End-User Programmers in Trouble: Can the Idea Garden help them to help themselves?

Jill Cao[1], Irwin Kwan[1], Faezeh Bahmani[1], Margaret Burnett[1],
Scott D. Fleming[2], Josh Jordahl[1], Amber Horvath[1], Sherry Yang[1,3]

[1]Electrical Engineering & Computer Science
Oregon State University
Corvallis, OR, USA

[2]Computer Science
University of Memphis
Memphis, TN, USA

[3]Computer Systems Engineering Technology
Oregon Institute of Technology
Klamath Falls, OR, USA

{caoch, kwan, bahmani, burnett}@eecs.oregonstate.edu; scott.fleming@memphis.edu; sherry.yang@oit.edu

***Abstract*—End-user programmers often get stuck because they do not know how to overcome their barriers. We have previously presented an approach called the Idea Garden, which makes minimalist, on-demand problem-solving support available to end-user programmers in trouble. Its goal is to encourage end users to *help themselves* learn how to overcome programming difficulties as they encounter them. In this paper, we investigate whether the Idea Garden approach helps end-user programmers problem-solve their programs on their own. We ran a statistical experiment with 123 end-user programmers. The experiment's results showed that, even when the Idea Garden was no longer available, participants with little knowledge of programming who previously used the Idea Garden were able to produce higher-quality programs than those who had not used the Idea Garden.**

Keywords—Idea Garden; end-user programming; problem solving; barriers; mashups; quantitative empirical evaluation

I. INTRODUCTION

When doing a programming task, end users face many barriers such as decomposing design problems [4], using loops [5], and choosing and coordinating multiple modules [16]. To help users overcome such barriers on their own without the need for guided instruction, we have previously presented the Idea Garden approach [5], an add-on for end-user programming environments to help end-user programmers in trouble solve their own problems. The Idea Garden draws from Simon's problem-solving theory [21] and Minimalist Learning Theory [7], and delivers its help in the form of information snippets that, on demand, deliver problem-solving strategies and programming domain knowledge in the context of a user's own programming tasks. The core philosophy of the Idea Garden is not to automatically remove barriers for the user, but to rather enable the user to solve problems on their own with only minimal, self-guided assistance.

We previously performed an empirical study on Idea Garden' ability to help end-user programmers learn problem-solving strategies and programming knowledge during a programming task in which learning was not the primary goal [6]. This previous study revealed that after actively using the Idea Garden, users were able to demonstrate having learned the relevant problem-solving strategies and programming knowledge, as evidenced by their ability to explain the relevant problem-solving strategies and programming knowledge.

In this paper, we move beyond learning to doing. We investigate whether end-user programmers who have used the Idea Garden can put their learning into practice in future programming tasks even when Idea Garden support is no longer available, via the following research questions:

RQ 1: Does the Idea Garden help end-user programmers learn enough to do a programming task on their own without support?

RQ 2: Are there particular factors that help to determine end-user programmers' future success after using the Idea Garden?

II. BACKGROUND

A. The Idea Garden's Host: CoScripter

We implemented the Idea Garden prototype within CoScripter/Vegemite [18], an end-user programming-by-demonstration environment for web automation in Firefox. Using CoScripter, a user can demonstrate how to carry out a task by navigating to web pages, entering data in forms, and interacting with page elements. CoScripter translates the user's actions into a "web macro" script that the user can edit and execute (Fig. 1a). CoScripter also provides a table (Fig. 1b) that makes it possible to create mashups that combine data from multiple web pages. For example, a user can create a script to mash restaurant location with public transit by loading a web page of restaurants (Fig. 1c), copying its addresses to the table (Fig. 1b), then iterating to send each address to another web page (e.g., Google Maps) to compute travel time via transit. Thus, CoScripter requires understanding of programming concepts such as control flow and dataflow.

B. Helping Users Learn to Do with the Idea Garden

The goal of the Idea Garden is to help users form ideas to overcome programming barriers on their own. As we have described in previous work ([5, 6]), we leveraged Simon's problem-solving theory [21] to guide the design of the Idea Garden features. According to Simon's theory, two types of skills are necessary for solving problems in a specific domain: domain-specific knowledge and general problem-solving strategies [21]. The Idea Garden features encourage both of these skills: they aim to encourage users to adopt new strategies and pick up new programming knowledge that is relevant to the problems they are currently trying to solve.

The Idea Garden features are designed to help users overcome barriers, such as those identified by Ko et al. [16] and in our prior work ([3, 4, 5]), by providing programming knowledge as well as strategies (Table I). The Idea Garden prototype's features target three programming barriers: "How-to-start", where users had problems figuring out how to start their scripts; "Composition", where users had problems combining multiple web page functions to come up with a result; and "More-than-once", where users were unable to use iteration to repeat actions. The features that we developed were the *Getting started* feature, which addresses the "How-to-start" barrier by providing suggestions on what initial actions to take; the *Second web page* feature, which addresses the "Composition" barrier by suggesting that users can use the output from one page as input to a second page; and the *Generalize-with-repeat* feature, which addresses the "More-than-once" barrier by suggesting a process and commands that the user can use to repeat actions. Table I summarizes the relationships among the features, barriers, strategies, and programming knowledge. The associated strategies are described in Table II and the associated programming knowledge is described in Table III.

