The Impact of Culture on Learner Behavior in Visual Debuggers

Kyle Thayer
Paul G. Allen School of Computer Science & Engineering
University of Washington
Seattle, WA, USA
kthayer@cs.washington.edu

Philip J. Guo
Dept. of Cognitive Science
UC San Diego
La Jolla, CA, USA
pg@ucsd.edu

Katharina Reinecke
Paul G. Allen School of Computer Science & Engineering
University of Washington
Seattle, WA, USA
reinecke@cs.washington.edu

***Abstract*—People around the world are learning to code using online resources. However, research has found that these learners might not gain equal benefit from such resources, in particular because culture may affect how people learn from and use online resources. We therefore expect to see cultural differences in how people use and benefit from visual debuggers. We investigated the use of one popular online debugger which allows users to execute Python code and navigate bidirectionally through the execution using forward-steps and back-steps. We examined behavioral logs of 78,369 users from 69 countries and conducted an experiment with 522 participants from 82 countries. We found that people from countries that tend to prefer self-directed learning (such as those from countries with a low Power Distance, which tend to be less hierarchical than others) used about twice as many back-steps. We also found that for individuals whose values aligned with instructor-directed learning (those who scored high on a "Conservation" scale), back-steps were associated with less debugging success.**

***Index Terms*—program visualization, cross-cultural studies, non-linear learning**

I. Introduction

People from over 180 countries are learning computer programming from online resources [1], [2]. Though interest in learning to code is widespread internationally, the dominant programming education tools and MOOC platforms that teach such skills (e.g., edX, Coursera, Udacity, Codecademy, Code.org) were developed by people in the United States. This creates a risk that designers may have unconsciously embedded their cultural values into these platforms, making them less suitable for people in other countries. Indeed, researchers have raised concerns about whether MOOC and programming education resources are optimized for, and primarily benefit, those who come from more privileged Western-centric backgrounds [3]–[7]. One of the reasons for these concerns is that a country's national culture has been found to influence how people learn [7]–[9]. For example, prior work found that countries with a high power distance—an indicator of strong hierarchies, such as found in India and China [10]—often employ instructor-directed education [8]. Students who grow up in these countries are thought to be more used to and may prefer step-by-step instructions and more linear navigation [2], [11]. In contrast, students from countries with a low power distance, such as the U.S. and Denmark, exhibit more self-directed learning and might prefer more self-guided and less linear navigation. This phenomenon has been seen in MOOCs, where students from low power distance countries navigate the content more non-linearly (i.e., jumping back and forth between different sections) than those from high power distance countries [2].

978-1-5386-4235-1/18/$31.00 ©2018 IEEE

Our main question in this work is whether such differences can also be found among users of visual debuggers. *Do people from different cultures use and benefit from visual debuggers in distinct ways?* and more specifically, *Does a propensity for self-directed learning explain some of these differences?* If yes, this would suggest that visual debuggers might have to be adjusted to optimally support users from different cultural backgrounds. It would also indicate that cultural differences in behavior prevail even within a relatively homogeneous group of users who seek out online debugging tools to learn programming.

As a first concrete step toward investigating these questions, we evaluate how learners from over 60 countries engage with a specific feature within Python Tutor, a popular visualization-based online debugger often used in conjunction with programming tutorials [12]. Central to Python Tutor is the beginner-friendly feature of bidirectional navigation of code executions [13]. This feature allows users to jump both to earlier steps ("back-steps") and later steps ("forward-steps") of the code execution while running a piece of code (Figure 1). Given the prior work on the influence of culture on the level of self-directed learning [2], [8], we hypothesize cultural levels of self-directed learning will correlate with navigation by back-steps in Python Tutor. We conducted two quantitative studies to probe this hypothesis, using two proxy measures for instructor-directed learning: *Power Distance Indicator* (PDI, a measure of how hierarchical a country is) and *Conservation* (a measure of how much an individual values tradition, conformity, and security) [10], [14].

In our first study, we analyzed behavioral log data from 78,369 users of Python Tutor over six months. We found significant differences between countries in how many back-steps their users took. In particular, users from low and medium PDI countries, such as Israel, Germany or the US, took *more* back-

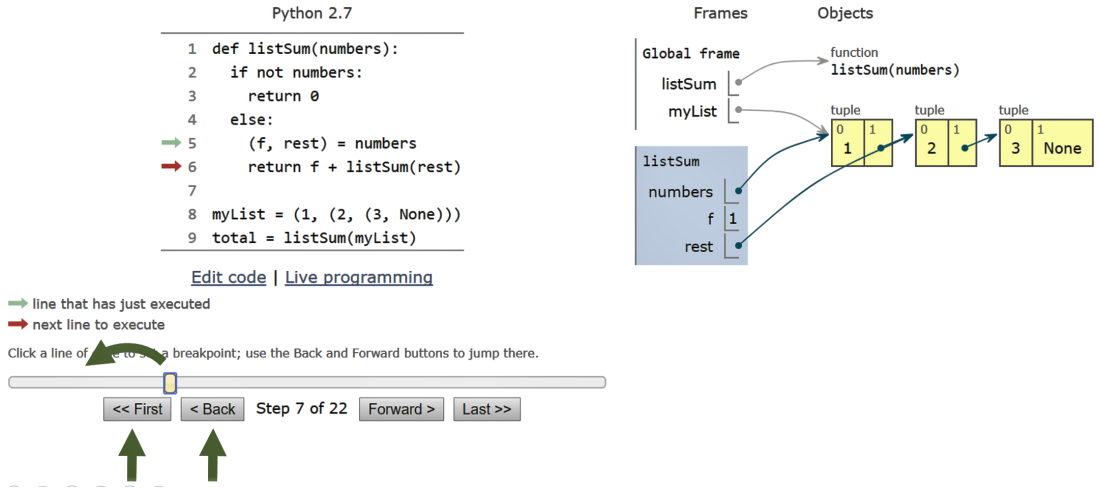

Fig. 1. Python Tutor [12] lets learners navigate through example programs and visually debug their code. The large green arrows annotate the three ways of performing *back-steps* to jump back to earlier execution points.

steps when following code executions on Python Tutor than those from high PDI countries, such as India, China, or Russia. People in the most egalitarian countries (with low PDI and self-directed education, where students are often encouraged to find their own way to solve problems) took about twice as many backward steps through code executions than those from the most hierarchical countries (with high PDI and instructor-directed education).

