Employing Retention of Flow to Improve Online Tutorials

Ashok Basawapatna
SUNY Old Westbury
223 Store Hill Road
Old Westbury, NY, 11568
1-516-628-5609
basawapatnaa@oldwestbury.edu

Alexander Repenning
University of Applied Sciences and
Arts Northwestern Switzerland
Windisch, Switzerland
alexander.repenning@fhnw.ch

ABSTRACT

Online CS Ed Week and Hour of Code activities attempt to motivate hundreds of millions of student participants across the world in computer science each year. A key goal of these endeavors is long-term student engagement. However, if the activity experience is bad, it could have effects adverse to the stated goal. Thus, it is imperative upon designers to actively improve the online activity ensuring the maximum numbers of students are retained throughout the exercise. We present a simple proof of concept method outlining a means for Computer Science Education Week and Hour of Code online activities to identify and improve hazardous points where students tend to drop out. This is achieved by finding so called flow stoppers in activity retention that diverge from an ideal theoretical Markov chain model, and scaffolding the activity at that point to better support participants. Initial data presented indicates that even minor changes can have a significant effect on keeping a greater number of students engaged.

CCS Concepts

* Applied computing–Interactive learning environments
* Applied computing–Computer-managed instruction

Keywords

Evaluation; online education; computer science education; retention; Flow; cyberlearning; CS Ed Week; Hour of code; usability

1. INTRODUCTION

Currently, there are many exciting and encouraging developments in the effort to expose students to computer science through programming and computational thinking. This is apparent in initiatives such as US Computer Science for All that, according to whitehouse.gov, aims to “empower a generation of American students with the computer science skills they need to thrive in a digital economy.” At the forefront of these initiative are organizations such as the Computer Science Teachers Association, Computer Science Education Week (CS Ed Week), and Code.org that are exposing participants in the hundreds of millions to computer science related activities [1,2]. These large participation numbers are in no small part due to online activities. aimed at students with little to no prior programming experience, offering a gentle-slope introduction to coding in a cyberlearning context [3].

Online programming activities are typically self-directed in highly scaffolded environments with embedded challenges. Examples of such activities include puzzle based coding challenges, such as Lightbot [4] Angry Birds [5], wherein users gradually learn sophisticated coding concepts as they solve a discrete sequence of increasingly difficult problems, and open-ended visual language environments, such as the Scratch Hide-And-Seek [3] and AgentCubes Online Frogger [6], that guide users step-by-step through the creation of an artifact such as a game or animation. In each case, a major aim of the activity designer is to provide users with the necessary affordances to engage them through activity completion and beyond.

Research indicates that a student’s first exposure to computer science, especially at the middle school level, is a crucial deciding factor in whether or not to continue in further pursuits of these activities [7,8]. If the experience is unsatisfactory, these activities risk having a negative effect on student motivation in computer science [9,10]. As participant numbers in these online instructional environments increase, activity evaluation and improvement is critical to ensure that the maximum numbers of students are engaged through activity completion and beyond. This in part necessitates systematic methods for identifying and improving instruction locations wherein students tend to drop out. Applying Flow theory to a Cyberlearning context provides a theoretical framework in which we can begin to understand one avenue of online activity improvement.

1.1 The Cyberlearning Loop

Cyberlearning is defined as networking and communication technology that support learning [11]. The Cyberlearning Loop posits that the same networking mechanism used to provide students with the activity can feedback valuable user data to the activity designer in order to increase the efficacy of the activity for future users [12]. Figure 1 depicts this general process.

![Closing the Cyberlearning Loop](image)

**Figure 1: Online activity instructions are informed by the learning objective and analysis of student created artifacts**

In Figure 1, online activity instructions are crafted to meet a learning objective. Ideally, the instruction difficulty and sequence are such that students are in a state of what Csikszentmihalyi describes as Flow [14] throughout the activity. To achieve flow at any given point of an activity necessitates that the challenge...
presented through instruction is closely matched to the student skill level. A person in a state of flow is characterized by complete immersion, focus, and enjoyment in the activity [13,14].

According to Flow theory, activity instruction that presents cognitive challenges not matched to a given student’s skill level risks an affective challenge inducing decreased student motivation [14]. In the Cyberlearning Loop context of Figure 1, an activity designer can provide an online exercise used by many students. Then the designer can analyze student data to improve the activity avoiding flow stoppers or points wherein student engagement is decreased. For example, let us say the feedback data shows that 90% of student participants stopped the activity at a given point because of a cognitive challenge presented by an instruction at that point. The activity designer can modify that instruction and analyze subsequent participant data to determine whether the instruction change reduced student drop out due to the given challenge.

