Predicting Learning in a Troubleshooting Task using a Cognitive Architecture-Based Task Analysis

Frank E. Ritter (frank.ritter@psu.edu), Deja Workman, and Shan Wang
College of IST, Penn State
University Park, PA 16802

Abstract

We present a new way to do task analysis that includes learning. This approach starts with a hierarchical task analysis of a troubleshooting strategy and applies a power law of learning to modify the time, mimicking the ACT-R learning equations. We apply this approach to finding faults in the Ben Franklin Radar (BFR) system, a 35-component system, designed to study troubleshooting and learning. In this task, faults are introduced into the BFR, and the participants are responsible for finding and fixing these automatic faults. Previous models in Soar took up to 6-9 months of graduate student to create. This model can be created more quickly and provides a model between GOMS and a full cognitive architecture-based model. The predictions will be compared to the aggregate and individuals' data (N=111) and lessons will be reported.

Keywords: ACT-R, learning, task analysis, troubleshooting

Acknowledgements: This work was supported by ONR, N00014-18-C-7015 and N00014-15-1-2275. Charles River Analytics implemented the MENDS simulation, provide it under license, and provided comments and support on the process.

Introduction

Can we examine a particularly complex task and predict how long each trial will take while the task is learned without creating a full information processing cognitive model? In this paper we create predictions of a complex task using GOMS and the learning equations, thus extending the GOMS approach and providing a way to make approximate predictions of learning a task.

To illustrate this approach, we make detailed predictions about trouble shooting a complex task where the fault in the circuits may vary in difficulty. We use GOMS and learning equations.

The rest of the paper describes the task and a model used to create a series of predictions of doing the task and learning. We then present the study used to gather human performance data. We then describe the comparison we will make with the model's predictions to the human data in aggregate form and individually, and already can draw some insights.

Task

We needed a complex task to study. The Ben Franklin Radar (BFR), shown in Figures 1 and 2, is a deliberately 5x larger system than the Klingon Laser Bank task that has been previously used to study problem solving, learning, and transfer (Bibby & Payne, 1996; Friedrich & Ritter, 2020; Ritter & Bibby, 2008). The Klingon Laser Bank task with 7 components initially takes about 30 s and with 20 trials takes about 7 s.

The MENDS simulator was created for the BFR, shown in Figure 1. The schematic and interface can be taught to participants in a 32-page online tutor created in D2P (Ritter et al., 2013). It takes about 30 min. to learn declarative information about it and the task (Ritter, Tehranchi, Brener, & Wang, 2019). The schematic shows five subsystems. The subsystems vary in their complexity and connectivity within them and across subsystems. The blue lines in Figure 1 are power connections; the red lines are information; the purple lines are both. The schematic also identifies certain components that have their status displayed on the front panel of the BFR.

Our task was created to support learning troubleshooting within the confines of a study, and to be more complex than the Klingon Laser Bank task, but not so complex that it would take more than an hour to learn. This system can be and has been realized in several ways with different complexity. The task that we will focus on is to find a single broken component. Single broken faults create a unique light configuration and are always solvable.

The task requires declarative knowledge about the schematic and interface. It also requires some recognition memory and perhaps recall of the components. The task also supports creating procedural knowledge from the declarative knowledge by doing the task.

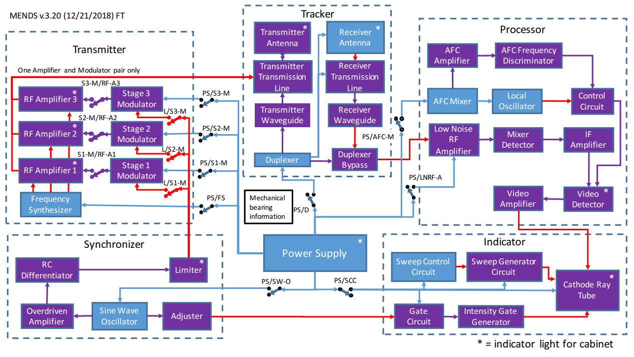

Figure 1. Schematic of the Ben-Franklin Radar simulation.

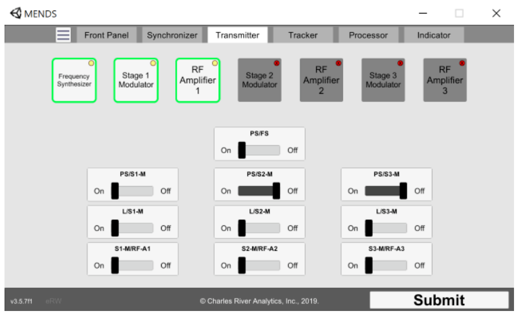

Figure 2. A snapshot of the MENDS simulation. In this picture, the fault is in the Processor subsystem.

