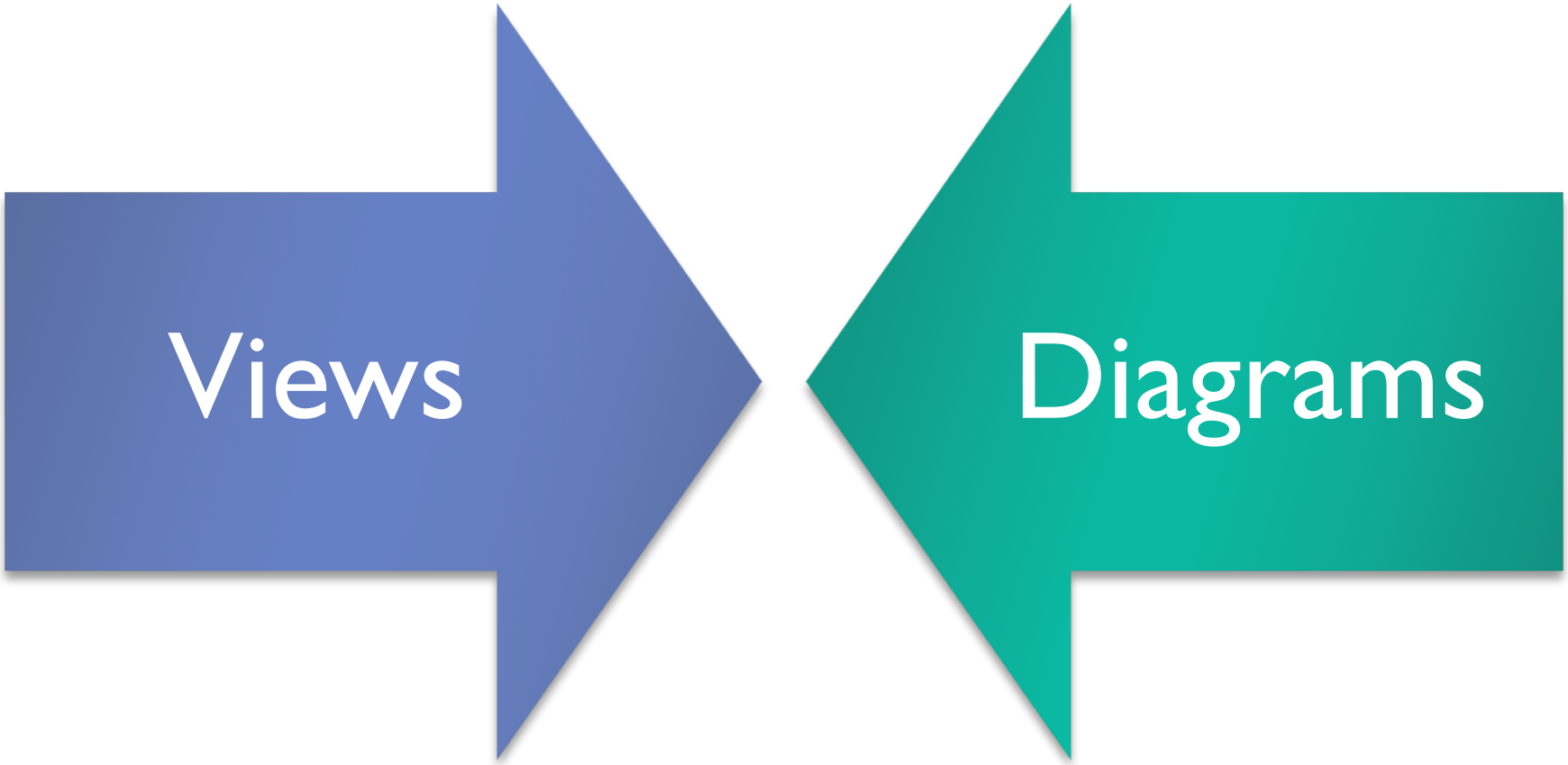
UNIFIED MODELING LANGUAGE™

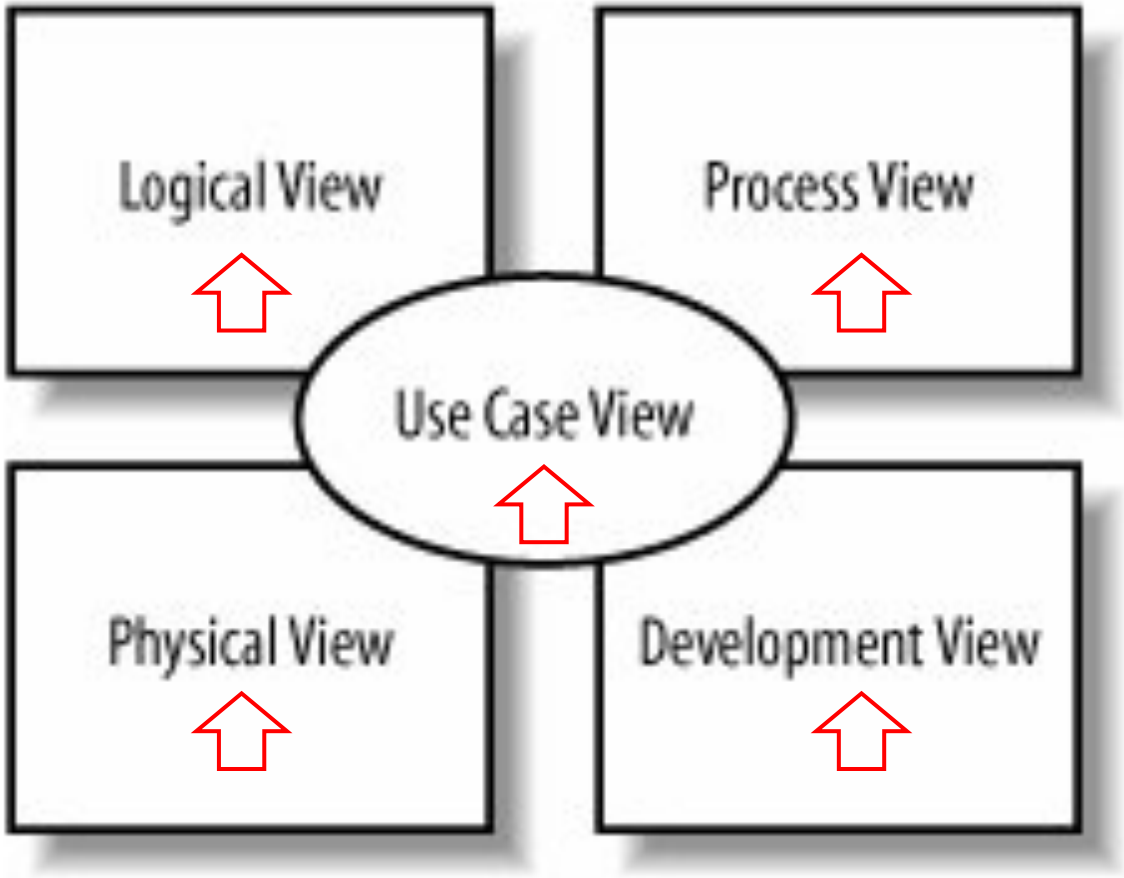
14. Modeling an Object's State: State Machine Diagrams

