# Applying Cognitive Load Theory to Generate Effective Programming Tutorials

Kyle J. Harms
Department of Computer Science & Engineering
Washington University in St. Louis
St. Louis, Missouri, United States
harmsk@seas.wustl.edu

## I. Introduction

Many programmers, including novices, often attempt to solve their coding problems by searching the web for example code [1]. Because these programmers are invested in their current project, they are motivated to discover a solution to their problem [2]. While searching the web for a solution, they may encounter source code that addresses their issue and also contains programming concepts that are new to user. Unfortunately, novice programmers often struggle to understand the source code and can fail to successfully integrate the code into their programs [3]. To help novice programmers understand unfamiliar source code, we propose automatically generating tutorials from source code to help these programmers acquire new programming skills in the course of working on their own chosen projects.

In this paper, I describe our prior work on how automatically generating walk-through tutorials helps novice programmers learn new programming concepts found in unfamiliar code [4]. Following this, I introduce my proposal to use Cognitive Load Theory to improve the effectiveness of learning new programming concepts from generated tutorials. To accomplish this, I propose automatically generating personalized tutorials based on a user's programming expertise.

## II. Automatically Generating tutorials

We initially hypothesized that novice programmers would effectively learn programming constructs from automatically generated interactive, in-context tutorials that walk the user through every step necessary to reconstruct unfamiliar code within their program [4]. To evaluate our hypothesis we developed automatically generated programming tutorials in the novice programming environment Looking Glass [5]. To automatically generate a tutorial, we use a model-driven architecture to translate each statement in the code into a sequence of actions that a user must take in order to reconstruct that statement.

We conducted an evaluation to assess whether the generated tutorials would help users learn new programming concepts found within unfamiliar code. We asked participants to complete several near transfer tasks that required the use of a programming concept found in the unfamiliar code. Compared to the control condition, whose participants did not complete a tutorial, the users in the experimental condition performed 64% better on the near transfer tasks. While the experimental condition did perform better than the control condition, users given the tutorial only averaged a 52.2% success rate when completing the near transfer tasks. This result advocates for refining the tutorials to improve their effectiveness.

Interestingly, the limited learning that occurred in our evaluation is predicted by Cognitive Load Theory. Fundamentally, learning depends on working memory, which is a necessary and limited resource that can be easily overwhelmed. Upon reflection, we believe our generated tutorials present too much information for the user to absorb and thereby overwhelm the user's working memory. The tutorials contained a step for every action required to reconstruct the code. However, many of these tutorial steps are unrelated to programming constructs; instead these steps configure the state of the interface (e.g. selecting a tab). In fact, the number of steps related to a programming construct for a tutorial averaged 17% of the total steps. Cognitive Load Theory suggests that many extraneous steps will increase cognitive load and reduce learning.

We predict that by employing techniques from Cognitive Load Theory we can improve the effectiveness of our generated tutorials by personalizing each tutorial to a user's programming expertise. While other generated tutorial systems do exist for photo manipulation [6] or drawing [7], we are unaware of any tutorial generation system that generates personalized tutorials based on user expertise.

## III. Cognitive Load Theory

Cognitive Load Theory (CLT) suggests that by carefully managing the working memory needs of a learner during an instructional task, we can increase the efficiency of learning [8]. Conversely, if a task overwhelms working memory capacity, learning will not occur [8]. A learner's working memory is affected by both intrinsic and extraneous cognitive load for an instructional task.

Intrinsic cognitive load is the inherent complexity of a task for a learner. Because intrinsic cognitive load is grounded by the learner's expertise, intrinsic cognitive load cannot be altered by the design of an instructional task.

Extraneous cognitive load is the extra load beyond the intrinsic cognitive load that is not necessary for learning a

concept, but is generated by the manner in which an instructional task is presented to a learner [8]. Extraneous cognitive load is controlled by changing the design of the task. Often an instructional task will require cognitive resources that are not directly related to the learning objective. An effective way to lower the extraneous load in this situation is to tailor each instructional task to the learner's expertise by eliminating the elements that are not necessary for the concept [9]. In this way, a learner's working memory can be devoted more fully to understanding the desired concept.

Imagine the following scenario. We would like Lucia, a novice programmer, to understand how to use an iterator in a loop. If we ask Lucia to program a loop with an iterator in a traditional programming language, she would also need to learn the programming syntax for the iterator. The programming syntax adds extraneous cognitive load to the task that requires additional cognitive resources beyond what is required to understand iteration. To lessen the cognitive load for the task, we lessen the extraneous cognitive load by asking Lucia to program a loop with an iterator in a drag and drop programming environment. Now Lucia can expend more of her working memory on learning iteration.

