STATE MACHINE DIAGRAM

Budi Susanto

Tujuan

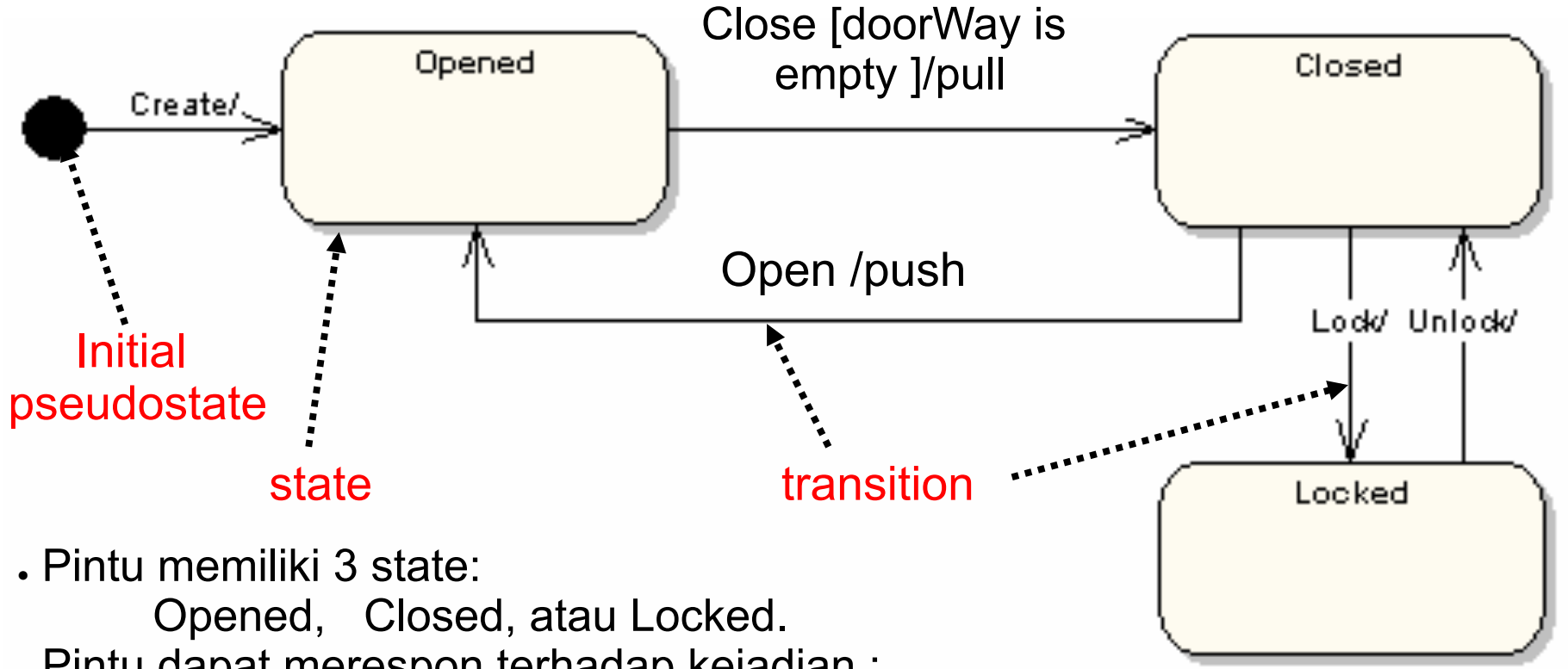
- Memahami peran yang diberikan dari state machine diagram
- Dapat memahami notasi-notasi yang digunakan dalam state machine diagram

Dasar State Machine Diagram

- Secara umum, State Machine Diagram adalah sebuah teknik untuk menggambarkan perilaku dinamis sebuah sistem.
- Dalam pendekatan OO, sebuah State Machine Diagram memodelkan perilaku dari **sebuah objek tunggal**, menunjukkan urutan kejadian yang terjadi pada sebuah objek selama hidupnya dalam merespone suatu event
- Elemen dasar adalah state dan transisi dari satu state ke state lainnya.

Dasar State Machine Diagram

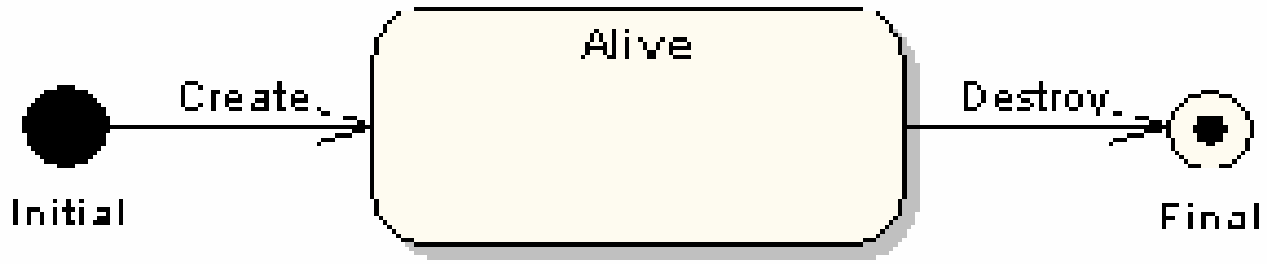
- Contoh 1. State Machine Diagram memperlihatkan state siklus hidup sebuah pintu



- Pintu memiliki 3 state:
  Opened, Closed, atau Locked.
- Pintu dapat merespon terhadap kejadian :
  Open, Close, Lock dan Unlock.

Elemen State Machine Diagram

- **State**: sebuah state ditandai dengan sebuah kotak dengan sudut tumpul dan terdapat nama state yang ditulis di dalamnya.
- **Initial** dan **Final States** (*pseudostate*) : Initial State ditandai dengan sebuah kotak terisi warna hitam penuh dan dapat diberi label. Final State ditandai dengan sebuah lingkaran dengan titik ditengahnya dan dapat diberi label.

