Students Want to Create Apps:

Leveraging Computational Thinking to Teach Mobile Software Development

Ilenia Fronza
Free University of Bolzano
Piazza Domenicani, 3
39100, Bolzano (Italy)
Ilenia.Fronza@unibz.it

Nabil El Ioini
Free University of Bolzano
Piazza Domenicani, 3
39100, Bolzano (Italy)
nelioini@unibz.it

Luis Corral
Free University of Bolzano
Piazza Domenicani, 3
39100, Bolzano (Italy)
Luis.Corral@unibz.it

ABSTRACT

Computational Thinking (CT) is recognised as one of the fundamental skills of all graduates. Nevertheless, some issues can emerge when trying to introduce CT into schools; for example, teachers might not be willing to add topics to their intensive syllabi. Therefore, out-of-school venues such as summer schools can be considered a great opportunity for exposure to CT. Moreover, summer schools allow students to meet first hand researchers and help them pursue their interest far from the regular school climate. High school students in general are very curious for the creation of mobile apps; however, most of them get discouraged because they perceive this activity as a very difficult task. Here we describe the MobileDev summer school, a one-week training and hands-on experience in current topics of software development for mobile devices. The curiosity of the students for developing mobile apps is used to introduce and train CT via programming mobile applications through exercises (also with "pen and paper") of increasing difficulty. The school was repeated twice at our university and was targeted to a reduced class of students concluding the third and the fourth year of high school. Participants were in total 19 and coming from different types of schools. This paper describes the structure of MobileDev and discusses the results to provide directions for further research.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education

General Terms

Design, Human Factors

Keywords

Mobile development, Summer school, Computational thinking

SIGITE'15, September 30–October 3, 2015, Chicago, IL, USA.
© 2015 ACM. ISBN 978-1-4503-3835-6/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2808006.2808033.

1. INTRODUCTION

Computer Science (CS) can teach how computational processes can be applied on problems that occur during everyday life and involve many disciplines. To do so, a person needs to acquire the following skills [11]: 1) computer literacy, which is the ability to use basic computer applications, 2) computer fluency, which consists of understanding at a high level how computer systems work, and 3) Computational Thinking (CT) [15], which is "the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer - human or machine - can effectively carry out" [16]. Despite this, in the last decades CS curricula have often aimed at teaching computer literacy. This choice resulted almost inevitably in the identification of the concept of CS with the concept of "ability to use technology" [4].

CT is a key aspect of programming but is not unique to CS; instead, it can be applied to many other fields, including astronomy [5] and neurosciences [18]. CT is promoted as "one of the fundamental skills" for everyone (not just for computer scientists) [1, 15]. However, a small minority of people graduates from colleges and universities; therefore, the potential of any approach to integrate CT into the curriculum is limited by a focus on undergraduate education [12]. To address this issue, research has recently focused on defining curricula for teaching these competencies. However, gaps still exist that call out for empirical inquiries, in particular in the K-12 context [6]. Unfortunately, there are often issues related to introducing CT into schools [10], such as: 1) teachers should agree on changing their teaching style, 2) syllabi are very intensive and there is a worry of not being able to cover all the topics, 3) all the activities and projects should be decided ahead of time, and 4) teachers often change from one year to another. For these reasons, out-of-school venues are considered a possible solution since they provide great opportunities for exposure to CT, as students have the time to engage in challenging projects that are needed to foster CT. Moreover, "out-of-school environments can provide curricular flexibility, appropriate staff capacity, infrastructure access, and access to effective programs" [10]. In particular, summer classes have a more relaxed atmosphere. Moreover, students might attend different schools and, therefore, have different backgrounds and skills.

In this paper, we describe MobileDev, a summer school dedicated to high school students to learn CT via programming mobile applications. The idea is to use students' general curiosity for mobile development to foster CT and to convince them that, with these skills, mobile development can be accessible to them. Each phase of the software development process is used to foster specific skills of CT. The summer school has been organised in two editions at our University and was targeted to a reduced class of students concluding the third and the fourth year of high school (i.e., 11^{th} and 12^{th} grade of the K-12 structure). The paper is organised as follows: Section 2 provides background information; Section 3 details the structure of MobileDev; Section 4 discusses the results and provides directions for further research; Section 5 summarises this work and draws conclusions.

