Quizly: A Live Coding Assessment Platform for App Inventor

Francesco Maiorana\textsuperscript{1,2}  
\textsuperscript{1}Department of Electrical, Electronic and Computer Engineering, University of Catania, Viale A. Doria, 6, Catania, Italy  
\textsuperscript{2}I.I.S.S. G.B. Vaccarini, Catania  
francesco.maiorana@dieei.unict.it

Daniela Giordano  
Department of Electrical, Electronic and Computer Engineering, University of Catania, Viale A. Doria, 6, Catania, Italy  
daniela.giordano@dieei.unict.it

Ralph Morelli  
Computer Science Department  
Trinity College  
Hartford, CT, USA  
ralph.morelli@trincoll.edu

Abstract—There is a strong worldwide movement which is pushing for the teaching of serious computer science principles besides reading, writing and basic numeracy starting from first grade and reaching all students across all grades. This is being done through both formal initiatives carried out by international organizations and at the national level by putting forward curricula, some of which are mandatory. In order to accomplish this goal, and based on the consensus that computer science is not programming and that programming languages are a tool, visual languages have become the preferred method for teaching introductory courses in computer science. The absence of rigid syntactic rules makes them the ideal tool for focusing on problem solving and computational thinking activities. Recent reports have pointed out the need for supporting the international community of teachers by providing assessment methods, an internationally validated question repository as well as tools and assessment platforms. In this context our work presents an assessment platform for formative, summative and informal assessment of computer science competencies by using a visual language, namely App Inventor, which allows for the rapid development of a mobile app and has a strong appeal to the younger generation of students. The capability to log user activity allows the teacher to monitor the progression in the student’s learning path as well as her/his solution-building approach.

Keywords—Computer Science curricula, Visual languages; assessment; assessment tools and platforms

I. INTRODUCTION

There is a strong worldwide movement advocating for the teaching of serious computer science concepts, besides reading, writing and numeracy, as early as possible in the student’s learning path. This has brought to international attention the need to reform computer science teaching practices in order to support its introduction in all stages of school and for all students. A remarkable example of this trend can be found in the United Kingdom when since 2014 computer science has been considered mandatory from the first year of school \cite{1}, and in the United States where a strong informal movement (https://hourofcode.com, https://code.org/learn) \cite{2} is reforming the attitude towards computer science worldwide by engaging millions of people by way of games and activities with the aim of pushing a legislative reform to introduce computer science in all schools. The movement has flourished in many similar initiatives across the globe such as http://codeweek.eu/, http://uk.code.org/, http://codeweek.it/, and http://www.programmailfuturo.it/. These curricula are aligned with modern 21st century competency frameworks such as [3-6] and the Framework for 21st Century Learning developed by the P21 Partnership for 21st Century Learning (http://www.p21.org/our-work/p21-framework). One of the key elements of these efforts is the importance given to visual programming languages. Even if it is well recognized that computer science is not programming, visual languages allow for a programming approach based on snapping together blocks that differ in shape and color, thus avoiding much of the syntactic burden common to textual languages. This in turn allows students to focus on the problem solving process and sharpen their computational thinking skills \cite{7-10}.

Besides this flourishing of curricula there is an urgent need for an assessment in computer science (CS) education, which is aligned with the modern curricula and competency frameworks as stated recently in \cite{11} which recommends that the “CS education community leads the development of a curated assessment library for teachers” and launches a “call to action for the computer science education community to develop valid and reliable assessments.” The importance of assessment is recognized by other international networks such as the assessment and accountability roadmap (http://www.roadmap21.org/assessment.html). The assessment has been the focus of many working groups, such as the Conference on Innovation and Technology in Computer Science Education (ITICSE), which has a long tradition in computer science assessment at the university level \cite{12}, with a proposal to also cover computer science assessment in schools \cite{13}. This work will present two tools: 1) Quiz Maker for the creation of quizzes, and 2) Quizly for assessing and automatically grading exercises done through the visual language App Inventor \cite{14}, with its extension into an assessment platform able to manage students and classes, administer and propose formative, summative and informal tests and to be able to track user progress as well as the question solving process. The work is organized as follows: Section 2 reviews the state of the art and is organized into three main subsections: Computer Science Curricula, Visual Languages, and Assessment and Assessment Tools; Section 3
II. STATE OF THE ART