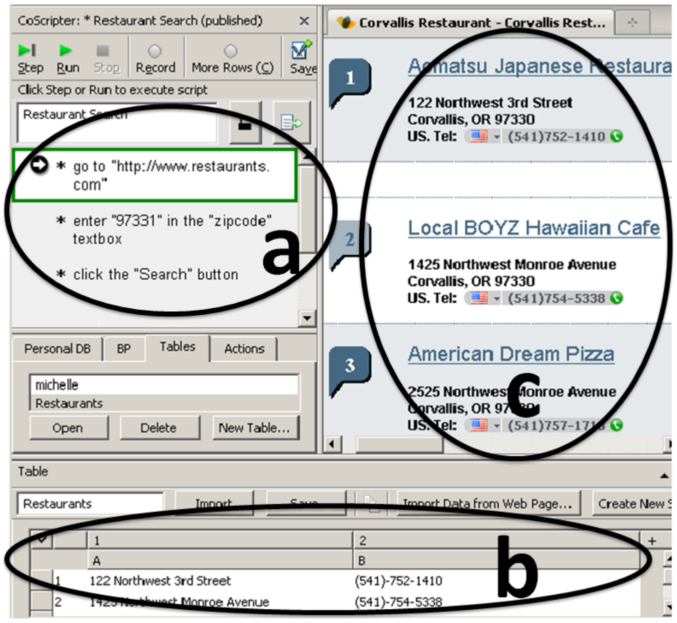
Each feature has two versions, a context-sensitive version and a context-free version. The context-sensitive versions are available when the Idea Garden detects specific user action sequences suggesting barriers that the Idea Garden targets. The context-free versions are always accessible from a "Help" button at the top of the screen.

III. Experiment

A. Experiment Design

To answer our research questions, we conducted a between-subjects experiment with four treatments: one Control condition and three Idea Garden conditions: Strategy, Programming, and Combined. We asked non-Control participants to first work on the learning task, in which participants completed a programming task in CoScripter with the Idea Garden present. Then, we asked them to perform a learning transfer task [2] in which participants completed a programming task in CoScripter with the Idea Garden not present. Control participants did not have access to the Idea Garden during either of their tasks.

Each Idea Garden treatment contains features that address the same barriers (Table I), but each treatment's features address the barriers differently. The *Strategy* treatment provides suggestions to apply a problem-solving strategy; the *Programming* treatment provides programming knowledge and



* go to "http://www.restaurants.com"
* enter "97331" in the "zipcode" textbox
* click the "Search" button

	1	2
	A	B
1	122 Northwest 3rd Street	(541)-752-1410
2	1425 Northwest Monroe Avenue	(541)-754-5338

1 Aomatsu Japanese Restaura
122 Northwest 3rd Street
Corvallis, OR 97330
US. Tel: (541)752-1410

2 Local BOYZ Hawaiian Cafe
1425 Northwest Monroe Avenue
Corvallis, OR 97330
US. Tel: (541)754-5338

3 American Dream Pizza
2525 Northwest Monroe Avenue
Corvallis, OR 97330
US. Tel: (541)757-1713

Fig. 1. CoScripter's (a) script area, (b) table area, and (c) browsing area.

TABLE I. EACH FEATURE ADDRESSES A BARRIER WITH STRATEGIES AND PROGRAMMING KNOWLEDGE.

Feature	**Barrier addressed**	**Strategy**	**Programming knowledge**
Getting Started	How-to-start	Working backwards	Data extraction concept, Finder design pattern
Second web page	Composition	Divide-and-conquer (context-sensitive), Working backward (context-free)	Dataflow concept, Webpage-as-component design pattern
Generalize-with-repeat	More-than-once	Generalization	Iteration concept, Repeat-copy-paste design pattern

TABLE II. PROBLEM-SOLVING STRATEGIES IN IDEA GARDEN FEATURES.

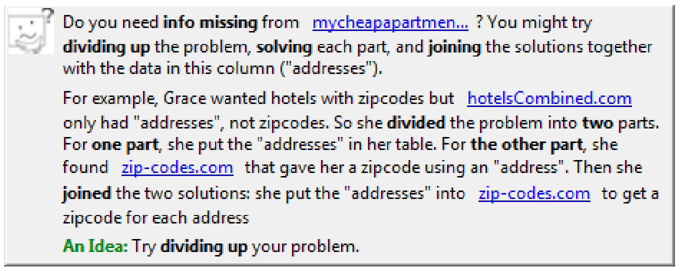
Strategies: information that helps users problem-solve	
Working backward	Identify the end goal, then figure out the last step to the goal, second to the last step, and so on until the givens are reached.
Divide-and-conquer	Break a problem into individual pieces, solve each piece, and join the individual solutions together.
Generalization	Solve one instance of a problem and generalize the solution to all instances in the problem.

TABLE III. PROGRAMMING KNOWLEDGE INCLUDES PROGRAMMING CONCEPTS AND MINI DESIGN PATTERNS.

Programming concepts: information that helps users build scripts	
Data extraction	The concept of selecting a slice of structured data from a web page and putting it into the table.
Dataflow	The concept of flowing data between web page and table, or between web pages.
Iteration	The concept of looping through rows of table to operate on each row.
Mini Design patterns: common ways that users structure their scripts	
Finder	Use a web page to find information (as opposed to computing information)
Webpage-as-component	Use a web page to compute information (as opposed to finding information).
Repeat-copy-paste	For each row in the scratchtable, copy-paste value from table to web page and submit.

the *Combined* treatment contains both strategy and programming knowledge. For example, to address the Composition barrier, the context-sensitive Second Webpage feature from the Strategy treatment contained the divide-and-conquer strategy (Fig. 2) whereas the *Programming* treatment contained the webpage-as-component design pattern and the dataflow concept (Fig. 3). The Second Webpage feature for the *Combined* treatment included both strategy information and programming knowledge (Fig. 4).

