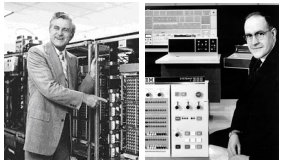
IBM System/360

- 1964-1978 Mainframe computer and peripherals
- Family of compatible low to high-end computers

Model 50 (1964) → IBM z15 (2019)

Gene Amdahl, Chief Architect (L)
Fred Brooks, Project Manager (R)



Computer architecture, 8-bit byte, 32-bit words, IBM floating-point arch., microcoded control, cache, virtual mem., system/prog. state, **software engineering, comp sci.**

1

Software Engineering

soft·ware
noun /ˈsôftˌwe(ə)r/

The programs and other operating information used by a computer

en·gi·neer·ing
noun /ˌenjəˈni(ə)riNG/

The branch of science and technology concerned with the design, building, and use of engines, machines, and structures

The work done by, or the occupation of, an engineer

The action of working artfully to bring something about

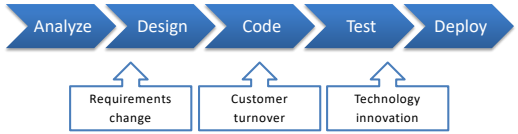
Google Dictionary

2

Coding is an important but small part

- Software development life cycles (processes)

Waterfall: rigid, high assurance



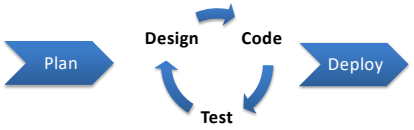
Incremental: less rigid, rapid development

3

Coding is an important but small part

- Software development life cycles (processes)

Agile: flexible, fast incremental development



- Scrum
  - Small team, time-boxed iterations (sprints)
- Extreme programming
  - Pair programming
  - Unit test everything
  - Continuous code review
  - Code only what's needed

4

Coding is an important but small part

- Software development processes
  - Waterfall/incremental/spiral
  - Agile/scrum/extreme programming
- Requirements, specifications: what
  - Complete, consistent (testable), and correct (user)
- Object-oriented design: how – abstract
  - Design patterns: recipes, not code reuse
- Code implementation: how – concrete
  - Debug: assert pre and post conditions, invariants
  - Release: system call return values (file i/o etc)
  - Unit tests: correctness, test driven development

5

Coding is an important but small part

- Performance analysis: how well
  - Code behavior: gprof, valgrind, hardware counters
- Code refactoring: better / cleaner
  - Goal: structure = maintainability, extensibility
- Integration/system testing: check
  - Black box, clear box, state/path/function coverage
  - Statistical usage model derived from requirements
- Project management: human elements
  - Cost / time estimate: mythical man-month
  - "Adding man-power to a late project makes it later"

6

Design patterns: Big picture

- Creational patterns
  - Handle object instantiation , decouple clients from objects
- Structural patterns
  - Change class/object interface, handle complex compositions
- Behavioral patterns
  - Handle class/object interaction, distribute responsibility
- Class vs object patterns
  - Class pattern: compile-time relationships based on inheritance
  - Object pattern: run-time relationships based on composition
- Compound patterns
  - Generic design pattern comprised of multiple patterns
  - Example: model-view-controller (GUI, web applications)

7

Design patterns: Common examples

Creational	Description
Singleton	Ensure class has just one instance and provide global access
Factory Method	Define an interface for creating an object but let subclasses decide which class to instantiate (deferred instantiation)
Abstract Factory	Provide an interface for creating families of related objects without specifying their concrete classes

Structural	Description
Adapter	Convert interface of a class into one clients can work with
Decorator	Attach responsibilities to an object dynamically
Composite	Compose objects into tree structures to represent part-whole hierarchies. Allows client to treat individual and composition objects the same way.

8

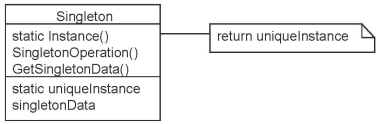
Design patterns: Common examples

Behavioral	Description
Strategy	Define family of algorithms, encapsulate each one, and make them interchangeable
Observer	Establish publisher-subscriber relation between objects
Template	Define skeleton of an algorithm in an operation, deferring some steps to subclasses
Iterator	Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation

Compound	Description
Model-view-controller	Decouple view (screen presentation), model (application) and controller (user interface) objects using Factory, Decorator, Observer, Composite, and Strategy design patterns

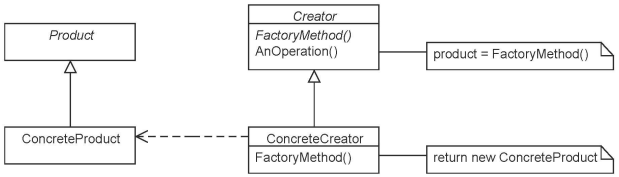
9

Singleton



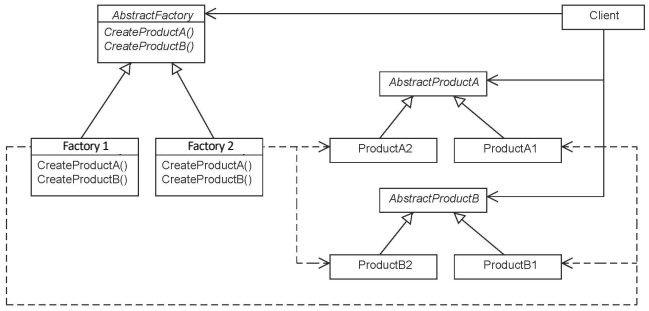
10

Factory Method



11

Abstract Factory



12