A. Curriculum development

In 2013 the United Kingdom’s Department of Education released a mandatory computer science curriculum, beginning in the 2014 scholastic year and starting from the first year of school at Key Stage 1 up to Key Stage 4. Since this regulation was passed, national associations such as Computing At School (CAS) has put forward a set of guidelines for primary [15] and secondary teachers [16]. The curriculum guidelines have been organized into two progression pathways: the first concerns topics, namely: Algorithms, Programming & Development, Data & Data Representation, Hardware & Processing, Communications & Networks, and finally Information Technology with the topics arranged by difficulty level. The second progression pathway is for strands: Computer Science, Information Technology and Digital Literacy. During the first three Key Stages, the focus of the teaching and learning activities should be on the computational thinking aspects and at Key Stage 4 students can choose between the Computer Science and the Information Technology strands. The CAS network has developed a QuickStart Computing Guide for primary and secondary teachers [17] [18], which can be found at http://www.quickstartcomputing.org/. This is a general curriculum suited for all students and does not consider any specialized program of study focusing on computer science.

In the USA the three main curricula are organized around the Computer Science Principles framework [19], the Computer Science Teacher Association (CSTA) curriculum, which is currently under review [20], and the Computer Science Equity Alliance curriculum [21] as well as other state level curricula. The AP Computer Science Principles Curriculum Framework is equivalent to a first semester introductory computer science course at the college level. It is organized around seven computational thinking practices: Connecting, Computing, Creating Computational Artifacts, Abstracting, Analyzing Problems and Artifacts, Communication and Collaboration; it represent the competences that students must have. The areas of the course are organized around seven big ideas: Creativity, Abstraction, Data and Information, Algorithms, Programming, The Internet, and Global Impact. Each big idea is associated with a set of essential questions useful for guiding students in finding connections to the content of the big ideas; it contains enduring understanding which specifies core concepts that have to be mastered by the students. Each enduring understanding is aligned with Learning Objectives (LO) that provide a more detailed articulation of what the students should be able to do. These objectives integrate computational thinking practice with specific content. Next to each LO there is a list of essential knowledge statements which specify facts or content which has to be mastered by the students. The assessment is done through two performance tasks: Explore – Implications of computing innovations; Create applications from ideas. An end of course assessment with a time restriction completes the assessment process. The performance task takes longer but allows students to demonstrate a diverse skill set, beginning with creativity and should engage them in real work and motivating tasks which may be socially relevant. Students can present their work in an online portfolio, such as a website.

The general framework has been implemented into several courses, notably the Mobile Computer Science Principle course both for students as well as professional development for teachers [22-24], and the Beauty and Joy of Computing [25]. For an overview of the course approach the reader can refer to [26 – 28].

The CSTA curriculum is organized into three levels: the first for grades K1-K6, the second for grades K6-K9 and the third for grades K9-K12. The latter is divided into three courses: Computer Science in the Modern World (K9-K10), Computer Science Concepts and Practices (K10-K11), and Topics in Computer Science (K11-K12), with each course intensifying in depth and material. Each level is organized into five strands: Computational Thinking, Collaboration, Computing Practice and Programming, Computers and Communication Devices, and finally Community, Global and Ethical Impacts. The curriculum describes the specific computer science concepts and skills associated with each strand that has to be mastered by the student at each level.

The Computer Science Equity Alliance curriculum was developed by combining computer science content and computational practice and has been mapped to several standards (http://pact.sri.com/index.html). The curriculum is shaped around four main elements: Curricular Material, Professional Development, Assessment (forthcoming) and Local Policy Support, with the aim of teaching the creative, collaborative, interdisciplinary and problem solving nature of computing. It is organized into six units: Human Computer Interaction, Problem Solving, Web Design, Introduction to Programming, Computing and Data Analysis, and Robotics. Each unit has daily lesson plans with student activities and teaching strategies. The lessons are developed to reinforce the three main themes of the curriculum: the creative nature of computing, technology as a tool for solving problems, and the relevance of computer science and its impact on society. For a wider overview of the curriculum, the reader can refer to two recent special issues [29-30] and a review about high school curricula [31] as well as an international effort on reforming higher education curricula in Computer Science and Information Technology [32] to grasp the worldwide push to introduce computer science as early as possible and along as many learning paths as possible.

