REKAYASA PERANGKAT LUNAK

(PMI205)

PERTEMUAN 17

OBJECT-ORIENTED ANALYSIS & DESIGN (OOAD)

Presentasi dibuat oleh:

SASMITO BUDI UTOMO

KRISTINA HUTAJULU

MATERIAL STRUCTURE OOAD #1

An Introduction to Object-Oriented

- Why & What is Object-Oriented?
- Benefits of OOAD
- Object Oriented Methodologies
- Object Oriented Notations

Fundamental Concept of Object-Oriented

- Class
- Attribute
- Method
- Message
- Encapsulation
- Inheritance
- Polymorphism
- Abstraction
- Practical - 1

Modeling Techniques (1): Use Case Diagram

- Use Case Definition
- Use Case Communication
- Extension Point with Condition
- Complete Use Case
- Practical - 2

Modeling Techniques (2): Activity Diagram

- Definition
- Activity Diagram Notation
- Activity Diagram Step by Step
- Practical - 3

MATERIAL STRUCTURE OOAD #2

Modeling Techniques (3): CRC Cards (Class Responsibility Collaboration)

- Definition
- Identify Classes and Collaboration using CRC Cards
- Practical - 4

Modeling Techniques (5): Robustness Analysis

- Definition
- Introducing Robustness Diagram
- Robustness Diagram Rules
- Step & Guidelines
- Practical - 7

Modeling Techniques (4): Domain Modeling

- Definition
- Introducing a Class Diagram
- Association Relationship
- Aggregation & Composition
- Class Generalization
- Practical – 5
- Introducing an Object Diagram
- Object Links & Composite Object
- Practical - 6

MATERIAL STRUCTURE OOAD #3

Modeling Techniques (6): Object Interaction Modeling

- Communication Diagram Definition
- Communication Diagram Example
- Workshop (Comm. Diagram)
- Sequence Diagram Definition
- Message Broadcast Notation
- Sequence Diagram Example
- Object Creation, Deletion and Reflective Message
- Another Sequence Diagram Example
- Workshop (Sequence Diagram)
- Interaction Overview Diagram Definition
- Interaction Overview Diagram Example
- Timing Diagram Description
- Timing Diagram Example

Modeling Techniques (7): State Machine Diagram

- State Diagram Definition
- State Diagram Notation
- State Diagram Action & Action Types
- State Diagram Notation for Actions
- Composite State & Example
- State Diagram Step by Step
- Workshop (State Diagram)
- Package Diagram Definition
- Package Diagram Example
- Component Diagram Definition
- Component Diagram Example
- Deployment Diagram Definition
- Deployment Diagram Example
- Deployment & Component Diagram Example

