# What constitutes the “understanding” stage of skill development?

The progression of skill development is complex and different studies often have different definitions and hallmarks of what expert performance is. Anderson (1982) asserts that skill learning involves interpreting recently acquired declarative information about how to perform a task into efficient procedural rules that we can observe in expert skill performance. But several reviews of traditional programming-language learning describe this transference of declarative programming knowledge to practiced proceduralization as understanding, a term that makes the process more obscure. Anderson et al. (1999) helpfully expands the concept and progression of understanding into closely observable learning strategies: direct recall, analogical application of an example, declarative rule formation, and procedural rule formation with further practice. How Anderson expands upon more general thoughts on skill development is discussed in the following paragraphs.

Mayer (1981), an influential review of programming language learning, draws a distinction between “understanding” and merely accumulating detailed information of the parts of a programming language, which Anderson would call declarative knowledge. According to Mayer (1981), understanding is the ability to manipulate and successfully apply the acquired information towards new problem-solving situations. Much like Anderson’s description of skill development (Anderson, 1982), Mayer’s definition above suggests a trajectory of learning from beginner to expert that is attainable through practice. But the similarities between the two, Mayer and Anderson, stop there as Mayer and others fail to present a detailed view of how understanding progresses from novice to expert. Anderson et al. (1999), on the other hand, expands the progression of skill development or understanding by examining the different strategies that learners use to apply declarative information.

According to Anderson et al (1999), learners use several strategies to transfer practiced knowledge to new situations. This was determined in a simple but programming-adjacent problem-solving experiment. Here, subjects form intuitions from explicit examples on how to solve specific problems using a mixture of strategies that vary in levels of efficiency: 1) learners, rapidly but limitedly, can directly recall responses to problems they have seen before; 2) they can analogically extrapolate a solution by recalling a similar example, but this is taxing and inefficient; 3) with practice, they can extract a declarative rule and apply it without referring back to examples; 4) and lastly, with more practice, they can form very efficient procedural rules that lead to a type of performance we would often recognize as expert level. However, the study had observed that a mixture of these strategies was used by learners throughout the experiment sessions. The strategies selected by the learners did not follow each other linearly from early, more declarative strategies to later, more procedural strategies. This shows that different avenues maybe taken to arrive at expert performance or understanding and that it is more complex than the term understanding merely suggests. Anderson et al (1999) sheds some light on this complex process and presents a multi-dimensional view that includes the problem-solving dimension as well as the memory systems that support the learned skill.

**Cut**

~~Teaching coding usually starts out by defining the components of the programming language that then must be practiced and streamlined (Robins et al. 2003). For example, some fundamental items taught to beginners are how to use functions, how to define variables, and how to use different data types and operators. Mayer (1981) examines several strategies as to how this declarative information can be acquired that have implications for teaching and are necessary first steps in skill acquisition. These learners are then assisted in practicing how these items may be combined to produce functioning programs that often involves exposure to new problems.~~