Shaoning Zeng, http://zsn.cc

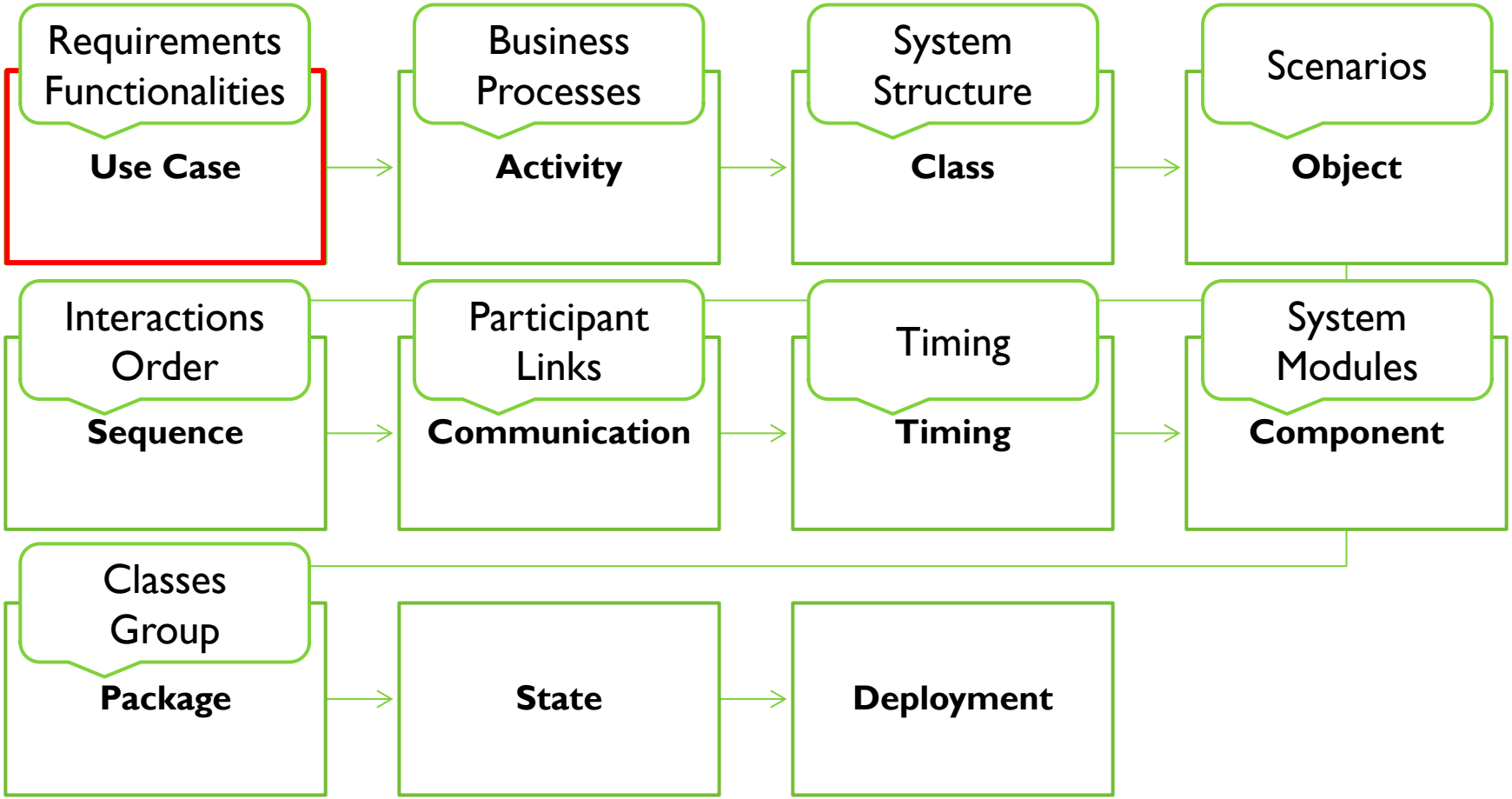
Modeling Language



Views of Your Model



Roadmap


Requirements
Functionalities
Use Case
Business
Processes
Activity
System
Structure
Class
Scenarios
Object
Interactions
Order
Sequence
Participant
Links
Communication
Timing
Timing
System
Modules
Component
Classes
Group
Package
State
Deployment

Diagrams in each view

Use case View	• User case diagrams
Process View	• Activity diagrams
Logical View	• Class diagrams • Object diagrams • Sequence diagrams • Communication diagrams • Timing diagrams • State machine diagrams
Development View	• Component diagrams • Package diagrams
Physical View	• Deployment diagrams

Why using State Machine Diagrams?

- Sometimes **the state of an object** or system is an important factor in its behavior.
  - AccountApplication object in CMS: approved or rejected
- Some typical usages:
  - Real-time/mission-critical systems, such as heart monitoring software
  - Dedicated devices whose behavior is defined in terms of state, such as ATMs
  - First-person shooter games, such as Doom or Half-Life

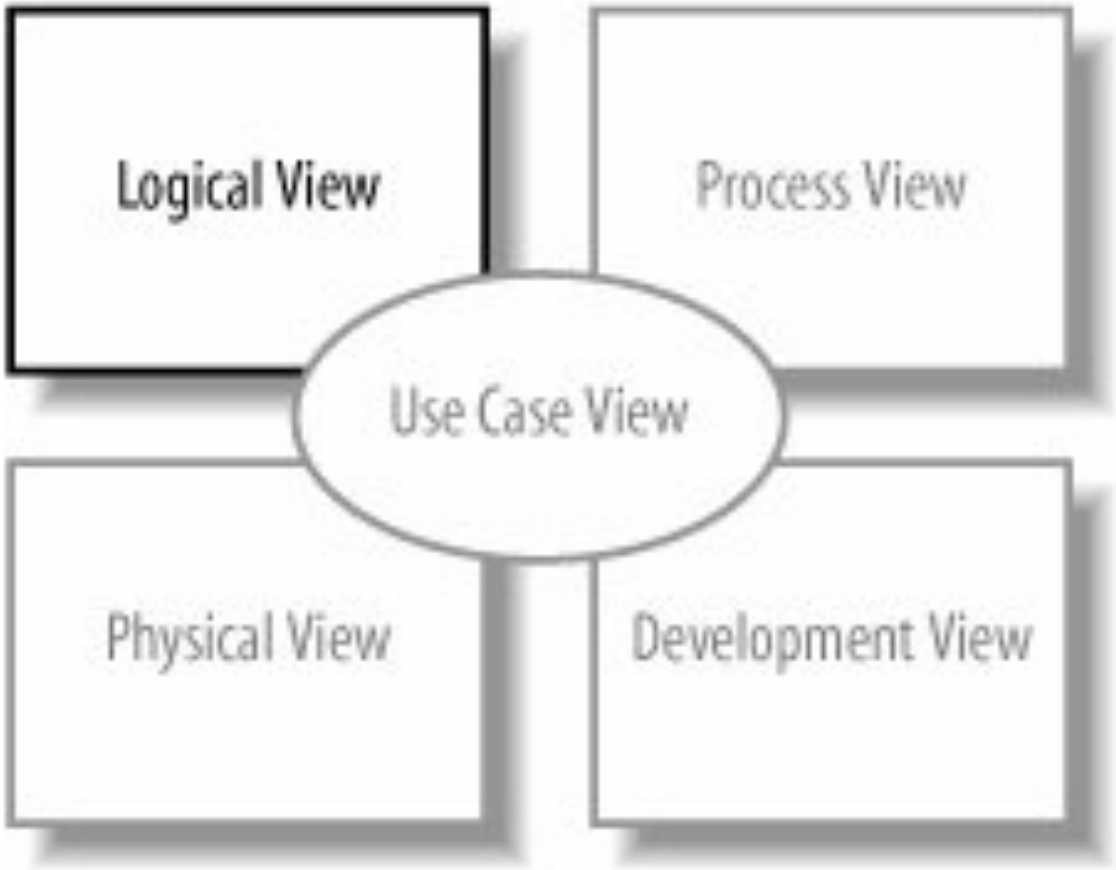
14. Modeling an Object's State: State Machine Diagrams