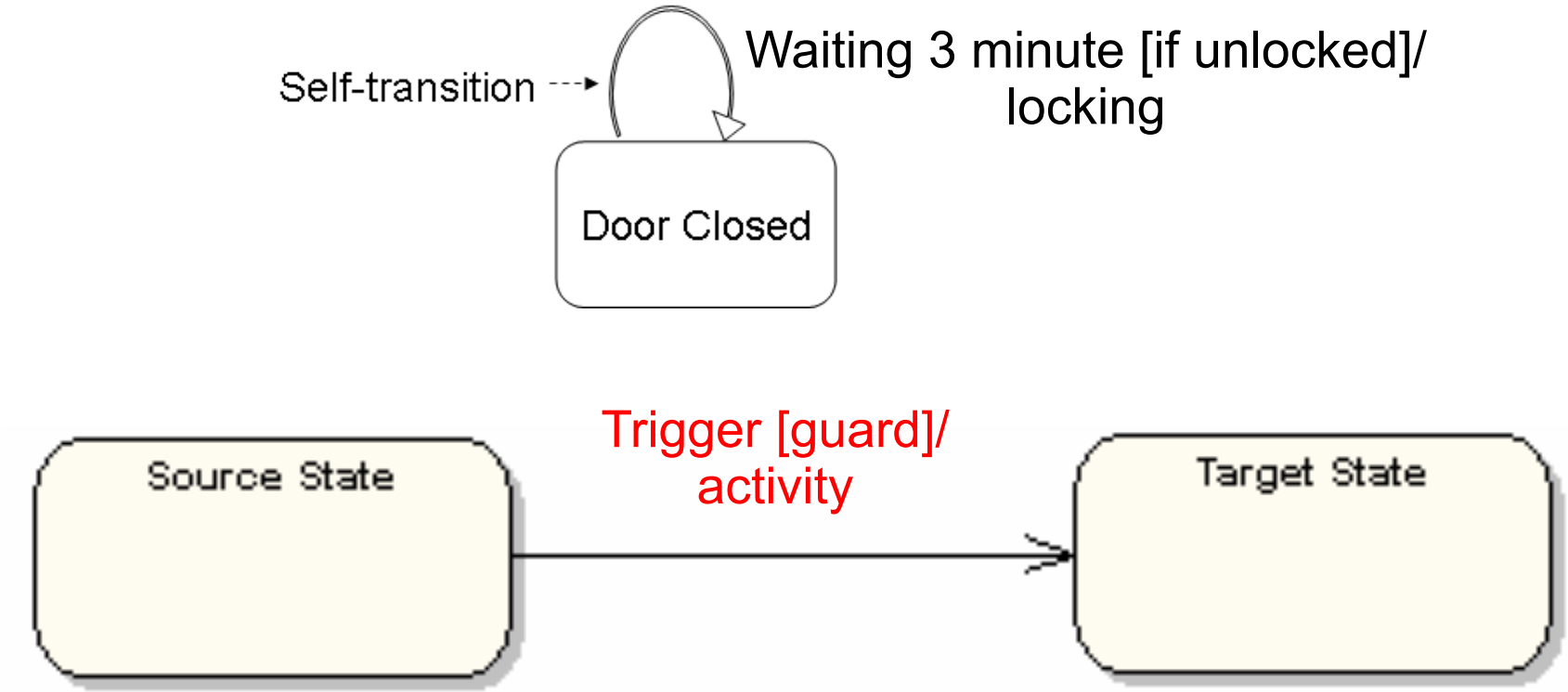


Elemen State Machine Diagram

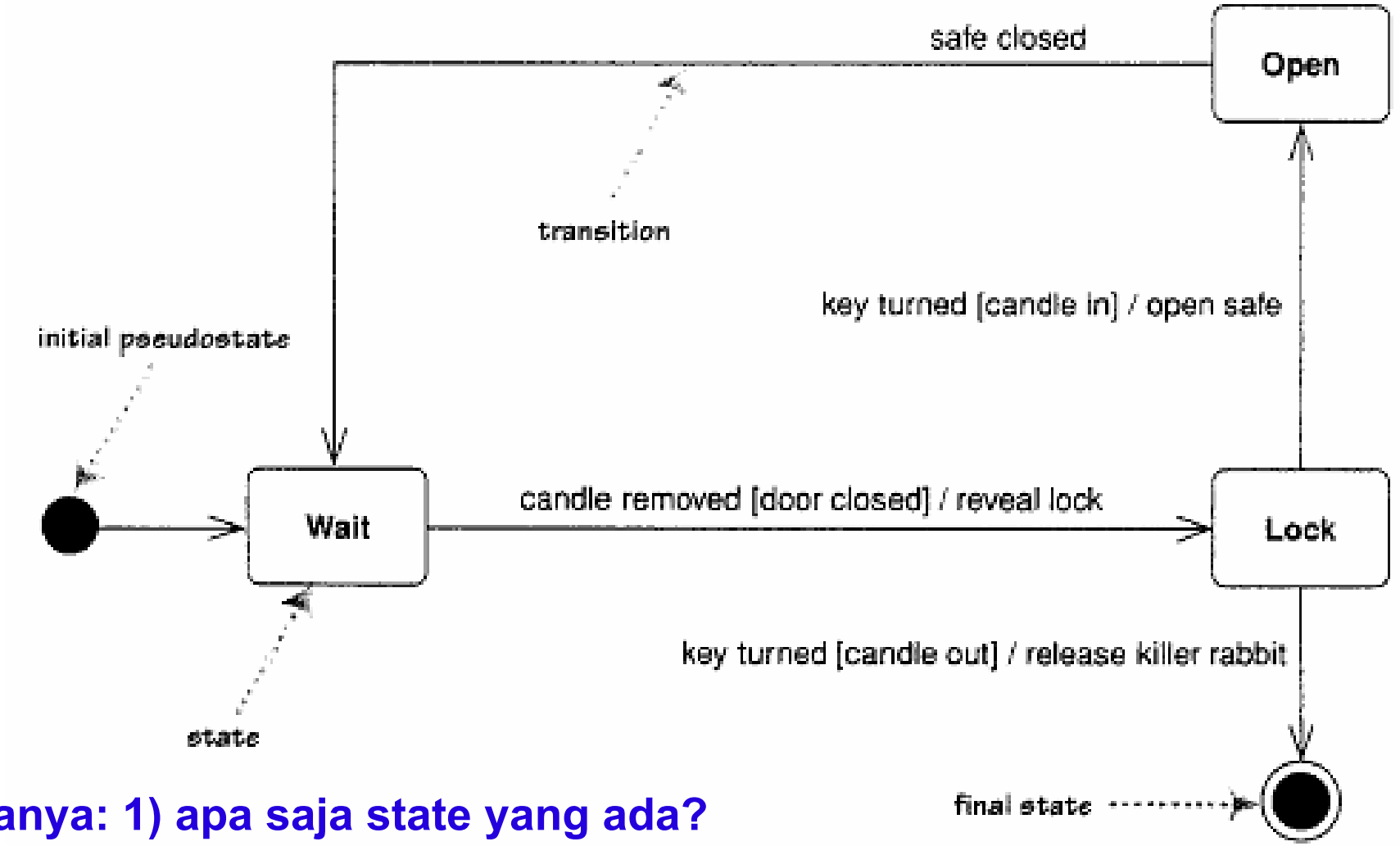
- **Transition**: sebuah transisi menyatakan perpindahan dari satu state ke state berikutnya dengan sebuah anak panah. Transisi memiliki label dalam 3 bagian: *trigger [guard]/activity*. Semuanya optional.
  - **Trigger**: sinyal kejadian yang memicu perubahan state
  - **Guard**: jika ada, sebuah kondisi Boolean harus true sehingga trigger menyebabkan transisi
  - **Activity**: beberapa perilaku yang telah dijalakan selama transisi

Elemen State Machine Diagram

- **Self-Transitions**: adalah sebuah transisi dimana state sumber dan sasaran sama.