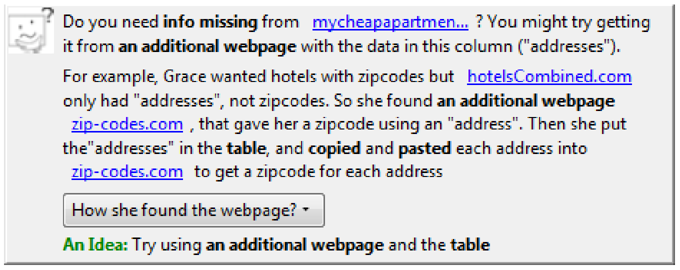
Although Simon emphasized the importance of both domain knowledge—programming knowledge in our context—and problem-solving strategies, including both parts as in our Combined treatment has trade-offs. One trade-off is length versus effectiveness. As suggested by the Attention Investment model [1], the probability that a user would invest attention in a feature depends on the perceived cost of the investment. If a feature is too long, the user may perceive the cost of processing it as being too high and ignore it. In addition, too much information might lead to cognitive overload [22] which reduces the quality of information a user is able to get out of a feature. Thus, including both strategy and programming information as in the Combined treatment may potentially be less effective than just including one piece as in the Strategy and the Programming treatments.


Do you need **info missing** from mycheapapartmen... ? You might try **dividing up** the problem, **solving** each part, and **joining** the solutions together with the data in this column ("addresses").

For example, Grace wanted hotels with zipcodes but hotelsCombined.com only had "addresses", not zipcodes. So she **divided** the problem into **two** parts. For **one part**, she put the "addresses" in her table. For **the other part**, she found zip-codes.com that gave her a zipcode using an "address". Then she **joined** the two solutions: she put the "addresses" into zip-codes.com to get a zipcode for each address

An Idea: Try **dividing up** your problem.

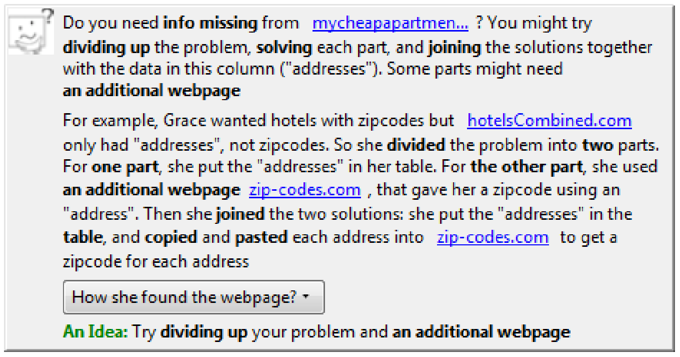
Fig. 2. The Strategy treatment's Second web page feature (context-sensitive). This feature describes the "divide and conquer" strategy.


Do you need **info missing** from mycheapapartmen... ? You might try getting it from **an additional webpage** with the data in this column ("addresses").

For example, Grace wanted hotels with zipcodes but hotelsCombined.com only had "addresses", not zipcodes. So she found **an additional webpage** zip-codes.com, that gave her a zipcode using an "address". Then she put the"addresses" in the **table**, and **copied** and **pasted** each address into zip-codes.com to get a zipcode for each address

An Idea: Try using **an additional webpage** and the **table**

Fig. 3. The Programming treatment's Second web page feature (context-sensitive). This feature presents the "dataflow" concept and the "webpage-as-component" pattern.


Do you need **info missing** from mycheapapartmen... ? You might try **dividing up** the problem, **solving** each part, and **joining** the solutions together with the data in this column ("addresses"). Some parts might need **an additional webpage**

For example, Grace wanted hotels with zipcodes but hotelsCombined.com only had "addresses", not zipcodes. So she **divided** the problem into **two** parts. For **one part**, she put the "addresses" in her table. For **the other part**, she used **an additional webpage** zip-codes.com, that gave her a zipcode using an "address". Then she **joined** the two solutions: she put the "addresses" in the **table**, and **copied** and **pasted** each address into zip-codes.com to get a zipcode for each address

An Idea: Try **dividing up** your problem and **an additional webpage**

Fig. 4. The Combined treatment's Second web page feature (context-sensitive). This feature describes both the "divide and conquer" strategy as well as the "dataflow" concept and the "webpage-as-component" pattern.

For our study, we hypothesize that, even when the Idea Garden is no longer available, participants who previously had access to the Idea Garden, regardless of treatment, will be able to write a higher-quality program for a programming task compared to Control participants who had no previous access to the Idea Garden.

B. *Participants*

We recruited undergraduate and graduate students at Oregon State University from 53 majors (e.g., English, biology, chemical engineering, human development and family studies), but excluding computer science and electrical engineering. We also disqualified any participants who had taken programming courses beyond an introductory level required for many majors' computer literacy requirements as well as anyone who had used two or more mainstream general programming languages (such as C/C++, Python, or PHP). We recruited 127 participants who met these criteria but due to data collection issues involving four participants, we were left with usable data for 123 participants.