2. BACKGROUND

In the last years, CT has caught the attention of a broad academic community and many works have tried to capture the essence of CT and to create an agreed definition, as CT was a rather broad term. In 2011, the CS Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) suggested an operational definition of CT that provides a framework and vocabulary targeting K-12 educators (Table 1) [7]. In 2012, Brennan and Resnick developed another operational definition of CT that involves three key dimensions: computational concepts, computational practices, and computational perspectives [3].

Table 1: Skills improved by CT [7].

	Skill	Definition
1	Data collection	Gather appropriate information
2	Data analysis	Make sense of data, find patterns, and draw conclusions
3	Data representation	Depict and organise data in appropriate graphs, charts, words, or images
4	Problem decomposition	Break down a problem into smaller, manageable parts
5	Abstraction	Reduce complexity to define main idea
6	Algorithms and procedures	Find a series of ordered steps to solve a problem
7	Automation	Have computers or machines do repetitive or tedious tasks
8	Simulation	Run experiments using models
9	Parallelization	Organise resources to simultaneously carry out tasks to reach a common goal

CT is usually learned through programming; despite that, fostering CT should not be interpreted as an attempt to increase the number of professionals in CS. Programming is a key tool for supporting the cognitive tasks involved in CT [6], but there are other practices that let the students embrace and exercise CT skills. For example, teachers and students should use an appropriate vocabulary to describe problems and solutions. Moreover, they should learn to accept wrong solutions, to work as a team and to use decomposition techniques, abstraction, negotiations (to integrate multiple solutions), and consensus-building [2]. Students should also become familiar with the control flow of an algorithm, and be able to abstract and represent information. To this end, Lodi [8] suggests to propose exercises that require to visualise a "mental model" of the solution, so that students learn to think in terms of the program itself.

Graphical programming environments are probably the most popular tools to foster CT; such environments allow to program by combining blocks that are compatible with forms and provide graphical interfaces to control the actions of the different dynamic actors. Therefore, the simplicity and the intuitiveness of these tools (what you see is what you get) help to light the spark of curiosity and implant hunger in the students to want to learn more from the early sessions by focusing on design and construction, rather than dealing with syntax problems in programming.

Most recent research addressed the issue of CT assessment, to judge the effectiveness of any curriculum by measuring what children have learned [3, 14]. Large gaps, however, still exist that call out for empirical inquiries. Grade and age-appropriate curricula for CT still need to be designed or improved, also exploring the possibility to use computing as a medium for teaching other subjects. Moreover, most of research has been conducted in a undergraduate context; therefore, empirical studies in schools are needed to understand the types of problems faced during the first programming experiences that go beyond syntactical issues. Additionally, student attitudes toward computing should be explored [6].

2.1 App Inventor

The practical framework used in MobileDev, App Inventor[1], is a web based application created by the Massachusetts Institute of Technology (MIT). It is a visual development environment for building Android applications, and it is composed of two main views: 1) designer, which allows users to drag-and-drop all the components needed to design their application (i.e., graphical interface as well as all the non visual components such as sensors), 2) blocks, which allows to attach actions and add behaviour to the components defined in the designer. App Inventor is considered an attractive platform for engaging students at all levels in the computing curriculum [9]; therefore, effort has been spent in designing courses to teach programming using App Inventor [17]. Moreover, since it allows *problem driven learning* [9], this tool is a good framework to teach CT [13].

3. MOBILEDEV

MobileDev is a summer school on mobile development that provides a one-week classroom training and hands-on experience in current topics of software development for mobile devices. It is targeted to a class of high school seniors and it takes place at our University. The only requirement to enrol in MobileDev is to have just concluded the third or the fourth year of high school. No restrictions on the type of high school are given, to involve students from different backgrounds. This way, the environment of MobileDev can be more stimulating, as students can experience working in a multidisciplinary environment and accepting different points of view. When they are engaged in a discussion about the usefulness of CS, different backgrounds and opinions also help students to understand how CT (and programming) can be useful in their lives, even if they are not going to be professional programmers. Participation is free of charge. MobileDev has been organised in two editions, with a total participation of 19 students (4 female and 15 male) that covered a range of types of schools (Table 2).

[1]http://appinventor.mit.edu/explore/

Table 2: Participants' demographics.