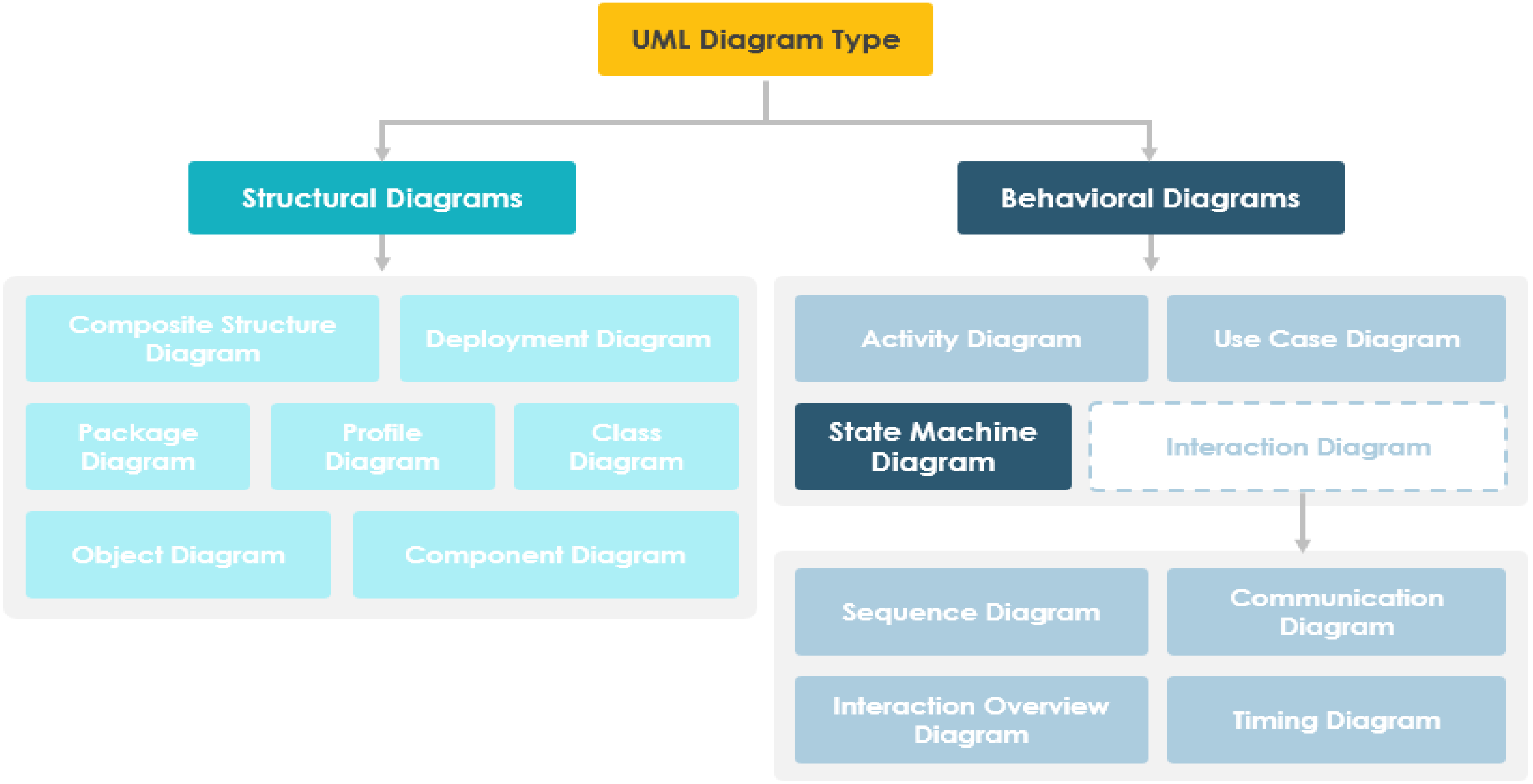
MATERIAL STRUCTURE OOAD #4

Beware of Object-Oriented Challenges

- Applications that are not suitable for OO
- Integration with Legacy systems
- Project Management
- Typical Testing Life Cycle in OOAD
- The Price to Pay by Ignoring Testing
- Testing architect Manages Testing Life Cycle & Test Plan
- Application Partitioning – MVC
- Summary – OOAD in Action
- Next step from here
- Reference, Copyright & Acknowledgement

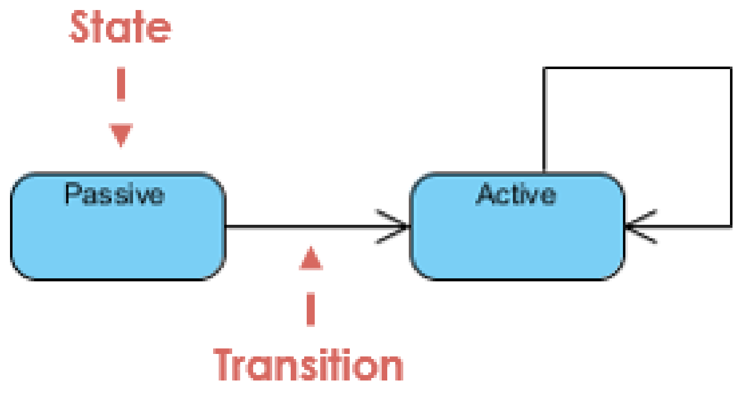
State Machine Diagrams

UML DIAGRAM TYPES (REMIND)



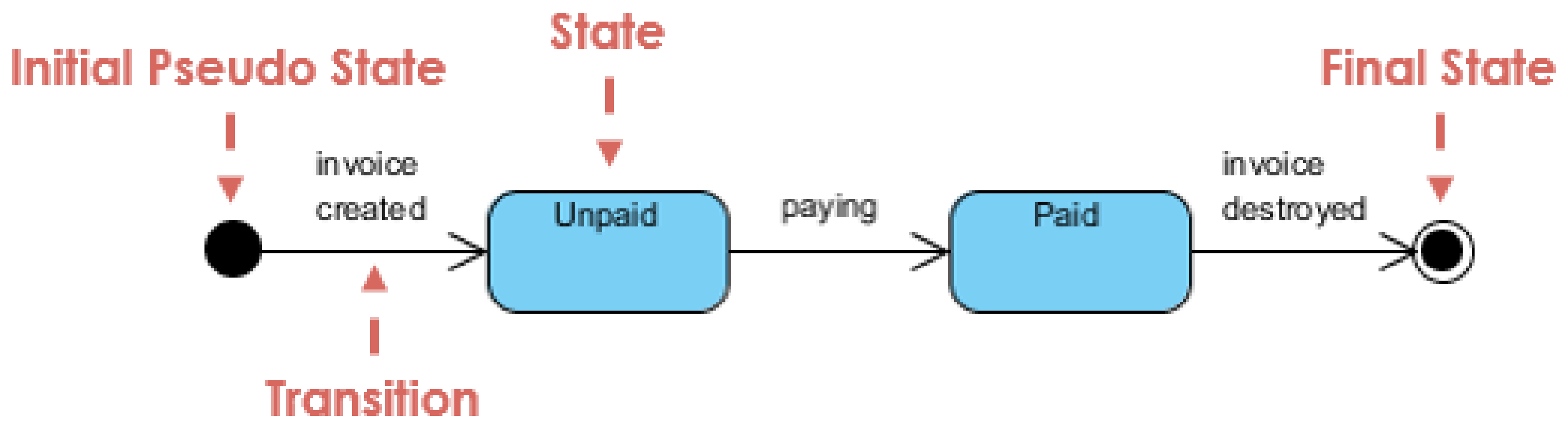
DEFINITION #1

- The behavior of an entity is not only a direct consequence of its inputs, but it also depends on its preceding state.
- State Machine Diagrams (or sometimes referred to as state diagram, state machine or state chart) show the different states of an entity.
- State machine diagrams can also show how an entity responds to various events by changing from one state to another.
- State machine diagram is a UML diagram used to model the dynamic nature of a system.
- A state machine diagram is a graph consisting of:
  - **States** (simple states or composite states)
  - **State transitions** connecting the states