Since individuals' culture and their propensity for self-directed learning varies within countries, we conducted a second study to investigate the relationship between back-steps and self-directed learning at an individual level. For this study we recruited 522 participants to perform a debugging activity on Python Tutor and asked them to answer a questionnaire to assess Conservation as an individual measure of cultural values. Participants' Conservation scores were marginally correlated with the number of back-steps. We did not find a correlation between PDI and back-steps as in the first study, but we again found differences between some countries in the use of back-steps. In addition, contrary to our expectations, back-steps correlated negatively with debugging success, but this effect varied with Conservation score.

Altogether, our results show that people do not uniformly use visual debuggers and do not equally benefit from certain functionalities. The national cultural dimension Power Distance and individual's Conservation score can predict some of these differences. Our study makes the following new contributions to the research area of cross-cultural influences on learning technologies:

1) We contribute the first studies of the effects of learners' culture on their use of visual debuggers. We found differences in how learners from various countries use the back-step feature in Python Tutor, a widely-used online debugger commonly used with tutorials. Our studies suggest that these differences can be partially explained by users' level of self-directed learning as measured by Power Distance and Conservation.
2) Our results showed that for individuals whose values aligned with instructor-directed learning (high Conservation), back-steps were associated with less debugging success.
3) Our findings also point to how users from different cultures may benefit from different presentations of non-linear navigation features in online programming education tools. The Power Distance Index, which can be easily derived from IP addresses without any additional input from the user, can be used as a rough approximation of culture when doing these adaptations.

II. The Python Tutor Website

The Python Tutor website [12], [15] is an open-source code visualization system that allows learners to edit and debug code directly in their web browser. The system has two views: 1) a code editor view allows users to write code and press a button to run their code, which opens 2) a run-time state visualization view (Figure 1) that lets users debug their code by allowing them to navigate all of the steps of program execution, both forwards and backwards, using a slider or buttons. At each step the user sees all variables, values, stack, heap, and textual output at that point in execution.

The Python Tutor website hosts a set of basic programming examples where learners can execute the example code and step through visualizations of its run-time state. In addition, many users copy and paste code from other websites (e.g., MOOCs, blogs) into Python Tutor's code editor to understand and debug it using the visualizations of its run-time state.

III. Background and Hypothesis Development

We developed hypotheses based on prior work on cultural measures and how those relate to people's behavior.

A. Culture and Back-Steps

According to Hofstede, culture describes a shared "programming of the mind" [16], which results in groups of people having shared values and preferences [17]. Culture is not easily defined; in fact, researchers debate what exactly it describes and what influences culture has. Culture cannot be constrained

to country borders [18], but people from the same country can still share a national culture and might often adhere to certain behavioral trends [10], [16].

Grappling with the issue of trying to define cultures, researchers have attempted to quantify differences between cultures, while acknowledging that any differences can only describe trends and are not going to generalize to all members of a specific culture. Two notable efforts to measure culture are by Hofstede [10] and Schwartz [14]. Hofstede's cultural dimensions measures culture at a national level, while Schwartz found a universal structure to the value trade-offs individual people make, holding true across different countries.

From Hofstede's cultural dimensions, his Power Distance Index (PDI) is the most relevant to the aspect of self-directed learning that we are investigating. PDI measures "the extent to which the less powerful persons in a society accept inequality in power and consider it normal" [8]. Societies with a higher PDI (e.g., India or China) tend to have more "teacher-centered education (premium on order)," where the "students expect the teacher to outline paths to follow," the "teacher is never contradicted nor publicly criticized," and the "effectiveness of learning [is] related to the excellence of the teacher" [8]. Learning in these high PDI environments is centered on the authority of the instructor and thus is instructor-directed. In contrast, societies with a lower PDI (e.g., the US or many Western European countries) have more "student-centered education (premium on initiative)," where the "teacher expects students to find their own path," the "students [are] allowed to contradict or criticize the teacher," and the "effectiveness of learning [is] related to amount of two-way communication in class" [8]. Education in these low PDI environments is centered on each student's individual authority and thus is more self-directed. Lower PDI countries also tend to have more resources available to put toward education: they have smaller class sizes[1] and higher GDP per capita[2] (see also [2]).

Such differences in day-to-day education likely translate into people's learning behavior online, even after they have finished school. Indeed, low student-teacher ratios in a country (which is associated with self-directed learning [8]), was found to correlate with students making more "backjumps" in MOOCs, where they navigate to earlier course content [2]. For those in high PDI countries, prior research has suggested providing linear navigation, reducing navigation choices and providing support through wizard interfaces [11], [21].

Inspired by this line of work, we expect programming learners to view Python Tutor as a computerized "instructor" and thus that learners from high PDI countries (with more instructor-directed learning) will view individual steps in the code visualization as the canonical intended path offered by Python Tutor. Since Python Tutor gives no explicit instructions to step either forward or backwards, we expect these users to assume any steps, from the first to last execution step, were intended to be followed forward in a linear order.

Conversely, we expect users from low PDI countries (with more self-directed learning) assume that Python Tutor (as a computerized "instructor") gives them a space of options to explore, and that they will see the execution steps as intended to be navigated in whatever order best fits their needs. Thus, we hypothesize that users from higher PDI countries will take fewer back-steps, and users from lower PDI countries will take more back-steps:

[H.1] *PDI negatively correlates with the number of back-steps that users take in Python Tutor's code visualizations.*

Since national cultural measures (like PDI) do not take individual differences into account, we also wanted to investigate how personal values of self-directed learning relate to using back-steps. We wanted to use a validated individual cultural measure for this, so we chose the one most relevant to the aspects self-directed learning that we are investigating: Conservation vs. Openness-to-change from Schwartz's universal values work [14]. Schwartz's values have been shown to correlate with decision making, political views, and observed behavior [22]–[24], with those who score higher on openness-to-change being more willing to follow their own interests in unpredictable directions [14]. Since Conservation can be an appropriate proxy measure of self-directed learning like PDI, we assume it will relate to how back-steps are used. Our second hypothesis is therefore:

[H.2] *Conservation score will negatively correlate with back-steps in Python Tutor code visualizations.*

B. The Efficacy of Back-Steps for Code Debugging

The closest related technical systems to Python Tutor are backwards-in-time debuggers that allow a user to navigate from a given point in code execution back to earlier execution steps. This feature allows programmers to find the causes of bugs without guessing where to set breakpoints [25]–[27]. Most research on backwards-in-time debuggers has focused on the debugging techniques and technical implementations [25]–[31]. To our knowledge, there have been no prior studies of how users' national culture and personal values affect their use of code debuggers.