C. *Procedure*

We assigned two tasks to each participant. Idea Garden participants (those in Strategy, Programming, and Combined) had access to the Idea Garden during the first task whereas Control participants did not have access to the Idea Garden during the first task. In the second task, no participants had access to the Idea Garden. Thus, the first task was a learning task and the second task was a learning transfer task. Idea Garden participants were not informed that the Idea Garden would be unavailable during the second task.

Participants filled out a background questionnaire and then took a 25-minute, hands-on tutorial about CoScripter functionality. The tutorial walked participants through how to create three scripts: one to look up information from a webpage, one to pull data from a webpage into the table, and one to push data from the table to a webpage. Following the tutorial, participants had 6 minutes to practice. We encouraged the participants to ask questions during this practice period. Participants then filled out a standard computer self-efficacy questionnaire [8] regarding CoScripter-related tasks.

Participants then had 25 minutes to work on the first task. Participants in the Idea Garden treatment had the Idea Garden enabled. To ensure that every Idea Garden participant was aware of the Idea Garden features, we interrupted the participants twelve minutes into the task to draw their attention to the context-free features. Scripts and tables were automatically saved every 15 seconds or whenever the user pressed the "save" button.

After the first task, Idea Garden participants filled out an opinion questionnaire regarding the context-sensitive and context-free versions of the three features. The questions displayed a picture of the feature and asked, "This feature helped me accomplish my task". A participant could respond using a five-point Likert scale or could indicate "Never saw". Participants could also leave comments about the features. To be consistent across all conditions, Control participants were asked to fill out a questionnaire containing questions that did not relate to our study.

Participants were given 30 minutes to work on the second task, during which the Idea Garden was not available. After the second task, participants filled out a post-task self-efficacy questionnaire. Every participant was provided the opportunity to leave feedback about each task directly on the task sheet.

D. *Tasks*

Each participant worked on two tasks assigned in random order. The apartment task (Apt) asked a participant to create a script that searched for two bedroom apartments within ten minutes' driving time of the Ohio State University campus and were under $1,300. The Pet task asked a participant to create a script that searched for cats to adopt in the Corvallis area that were shorthair breed and from a reputable shelter. In the task descriptions, we listed the expected outputs of the scripts: a record of time from each apartment to campus in the table (for Apartment) or a record of the number of reviews for each shelter (for Pet) in the table.

The two tasks were intended to be equally difficult (although as we shall see, they were not). Each task consisted of three subtasks that required the same knowledge to accomplish: (1) using a second webpage to compute the missing information (e.g., using Yelp.com to find the number of reviews for a pet shelter listed on PetFinder.com), (2) using the `repeat` command to iterate over data (e.g., pet shelter names from PetFinder.com) in the table rows to compute the missing information, and (3) using the `copy` and the `paste` commands to pull the result of each computation (e.g., number of reviews for each shelter from Yelp.com) into the table. Both tasks had three implicit subtasks: (1) import a list of apartment addresses or shelter names from a webpage into the table; (2) iterate over the addresses and compute driving time, or iterate over the shelter names and look up shelter ratings; and (3) copy each driving time or shelter rating back to the table.

IV. ANALYSIS METHODOLOGY

A. *Task Performance*

Because we were interested in learning-to-doing, we evaluated the transfer task's performance only. Thus, whenever we mention "task performance", we mean the second (transfer) task, in which the Idea Garden was not available.

To evaluate the quality of each participant's performance in the second task, we graded the scripts and tables generated during the task. We graded three scripts: the largest auto-saved script, the most recent auto-saved script, and latest user-saved script, along with the accompanying tables. We graded all three because many users kept starting additional scripts, making it difficult for us to know which one had finally won out as the user's "intended" solution. This resulted in three scores per participant, from which we used the participant's highest score.

We graded the scripts and tables against a rubric based on the three subtasks listed in Section III.D. Each correct answer was defined precisely, so subjective interpretation was not needed to grade them. Specifically, each task's three subtasks were worth 5 points, for a total of 15 points possible. Within a subtask, each correct command or table column entry was worth 1 point. For example, the Apartment task's subtask 1 needed four commands (extract addresses, go to maps page, copy address from table, paste into maps page) and one table column (addresses), each worth 1 point. A participant with two correct commands and the correct table column would score 3 of 5 for this subtask.

Two researchers split up the scripts and the tables and graded them independently. Then, one researcher double-checked the grading. Since the rubrics did not involve subjective judgment, we did not measure inter-rater agreement.

To compare Idea Garden participants' performance to that of Control participants, we used Fisher's exact test. We calculated a grand median score for all participants in the experiment and then assigned participants into the group of "equal to or above the grand median" or "below the grand median" and ran Fisher's exact test on the counts in these groups. We did not use ANOVA because the scores did not fit a normal distribution (Kolomgorov-Smirnov D=0.7361, p<2.2e-16) nor did we use Kruskal-Wallis because of a large number of ties.

B. *Ratings of Idea Garden's Helpfulness*

To assess participant's overall opinions of the features' helpfulness, we calculated each participant's average rating of the Idea Garden features the participant saw. Using a one-sample *t*-test, we compared the resulting average rating of the Idea Garden's helpfulness against the expected mean of 3.0, which was a neutral rating. Two researchers coded whether participants reported difficulties about each task.

V. RESULTS

A. *RQ1: Does the Idea Garden help end users do programming tasks on their own?*

Evidence that the Idea Garden helped participants' task performance in the transfer task was not strong. Idea Garden participants averaged higher scores than Control participants (Table IV), but the difference was not significant at p < .05. Also, no one treatment had significantly higher scores than the others.