High School Type	Number of Participants
Technical High School	6
Scientific High School	7
Liberal Arts High School	3
Linguistics High School	3

3.1 Goal

The goal of MobileDev is to provide students with the theoretical basis and the practical experience to develop simple applications for mobile devices (e.g., cellular phones or tablets) operated by the Android OS. In particular, MobileDev helps students learning the analytical thinking skills they need to develop mobile applications. To this end, MobileDev promotes conditions that foster the growth of skills and competencies of CT (Table 1). The educational objective to be pursued is to introduce and train CT via programming mobile applications through exercises (also with "pen and paper") of increasing difficulty [4]. The intuitive visual environment of App Inventor has been used as a trojan horse to introduce CT concepts to students.

3.2 Structure

MobileDev is a five-day (40 hours) training course starting on Monday morning and concluding on Friday afternoon. Even though the school counts on several hours of introductory lectures, it is mainly driven by laboratory exercises. By the end of the school, students are expected to present a final project that applies the content of the course. Table 3 presents the structure of MobileDev. The summer school is divided into three parts: i) theoretical knowledge, ii) learning the practical framework, and iii) application of the theoretical knowledge using the practical framework. In each part, particular importance has been attributed to organised work and documentation.

Table 3: Timetable structure.

Day	Morning	Afternoon
1	Introduction to CT with real world examples	
2	Introduction Operating systems and Android	Introduction to App Inventor and its toolset
3	Examples with App Inventor	Exercises with App Inventor
4	Examples with App Inventor	Working on projects
5	Working on finalizing the projects	Projects presentations

3.2.1 Theoretical knowledge

The first part of the school is dedicated to a discussion on the role of CS in our life and on its possible applications. Students are involved actively in the discussion and report about their experience and the usage of CS in their schools. Then, the CT skills required for software development are introduced. Students work on simple real examples in which they have to understand the problem itself and design a proper solution.

The teacher helps the students to exercise CT skills, proposing first non-software examples, evolving to programming-specific problems. To foster CT, even in the simplest example, students need to pay attention to the design of the solution. For instance, when students are asked to solve a simple operation, like preparing a sandwich. Students have to identify the problem, explain a high-level solution, and then model and structure an algorithm down to the smallest details. At times, students are surprised after noting that, according to their algorithm, they skipped steps that seem to be obvious for them (like "take a knife with the hand" before "spread mustard in the bread"), but then it is explained that when one needs to give precise instructions to a computer, there is no way to skip steps, even though they seem evident to a human.

3.2.2 Learning the practical framework

After a short introduction to App Inventor, students are guided to solve some simple exercises of increasing degree of difficulty, following the steps of CT (Table 1) and dedicating special attention to the design of the solution. For instance, an example proposed in the lecture consists of exploring the basic organisation of a graphic user interface. This example includes the visual aspects of a graphic user interface, but also requires an introduction on how to implement the basic functionality on a component like a button. Students are shown how to write a small method, and how to associate the "click" event of the button to the method. With the example, we create a functioning mobile application, which, even though with a basic behaviour, illustrates the capacity of the product to receive an input from the user, elaborate it internally, and produce an output. These concepts are then extended in examples and exercises of increasing complexity, up to the creation of the final project.

3.2.3 Application of the theoretical knowledge using the practical framework

To put together the theoretical and the practical knowledge acquired during MobileDev, students implement some of the examples introduced in the theoretical session in App Inventor. The goal we set for this phase is to assist them on learning the whole process of software development from defining requirements till delivering a working system. The next Section details the projects students worked on during the two editions of MobileDev.

3.3 Projects

After gaining knowledge on how to analyse the problems and design an appropriate solution using App Inventor, one day is dedicated to the development of a final project. Before starting to work on the technical details of the projects, students have the possibility to decide either to work in pairs or individually. In the two editions of MobileDev, we ended up having 3 pairs and 13 one person teams.

During all the steps of the project, we follow an Agile approach, as it facilitates flexibility and collaboration, and it fits our short time frame. The first step in the process starts by collecting projects' ideas. The students are free to choose whatever idea they want to develop and our role is to help them understand the level of complexity and feasibility. In the two editions of MobileDev, 13 student have proposed new ideas whereas 3 students have proposed extensions to exercises we did during the course. While the students were looking for the projects' ideas we observed that most of them were using the Internet to browse existing Android apps to get inspired.

