SAGE Gameful Constructionism Assessments and Analytics - Proposal

COMS6901 – Projects in Computer Science, Fall 2017

Johan Sulaiman/js5063, with Tim Kartawijaya

Contents

Introduction

Since its inception in the 2nd half of the 20th century, video games have integrated into the daily lives of school-age children and captivated their mindshare with unprecedented and, some parents might say, devastating force. Even so, there are some among educators and researchers who take the view that every phenomenon that attracts voluntary investments of time, energy, and attention among children is worth evaluating for its redeeming qualities towards fostering learning.

One researcher for example, views game-playing in a more positive light, as a problem-solving activity approached with a playful attitude (Schell, 2008). Some students approach game-playing sessions as experimentations of different problem-solving ideas and strategies while remaining in the safety of the game's controlled and simulated environments. Students through simulated role-playing in games can also assume roles and professions they are not quite ready yet to take on in real-life, immersing and familiarizing themselves with the rich context and various facets of the job. In effect, this type of training through epistemic games (Rupp et al, 2010) can serve as excellent preparation for entering the professional workforce in the future. All these foster creativity, deep familiarity, and risk-taking that build on the students' knowledge and skills, while limiting potentially permanent and considerable loss of financial, relational, or even physical capital if similar experimentations were conducted in real-life.

Related Work

Computational Thinking

One specific area of learning that is especially strategic to prepare students for the jobs of the 21st century is Computational Thinking. Popularized by Dr. Jeannette Wing in 2006, Computational Thinking recognizes that the jobs of the future will increasingly rely on the aid of programmable machines to meet increasingly higher standard of acceptable productivity in all industries and workplaces. Workers who seek to maximize their output through machine-and-computer-based enablers are required to master multi-layered concepts and mental competencies such as abstraction, decomposition, recursion, iteration, modeling, caching, and many others (Wing, 2006).

Scratch and SAGE

The Social Addictive Gameful Engineering (SAGE) project aims to leverage games' addictiveness and their allure for voluntary temporal and energy investments to infuse this computational thinking within grade 6-8 curricula. In doing so, the construction-oriented Scratch programming language is an ideal medium (Bender, 2015). Scratch (Resnick et al, 2009) provides the unrestrictive open-ended game design space that students respond well to, allowing for self-paced explorations and personalization of virtual Lego-like game blocks that

when linked correctly become executable programs that come alive and shareable with Scratch's active multimillion user-base.

Consctructionism

Because free-ranging constructionism is an essential and undivided part of Scratch, it is important to realize the unique characteristics of constructionistic games. Supplying a sufficient degree of freedom that encourages exploration for example, is an important prerequisite for a gameful implementation of Scratch-based curricula (Weintrop et al, 2012). It is also important to recognize and anticipate constructionist games' resistance to grading, ranking, and classifying children as bad/underachievers, because inherent in the idea of constructionism is that all students can engage in deep learning if the environment, tools, and facilitation are well-designed (Berland et al, 2014). On the data gathering spectrum, constructionist games do provide a wealth of unstructured data conducive for application of data mining techniques, much more so compared to traditional games (Berland, Baker, Blikstein, 2014). All these taken together should influence current and further work for SAGE.

Evidence-Centered Design

Despite all these potential real-world gains from games that foster learning, proponents of game-based learning acknowledge more empirical evidence and curriculum-specific learning outcomes are needed to conclusively prove that games are an educationally effective solution (Kazimoglu et al, 2010). Devising an effective assessment strategy in measuring student's learning progress while using construction-oriented games thus is an area of importance. Frameworks such as evidence-centered design (ECD) are attractive (see Figure 1), because they aim to structure learners' performance data that are continuously collected from the game, utilizing machine-based reasoning techniques to make inferences about the learner's competences across a network of competences and skills resulting in a learner model (Rupp et al, 2010). It is argued that stealth assessment and ECD can assess not only content-specific learning but also general abilities, dispositions and beliefs, thus being more adequate to assess the so-called 21st century competences such as problem-solving skills, persistence, and creativity (Carvalho, 2017).