However, Idea Garden participants' reports of the Idea Garden's helpfulness from the post-session questionnaire were

TABLE IV. SUMMARY OF PARTICIPANTS' PERFORMANCE SCORES

Treatment	N	Mean	Median	StdDev
Control	28	5.3	3	5.1
Idea Garden (3 treatments)	95	5.9	4	4.8

significantly higher than neutral (one-sample t=3.22, p=.00176). (Neutral or below is what we might expect if the approach were not helpful. The one-sample t-test compares a sample value against an expected population mean). Table V summarizes.

Given that so many Idea Garden participants found the Idea Garden features helpful, what might this suggest? One possible explanation of our results might be that, as in our previous study [6], some participants who learned something from the Idea Garden were simply not able to transfer their learning to overcoming barriers on their own. However, another possibility, posed by RQ2, is that the Idea Garden may have been helpful to only particular participants for only particular situations. We investigate this possibility next by considering the possible factors of *who* the Idea Garden may have helped and *when* it may have helped them.

B. *RQ2: Factors affecting success with the Idea Garden: Who and When?*

Regarding *who*, it is common for empirical studies of end-user programmers to include people with "little or no knowledge" of programming (e.g., [10, 13, 17, 23]), but was there an important difference between the "little" vs. the "no" subpopulations?

To investigate, we separated these two subpopulations as follows. We counted anyone who said they had ever done any form of "programming" (even a course in high school, or having worked with HTML) as having little knowledge: 56 participants fell into this category. Otherwise we classified them as having no knowledge: 67 participants were in this category. We emphasize that "little" here indeed means very little: recall from Section III.B, that *nobody* beyond a bare minimum of programming background was allowed to participate in the study.

Although we believed that users in the "no knowledge" category would do equally well as the "little knowledge" category because of our experiment's tutorial, those with little programming knowledge scored significantly higher than those with none at all (Fisher's test on task performance (Little knowledge: 35 participants scored at or above the grand median and 21 did not; No knowledge: 28 participants scored at or above the grand median and 39 did not) p=.0297).

TABLE V. AVERAGE PARTICIPANT RATINGS OF THE HELPFULNESS OF THE 6 IDEA GARDEN FEATURES. (ON 94 INSTEAD OF 95 PARTICIPANTS BECAUSE ONE IDEA GARDEN PARTICIPANT DID NOT RATE ANY FEATURES.) IN THIS PAPER, SIGNIFICANT VALUES ARE HIGHLIGHTED. ***: P<.001, **: P<.01, *: P<.05.

Average response to "This feature helped me accomplish my task" (5-point Likert)	**Number of Participants**
>3.0	57 (60.6%) ratings averaged agreement
=3.0	13 (13.8%) ratings averaged neutral
<3.0	24 (22.5%) ratings averaged disagreement
Sample mean = 3.22 One-sample t-statistic = 3.22, DF = 93, p-value =.00176***	

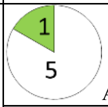
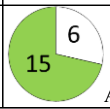
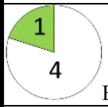
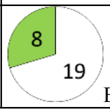
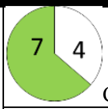
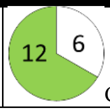
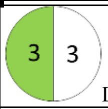
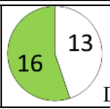
This factor seems particularly important to Idea Garden evaluation because the Idea Garden targets users most like the "little" subpopulation—i.e., users who can already do enough in the programming environment to actually encounter a barrier and get stuck. To illustrate, the recorded log for one "little knowledge" participant, P11544 shows that she did not know to try to incorporate a second webpage—but when the Idea Garden suggested it, she followed the suggestion and succeeded. In contrast, a "no knowledge" participant, P22066, also saw the suggestion—but instead of trying to use two pages together, he switched to a different webpage altogether, which was not useful to his problem.

Regarding *when* (i.e., situation), a possibility that arose was a difference in difficulty between the Pet and the Apartment tasks. Participants seemed to have more trouble with the Pet task than with the Apartment task. For example, participants across *all* treatments scored an average of 2.1 points lower on Pet than on Apartment, and a significantly higher portion of participants commented on difficulties with the Pet task than did with the Apartment task (Fisher's test on comments regarding task difficulties (Pet: 25 participants described difficulties and 98 did not; Apartment: 7 described difficulties and 116 did not), p=.001). Task difficulty is a relevant issue here, because the Idea Garden is called upon only when a task is hard enough that a user runs into difficulties. One example of such difficulties with Pet came from Participant P12344:

P12344: "Couldn't find '# of reviews' for the shelter, then realized too late that I could find the info. on another web page."

Thus, taking the "who" and "when" factors into account, we used Fisher's exact test to compare the number of participants who scored above the grand median to those who scored below, separating by "little" vs. "no" subpopulation and separating the difficult (Pet) task from the easier (Apartment) task. For the targeted situation as per the discussion above—those with little knowledge of programming working in the fairly difficult Pet task—significantly more Idea Garden participants than Control participants scored above the grand median (Fisher's (1, 5; 15, 6), p=.0265), as illustrated by Table VI. Participant means in this category echo this summary, with

TABLE VI. SCORES AT OR ABOVE (COLORED SLICES) OR BELOW (WHITE SLICES) THE GRAND MEDIAN FOR IDEA GARDEN VS. CONTROL, SHOWN FOR ALL WHO/WHEN COMBINATIONS. (IDEA GARDEN HAS MORE PARTICIPANTS BECAUSE IT HAD THREE TREATMENTS.) IDEA GARDEN PARTICIPANTS SCORED SIGNIFICANTLY BETTER THAN CONTROL PARTICIPANTS IN THE IDEA GARDEN TARGET SITUATION (THICK BORDER).

