To start, this month's column brings the experimental exploration of category theory from Lewvere’s and Schanuel’ book to an end. Why? Well, up to this point, the invested effort seems to be far larger than the gained benefit. Category theory promises deep insight into more complex ideas – otherwise why would anyone study it – but the amount of effort required to get to that place in the text by Lewvere and Schanuel seems overwhelming. There is no middle ground in their work. Either it is painfully pedantic or deeply complex.

So, as a change of pace, this column will be returning to simpler, one-off topics. For this month, I thought it would be interesting to share a thought about the overlap between philosophy and computer programming that came about from an interesting video. In this video, Dave Farley of Constant Development discusses the history of computing paradigms with an eye towards the tribal standoffs between the object oriented crowd versus the functional programming adherents, with particular attention between the tribal standoffs between the two over which paradigm is best.

<iframe width="560" height="315" src="https://www.youtube.com/embed/wyABTfR9UTU" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>

<https://youtu.be/wyABTfR9UTU>

While it is both comforting and amusing is the Dave says that both positions are ‘rubbish’ (with a delightful accent no less). I heartily agree with his basic point that one should pick the right paradigm for the right job. Along those lines, the most insightful thing that he had to say is an observation he attributed to “Uncle” Bob Martin. Martin, a computing pundit of some note, says the purpose of a programming paradigm to constrain us, to take away a degree of freedom that we had otherwise been allowed.

After reflecting on this point on more than one occasion, I realized, even though Farley didn’t say this, that the idea of constraining the world as part of a paradigm is both an ontological and epistemological metaphysical process whose scope is far larger than just computer programming. A paradigmatic way of thinking constrains the way we perceive reality by constraining how we divvy up the world around us into pieces we understand.

One of the earliest texts covering the idea of chopping up the world into useful chunks is found in the philosophical work of Aristotle entitled the [Categories](https://en.wikipedia.org/wiki/Categories_(Aristotle)). In this work, Aristotle lists 10 separate categories that that cover all the possible things that can be the subject of a proposition. The work centers around language and how parts of speech reflect the ideas we hold and this ten-fold division was meant to encompass everything a person can think (or perhaps just talk) about. While the work is useful, because it deals with common observations and mode of thought we all share, it is dense and sometimes difficult. Mortimer Adler, in his book Aristotle for Everyone, calls it uncommon common sense. Fortunately, the key point for this discussion is that Aristotle offers a paradigm (perhaps hard to grasp) that constrains how we think (or describe) the world.

A more prosaic example comes from our everyday experience of communicating with our fellow beings. For example, many of us would consider the word ‘automobile’ to be straightforward to define and we expect that the dictionary entry would be non-controversial. However, exactly how each of us thinks about ‘automobile’ depends on the paradigm being used.

For a thermal engineer, whose livelihood centers on the study of work and heat, an automobile make conjure the concept of a gasoline or diesel engine, a drive chain and an exhaust. He has constrained the world to one dealing with how energy is stored, how much of it can be stored in a given volume, and how it moves from place to place. A systems engineer, whose livelihood derives from understanding the interplay of various technologies, would mostly see a car as a set of subsystems designed to interact appropriately. Does the engine provide the necessary power to match the steering and suspension? Does the fuel tank and milage match the intended purpose? And so on. Our systems engineer has constrained his view of the automobile in such a way ignores the details of thermodynamics and deals only with the abstract ways in which such subsystems interface with each other. To an urban planner, the automobile represents an atomic entity whose characteristics and behaviors are constrained to be at the driving level. Doing so, allows the urban planner to deal with matters of traffic congestion, proper placement of tolls, and the like without concerning himself with the way in which the automobile moves or how well it moves.

Now let’s returning to the question of computer programming paradigms. Certainly, a programmer must know, when modeling an automobile, what level his solution must deal with. Is he programming a finite-element thermal model, an abstract entity-relationship, or a stochastic Monte Carlo simulation to match the fictitious engineers of the previous discussion. However, the interesting aspect of a seasoned programmer is that when thinking about programming paradigms in the abstract he is actually engaging in metaphysics. He is not asking how to describe how a thing can be an automobile but how a thing can be. The comparisons of various paradigms naturally leads to the study of being itself. The computer programmer has to deal with not only the problem of deciding into what category the thing he is interested falls but also with the meta-problem of how to categorize the categorizations; how to pick between the various ways of abstracting and how to apply them to the ontology that is the world.

Farley raises another point that is worth pondering. He believes that human beings are intrinsically classifying animals, even if he doesn’t phrase it quite that way. For that reason, he believes that the object-oriented paradigm is more naturally matched to human thought. However, his definition of object oriented programing as the marriage of data and behaviors with polymorphism as the constraint agent (each object owns its specific response to a common message like fall) does lead to some difficult ontological questions when modeling the physical world. Does a rock know how to behave under the influence of gravity because it knows how to interpret the message sent from the Earth or does the Earth know how to make the rock fall because of a message that the rock sends, or is the message loop (i.e. the gravitational interaction) the key point? Even though an exploration of this question is beyond the scope of this post the ability to pose this question shows the power of thinking about paradigms as constraints.

It is for this reason that I think programming will becoming increasingly more and more important component of our day-to-day lives; not just because of the practical applications involving our smartphones and tablets, our X-boxes and Play Stations, and our computers working to solve this problem or that, but because it will trigger in us a way to look at the world anew. If only Aristotle could have lived to participate in these debates.