This paper builds upon previous research [15,16] developing a method that online activity designers can easily employ to help participant retention by finding flow stoppers, addressing them and assessing the efficacy of this adaption process. Using data from the 2014 Hour of Code exercise, an activity flow stopper is identified by comparing aggregated participant data to an ideal model of user participation. The activity instruction is then improved with respect to this point, and user data from the 2015 Hour of Code activity is analyzed to determine if the change enabled participants to overcome this cognitive challenge. Finally the possible further applications and limitations of this method are discussed.

2. Moving Towards The Cyberlearning Loop

As part of Hour of Code 2014 and 2015, an online activity was created aimed at students with little prior programming experience. The activity guided participants through the creation of a 3D version the classic 80’s arcade game Frogger in the AgentCubes online end-user programming environment [6,15,17]. This activity was based on the Scalable Game Design curriculum, which aims to broaden participation in computer science and engage middle school classrooms in computational thinking through the gentle-slope creation of games and simulations [18].

The browser-based online activity instructions are provided by an embedded video with an index of topics allowing users to stop and minimize the video at points when instruction is deemed unnecessary or jump to a given topic during instances where guidance would be helpful. In addition to US participants, this activity was used to launch CS Ed Week in Switzerland and as part of a CS Ed Week pilot in Mexico [6,17]. To support all regions of the participating nations, the activity instructions can be played in English, Spanish, German, Swiss, French, and Italian. A more in-depth description of the activity can be found in [17].

The activity including the Computational Thinking Tool [24] is cloud based wherein every time the user updates their Frogger game, the change is saved on a central server. This enables post-activity analysis of the submitted games to better understand where students might have had issues with the activity. The submitted games can be quite complex, with multiple agents, rules, conditions and actions. To get a clear picture as to which parts of the activity were problematic for students, we created an instrument to measure student retention through the activity entitled the “Retention of Flow” [15].

2.1 Hour of Code Activity Data

One simple piece of data that was illuminating in terms of how students went through the activity involved plotting the percent of total participants that were able to make it to a given tutorial instruction. This involves gathering all the games made during the Hour of Code and analyzing their generated XML files to establish a Lines of Code (LOC) measure for each game. It should be noted that this measure could be applied to the analysis of any activity that has a large number of saved participant user data. For example, the LOC measure can be as simple as counting the number of ‘<’ characters generated in agent behavior tags. Next a determination can be made as to which LOC values correspond to which activity task. Finally, by summing up the percentage of games equal to or greater than a given LOC value, we can calculate the participant retention through each point of the activity [15].

For example, at the beginning of the 2014 activity, we have retained 100% of participants. The end of the first task, wherein the user programs the frog to move up, occurs at 9 LOC. By analyzing the games created we see that 96% of the participants have games greater or equal to 9 LOC [15]. To put this another way, 4% of participants dropped out between activity beginning and the first instruction. Of course this drop in and of itself does not indicate the type of challenge these users faced. In [16] we theorize that this drop in participant retention could be due to technical issues, such as a power outage, cognitive issues, such as confusing activity instruction, and practical issues, such as the class period ending. Since this drop gradually occurs before the first instruction, it could be anything from students who are window shopping Hour of Code activities (practical) to changing computers or even trying to run AgentCubes online, which requires WebGL, on the Internet Explorer browser (technical).

Interestingly, by plotting the percentage of students retained vs. the LOC value itself, we obtained a curve that could be closely fit to a negative exponential [6,15]. Piech et al. found a similar negative exponential trend when analyzing the retention of students through the Angry Birds Hour of Code activity, which is a discrete puzzle based coding activity [5].

As we analyzed the data, we noticed interesting patterns at points where the data diverged from the negative exponential trend. Attempting to better understand these interesting inflection points and explain the negative exponential trends apparent in the AgentCube Online and Angry Birds Hour of Code activities, led to a derivation of a simple mathematical model.

2.2 Modeling Hour of Code Type Activities

As we further explored activity locations wherein the participant retention data diverged from the negative exponential trend, we began to notice that these points corresponded to flow stoppers. We began wondering why this might be the case.