Task	**Little knowledge**		**No knowledge**	
	Control	Idea Garden	Control	Idea Garden
Pet	 A1	 A2	 B1	 B2
	Fisher's exact test p = .0265*		not significant	
Apt	 C1	 C2	 D1	 D2
	not significant		not significant	

Idea Garden participants averaging a score of 6.08 vs. the Control participants' mean of 3.51. Participants' ratings confirmed this result: as Table VII shows, participants in the target situation rated the helpfulness of the Idea Garden features significantly higher than the expected population mean of 3.0 (one-sample t=2.46, df=20, p=.0231). In essence, these results say that the Idea Garden helped participants with little knowledge learn enough to do a programming task on their own, without support, provided that the task was sufficiently difficult.

C. *RQ2: Who alone? When alone?*

Finally, we consider whether combining "who" and "when" as above obscures one of the "who" or "when" factors *alone* being responsible for the significant difference in performance in Idea Garden Participants versus Control Participants.

The result was that neither factor alone explained the results. Table VIII shows suggestive differences based on subpopulation alone, and Table IX shows suggestive differences based on task difficulty alone, but these differences did not rise to significance.

The lack of significance for either factor alone could be due to the combination of the ceiling and floor effects in our data. Specifically, the Apartment task showed a "ceiling effect" in which *everyone* did pretty well, which diluted differences in the more difficult Pet task when the task data were combined. Likewise, no-knowledge participants' floor effects (i.e., most gained little from the Idea Garden) diluted the differences the other participants showed. Investigating this possibility by isolating the factors for separate analysis did not resolve the question, because it left sample sizes so small that statistical differences would be unlikely. Thus, answering the impact of each factor alone will require follow-up empirical investigation.

TABLE VII. PARTICIPANT RATINGS OF IDEA GARDEN FEATURE HELPFULNESS FOR PARTICIPANTS WITH LITTLE KNOWLEDGE, IN THE PET TASK.

Response to "This feature helped me accomplish my task"	Number of Participants
>3.0	16
= 3.0	0
<3.0	5
Sample mean = 3.29	
One-sample t-statistic = 2.46, DF = 20, p =.0231*	

TABLE VIII. PARTICIPANTS WHO SCORED AT OR ABOVE (COLORED SLICES) THE GRAND MEDIAN SEPARATED BY LITTLE OR NO KNOWLEDGE. IN BOTH SUBPOPULATIONS, IDEA GARDEN PARTICIPANTS SCORED SOMEWHAT HIGHER THAN CONTROL PARTICIPANTS, BUT WHEN TASK WAS NOT TAKEN INTO ACCOUNT, THE DIFFERENCES DID NOT RISE TO SIGNIFICANCE.

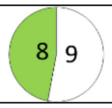
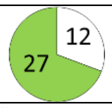
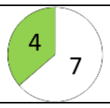
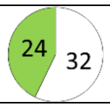
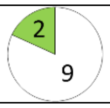
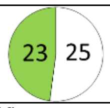
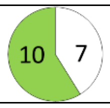
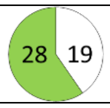
Little Knowledge		No Knowledge	
Control	Idea Garden	Control	Idea Garden
			
Not significant		Not significant	

TABLE IX. PARTICIPANTS WHO SCORED AT OR ABOVE (COLORED SLICES) THE GRAND MEDIAN SEPARATED BY TASK. IN THE MORE DIFFICULT PET TASK, IDEA GARDEN PARTICIPANTS SCORED SOMEWHAT HIGHER THAN CONTROL PARTICIPANTS, AND IN THE EASIER APARTMENT TASK, THEY SCORED ALMOST IDENTICALLY. WHEN SUBPOPULATION WAS NOT TAKEN INTO ACCOUNT, THE DIFFERENCES DID NOT RISE TO SIGNIFICANCE.

Pet		Apartment	
Control	Idea Garden	Control	Idea Garden
			
Not significant		Not significant	

VI. OUR RESULTS IN CONTEXT

Most empirical studies of systems supporting end-user programmers have not considered the difference between "little knowledge" and "no knowledge". In fact, when Dorn's study of a case-based informal learning system for learning Adobe Photoshop scripting did not significantly increase participants' performance, Dorn hypothesized that a reason may have been the variety of his participants' prior programming experience [10]. Our results provide evidence to support Dorn's hypothesis.

Another approach with some similarities to the Idea Garden is Wrangler's proactive suggestions that recommend actions for users to take [14]. Guo et al. investigated Wrangler's suggestions in the context of a data-transformation tool but found that these suggestions did not improve task performance [14]. The Wrangler participants, unlike ours, were computer science students, and they generally ignored the suggestions. This result seems consistent with our result about the difficulty of the task: suggestions seem unlikely to make much difference when the participant does not need them.

Like the Idea Garden, the stencils-based tutorials investigated by Harms et al. aimed to facilitate learning of a UI in order to transfer the skills to a new context [15], but unlike the Idea Garden, that approach used scripted tutorials. With this approach, children were able to ask for step-by-step guidance when using a visual progamming system. Results showed that children using stencils completed more transfer tasks. This result is consistent with our transfer task results.