B. Learning environments for initial programming

Visual languages have become a common choice for an initial programming environment suitable for an introductory course in computer science. Examples of these include: Scratch [33] which is suitable for an introductory approach to
computing and for primary and lower middle school students. App Inventor [34] which allows the student to build apps for mobile devices and enables him/her to manage a rich set of components that respond to events. It is suited for middle and high school students. BYOB/SNAP [35] which builds on top of Scratch and allows students to practice with an even richer set of computer science concepts such as “procedures as first class data, from the Scheme language”. Enchanting (http://enchanting.robotclub.ab.ca/tiki-index.php) which is built on top of BYOB/SNAP. It offers a programming environment for LEGO Mindstorms robots. Robotics has been seen as a way to introduce and engage students in STEM education [36] with its rich set of initiatives, such as the collaborative robotics programming competition [37] performed either by a visual language automatically translated in C or directly in the C language. Flip [38] which combines a visual editor with a natural version of the scripts, thus allowing students to use natural language as a means of improving code comprehension as well as their computational communication skills.

C. Assessment and assessment tools

In this rapidly changing environment of competency frameworks, which focus on key competencies that students have to acquire in order to succeed in modern working environments, and curricula that are issued worldwide there is an urgent need for good assessment practices aligned with the new frameworks and curricula. Assessments should be seen as powerful learning tools in order to teach students the skills needed to prepare their questions [39] and to guide them in asking the right questions during their lessons [40]. Beside this it is necessary to have assessment tools which allow for an accurate and quick verification of student performance with immediate feedback, thus fostering self-reflection. Several useful tools exist for visual block languages, such as:

- Scrape [41] and the related set of tools that allow for the analysis of single/multiple Scratch projects with statistics such as the number of blocks or the types of computational constructs used.
- Scratch Explorer [42], a tool that depicts relationships between different parts of the program.
- Hairball [43] which is a system that can be useful to students and teachers. For students it can point out potential errors or unsafe practices while allowing teachers to inspect Scratch programs.
- Dr. Scratch [44][45] which is a free/open source web tool that allows for the analysis of Scratch projects by automatically assigning a Computational Thinking score in terms of abstraction, logical thinking, synchronization, parallelism, flow control, user interactivity and data representation. It can be used by students for self-assessment as well as by teachers.
- REACT (Real-Time Evaluation and Assessment of Computational Thinking) [46] is a tool that provides a teacher with a sortable dashboard showing the characters used as well as the semantic meaning behind what students have programmed in AgentSheets [47].

The basic approach of these tools is an insightful static analysis of the type of blocks used, an analysis that leads to the detection of bad programming practice, potential problems, mastery of computational thinking concepts by analyzing the type and number of blocks used, something that offers teachers a dashboard for summarizing students' projects even in groups. Our work will describe a powerful platform that allows for assessment by means of live coding using the App Inventor visual language which also offers an automatic check of the solution, giving freedom for a rich problem space thus engaging the students in different types of problems, from ordering a set of blocks to producing a complete solution to a problem using all the blocks available on App Inventor and by posing questions in different types of formats.

III. QUIZ MAKER AND QUIZLY

Quiz Maker is a web-based tool that allows teachers and students to create questions based on the App Inventor language. Its companion tool, Quizly, allows teachers to assess their students through formative, summative and informal means while allowing students to play with challenging questions using a live coding platform which provides hints and real-time automatic feedback. The tool is built upon Blockly (https://developers.google.com/blockly/), a library for building visual programming editors. The two apps are available at https://github.com/ram8647/quizly. The interface of Quiz Maker is shown in Fig. 1. The tool allows for the creation of questions using the following workflow, as is clearly described in the Quiz Maker site:

- Select the type of quiz. Fill in the name of the quiz and a brief description. Compose the quiz question, including HTML and hints, then select the built-in and App Inventor component blocks, i.e., those needed to solve the problem.
- The Quiz tab sets up the starting blocks.
- The Solution tab constructs the solution to the problem.
- The Preview tab allows the participant to try the quiz.
- The JSON tab generates the quiz as a JSON string, which is saved in the back end database.