Figure 2, from [16], depicts a simple Markov chain derived to model the activity retention. If we conceptualize a tutorial as a sequence of discrete steps, at each step the user makes a binary decision. Specifically, if the user is engaged in the activity the user will proceed, and be retained, to the next step. However, if the user’s motivation falls below a certain threshold, they will stop activity participation and do something else. These state transitions can be expressed as probabilities $P(\text{cont})$ to continue from the current step to the next one. $P(\text{stop})$ is the probability to stop with $P(\text{stop}) = 1 - P(\text{cont})$. These probabilities reflect how much participants have enjoyed an activity so far combined with
their subjective prediction of how challenging the next steps appear to be [16].

Ideally, Activity designers want to keep participants in Flow by matching the challenges posed with acquired skills. Given that the Hour of Code hosts a large number of diverse participants worldwide, an activity cannot possibly have every participant be in a state of Flow. However, the activity designer can attempt to design a set of instructions wherein if the user can do step one, they can do all the subsequent steps in the activity. For example, in some IKEA furniture and LEGO construction assembly guides, it makes sense to keep all the instructions roughly at the same level of challenge. The Retention of Flow conjecture suggests that devoid of any marked increase in instruction challenges, all the continuation probabilities at each step of the tutorial are more or less identical. Given this to be the case, it is sufficient to conclude that the resulting function would be a negative exponential function with a P(cont) probability to reach the endpoint of an n-step tutorial. Figure 2 depicts a Markov chain model based on this conjecture as well as the resulting retention plot predicted by the model.

The Markov chain model in Figure 2 suggest a negative exponential retention curve emerging from identical probabilities to continue (red curve in Figure 2). Figure 3 shows actual probabilities (red curve) computed from the actual retention data (blue curve). The black line is the ideal negative exponential fit produced by the Markov chain-based model. Overall, the actual probability to continue shows remarkable constancy with a minimal negative trend. This finding is consistent with the Retention of Flow conjecture. The probabilities scales of Figure 3 are on the right hand side (with an average of 0.989) and the retention percentage scales are on the left hand side. Identifiable flow stoppers (labeled “Drop”) and more general trends of increased or decreased Flow (labeled “+Kink” and “−Kink”) are explained in detail in [16].

2.3 Data-Model Divergence

Based on the model outlined in Figure 2, we can start to better understand the possible meaning of why data might diverge from the trend line. Specifically, given that the probability of continuing (Figure 3) the activity is identical for each step, points in the activity where the retention data diverges from the model would imply that the challenge at that step was markedly different from the challenge presented at previous steps. Or to put it another way, being able to complete the first step of this activity did not necessarily mean that users could complete all subsequent steps. As mentioned in section 2.1, the nature of this challenge could be cognitive, technical, and practical. As a means of modifying activity instructions this paper will focus on the affective cognitive challenge [16].

Figure 2: Retention of Flow model based on Markov Chain with identical probabilities [16].

Figure 3: 3D Frogger Retention of Flow (blue, 0-100%) and Continuation Probability (red, 0-1.0) graphs as function of Lines of Code (0-200) [16].

3. METHOD FOR IMPROVING HOC ACTIVITIES

By identifying where retention data for a given activity deviates from the model, we can quickly identify flow stoppers where student engagement is decreased. By investigating these points, for example by looking at a sample of games created with that many Lines of Code, we can begin to understand whether the challenge is cognitive, technical, and/or practical. If there is evidence that the challenge is cognitive, we can improve the activity by adding scaffolding such that student skills at that point better match the instructional challenge, keeping students in the Flow. Ideally, applying this procedure to all the cognitive challenges discovered in an activity will increase student retention.

As outlined in section 2, we created an Hour of Code activity that guided users through the creation of a 3D Frogger game in the AgentCubes online environment. In [16] we were able to analyze the 2014 retention data to identify a cognitive challenge at 27 LOC, wherein retention drops a around 2%. For the 2015 activity, in order to test this method, we improved the activity just at one point, by scaffolding the activity instruction to spare students encountering this flow stopper. Our hope was that by modifying the activity in this way, we could decrease student drop out due to this cognitive challenge.

3.1 The Cognitive Challenge

Investigating the retention drop at 27 LOC, by looking at the behavior code of 2014 Frogger games, it became apparent that students who stopped at this point had trouble programming the frog agent to move four directions (up, down, left, and right) when the corresponding arrow keyboard key was hit. This is the first interaction that users program in the activity, and this pitfall is one Scalable Game Design teachers often run into with their students. The paper focuses on this particular flow stopper because we have observed it and even gave it a name (the “Jumbo Rule”) in many classrooms.
The first behavior rule students are instructed to create in Frogger moves the Frog agent up when the up arrow key is hit. Figure 4 depicts this. Very few students get this wrong.