In the context of these other studies, our statistical results are among the strongest that we have seen on learning-to-doing by end-user programmers. Learning-to-doing takes time, and producing significant effects after only a 25-minute learning task demands a very effective approach. For example, Dorn's results were able to support only learning, not learning-to-doing [10]. Harms et al. [15] succeeded at showing learning-to-doing, but in that study the learning support tools were still available during the transfer task, so additional learning was allowed to take place. The learning support tools were also present in the case of Dorn's study. In contrast, in our study, we isolated learning transfer from learning, by requiring Idea Garden participants to demonstrate learning transfer after the Idea Garden was no longer available to them.

VII. Open Questions for the Idea Garden Approach

Our results raise a number of open questions regarding the Idea Garden approach.

One question that arises is whether there is a "best" Idea Garden variant. Although no treatment was significantly better than any other, the Programming treatment trended better for the little-knowledge participants doing the difficult task. These participants scored on average 4.56 points higher than Control participants (Table X), and had the largest percentage of participants who scored at or above the median (83.3%) (Table XI). Also, all of these Programming participants rated the Idea Garden as helpful (Table XII). These trends lead to this open question:

Open Question 1: Is the Programming variant of the Idea Garden more effective than the others? If so, why?

If Programming is the best variant, one attribute that may account for it may be that it was concrete enough for participants to act upon. The Programming content focused on programming concepts and mini design patterns in particularly concrete and actionable ways. For example, Programming's Generalize-with-repeat feature, triggered by the participant's own code, explained iteration in the context of that code. Participants' favorable comments afterward suggest that they knew how this content applied to their current barrier:

P13411: "It was nice that [the Idea Garden] recognized when I would want to use the `repeat` *command".*

P23344: "This was helpful because getting the script to work for all rows and columns was tricky for me at first".

The Strategy features, on the other hand, were a little less situated, providing more general problem-solving guidance. Strategy content helped a number of participants (Table XII), but others could not figure out how to apply the strategy guidance:

P21055: "[It was] not clear enough on how to work backwards."

P23255: "It[']s an Ok suggestion, but it doesn't say how to 'join the solutions together' ."

Idea Garden content length may also be implicated. The Programming and Strategy contents were shorter than the Combined variant's content, and the Attention Investment model [1] predicts that users may therefore find the Combined variant less cost-effective. This prediction is consistent with the Combined variant's lower ratings than the other two variants in Table XII.

However, at odds with shorter length is the notion of comprehensiveness. This trait was one of the goals of the Combined variant—to provide both relevant problem-solving guidance and relevant programming knowledge all in one place. Because comprehensiveness of information has been positively associated with users' trust in a system [9], a decision to reduce comprehensiveness in favor of brevity should not be made lightly.

Further, the issue of trust is not a matter of comprehensiveness alone [9]. People form impressions of trust quickly, and there are many factors involved. Further, a lack of trust in a system has been linked to disuse of the system. Thus, the issue of end users' trust in a system's advice seems important:

Open Question 2: What factors influence an end user's trust in advice offered by systems like the Idea Garden, and how do these factors influence ways users process and act upon the offered advice?

Comprehensiveness raises another issue as well. Research has shown that, in the aggregate, males and females process information differently, with males preferring to selectively follow and act upon salient cues and females preferring to process information comprehensively before acting upon it [19]. This phenomenon may in part explain why male and female end-user programmers make use of different features when programming and debugging [3, 11]. Our data are consistent with these results, with females trending better with the Combined treatment than with other treatments, but males trending better with the Programming treatment (Table XIII). This leads to our third open question:

Open Question 3: How can we design Idea Garden features to support both the comprehensive information processing style that is statistically associated with females and the selective

Table X. Summary statistic for Idea Garden participants with little knowledge who did the Pet task.

Treatment	N	Mean	Median	StdDev
Control	6	3.51	2.25	4.74
Strategy	8	5.23	5.13	4.58
Programming	*6*	*8.06*	*7.9*	*4.17*
Combined	7	5.34	4	4.04

Table XI. Task performance of participants with little knowledge who did the Pet task. Programming treatment had the highest percentage of participants scoring at or above the median.

Treatment	< grand median	>= grand median
Control	5	1 (16.7%)
Strategy	3	5 (62.5%)
Programming	*1*	*5 (83.3%)*
Combined	2	5 (71.4%)

Table XII. Subjective ratings of participants with little knowledge who did the Pet task. The Programming treatment had 100% of its participants finding the Idea Garden helpful.

Treatment	Not Helpful	Neutral	Helpful
Strategy	2	0	6 (75%)
Programming	*0*	*0*	*6 (100%)*
Combined	3	0	4 (57%)

Table XIII. Task performance of all males and females broken down by treatment. Females performed best with combined whereas males performed best with programming.

Treatment	Females		Males	
	< grand median	>= grand median	< grand median	>= grand median
Strategy	13	8 (38%)	3	5 (63%)
Programming	9	8 (47%)	*6*	*11 (65%)*
Combined	*8*	*13 (62%)*	5	6 (55%)

information processing statistically associated with males [20]?

We plan to investigate these and similar questions to better determine how to improve the effectiveness of Idea Gardens on busy end users when they encounter barriers to getting their tasks done.

VIII. Conclusion

In this paper, we have presented a learning-to-doing (learning transfer) study of the Idea Garden's ability to help end-user programmers help themselves.

