., Martin, F., Wolber, D., & Pokress, S. C. (2014). Events-first programming in App Inventor. *Journal of Computer Science in Colleges, 29*(6), 81-89.

Wolber, D. (2011). *App Inventor and real-world motivation*. Paper presented at the Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, Dallas, TX, USA.

Wolber, D., Abelson, H., Spertus, E., & Looney, L. (2011). *App Inventor: Create your own Android Apps*: O'Reilly Media.

APPENDIX

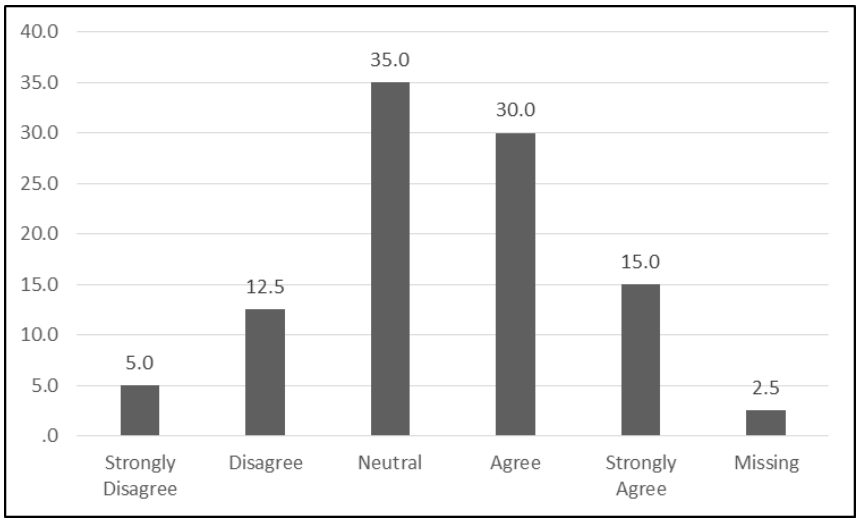


Figure 1: Student response to "no programming experience required".

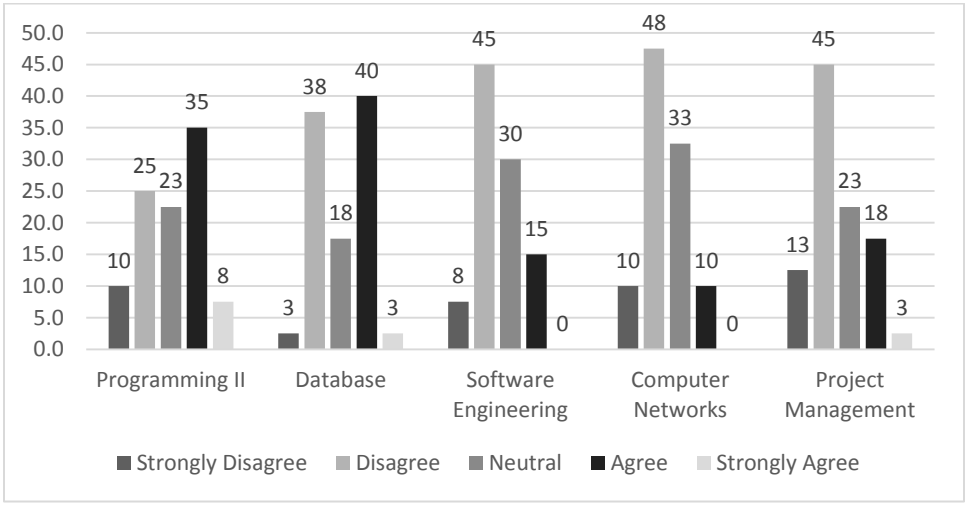


Figure 2: Student responses about Android course prerequisites (in percentages).

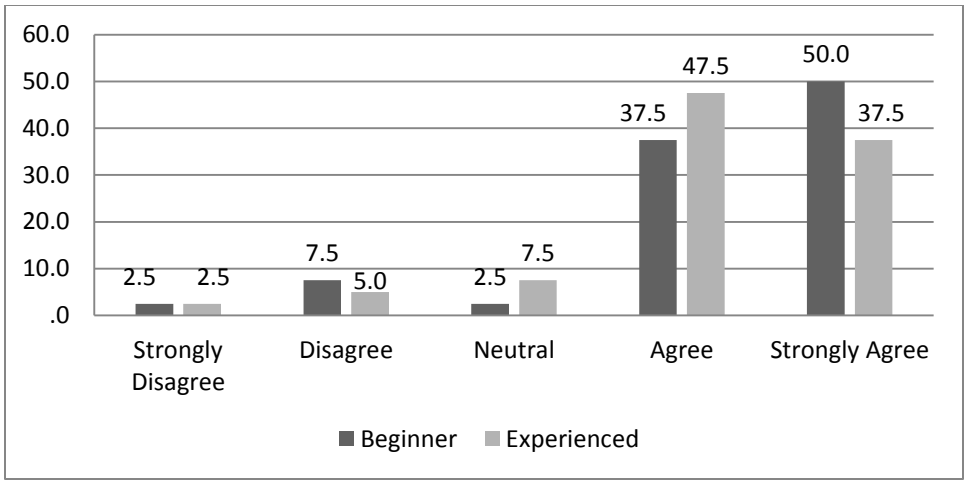
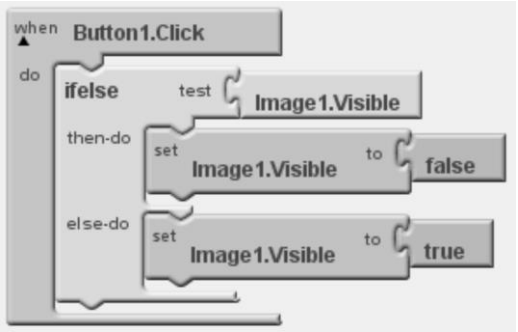