## IV. Generating Personalized Tutorials

I hypothesize that by generating personalized tutorials, informed by CLT, we can improve the efficiency of learning new programming concepts found in unfamiliar code for novice programmers. Fundamentally, there are two ways to improve the effectiveness of our generated tutorials by employing CLT: 1) tracking the user's intrinsic cognitive load by modeling their programming expertise, and 2) reducing extraneous cognitive load by carefully selecting programming concepts that do not overwhelm a learner's working memory.

### A. Tracking Intrinsic Cognitive Load

In order to manage the user's intrinsic cognitive load, we need to select instructional tasks that are challenging but not unrealistically difficult for a learner. In order to accomplish this we need to have an understanding of the user's current programming expertise and what programming concepts are contained within an unfamiliar piece of code. To accomplish this, I plan to introduce a user model of the user's programming expertise to the tutorial generation process. I will use this model when automatically generating tutorials to carefully choose one programming concept to present to the user that pushes the boundaries of the user's expertise.

To build the user model, I plan to automatically examine a user's programs for how effectively they use programming concepts. I will also measure the user's cognitive load when following a tutorial and associate the programming concept from the tutorial with the user's current cognitive load. To measure the user's cognitive load, I will use a CLT unidimensional rating scale to measure a user's mental effort during tutorial presentation [10]. The CLT rating scale technique, whereby a learner rates their mental effort on a scale from 1 to 9, is a reliable and sensitive method to measure mental effort [10]. Altogether, I believe that analyzing users' programs and measuring their cognitive load can accurately portray a user's programming expertise.

### B. Reducing Extraneous Cognitive Load

The current version of our generated tutorials likely has a very high cognitive load because most of the steps in the tutorial, while not directly related to new programming constructs, are new information to the user. To effectively manage the user's working memory, I propose tailoring each generated tutorial to only present a single programming concept that meets or slightly expands upon the user's expertise. By employing the user model, I will build a tutorial generation engine which selectively emphasizes tutorial content for a single programming concept, thereby reducing the cognitive load for this instructional task.

For complex programming concepts which have high extraneous cognitive load, I will utilize the progressive method to present the concept to the user. In the progressive method, a simple version of a task is first presented to the learner [8]. Then, progressively over time as the intrinsic cognitive load is reduced, present the entire task to the learner. For the generated tutorials, this means gradually revealing difficult programming concepts over a series of tutorials rather than one cognitively burdensome tutorial.

By employing these CLT techniques, our generated tutorials will be personalized to a learner's skill set and expertise. I will conduct a longer term evaluation to determine if generated personalized tutorials are an effective way to help novices learn new programming concepts.

## References


[1] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two studies of opportunistic programming: interleaving web foraging, learning, and writing code," in *Proc. CHI*, 2009.

[2] M. B. Rosson, J. Ballin, and J. Rode, "Who, What, and How: A Survey of Informal and Professional Web Developers," in *Proc. VL/HCC*, 2005.

[3] M. B. Rosson, J. Ballin, and H. Nash, "Everyday Programming: Challenges and Opportunities for Informal Web Development," in *Proc. VL/HCC*, 2004.

[4] K. J. Harms, D. Cosgrove, S. Gray, and C. L. Kelleher, "Automatically Generating Tutorials to Enable Middle School Children to Learn Programming Independently," in *Proc. IDC*, 2013.

[5] "Looking Glass." [Online]. Available: http://lookingglass.wustl.edu.

[6] F. Grabler, M. Agrawala, W. Li, M. Dontcheva, and T. Igarashi, "Generating photo manipulation tutorials by demonstration," in *Proc. SIGGRAPH*, 2009.

[7] J. Fernquist, T. Grossman, and G. Fitzmaurice, "Sketch-sketch revolution: an engaging tutorial system for guided sketching and application learning," in *Proc. UIST*, 2011.

[8] J. J. G. van Merriënboer and J. Sweller, "Cognitive Load Theory and Complex Learning: Recent Developments and Future Directions," *Educ. Psychol. Rev.*, vol. 17, no. 2, pp. 147–177, Jun. 2005.

[9] J. J. G. van Merriënboer and P. Ayres, "Research on cognitive load theory and its design implications for e-learning," *Educ. Technol. Res. Dev.*, vol. 53, no. 3, pp. 5–13, Sep. 2005.

[10] F. Paas, J. E. Tuovinen, H. Tabbers, and P. W. M. Van Gerven, "Cognitive Load Measurement as a Means to Advance Cognitive Load Theory," *Educ. Psychol.*, vol. 38, no. 1, pp. 63–71, 2003.