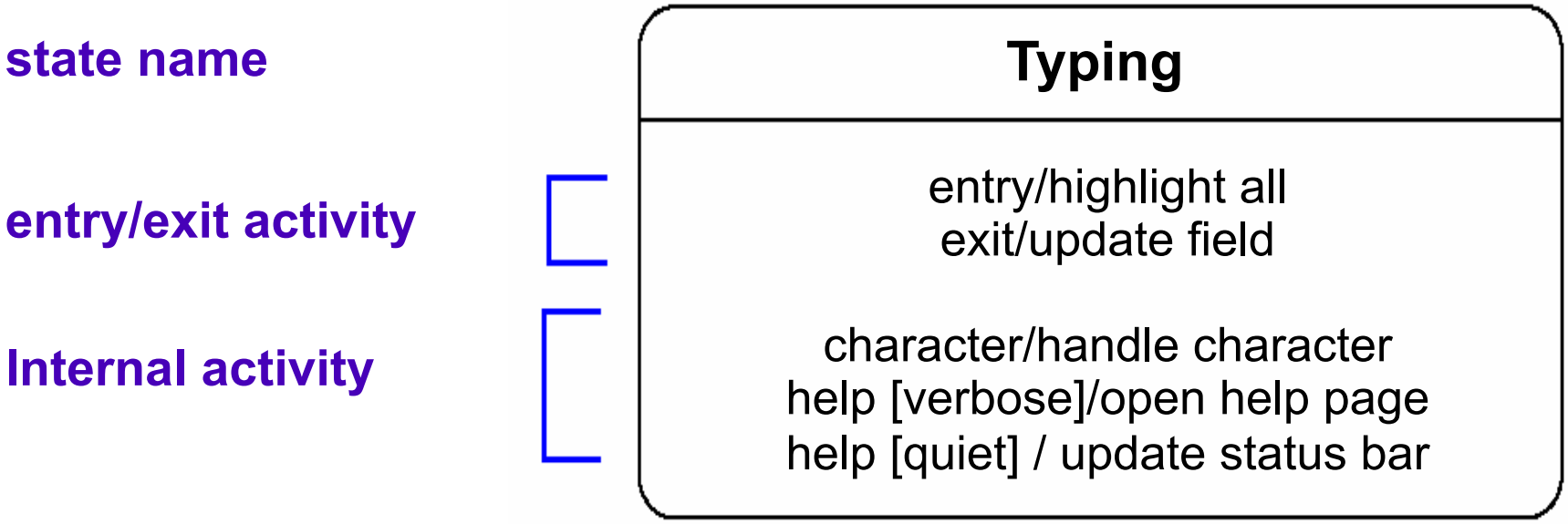
Contoh 2



Tanya: 1) apa saja state yang ada?
2) Apa arti dari transisi (candle removed [door closed]/ reveal lock) ?

Aktifitas Internal

- Untuk memperlihatkan kasus dimana state bertindak terhadap event tanpa transisi.
- Ditulis dengan meletakkan event, guard dan activity dalam kotak state itu sendiri.

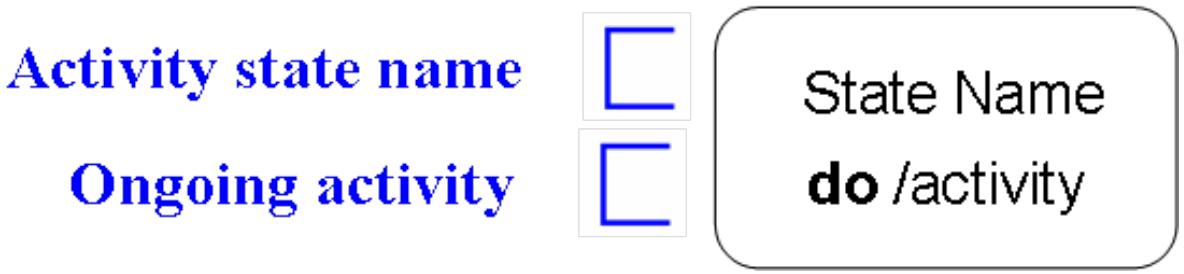


Aktifitas Internal

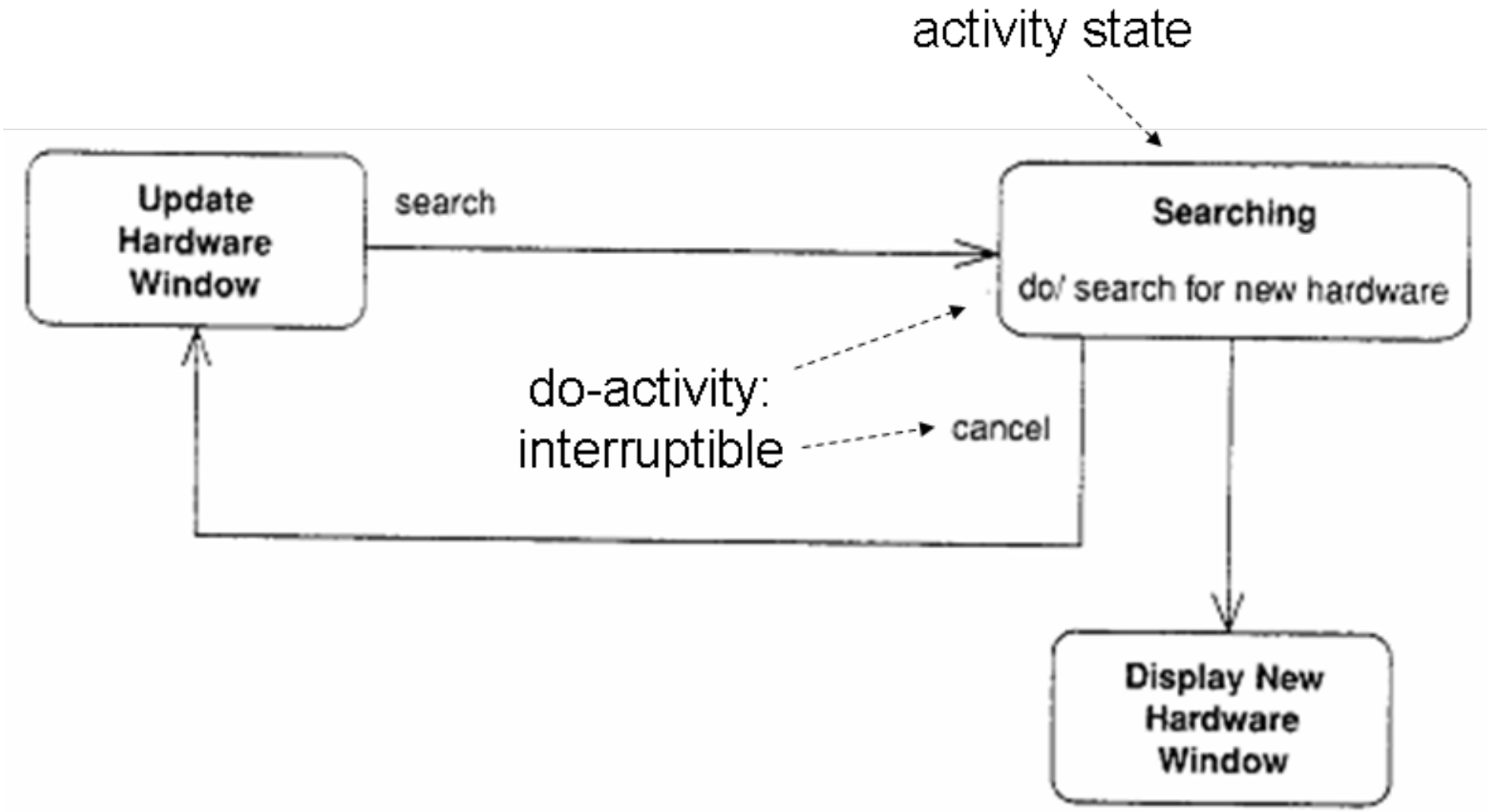
- *Entry activity*: aktifitas dijalankan ketika anda masuk ke sebuah state
- *Exit activity*: aktifitas dijalankan ketika Anda meninggalkan sebuah state
- Berbeda dari self-transition, dalam internal activitiy tersebut tidak memicu aktifitas masuk dan keluar.

