CrossTalk would like to thank NAVAIR for sponsoring this issue.

So where does one start when writing about the Immutable Laws of Software Development? As I often do, I went right to my friends at Wikipedia to understand "law" itself and came up with these initial thoughts. First, is it possible, or even desirable, to define law? After all, law is a term that does not have a universally accepted definition. In the broad legal world of international, constitutional, and criminal law, to name a few, it is generally a system of rules and guidelines enforced through social institutions to govern behavior.

When I think about how this definition extends to software, I see the need to transition from philosophically based laws of history (including great thinkers like Plato and Aristotle), to laws where data and observation are combined with documented processes and project roles. A good illustration of this is a phrase usually credited to W. Edwards Deming: "In God we trust; all others must bring data." Capers Jones, just this year, put together a short paper describing many of the laws of software development captured over the last 60 years. In almost all of them there is a reference to large quantities of empirical data from many projects. It is the lasting nature of these laws, in the very fluid world of software development, that lead us to the idea that software laws must be empirical.

As a Team Software Process (TSP) coach I have applied the teachings of Watts Humphrey for nearly 20 years. Much of what Humphrey brought together in the TSP was not revolutionary but rather a gathering of many laws of software engineering from other experts over previous decades. Starting with his experiences and data, I have applied laws such as: the larger a component, the longer it will take to build; project schedules are based on the total estimated hours combined with team members' availability; early defect detection will help schedules remain true and ensure the project will deliver low defect products to the end user. Sources of these software laws come from famous work such as "Quality is Free" by Phil Crosby, "Software Engineering Economics" by Dr. Barry Boehm, and "The Mythical Man-Month" by Fred Brooks.

For these laws to be considered immutable means they are not susceptible to change. While we will continue to go forward with the application of this proven body of work, we must always remain open to change through analysis of data. So keep collecting data, doing postmortem analysis, and evolving these laws as we continue to close this loop.

I welcome you to this issue of **CrossTalk** and invite you to enjoy and benefit from these great articles.

Jeff Schwalb
NAVAIR Process Resource Team