A Simple Task Model of Learning and Fidelity

To troubleshoot the task (simplest method) the user clicks for the next problem, examines the lights, clicks on a tray, and examines its contents. They must then choose the broken component by clicking on it and clicking done.

We can start with some insights that are already apparent. The BFR task is designed to be more complex than the Klingon Laser Bank task. Thus, there are more predictions to generate (7 vs. 35). Ritter and Bibby (2008) found that one strategy matched the majority of participants. Seven you can do by hand, 35 requires more infrastructure. Friedrich and Ritter (2020) found with more relaxed instructions there were more strategies. With this complexity may come even further strategies to solve this task. Thus, this model will initially present just one of these strategies.

The model represents the structure of the BRF as a series of connections in a matrix. To create the structure of the model, a binary schematic was created in Excel to represent the component dependencies found in the BFR. Python converted this matrix into a data frame storing components' input and outputs—that is, the list of other components that a component itself expects to receive power from or send power to. The data structure also dynamically stores the component status and light, which respectively represent whether the component is functioning properly and whether it is receiving power, they are both binary variables.

In some sense, we thus create an ad hoc domain specific language (DSL) cognitive modeling language (Kaulakis, 2020) using Excel and Python for reading in circuit matrices. This approach would support creating models of similar structures and is at this point be fairly direct and quick to use. The model uses these structures to generate times to find a fault. Currently, the model can set up and run the task, and solve for a fault.

MakeFault is responsible for the creation of the fault, it begins by calling the *clearFault* function and then it generates a random number that is within the bound of the size of the data frame, which corresponds to one of the 36 components of the system—it cannot choose the power supply or any of the switches as a fault because these are not tasks we present participants within the actual study. *Propagate* is responsible for computing the effect of the fault. This function identifies all the outputs of the piece, and subsequently turns their light off as they are no longer receiving all of their necessary inputs due to the fault.

The holistic responsibility of this operation is not only to recreate the way in which the system breaks itself, more importantly is the program's ability to locate and fix its own faults. This logic is stored predominantly in the *FindFault* function, though it calls upon external elements as well as the mental operator function. *FindFault* is meant to mimic the way in which we believe participants solve the problem using a simple strategy that they are presented with.

Human Participant Data that We Have So Far

We have two sets of data. In the MENDS task, Ritter et al. (2019) saw a subject with 10 minutes of practice that went from 60 s to 22 s. And, we have finished a study (N=110) that has more data. We are analyzing it now and have learning curves for all participants that we will compare to this model. The BFR task does take about 5 times longer than the original task both initially and at 20 trials.

Discussion and Conclusion

The model is still being developed, but offers a way to predict learning for static approaches. This model already predicts that later faults take longer, that learning is inevitable, that many faults take different times, and surprisingly, that many faults would take the same time but use different subtasks to get there. For example, fixing a component late in an early tray may take as long as a component in a later tray but displayed earlier within that tray.

References

Bibby, P. A., & Payne, S. J. (1996). Instruction and practice in learning to use a device. *Cognitive Science, 20*(4), 539-578.

Friedrich, M. B., & Ritter, F. E. (2020). Understanding strategy differences in a diagrammatic reasoning task. *Cognitive Systems Research, 59*, 133-150.

Kaulakis, R. (2020). *Domain-specific languages as a method for representation and evolutionary search among global solution space of cognitively plausible algorithmic behaviors.* Unpublished PhD thesis, College of IST, Penn State, University Park, PA.

Ritter, F. E., & Bibby, P. A. (2008). Modeling how, when, and what is learned in a simple fault-finding task. *Cognitive Science, 32*, 862-892.

Ritter, F. E., Tehranchi, F., Brener, M., & Wang, S. (2019). Testing a complex training task. In *Proceedings of the 17th International Conference on Cognitive Modeling (ICCM 2019)*, 184-185.

Ritter, F. E., Yeh, K.-C., Cohen, M. A., Weyhrauch, P., Kim, J. W., & Hobbs, J. N. (2013). Declarative to procedural tutors: A family of cognitive architecture-based tutors. In *Proceedings of the 22nd Conference on Behavior Representation in Modeling and Simulation*, 13-BRIMS-127. 108-113. Centerville, OH: BRIMS Society.