Aktifitas Internal

- Regular vs Activity State:
  - Regular state : “diam” dan menunggu untuk kejadian berikutnya sebelum melakukan sesuatu
  - Activity state dapat melakukan sesuatu pekerjaan
- Perbedaan penting :
  - regular activities terjadi secara instan dan tidak dapat diinterupsi
  - Activity menggunakan waktu terbatas dan tidak dapat diinterupsi



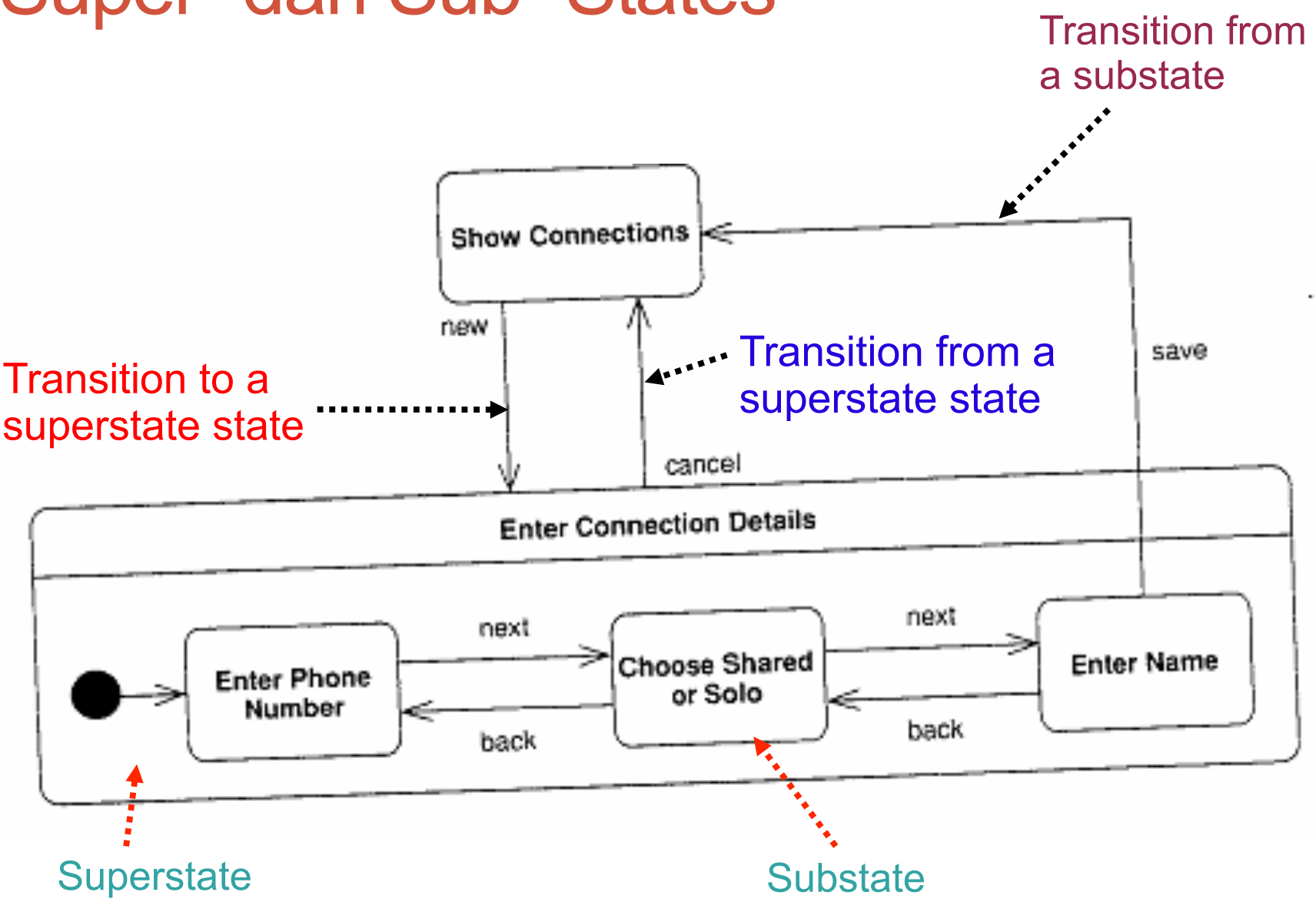
Contoh Activity State



Super- dan Sub- States

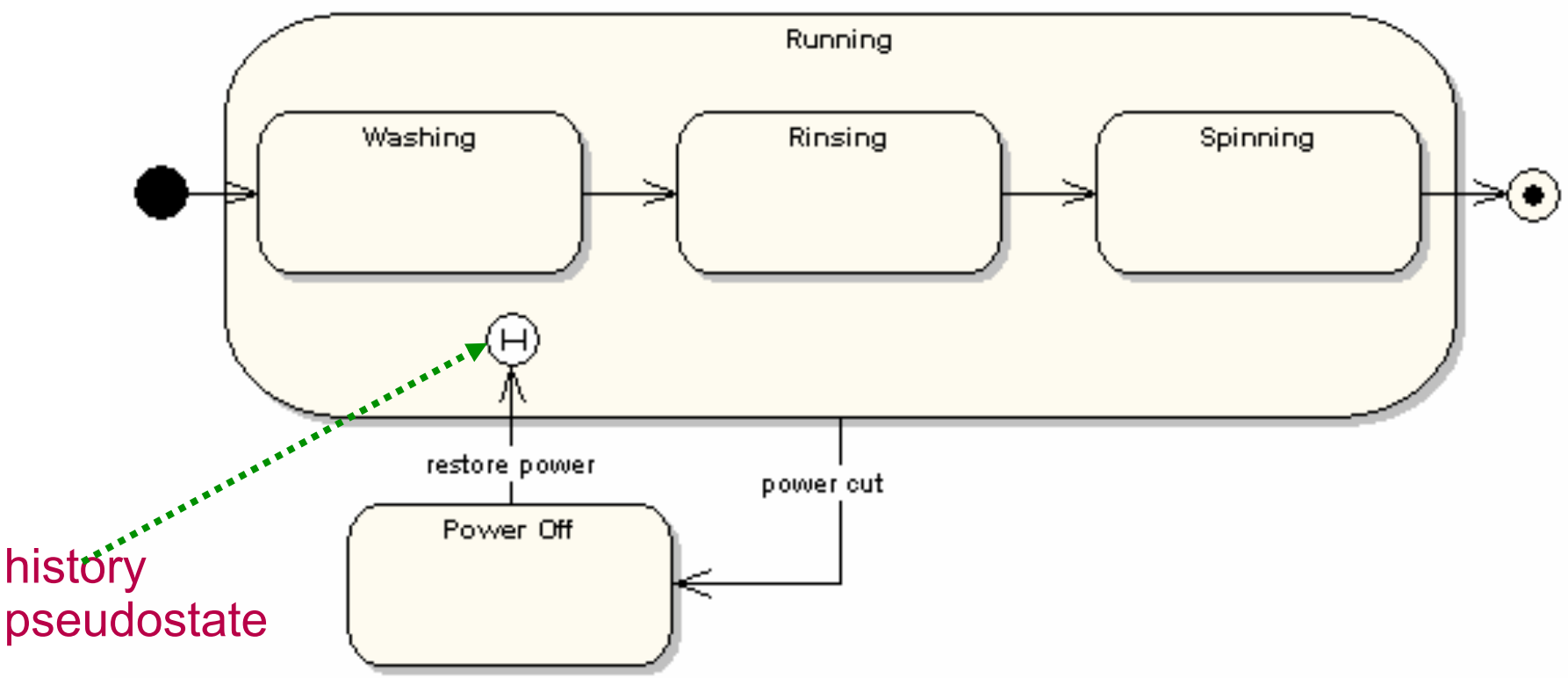
- Sebuah **substate** adalah sebuah state yang dirangkai dengan state lain
- Sebuah state yang memiliki **substates** disebut Superstate (atau composite state)
