Abstraction --- the Hard Core of Software Engineering

Dewayne Perry
ECE, University of Texas, Austin

perry@ece.utexas.edu

Abstract

Modularity, encapsulation and abstraction are the primary intellectual tools for managing complexity in software systems, but the greatest of these is abstraction. Finding the right, or most appropriate, abstractions is the most important part of engineering software systems: they provide understanding; they provide the right veneer over complex underlying implementations; they provide the basic vocabulary (data and operations; nouns and verbs) for each layer of our virtual machines to express the solutions to the problems to solve. They are the fundamental job of software engineering.

Indeed, abstraction is the fundamental job of software engineering research as well: finding the right abstractions for software engineers to use in their quest for the right abstractions to solve their problems. Structured Programming is one such "right" abstraction for programming: it provides precisely the right focus on what is critical to write the simplest, understandable programs whose static structure provides us with useful clues and understanding about their dynamic structure.

I will explore how various abstractions have affected the way we engineer software systems. One of the most important insights recently is one due to our honoree, Prof. Wlad Turski, made in a key note address several years ago in which he distinguished between programs as computations and programs as behaviors. These two abstractions provide a keen and fundamental distinction and a deep understanding of different views about programs and systems, and clarify much of the confusion about differing and often conflicting views of software engineering.