Since backwards-in-time debuggers were specifically built to help with debugging (and one study on a specific variant found them to be beneficial [31]), we hypothesize that back-stepping will generally correlate with more debugging success (in terms of how many tests the modified user code passes):

[H.3] *Back-steps in code visualizations will correlate with debugging success.*

We also hypothesize that self-directed learners (low Conservation) will have had more experience choosing their own path and be more comfortable breaking from linear orders. These learners might be more prepared and able to make effective use of back-steps. Therefore we expect the use of back-steps by self-directed learners to more likely result in debugging success than the back-steps of instructor-directed learners. Thus, we expect an interaction effect: Self-directed learners

[1]PDI is correlated with student-teacher ratio (using data provided by [19]), $r(47) = .37$, $p < .01$

[2]PDI is negatively correlated with GDP per capita (using data from [20]), $r(65) = -.62$, $p < .0001$.

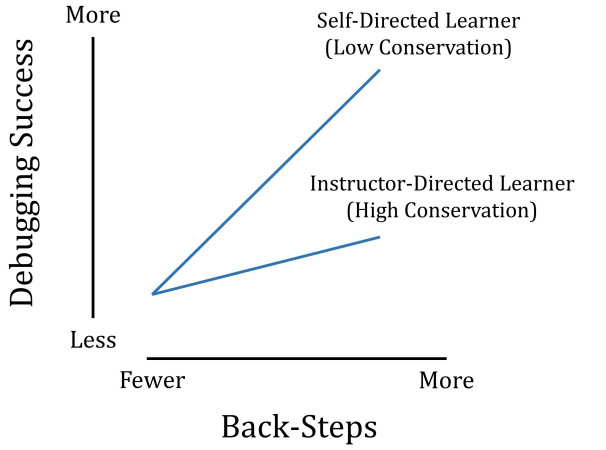

Fig. 2. Hypothesis H.4: We expect an interaction effect between self-directed learning, back-steps, and debugging success. We expect debugging success to positively correlate with back-steps for all learners. But we expect self-directed learners benefit more from any back-steps they take, and thus will have a larger positive correlation between back-steps and debugging success.

(low Conservation) benefit more from back-steps (thus a larger correlation between back-steps and debugging success) than those participants who are used to instructor-directed learning (see Figure 2):

[H.4] *Conservation interacts with the correlation between back-steps and debugging success. For lower Conservation learners, the correlation will be stronger, while for higher Conservation learners, the correlation will be weaker.*

IV. Study 1: Python Tutor Behavioral Log Analysis

For our first study, we examined behavioral log data from Python Tutor in order to test if back-steps are negatively correlated with Power Distance (H.1).

A. Methods

To test our hypothesis, we retrieved six months of behavioral log data from Python Tutor and supplemented the dataset with the Power Distance scores for each user's country. The dataset comprised the following:

- User events, allowing us to calculate features such as back-steps, forward-steps, time spent, and code length;
- 78,369 unique user IDs (UUIDs);
- Browser sessions, allowing us to track user events across multiple code visualizations in a session;
- User country, deduced from their IP address using the GeoLite2 Free database [32];
- The Power Distance Index for each user based on their country and Hofstede's official country PDI scores [33].

Python Tutor does not ask users to sign up or provide any demographic information, so we were unable to control for possible effects of demographics.

1) Users: Our dataset included 147,847 users who visited the Python Tutor website from 166 countries. We removed users who did not use code visualizations, who did not take any steps in a code visualization, or whose country was not part of Hofstede's study and therefore could not be linked to PDI. The final dataset included 1,236,863 code visualizations run by 78,369 unique users from 69 countries. The US accounted for 32% of the data, India for 7.8%, and the UK for 5.3%. The average PDI for users was 50.4 ($SD = 17.8$), which is roughly half of the maximum possible PDI value of 120.

2) Analysis: We conducted a series of mixed-model analysis of variances on code execution visualizations, with the back-step count as the dependent variable.[3]. We modelled PDI as an independent factor. Because we wanted to do our analysis on individual code execution visualizations and a large numbers of users who interacted with multiple code visualizations, we modelled user ID as a random factor. Since Python Tutor users' exact tasks were unknown to us (users were free to follow any example on the site or copy and paste in any code from elsewhere), we controlled for 13 additional variables (see Table I) that measured either engagement (such as time spent and forward-steps) or code properties (such as length of code and number of exceptions thrown when running the code). To further understand user's tasks and the code they were running, we also examined all code executions for 20 random browser sessions in four different countries with at least 10,000 code executions: two with high PDI and few average back-steps (Russia and India), and two with low PDI and more average back-steps (Israel and Australia).

B. Results

Our linear regression confirmed H.1: Power Distance was negatively correlated with the number of back-steps in a code execution visualization ($F_{(1,47489)} = 84$, $p < .0001$, $\beta = -.052$, $t\text{-}value = -9.1$) (Figure 3). For example, picking the most and fewest average back-steps per code execution for countries with at least 10,000 code executions, we found significant differences between Israel ($M = 1.7$, $SD = 4.2$, $PDI = 13$) and India ($M = 0.38$, $SD = 1.7$, $PDI = 77$); $t_{(-49)} = 26206$, $p < .0001$.[4] For the other variables, higher engagement with the Python Tutor tool correlated with more back-steps; most notably with forward-steps taken ($F_{(1,1235156)} = 178878$, $p < .0001$, $\beta = 1.2$, $t\text{-}value = 423$), time spent in the visualization ($F_{(1,1191571)} = 7090$, $p < .0001$, $\beta = 0.23$, $t\text{-}value = 84$) and length of the code (in characters) ($F_{(1,881703)} = 2201$, $p < .0001$, $\beta = 0.15$, $t\text{-}value = 46$). The full regression table is shown in Table I.