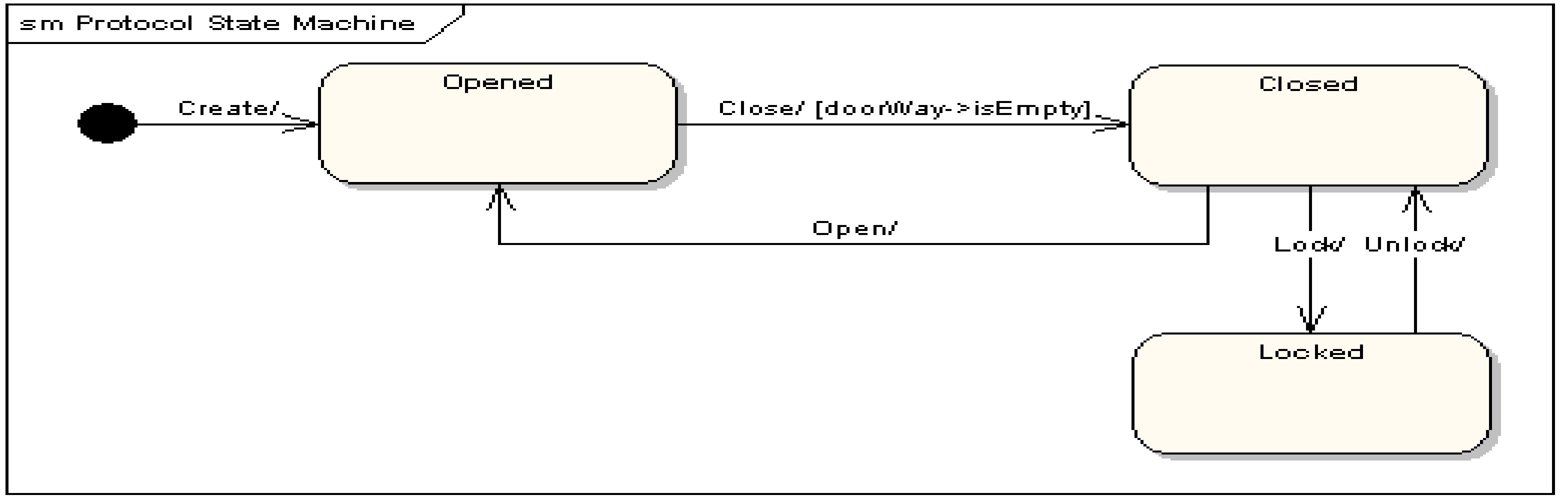
DEFINITION #2

- Simple State Machine Diagram Notation.



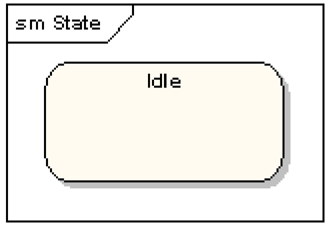
DEFINITION #3

- A state machine diagram models the behavior of a single object, specifying the sequence of events that an object goes through during its lifetime in response to events. As an example, the following state machine diagram shows the states that a door goes through during its lifetime.
- The door can be in one of three states: "Opened", "Closed" or "Locked". It can respond to the events Open, Close, Lock and Unlock. Notice that not all events are valid in all states; for example, if a door is opened, you cannot lock it until you close it. Also notice that a state transition can have a guard condition attached: if the door is Opened, it can only respond to the Close event if the condition

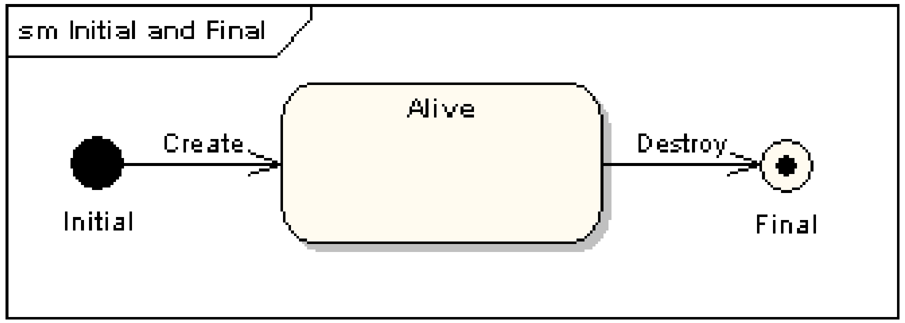


STATE #1

- **States** - A state is denoted by a round-cornered rectangle with the name of the state written inside it.



- **Initial and Final States** - The initial state is denoted by a filled black circle and may be labeled with a name. The final state is denoted by a circle with a dot inside and may also be labeled with a name.



STATE #2

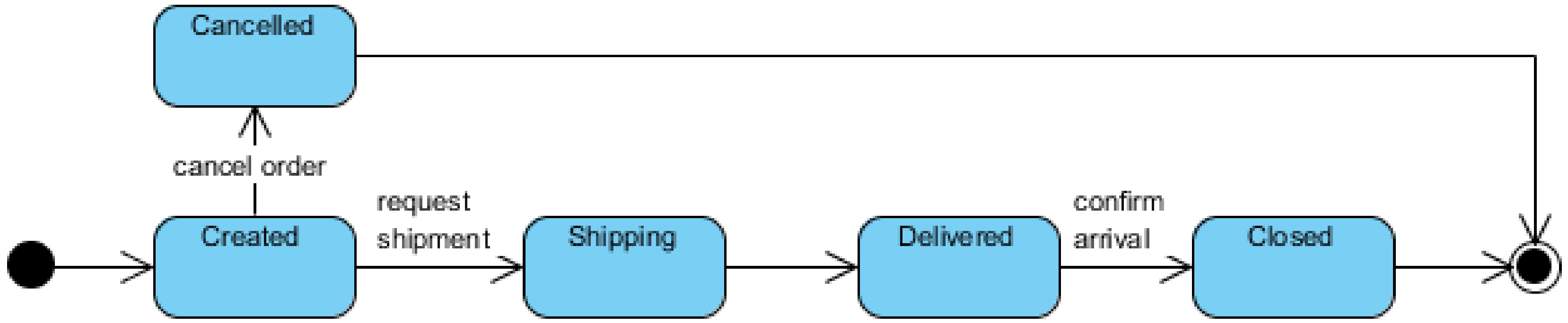
Characteristics of State:

- State represent the conditions of objects at certain points in time.
- Objects (or Systems) can be viewed as moving from state to state
- A point in the lifecycle of a model element that satisfies some condition, where some particular action is being performed or where some event is waited

STATE #3

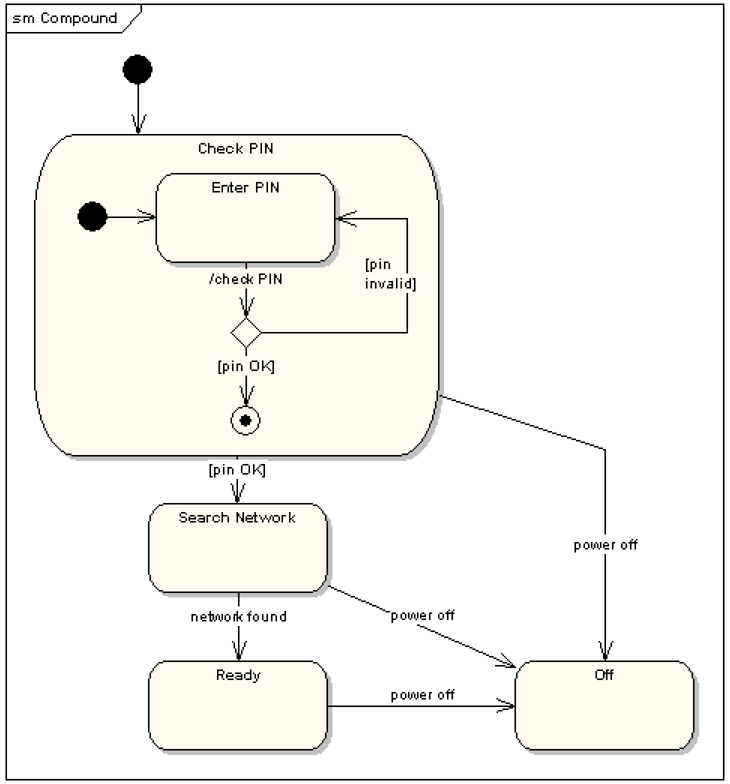
Characteristics of State:

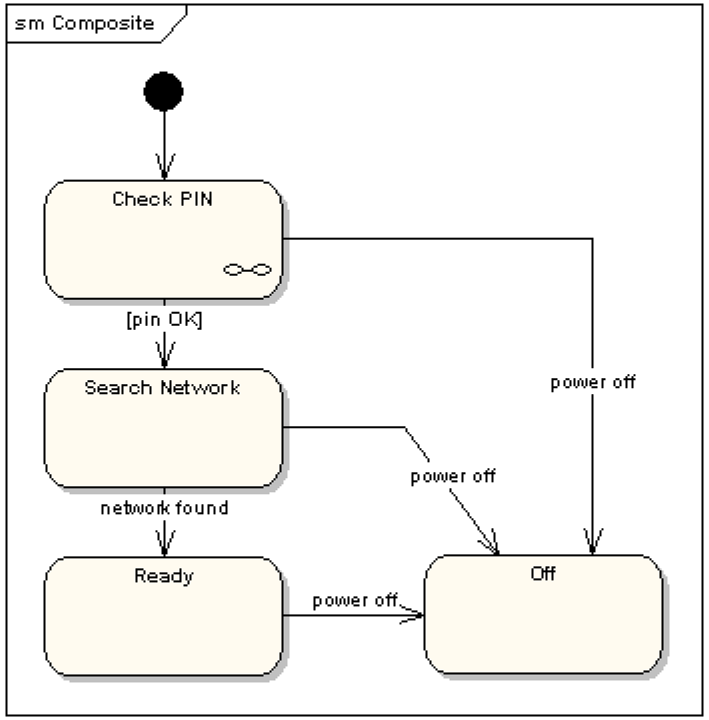
- **The initial state** of a state machine diagram, known as an initial pseudo-state, is indicated with a solid circle. A transition from this state will show the first real state
- **The final state** of a state machine diagram is shown as concentric circles. An open loop state machine represents an object that may terminate before the system terminates, while a closed loop state machine diagram does not have a final state; if it is the case, then the object lives until the entire system terminates



STATE #4

- **Compound States -** A state machine diagram may include sub-machine diagrams, as in the example below.



