On the Automatic Assessment of Computational Thinking Skills: A Comparison with Human Experts

Jesús Moreno-León
Programamos & Universidad Rey Juan Carlos
Seville, Spain
jesus.moreno@programamos.es

Casper Harteveld
Northeastern University
Boston, MA, USA
c.harteveld@neu.edu

Marcos Román-González
Universidad Nacional de Educación a Distancia
Madrid, Spain
mroman@edu.uned.es

Gregorio Robles
Universidad Rey Juan Carlos
Fuenlabrada (Madrid), Spain
grex@gsyc.urjc.es

CHI'17 Extended Abstracts, May 6–11, 2017, Denver, CO, USA.
ACM ISBN 978-1-4503-4656-6/17/05.
http://dx.doi.org/10.1145/3027063.3053216

Abstract
Programming and computational thinking skills are promoted in schools worldwide. However, there is still a lack of tools that assist learners and educators in the assessment of these skills. We have implemented an assessment tool, called Dr. Scratch, that analyzes Scratch projects with the aim to assess the level of development of several aspects of computational thinking. One of the issues to address in order to show its validity is to compare the (automatic) evaluations provided by the tool with the (manual) evaluations by (human) experts. In this paper we compare the assessments provided by Dr. Scratch with over 450 evaluations of Scratch projects given by 16 experts in computer science education. Our results show strong correlations between automatic and manual evaluations. As there is an ample debate among educators on the use of this type of tools, we discuss the implications and limitations, and provide recommendations for further research.

Author Keywords
Computational thinking; programming; assessment; Scratch

ACM Classification Keywords
K.3.2 [Computer and Information Science Education]: Computer science education; D.2.4 [Software/Program Verification]: Validation

Level of development for each CT dimension assessed by Dr. Scratch [16]

I. Logical Thinking:
1. If
2. If else
3. Logic operations

II. Data Representation:
1. Modifiers of object properties
2. Variables
3. Lists

III. User Interactivity:
1. Green flag
2. Keyboard, mouse, ask and wait
3. Webcam, input sound

IV. Flow Control:
1. Sequence of blocks
2. Repeat, forever
3. Repeat until

V. Abstraction and Problem Decomposition:
1. More than one script and more than one sprite
2. Use of custom blocks
3. Use of 'clones' (instances of sprites)

Introduction

In the last years programming is being promoted in schools worldwide [2, 9], as part of a movement that aims to promote Computational Thinking (CT) skills among young learners [3]. CT is the process involved in formulating a problem and expressing its solution so that a computer can effectively carry it out. It is based on an iterative process with three stages: formulation of a problem (abstraction), expression of a solution (automation, which includes its implementation by means of programming), and execution and evaluation of a solution (analysis). The term computational thinking was first used by Papert [19] in the 1980s, although it has been popularized recently by Wing [24].

Many efforts have been placed in the creation of tools to teach these skills, such as Alice [7], Kodu [14] or Scratch [20], a block-based, visual programming environment with more than 13 million users and over 16 million projects shared in its open online repository. However, not much attention has been put so far in the development of learning support tools, such as assessment and recommender systems. In the opinion of the authors, learners and educators can be supported with tools that help and guide them, in the same way that professional software developers benefit from using tools (e.g., the ones that analyze their code such as `lint`-like tools and other types of checkers that analyze code quality [12, 23]). That is why we have built Dr. Scratch [16], a free/libre/open source webtool that allows to analyze Scratch projects to assess the level of development in several aspects of computational thinking – such as abstraction and problem decomposition, data representation, user interactivity, parallelism, synchronization, logical thinking and algorithmic notions of flow control – by statically inspecting the source code of the projects.

Although already being used by several thousands of learners and educators monthly, the automatic assessment provided by Dr. Scratch has not been completely validated in a scientific manner. For this reason we explore the following question in this preliminary study: *What is the correlation between the automatic score provided by Dr. Scratch and the one provided (manually) by expert evaluators?* We address this question with the analysis of the results of over 450 evaluations of Scratch projects by 16 experts acting as members of the jury for a programming contest with over 50 participants. In consequence, in terms of contribution to Human-Computer Interaction research, the general implications of this investigation are for building automatic assessment tools, in particular for educational programming environments.

