One of the things I adore about the field of software testing is how intertwined it is with that of philosophy. Substantial discussion of testing very quickly segues into such philosophical domains as aesthetics, epistemology, and ethics.

This may at first seem like a surprising collision. Philosophy? Isn’t that ivory-tower stuff, where bearded academics argue endlessly over trees falling in the forest? Testers, on the other hand, live in the metaphorical trenches of the software industry, pressing our faces up against the code, scrutinizing it for flaws, and racking our brains for ways to coax the software we’re testing into giving up its secrets. What could such a technical field possibly have in common with one that’s firmly seated amongst the humanities?

Inquiry is the core of software testing—inquiry into how the software behaves, inquiry into how people believe it should behave, and, hopefully, inquiry into how best to inquire. Philosophy is the archetypical process of inquiry. It provides both the framework for how we think of inquiry and the means we use to carry it out. If we are to refine the way we go about our work, then the tools of philosophical analysis are indispensable.

**Branches of Philosophy**

There is no particularly neat way to divide the field of philosophical inquiry; a great deal of overlap occurs regardless of how the lines are drawn. That being said, one classical set of divisions runs through the following branches: ethics, metaphysics, epistemology, logic, and aesthetics.

**Ethics**

Ethics addresses questions of morality. What actions are right and wrong? By whom can one do right or wrong? By what standard should we evaluate the morality of choices or behaviors? Given a choice between two actions both damnable, what should we do?

Ethical questions often come into play in software testing, and unfortunately they’re not always given the consideration they deserve.
As testers, what should our ethical code be? What are our responsibilities to the owner of the software under test? To the software’s users? To our teammates? While it’s easy to pontificate on what the right thing to do is, living up to that standard is another thing entirely.

**METAPHYSICS**

Metaphysics is the study of reality and what is real. On the face of it, this seems laughable. Don’t we all take reality for granted? How would we go about our day if we didn’t believe in the existence of the world we perceive? However, metaphysics also delves into things that are not so readily observable, like causality, ontology (the study of categories), and possibility.

If the software exhibits a bug during testing, how do we know our actions precipitated it and not something else? When is it appropriate to classify features, users, or elements of a design as similar? If a risky and untested change is put into production but nothing goes wrong, is there still any basis for arguing that it was a bad idea?

**EPISTEMOLOGY**

Epistemology is the study of knowledge and knowing. What is knowledge? How can we tell what we know and what we don’t? Is observation a valid way of acquiring knowledge? What about reason or feelings?

Since testing is, at its core, a practice of acquiring knowledge about how a piece of software works, some of these questions may sound very familiar. When do we stop testing, knowing that we know enough about the state of this piece of software to do so? How can we separate what we know about the software from what we merely believe?

**LOGIC**

We think of logic as being the branch of philosophy most closely related to the field of software. Formal logic—mathematical logic in particular—dictates how hardware gates, software control structures, and database queries work. It gives us tools like truth tables and Boolean algebra, which we can then use to analyze possibilities, decisions, or the flow of a program.
According to the extract provided, the article discusses the importance of aesthetics in software development and how it can make things easier for users. The text mentions that programs that are well organized and straightforward to understand are better than impenetrable spaghetti code. Interfaces that are clean and easy to use are preferred over those that are cluttered and require contortions on the part of the user. Study of aesthetics helps one discover and articulate consistent principles behind subjective worlds of art and music. The article argues that our work should exhibit beauty, elegance, or value—judgments that are always to some degree subjective.

The text also discusses the overlap of ideas between software testing and the philosophic fields of ethics, metaphysics, epistemology, logic, and aesthetics. It encourages readers to consider the importance of informal or persuasive logic and argues that our work should be more than just technical, mechanical, and mathematical; it should have a human touch and be more than impenetrable spaghetti code.

The article concludes by encouraging readers to think more about the human and organizational side of making software as formal logic is to interfacing with computers.

This article is the first installment in a series exploring the overlap of ideas between software testing and the philosophic fields of ethics, metaphysics, epistemology, logic, and aesthetics. Visit TechWell.com to read Rick Scott’s other articles in the series.