Once the project idea is clear, ideas are developed through iterations. Each iteration is composed of the following steps:

- feasibility: an essential step is assessing the feasibility of the proposed features. Due to the lack of knowledge of some features of App Inventor as well as the limitation of App Inventor's environment, students are encouraged to look at the documentation and code examples to understand whether the features they want to implement are possible or not. Our role is to assist them when they can not find any solution;
- analysis: the analysis phase consists of looking at the features that will be implemented at the iteration and study their logical flow, the components needed to implement them and how to integrate them with the rest of the application;
- design: organising a working solution based on the analysed context. Identifying actors, roles and behaviours. In this phase, we assist students to set up a simpler design to eventually develop easier solutions. Moreover, in this phase students design with pen and paper the visual aspects of the application (Figure 1);
- development: developing the features by using the designer for the GUI and the non-visual components and using the blocks to connect the actions to the components;
- testing: running the newly developed features either on the emulator or on a real phone and check whether the features behave correctly or not;
- integration: this step does not apply to all cases, depending on the type of application. We have noted that in some cases features were developed separately and then they had to be integrated; in other cases, instead, each feature was built on top of existing ones, therefore, the integration was already done during development.

Table 4 shows how each of the phases of the software development process helps in fostering CT skills (Table 1).

Table 4: CT skills fostered during software development phases (phases are defined in Section 3.3).

	1	2	3	4	5	6	7	8	9
Feasibility	✓								
Analysis	✓	✓	✓	✓					
Design		✓	✓		✓	✓			
Development			✓			✓	✓		✓
Testing								✓	
Integration		✓						✓	✓

The kind of applications developed as part of the school span a large variety of types and goals. In general, a vast majority are applications of a moderate level of complexity that aim to provide interaction between the user and the application, that is, applications that react to a certain input from the user and provide a result. Examples of this kind of applications are games or quizzes. A second group of applications focused on exploiting the communication features of

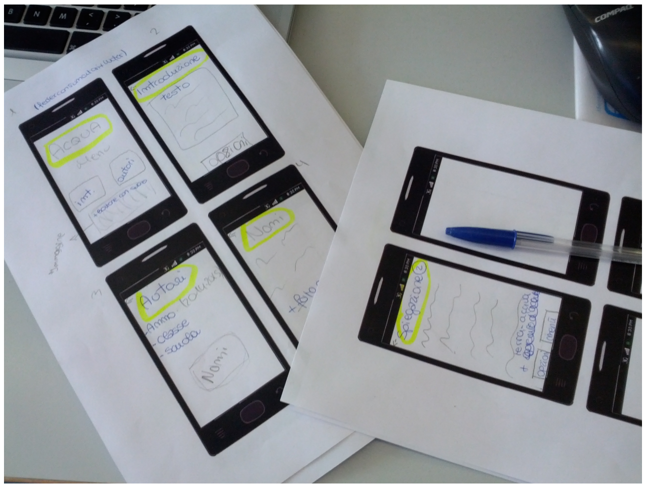

Figure 1: Mobile app design: manual sketches.

the cellular phone, creating interfaces for placing phone calls or messaging services. Finally, a smaller group of products display information unidirectionally, with a minimal interaction.

Internally, the applications can be relatively complex from the logic point of view. Developers are required to structure their solutions, identify and set the variables that are necessary to keep track of what is happening in the application, and to set the control blocks to guarantee the correct flow of execution of the application (for instance, conditions, loops, etc.). Moreover, developers should provide the means to guarantee the interaction with the user; this is facilitated by the native controls that App Inventor provides for the management of the input means of the cellular phones, for instance tap or swipe gestures, as well as the output channels like sounds, images, text, etc.

Table 5 shows the list of projects developed by participants; all the projects are available in the website of MobileDev[2].

3.4 Assessment

A framework was developed to assess the development of CT (i.e., practices, concepts, and perspectives), following the guidelines in [3]. Learners are engaged in a conversation about their progress, and at the same time their product is critically examined. For example, the following questions are asked: 1) what might you need to do next? 2) How would you fix it? 3) Have you used a loop? Where?. Answers can be used to provide help and to check that all the members of the team are progressing, even if with their own pace.