The structure of the paper is as follows: in the next section, we offer some background information on Dr. Scratch, the automatic assessment tool. Then, we present the methodology used in our comparison. We end with discussing our results, including their limitations and possible paths for future research.

Background

Dr. Scratch is inspired by Scrape [25], a visualizer of the blocks used in Scratch projects, and is based on Hairball [4], a static code analyzer for Scratch projects that detects potential issues in the programs, such as code that is never executed, messages that no sprite receives or object attributes not correctly initialized [8].

The Hairball architecture, based on plug-ins, is ideal to add new features. For instance, Hermans *et al.* have studied how *bad smells* affect negatively the learning process [10, 1]; we have thus developed two plug-ins to detect several bad programming habits (i.e., *bad smells*) educators fre-

Level of development for each CT dimension assessed by Dr. Scratch [16]

VI. Parallelism:
1. Two scripts on green flag
2. Two scripts on key pressed or sprite clicked
3. Two scripts on receive message, video/audio input, backdrop change

VII. Synchronization:
1. Wait
2. Message broadcast, stop all, stop program
3. Wait until, when backdrop changes, broadcast and wait

Figure 1: Source code of 'Catch me if you can'. Available at https://scratch.mit.edu/projects/138397021/

quently detect in their work as instructors with high school students [15].

However, the fact that Hairball is run from the command-line, as it is a set of Python scripts that evaluators have to manually run, makes it not suitable for many educators that are not confident with such an environment, let alone for young students. For this reason, we decided to create a web-based service called Dr. Scratch that facilitates the analysis of Scratch projects.

In addition, we reviewed prior work proposing theories and methods for assessing the development of programming and CT skills of learners [22, 21, 5, 13] to come up with a CT score system. As summarized in the sidebar, the CT score that Dr. Scratch assigns to projects is based on the degree of development of seven dimensions of the CT competence: abstraction and problem decomposition, logical thinking, synchronization, parallelism, algorithmic notions of flow control, user interactivity and data representation. These dimensions are statically evaluated by inspecting the source code of the analyzed project and given a punctuation from 0 to 3, resulting in a total evaluation (mastery score) that ranges from 0 to 21 when all seven dimensions are aggregated. Figure 1, which shows the source code of a single-script Scratch project, can be used to illustrate the assessment of the tool. Dr. Scratch would assign 6 points of mastery score to this project: 2 points for flow control, because it includes a *forever* loop; 2 points for user interactivity, as players interact with the sprite by using the mouse; 1 point for logical thinking, because of the *if* statement; and 1 point for data representation, since orientation and position properties of the sprite are modified. The rest of the CT dimensions would be measured with 0 points.

As part of the initial validation process of Dr. Scratch, workshops have been organized in schools and high schools to study the use that students of different ages and with diverse backgrounds make of the tool [16]. The scores of Dr. Scratch have been as well compared with other classic software engineering complexity metrics [17]. The results of these investigations have shown to be very promising and justify additional validations, such as the one presented in this paper.

Making an analogy between writing programs and writing text, this current study is remarkably similar to an investigation that examined the reliability of a system for automatic evaluation of written texts [18]. In that study two judges independently assessed 78 summaries written by students. These judges' scores were averaged and then were compared with the assessments provided by an automatic system, finding a correlation $r = +.82$.

Methodology

In order to answer our research question, we organized a programming contest for Spanish students during the months of October to December 2015 in collaboration with the Spanish Foundation for Science and Technology (FECyT) and Google.