- Aggregation View: ketika beberapa state berbagi transisi umum dan internal activity, mereka dapat dibuat substate dan dipindah dari berbagi perilaku menjadi sebuah superstate.
- Decomposition view: ketika sebuah state terlalu komplek, terkadang dapat diuraikan menjadi beberapa substate untuk mencapai pemahaman yang lebih baik terhadap perilakunya.

Super- dan Sub- States



History States

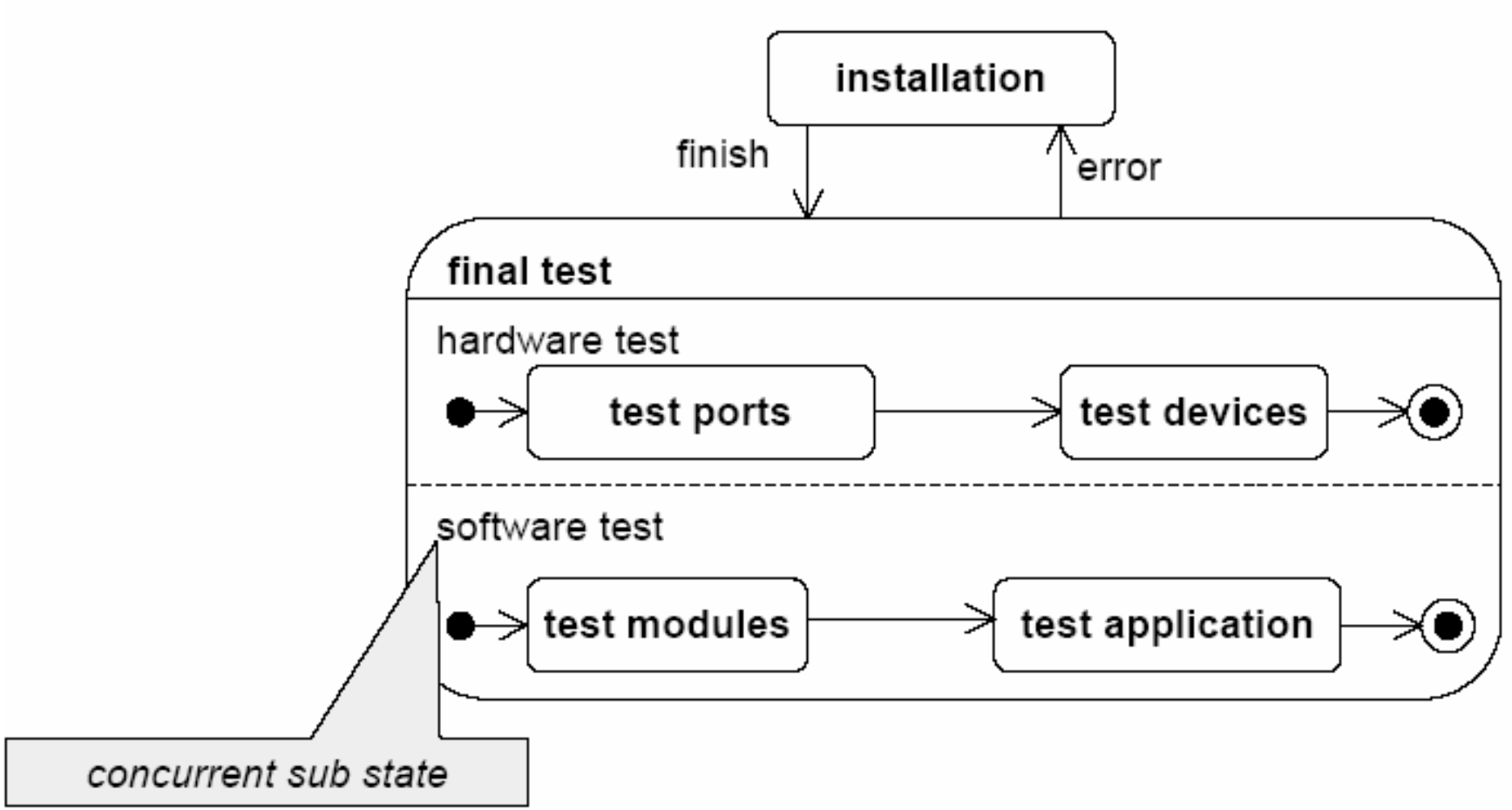
- Sebuah history State digunakan untuk mengingat state sebelumnya dari sebuah state machine ketika diinterupsi.



If there is a power cut, the washing machine will stop running and will go to the Power Off state. Then when the power is restored, the Running state is entered at the History State symbol meaning that it should resume where it last left-off.

