Early Validation of Computational Thinking Pattern Analysis

Kyu Han Koh
Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80309
+1 303 492 1349
kohkh@colorado.edu

Hilarie Nickerson
Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80309
+1 303 492 1349
hnickerson@colorado.edu

Ashok Basawapatna
Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80309
+1 303 492 1349
Ashok.basawapatna@colorado.edu

Alexander Repenning
Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80309
+1 303 492 1349
ralex@cs.colorado.edu

ABSTRACT

End-user game design affords teachers a unique opportunity to integrate computational thinking concepts into their classrooms. However, it is not always apparent in game and simulation projects what computational thinking-related skills students have acquired. Computational Thinking Pattern Analysis (CTPA) enables teachers to visualize which of nine specific skills students have mastered in game design that can then be used to create simulations. CTPA has the potential to automatically recognize and calculate student computational thinking skills, as well as to map students' computational thinking skill progression, as they proceed through the curriculum. The current research furthers knowledge of CTPA by exploring its validity based on how its performance correlates to human grading of student games. Initial data from this validation study indicates that CTPA correlates well with human grading and that it can even be used to predict students' future achievement levels given their current skill progression, making CTPA a potentially invaluable computational thinking evaluation tool for teachers.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computers and Information Science Education

General Terms

Measurement, Performance, Design, Experimentation

Keywords

Computational Thinking, Computational Thinking Assessment, Computational Thinking Pattern Analysis, Cyberlearning Infrastructure, End User Programming

1. INTRODUCTION

Since the 1990s there have been multiple efforts to fix the broken pipeline at the K-12 level in computer science education [1, 2, 3], and most of those efforts have focused on student motivation. The results indicate that these efforts have successfully increased student motivation in computer science. However, it is often not clear what educational benefits, if any, students receive from these motivational interventions. Part of the problem may stem from a lack of a proper instrument for measuring the knowledge students acquire through their activities in a class. Learning may be measured with existing tools such as grading rubrics, but these tools are extremely time consuming and have limited functionality with respect to assessing learning progress and providing ongoing feedback to students and teachers.

In this paper we present our early efforts to validate a method that we have developed for measuring concept learning in real time. Our technique is inspired by Latent Sematic Analysis (LSA) [4], in which calculations can be used to analyze the semantic meanings of a given context based on predefined subjects or phenomena. Theoretically, this idea can be applied to any domain in which low-level components are combined to form higher-level constructs, supporting analyses such as natural language processing and the examination of computer programs. Constrained environments, including those that make use of visual end-user programming, are especially suited to this kind of analysis. Therefore, this idea can be employed to build a learning assessment tool for computer science (CS) and/or computational thinking (CT) [5] education, where visual programming environments such as AgentSheets [6], Scratch [7], and Alice [8] are widely adopted.

The computer programming context used in our research is the construction of video games and science simulations. At the University of Colorado – Boulder we have four years of data from more than 20,000 students who created games and simulations using AgentSheets and its subsequent 3D version, AgentCubes [9]. Our analytical method, Computational Thinking Pattern Analysis (CTPA), examines the programmed rules of these student-created artifacts to unearth evidence of higher-level patterns that are found in such projects, which typically involve object interactions [10]. Computational Thinking Patterns (CTPs) are constructs students initially learn in game design that can be

applied to creating simulations. Examples of CTPs include one agent tracking another agent, one agent absorbing another agent, and one agent creating another agent [11]. The presence of these semantically meaningful patterns indicates that students have grasped the concepts being taught and that they also have the ability to operationalize the concepts by creating programs. Therefore, CTPA can serve as a concept and skill learning assessment tool for CS/CT education.

The outcomes of CTPA can be used to provide valid and useful feedback to educators and learners in CS/CT education by measuring and tracking student learning outcomes. Student-created artifacts from educational programs around the United States and in several other countries have been analyzed by CTPA. These results have shown promising potential in providing educational feedback in the areas of learning transfer [12], learning trajectories [13], and programming divergence [14]. This kind of assessment in CS/CT education should be able to provide better individual feedback and faster learning assessments to students and teachers by measuring student skills and analyzing learning at the semantic level.

Overall, this research offers a partially-validated method to assess student learning skills and compute student learning outcomes. This type of method can be used to create authentic cyberlearning systems that will help large numbers of teachers and students to learn computational thinking. We envision that other investigators will be able to repurpose our calculation method for use in their own educational research contexts.