The main objective of this contest was to promote scientific vocations among youngsters. Participating students, from primary and secondary education, had to create a Scratch project explaining a scientific concept. To stimulate and encourage students, the following questions were provided in the instructions of the contest: "Could you come up with a great way to explain the water cycle with Scratch? Or perhaps you can create a project to present the planets of the solar system? Or maybe you can author a game to help us know more about the noble elements?". Although 87 projects were submitted, only 53 of them met the criteria to

Figure 2: S: a game in which players have to place a ball on a base by interacting with its speed, height, kinetic and potential energy. Available at https://scratch.mit.edu/projects/88281683/

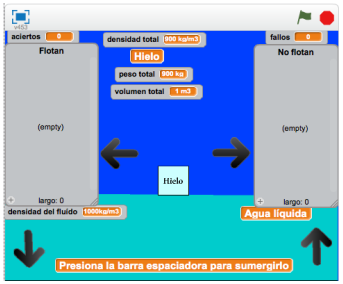

Figure 3: To float or not to float: a game in which players have to guess if different objects would float on diverse liquids based on their density. Available at https://scratch.mit.edu/projects/85723084/

participate[1]. Figures 2 and 3 show screenshots of two of the projects accepted for the contest, in which concepts related to the energy of objects in movement and density for solids and liquids have been respectively included.

The jury who evaluated the Scratch projects was formed by 16 specialists from different backgrounds with a solid knowledge of computer science education: policy makers, non-profit organizations promoting CT skills in schools, primary and secondary teachers, researchers and companies with programming and robotics programs.

We formed four groups of experts based on their experience with Scratch, so that the average years of experience with Scratch ranges from 3.5 to 4 years. Then, we divided the projects assigned to each group in a way that each project was at least assessed by six experts (at least by three experts from two groups). The evaluations were performed in two weeks, with a different approach in each of the weeks. With the aim of not influencing the experts' open evaluation during the first week, we did not offer many guidelines, so as to let them perform the evaluation in a very open way based on criteria that they, as experts, deem appropriate. During the second week we asked them to grade the projects based on the criteria we chose. So, during the first week, experts were asked to evaluate a set of projects filling out a form with only two fields: an overall score and general comments. During the second week, we asked them to evaluate a different set of projects and to provide, in addition to the overall score, an assessment for the technical mastery of the project, its creativity and originality, and the use of aesthetic and sensory effects. All the previous fields could be punctuated from 1 to 10, and a text box allowed to add comments if desired. Instructions and recommendations given to experts were minimal to not bias their assessment as we wanted them to establish the criteria they considered more appropriate based on their experience. We recorded 317 overall score evaluations and 160 technical complexity evaluations of the 53 projects that were considered valid. Some experts did not complete their "assignments"; on average we have around 6 global evaluations per project and 3 technical mastery evaluations per project. These have been compared with the score provided by Dr. Scratch for each of the projects.

A list with the URLs of the Scratch projects, information on the experts (name, affiliation and experience with Scratch), how we grouped the experts, an English translation of the emails sent to experts, the assessment questionnaires, and the results of the evaluations are publicly available in the replication package of the paper[2].

Findings

In this section we study the relationship between Dr. Scratch scores and the evaluations provided by experts, both for the overall score and the technical mastery. Dr. Scratch does not assess creativity or aesthetics, and these aspects are therefore out of the scope of this work. Even though the variables are considered quasi-interval, since data does not behave according to a normal distribution for all the variables, the analysis is based on Spearman's rho non-parametric correlations.

Relationship between Dr. Scratch score and experts' overall score
The 317 evaluations from experts collected in weeks 1 and 2 had scores that ranged from 1 to 10 points. These have been compared with the ones provided by Dr. Scratch,

[1]The two requirements to participate were: i) projects had to be publicly shared in the repository, and ii) they should be, at least vaguely, related to some scientific concept.

[2]https://github.com/kgblll/ReplicationPackage-2017-CHI-LBW

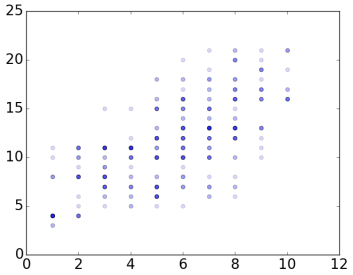

Figure 4: Scatter plot for experts overall evaluation (x-axis) and Dr. Scratch assessment (y-axis). Darker points represent a higher number of cases (i.e. more evaluations).

