Software Design and Architecture

Hierarchical Architecture

Overview

- The hierarchical software architecture is characterized by viewing the whole system as a hierarchy structure.
- The software system is decomposed into logical modules (sub-systems) at different levels in the hierarchy.
- Modules at different levels are connected by explicit or implicit method invocations.
- A lower level module provides services to its adjacent upper level modules, which invokes the methods or procedures in lower level.

Main-Subroutine

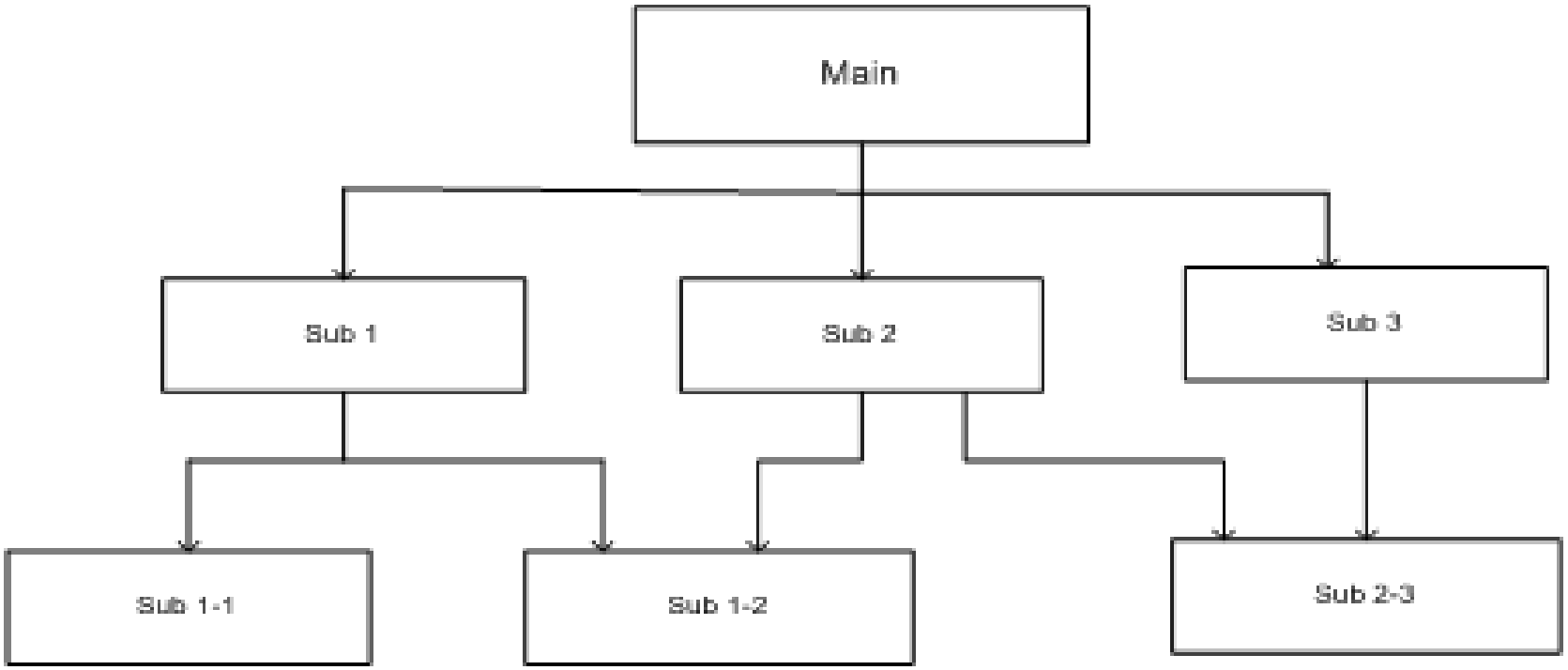
- The main-subroutine design architecture has dominated the software design methodologies for a very long time.
- The purpose of this architectural style is to reuse the subroutines and have individual subroutines developed independently.
- In the classical procedural paradigm, typically data are shared by related subroutines at the same level.
- With object orientation, the data is encapsulated in each individual object so that the information is protected.

