# 4. State Machine patterns

## Definition

Behavioural design pattern.

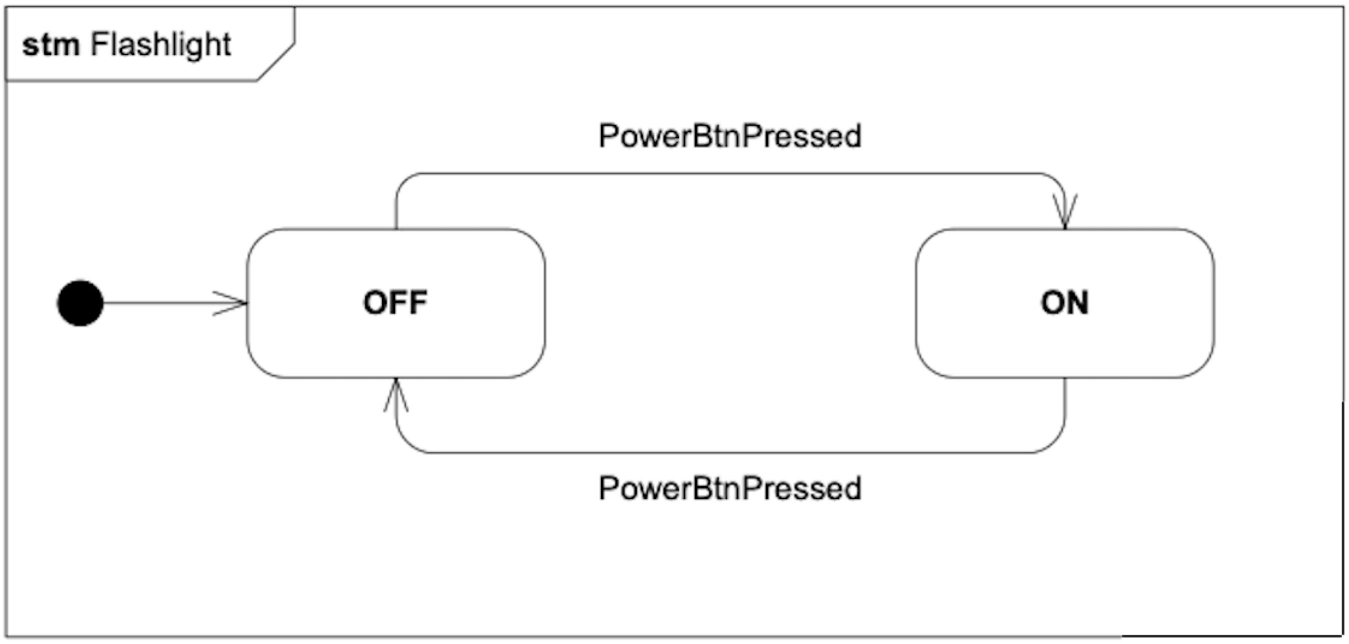
> **Definition:** A state machine is model with states that is in one of a finite set of states. The state machine can transition between states when event occur

## Different types:

- Switch/case
  - Good for small programs. Efficient.
- Table based
  - Runtime, Flexible, Logic and actions seperated.
- GoF State pattern
  - Combination (efficiency and flexibility)

## GoF State Pattern

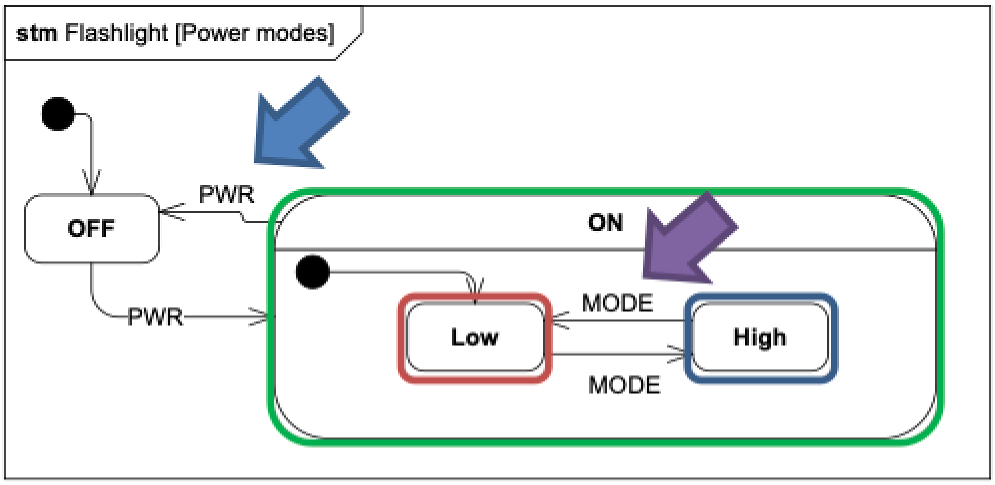
> **Definition:** The state pattern allows an object to alter its behaviour when its internal state changes. The object will appear to change its class