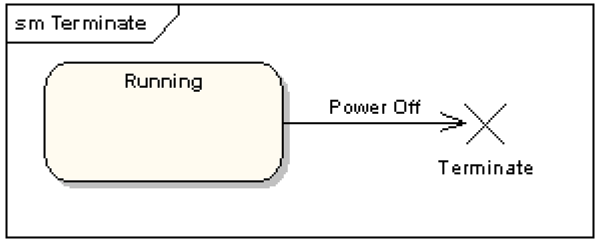


Alternative way to show the same information

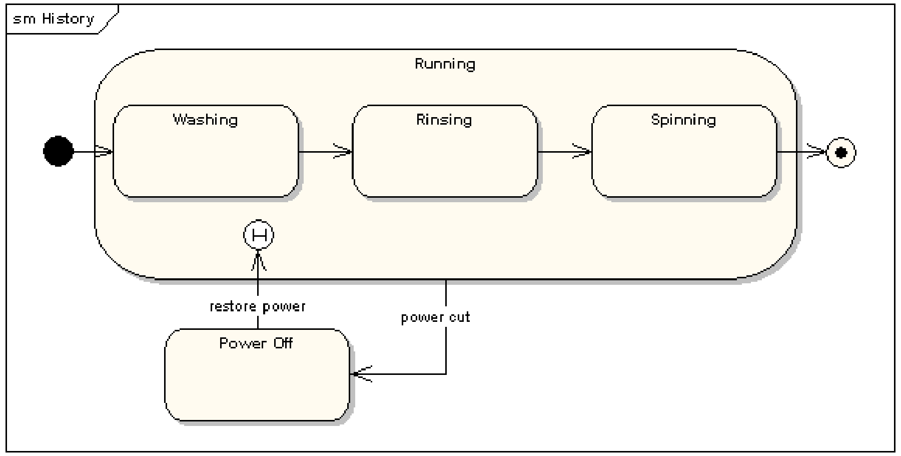
- **The ∞ symbol indicates that details of the Check PIN sub-machine are shown in a separate diagram.**

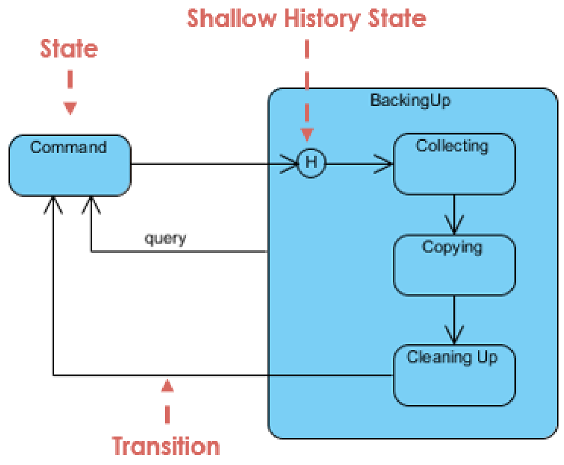
STATE #5

- **Terminate Pseudo-State -** Entering a terminate pseudo-state indicates that the lifeline of the state machine has ended. A terminate pseudo-state is notated as a cross.



- **History States -** A history state is used to remember the previous state of a state machine when it was interrupted. The following diagram illustrates the use of history states. The example is a state machine belonging to a washing machine.





TRANSITION #1

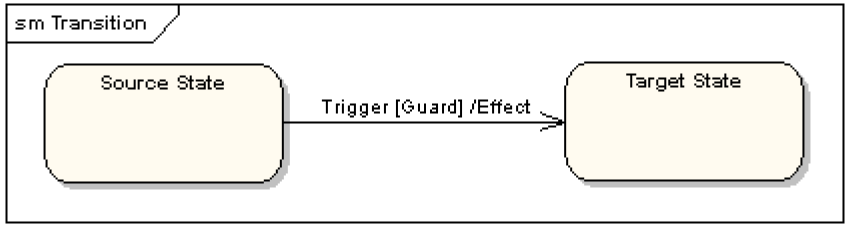
- Transition lines depict the movement from one state to another. Each transition line is labeled with the event that causes the transition
- Viewing a system as a set of states and transitions between states is very useful for describing complex behaviors
- Understanding state transitions is part of system analysis and design
- A Transition is the movement from one state to another state

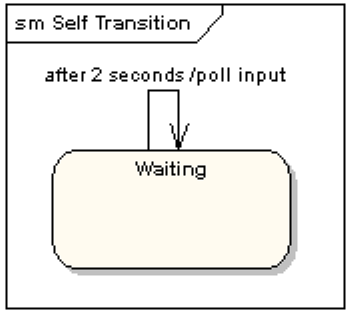
TRANSITION #2

- Transitions between states occur as follows:
  1. An element is in a source state
  2. An event occurs
  3. An action is performed
  4. The element enters a target state
- Multiple transitions occur either when different events result in a state terminating or when there are guard conditions on the transitions
- A transition without an event and action is known as automatic transitions

TRANSITION #3

- **Transitions** - Transisi dari satu state ke state berikutnya yang dilambangkan dengan garis dengan panah. Transisi mungkin memiliki trigger, guard, dan effect, seperti di bawah ini.
- **Self-Transitions -** Sebuah state dapat memiliki sebuah transisi yang dapat Kembali ke dirinya sendiri, seperti dalam diagram berikut. Ini paling berguna ketika suatu effect dikaitkan dengan transisi.

