Reading 2 Response

Liam Strand

Briefly describe one important lesson about modularity that the paper describes and that you think is still relevant today.

The paper emphasizes that it is important to decompose a program into modules in an intelligent and thoughtful way. The first instinct is often to decompose like a flowchart (A $\rightarrow$ B $\rightarrow$ C), but this strategy is usually less helpful than expected. It is usually a better idea to have each module focus on a single overarching problem or decision. This way, if the modules are (as they should be) designed to hide their implementations from their clients, changes and improvements to the solutions to these problems can be made safely and confidently.

Computing has advanced significantly since this paper was written. Briefly describe one challenge (not necessarily with modularity) or perspective that, while realistic in 1972, does not apply today.

The author ascribes a great cost to "procedure call overhead". Today, machines are much, much faster and have much more memory and larger caches. Whereas older machines often have to go to main memory (or worse, to the disk) to load instructions after a distant procedure call, modern machines can intelligently cache more of these distant instructions and very very rarely need to go to the disk to retrieve them.