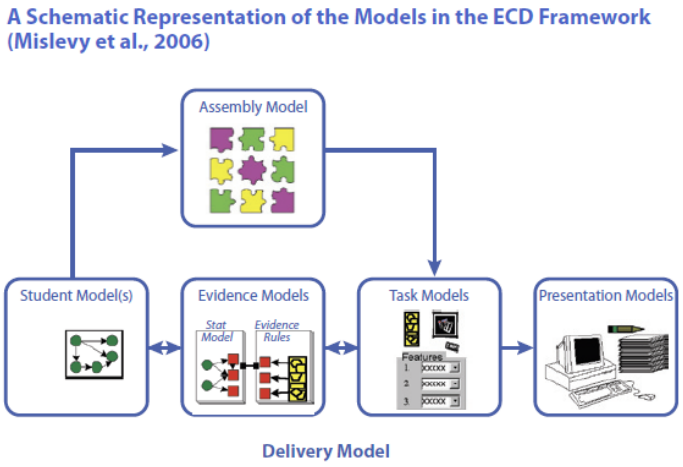


Figure 1. Example ECD Framework

ADAGE

ADAGE (Halverson, Owen, 2014) is another example of an open-source data collection and analysis framework that interprets real-time click-stream (telemetry) data from games and other digital learning environments into formative evidence of learning. These data can be used to identify patterns in play within and across players (using data mining and learning analytic techniques) as well as statistical methods for testing hypotheses that compare play to content models (Groff et al, 2015).

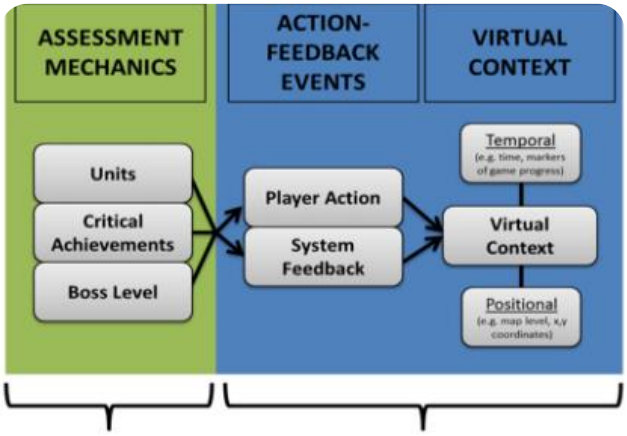


Figure 2. Adage Framework (Groff et al, 2015)

PECT

Works and researches related to the Progression of Early Computational Thinking (PECT) model (Seifer et al, 2013) are especially designed for understanding and assessing development of computational thinking in primary grades, and could be leveraged in any development of any Computational-Thinking-oriented data analytics framework. Past researches have also frequently cited Dr. Scratch (Moreno et al, 2015) and the Hairball plug-ins (Boe et al, 2013) to produce CT score from Scratch project analysis.

Khan Academy

The web-based learning management platform Khan Academy (www.khanacademy.org) provides some inspiration as it serves as an excellent example of a tight and unified integrations of different front-end elements and the associated back-end functionalities in production mode. Its system of points, badges, progress tracker, and accomplishments progressions implements a functioning Gamified Accomplishments tied with Persistent Progress that students can build upon session after session.