## Nested & Ortognal states

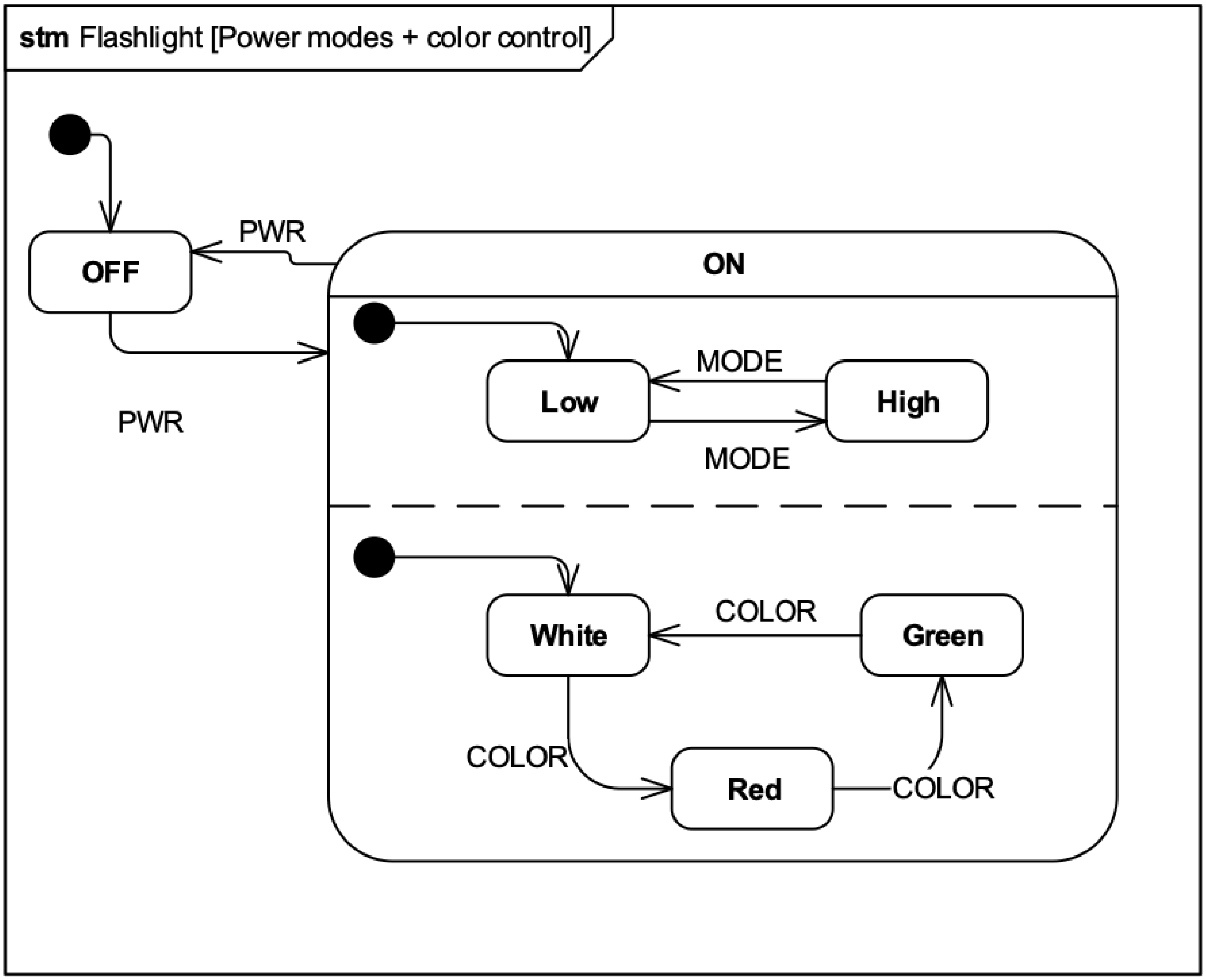
### Nested states

States within states



### Ortogonale states

Concurrent states.



## SOLID

**S** Seperates behaviour of an object into differenc states. Each state is responsible for handling specific behaviour.
**O** Ability to add more states without modifying exiting code.
**L** Any state object can be substituted with any other state object without affecting the correctness of the state machine.
**I** Adheres to ISP by defining a sperate interface which is responsible for the transisitions.
**D** Adheres to DIP by decoupling client from state implementations. The client interacts with an abstract interface, not relying on concrete implementations.

## Comparison

The State and Strategy Patterns same CD, but differ in intent.
Both encapsulate behaviour.
Strategy $\rightarrow$ client to select a specific behaviour at runtime, via an algorithm.
State machine $\rightarrow$ manages the behaviour of an object based on its state.