14.9. Protocol State Machines

14.4. States in Software
14.5. Advanced State Behavior
14.6. Composite States
14.7. Advanced Pseudostates
14.8. Signals

14.1. Essentials
14.2. States
14.3. Transitions

State machine diagrams are part of the logical model of your system



14.1. Essentials

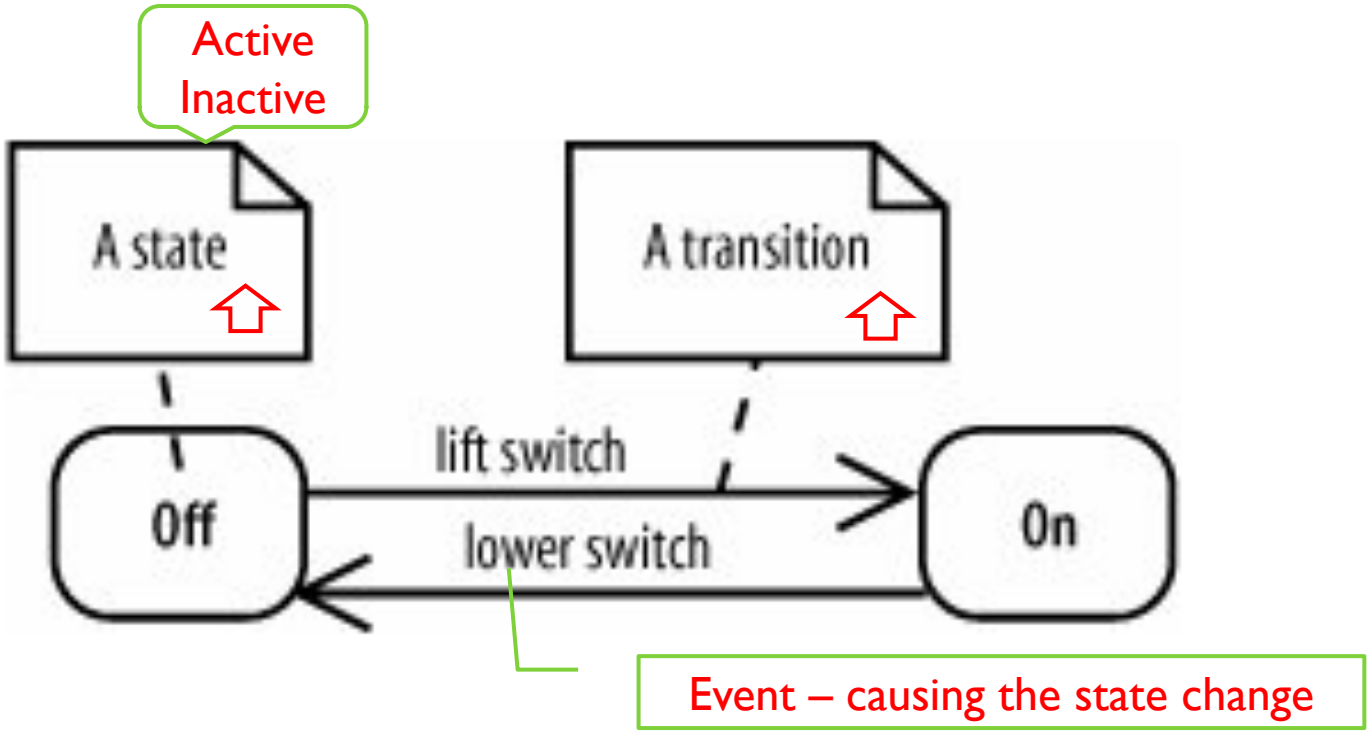
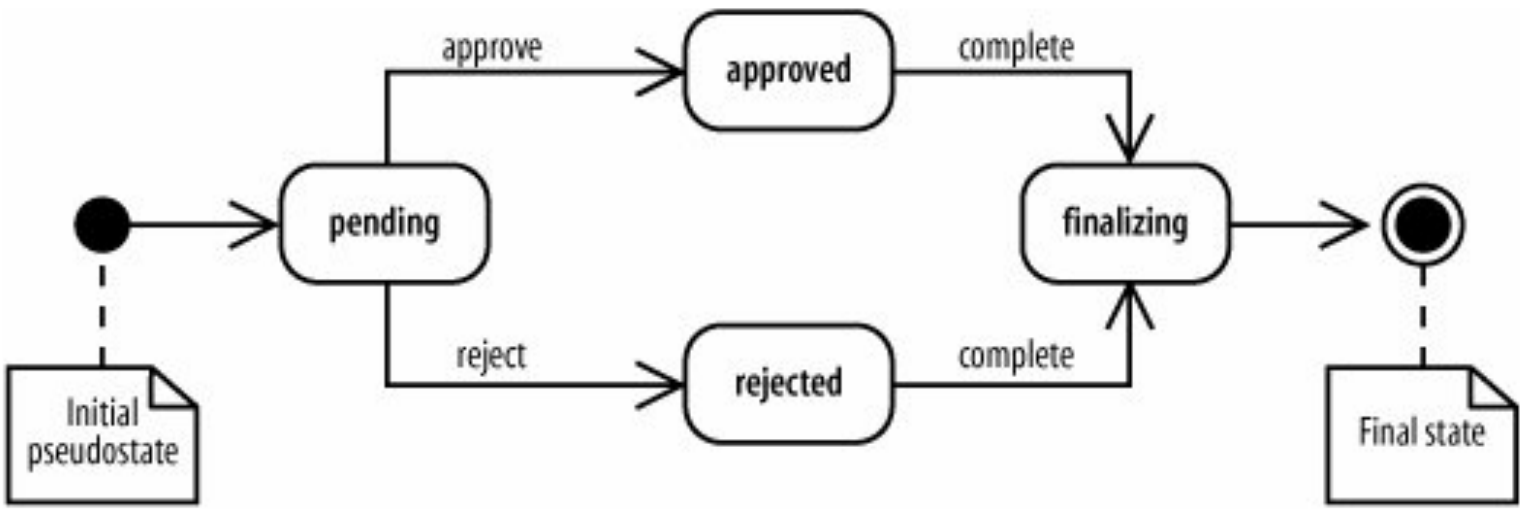


Figure 14-2. The fundamental elements of a state diagram: states and transitions between states

Figure 14-3. Initial pseudostate and final states in an AccountApplication state diagram



Pseudostates are **special markers** that direct the flow of traffic in a state diagram.

14.2. States

- A state is a **condition** of being at a certain time.
- A state can be a **passive quality**,
  - such as On and Off for the light object.
- A state can also be an **active quality**, or something that an object is **doing**.
  - For example, a coffeemaker has the state Brewing during which it is brewing coffee.
- A state is drawn as a rounded rectangle['rek,tæŋgl] with the name of the state in the center, as shown in Figure 14-4.

Figure 14-4. A rectangle with rounded corners and the name in the center is the most common way to draw a state

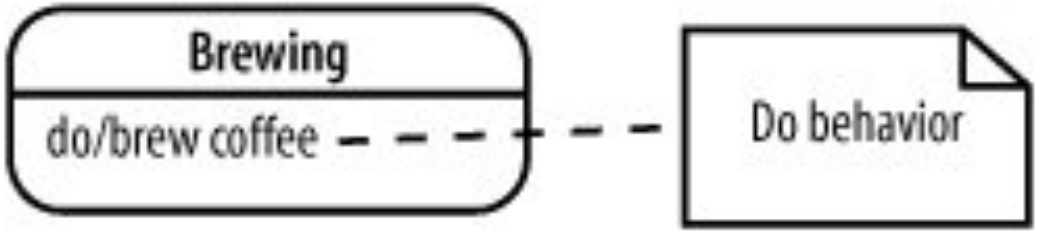


Figure 14-5. Showing the behavior details of a "doing" state

14.3. Transitions

- A transition, shown with an arrow, represents a change of states from a ***source state*** to a ***target state***.
- A ***transition description***, written along the arrow, describes the circumstances causing the state change to occur.

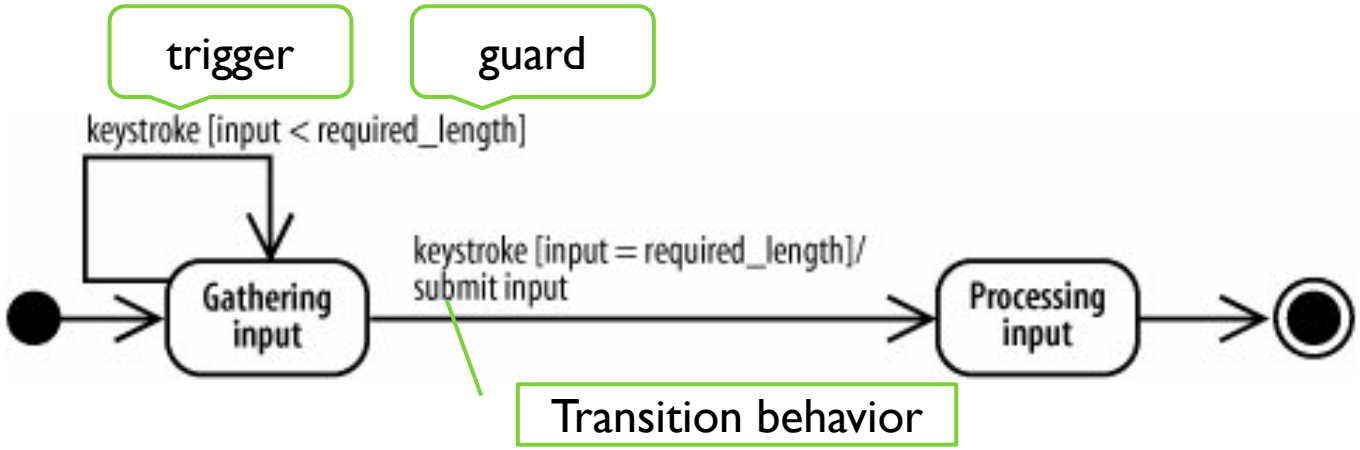


Figure 14-6. This input processing state diagram models features a trigger, guard, and transition behavior along one of its transitions

