

EECS4314 week 5

Jerry Wu

2024-02-07

Contents

1	Design patterns review	3
1.1	Evolution of programming abstractions	3
1.1.1	Abstractions from developers perspective	3
1.2	What are design patterns? (OODP)	3
1.2.1	Learning design patterns	4
1.3	Why design patterns?	4

Section 1

Design patterns review

1.1 Evolution of programming abstractions

- First modern programmable computers in the 1950s were largely hardwired (first software was written in machine code)
- Assembly introduced through symbolic assemblers, macro processors, etc
- In the 1960s high level languages were created (FORTRAN, COBOL, C, etc.)
- HL languages were independent of machine and problem domain

1.1.1 Abstractions from developers perspective

- Typed variables and user defined types created in late 1960s
- Modules created in early 1970s
- ADTs and OOP created in mid 1970s
- OOP design patterns, refactoring in 1990s

1.2 What are design patterns? (OODP)

- Design patterns are reusable solutions to common problems
 - An OODP involves a small set of classes cooperating to achieve a desired end
 - Done via adding a level of indirection in some clever way

- New solutions provide the small functionality as an existing approach but in some more desirable way in terms of elegance, efficiency, and adaptability
- OODPs make heavy use of interfaces, information hiding, polymorphism, and intermediary objects
- Typical presentation of an OODP
 - A motivating problem and its context
 - Discussion of the possible solutions
 - Common variations and tradeoffs

1.2.1 Learning design patterns

- Think of OODP as high level programming abstractions
 - First, learn the basics (data structures, algorithms, tools and language details)
 - Then learn modules, interfaces, information hiding, classes/OOP
 - Design patterns are the next level of abstractions
 - Architecture

1.3 Why design patterns?