Examining all code executions for randomly selected browser sessions allowed us to see how users were modifying and executing code. Our observations included that code and apparent task varied greatly within countries; in addition, we saw few differences between the countries. Code being edited ranged widely, from apparent tests of how python list functions worked to sorting functions to dice rolling games to string processing, all with no apparent difference between the high

[3]While back-steps were not normally distributed, linear regressions are thought to be robust to outliers and other violations of assumptions for large samples such as ours [34]

[4]While these are the medians of skewed data, the large sample size still allow for comparison. [34]

TABLE I

ANALYSIS OF VARIANCE RESULTS FOR ALL FACTORS IN THE REGRESSION MODEL FOR *back-steps* (STUDY 1), EXCLUDING USERID, WHICH WAS A RANDOM FACTOR. FACTORS CAN BE AT ONE OF THREE SCOPES: $User$ IS A VALUE THAT IS CONSTANT FOR A USER ACROSS ALL TIME; $Execution$ IS A VARIABLE SCOPED TO A SINGLE CODE VISUALIZATION EXECUTION; $Session$ IS A VARIABLE THAT IS SHARED ACROSS A BROWSER SESSION BY A USER WHERE THE USER MAY HAVE RUN MULTIPLE CODE VISUALIZATION EXECUTIONS. THIS MODEL SHOWS THAT PDI NEGATIVELY CORRELATED WITH *back-steps*, AND THAT MANY FACTORS MEASURING ENGAGEMENT (SUCH AS $Time$) WERE POSITIVELY CORRELATED WITH BACK-STEPS. WE ALSO INCLUDED THE COEFFICIENTS FROM THE LINEAR MODEL TO ALLOW COMPARISON (ALL NON-BOOLEAN INDEPENDENT VARIABLES WERE NORMALIZED). MARGINAL R^2 = .18 (VARIANCE EXPLAINED BY FIXED FACTORS), CONDITIONAL R^2 = .28 (THE VARIANCE EXPLAINED BY FIXED AND RANDOM FACTORS COMBINED).

Scope	Variable	Coeff.	df	F	p-value
User	PDI	-0.052	1	84	$<$ .0001 ***
Execution	Time	0.23	1	7090	$<$ .0001 ***
Execution	# steps available	0.024	1	55	$<$ .0001 ***
Execution	# of forward-steps	1.2	1	178878	$<$ .0001 ***
Execution	Length of code (# chars)	0.15	1	2201	$<$ .0001 ***
Execution	Edit-dist. from previous execution	-0.041	1	240	$<$ .0001 ***
Execution	Execution number in session	0.0052	1	1	= .22 (n.s.)
Execution	Was code just edited?	-0.0028	1	15	$<$ .0001 ***
Execution	# of function calls	0.0060	1	0	= .05 *
Execution	# of exceptions	0.063	1	598	$<$ .0001 ***
Session	Total forward-steps	-0.017	1	13	= .0004 ***
Session	Total edit-distance	0.015	1	16	$<$ .0001 ***
Session	# of executions	-0.030	1	23	$<$ .0001 ***
Session	Was code ever edited?	0.067	1	0	= .66 (n.s)
Session	Did any code in session match code from another user?	-0.042	1	22	$<$ .0001 ***

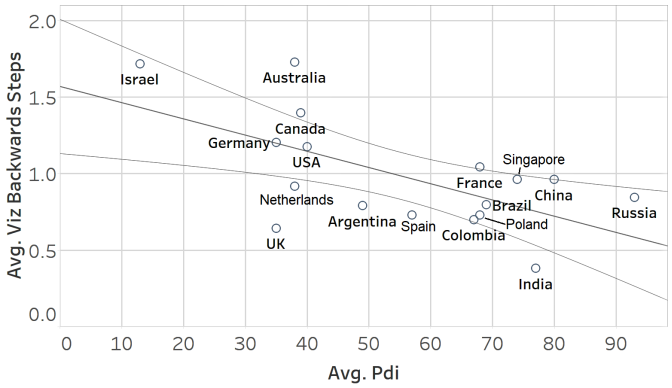

Fig. 3. Average *back-steps* per visualization vs. PDI, labeled by country, for countries with at least 10,000 code executions. Linear regression line with confidence bands included.

and low PDI countries. Some observations we made about potential confounds were:

- Each country had multiple sessions with only one code execution and multiple with over 5. We control for this with *# of executions* and *execution number in session*.
- Each country had both short one or two line programs, and longer 20+ line programs. We control for this with *Length of code (# chars)*.
- Each country had programs that were simple and had no functions, and ones that used functions in a complicated way, such as recursion. We control for this with *# steps available* and *# of function calls*.
- Israel, India, and Russia each had one program that implemented a class. We did not control for this because we do not expect it to make a difference.
- Each country had at least one program that matched another program from the dataset. We control for this with *Did any code in session match code from another user?*.
- The programs across all 80 of our chosen sessions did not match each other, except two for Australia who had very similar text processing code on the same tongue twister. One of these was marked as true for *Did any code in session match code from another user?*.
- Each country had some users making no edits between executions, making small edits between executions, and completely replacing the program between executions. We try to control for this with *Edit-dist. from previous execution*, *Was code just edited?*, and *total edit-distance*.

Given that the programming tasks did not appear to be tied to countries and that we controlled for many of the differences that might exist, we believe our regression analysis to be valid. Therefore our findings suggest that (1) there are significant differences between the use of Python Tutor's back-stepping feature across people from various countries, and (2) a country's Power Distance, which has been previously related to a tendency for self-directed learning, can explain some of these differences.

V. STUDY 2: INDIVIDUAL VALUES AND CODE DEBUGGING

Study 1 showed that users from countries with a low PDI (associated with more self-directed learning) were more likely to back-step in code visualizations than users from countries with a high PDI (H.1). Next we wanted to investigate the role of individually-reported values as opposed to only country-level generalities and do so in a more controlled setting. We therefore launched a second study to investigate how culture relates to back-steps (H.1, H.2), and how back-steps and personal values relate to debugging success (H.3, H.4).