Figure 14-7. CD player state diagram, featuring a variety of transition descriptions

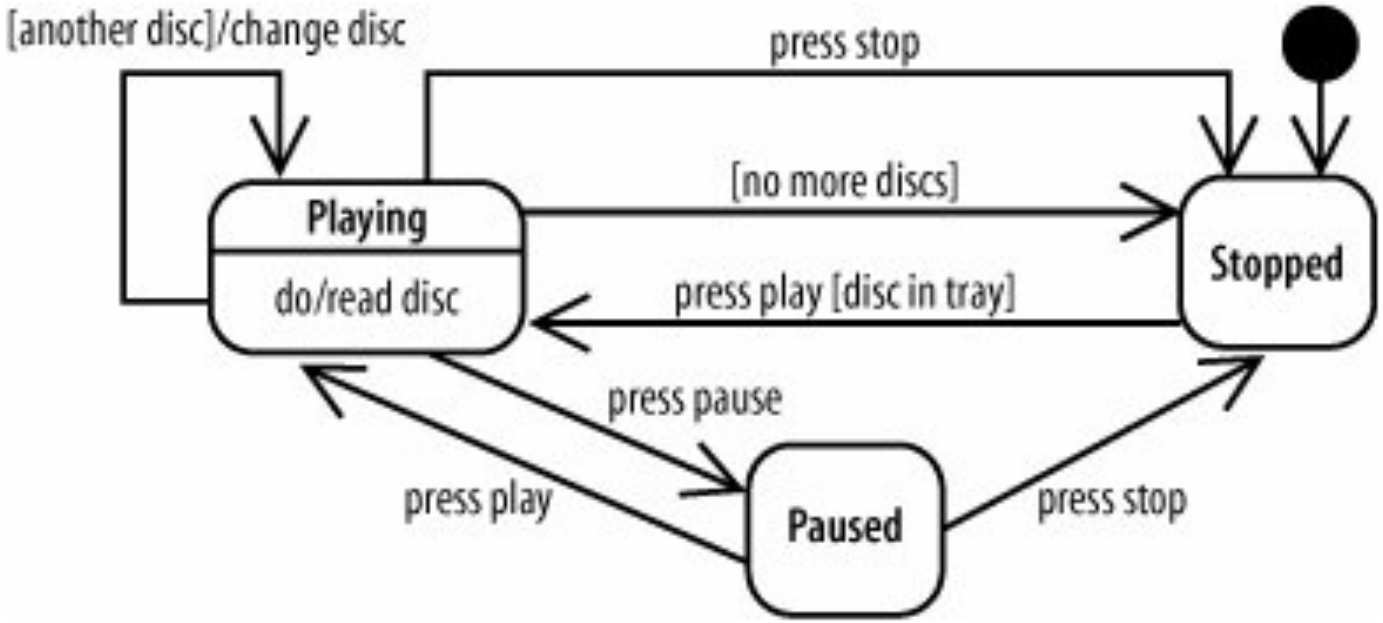


Figure 14-8. The most common type of transition features only a trigger

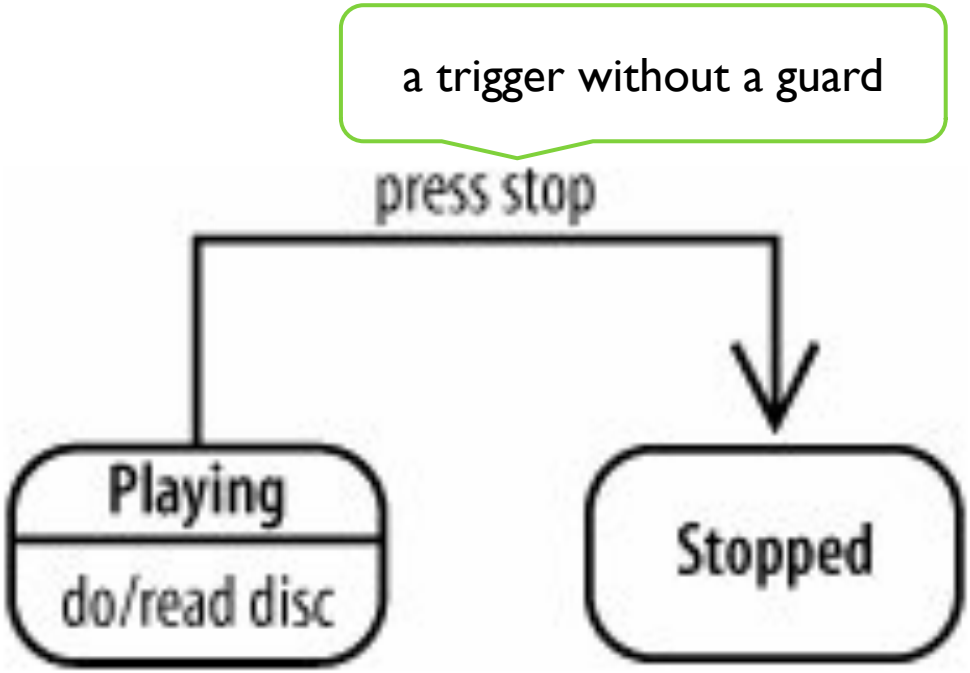


Figure 14-9. A guard will **block** a transition if it evaluates to **false**

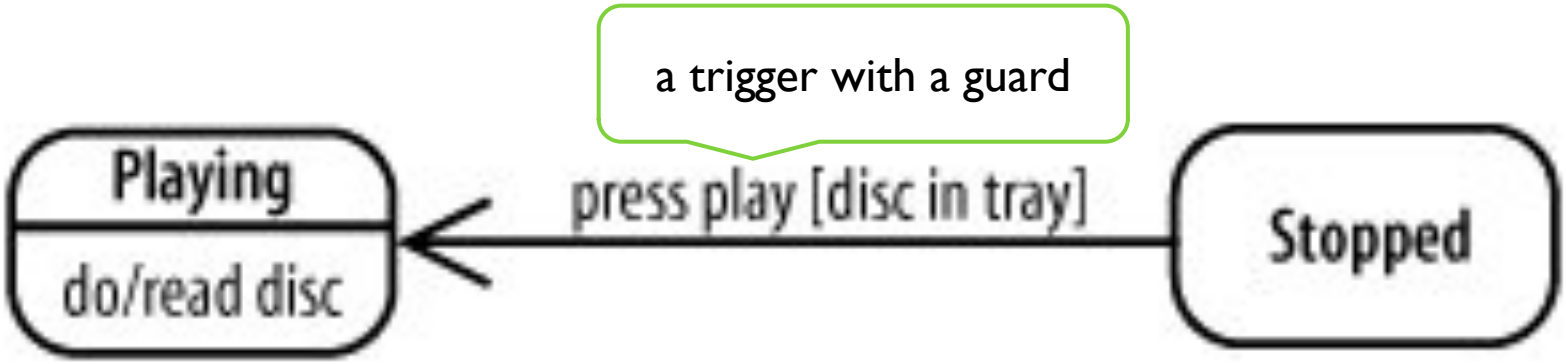


Figure 14-10. In this example, a transition is caused by the completion of **internal behavior**