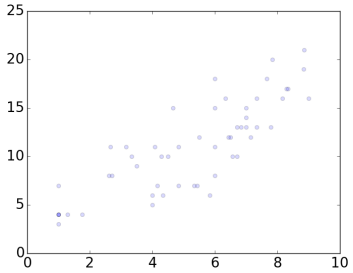

Figure 5: Scatter plot for the mean of experts overall evaluation (x-axis) and Dr. Scratch assessment (y-axis). Darker points represent a higher number of cases (i.e. more evaluations).

Statistics	Value
mean	1.121
min	0
25%	0.898
50%	1.072
75%	1.380
max	2.250

Table 1: Summary statistics (mean, minimum, maximum and quartiles) of the standard deviations of the overall evaluations provided by experts.

ranging from 0 to 21 points, resulting in a strong correlation: $r = .682$, $p(r) < .0001$. Figure 4 presents the scatter plot for this relationship, where a monotonic direct correlation between Dr. Scratch scores and experts' evaluations is depicted.

Scores provided by the experts, with different backgrounds and expertise, are not completely uniform. Table 1 shows the summary statistics of the standard deviation of the evaluations for each project, where we can see that there are projects where experts coincided in their assessments (min = 0). In other cases the standard deviation is as high as 2.250 points, being the mean of the standard deviations $M = 1.121$.

If we compute the mean of the evaluations of the experts for each of the projects, and then compare them with the Dr. Scratch scores, a strong correlation is found: $r = .834$, $p(r) < .0001$. The monotonic positive correlation between variables is clearly shown in Figure 5.

Relationships between the Dr. Scratch score and the experts technical mastery score

During the second week of the investigation, members of the jury were asked to evaluate the technical mastery of the projects. If we compare the 160 evaluations with the Dr. Scratch score, a strong correlation is found: $r = .779$, $p(r) < .0001$. Figure 6 presents the scatter plot for this relationship, showing a monotonic direct correlation between variables.

If the means of the expert evaluations are computed and compared with the Dr. Scratch scores, the correlation is stronger: $r = .824$, $p(r) < .0001$. As can be seen in Figure 7, which presents the scatter plot of the relationship, there is a positive correlation between these measurements.

Discussion, limitations and future research

The organization of a programming contest has allowed us to compare the scores of an automatic assessment tool, Dr. Scratch, with the evaluations of a group of experts in computer science education. This analysis showed strong correlations, which could be considered as a validation of the metrics used by the tool. We argue, in consequence, that this investigation represents a step in the validation of Dr. Scratch as a tool to support learners, educators and researchers in the assessment of programming and CT skills.

When expert's evaluations are considered individually, the relationship with Dr. Scratch assessments is stronger for the technical mastery of the projects than for the overall score. However, when project evaluations are averaged, the relationship is slightly stronger for the overall score, although the difference is minor. This is explained by the strong relationship between overall and techical scores; in fact, an ad-hoc calculation of this relationship indicates a

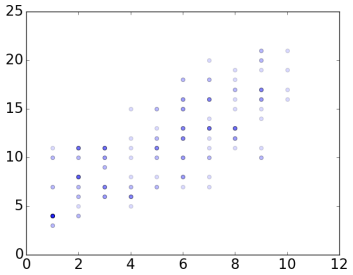

Figure 6: Scatter plot for experts technical mastery evaluation (x-axis) and Dr. Scratch assessment (y-axis). Darker points represent a higher number of cases (i.e. more evaluations).

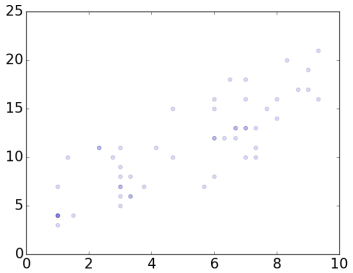

Figure 7: Scatter plot for the mean of experts technical mastery evaluation (x-axis) and Dr. Scratch assessment (y-axis). Darker points represent a higher number of cases (i.e. more evaluations).

very strong correlation $r = .942$, which could be of interest for future research. In any case, according to the assessment research literature [6], Dr. Scratch is *ideally convergent* with expert evaluators, as the correlation found is greater than $r = .70$ when considering the experts' technical mastery scores and the ones provided by Dr. Scratch.