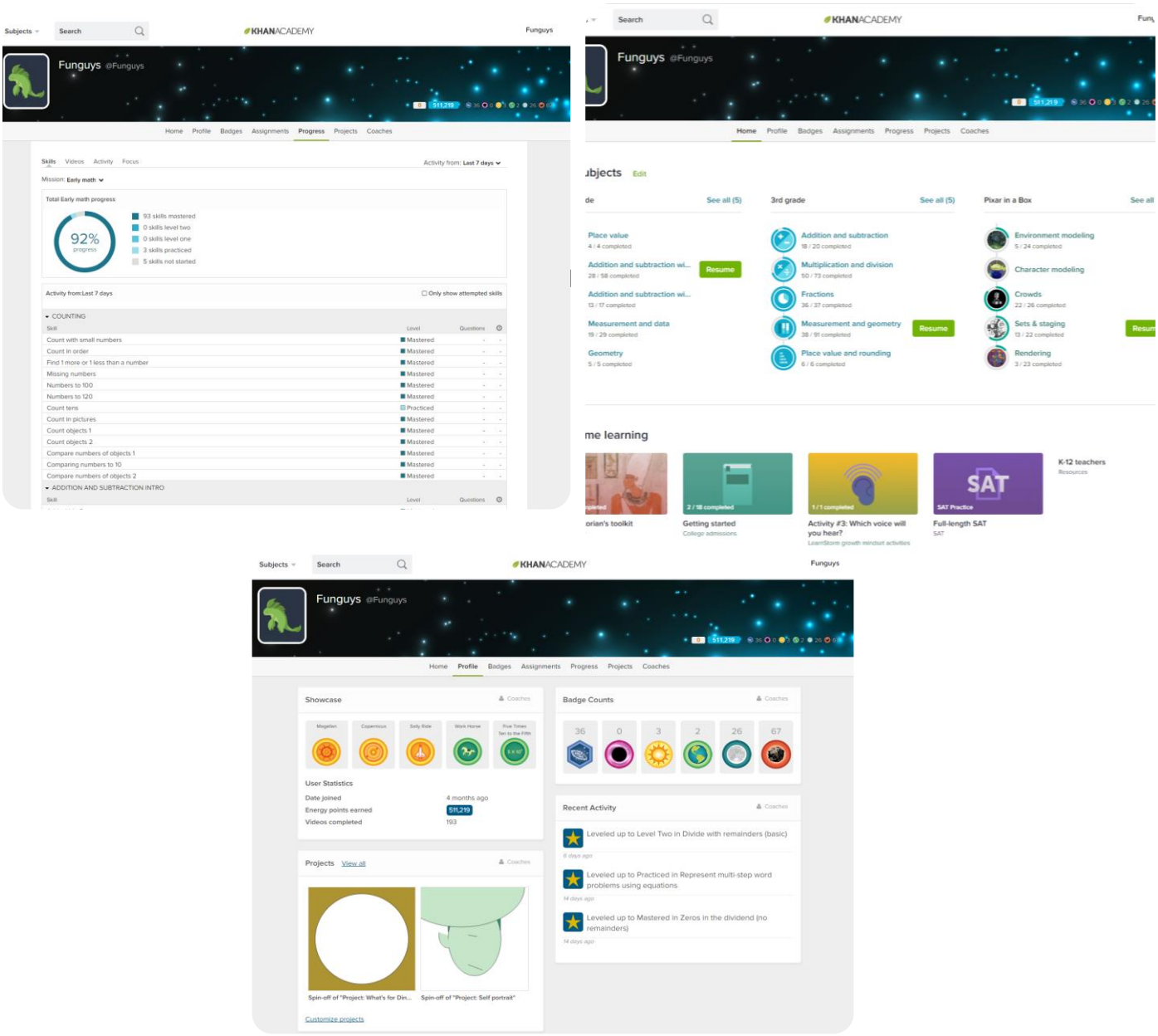


Figure 3. Khan Academy Gamified User Dashboard

Proposal

Looking Back at Three Years of SAGE

The past 3 years of SAGE development has created various modules and functionalities. We wish to continue the focus in maximizing Social Addictive Gamefulness during instillation of computational thinking, but with the added awareness that there is also a growing need to provide up-to-date, unifying structure on the conceptual and architectural level with a Gameful orientation. In an iterative manner, during the process of building a platform that supports