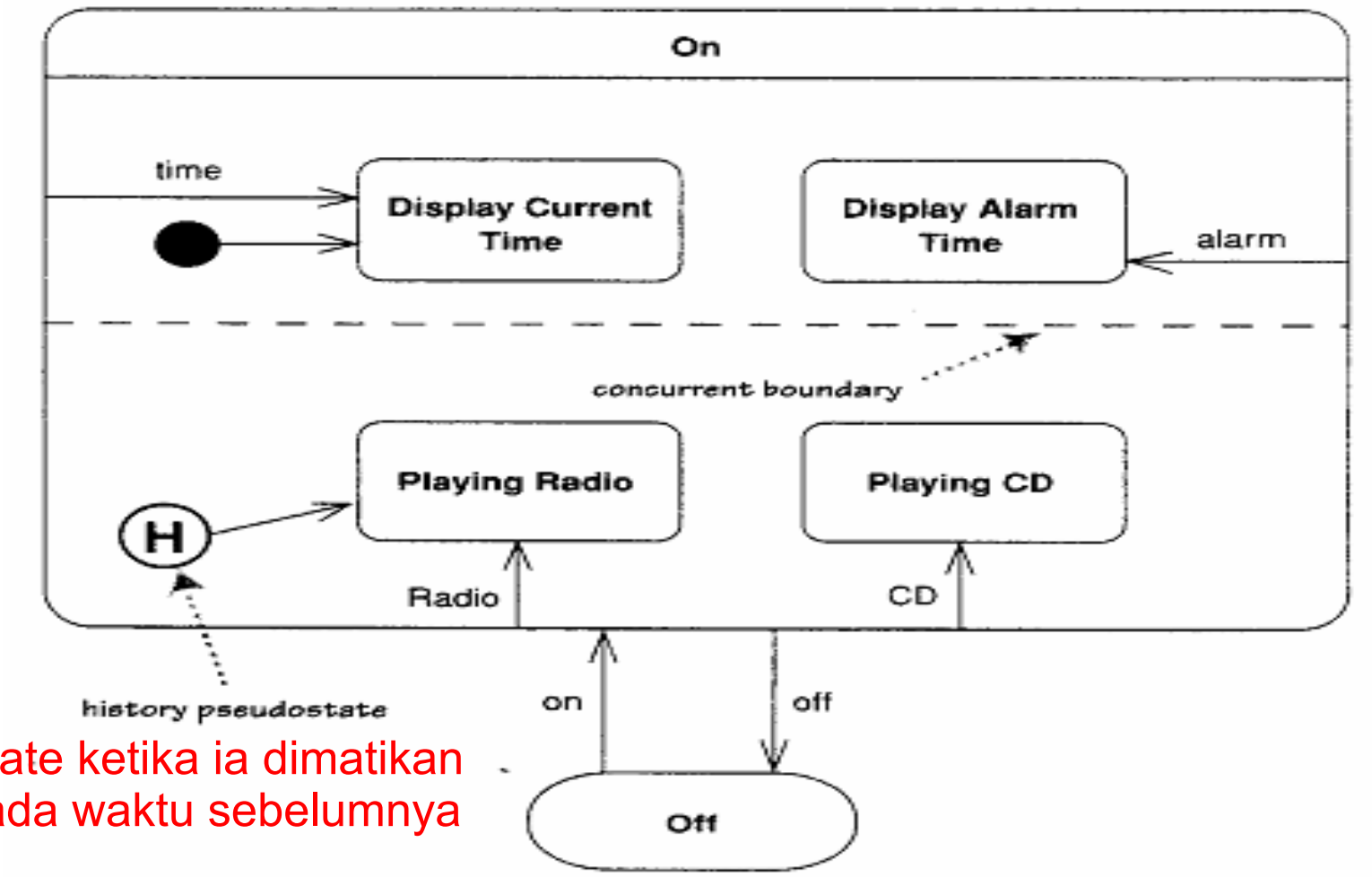
Concurrent States

- State dapat dipecah ke dalam beberapa (sub-)state diagram yang berjalan secara bersamaan



Concurrent States

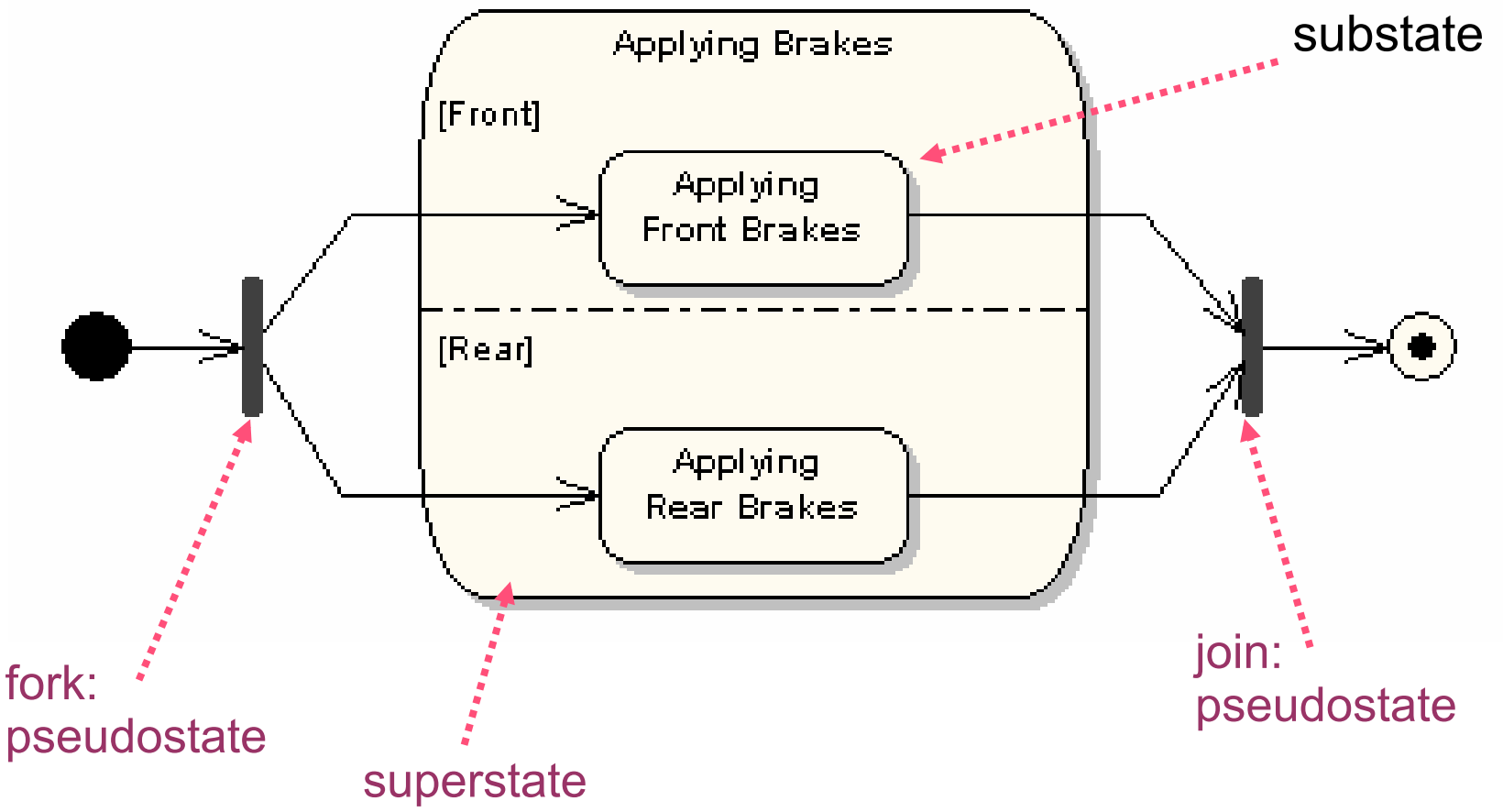
- Contoh : Concurrent orthogonal (sub-)states