2. METHODOLOGY
Methodology for this research incorporates rubric-based grading of student projects from multiple classes, the Computational Thinking Pattern Analysis of these submissions, and skill progression calculation to measure learning performance.

2.1 Class Descriptions
The University of Colorado – Boulder Educational Game Design class is a course teaching undergraduate and graduate students to prototype and test educational games. The course has a fairly aggressive programming schedule in which each student builds a playable game every week. The course has three parts. In the first part students learn about computational thinking patterns [10] in the context of making four prescribed games ranging from simple 1980 arcade games such as Frogger to more contemporary games such as the Sims. The design and creation of a Sims-like game exposes students to computational thinking [5] concepts related to artificial intelligence such as collaborative diffusion [15] as well as psychological models such as Maslow's hierarchy of needs. In the second part students design four of their own games, but these games have to be educational. Evaluation includes peer assessment of engagement and the educational value of games. In the last part students work on their final educational game, receiving three weeks time to include more evaluation iterations. Students conduct game testing at a local middle school where they collect empirical evidence from game testers who have not been exposed to their designs.

The validation study presented here focuses on the prescribed games from the first part of the course. The validation process correlates scores produced by a human grader using a rubric with scores produced by Computational Thinking Pattern Analysis. We have correlated the human and machine scoring for individual games as well as for game clusters. This paper presents the correlation between CTPA-measured skills and human-graded scores from the 2012 and 2013 classes graded by two different individuals. Additionally, the paper presents the inter-grader agreement for the 2012 class graded by two different human graders.

2.2 Computational Thinking Pattern Analysis
Computational Thinking Pattern Analysis (CTPA) is designed to evaluate the semantic meaning of the students' submitted games and simulations. The Latent Semantic Analysis [4] approach as applied in CTPA detects which computational thinking patterns (CTPs) are implemented in a given submission. CTPA looks for nine pre-defined canonical computational thinking patterns within a given game/simulation: user control, generation, absorption, collision, transportation, push, pull, diffusion, and hill climbing. These particular nine patterns are commonly found in video games and science simulations. In the future we could include more CT patterns in CTPA, but to date our research has focused on these nine.

2.2.1 Computational Thinking Patterns (CTP)
Computational thinking is a high-level concept that experts have still not been able to clearly define in a single sentence. We therefore conceptualized Computational Thinking Patterns within the game design context to help students and teachers understand how computational thinking can be practically utilized [10]. A Computational Thinking Pattern (CTP) is an abstract representation that can be easily found in game and simulation programming. For example, the Absorption CTP represents one agent removing another agent (e.g., big fish eats small fish in an ecosystem). In this way, each CTP represents one complete phenomenon or behavioral concept in a game or science simulation design.

Because Computational Thinking Patterns are high-level programming concepts, each pattern requires multiple rules and/or programming primitives to be implemented. For example, the following figure illustrates how the Absorption pattern is programmed using a single rule that contains one condition and one action.
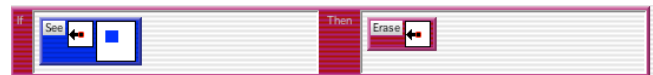

Figure 1. The Absorption pattern implementation

To perform CTPA, a given AgentSheets project is converted such that the degree to which each CTP is present is expressed as a vector component. An AgentSheets project vector is calculated with the equation below, similar to that used in Latent Semantic Analysis [4] to describe semantic meaning [12].

$$CTPA\,(m) = \left[\frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \sqrt{\sum_{i=1}^{n} v_i^2}} \right]_{1}^{m}$$

Equation 1. Computational Thinking Pattern Analysis

In this equation, m is the number of computational thinking patterns that are sought (currently nine). The calculated result of CTPA—CTPA (1) to CTPA (m)—can thus be represented as an m length vector. Also, n is the total number of different primitives (conditions and actions) that could possibly appear in any

game/simulation (currently 39). Vectors u and v, both of length n, respectively describe the primitives that actually appear in a given game/simulation and in a canonical project that implements one of the m CT patterns.

2.2.2 Computational Thinking Pattern Analysis Graph

The Computational Thinking Pattern Analysis (CTPA) graph visualizes the semantic meaning and computational thinking patterns of the submitted games represented by the nine-dimensional vector calculated through CTPA. The computational thinking patterns implemented in each given game are depicted through this graphic (Figure 2).