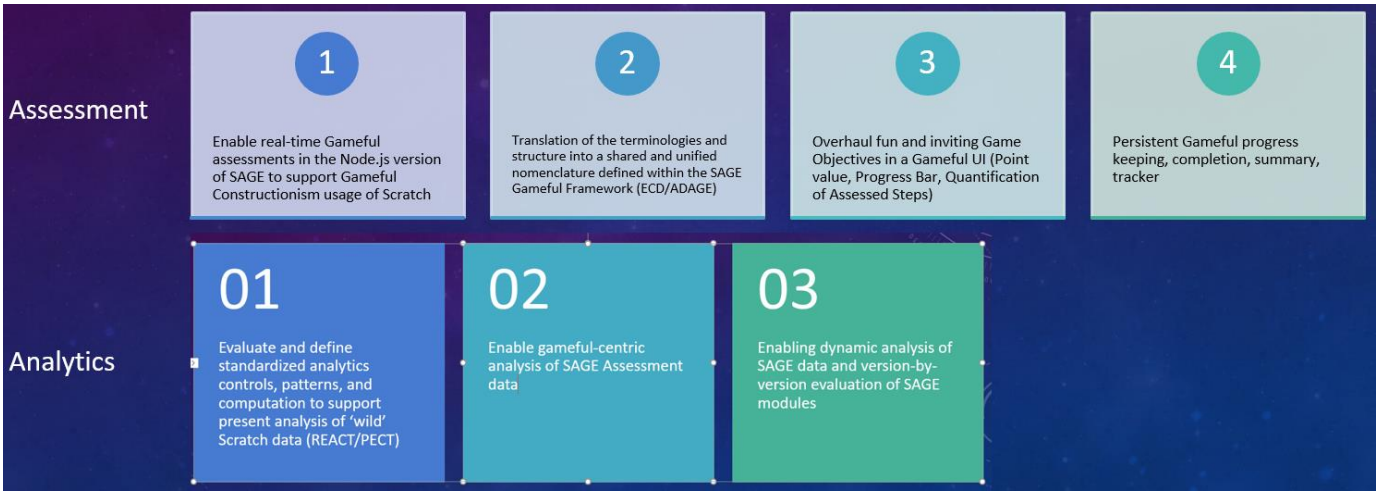
Socially Gameful Engineering activities, the platform itself requires frequent re-engineering. It was identified by past researchers that building a one-stop platform for SAGE that integrates all SAGE components under one web platform may lead to better engagement (Khandelwal & Mohanty, 2017). An analysis on the evolution of SAGE modules as each releases new functionalities shows that there is a high degree of inter-connectedness and inter-dependencies between modules (SAGE Wiki, 2017) that demands that the SAGE platform be developed with a holistic, integrated-product paradigm in mind. Part and parcel to this is the need for standardized terminologies and compatible object names in codes and schemas that show cohesiveness that are consistent across modules.

At the same time, there is also desire to evaluate how previously built modules perform when being used in experimentations, both in classroom settings (Anand & Sawyer, 2017) and within the Scratch community at large (Khandelwal & Mohanty, 2017). Conducting experiments and collecting usage and experiments data are of special importance for the SAGE Machine Learning module, which utilized algorithms that would improve with increased volume of usage data.

Maintaining a high level of integration between the different SAGE modules (SAGE Assessment Server, Scratch Analyzer, Scratch Affinity Space, Scratch Editor, Visual Assessment Editor, SAGE Machine Learning, MongoDB repositories, etc.) will allow for better cross-module compatibility, research clarity, and enablement of a more formal experimentation and evaluation of the effectiveness of SAGE modules. We also identified that the work to enhance SAGE gamified assessment feature would benefit greatly if it also is done side-by-side with the work to build a SAGE data analytics framework. This way, the designers of the data analytic framework keeps up-to-date with the latest changes and expansions done in the SAGE Assessment back-end and front-end, and similarly the SAGE Assessment functionalities develops in real-time and taking into consideration the types of analytics that are being built in the data analytics framework.

SAGE Constructionism Assessment and Data Analytics Framework – A Dual Track Development Approach

This dual-track approach of 1) incremental Gameful Assessment System enhancements that increases platform cohesiveness coupled with 2) the establishment of SAGE data analytics framework is further broken down below:



SAGE Gameful Assessment System

The first step towards enhancing SAGE's Gameful Assessment System involves the migration and refactoring of all assessment related features that exist in the various SAGE modules (especially the Go-based Assessment Server, the Visual Assessment Editor, the Node.js version of Assessment Server, and SAGE Affinity Space). Part of the work in this first step will be a joint-effort with the concurrent work proposed by the Gameful Affinity Space team. Around the same time, a review will be conducted to evaluate and when necessary refactor the naming system used for the various objects that exist in these modules and repositories, with the objective to achieve cohesiveness and a one-platform feel for SAGE, that is also compatible and comprehensible in the context of the various related research and findings from the academic literature that has influenced SAGE over the years.

After this is accomplished, what comes next is the work to build gameful features on top of this newly integrated platform. Of special interest is further utilization of the point systems throughout the play experience to relay to the students the various feedback from the game: persistent progress indicator, model knowledge growth through accumulation of points, milestone reaching notifications, and the associated rewards and celebratory messages. In addition to real-time communications at relevant points of the gameplay, these feedback occurrences are logged, summarized, and displayed in a pleasing, meaningful, and gameful manner in multiple dashboard and visualization pages.

SAGE Data Analytics Framework

Alongside the development of gameful assessment features, we also aim to design an appropriate analytics framework for SAGE efficacy analyses. The work is divided into four parts: 1) data extraction, 2) score computation using PECT rubric, 3) statistical analysis, and 4) integration into Scratch SAGE.