A. Methods

We designed an online experiment as a debugging activity and we embedded the Python Tutor editor and visualizer into our experiment page for participants to use. We advertised our study with a banner on the main Python Tutor website. In our study, all users were given the same content and tasks. We collected demographics and values information from each participant and measured their behaviour in stepping through the debugger and modifying their code. Participants did not receive financial compensation.

1) Procedure: Participants provided consent, demographic information, values information (using the 10 question Short Schwartz Value Survey [35]) and then engaged in two time constrained (six-minute) debugging activities by attempting to fix buggy Python code: fixing a broken function to reverse an array, and extracting data from an array of dictionaries. They then were asked follow-up questions about how they used Python Tutor, how they used back-steps, how easy and useful they perceived Python Tutor to be, and how important back-steps were. After completing these questions, participants were shown a score of how many tests their code passed and they could continue working on the problems using Python Tutor if they wished. We included this additional data for testing use of back-steps (H.1 and H.2), but excluded it when testing debugging success (H.3 and H.4).

2) Analysis: We conducted mixed-model analyses of variance to test the correlation between back-steps and PDI (H.1) and back-steps and Conservation (H.2). Because 88% of the code execution visualizations had no back-steps and the rest of the data was skewed, we used zero-inflated negative binomial models [36]. Our analysis level was, as in Study 1, on code execution visualizations, with the number of back-steps as the dependent variable. We modeled participantId as a random variable and added either PDI or Conservation as an independent variable (for H.1 and H.2 respectively). We included four more independent variables: number of forward-steps, age,[5] gender[6], and reported programming experience (which could influence how they used Python Tutor).

For correlations with debugging success (H.3 and H.4) we conducted mixed-model analyses of variance using Gaussian models. We set the unit of analysis on code execution visualizations with the change in code tests passed for the next visualization as the dependent variable ($\Delta tests\ passed$)[7]. For independent variables, we used the previous run's passed code tests, the number of forward-steps and the number of back-steps as well as programming experience, which we expected to affect the changes in tests passed. To test for differentiated effects of values aligned with self-learning, we added Conservation along with the interaction between Conservation and back-steps.

[5]Age has previously been shown to influence non-linear navigation [2]

[6] Gender has previously been shown to influence non-linear navigation [2], tinkering [37], [38], and using new features [39]

[7]We also did this with the unit of analysis on problem numbers with the dependent variable as final tests passed, with comparable results.

TABLE II
RESULTS OF *back-step* (H.2) REGRESSION FOR STUDY 2. WE FOUND THAT CONSERVATION WAS MARGINALLY NEGATIVELY CORRELATED WITH THE NUMBER OF BACK-STEPS ($\beta = -0.11$, $p < .089$).

variable	coeff.	z	p
Conservation	-0.11	-1.7	.089 .
# of forward-steps	0.077	12.34	$\leq$ .0001 ***
age	-0.011	-1.7	.089 .
gender-Female	-0.36	-1.7	.084 .
gender-Other	0.41	-0.61	.50
Prog. Experience (Linear)	-0.23	-1.2	.23
Prog. Experience (Quadratic)	-0.38	-2.1	.039 *
Prog. Experience (Cubic)	-0.091	0.53	.60
Prog. Experience (^{4})	-0.26	-1.55	.12

3) Participants: We ran the study between July and September 2017. During this time, 857 participants completed the demographics and values survey, 522 finished the first problem, and 348 finished the entire activity, providing us with 2,697 visualization sessions. Of those sessions, 2,003 had forward-steps (458 users), and 504 had back-steps (276 users). The average age of users was 27 years ($SD = 12$ years) and 17% identified as female. Users were fairly evenly distributed across five levels of self-reported programming background (Little or none, $\leq$ 3 months, $\leq$ 6 months, $\leq$ 1 year, more). The average Conservation score was -0.71 ($SD = 1.2$) and the country averages ranged from -2.1 to 2.5, representing large differences along the Conservation vs. Openness-to-change dimension. Most participants were from the US (17%), India (17%), China (8.2%), and Russia (4.9%). The average PDI was 61 ($SD = 20.5$), roughly half the highest PDI of 120.

B. Results

For H.2, Conservation was marginally negatively correlated with the number of back-steps in a code execution visualization ($\beta = -0.11$, $p < .089$, see Table II). We also tested the relation between PDI and back-steps (H.1) using a similar model, but with PDI in the place of Conservation. PDI and the number of back-steps were not significantly correlated ($\beta = 0.0034$, $p < .39$).

PDI and Conservation were weakly correlated ($r_{(735)} = .17$, $p < .0001$), though the correlation was much smaller than we expected. This may be due to high variance of participants' values within countries, since when we averaged Conservation values by country, the correlation was higher ($r_{(55)} = .31$, $p < .02$).

While PDI did not explain the differences in the number of back-steps between countries, we did find significant differences between countries, such as between Canada ($M = 1.1$, $SD = 4.0$) and Japan ($M = 0.31$, $SD = 1.1$); $t_{(185)} = 2.02$, $p < .044$.

Our analysis of debugging progress (Table III) showed that back-steps correlate negatively with $\Delta tests\ passed$ ($F_{(1,1692)} = 5.8$, $p = .016$), the opposite of what we predicted in H.3 . We additionally found a significant negative coefficient for the interaction of back-step and Conservation

TABLE III
RESULTS OF $\Delta tests\ passed$ (H.3, H.4) REGRESSION FOR STUDY 2.

variable	coeff.	df	F	p
Previous Run Tests Passed	-0.19	1	33	$\leq$ .0001 ***
# of forward-steps	0.0059	1	8.5	.0036 **
# of back-steps	-0.023	1	5.8	.016 *
Conservation	-0.014	1	0.30	.59
Programming Experience	N/A	4	13	$\leq$ .0001 ***
# of back-steps : Conservation	-0.012	1	4.3	.038 *

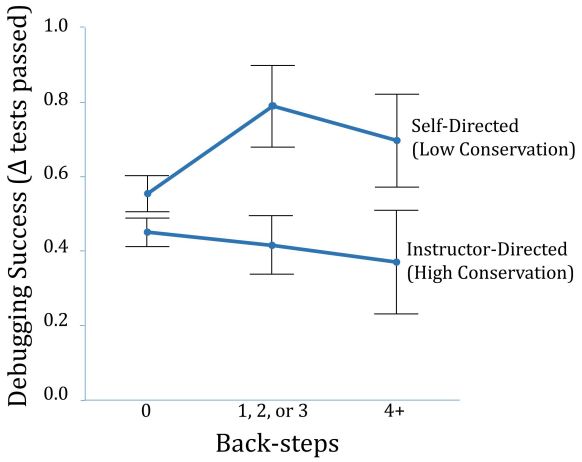

Fig. 4. Average $\Delta tests\ passed$ by number of *back-steps* and $Conservation$ (mean split) for code executions with at least one forward-step. Bars indicate standard error. Compare to Figure 2 showing H.4.