- Using this style, a software system is decomposed into subroutines hierarchically refined according to the desired functionality of the system.

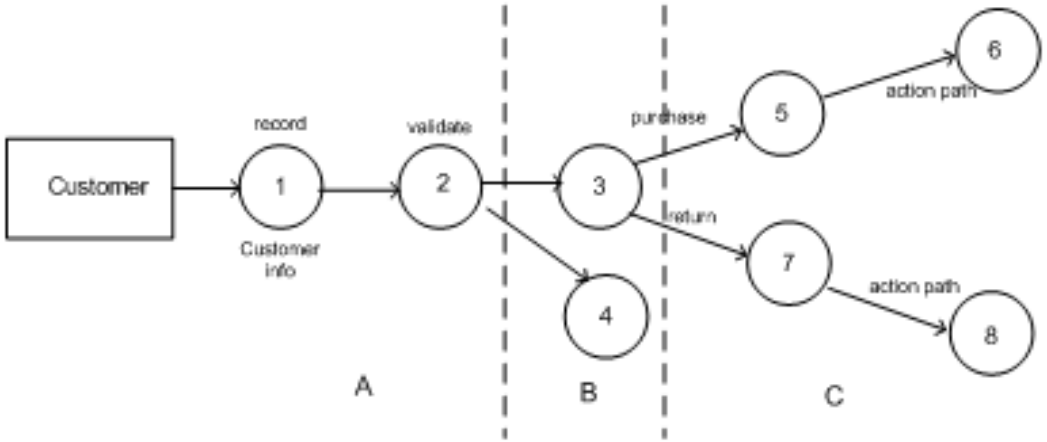
- Refinements are conducted vertically until the decomposed subroutine is simple enough to have its sole independent responsibility, and whose functionality may be reused and shared by multiple callers above.

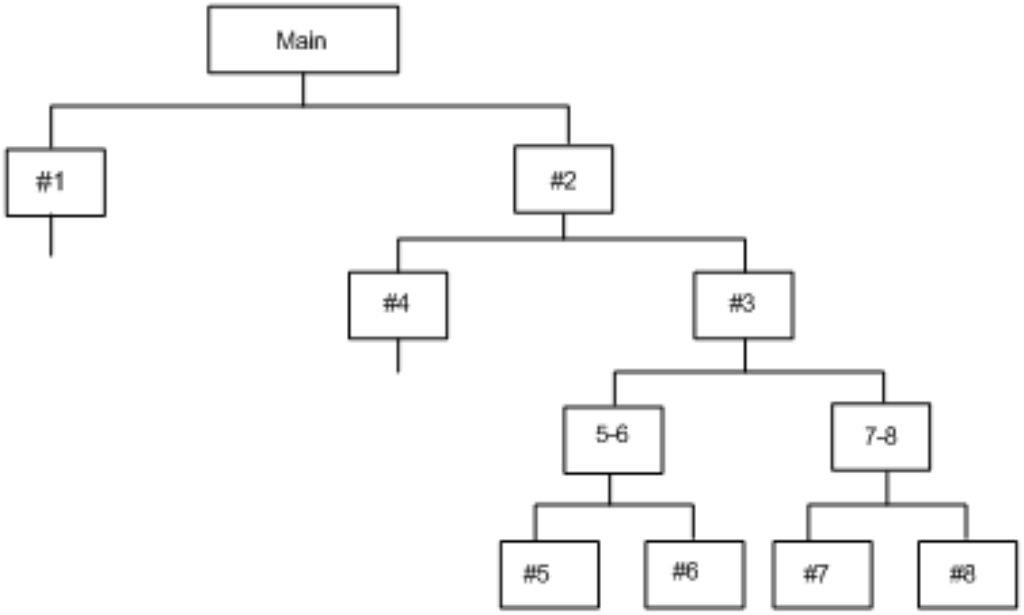
- Data is passed as parameters to subroutines from callers.