The types of questions that can be generated range from building a solution with App Inventor to writing the answer to the given question in a text box. By providing clear and careful instructions in the questions, the solution provided by the students can be automatically compared to the reference solution, thereby providing immediate feedback and offering a means both for self-learning and automated assessment. The Quizly interface is shown in Fig. 2. The quiz generated using Make Quiz can be executed with Quizly. The student has the option to choose a quiz. The available quizzes, which are within the assessment platform, are assigned by the teacher to the student, a group of students, or to the class. When the student chooses the quiz, its description appears. The question can be associated with a tutorial related to the problem at hand. The quiz generated using Make Quiz can be executed with Quizly. The student has the option to choose a quiz. The available quizzes, which are within the assessment platform,
are assigned by the teacher to the student, a group of students, or to the class. When the student chooses the quiz, its description appears. The question can be associated with a tutorial related to the problem at hand.

The student has the option to request hints that should be as much as possible inquiring questions for scaffolding student's discovery of the solutions. When the student is satisfied with the answer, he/she can submit it and obtain an immediate response which informs him/her whether or not the answer is correct. This is done by comparing the student’s solution with the solution prepared by the teacher when the question was first created. The Quizly interface shown on the left pane of Fig. 2 has been embedded into an assessment platform, which is described in the next section. The left pane includes other information, summarizing her learning journey by displaying the list of questions with an indication of the number of errors and correct solutions submitted for each question.

IV. THE ASSESSMENT PLATFORM

The two tools, Quiz Maker and Quizly, are embedded into an assessment platform which was developed with a three tier architecture using AngularJs (https://angularjs.org/) as a client scripting language, PHP as a server side scripting language and MYSQL as the underlying database. The platform allows teachers to manage the assessment activities of their classes and to monitor their students’ progress. The platform allows for three types of users: administrators, teachers and students.

The main functionalities available to them are shown in Fig. 3. The whole platform will be available on GitHub.

The administrators have the ability to create a class and to create teachers that are associated with a class; they can also manage the level of difficulty of the questions. Since the assessment platform manages multiple questions and multiple classes, when the teacher assembles a test he/she can assign a certain level of difficulty to each question depending on the level of the class. Each level of difficulty has an associated weight given to the number of errors and the number of hints. These weights are used to automatically assign a grade to the final answer submitted by each student. The level of difficulty is assigned by the teacher to each question assembled in a test created for a class. The teacher has the ability to manage:

- new groups inside the class, e.g. reinforcement or advanced groups within the same class.
- new questions, even uploading their JSON description from a text file. The teacher can try the questions inside the Quizly application using “Run questions.”
- tests composed of a set of questions. The test could be used as a learning tool, e.g. guiding the student in the construction of a game by designing and developing each stack of blocks inside each question. As a further avenue of research we envisage using the platform to create tests, seen as fun and engaging activities where the students, by solving each question, proceed in the development of a small application. The tutorial that is already linked to each question can offer scaffolding material. Since Quiz Maker and Quizly allow for the use of both the control structures and the components of App Inventor, the teacher has the complete freedom to build questions and tests that guide the student in developing a reach set of activities that could also be engaging. The test can be assigned to a class or to a group of students.
- view statistics either for each single student or for an entire class. Statistics for class results range from max,
min, average as well as standard deviation of the grades, including graphics for statistics related to the
questions inside the test, e.g., the simpler, more
difficult, or more time-consuming questions, with the
max. number of suggestions and detailed statistical
results for each question. Similar statistics are available
for each student. The platform automatically alerts
teachers when the activities of a student fall below
her/his average, e.g., frequency of test completion.
- create abilities. The abilities can be created on the basis
of an extendible taxonomy, such as [48 – 50] or 21st
century skills, or skills associated with progression
pathways. The abilities can be associated with each
question within each test assigned to a given class and
be arranged in various levels (from beginner to
master). Further development is necessary.