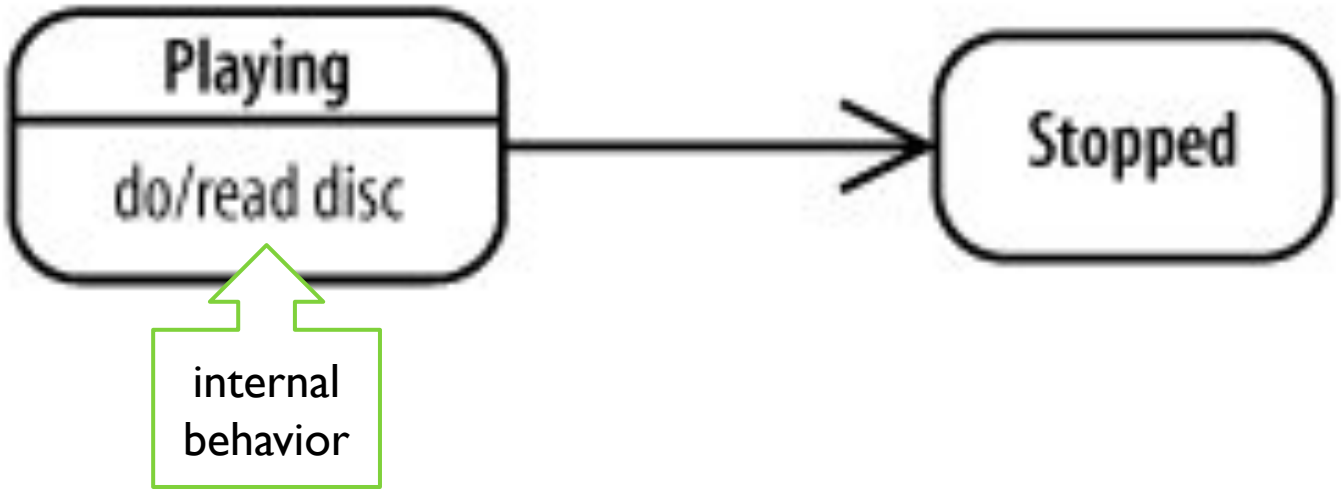
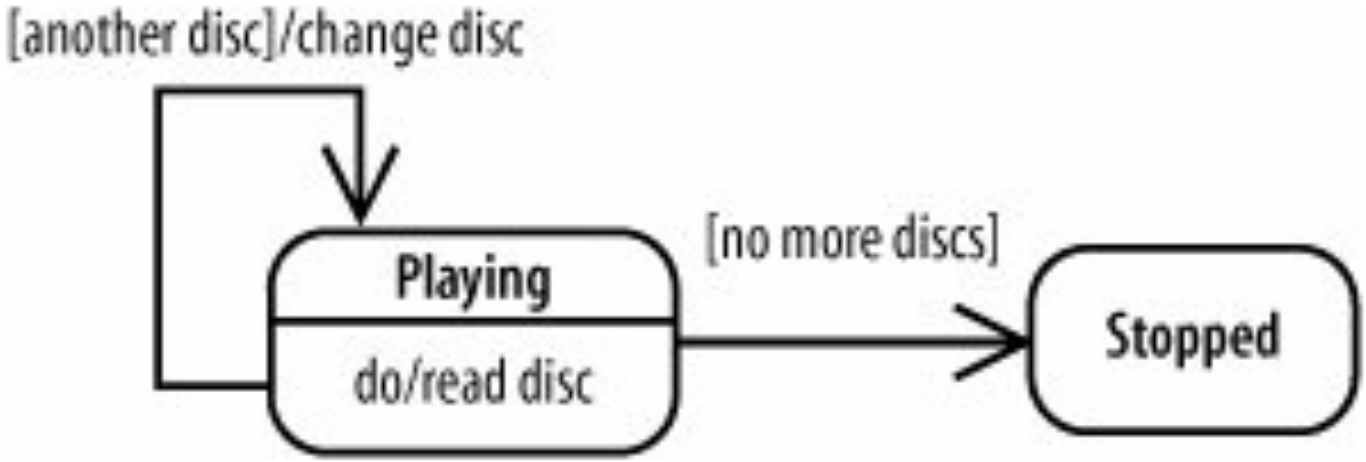


Figure 14-11. Using guards to model a choice between paths



14.4. States in Software

- In software, state diagrams model an **object's *life cycle***, or the states it goes through during its lifespan.

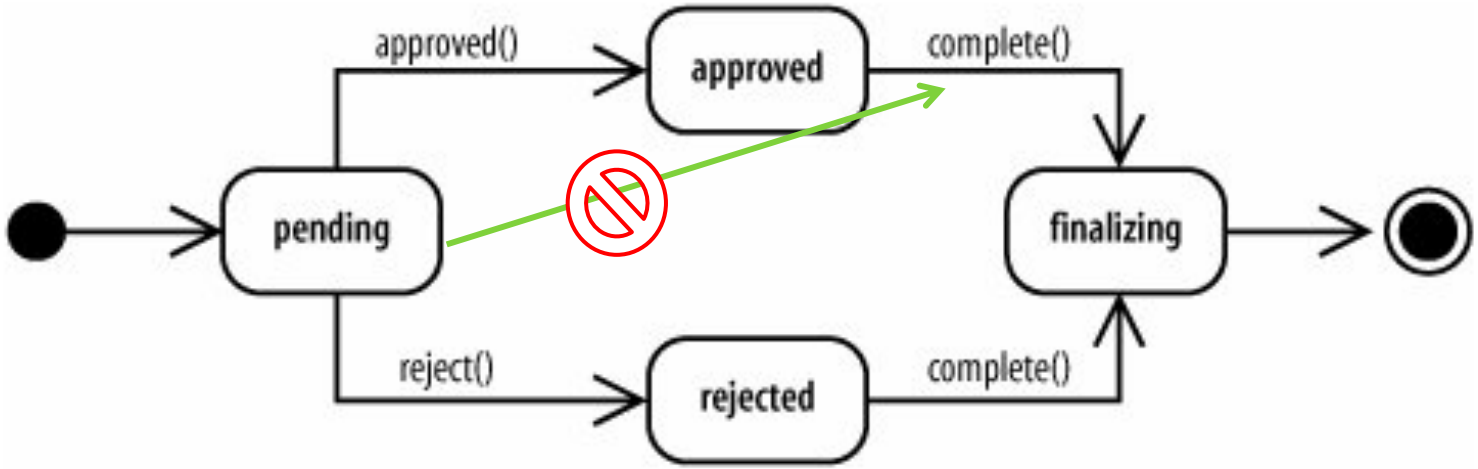
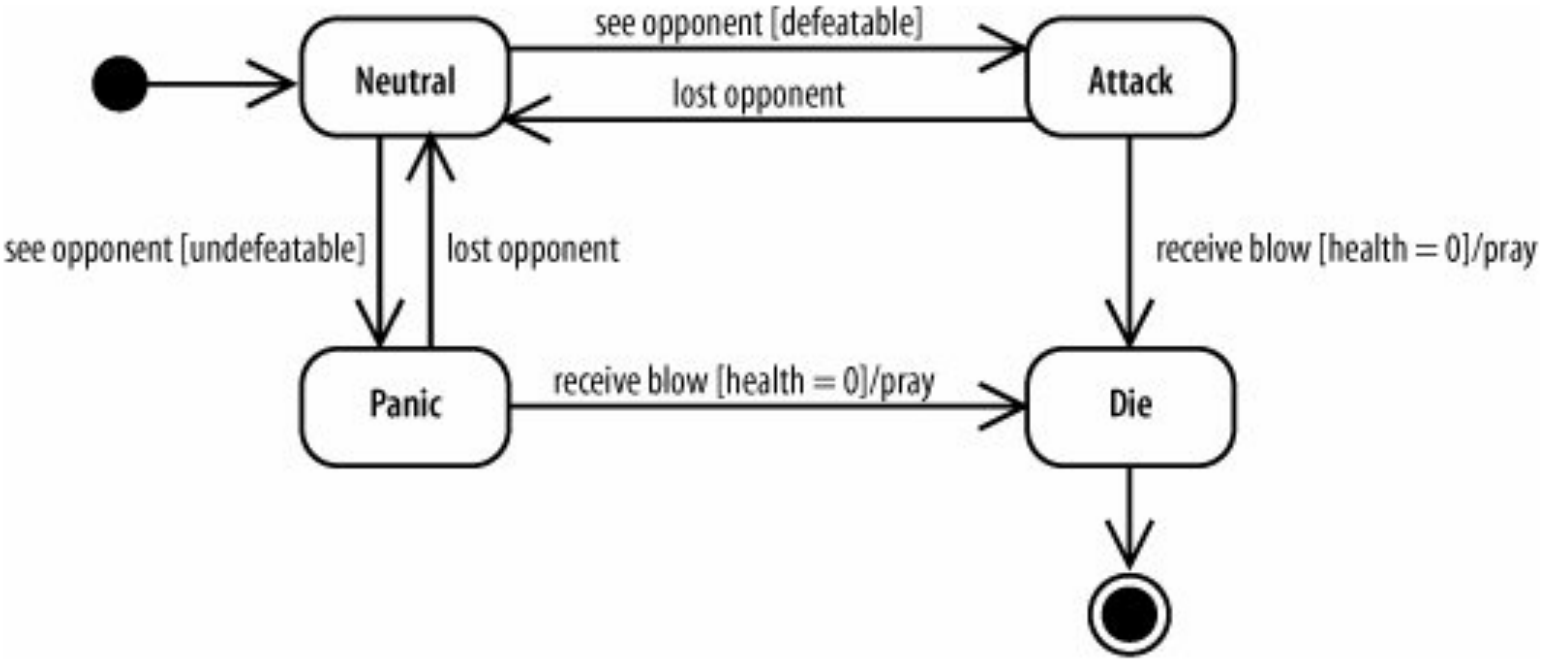