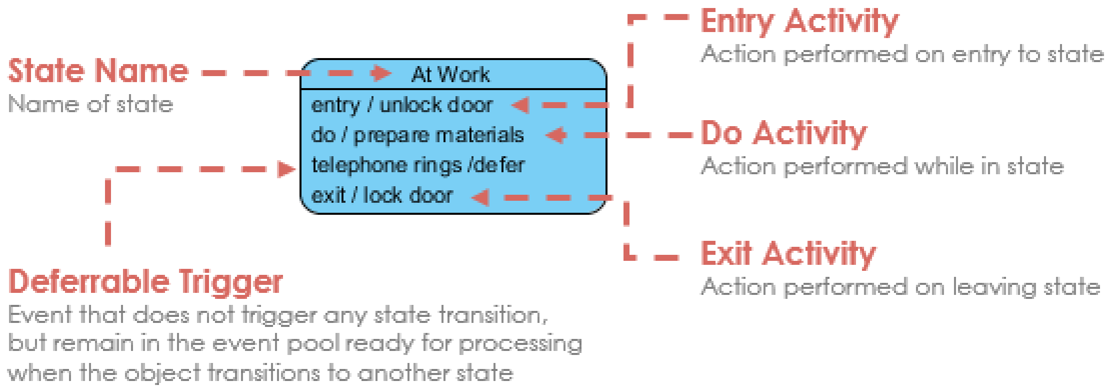




- **"Trigger" adalah** penyebab dari transisi, bisa jadi seperti sebuah signal, sebuah event, sebuah perubahan di beberapa kondisi, atau dengan berlalunya waktu. **"Guard"** adalah sebuah kondisi dimana harus true untuk sebuah trigger untuk menyebabkan terjadinya transisi. **"Effect"** adalah sebuah tindakan yang akan dipanggil langsung pada objek yang memiliki state machine sebagai hasil dari transisi.

ENTRY / EXIT ACTION #1

- **State Actions** - Dalam contoh transisi di atas, efek dikaitkan dengan transisi. Jika target state memiliki banyak transisi yang tiba di sana, dan setiap transisi memiliki efek yang sama yang terkait dengannya, akan lebih baik untuk mengaitkan efek dengan target state daripada transisi. Ini dapat dilakukan dengan mendefinisikan entry action untuk state. Diagram di bawah ini memperlihatkan state dengan entry action dan exit action.

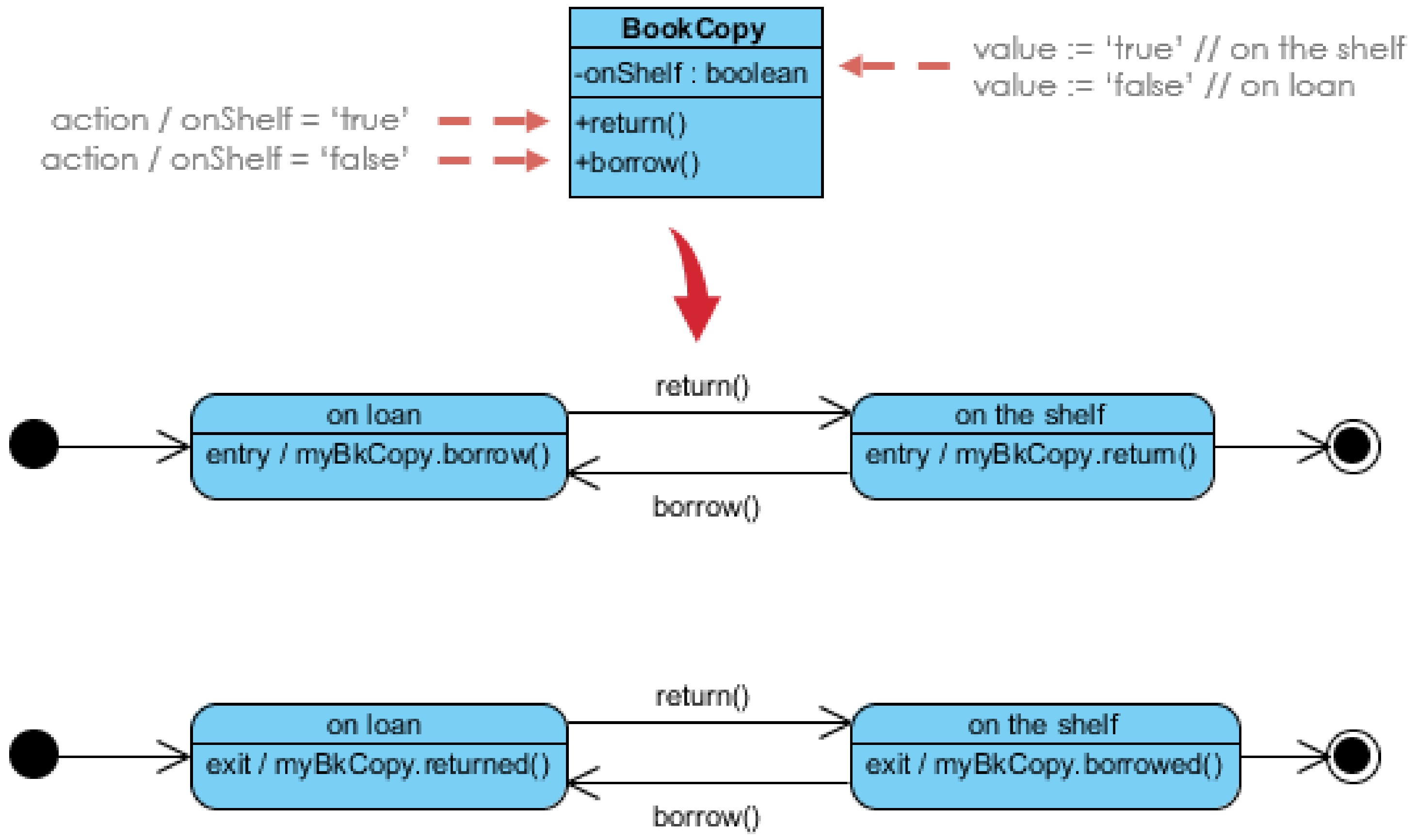


State Name
Name of state
At Work
entry / unlock door
do / prepare materials
telephone rings /defer
exit / lock door
Entry Activity
Action performed on entry to state
Do Activity
Action performed while in state
Exit Activity
Action performed on leaving state
Deferrable Trigger
Event that does not trigger any state transition, but remain in the event pool ready for processing when the object transitions to another state

- Dimungkinkan juga untuk mendefinisikan tindakan yang terjadi pada peristiwa, atau tindakan yang selalu terjadi. Dimungkinkan untuk menentukan sejumlah tindakan dari setiap jenis.