Figure 3: Student responses about App Inventor as a useful tool (in percentages).




```
Button1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(Image1.isShown()) {
            Image1.setVisibility(ImageView.INVISIBLE);
        }
        else {
            Image1.setVisibility(ImageView.VISIBLE);
        }
    }
});
```

Figure 4: Handling the event of a button clicked using Visual code (left) and Textual code (right) (Soares, 2014, p. 58).

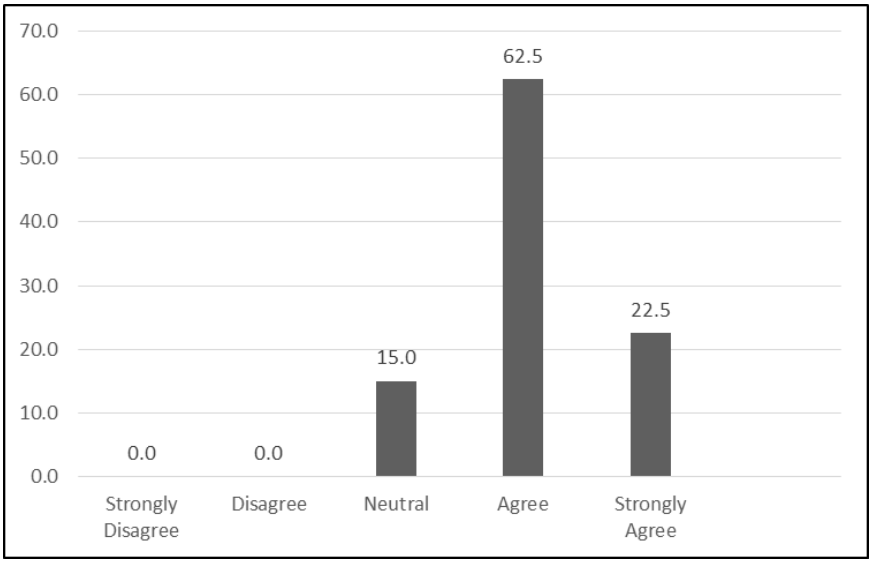


Figure 5: Student response to the ease of applying prior programming knowledge.

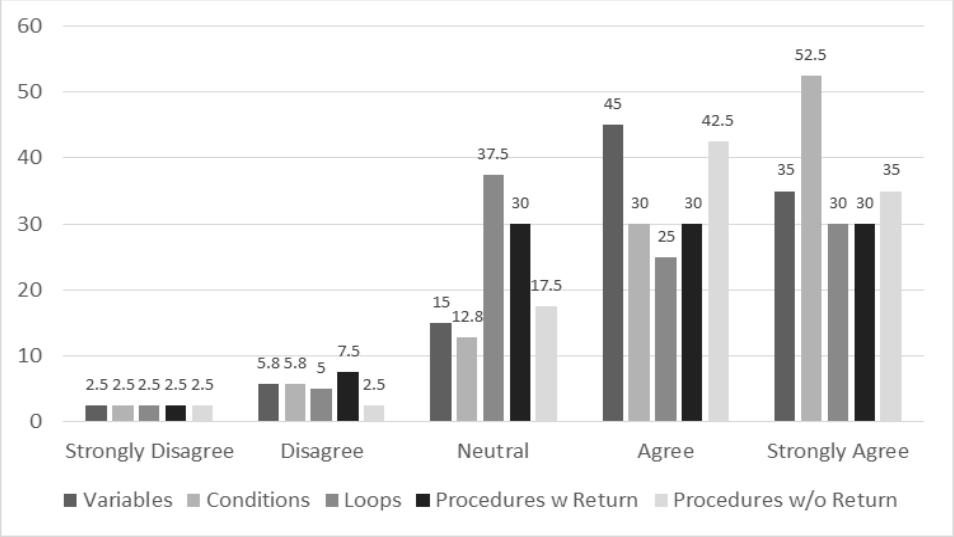


Figure 6: Student responses about the reinforcement of fundamentals (in percentages).