Several fundamental aspects of programming, such as debugging, design or remixing skills, are not assessed by Dr. Scratch. Other crucial aspects of CT skills, such as originality, creativity or correctness, are not taken into account either. Furthermore, the fact that a programming construct appears in a project does not necessarily mean that the author understands it [5], as it could have been copied from another project, for instance. In consequence, Dr. Scratch should not be understood as a replacement of evaluators or mentors, but as a supporting tool that assists them in some of the assessment tasks. Automatic assessment of learning outcomes in programming is an emerging area that needs further attention by educators and researchers.

Several cases were found in which there is a notorious difference between assessments. These discrepancies were discussed in a non-structured way with some members of the jury. The main reason for the detected differences was because of the functionality of the projects. While experts took into account if projects achieve their goals as stated in the instructions or the usability of the project, this is not the case for Dr. Scratch, unable to evaluate such issues. In addition, at the time of running the contest, Dr. Scratch measured all scripts in a project, even the ones that were never executed. Experts, on the contrary, stated that those scripts should not be considered, as they would not have influence on the functionality. This has been modified in newer versions of Dr. Scratch [11].

Being early research, this investigation is based on a low number of projects, limited to 53, which can be seen as a threat for the generalizability of the results (external validity), and has not followed formal interviews with judges (internal validity). Further research should address these issues to offer more solid scientific evidence.

Thus, we plan to replicate the investigation with a higher number of participating students. We also intend to count with more judges. We would like to study potential differences in the correlations between measurements in terms of types of projects (highly *vs.* lowly ranked by experts projects), years of experience of the expert or his/her background field of expertise, etc. In addition to the scores provided by experts, we would like to carry out semi-structured interviews with them to discuss the discrepancies between their assessment and the one by Dr. Scratch, in the hope that these interviews may provide us reliable, comparable qualitative data.

All in all, in this paper we have presented evidence that the automatic assessment of CT skills is a promising area. We think that further research could result in tools that assist learners and educators, in the same way professional software developers benefit from modern development-supporting tools.

Acknowledgments

We would like to thank the participants in the competition, the expert judges and, as sponsors of the contest, FECyT and Google. This work has been funded in part by the Region of Madrid under project "eMadrid - Investigación y Desarrollo de tecnologías para el e-learning en la Comunidad de Madrid" (S2013/ICE-2715). The last author also wants to acknowledge EU Funded SENECA project, a Marie Skłodowska-Curie action.

References

[1] Efthimia Aivaloglou and Felienne Hermans. 2016. How Kids Code and How We Know: An Exploratory Study on the Scratch Repository. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. ACM, New York, NY, USA, 53–61. DOI: http://dx.doi.org/10.1145/2960310.2960325

[2] Anja Balanskat and Katja Engelhardt. 2015. *Computing Our Future: Computer Programming and Coding-Priorities, School Curricula and Initiatives Across Europe*. Technical Report. European Schoolnet.

[3] Valerie Barr and Chris Stephenson. 2011. Bringing computational thinking to K-12: What is Involved and what is the role of the Computer Science education community? *ACM Inroads* 2, 1 (2011), 48–54.

[4] Bryce Boe, Charlotte Hill, Michelle Len, Greg Dreschler, Phillip Conrad, and Diana Franklin. 2013. Hairball: Lint-inspired Static Analysis of Scratch Projects. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 215–220.

[5] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of Computational Thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*. 1–25.

[6] Kevin D Carlson and Andrew O Herdman. 2012. Understanding the impact of convergent validity on research results. *Organizational Research Methods* 15, 1 (2012), 17–32.