Figure 14-12. The life cycle of an AccountApplication object

Figure 14-13. State diagram modeling a troll in a FPS game; the troll's behavior is determined by his state



14.5. Advanced State Behavior

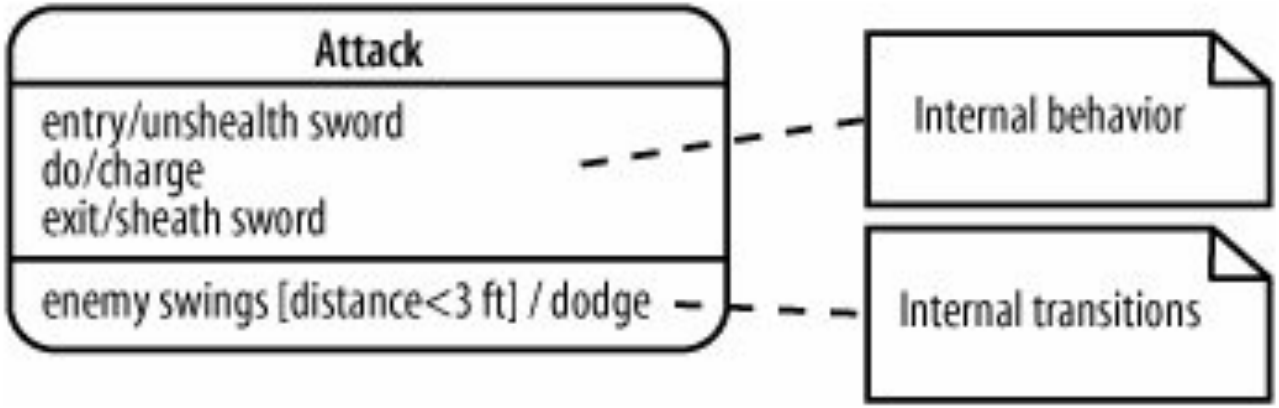


Figure 14-14. Internal behavior and transitions of the Attack state

14.5.1. Internal Behavior

- Internal behavior is any behavior that happens while the object is in a state.
- Internal behavior is written as ***label / behavior***.
  - do, entry, exit
- Unlike ***do*** behavior, ***entry*** and ***exit*** behaviors can't be interrupted.

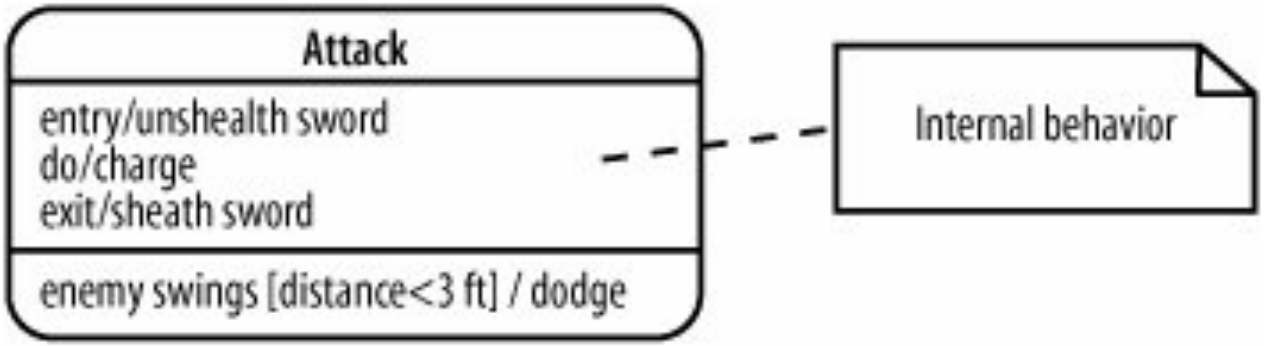


Figure 14-15. The middle compartment shows internal behavior

14.5.2. Internal Transitions

- An internal transition is a transition that causes a **reaction** within a state, but doesn't cause the object to change states.
- An internal transition is **different** from a self transition because self transitions cause entry and exit behavior to occur whereas internal transitions don't.

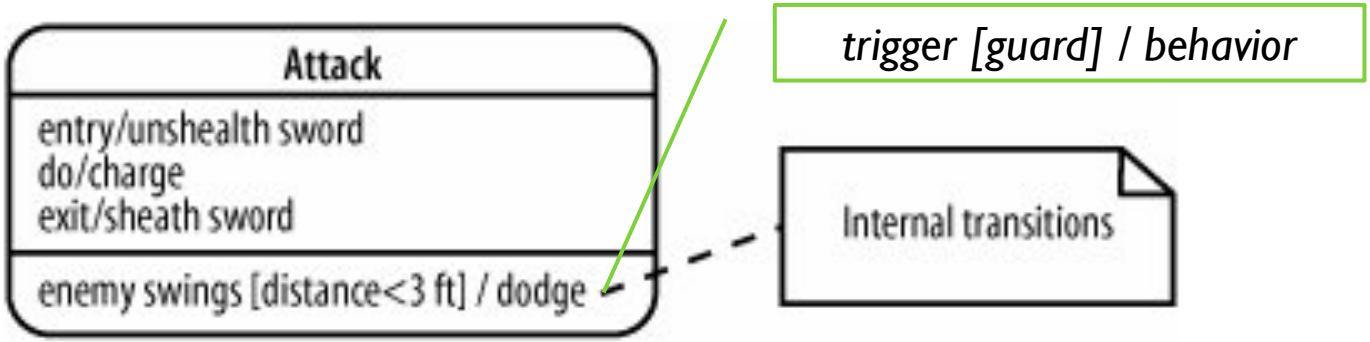


Figure 14-16. The bottom compartment shows internal transitions

14.5.2. Internal Transitions (Cont.)

- Use internal transitions to **model reactions** to events that **don't** cause state changes.
  - For example, you could use internal transitions to show that a pause-and-serve coffee-maker suspends dispensing the coffee when you remove the coffee pot but doesn't leave the Brewing state,

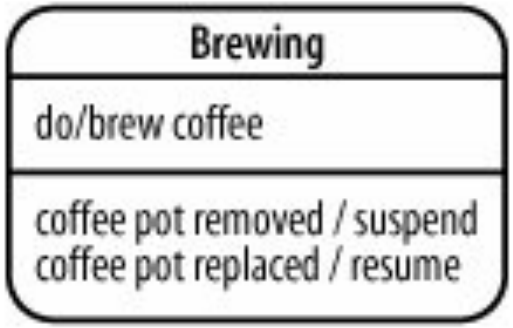


Figure 14-17. An internal transition models a reaction while staying in the same state