Firstly, we need the SAGE assessment data and the Scratch "wild" data in its proper format for appropriate data analysis. Note that R is the chosen environment for data exploration and cleaning. The Scratch "wild" data is in the format shown in Figure 4.

	project_id	scratchcomment	keyeventhatmorph	eventhatmorph_startclicked	eventhatmorph	mouseclickeventhatmorph	whenhatblockmorph	and_operator	multiply_operator
1	2437817	0	0	1	0	0	0	0	0
2	2437816	0	0	1	0	0	0	0	0
3	2437815	1	1	1	0	0	0	0	0
4	2437814	0	0	1	0	0	0	0	0
5	2437813	0	1	14	2	4	0	0	0
6	2437812	0	14	7	0	0	0	0	0
7	2437811	0	0	2	0	0	0	0	0
8	2437810	0	0	7	7	4	0	0	0
9	2437809	0	0	1	0	0	0	0	0
10	2437808	0	0	1	0	0	0	0	0
11	2437807	0	0	48	0	0	0	0	0
12	2437806	0	0	0	0	0	0	0	0
13	2437805	0	0	1	0	0	0	0	0
14	2437804	0	0	4	0	0	0	0	0
15	2437803	0	0	1	0	0	0	0	0
16	2437802	0	0	1	0	0	0	0	0
17	2437801	0	7	25	67	1	0	3	0
18	2437799	0	0	0	0	0	0	0	0
19	2437798	0	0	0	0	0	0	0	0
20	2437796	0	1	1	0	0	0	0	0
21	2437795	0	0	1	0	0	0	0	0
22	2437794	0	0	1	0	0	0	0	0
23	2437793	0	3	1	0	1	0	0	0
24	2437792	0	11	13	57	18	0	0	0
25	2437791	0	0	2	1	1	0	0	2
26	2437790	0	0	1	0	0	0	0	0
27	2437789	0	0	14	26	0	0	0	0
28	2437788	0	0	1	0	0	0	0	0
29	2437787	0	0	5	1	0	0	0	0

Figure 4: Snapshot of Scratch "wild" data

The dataset is in the format of a .csv file, with 1,925,606 rows and 172 columns. Each row represents the ID number of a project shared on the Scratch website at the time of data collection. Each column represents the frequency of each block type (add_operator, whengreenflagclicked, etc.) used in a Scratch project. The value of each cell indicates the number of times a particular block is used in a project. This dataset will be used as the 'baseline data' for assessing CT thinking in comparison with SAGE.

The SAGE data is extracted with an enhanced Scratch Analyzer from .sb2 files, which is associated with a past project (Anand and Sawyer, 2017). Thanks to the current version of Scratch analyzer, a .csv output file can be retrieved from a project, containing statistical data regarding the number of blocks containing in the project, the number of times a block's state is changed, and other data concerning the user behavior. The data output is shown in Figure 5.