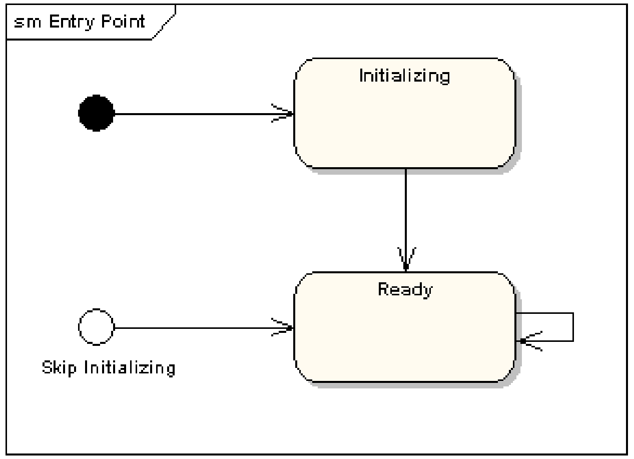
ENTRY / EXIT ACTION #2

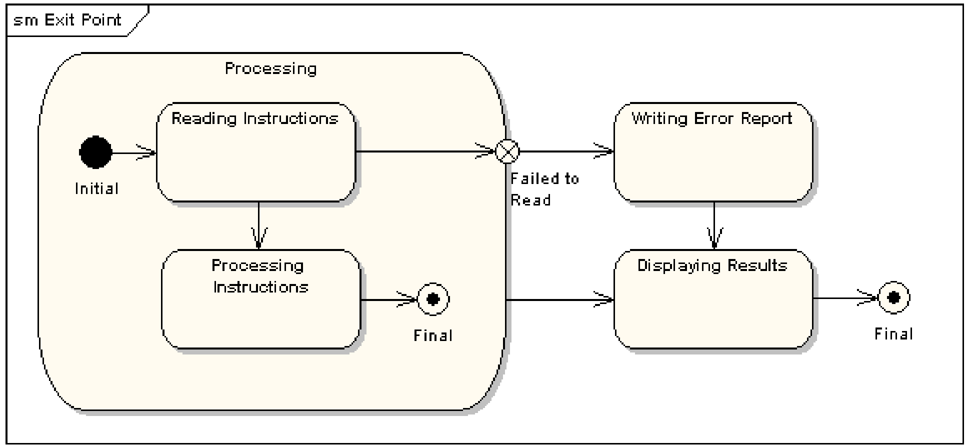
This example illustrates a state machine diagram derived from a Class - "BookCopy":


BookCopy
-onShelf : boolean
+return()
+borrow()
value := 'true' // on the shelf
value := 'false' // on loan
action / onShelf = 'true'
action / onShelf = 'false'
on loan
entry / myBkCopy.borrow()
return()
borrow()
on the shelf
entry / myBkCopy.return()
on loan
exit / myBkCopy.returned()
return()
borrow()
on the shelf
exit / myBkCopy.borrowed()

ENTRY / EXIT ACTION #3

- **Entry Point -** Terkadang Anda tidak ingin memasukkan sub-machine pada keadaan awal normal. Misalnya, dalam machine berikut akan normal untuk memulai dalam keadaan "Initializing", tetapi jika karena alasan tertentu tidak perlu melakukan inisialisasi, akan mungkin untuk memulai dalam keadaan “Ready" state beralih ke entry point.



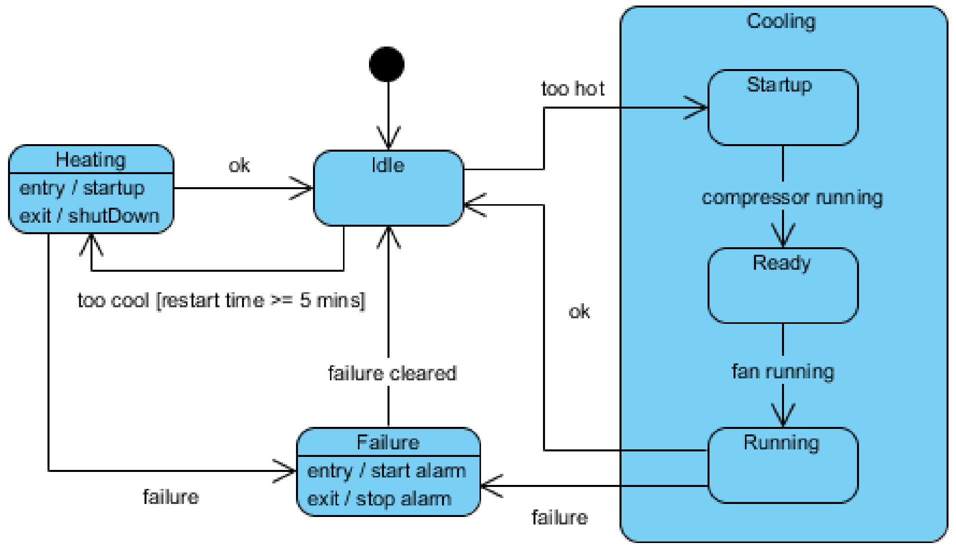


- **Exit Point -** Dengan cara yang mirip dengan entry point, ada yang dinamakan exit point. Diagram berikut memberikan contoh di mana state yang dijalankan setelah state pemrosesan utama tergantung pada rute mana yang digunakan untuk transisi keluar dari state.