14.6. Composite States

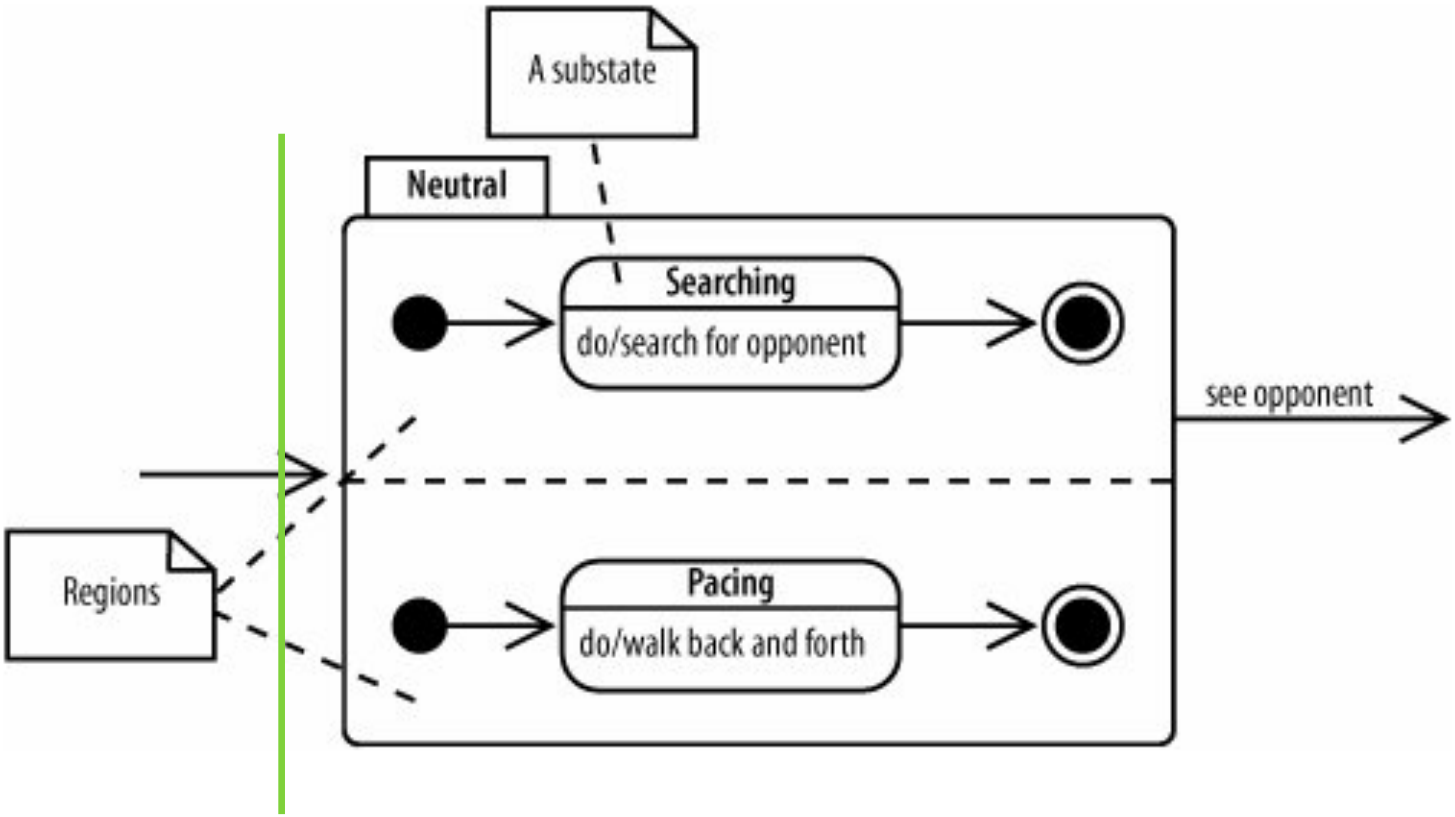


Figure 14-18. Composite states contain **one or more** state diagrams; if they contain more than one state diagram, then the state diagrams **execute in parallel**

14.7. Advanced Pseudostates

- A ***choice*** pseudostate is used to emphasize that a Boolean condition determines which transition is followed.
- A choice has **guards** on each of its outgoing transitions, and the transition that is followed depends on the guard.

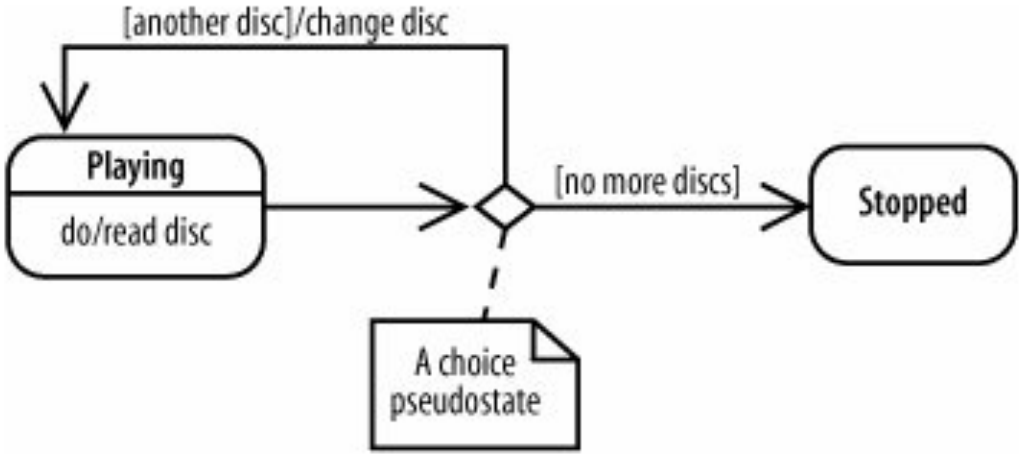


Figure 14-19. The path followed after a choice depends on the guard

14.7. Advanced Pseudostates

- ***Fork*** and ***join*** pseudostates show branching into concurrent states and then rejoining.