FirstTimeSta	SecondTime	BlockName	BlockID	NoOfChanges	Change
20-57-51-GM	20-57-52-GM	whenKeyPre	9818C6E6-BE	1	TRUE
20-57-52-GM	20-57-53-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-57-53-GM	20-57-54-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-57-54-GM	20-57-55-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-57-55-GM	20-57-56-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-57-56-GM	20-57-57-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-57-57-GM	20-57-58-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-57-58-GM	20-57-59-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-57-59-GM	20-58-00-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-58-00-GM	20-58-01-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-58-00-GM	20-58-01-GM	doIfElse add	2A90EFE3-17	1	TRUE
20-58-01-GM	20-58-02-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-58-01-GM	20-58-02-GM	doIfElse	2A90EFE3-17	1	FALSE
20-58-02-GM	20-58-03-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-58-02-GM	20-58-03-GM	doIfElse	2A90EFE3-17	1	FALSE
20-58-03-GM	20-58-04-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-58-03-GM	20-58-04-GM	doIfElse	2A90EFE3-17	1	FALSE
20-58-03-GM	20-58-04-GM	doRepeat ad	85DC6A80-5	6	TRUE
20-58-04-GM	20-58-05-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-58-04-GM	20-58-05-GM	doIfElse	2A90EFE3-17	1	FALSE
20-58-04-GM	20-58-05-GM	doRepeat	85DC6A80-5	6	FALSE
20-58-05-GM	20-58-06-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-58-05-GM	20-58-06-GM	doIfElse	2A90EFE3-17	1	FALSE
20-58-05-GM	20-58-06-GM	doRepeat	85DC6A80-5	6	FALSE
20-58-06-GM	20-58-07-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-58-06-GM	20-58-07-GM	doIfElse	2A90EFE3-17	1	FALSE
20-58-06-GM	20-58-07-GM	doRepeat	85DC6A80-5	6	FALSE
20-58-07-GM	20-58-08-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-58-07-GM	20-58-08-GM	doIfElse	2A90EFE3-17	1	FALSE
20-58-07-GM	20-58-08-GM	doRepeat	85DC6A80-5	6	TRUE
20-58-07-GM	20-58-08-GM	doRepeat ad	85DC6A80-5	6	TRUE
20-58-08-GM	20-58-09-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-58-08-GM	20-58-09-GM	doIfElse	2A90EFE3-17	1	FALSE
20-58-08-GM	20-58-09-GM	doAsk	C7E1BA7D-B	3	FALSE
20-58-08-GM	20-58-09-GM	doRepeat	85DC6A80-5	6	FALSE
20-58-09-GM	20-58-10-GM	whenKeyPre	9818C6E6-BE	1	FALSE
20-58-09-GM	20-58-10-GM	doIfElse	2A90EFE3-17	1	FALSE

Figure 5: The above screenshot shows how the per-project output is formatted.

After the datasets are formatted and configured in a preferred environment, next comes the computation of scores that reflect the assessment of computational thinking (CT) from each Scratch project. Similar to Dr. Scratch's empirical study, we calculate these scores from a set of rules, which infers the user's competence in CT. In our framework, we propose to implement the Progression of Early Computational Thinking Model (PECT) to assess CT from usage data. We choose to use the PECT model since the model not only assess the complexity of blocks that users use (evidence variables) but it also assesses design patterns in a project (Design Pattern Variables). In the model, the evidence variables are scored according to which blocks are evident in a project (Figure 6) and the design pattern variables are scored based on the scores of the evidence variables (Figure 7). In the final stage, these two variables are then mapped to Computational Thinking Concepts, from which we can infer the user's development

in CT (Figure 8).

	1 -Basic	2 - Developing	3 -Proficient
Looks	Say, think.	Next Costume. Show. Hide.	Switch to costume. Set, change color/size/etc.
Sound	Play sound,note,etc.	Play vs Play until done.	
Motion	Move, goto sprite, point,turn.	Goto x,y. Glide to x,y.	Set,change X, Y.
Variables	Scratch variable (sprite, mouse pointer, answer, etc)	New variable (set,change)	New list.
Sequence & Looping	Sequence	Repeat, Forever.	Forever If. Repeat Until.
Boolean Expressions	Sensing operators.	<, =, >.	And, or, not.
Operators	Math	String and random	List
Conditional	If.	If... else....	Nested If/ If... Else...
Coordination	Wait.	Broadcast, When I Receive.	Wait Until.
User Interface Event	Green Flag clicked.	Key press, sprite clicked.	Ask and wait.
Parallelization	2 scripts start on same event.		
Initialize location	Set location properties (x,y,etc.) on green flag.		
Initialize looks	Set looks properties (costume, visibility,etc.) on green flag.		

Figure 6. CT Concepts and Associated Evidence during Scratch Usage

Animate Looks	Basic		Developing	Proficient	
	change appearance		initialize and change	state synchronized	event sychronized
Looks Initialization	0	0	1	1	1
Sequencing and Looping	1	2	1	3	1
Looks	2	2	3	3	3
Coordination	1	0	1	0	2
Parallelization	0	0	0	1	0

Animate Motion	Basic		Developing	Proficient
	change location		initialize and change	relative movement
Location Initialization	0	0	1	1
Sequencing and Looping	1	2	1	1
Motion	1	1	2	3
Coordination	1	0	1	0

Conversate	Basic	Developing	Proficient	
	monologue	time synchronized dialog	state synchronized	event synchronized
Sequencing and Looping	1	1	3	1
Looks or Sound	1(looks), 2(sound)	1,2	1,2	1,2
Coordination	0	1	0	2
Parallelization	0	1	1	0