The student can answer questions and take tests assigned to
her, view her results and statistics as well as check her abilities.

V. LOGGING USER ACTIVITIES

The logging mechanism of the assessment platform has
been extended in order to log the activities related to the block
movements, which will allow for insight into the student’s
solution building process. In particular, the platform allows the
following activities to be logged: inserting a block in the
workspace, canceling a block from the workspace, connecting
or disconnecting two blocks as well as the activities of
requesting a hint and submitting an answer. All the activities
are logged with a time indication so that inference on the
student’s solution building process can be made on the basis of
objective parameters such as time, number of trials and so on.
The number of clicks and block connects and disconnects with
the related statistical metric can be the starting point of these
analyses. An in-depth analysis of the different types of blocks
used in developing the solution requires a careful design and
will be considered in further research. All the collected
information is permanently stored in the database.

VI. CONCLUSION AND FURTHER DEVELOPMENTS

In this paper we have presented an assessment platform
based on a live coding web-based tool for App Inventor with
real-time feedback and automatic grading. The platform
allows teachers to manage classes and groups of students,
create questions and tests, assign them to students, track their
performance, detect learning paths as well as the progress of
the course and to log the activities related to the solution
construction in order to gain insight into the solution building
process, thereby detecting user difficulties. As future work we
plan to extend the types of questions [51] that can be asked
and used with the platform and their format, enlarge the set of
blocks that can be used in the assessment platform by using
the building block capabilities of Blocky, which allows for
constructs from other languages such as SQL [52]. This can be
further developed by designing blocks and environments
which will allow for the assessment of design capabilities
related to Flowchart [53], an Entity Relationship model or a
Unified Modeling Language diagram such as Class Diagram.

The basic graphics element of each design language can be
used as a basic building block which can be snapped together
in order to construct the correct design. The platform can also
be extended, allowing digital ink [54] in the assessment
process and in the annotation of both the questions and the
grading/commenting process by the teacher. Furthermore the
assessment platform is currently being validated by courses
both at the school and at the university levels [55] as well as
in professional development courses for teachers, such as the
mobile computer science principles course (http://mobile-
csp.org/). The questions have been used to assess the enduring
understanding, learning objectives and essential knowledge
described in [19]. Ways on how to integrate the questions and
activities developed inside Quizly in the App Inventor
environment will be explored.

ACKNOWLEDGMENT

This work was supported in part by National Science
Foundation grants CNS-1240841 and TUES-1225976.

REFERENCES

programmes of study. September 2013 Retrieved on-line from
https://www.gov.uk/government/publications/national-curriculum-in-
england-computing-programmes-of-study
[3] M. Binkley, O. Erstad, J. Herman, S. Raizen, M. Ripley, M. Miller-
Ricci, and M. Rumble, “Defining twenty-first century skills”, In
Assessment and teaching of 21st century skills, 2012 (pp. 17-66).
Springer Netherlands.
Delors, J. (1996). Learning, the Treasure Within: Report to UNESCO of
the Internationl Commission on Education for the Twenty-First Century.
UNESCO.
[5] J. Gordon, G. Halász, M. Krawczyk, T. Leney, A. Michel, D. Pepper, ... and
J. Wiśniewski, “Key competences in Europe: opening doors for
lifelong learners across the school curriculum and teacher education”,
CASE network Reports, (87) 2009.
new millennium learners in OECD countries.
computational thinking through programming: What is next for K-12?,
Sowing the Seeds: A Landscape Study on Assessment in Secondary
Computer Science Education. CSTA annual conference 2015.
C. Johnson, ... and J. Spacco, The Canterbury questionbank: Building a
repository of multiple-choice CS1 and CS2 questions. In Proceedings of
the ITiCSE working group reports conference on Innovation and
technology in computer science education-working group reports,
computer science questions, visual assessment tools and metadata
annotations” Working group proposal submitted at the 20th Conference
on Innovation and Technology in Computer Science Education (ITICSE 2015).