This research implementation uses regular class curriculum and is assessed using official game tutorials provided by the Scalable Game Design project researchers and educators. In the case of Figure 2, the student CTPA (Brown in Figure 2) is overlaid with a graph of the tutorial CTPA (Green in Figure 2). Each CT pattern axis is aligned according its implementation difficulty level and its significance to the relationship between adjacent axes. For example, Generation, Absorption, and Collision usually happen in sequence or are highly relevant to each other.

This graphic analysis can work as a self-assessment tool and/or a learning path indicator through a semantic comparison of the submitted project to that specific submission's tutorial standard. In the absence of a comparative tutorial, standardized information can be programmed into the graphic analysis tool to serve as an appropriate comparison.
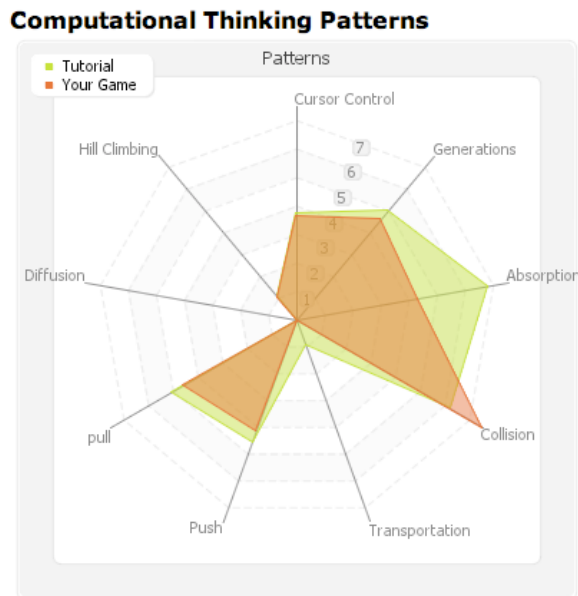

Figure 2. Computational Thinking Pattern Analysis Graph

2.3 Computational Thinking Skill Progression

While the semantic information from individual games/simulations provides useful insight into student learning development, the semantic analysis of individually created games or simulations could also provide an indication of a student's overall skill progress. Representing semantic meaning in measureable units to visually demonstrate student learning trends can benefit students and teachers directly. This approach could also indicate possible curriculum failings at a fundamental level.

The value of each axis on the CTPA Graph translates as the amount of implemented knowledge for a given computational thinking pattern within a game/simulation. The sum or average of these values is interpreted as the student's skill in designing the game/simulation. That is, the nine computational thinking patterns are target-learning categories. The score of each CT pattern in a tutorial represents that pattern's target-learning goal. Thus, the CTPA Graph illustrates how well students meet the target-learning goal in each assignment or group of assignments. Within the CTPA, a one-time assignment analysis is referred to as a Demonstrated Skill Score. Learning that takes place over time through several assignments is referred to as a Comprehensive Skill Score. Both Demonstrated and Comprehensive Skill Scores are calculated from the length (norm) of a vector of the nine computational thinking patterns reduced to one dimension (unit).

The Demonstrated and Comprehensive Skill Scores are calculated using the following equations.

$$\text{Demonstrated Skill Score } (n) = \frac{\sqrt{\sum_{i=1}^{n}(P_i)^2}}{\sqrt{n}}$$

Equation 2. Demonstrated Skill Score

$$\text{Comprehensive Skill Score } (m) = \frac{\sqrt{\sum_{i=1}^{n}[\max_{j=1}^{m}(P_{i,j})]^2}}{\sqrt{n}}$$

Equation 3. Comprehensive Skill Score

In these equations, P is a computational thinking pattern, n is the number of computational thinking patterns on the CTPA Graph, and m is the number of submitted assignments. The equations are derived from the formula for the length of a vector.

