I. INTRODUCTION

End-user programmers encounter many barriers as they try to accomplish their tasks. Researchers have developed solutions to help end users overcome some of these barriers, such as arcane language constructs that are difficult to learn, syntax rules that are hard to memorize, and suitable example programs that other end-users have written [1].

Approaches developed previously to tackle these issues include programming by demonstration [2], natural programming methodology [3], and mashup environments. However, in spite of such efforts, empirical experiments show many end-user programmers still struggle (e.g., [2, 3]). One possible explanation for this phenomenon is that these solutions try to remove these barriers rather than trying to empower users through increasing end users’ problem-solving skills, creativity, and design thinking. To fill this gap, recently, Cao et al. [1, 4] introduced the Idea Garden approach. Idea gardening is not intended to solve problems automatically on its own or replace online tutorials. Instead, it supplements online tutorials by suggesting problem solving strategies, programming domain concepts, and patterns based on the users’ actual tasks.

II. RESEARCH QUESTIONS

Since the Idea Garden approach has been prototyped only in CoScripter, my main research goal is whether the Idea-Garden approach can be generalized across end-user programming environments such as gaming programming environments and if yes, to what extent.

The following sub-questions arise from this main goal.

RQ1. To what extent do end-users’ programming barriers found in previous settings generalize? Cao et al. [5], Ko et al. [6], and Kissinger et al. [7] have studied barriers in end-user programming environments. Although there is an overlap among these sets of barriers, the literature has not reported how the barriers generalize across environments.

RQ2. What is the appropriate trigger for suggesting didactic problem-solving approaches or concepts to end users? Building on the Surprise-Explain-Reward strategy [8], Cao et al. [1, 4, 5, 9] studied triggers for showing suggestions in CoScripter. As part of Idea Garden generalization, I would like to devise a principled, more general approach to triggering Idea Garden suggestions in end-user programming environments which ensures that information is presented at a time that the user benefits the most from it while being minimally disrupted.

RQ3. How can a suggestion become balanced in terms of its intentionally flawed part and its correct part? On the one hand, the Idea Garden approach targets the population that Carroll et al. refer to as “active users” [10] under the “Minimalist Learning Theory” [10]. This theory implies that only the material most relevant to the active users’ tasks should be taught to them. On the other hand, some parts of Idea Garden features are intentionally flawed so they do not simply provide working solutions. The primary reason of having some parts as flawed is to arouse end users’ curiosity and keep them intellectually engaged. Having mentioned these two points, detecting the appropriate degree of “flawness”, identifying the parts that could be flawed and the parts that should be correct is hard.

RQ4. What problem-solving strategies are especially helpful to end-user programmers “stuck” on how to make their programs work? The existing literature talks about several problem-solving strategies such as “Divide and Conquer”, “Sleep on it”, “Working backward”, “Analogy” and “Generalization”. However, in [9] we saw that some users had problems understanding some of the strategies. So, further work is needed in order to find the most helpful strategy of helping end users in each circumstance.

III. RESEARCH APPROACH

I am currently working on RQ1. As part of this, I chose a game programming environment for end users called Gidget [11], a programming game in which the user corrects the faulty code of a broken robot so that it completes its missions (see Fig. 1). I am currently conducting a one-on-one think-aloud study that began this spring. In the study, I have been observing teenagers playing Gidget, collecting audio video recordings and observation notes. I am in the process of qualitative coding using code sets derived from [5, 6, 7].

Studying barriers in Gidget is the first step toward generalizing the Idea Garden approach to this environment. The next step will then be to design features that address these barriers. There are at least two factors that make this step challenging. First, Gidget is a gaming environment and
so the design of features must conform to the design practices of gaming environments. For example, I have started implementing a “telephone call” metaphor as a means of communication between the player and a character who provides information in the style of the Idea Garden. The telephone call metaphor is a common way of offering help to players in many games.

Second, Gidget’s target population is teenagers which is distinguished from the adult populations originally studied in order to derive the Garden approach. There may be differences in the learning approach and computer problem-solving strategies of these two populations. Consequently, the current Idea Garden approach may need to adjust to age-related end-user programming differences. How these differences affect the Idea Garden approach may require further investigation.

IV. HOPED-FOR BENEFITS FROM THE CONSORTIUM

I hope to receive feedback, expertise and insight on:

1. Going back to my main research goal, finding the appropriate methodologies for investigating the generalization of the Idea Garden approach. The Idea Garden approach was originally implemented in the CoScripter. Currently, I am implementing Idea Garden features within Gidget. However, these two implementations do not provide enough evidence for the generalization of the Idea Garden approach.

If we can feasibly show that the Idea Garden approach can be generalized, we will then be able to create a stand-alone plugin that can be applied to a variety of end-user programming environments and thus we will see less instances of end users feeling “stuck,” saying: “I am not sure what to do next.”

2. Going back to research questions 2, 3 and 4, finding the appropriate timing for Idea Garden triggers, the appropriate “flawness” of suggestions and the appropriate problem-solving strategies if we can identify the proper triggers, proper flawness, and the proper problem-solving strategies, then we can further refine the Idea Garden approach, and therefore more effectively help end users.

REFERENCES