Collide	Basic	Developing	Proficient
	discrete test	continuous, asynchronous	discrete, blocking
Sequencing and Looping	1	3	0
Boolean Expression	1	1	1
Conditional	1	0	0
Coordination	0	0	3
Parallelization	0	1	1

Maintain Score	Basic	Developing	Proficient	
	count	test counter	state synchronized	event synchronized
Sequencing and Looping	1	1	3	1
Variable	2	2	2	2
Conditional	0	1	1	1
Boolean Expressions	0	2	2	2
Coordination	0	0	0	2
Parallelization	0	0	1	1

User Interaction	Basic	Developing	Proficient
	key press	mouse	keyboard input
Sequencing and Looping	1	1	1
Motion	0	1	0
User Interface Event	2	0	3
Variable	0	mouse pointer	answer

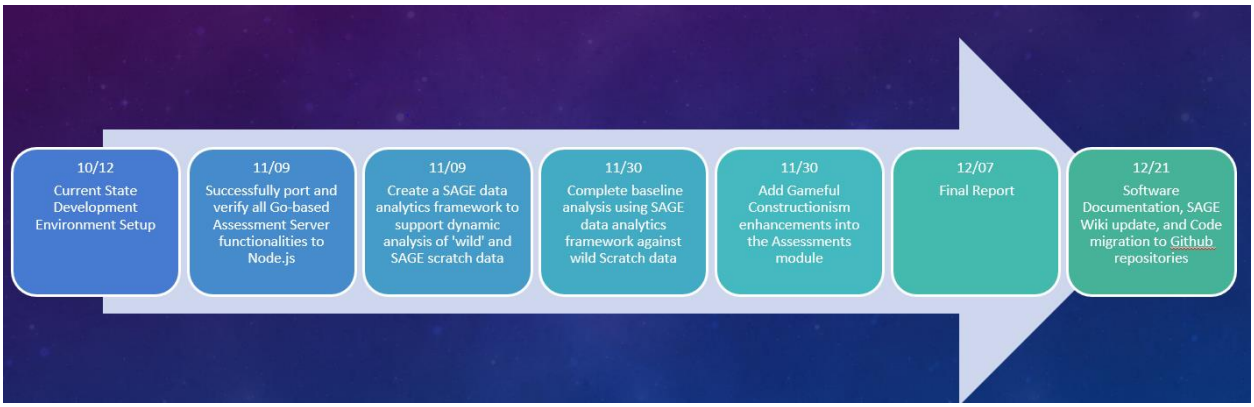
Figure 7. Numeric CT Scoring

	Procedures and Algorithms	Problem Decomposition	Parallelization and Synchronization	Abstraction	Data Representation
Basic					
Animate Looks	sequence				sprite properties
Animate Motion	sequence				sprite properties
Conversate	sequence				
Collide	conditional				
Maintain Score	initialize, then increment				new integer variable
User Interaction	sequence triggered by user interface				
Developing					
Animate Looks	initialize, then change			initial state	property assignment
Animate Motion	initialize, then change			initial state, 2D canvas	property assignment
Conversate	alternating say,wait sequence		mutlithreaded, blocking, synchronize on time delay		
Collide	test for collision in infinite loop	modularize collision detection	multithreaded, asynchronous	program state, continous event	
Maintain Score	increment, then test				new integer variable
User Interaction	move relative to mouse pointer				mouse pointer variable
Proficient					
Animate Looks	trigger animation as needed	modularize sprite animation	multithreaded, asynchronous, synchronize on state and event	program state and events	property assignment
Animate Motion	move relative to current location				relative property assignment
Conversate	trigger speaker as needed	modularize speaker role	multithreaded, asynchronous, synchronize on state and event	program state and events	
Collide	wait for collision		multithreaded, blocking	program state, discrete event	
Maintain Score	global reaction to score	modularize scene change	multithreaded, asynchronous, synchronize on state and event	program state and events	new integer variable
User Interaction	prompt, then use value		block on keyboard input		input storage variable

Figure 8. Computational Thinking Areas of Competencies

Milestones

The estimated timeframe of tasks related to this proposal are laid out as follows:



Future Work

Taking a peek of what future work we would be able accomplish once the work in this proposal is complete, we offer the following possible trajectories:

- Create an over-arching story and the accompanying assessment for every concept (Conditionals, Data, Events, Loops), practice, and perspective outlined in a Computational Thinking curriculum
- Devise guidance that integrate into optional learning paths to help students along as they enter Construction mode in Scratch Design mode
- Creation of specialized Game Mechanics Blocks to support constructionist game design
- Setup A/B Testing capabilities to aid in experimentation of changes within various SAGE Features
- Availability of teacher-friendly settings such a Constructionism Granularity Slider as lesson and assessment design aid for teachers
- Integration of the data analytics framework into Scratch Analyzer

References

- Anand, S., Sawyer, A., 2017. Enhanced Data Collection, Student Progress Modeling, and Intelligent Hinting in SAGE. Columbia University, New York, NY
- Bender, J., 2014. Scratch Analyzer: Transforming Scratch Projects into Inputs Fit for Educational Data Mining and Learning Analytics. Columbia University, New York, NY
- Bender, J., March, 2015. Tooling Scratch: Designing a Collaborative Game-Based Learning System to Infuse Computational Thinking within Grade 6-8 Curricula - Midterm Paper" Columbia University, New York, NY
- Berland, M., Baker, R.S. & Blikstein, P., 2014. Educational Data Mining and Learning Analytics: Applications to Constructionist Research. Technology, Knowledge and Learning, Volume 19, Issue 1-2, pp 205-220.

- B. Boe et al., 2013. Hairball: Lint-inspired static analysis of scratch projects. *Proceeding of the 44th ACM technical symposium on Computer science education*
- Carvalho, M. B., 2017. Serious games for learning : a model and a reference architecture for efficient game development. Eindhoven: Technische Universiteit Eindhoven
- Groff, J., Clarke-Midura J., Owen, V.E., Rosenheck, L., Beall, M., 2015. Better Learning in Games: A Balanced Design Lens for a New Generation of Learning Games
- Khandelwal, S., Mohanty, P., 2017. Computational Thinking Learning Platform for SAGE. Columbia University, New York, NY
- Kazimoglu, C., Kiernan, M., & Bacon, L., 2010. Learning introductory programming through the use of digital games in higher education. Paper presented at the Game Based Learning Conference.
- Kickmeier-Rust, M. D., Albert, D., 2010. "Micro-adaptivity: protecting immersion in didactically adaptive digital educational games," *Journal of Computer Assisted Learning*, vol. 26, no. 2, pp. 95-105
- Koh, K. H., Basawapatna, A., Nickerson, H., Repenning, A., 2014. Real Time Assessment of Computational Thinking, IEEE International Symposium on Visual Languages and Human-Centric Computing, Melbourne, Australia
- Halverson, R. & Owen, V.E., 2014. Game-Based Assessment: An Integrated Model to Capture Evidence of Learning in Play. International Journal of Learning Technology, 9(2), 111–138
- Kickmeier-Rust, M. D., and Albert, D., 2010. "Micro-adaptivity: protecting immersion in didactically adaptive digital educational games," *Journal of Computer Assisted Learning*, vol. 26, no. 2, pp. 95-105
- Mislevy, R., Haertel, G., 2006. Implications of Evidence-Centered Design for Educational Testing. *Educational Measurement: Issues and Practice,* volume 25, issue 4, Blackwell Publishing.
- Moreno-Leon, J., Robles, G., Roman-Gonzalez, M., 2015. Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking.
- Resnick, M., et al, 2009. Scratch: programming for all. Commun. ACM 52, 11 (November 2009), 60-67
- Rupp, A., Matthew G., Mislevy, R., Shaffer, D., 2010. Evidence-centered Design of Epistemic Games: Measurement Principles for Complex Learning Environments, JTLA, Vol 8, No 4
- SAGE Wiki. (2017, September). Columbia Programming Systems Lab. Retrieved September 2017, from SAGE Wiki: https://gudangdaya.atlassian.net/wiki/spaces/SAGE/pages/60293127/Getting+Started
- Schell, J., 2008. The art of game design: A book of lenses. San Francisco, CA: Morgan Kauffman
- Seiter, L. and Foreman, B., 2013, August. 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* (pp. 59-66).ACM

- Weintrop, D., Holbert, N., Wilensky, U., and Horn, M., 2012. "Redefining constructionist video games: marrying constructionism and video game design," in *Proceedings of the Constructionism 2012 Conference*
- Wing, J. M., 2006. "Computational thinking," Communications of the ACM, vol. 49, no. 3, p.33