If the user creates the Jumbo Rule, by adding all four key conditions and all four move actions into the same rule, their frog will not react to cursor keys at all and, consequently, their game will be unplayable. This is likely to cause frustration among students and result in a decrease in activity retention.

3.2 Improving The Activity

The 2014 version of the Frogger 3D activity, participants, mostly novice programmers, were left to their own devices to figure out how to debug this problem. In 2015 we added a small critiquing system [20] to AgentCubes online to address this kind of cognitive challenges. Whenever a user adds a second key condition, a warning message pops up, depicted in Figure 7, not only pointing out the potential problem, but also suggesting a fix.

Users are then instructed to test their program. At this point the Frog will only move one direction: up when the up arrow key is hit. Further instructions illustrate how to create three new rules enabling the frog move in all four directions. The correct implementation of this is depicted in Figure 5.

Multiple key conditions detected in the same rule. This is a programming error. Please make new rules for each key condition.

Figure 7: Trying to prevent the Jumbo Rule with a critiquing system: Error message with suggested fix

The pop up slowly fades away on its own, and is such that a more advanced programmer can ignore the message if placing two key conditions in the same rule is in fact what they were intending to do.

It should be noted that fixing this activity instruction could have been integrated into the instructional video. For example, by showing the process of creating the four necessary rules and perhaps even showing a bad example illustrating how not to implement this behavior a user, in this specific situation, might be guided in the right direction. As a general guideline this instructional strategy, particularly for novice programmers, could be overwhelming because for each correct way there are, generally speaking, many bad ways to implement something. This strategy could significantly expand the instructional material to the point where the instructions become overwhelming or boring to a beginning programmer.

We decided against this strategy and, instead, implemented a simple critiquing system [20]. A critiquing system is a design environment that provides a large degree of freedom to users but will point out potential problems. In other words, the idea is that we allow users to make mistakes, and only then does the system intervene. This approach seems to be more aligned conceptually with events, such as the Hour of Code, which prioritize activity completion over teaching specific topics. Furthermore, this strategy helps users generally avoid this pitfall in other, non Hour of Code activity contexts.

4. RESULTS

Table 1 compares the number of students in the 2014 and 2015 AgentCubes Hour of Code activities that placed multiple key conditions in one rule. The data is split into three categories: Switzerland, United States and Total (note that Switzerland and the United States made up the vast majority of submitted projects both years).
The method assumes that activities have a measure of success in terms of the number of projects observed with key-condition cognitive error and total number of projects for Switzerland (CH) in green, United States (US) in blue and Total (worldwide) in red. * denotes significance at p<0.05.

<table>
<thead>
<tr>
<th></th>
<th>Number of Projects with &gt;1 key cond. in a rule</th>
<th>Total Projects</th>
<th>Proportion of projects with &gt;1 key cond. in a rule</th>
<th>% Reduction in Projects with &gt;1 key cond. in a rule</th>
<th>p value two tailed z-test on 2 population proportions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH 2014</td>
<td>41</td>
<td>1976</td>
<td>2.1%</td>
<td>38.1%</td>
<td>0.02852*</td>
</tr>
<tr>
<td>CH 2015</td>
<td>47</td>
<td>3589</td>
<td>1.3%</td>
<td></td>
<td>0.00854*</td>
</tr>
<tr>
<td>US 2014</td>
<td>50</td>
<td>1622</td>
<td>3.1%</td>
<td>41.9%</td>
<td>0.00056*</td>
</tr>
<tr>
<td>US 2015</td>
<td>36</td>
<td>2047</td>
<td>1.8%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total 2014</td>
<td>94</td>
<td>3813</td>
<td>2.5%</td>
<td>40.0%</td>
<td></td>
</tr>
<tr>
<td>Total 2015</td>
<td>87</td>
<td>5836</td>
<td>1.5%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Analyzing the data we see that the activity improvement between 2014 and 2015 dropped the proportions of projects containing the error described in section 3.1. Using a two tailed z-test for 2 population proportions, the drop between 2014 and 2015 were found to be significant in the Swiss, US, and Total populations for p<.05.

