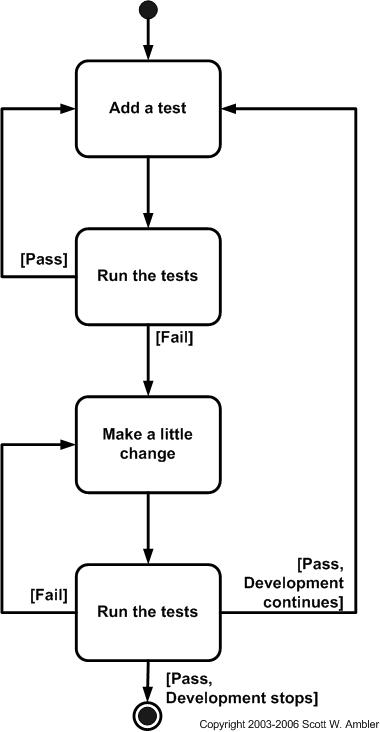
# [Clean Architecture Book Notes]

## [Chapter 1]

* The word “architecture” is often used in the context of something at a high level that is divorced from the lower-level details, whereas “design” more often seems to imply structures and decisions at a lower level. But this usage is nonsensical, and so it is with software design, The low-level details and the high-level structure are all part of the same whole.
* TDD by Jason Gorman: Test-Driven Development is a methodology in software development that focuses on an iterative development cycle where the emphasis is placed on writing test cases before the actual feature or function is written.It combines building and testing.( adapted by Agile S.D)



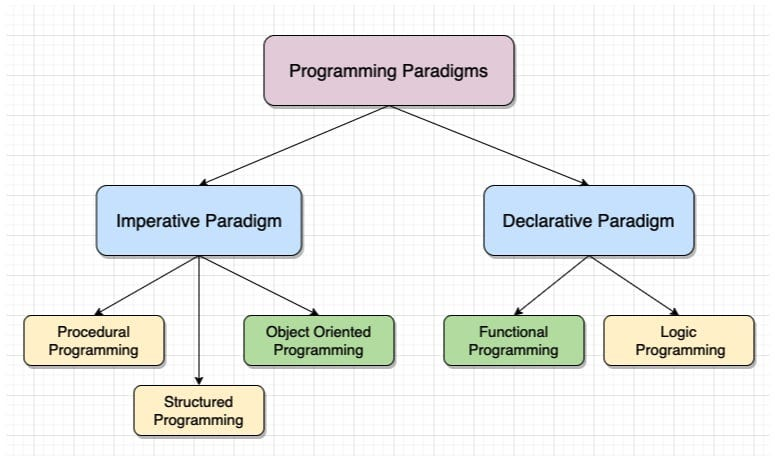
## [Chapter 2]

* Every software system provides two different values to the stakeholders: (behavior and structure).
* BEHAVIOR: machines behave in a specific way.
* ARCHITECTURE (structure) has to do with the word “software”—a compound word composed of “soft” and “ware.” The word “ware” means “product”; the word “soft” that’s where the second value lies, Software was invented to be “soft.” It was intended to be a way to easily change the behavior of machines.
* The first value of software—behavior—is urgent but not always particularly important.
* The second value of software—architecture—is important but never particularly urgent.
* Software architecture revolution was in programming paradigms, Paradigms are ways of programming, relatively unrelated to languages, A paradigm tells you which programming structures to use, and when to use them.

## [Chapter 3]

- Programming Paradigm [strategy, style followed while implementation]

* STRUCTURED PROGRAMMING: replaced the unrestrained jumps with if/then/else and do/while/until constructs. (discipline on direct transfer of control.)
* OBJECT-ORIENTED PROGRAMMING: The function became a constructor for a class, the local variables became instance variables, and the nested functions became methods. This led inevitably to the discovery of polymorphism through the disciplined use of function pointers. (discipline on indirect transfer of control.)
* FUNCTIONAL PROGRAMMING: a paradigm of building computer programs using expressions and functions based on mathematical functions without mutating state and data.(discipline upon assignment.)



## [Chapter 4]