Introduction

This is my first experiment with blogging about computing.

I have been writing for about one year and am posting many topics at once in no particular order.

I suggest that you examine the list of topics first, and choose articles that interest you.

I try to write in a personal style.

I use hyperbole and controversy liberally as a way to underline ideas. Don't believe everything I say, without further introspection.

My intention is to report on what I see and, with hope, get the reader to stop and to think about the various topics I discuss.

I try to simplify. I leave out details. Details kill.

First comes breadth of understanding, then comes depth of understanding. Details can be filled in later. I believe that the details are important, but need not be addressed in the first pass through. Divide & conquer also applies to writing, reading, understanding.

Main Points

1. Multitasking is easy, if you remove time-sharing and memory-sharing.
2. Sequencing of programs is part of design.
3. DI (Design Intent) is important.
4. The word "Architecture" is diluted and has many meanings.
5. Concurrency is not Parallelism.

6. Diagrams should be considered part of PL[1] syntax. I call this DaS.[2]
7. Flat <*anything*> will not scale.[3]
8. Text-only thinking has caused accidental complexity in the design of PLs, e.g. *exceptions* are not exceptional, *parameter lists* are only one way to transfer data to a routine, *return* values are only one way to transfer result data to the caller, etc.
9. Loop is the exception, not the rule (this includes recursion). Loop syntax limits our thinking about distributed system designs.
10. Concerns for memory sharing, time-sharing, etc., have caused accidental complexity. Newer computing hardware is no longer limited by concerns for memory recycling and processing power. PL design has not kept up with such advances in computer abilities.
11. I believe that multiple paradigms should be used in practice when solving real problems.
12. I highly endorse the use of multiple DSLs to solve real problems.
13. Engineering is not coding. Coding is not Engineering.
14. Certain languages have a plethora of features. We should treat these languages as *assembly languages* and build new-breed "HLLs" with them.
15. The Divide and Conquer paradigm is under-used.
16. Simplicity is hard.[4]
17. Systems built on a single paradigm are bound to fail.[5]

Comments Section

I haven't settled on a way to allow comments in my blog.

---

[1] Programming Language

[2] Diagrams as Syntax.

[3] That includes GOTOs, global variables, global types, global functions, global namespaces, etc.

[4] Corollary: complicated solutions are a cop-out. Complicated solutions should not be rewarded through acceptance and use. TC;DU - too complicated, didn't use.

[5] N.B. FP (Functional Programming) is a paradigm. A single paradigm.

TC;DU - too many options.

I need shuhari regarding blog comments.

For the time being, send email to ptcomputingsimplicity@gmail.com.

Maybe I will duplicate some comments in my posts, or, maybe I will summarize comments, or maybe I will ignore them, or, maybe someone will suggest the perfect comment gathering solution that fits my particular set of biases…