4. RESULTS

The quality and functionality of the final projects let us consider that the outcome of MobileDev is clearly positive. The diverse profiles of the participants show how, regardless of their background (see Table 2), students can exercise CT skills to walk through the process of identifying a problem to solve it in the form of a functional product. By using App Inventor, students implement their solutions without requiring specialised training on software development tools.

[2]http://www.mobiledev-summerschool.it

Table 5: List of projects.

Name	Description	Students
Flag Quiz	A game to challenge your knowledge of the European flags by naming a country after checking their flag	1
Rorschach Test	With this app, the user can generate images on perfect symmetry	1
English Phonetics	Application that uses the vocal synthesis library to propose exercises to improve the pronunciation of the English language	1
My Favorites	This app lets the user store bookmarks of web pages and then share them using social tools	1
Phone Manager	Application to compose a new SMS message or place a call using the keyboard or vocal synthesis	1
Automatic SMS composer	If you are busy, use this app to send back automatically a custom answer	1
CloneApp	An alternative version of an app to chat with other contacts	2
m-Card	An app to design mobile Business Cards	1
Colorful Circles	In this app, the user generates canvas with random circles of different colors	1
ShootGame	A classic "shoot-em-up" game, aiming an objective to shoot it	1
Snake	A clone of the Snake game, proposing a different version that introduces more challenges	2
Mole Mash (2 alternate versions)	A clone version of the Mole Mash game, to use your fingertips to hit a moving figure	1
Mobile Arkanoid	Clone of the Arkanoid game; a custom version for one player	1
Macedonia	A clone version of Fruit Ninja. The user creates a fruit salad by tapping over the fruits that appear randomly on the screen	2

Moreover, students are required to describe the architecture, design and implementation of their products, revisiting the customised development process that was followed during the summer school sessions. As a consequence, all participants explain their products in terms of a series of sequences expressed in blocks, which are carried out by the different GUI elements at hand.

Despite being time consuming, the assessment framework helped in revealing conceptual gaps. Moreover, participants did not perceive the conversations as an assessment; instead, they took advantage of those moments to get help or solve issues, and to this end they were happy to explain their work. Therefore, it is worth conducting more experiments to validate the assessment framework.

Using the assessment framework, we assessed the development of the three dimensions of CT; moreover, we checked that the projects were correctly functioning and responding to requirements. We are working on integrating these two types of assessment in future, by checking also if projects contain CT features [13]. A third type of assessment is represented by data on participants' satisfaction. The fact that participants were excited to show their friends and family what they created can be used as an informal indicator of satisfaction. Nevertheless, we are planning to somministrate pre and post surveys during future editions of MobileDev.

It is important to note that the informal environment created by MobileDev is an ideal atmosphere to boost the creativity of the participants, and to encourage their curiosity to discover, try and achieve, without being formally evaluated. Students are invited to identify problems, architect and design their solutions on their own, while the role of the teaching staff is to direct and organise their approaches, and advise on the feasibility of the projects using the available development tools.

Having participants of diverse backgrounds also is a resource of outmost importance, since students discuss problems from different standing points, brainstorm ideas from different approaches, and work in teams where the skills may complement each other. Moreover, the fact that the final product is a mobile application increases the interest and curiosity of students, as they belong to a range of age that makes strong use of devices like cellular phones or tablets.

MobileDev promotes gender balance and improves the participation of women in the summer school. A positive experience with CS is given to them during the MobileDev, and particular attention is given to overcoming the stereotype of the male "nerd". Girls' interest in CS is stimulated by showing how it can improve their life, in its social, working, medical and environmental aspects. Moreover, examples are given of women who have had success in Computer Science.

According to the informal feedback provided by students, a very appreciated value of MobileDev is the possibility of solving a problem by creating an actual product. This is underlined on the presentation of the final projects, when participants are able to experience the functional value of their products by executing and utilising them in the real platform, that is, in a mobile phone or tablet.

5. CONCLUSION AND FUTURE WORK

In this paper, we described the design and implementation of a short course to teach Computational Thinking at High School Level. The MobileDev summer school organised and carried out at our University successfully helped in the assimilation and exercising of CT skills, exposing students with little or no background in Computer Science to design and implement a mobile software application. From the first day, participants take an active role in analysing problems, architecting solutions, implementing software tools, and validating them. The accomplishments of the teaching program are reflected by the set of applications that students successfully develop working alone or in pairs, under the supervision of the teaching staff.

