Functional Programming and Software Architecture

What’s the big deal?

Paul Newell

Functional Programming brings many beneficial quality attributes that Software Architects are always striving to meet with today’s existing programming paradigms: Reliability and scalability. Functional programming may provide the best support for these two quality attributes, but if designed and implemented correctly, other programming paradigms can also meet these quality attributes. Does functional programming allow a software architect to achieve these quality attributes easier? Maybe, but are there other quality attributes that functional programming makes harder. Is performance or maintainability goals, for example, harder to meet with functional programming? In the end, does it really matter what programming paradigm is used, as long as the Software Architect and developers can meet the business requirements and goals?

Functional programming is another tool that developers and software architects can use to help meet the business requirements and goals. When a stateless system is being developed, functional programming should be considered. However, designing the software control system for traffic lights, functional programming is probably not the correct paradigm to pick. What should a software architect look for in a paradigm to determine if functional programming is the right paradigm? The example above is one indication, a state or stateless system. Distributed or non-distributed systems, immutable or mutable objects, intended size of the code base, are some of the questions that software architects should consider when identifying the paradigm.

Is the thought process change needed for functional programming worth the change from most of the other paradigms? While several of the other programming paradigms have the common loop structure (i.e., while and for), functional programming is looping using recursion. There is a thought process change that is needed to make a For loop into a recursive loop. Is the recursive call easier to maintain than a For loop? The hardest aspect of designing recursive calls is getting the stop condition correct. If I modify the recursive call, did I just violate the stop condition? True functional programming prevents changing a variable once it is set. Do multiple copies of the variables for every change cause problems in meeting the business requirements and goals?

In summary, while functional programming has its benefits, it should not be the go to programming paradigm for the software architect. The software architect needs to evaluate the programming paradigm that best meets the business requirements and goals. Functional programming may be the best at reliability and scalability, but it does have weakness as well. The software architect needs to understand the strengths and weakness and make the choice based on their needs.