Variability of behavior in complex skill acquisition

Niels Taatgen

Abstract

In experiments where participants have to perform some new skill, we usually see a much higher variance in behavior than our current models of complex skill acquisition can explain. Currently, cognitive models employ two ways to add extra variance: noise, and the manipulation of architectural parameters that we associate with individual differences, of which the goal activation (W) parameter is most well-known (e.g., Lovett, Reder & Lebiere, 1999). These two sources of variance are probably sufficient in tasks of low complexity in which participants are told exactly what to do, but even in mildly complicated situation other sources of variance can play a major role. Apart from architectural parameters and noise, we can at least identify the following sources of variance:

- General problem solving strategies: differences that we can attribute to whether or not a certain (task-independent) cognitive strategy is mastered or preferred (e.g., attempt to multi-task, verbal rehearsal or mental imagery as a mnemonic strategy)
- Prerequisite skills: some tasks assume that certain skills are already mastered, like operating a mouse and a keyboard, or doing mental arithmetic. Individuals may differ in their mastery of these skills, or may even lack certain skills.
- Task ambiguity: complex tasks can almost always be performed in several different ways. The method that is chosen to perform the task also influences performance, especially at a more fine-grained level (e.g., keystrokes, mouse-movements).

In my talk I will discuss how we can model these sources of variance in ACT-R, but will focus on the third: task ambiguity. I will show examples of task ambiguity in two complex dynamic tasks (Air Traffic Control – KA-ATC, and Airplane identification – CMU-ASP). To model task ambiguity, a representation of instructions is used inspired on the APEX architecture, in which instructions are only partially ordered – allowing several different orders of execution. Once these instructions are proceduralized, ACT-R tends to choose the more optimal strategies (utility learning). It can, however, get stuck in a suboptimal strategy. This is the case when a potentially more optimal strategy hasn't been practiced yet, and is therefore slower then the suboptimal strategy.
I will demonstrate these issues by showing some comparisons between model runs and data from individuals in the CMU-ASP task.