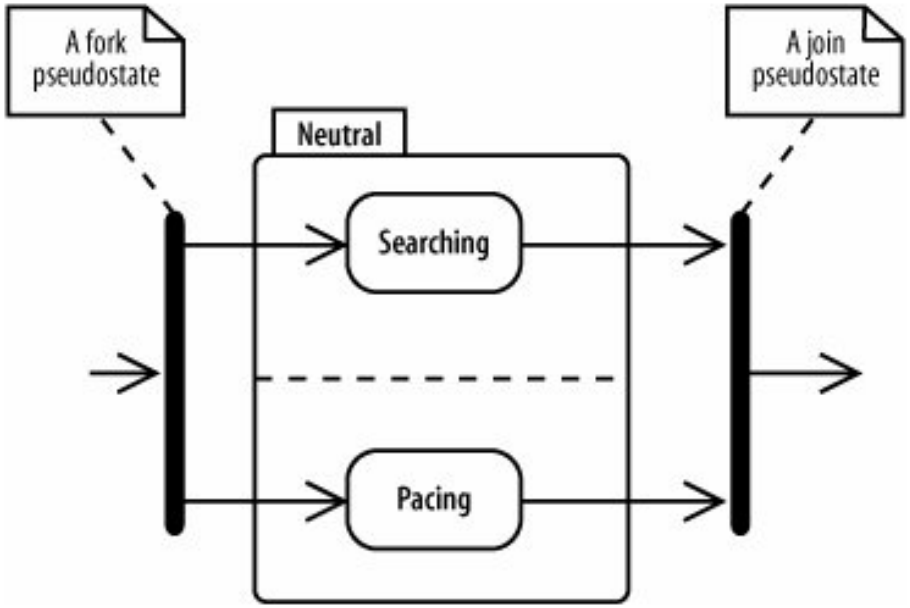


Figure 14-20. Forks and joins show concurrent states

14.8. Signals

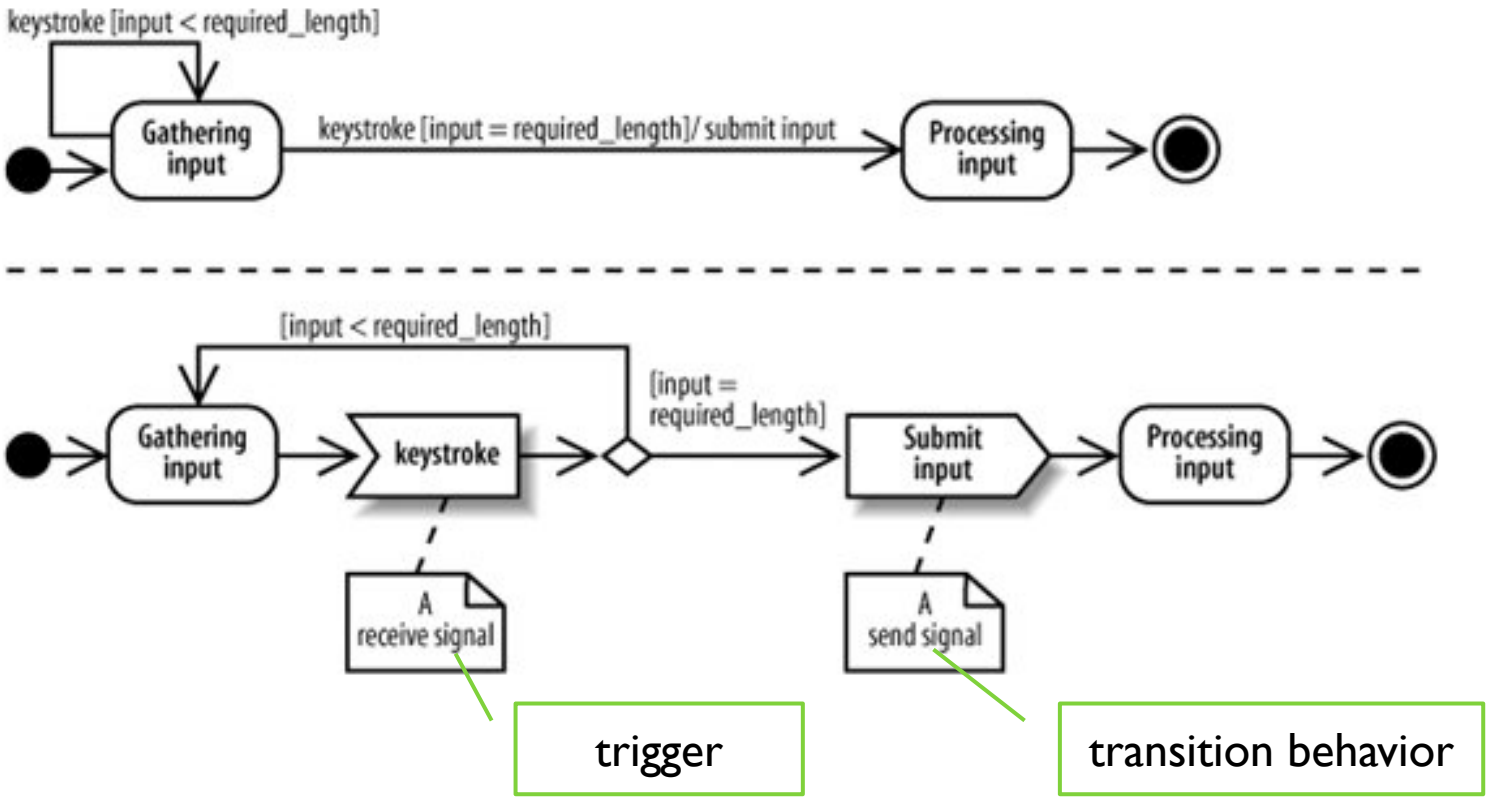


Figure 14-21. The bottom diagram draws transitions and transition behavior as receive and send signals

14.9. Protocol State Machines

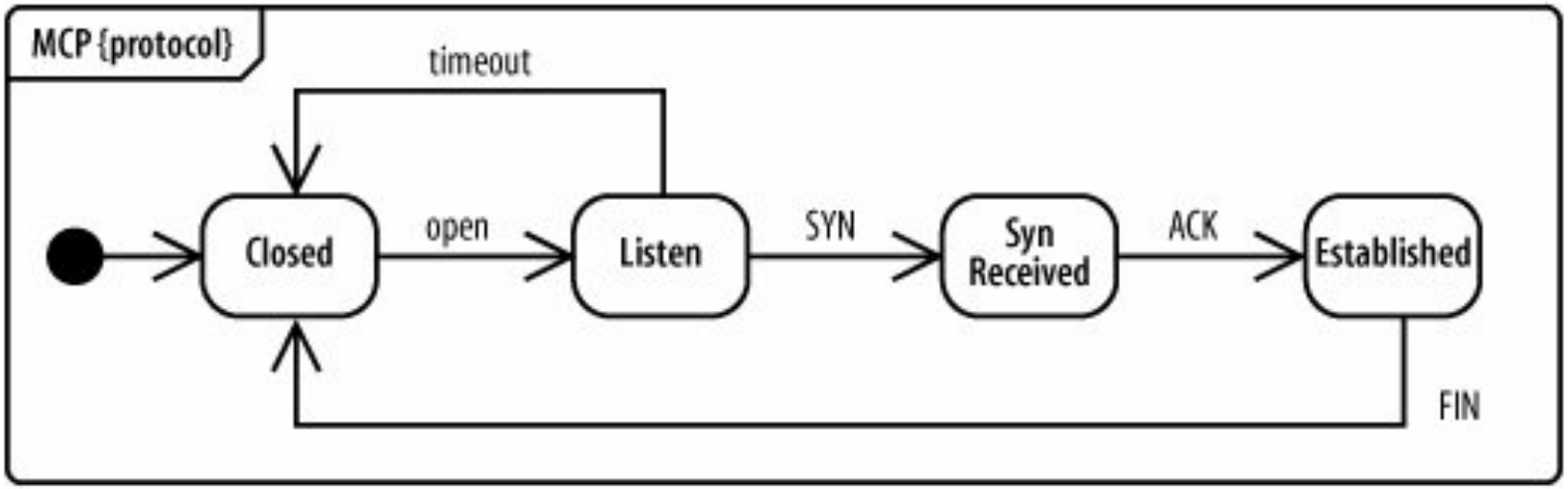


Figure 14-22. Protocol state machine modeling the receiver side of a simplified communication protocol called My Communication Protocol (MCP)

14.9. Protocol State Machines

- Protocol state machines are a **special kind of** state machine focusing on how a protocol, such as a communication protocol (e.g., TCP), works.
- The main **difference** between protocol state machines and behavioral state machines is that protocol state machines don't show behavior along transitions or inside states. Instead, they focus on showing a legal sequence of events and resulting states.
- Protocol state machines are drawn in a **tabbed** rectangle with the name of the state machine in the tab followed by **{protocol}**.

Summary

14. Modeling an Object's State:
    State Machine Diagrams

14.9. Protocol State Machines

14.4. States in Software
14.5. Advanced State Behavior
14.6. Composite States
14.7. Advanced Pseudostates
14.8. Signals

14.1. Essentials
14.2. States
14.3. Transitions

状态图练习

绘制状态图

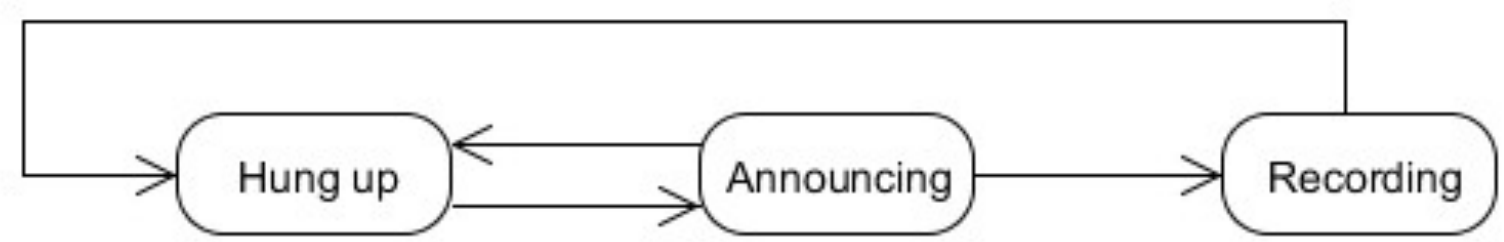
- 伸缩梯上有绳子、滑轮和插销，可以升起、放下和锁住。当插销锁住时，伸缩梯被固定住，就可以安全地爬上梯子。想要松开插销，就要用绳子稍微把伸缩梯提起来，然后就可以自由地升降伸缩梯。当插销穿过梯子的横档时，会发出噼啪的声响。在反方向升起伸缩梯时，插销会重新啮合，就像插销正在穿过横档一样。绘制一个伸缩梯的状态图。

绘制状态图

- 简单的数字手表上面有一个显示屏和两个设置按钮，即按钮A和按钮B。此表有两种操作模式：显示时间和设定时间。在显示时间模式下，手表会显示小时和分钟，小时和分钟由闪烁的冒号分隔。
- 设定时间模式有两种子模式：设定小时和设定分钟。按钮A选择模式。每次按下此按钮时，模式会连续前进：显示、设定小时、设定分钟和显示分钟等。在子模式内，每次只要按下按钮B，就会拨快小时或分钟。在按钮生成另一个事件之前，必须释放它们。
- 绘制一个数字手表的状态图。

填补状态图

- 下图是部分完成的简化状态图，描述电话应答机的控制过程。首次响铃时，应答机会检测到达呼叫，然后用预告录制好的应答回复呼叫。当应答完成时，机器会记录呼叫端的消息。当呼叫端挂断时，机器也会挂断并关闭。在图中安排下面的这些内容：
  - 检测到的呼叫（call detected）、应答呼叫（answer call）、播放应答（play announcement）、记录消息（record message）、呼叫端挂断（caller hang-up）、应答结束（announcement complete）



修改状态图

- 前一题中的电话应答机在第一次呼叫时就会激活。修改状态图，使得机器在五次响铃之后才应答。如果有人在五次响铃之前应答了电话，机器什么都不会做。注意区分电话在首次响铃过程中的五次呼叫与响铃五次的一次呼叫。

See you …