The Demonstrated Skill Score shows a student's programming skill as of when the game was submitted, while the Comprehensive Skill Score shows a student's progressed skill acquisition over time. Each Skill Score is the normalized size of the value on each axis of the CTPA Graph. For the Comprehensive Skill Score calculation, we make the following assumption to track students' skill progression: if there is a skill that a student has learned and demonstrated accurately at least once, then that skill is available for the student to use for the entire duration of the course even if it is not used again. In other words, a maximum value of any given game represents its creator's best achieved level in CT pattern implementation. Consequently the maximum value is selected in this equation.
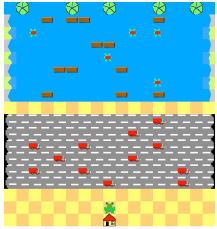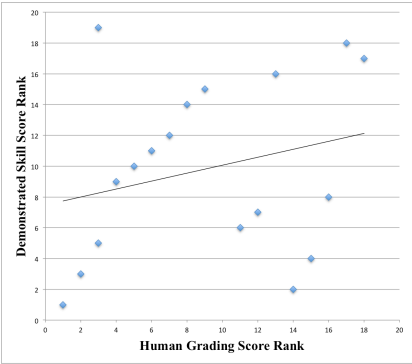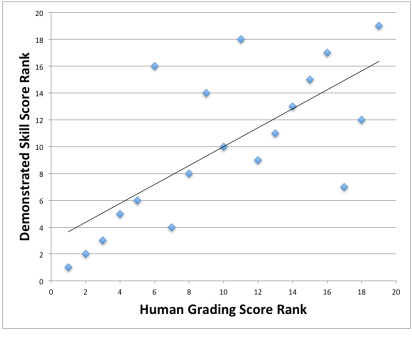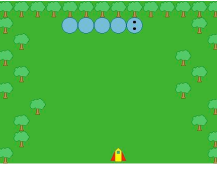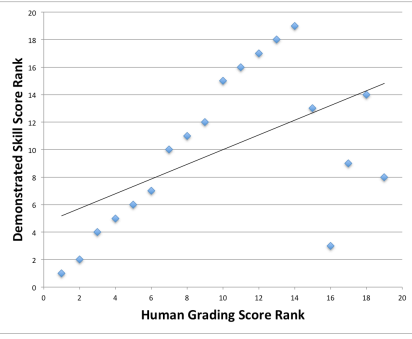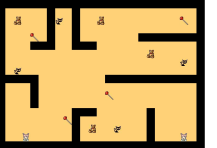
3. RESULTS: ASSESSMENT VALIDATION

To gauge the value of CTPA as a Computational Thinking assessment tool, we conducted the early stages of concurrent validity and predictive validity evaluation using data from undergraduate and graduate students who took an Educational Game Design class in 2012 and 2013. For concurrent validity we compared student grades with CTPA-measured skills for four basic games: Frogger, Sokoban, Centipede, and the Sims. To assess predictive validity we computed students' comprehensive skill scores based on the four basic games and compared them to the demonstrated skill scores of their final projects.

3.1 2012 Class Concurrent Validity Results

For the 2012 class (19 students), we hired two graders for this research who were asked to provide grades based on the official grading rubric for each game. We also used CTPA to calculate a demonstrated skill score for each game.

Table 1. Four Basic Games and Spearman Rank Correlation Charts for the 2012 Class

Game	Spearman's Rank Correlation Coefficients
Frogger	0.246 (Spearman Correlation Coefficient)
Sokoban STEPS 0	0.705 (Spearman Correlation Coefficient)
Centipede	0.535 (Spearman Correlation Coefficient)
Sims	0.821 (Spearman Correlation Coefficient)
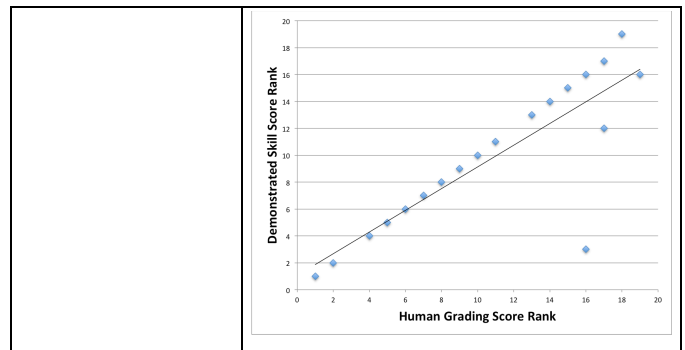
The human grades and the demonstrated skill scores are not normally distributed. Instead, they are skewed negatively. Therefore, we calculated Spearman's rank correlation coefficient to measure the statistical dependence between the CTPA-measured skills of students and the grades that they actually received.