($F_{(1,1701)} = 4.3$, $p = .038$), confirming H.4 when the negative result of H.3 is accounted for (see Figure 4).

In free-response answers, participants mentioned taking back-steps to check their understanding of code, find the source of bugs, view a set of steps again, and go back to a step they had accidentally skipped over. Some who did not take back-steps mentioned running out of time, finding the problem too easy to require back-steps, or finding forward-steps sufficient.

VI. DISCUSSION

A. Power Distance Negatively Correlates with Back-Steps

Our results demonstrate that the use of non-linear navigation within an online visual debugger varies with national culture: the more a country's people tend to value self-directed learning, the more back-steps they will take. Study 1 confirmed that Power Distance (PDI) negatively correlated with back-steps (H.1), meaning that users in countries with a higher PDI were less self-directed in their use of Python Tutor. This is consistent with the results of a previous MOOC study [2], where users from more student-centered countries were more likely to navigate with non-linear "backjumps". It is also consistent with prior literature on culture and PDI, supporting the claim that PDI is related to self-directed learning [8].

Study 2 revealed no correlation between PDI and back-steps, but showed that Conservation and back-steps are marginally negatively correlated (H.2). This suggests that a user's Conservation score has a larger effect than national culture on the use of back-stepping.

TABLE IV
RESULTS FOR EACH HYPOTHESIS.

	Hypothesis	Outcome
H.1	PDI of a user's country will negatively correlate with the number of back-steps that user takes.	Supported by Study 1. Not supported by Study 2.
H.2	Conservation will negatively correlate with back-steps.	Marginally supported by Study 2.
H.3	Back-steps in code visualizations will correlate with debugging success.	Study 2 found negative correlation instead.
H.4	Conservation will interact with the correlation between back-steps and debugging success.	Supported by Study 2.

We also found several significant differences between countries in the use of back-stepping. Because the cultural measures PDI and Conservation could not fully explain these differences, it is likely many differences are due to factors other than self-directedness, such national and individual differences in background, experience, socioeconomic status and math competency, and reason for using Python Tutor. PDI and Conservation also may not be sufficient measures of self-directedness to account for the variation we saw.

B. Back-Steps Correlate With Less Debugging Success (Depending on Personal Values)

We were surprised to see that back-steps were negatively correlated with debugging success, the opposite of our hypothesis (H.3). The finding raises doubts about whether back-stepping is helpful in debugging (though there may be other learning benefits to back-stepping that we did not capture). It is possible that instead of measuring back-steps being used in a helpful, intentional way, the back-steps we measured were instead a symptom of struggle [40]. For example, back-steps may have been used in a haphazard way by someone having trouble [41], [42], or as a way of verifying a result they did not believe at first [41].

Finally, we confirmed our last hypothesis (H.4), though we have to modify the phrasing given the result of H.3: For higher Conservation learners, the negative correlation between back-steps and debugging success was stronger, and for lower Conservation learners the negative correlation was weaker. That is, for instructor-directed learners, many back-steps meant less debugging success, while self-directed learners saw less of a relationship between back-steps and debugging success (Figure 4). These results demonstrate that the benefits of some debugging features may vary with personal values for self-directed or instructor-directed learning.

C. Relations to Research on Tinkering and Gender

Our study shares a number of parallels with a previous study on tinkering and gender [37]. Our study supports this prior work in finding a marginally significant trend of females taking

fewer back-steps in Study 2. We then extend that work by showing similar effects to what they found, but with different groups (countries in ours, gender in theirs) varying along different measures of independence (self-directed learning in ours, self-efficacy in theirs) and varying in feature use (back-steps in ours, tinkering in theirs).

The prior work raises further questions about ours, such as: *To what extent are back-steps used in a way that can be considered tinkering? How does membership in different groups interact (e.g., females in India)?* The prior work showed different benefits for exploratory tinkering vs. repeated tinkering, so: *Are there similar different uses of back-steps?*

D. Design Implications

Our results suggest several implications for when back-steps (and other non-linear navigation features) may need to be emphasized, de-emphasized or scaffolded for instructor-directed learners (i.e., those who prefer instructors to direct their learning). Back-steps (and potentially other non-linear navigation) are either detrimental to users or a symptom of struggle. Whatever the cause, this relationship was especially strong for instructor-directed learners. If backward navigation is in fact detrimental to instructor-directed learners, designers may want to de-emphasize or hide backward navigation for those users. Alternatively, if backward navigation is a symptom of struggle for instructor-directed learners, designers may want to provide support and intervention when they detect those learners navigating backwards.

Designers who want to help users across cultures make effective and efficient use of back-steps (or other non-linear navigation features) may need to make these features more prominent. They may also want to provide tutorials or wizards to give instructor-directed users a forward path for learning to navigate backwards (in line with previous suggestions for high PDI countries [21]). Additionally back-steps could be augmented with additional information, such as suggested relevant backward slices of steps for user-selected variables or output (following Whyline [31]), or higher-level holistic views showing context at a glance, providing an alternative way of learning that doesn't involve taking back-steps (follwing Omnicode [43]).

As evident from the above, programming education tools are unlikely to optimize learning if they are developed in a one-size-fits-all manner. Instead, our results show that people from different countries make different use of key features, suggesting that programming education tools should adapt to preferences and behaviors to optimally support the learner. We showed that PDI and Conservation can be useful as proxy measures for self-directed learning to guide such adaptations, even though they only partially explain the variance between countries. PDI is particularly convenient for designers because it can be derived based on a user's IP address, needing no extra input from learners. Still, designers should be aware that the trends we found for PDI are averaged across large samples, and individual variations may make appropriate adaptation challenging. Prior efforts have worked around this issue by bootstrapping an initial adaptation with the help of PDI (and other dimensions) before extracting individual information about a user's behavior and preferences from behavioral data [21]. Such an adaptive system circumvents the problem of data sparseness, preventing initial shortcomings for most people, but still updating its priors over time.

