Clean Architecture

Chapter 3. 패러다임 개요
Chapter 4. 구조적 프로그래밍

3

패러다임 개요

패러다임 개요

구조적 프로그래밍(Structed Programming)

최초로 적용된 패러다임

데이크스트라가 goto문은 프로그램 구조에 해롭다는 사실을 제시
이러한 점프문들을 if/then/else와 do/while/until과 같은 구조로 대체

구조적 프로그래밍은 제어흐름의 직접적인 전환에 대해 규칙을 부과한다.

패러다임 개요

객체 지향 프로그래밍(Object-oriented Programming)

ALGOL 언어의 함수 호출 stack frame을 heap으로 옮기면,
함수 호출이 반환된 이후에도 함수 지역 변수가 오랫동안 유지될 수 있음

이런 함수가 클래스 생성자, 지역변수가 인스턴스 변수, 중첩함수가 메서드

함수 포인터를 특정 규칙에 따라 사용 → 다형성 등장

객체 지향 프로그래밍은 제어흐름의 간접적인 전환에 대해 규칙을 부과한다.

패러다임 개요

함수형 프로그래밍(Functional Programming)

Alonzo Church가 발명한 람다(lambda) 계산법이 LISP 언어의 근간

람다 계산법의 기초 개념은 불변성(immutability)

함수형 프로그래밍은 할당문에 대해 규칙을 부과한다.

패러다임 개요

각 패러다임은 ' 무엇을 해서는 안 되는지' 말해준다.

구조적 프로그래밍 : goto문

객체 지향 프로그래밍 : 함수 포인터

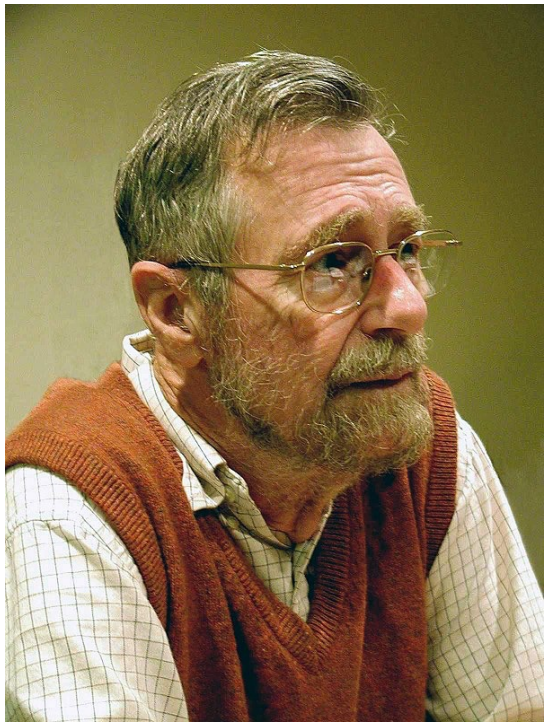
함수형 프로그래밍 : 할당문

프로그래밍 패러다임은 앞으로도 위의 세 가지밖에 없을 것이다!

4

구조적 프로그래밍

Structed Programming



데이크스트라

프로그래밍은 어렵고, 프로그램은 인간의 두뇌로 감당 못함

유클리드 계층구조 방식을 사용하여 증명하자

goto문이 재귀적으로 분해하는 과정에서 방해됨

if/then/else와 do/while과 같은 분기와 반복으로 바꾸자!

Structed Programming

모든 프로그램은 순차, 분기, 반복 구조로 표현할 수 있다. (by 뵘, 야코피니)

데이크스트라는 열거법과 귀납법을 사용해 프로그램 입증

1968.03 CACM에 실린 "goto문의 해로움" 서한은 계속 논쟁이 됨

컴퓨터 언어가 진화하면서 goto문이 거의 사라졌음

우리 모두는 구조적 프로그래머

Structed Programming

하지만 끝내 프로그램 관점에서 정리한 유클리드 계층구조는 만들어지지 않음

과학적 방법

과학 이론과 법칙은 그 올바름을 증명할 수 없다.

예를 들어, F=ma는 증명할 수 없다. 언젠가 다른 실험을 통해 이 법칙이 잘못됐음이 밝혀질지도

각고의 노력으로도 반례를 찾을 수 없다면, 그 서술은 참이라고 본다.

Structed Programming

"테스트는 버그가 있음을 보여줄 뿐, 버그가 없음을 보여줄 수는 없다."

테스트에 충분한 노력을 들였다면, 그 프로그램은 충분히 참이라고 여기자

소프트웨어 아키텍트는 모듈, 컴포넌트, 서비스가
쉽게 반증 가능하도록(테스트하기 쉽도록) 만들기 위해 노력해야 한다.