3.1.1 Demonstrated Skill for Individual Games

As Table 1 shows, the Spearman rank correlation coefficients for three of the four basic games are high enough to demonstrate a correlation between human graded scores and CTPA-measured skills. These results indicate that CTPA is capable of measuring students' skills, and its measured results connect well with the human grades.

Although the originality and the design of the game were part of human grading, CTPA measures only programming skills. So for the tied scores, the person who received a higher grade in programming is ranked higher than the person who got a higher grade in originality and design. For example, there are two students who received 100 points where student A received 90 points for basic programming and 10 points for advanced design and student B received 80 points for basic programming and 20 points for advanced design. In this case, student A is ranked higher than student B. If students received exactly same scores for basic and advanced programming, then they are ranked based on their programming completeness (i.e., avoiding undeclared variables/methods or unnecessary programming components).

3.1.2 Comprehensive Skill Across Several Games

We also calculated students' comprehensive skill scores to reflect the correlation between the average student grades and CTPA-measured skill scores when students finished making all four basic games.

The Spearman rank correlation coefficient value between students' grades and their CTPA-measured skill scores is 0.415 (Figure 3). This number indicates a moderate level of positive correlation between students' grades and their CTPA-measured skill scores. Due to the small sample size, we verified its significance with critical values for the Spearman rank correlation coefficient. The critical value for N=19 with a significance level of 0.05 is 0.391, which is lower than the calculated correlation coefficient, 0.415. This calculation indicates that there is a 95% chance of the correlation being truly significant. This result offers another positive indication of the CTPA's validity as a programming assessment tool, suggesting that it would be usable in a real classroom situation.
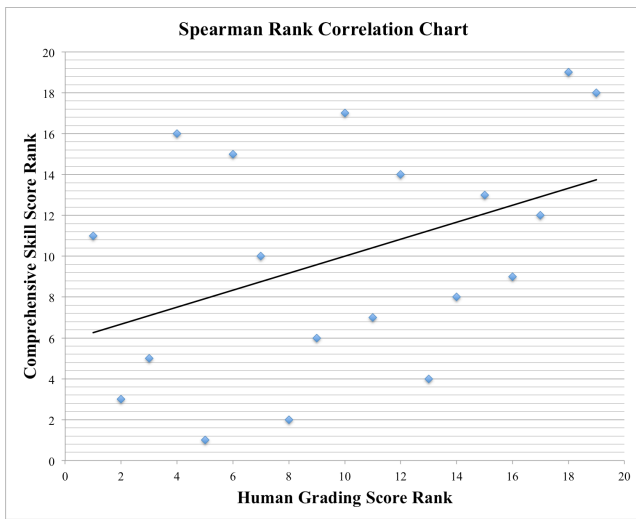
Figure 3. Spearman Rank Correlation Chart from 2012 Class

3.1.3 Inter-Rater Agreement

To check the inter-rater agreement between the two graders, we converted the original 1 to 100 scale scores to a letter grades from A to F. In a 1 to 100 scale score, there are 100 options for grades, and it was difficult to get high inter-rater agreement percentages since there were so many close scores but not exactly the same score (i.e. 93 vs. 95). We converted the scores above 90 to A, the scores above 80 to B, the scores above 70 to C, the scores above 60 to D, and the scores below 60 to F.

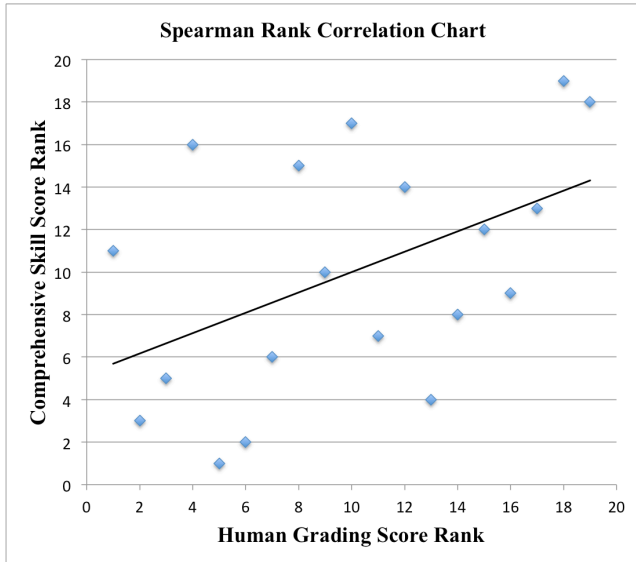The inter-rater agreement percentage between the two graders was 95% on average for the four basic game grades.