VII. Limitations and Future Work

Our study compared use of a single debugger interface feature with one national cultural measure and one individual value measure. Testing only one feature of one code visualization tool limits our ability to generalize to other interfaces. In the future, we plan to further investigate other features of programming education environments, such as information density, cooperative programming, or prominent achievement scores, to evaluate possible effects of country and culture.

In our two studies, we did not directly measure self-directed learning but used proxy measures which may not adequately capture this concept. This is particularly the case with the national measure of PDI, since generalizing by country collapses many meaningful variations between groups of people and individuals. While prior work has repeatedly suggested a link between self-directed learning, PDI, and Conservation, more research is needed to investigate whether these cultural dimensions indeed predict different levels of self-directed learning. This was further complicated by potential bias in our sample from each country. In particular, the subset of visitors to Python Tutor who chose to participate in Study 2 might have different demographics and levels of self-directed learning than those who did not chose to participate.

Future work should also further investigate the benefits, detriments and uses of non-linear navigation, such as back-stepping, especially since our results contradicted our hypothesis that back-steps use would correlate with debugging success. We especially hope to see more work evaluating alternative functionalities that enable users to better learn programming, and evaluate whether such functionalities have a differential effect on debugging success depending on a user's culture.

VIII. Conclusion

Our findings show that visual debuggers are used differently by different groups and do not equally benefit all learners. Importantly, we found that these differences can be measured and predicted. We hope that our work will inspire designers and developers to create programming education tools that adapt to their user's cultural backgrounds.

IX. Data Set

To enable replication and extension of our work, all of the code and data sets from both studies are on GitHub: https://github.com/kylethayer/culture-debugging-study-data

Acknowledgment

Special thanks to Jacob O. Wobbrock, Nigini A. Oliveira, Daniel Epstein, Amanda Swearngin, Eunice Jun, William Tressel, Kurtis Heimerl, and Rahul Banerjee.

REFERENCES

[1] "Code.Org: About Us," 2018. [Online]. Available: https://code.org/about

[2] P. J. Guo and K. Reinecke, "Demographic differences in how students navigate through MOOCs," in *Proceedings of the first ACM conference on Learning@ scale conference*. ACM, 2014, pp. 21–30. [Online]. Available: http://dl.acm.org/citation.cfm?id=2566247

[3] J. D. Hansen and J. Reich, "Democratizing education? Examining access and usage patterns in massive open online courses," *Science*, vol. 350, no. 6265, pp. 1245–1248, Dec. 2015, 00000. [Online]. Available: http://www.sciencemag.org/content/350/6265/1245

[4] C. Sturm, A. Oh, S. Linxen, J. Abdelnour Nocera, S. Dray, and K. Reinecke, "How WEIRD is HCI?: Extending HCI Principles to other Countries and Cultures," in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2015, pp. 2425–2428, 00000. [Online]. Available: http://dl.acm.org/citation.cfm?id=2702656

[5] M. Guzdial, "Limitations of moocs for computing education- addressing our needs: Moocs and technology to advance learning and learning research (ubiquity symposium)," *Ubiquity*, vol. 2014, no. July, pp. 1:1–1:9, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2591683

[6] P. J. Guo, "Non-native english speakers learning computer programming: Barriers, desires, and design opportunities," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18, 2018.

[7] R. F. Kizilcec, A. J. Saltarelli, J. Reich, and G. L. Cohen, "Closing global achievement gaps in MOOCs," *Science*, vol. 355, no. 6322, pp. 251–252, Jan. 2017. [Online]. Available: http://science.sciencemag.org/content/355/6322/251

[8] G. Hofstede, "Cultural differences in teaching and learning," *International Journal of Intercultural Relations*, vol. 10, no. 3, pp. 301–320, Jan. 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0147176786900155

[9] R. F. Kizilcec and G. L. Cohen, "Eight-minute self-regulation intervention raises educational attainment at scale in individualist but not collectivist cultures," *Proceedings of the National Academy of Sciences*, p. 201611898, Apr. 2017. [Online]. Available: http://www.pnas.org/content/early/2017/04/07/1611898114

[10] G. Hofstede, *Culture's Consequences: International Differences in Work-Related Values*. SAGE, Jan. 1984.

[11] A. Marcus and E. W. Gould, "Crosscurrents: cultural dimensions and global Web user-interface design," *interactions*, vol. 7, no. 4, pp. 32–46, 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=345238

[12] P. J. Guo, "Online Python Tutor: Embeddable web-based program visualization for CS education," in *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13. New York, NY, USA: ACM, 2013, pp. 579–584. [Online]. Available: http://doi.acm.org/10.1145/2445196.2445368

[13] J. Sorva, *Visual program simulation in introductory programming education*. Aalto University, 2012. [Online]. Available: https://aaltodoc.aalto.fi:443/handle/123456789/3534

[14] S. H. Schwartz, "Universals in the content and structure of values: Theoretical advances and empirical tests in 20 countries," *Advances in experimental social psychology*, vol. 25, pp. 1–65, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0065260108602816

[15] P. Guo, "Visualize python, java, javascript, typescript, ruby, c, and c," 2018. [Online]. Available: http://pythontutor.com/

[16] G. Hofstede, "Cultures and organizations: Software of the mind, intercultural co-operation and its implications for survival," 1997.

[17] E. Callahan, "Cultural similarities and differences in the design of university web sites," *Journal of Computer-Mediated Communication*, vol. 11, no. 1, pp. 239–273, 2005.

[18] B. McSweeney, "Hofstedes model of national cultural differences and their consequences: A triumph of faith-a failure of analysis," *Human relations*, vol. 55, no. 1, pp. 89–118, 2002.