The results were that the Idea Garden helped end users with little knowledge of programming write significantly higher-quality programs in the difficult programming task, as compared to participants who had not previously used the Idea Garden.

This finding is somewhat remarkable in that learning transfer occurred after only 25 minutes exposure to Idea Garden support. In addition, this result is the first learning transfer investigation of end-user programming that we have been able to locate in which participants did not have access to the learning supports during the transfer task itself. Thus, it showed both that participants retained the learned information and that they were able to apply it to new contexts later without help.

Finally, this study is also the first we have seen in end-user programming that investigates the difference between end users with little knowledge of programming (e.g., prior experience with html or with statistical scripts) and those with none at all. Prior studies have combined these two subpopulations, and our results suggest that, at least in some situations, the distinction is important.

In summary, the Idea Garden helped make a little programming knowledge go a long way in helping these end-user programmers in trouble to help themselves. As "active users" with no particular motivation to learn programming, these end users were able to synthesize the knowledge presented by Idea Garden and apply that knowledge without guidance or assistance. Thus, with the Idea Garden's help, they not only learned—they learned to do.

Acknowledgments

We thank our study participants and Romina Rodriguez for her assistance with the study. This work was supported in part by NSF grants 0917366 and 1240786.

References

[1] A. Blackwell, First steps in programming: A rationale for attention investment models, IEEE HCC, pp. 2–10, 2002.

[2] J. Bransford, A. Brown, and R. Cocking, How People Learn: Brain, Mind, Experience, and School, Expanded ed., National Academy Press, 2000.

[3] J. Cao, K. Rector, T. Park, S. D, Fleming, M. Burnett, and S. Wiedenbeck, A debugging perspective on end-user mashup programming, IEEE VL/HCC, pp. 149–156, 2010.

[4] J. Cao, Y. Riche, S. Wiedenbeck, M. Burnett, and V. Grigoreanu, End-user mashup programming: Through the design lens, ACM CHI, pp. 1009-1018, 2010.

[5] J. Cao, S. D, Fleming, and M. Burnett, An exploration of design opportunities for 'gardening' end-user programmers' ideas, IEEE VL/HCC, pp. 35-42, 2011.

[6] J. Cao, I. Kwan, R. White, S. D. Fleming, M. Burnett, and C. Scaffidi, From barriers to learning in the Idea Garden: An empirical study, IEEE VL/HCC, pp. 59-66, 2012.

[7] J. Carroll. Minimalism Beyond the Nurnberg Funnel. MIT Press, 1998.

[8] D. Compeau and C. Higgins, Computer self-efficacy: Development of a measure and initial test, MIS Quarterly 19(2), pp. 189–211, May 1995.

[9] C. L. Corritore, B., Kracher, B., and S. Wiedenbeck, On-line trust: Concepts, evolving themes, a model. International Journal of Human-Computer Studies, 58(6), pp. 737 – 758, 2003.

[10] B. Dorn, ScriptABLE: Supporting informal learning with cases, ICER, pp. 69-76, 2011.

[11] V. Grigoreanu, J. Brundage, E. Bahna, M. Burnett, P. ElRif, and J. Snover. Males' and females' script debugging strategies. Second International Symposium on End-User Development, Siegen, Germany, March 2-4, 2009.

[12] V. Grigoreanu, M. Burnett, and G. Robertson. A strategy-centric approach to the design of end-user debugging tools, ACM CHI, pp. 713-722, 2010.

[13] P. Gross, J. Yang, and C. Kelleher, Dinah: An interface to assist non-programmers with selecting program code causing graphical output, ACM CHI, pp. 3397-3400, 2011.

[14] P. J. Guo, S. Kandel, J. M. Hellerstein, and J. Heer. Proactive wrangling: Mixed-initiative end-user programming of data transformation scripts. ACM UIST, pp. 65-74, 2011.

[15] K. J. Harms, C. H. Kerr, and C. L. Kelleher, Improving learning transfer from stencils-based tutorials, ACM IDC, pp. 157-160, 2011.

[16] A. Ko, B. Myers, and H. Aung, Six learning barriers in end-user programming systems, IEEE VL/HCC, pp. 199–206, 2004.

[17] S. Kuttal, A. Sarma, and G. Rothermel, History repeats itself more easily when you log it: Versioning for mashups, IEEE VL/HCC, pp. 69–72, 2011.

[18] J. Lin, J. Wong, J. Nichols, A. Cypher, and T. Lau, End-user programming of mashups with Vegemite, ACM IUI, pp. 97–106, 2009.

[19] J. Myers-Levy, Gender differences in information processing: A selectivity interpretation, in Cognitive and Affective Responses to Advertising, P. Cafferata and A. Tybout (eds.) Lexington Books, 1989.

[20] O'Donnell, E. and Johnson, E. N. Gender effects on processing effort during analytical procedures. International Journal of Auditing 5, pp.91-105, 2001.

[21] H. Simon, Problem solving and education, in Problem Solving and Education: Issues in Teaching and Research, D. Tuma and F. Reif (eds.) Lawrence Erlbaum, 1980.

[22] J. Sweller, Cognitive load during problem solving: Effects on learning, in Cognitive Science 12, pp. 257-285, 1988.

[23] N. Zang and M. B. Rosson, What's in a mashup? And why? Studying the perceptions of web-active end users, IEEE VL/HCC, pp. 31-38, 2009.