Figure 4. Spearman Rank Correlation Chart from 2013 Class

3.2 2013 Class Concurrent Validity Results

For the 2013 class (20 students), we hired one of the two graders who graded the 2012 class. The same rubric was provided for grading consistency. As for the 2012 class, the students' comprehensive skill scores were calculated as the basis for determining the correlation between average student grades and CTPA-measured skill scores when students finished making four basic games.

The Spearman's rank correlation coefficient value between students' grades and their CTPA-measured skills is 0.476 (Figure 4). This number indicates a moderate level of positive correlation between students' grades and their CTPA-measured skill scores. Due to the small sample size, we again verified its significance with critical values for the Spearman rank correlation coefficient. The critical value for N=20 with a significance level of 0.025 is 0.447, which is lower than the calculated correlation coefficient, 0.476. This calculation indicates that there is a 97.5% chance of the correlation being truly significant. This result illustrates CTPA-measured skill's reliability over two consecutive classes.

3.3 Predictive Validity Results

We then performed a predictive validity test to confirm CTPA's validity as a programming assessment tool. In contrast to the four basic games, the final project was graded based on originality, educational facts, engagement, and student presentation skills rather than programming skills. Thus, for predictive validity, it was not adequate to compare CTPA-measured student skills and student grades.

However, it is possible to use a pure programming comparison to predict students' future achievements based on their previous skills. In other words, if a student has shown high achievement through previous assignments, then s/he is expected to show high achievement in the final project, too. We therefore computed student CTPA-measured skills to show their correlation between pre-final projects and the final project. As Figure 5 illustrates, those who showed better performance through pre-final assignments tended to show better performance in the final project also. For the 2012 class, the Pearson correlation coefficient value between pre-final projects and the final project is 0.676, and there is a 99.5% chance of this correlation being truly significant. For a better correlation calculation, we excluded two students who missed more than three assignments and one student who didn't submit his final project.
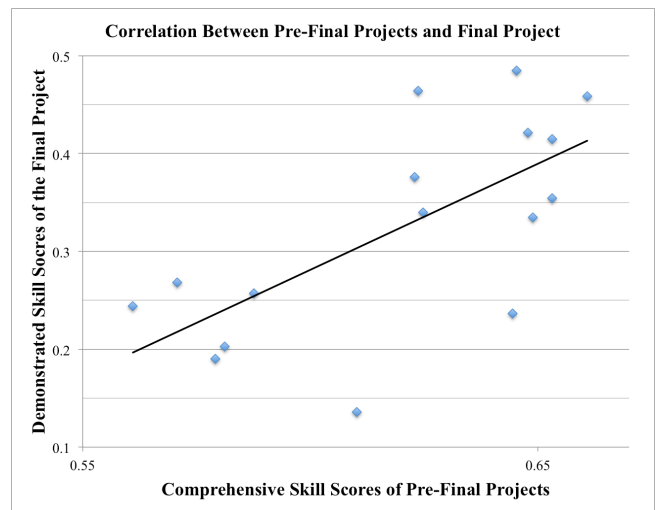

Figure 5. Predictive Validity Evaluation from 2012 Class

This high correlation between Skill scores from pre-final projects and the final project implies that CTPA is able to predict a student's future learning performance and skill trajectory. This capability of CTPA can be applied to build a cyberlearning infrastructure, including automated tutoring systems. For example, right now we are working on a system called REACT (Real time Evaluation and Assessment of Computational

Thinking) that provides the teacher with a dashboard to see what students are programming in real time using CTPA graphs and other visualizations. REACT provides teachers with a useful representation of class and individual progress, allowing them to make effective instructional decisions. The REACT system's feedback is based on CTPA-measured skill.