  There are many different ways to pass on parameter data:
- Passed by reference where the subroutine may change the value of data referenced by the parameter.
- Passed by value where the subroutine only uses the passed data but cannot change it.


Main
Sub 1
Sub 2
Sub 3
Sub 1-1
Sub 1-2
Sub 2-3

- A Data Flow Diagram (DFD) is often used to model the software requirement in this case, where bubbles or circles represent processing or activities and arrows represent data flow.


Customer
record
1
Customer info
validate
2
3
purchase
5
6
action path
return
7
action path
8
4
A
B
C


Main
#1
#2
#4
#3
5-6
7-8
#5
#6
#7
#8

Benefits

- Easy to decompose the system based on the definition of the tasks in a top-down refinement manner.
- This architecture can still be used in a sub-system of OO Design.

Limitations

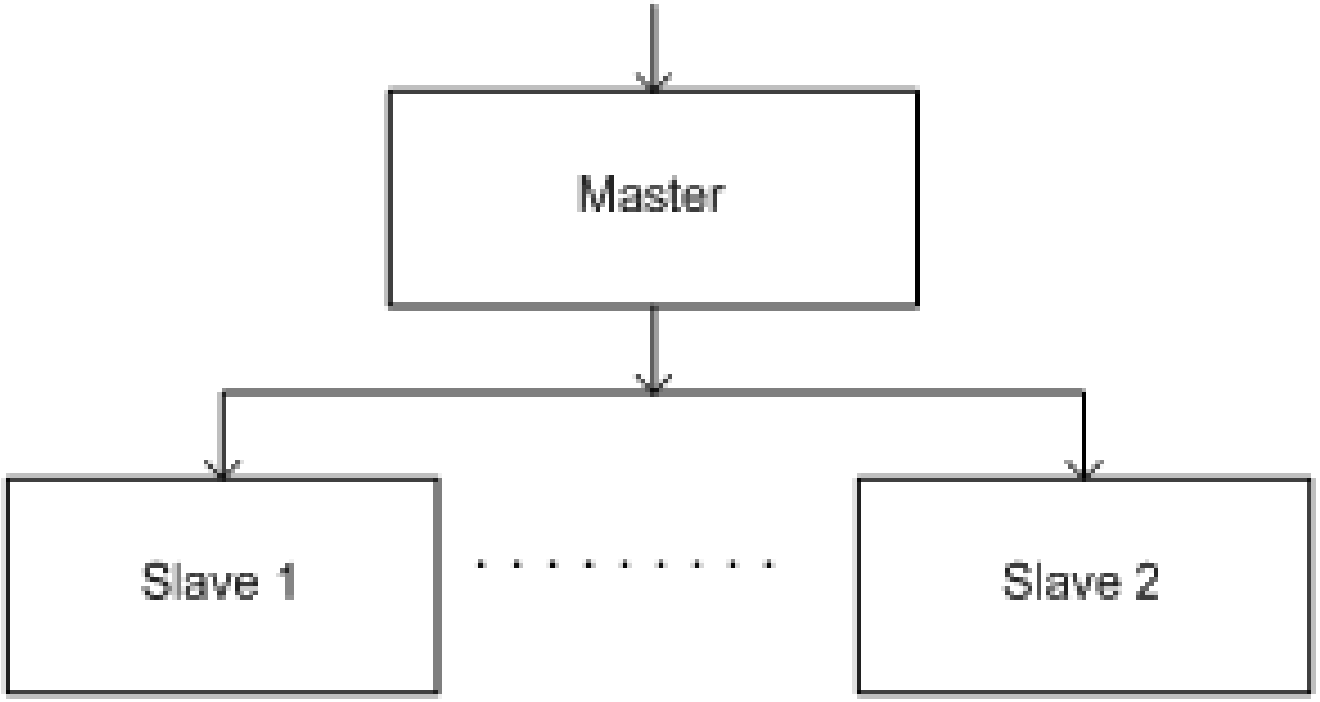
- Globally shared data in classical main-subroutines introduces vulnerabilities.
- Tight coupling may cause more ripple effects of changes as compared to OO Design.