[19] "Pupil-teacher ratio in primary education (headcount basis) | Data," 2017. [Online]. Available: https://data.worldbank.org/indicator/SE.PRM.ENRL.TC.ZS

[20] "GDP per capita (current US$) | Data," 2017. [Online]. Available: https://data.worldbank.org/indicator/NY.GDP.PCAP.CD

[21] K. Reinecke and A. Bernstein, "Knowing what a user likes: A design science approach to interfaces that automatically adapt to culture," *Mis Quarterly*, vol. 37, no. 2, pp. 427–453, 2013. [Online]. Available: http://www.misq.org/skin/frontend/default/misq/pdf/V37I2/ReineckeBernstein.pdf

[22] A. Bardi and S. H. Schwartz, "Values and Behavior: Strength and Structure of Relations," *Personality and Social Psychology Bulletin*, vol. 29, no. 10, pp. 1207–1220, Oct. 2003. [Online]. Available: http://psp.sagepub.com/content/29/10/1207

[23] N. T. Feather, "Values, valences, and choice: The influences of values on the perceived attractiveness and choice of alternatives," *Journal of Personality and Social Psychology*, vol. 68, no. 6, pp. 1135–1151, Jun. 1995. [Online]. Available: http://offcampus.lib.washington.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=pdh&AN=1995-32996-001&site=ehost-live

[24] S. Schwartz, "Value priorities and behavior: Applying a Theory of Integrated Value Systems," in *The psychology of values: The Ontario symposium*, vol. 8, 2013. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=DACsdMk7qqoC&oi=fnd&pg=PA1&dq=schwartz+values+behavior&ots=u3nzArGxx6&sig=oe4XQPtH6yS2Rea6qkcYRGLs5h8

[25] C. Hofer, M. Denker, and S. Ducasse, "Design and implementation of a backward-in-time debugger," in *NODe 2006*. GI, 2006, pp. 17–32. [Online]. Available: https://hal.archives-ouvertes.fr/inria-00555768/

[26] B. Lewis, "Debugging Backwards in Time," *arXiv:cs/0310016*, Oct. 2003, arXiv: cs/0310016. [Online]. Available: http://arxiv.org/abs/cs/0310016

[27] G. Pothier and E. Tanter, "Back to the future: Omniscient debugging," *IEEE software*, vol. 26, no. 6, 2009. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/5287015/

[28] Z. Azar, "PECCit: An Omniscient Debugger for Web Development," *Electronic Theses and Dissertations*, Jan. 2016. [Online]. Available: http://digitalcommons.du.edu/etd/1099

[29] S. P. Booth and S. B. Jones, "Walk backwards to happiness: debugging by time travel," in *Proceedings of the 3rd International Workshop on Automatic Debugging; 1997 (AADEBUG-97)*. Linköping University Electronic Press, 1997, pp. 171–184. [Online]. Available: http://www.ep.liu.se/ecp/article.asp?issue=001&article=014

[30] J. Engblom, "A review of reverse debugging," in *System, Software, SoC and Silicon Debug Conference (S4D), 2012*. IEEE, 2012, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/6338149/

[31] A. J. Ko and B. A. Myers, "Finding Causes of Program Output with the Java Whyline," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 1569–1578. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518942

[32] "GeoLite2 Free Downloadable Databases ¡¡ Maxmind Developer Site," 2015, 00000. [Online]. Available: http://dev.maxmind.com/geoip/geoip2/geolite2/

[33] "Geert Hofstede | Hofstede Dimension Data Matrix," 2015, 00002. [Online]. Available: http://www.geerthofstede.nl/dimension-data-matrix

[34] T. Lumley, P. Diehr, S. Emerson, and a. L. Chen, "The Importance of the Normality Assumption in Large Public Health Data Sets," *Annual Review of Public Health*, vol. 23, no. 1, pp. 151–169, 2002. [Online]. Available: http://dx.doi.org/10.1146/annurev.publhealth.23.100901.140546

[35] M. Lindeman and M. Verkasalo, "Measuring Values With the Short Schwartz's Value Survey," *Journal of Personality Assessment*, vol. 85, no. 2, pp. 170–178, Oct. 2005. [Online]. Available: http://dx.doi.org/10.1207/s15327752jpa8502_09

[36] J. S. Preisser, J. W. Stamm, D. L. Long, and M. E. Kincade, "Review and Recommendations for Zero-inflated Count Regression Modeling of Dental Caries Indices in Epidemiological Studies," *Caries research*, vol. 46, no. 4, pp. 413–423, 2012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3424072/

[37] L. Beckwith, C. Kissinger, M. Burnett, S. Wiedenbeck, J. Lawrance, A. Blackwell, and C. Cook, "Tinkering and gender in end-user programmers' debugging," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 2006, pp. 231–240. [Online]. Available: http://dl.acm.org/citation.cfm?id=1124808

[38] M. G. Jones, L. Brader-Araje, L. W. Carboni, G. Carter, M. J. Rua, E. Banilower, and H. Hatch, "Tool time: Gender and students' use of tools, control, and authority," *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching*, vol. 37, no. 8, pp. 760–783, 2000.

[39] L. Beckwith, M. Burnett, S. Wiedenbeck, C. Cook, S. Sorte, and M. Hastings, "Effectiveness of end-user debugging software features: Are there gender issues?" in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2005, pp. 869–878. [Online]. Available: http://dl.acm.org/citation.cfm?id=1055094

[40] E. S. Tabanao, M. M. T. Rodrigo, and M. C. Jadud, "Predicting At-risk Novice Java Programmers Through the Analysis of Online Protocols," in *Proceedings of the Seventh International Workshop on Computing Education Research*, ser. ICER '11. New York, NY, USA: ACM, 2011, pp. 85–92. [Online]. Available: http://doi.acm.org/10.1145/2016911.2016930

[41] M. C. Jadud, "A First Look at Novice Compilation Behaviour Using BlueJ," *Computer Science Education*, vol. 15, no. 1, pp. 25–40, Mar. 2005. [Online]. Available: https://doi.org/10.1080/08993400500056530

[42] D. N. Perkins, C. Hancock, R. Hobbs, F. Martin, and R. Simmons, "Conditions of Learning in Novice Programmers," *Journal of Educational Computing Research*, vol. 2, no. 1, pp. 37–55, Feb. 1986. [Online]. Available: http://journals.sagepub.com/doi/10.2190/GUJT-JCBJ-Q6QU-Q9PL

[43] H. Kang and P. J. Guo, "Omnicode: A Novice-Oriented Live Programming Environment with Always-On Run-Time Value Visualizations," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, 2017, pp. 737–745.