State ketika ia dimatikan pada waktu sebelumnya

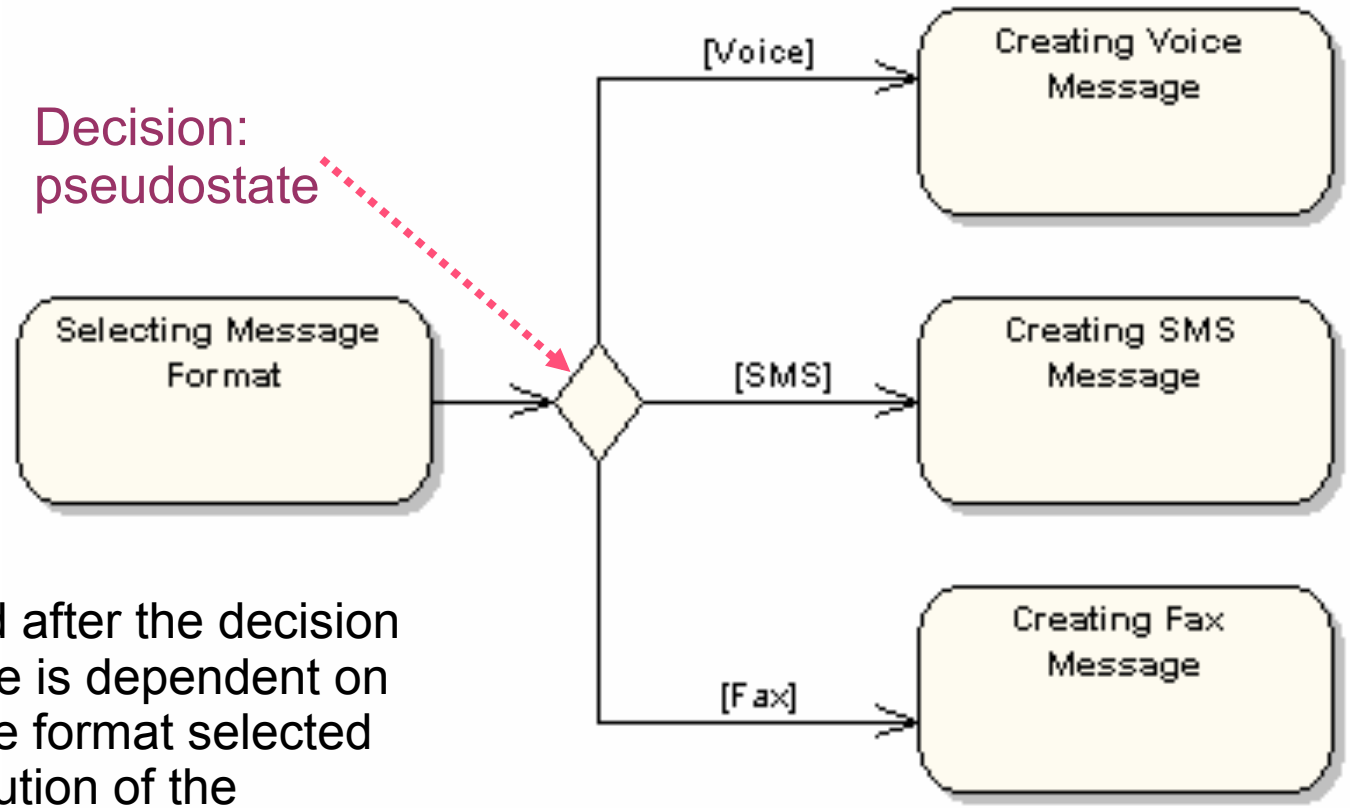
Concurrent States

- Concurrent states: Alternative diagram



Branch States

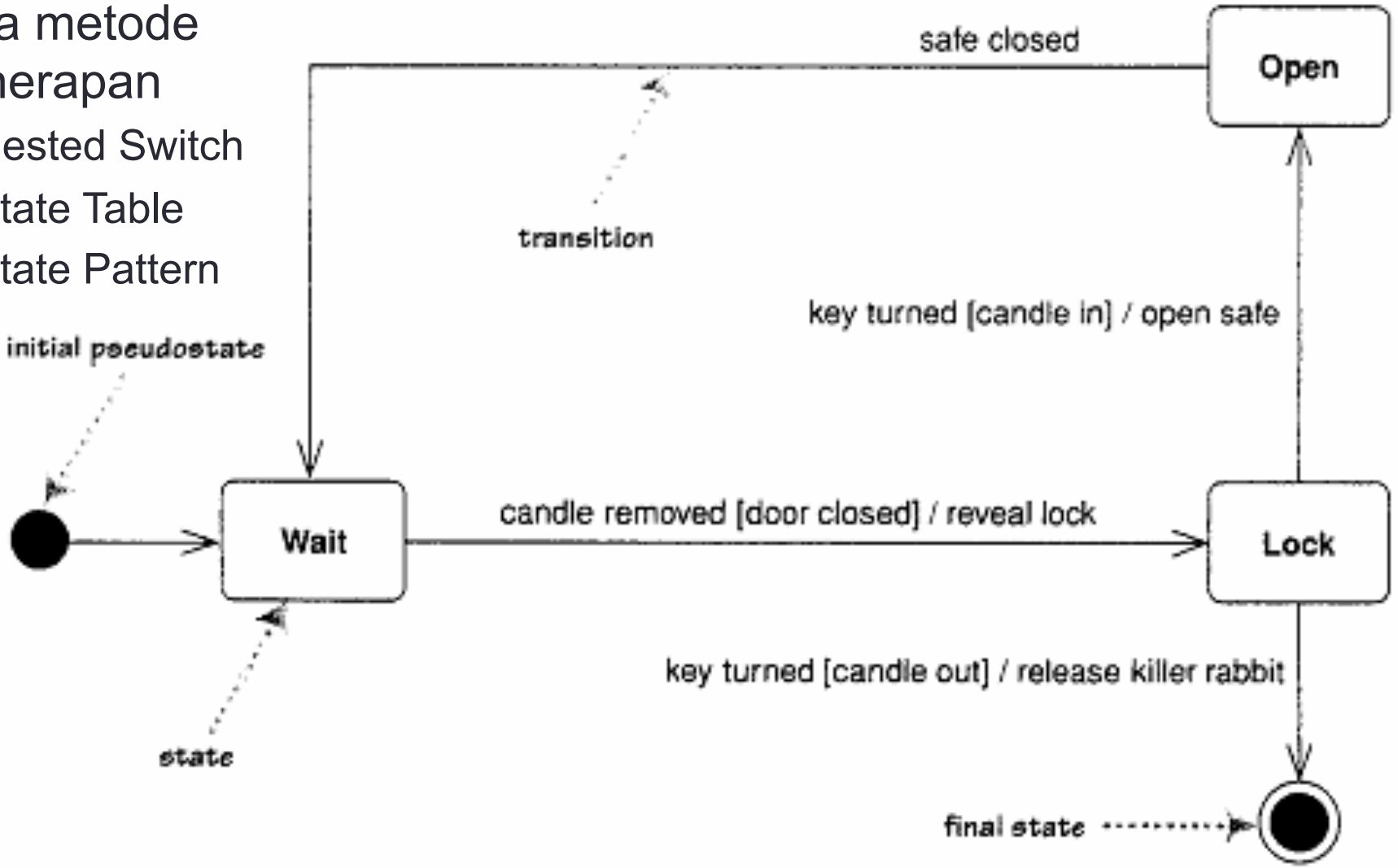
- Sebuah pseudo-state keputusan diperlihatkan sebagai sebuah belahketupat dengan satu transisi yang masuk dan dua atau lebih transisi yang meninggalkan.