The impact of MobileDev has as well attracted the attention of the local K-12 Education Authority of our Province, who has expressed its support and encouragement for the realisation of a new edition of the summer school during 2015. Moreover, the organisation board of the summer school, in coordination with the local Education Authority, is exploring the possibility of recognising credit points to the participants; the credit points shall be granted based on the number of hours spent.

CT is a resource that is valuable for everybody, even for students who will not pursue a Computer Science major. Initiatives like MobileDev strive for creating a proper environment to exercise and implement CT skills, assisting participants to develop a working product that they may consider very proudly as their own creation.

6. REFERENCES

[1] ACM and IEEE. Computer science curricula 2013. Technical report, Association for Computing Machinery (ACM) and IEEE Computer Society, 2013.

[2] V. Barr and C. Stephenson. Bringing computational thinking to k-12: what is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1):48–54, Feb. 2011.

[3] K. Brennan and M. Resnick. New frameworks for studying and assessing the development of computational thinking. In *2012 Annual Meeting of the American Educational Research Association (AERA'12), Vancouver, Canada*, 2012.

[4] I. Fronza, N. El Ioini, A. Janes, A. Sillitti, G. Succi, and L. Corral. If i had to vote on this laboratory, i would give nine: Introduction on computational thinking in the lower secondary school: Results of the experience. *Mondo Digitale*, 13(51):757–765, 2014.

[5] J. Gray, A. S. Szalay, A. R. Thakar, P. Z. Kunszt, C. Stoughton, D. Slutz, and J. vandenBerg. Data Mining the SDSS SkyServer Database. In *Distributed Data and Structures 4: Records of the 4th International Meeting*, pages 189–210, 2002.

[6] S. Grover and R. Pea. Computational thinking in k–12. a review of the state of the field. *EDUCATIONAL RESEARCHER*, 42(1):38–43, jan/feb 2013.

[7] ISTE and CSTA. Computational thinking. teacher resources. second edition. http://csta.acm.org/Curriculum/sub/CompThinking.html, 2011. Accessed: Dec. 2014.

[8] M. Lodi. Imparare il pensiero computazionale, imparare a programmare. In *DIDAMATICA*, 2014.

[9] R. Morelli, T. de Lanerolle, P. Lake, N. Limardo, E. Tamotsu, and C. Uche. Can android app inventor bring computational thinking to k-12? In *Proc. 42nd ACM technical symposium on Computer science education*, SIGCSE'11. ACM, 2011.

[10] National Research Council. Report of workshop of pedagogical aspects of computational thinking. Technical report, National Research Council of the National Academies, 2011.

[11] L. Perković, A. Settle, S. Hwang, and J. Jones. A framework for computational thinking across the curriculum. In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '10, pages 123–127, New York, NY, USA, 2010. ACM.

[12] A. Settle, B. Franke, R. Hansen, F. Spaltro, C. Jurisson, C. Rennert-May, and B. Wildeman. Infusing computational thinking into the middle- and high-school curriculum. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '12, pages 22–27, New York, NY, USA, 2012. ACM.

[13] M. Sherman and F. Martin. The assessment of mobile computational thinking. *J. Comput. Sci. Coll.*, 30(6):53–59, 2015.

[14] L. Werner, J. Denner, S. Campe, and D. C. Kawamoto. The fairy performance assessment: Measuring computational thinking in middle school. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 215–220, New York, NY, USA, 2012. ACM.

[15] J. M. Wing. Computational thinking. *Commun. ACM*, 49(3), Mar. 2006.

[16] J. M. Wing. Computational thinking benefits society. *Social issues in computing*, jan 2014.

[17] D. Wolber, H. Abelson, and M. Friedman. Democratizing computing with app inventor. *GetMobile: Mobile Comp. and Comm.*, 18(4):53–58, 2015.

[18] F. Yong, S. Dinggang, and C. Davatzikos. Detecting Cognitive States from fMRI Images by Machine Learning and Multivariate Classification. In *Computer Vision and Pattern Recognition Workshop*, page 89, 2006.