[7] Stephen Cooper, Wanda Dann, and Randy Pausch. 2000. Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges* 15, 5 (2000), 107–116.

[8] Diana Franklin, Phillip Conrad, Bryce Boe, Katy Nilsen, Charlotte Hill, Michelle Len, Greg Dreschler, Gerardo Aldana, Paulo Almeida-Tanaka, Brynn Kiefer, Chelsea Laird, Felicia Lopez, Christine Pham, Jessica Suarez, and Robert Waite. 2013. Assessment of Computer Science Learning in a Scratch-based Outreach Program. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 371–376.

[9] Google. 2015. *Searching for Computer Science: Access and Barriers in U.S. K-12 Education*. Technical Report. Gallup. https://services.google.com/fh/files/misc/searching-for-computer-science_report.pdf

[10] Felienne Hermans and Efthimia Aivaloglou. 2016. Do code smells hamper novice programming? A controlled experiment on Scratch programs. In *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*. IEEE, 1–10. DOI: http://dx.doi.org/10.1109/ICPC.2016.7503706

[11] Amy K. Hoover, Jackie Barnes, Borna Fatehi, Jesús Moreno-León, Gillian Puttick, Eli Tucker-Raymond, and Casper Harteveld. 2016. Assessing Computational Thinking in Students' Game Designs. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts (CHI PLAY Companion '16)*. ACM, New York, NY, USA, 173–179.

[12] Stephen C Johnson. 1977. *Lint, a C program checker*. Technical Report Computer Science 65. Bell Laboratories.

[13] Kyu Han Koh, Ashok Basawapatna, Vicki Bennett, and Alexander Repenning. 2010. Towards the Automatic Recognition of Computational Thinking for Adaptive Visual Language Learning. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*. 59–66. DOI: http://dx.doi.org/10.1109/VLHCC.2010.17

[14] Matt MacLaurin. 2009. Kodu: End-user Programming and Design for Games. In *Proceedings of the 4th International Conference on Foundations of Digital Games (FDG '09)*. ACM, New York, NY, USA, Article 2. DOI: http://dx.doi.org/10.1145/1536513.1536516

[15] Jesús Moreno and Gregorio Robles. 2014. Automatic detection of bad programming habits in Scratch: A preliminary study. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. 1–4. DOI:http://dx.doi.org/10.1109/FIE.2014.7044055

[16] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2015. Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED. Revista de Educación a Distancia* 15, 46 (2015), 23.

[17] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2016. Comparing computational thinking development assessment scores with software complexity metrics. In *2016 IEEE Global Engineering Education Conference (EDUCON)*. 1040–1045. DOI:http://dx.doi.org/10.1109/EDUCON.2016.7474681

[18] Ricardo Olmos, Guillermo Jorge-Botana, José M. Luzón, Jesús I. Martín-Cordero, and José Antonio León. 2016. Transforming {LSA} space dimensions into a rubric for an automatic assessment and feedback system. *Information Processing & Management* 52, 3 (2016), 359–373.

[19] Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

[20] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for All. *Commun. ACM* 52, 11 (Nov. 2009), 60–67.

[21] Linda Seiter and Brendan Foreman. 2013. Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research (ICER '13)*. ACM, New York, NY, USA, 59–66.

[22] Amanda Wilson, Thomas Hainey, and Thomas Connolly. 2012. Evaluation of computer games developed by primary school children to gauge understanding of programming concepts. In *European Conference on Games Based Learning*. Academic Conferences International Limited, 549.

[23] Cindy Wilson and Leon J Osterweil. 1985. Omega–A Data Flow Analysis Tool for the C Programming Language. *IEEE Transactions on Software Engineering* 11, 9 (1985), 832.

[24] Jeannette M Wing. 2006. Computational Thinking. *Commun. ACM* 49, 3 (2006), 33–35.

[25] Ursula Wolz, Christopher Hallberg, and Brett Taylor. 2011. Scrape: A tool for visualizing the code of Scratch programs. In *Poster presented at the 42nd ACM Technical Symposium on Computer Science Education*. Dallas, TX.