state arrived after the decision pseudo-state is dependent on the message format selected during execution of the previous state.

Implementing

- Tiga metode penerapan
  - Nested Switch
  - State Table
  - State Pattern



Method 1. Nested Switch

```
public void HandleEvent (PanelEvent anEvent) {
   switch (CurrentState) {
      ……
      case PanelState.Wait   //current state = "Wait"
        switch (anEvent) {
           case PanelEvent.CandleRemoved          //trigger
                if (isDoorClose) {                //guard
                  RevealLock( )                   //activity
                  CurrentState = PanelState.Lock  //state change
                }
                break
        }
        …….
```

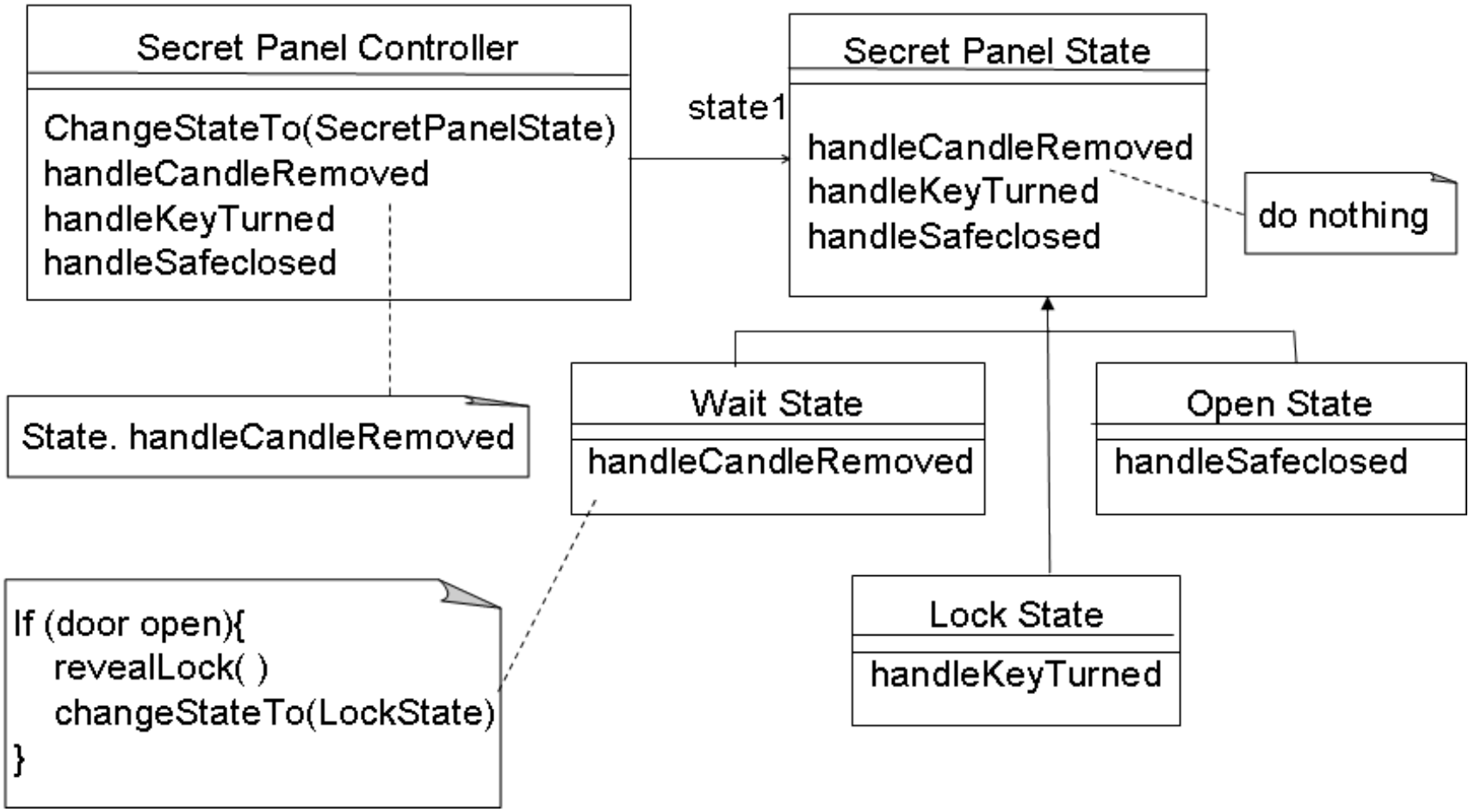
Method 2. State Table

- Menangkap sebuah informasi state diagram sebagai data
- Menghasilkan class-class berdasar tabel penerapannya

Source State	Target State	Event	Guard	Procedure
Wait	Lock	Candle remove	Door open	Reveal lock
Lock	Open	Key turned	Candle in	Open safe
Lock	Final	Key turned	Candle out	Release killer
Open	Wait	Safe closed		

Method 3. State Pattern

- Membuat sebuah hirarki dari class-class state untuk menangani perilaku



Kapan menggunakan?

- Baik untuk menggambarkan perilaku sebuah objek terhadap beberapa use case (namun tidak baik pada kolaborasi objek)
- Untuk memahami lebih baik kompleksitas class, secara khusus tindakan yang terjadi dalam beberapa perbedaan state mereka
- Bermanfaat untuk memodelkan bentuk perilaku objek UI dan kontrol

SELESAI!

Terima kasih!