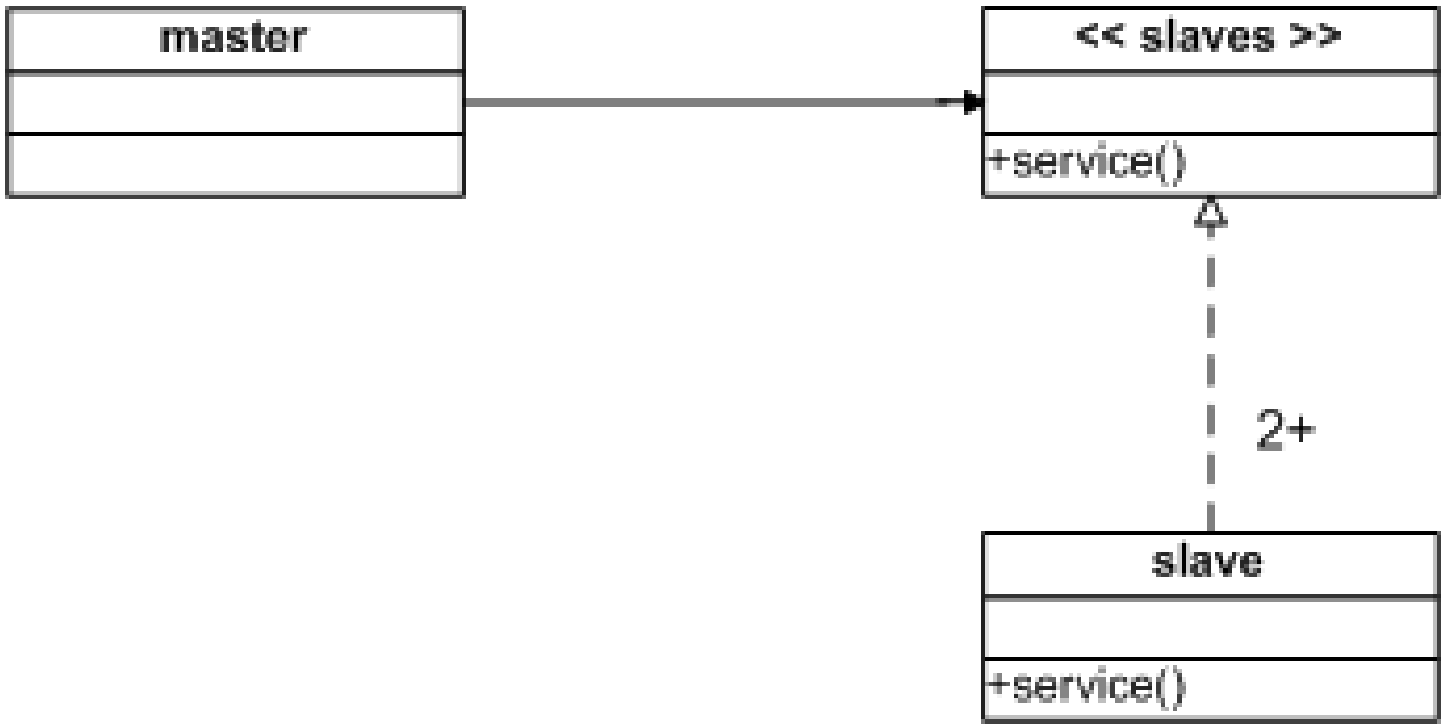
Master-Slave

- The Master-Slave architecture is a variant of the main-subroutine architecture style that supports fault tolerance and system reliability.

- In this architecture, slaves provide replicated services to the master, and the master selects a particular result among slaves by certain selection strategy.

- The slaves may perform the same functional task by different algorithms and methods or totally different functionality.


Master
Slave 1
Slave 2



Class diagram for Master Slave Architecture

Applicable Design Domains

- It is used for the software system where the reliability is critical.
- This is due to the replication (redundancy) of servers.
- If the master node is deleted then the slave node does not have any reason to stay.

Summary

- Introduction to hierarchical architecture
- Introduction to master slave architecture
- Introduction to main subroutine architecture