There are a few notable aspects to this data. First, though the Frogger Hour of Code activity has multilingual instruction, the error message depicted in Figure 7 only displayed English. However, Table 1 indicates that the error message had a significant effect on the Swiss students whose first languages are German, French, and Italian depending on the canton they reside. The second is that though the intervention led to a significant decrease in projects wherein the multiple key condition mistake was present (.031 in 2014 to .018 in 2015), one might wonder why the problem was not eliminated even more in the United States wherein English is the first language. This is hard to tease out with the current data, but one possibility is that this may be due to the tendency of avid Internet users to treat pop-up messages with suspicion [21].

5. DISCUSSION
This method for activity improvement parallels similar methods commonly employed in user testing. Namely, in A-B testing a website designer, for example, might serve two different versions of a given webpage to two different populations to test the efficacy of each design [22]. There are two modifications between this method and A-B testing. The websites in A-B testing are usually served out around the same time span. In the method presented, given the educational context, serving out two different versions of the activity might not be advisable based on the goal to maximize student retention. Furthermore, in this study, one cognitive challenge was identified and fixed as a proof of concept. However, now that the method has shown to be effective in this limited scope, in coming years many cognitive challenges can be fixed at once.

Additional points of discussion revolve around the model presented. The narrative behind the model uses an example of LEGO or IKEA construction wherein the user can drop the activity at any time and walk away. Of course this may or may not be true in an Hour of Code context. For example, a student might be participating in the activity as part of in-class instruction, and thus may not have the ability to simply drop the activity. However, it should be noted that even in cases where the student does not drop out because of the cognitive challenge, the presence of the challenge itself could hamper student-progress by delaying the student at that point in the activity. This student might not show up in the retention data as a drop-out at that particular point, but might not get as far or have as positive an experience as someone who was not hampered by this cognitive challenge.

A key part of the method, The Retention of Flow conjecture, suggests that devoid of any marked increase in instruction challenges, all the continuation probabilities at each step of the tutorial are more or less identical. This conjecture helps provide a plausible explanation for the observed phenomena of retention data diverging from the model at given points of the activity instruction. Furthermore, it yielded actionable steps towards activity improvement. However, though a set of instruction steps might approximately be at an equal level of challenge to some notion of an average student, an individual student, with different strengths and weaknesses, might have a much different view of how the challenges change between steps. This conjecture assumes an Hour of Code type activity with a high number of diverse participants. Future research should explore this conjecture deeply to better understand the conditions under which it is true and further possible implications it may have for activity design.

The intervention focused on a place early in the activity wherein a cognitive challenge was detected. One might wonder why this specific challenge was focused on given that only around 2% of 2014 participants dropped out at this point of the activity. One reason, alluded to above, is that this cognitive challenge affected more than just the people whose 2014 Frogger games ended with 27 LOC. It presented a potential flow stopper at possibly the most important point of the activity, the very beginning. Furthermore, given the three types of challenges: cognitive, practical, and technical, this point is one identified as purely cognitive, meaning that we could better test the method developed for activity improvement.

Finally, it is worth discussing the limitations of the method, and what types of Hour of Code activities this method can be used on. The method assumes that activities have a measure analogous to LOC and are used by a large number of participants. For open-ended activities, it assumes access to data files that grow proportionately with programming rules. In puzzle-type challenges, a combination approach might be employed. For example, retention at each discrete puzzle can be measured, but also, retention within each puzzle can be determined by analyzing generated files. In both types of activities, the method can be one of many tools an activity designer uses to create a better user experience.

6. CONCLUSION
This paper shows that adding a critiquing system to an Hour of Code like activity can significantly improve the retention of participants. More importantly, however, this paper presents a
method to improve learning activities. This method combines the Retention of Flow instrument with a theoretical model enabling activity designers to identify potential flow stoppers, adapt the activity, and validate the efficacy of the instructional change. In this way the activity designer can close the Cyberlearning Loop. Initial results indicate that this method can help significantly decrease a single participant coding error due to a cognitive instructional challenge. Further research will look at applying this method throughout the exercise and on a larger scale to see if overall activity retention improves as a result, and explore improvements to the method itself in order to keep more novice students engaged and motivated as they learn how to code.

7. ACKNOWLEDMENTS
This work is supported by the Hasler Foundation, the Swiss National Science Foundation under grant CRAGP2_158545, and the National Science Foundation under Grant Numbers 0833612, 1345523, and 0848962. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of these foundations. Thanks to Prof. Frank Sanacory and Mark Savignano for their help.

8. REFERENCES