SUBSTATES

- State sederhana adalah state yang tidak memiliki substruktur. State yang memiliki substate (nested states) disebut composite state. Substate dapat di nested ke tingkat mana pun. Nested state machine mungkin memiliki paling banyak satu state awal dan satu state akhir. Substate digunakan untuk menyederhanakan mesin flat state yang kompleks dengan menunjukkan bahwa beberapa state hanya dimungkinkan dalam konteks tertentu (status penutup).

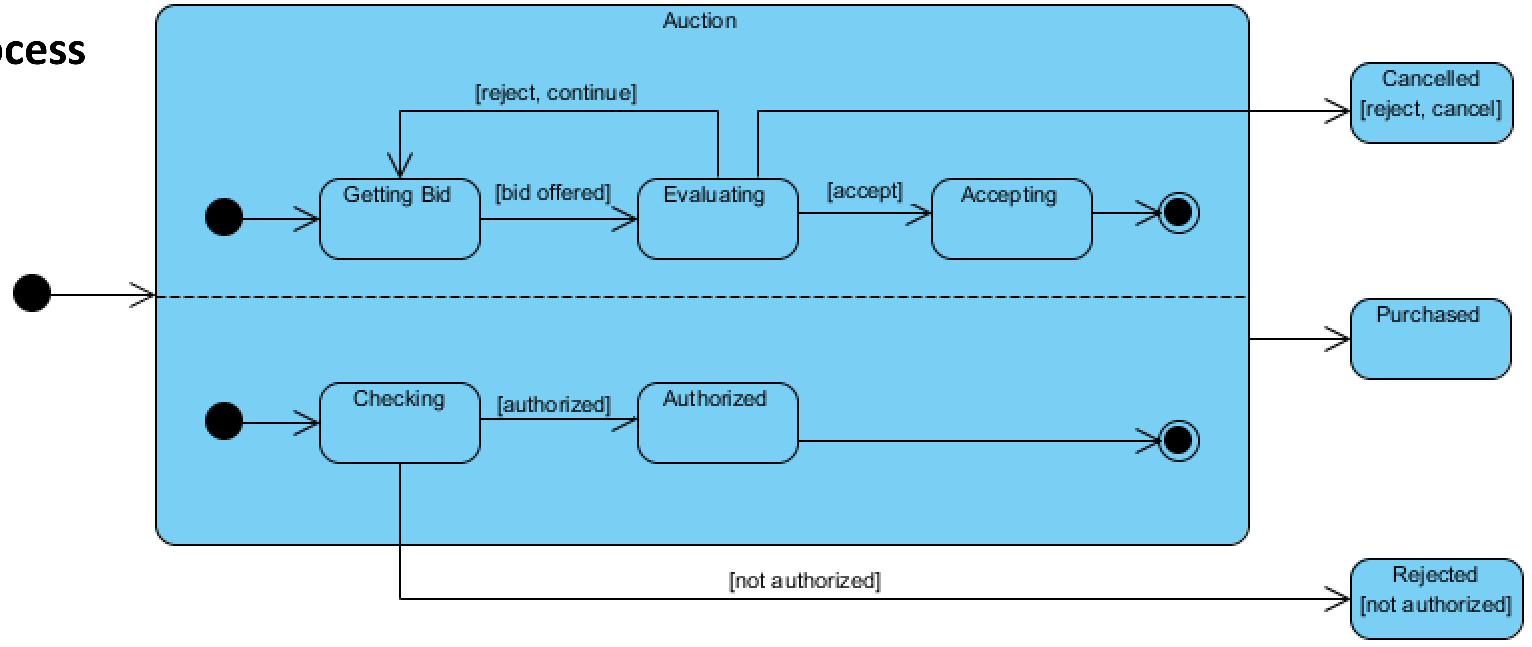
Substate Example - Heater



CONCURRENT STATES

- Seperti yang sudah disebutkan di atas, status dalam state machine diagrams dapat ditumpuk. State-state yang terkait dapat dikelompokkan menjadi satu composite state. Nesting state di dalam yang lain diperlukan ketika suatu kegiatan melibatkan sub-kegiatan bersamaan. State machine diagram bagian berikut memodelkan lelang dengan dua substate bagian bersamaan: memproses tawaran dan mengotorisasi batas pembayaran

Auction Process



Dalam contoh ini, state machine yang pertama kali memasuki Lelang memerlukan fork di awal menjadi dua thread awal yang terpisah. Setiap substate memiliki state keluar untuk menandai akhir thread. Kecuali ada jalan keluar yang tidak normal (Dibatalkan atau Ditolak), keluar dari composite state terjadi ketika kedua substate telah keluar

Workshop 8

State Machine Diagram

REKAYASA PERANGKAT LUNAK

(PMI205)

PERTEMUAN 17

OBJECT-ORIENTED ANALYSIS & DESIGN (OOAD)

SELESAI, TERIMA KASIH.