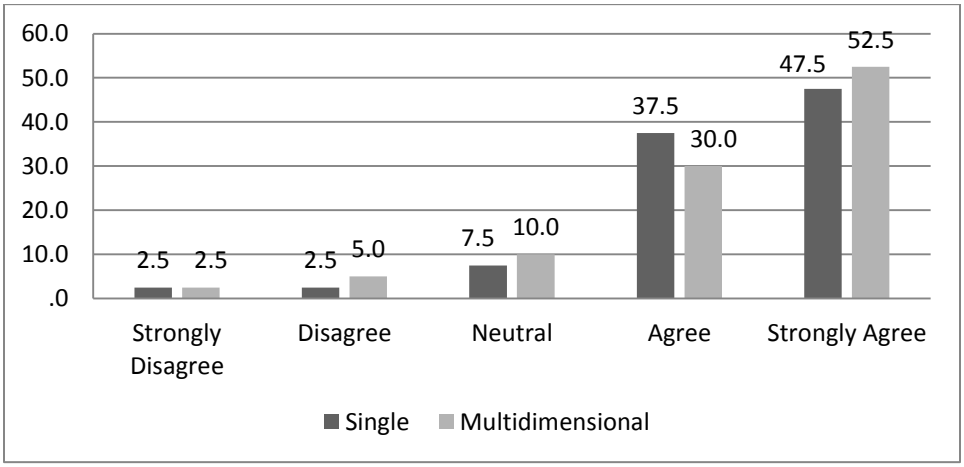


Figure 7: Student responses that App Inventor reinforced knowledge of Lists (in percentages).

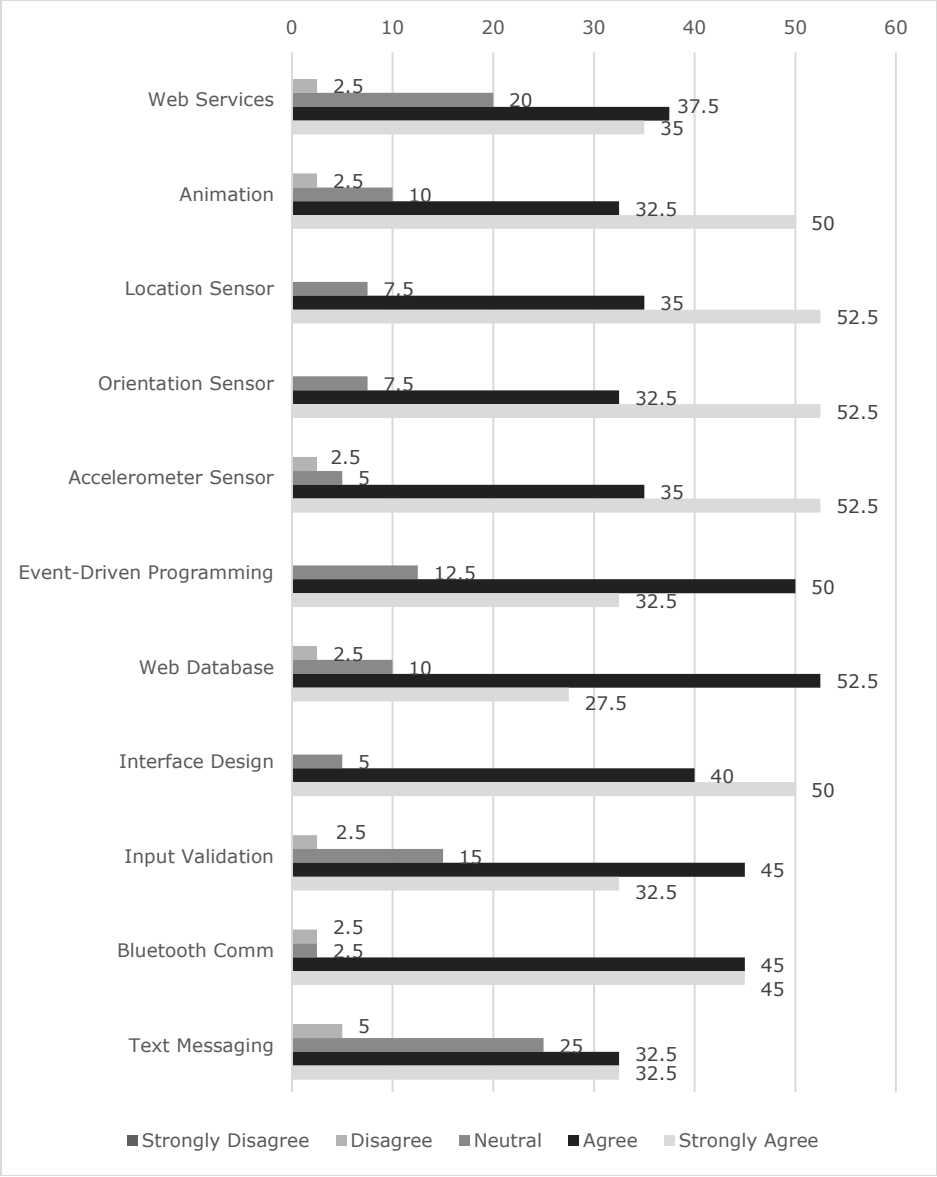


Figure 8: Student response to learning about various topics (in percentages).