4. DISCUSSION

In this initial foray into CTPA validation, we found satisfactorily strong positive correlations between scores given by human graders and students' comprehensive skill scores calculated by CTPA, giving us confidence about proceeding with further validation activities. Several factors suggest that the correlations described here are lower than those we might expect to find during additional validation, including the small size of the samples. The current human grader scoring rubric includes both programming skill items, which are closely related to the characteristics examined through CTPA, and other, less related items. For example, the graders checked for the presence of expected computational thinking pattern implementation, and also looked for what users should experience while the game is played. Therefore, human graders are evaluating game design skill along with programming skill. A revised rubric with greater emphasis on programming would be expected to lead to higher correlations. Additionally, these samples include a large percentage of high-performing students, and we believe that we would see more accurate correlations using students having a greater range of skill levels. Overall, the early validation results for the CTPA are promising, though further exploration with a larger data set is warranted. Beyond demonstrating that CTPA and human grader performance are well correlated when assessing foundational games, we showed the predictive value of this analysis tool for assessing students' skill in designing their own games. We anticipate that it will be possible to use CTPA in the future to provide trustworthy educational feedback, especially given the consistency of the findings using data from two consecutive years.

5. ACKNOWLEDGMENTS

6. REFERENCES

[1] Werner, L., Campe, S., Denner. J., Children learning computer science concepts via Alice game-programming. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM, New York, NY, USA

[2] Koh, K. H., Repenning, A., Nickerson, H., Endo, Y., Motter, P., Will it stick? exploring the sustainability of computational thinking education through game design. In *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13). ACM, New York, NY, USA, 597-602.

[3] Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., Rusk, M., Programming by choice: urban youth learning programming with scratch. *SIGCSE Bull.* 40, 1 (March 2008), 367-371.

[4] Landauer, T. K., Foltz, P. W., Laham, D. Introduction to Latent Semantic Analysis. Discourse Processes, 25, 1998, 259-284

[5] Wing, J. M. 2006. Computational Thinking. Communications of the ACM, 49(3), pp. 33-35, March 2006.

[6] Repenning, A. 2000. AgentSheets®: an Interactive Simulation Environment with End-User Programmable Agents. *In Proceedings of Interaction 2000*, Tokyo, Japan, 2000.

[7] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A, Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. 2009. Scratch: programming for all. Commun. ACM 52, 11 (November 2009), 60-67.

[8] Werner, L., Denner, J., Bliesner, M., and Rex, P. 2009. Can middle-schoolers use Storytelling Alice to make games? results of a pilot study. *In Proceedings of the 4th International Conference on Foundations of Digital Games (FDG '09)*. ACM, New York, NY, USA, 207-214.

[9] Ioannidou, A., Repenning, A., Webb. D., AgentCubes: Incremental 3D end-user development. *J. Vis. Lang. Comput.* 20, 4 (August 2009), 236-251

[10] Ioannidou, A., Bennett, V., Repenning, A., Koh, K., Basawapatna, A. 2011. Computational Thinking Patterns. *In Proceedings of 2011 Annual Meeting of the American Educational Research Association (AERA) in the symposium "Merging Human Creativity and the Power of Technology: Computational Thinking in the K-12 Classroom".* New Orleans, April 8-12, 2011

[11] Basawapatna, A., Koh, K. H., Repenning, A., Using Scalable Game Design To Teach Computer Science From Middle School to Graduate School, *ITiCSE '10: Annual Conference on Innovation and Technology in Computer Science Education*, Ankara, Turkey June 26-30, 2010.

[12] Koh, K. H., Basawapatna, A.,Bennett, V., Repenning, A. 2010. Towards the Automatic Recognition of Computational Thinking. *In Proceedings of IEEE International Symposium on Visual Languages and Human-Centric Computing 2010*, Leganés-Madrid, Spain, September 21-25, 2010

[13] Bennett, V., Koh, K. H., Repenning, A. Computing learning acquisition?, *IEEE International Symposium on Visual Languages and Human-Centric Computing* 2011, Pittsburgh, PA, USA, September 18-22, 2011

[14] Bennett, V., Koh, K. H., Repenning, A., Computing Creativity: Divergence in Computational Thinking, *ACM Special Interest Group on Computer Science Education Conference, (SIGCSE 2013)*, March 6-9, 2013, Denver, Colorado, USA

[15] Repenning, A., Excuse me, I need better AI! Employing Collaborative Diffusion to make Game AI Child's Play. *in Proceedings of the ACM SIGGRAPH Video Game Symposium*, (Boston, MA, 2006), ACM Press.