3. Modeling System Workflows: Activity Diagrams

Shaoning Zeng

UNIFIED MODELING LANGUAGE™

What's Learned?

- 2. Modeling requirements
  - 2.1. Capturing a System Requirement
    - actor
    - use case
  - 2.2. Use Case Relationships
    - <<include>>
    - <<extend>>
  - 2.3. Use Case Overview Diagrams

3. Modeling System Workflows: Activity Diagrams

- 3.1. Activity Diagram Essentials 概要
- 3.2. Activities and Actions 活动与操作
- 3.3. Decisions and Merges 决策与合并
- 3.4. Doing Multiple Tasks at the Same Time 多任务并发
- 3.5. Time Events 时间事件
- 3.6. Calling Other Activities 调用活动
- 3.7. Objects 对象
- 3.8. Sending and Receiving Signals 发送信号与接收信号
- 3.9. Starting an Activity 启动活动
- 3.10. Ending Activities and Flows 结束活动与结束流程
- 3.11. Partitions (or Swimlanes) 分区|泳道
- 3.12. Managing Complex Activity Diagrams

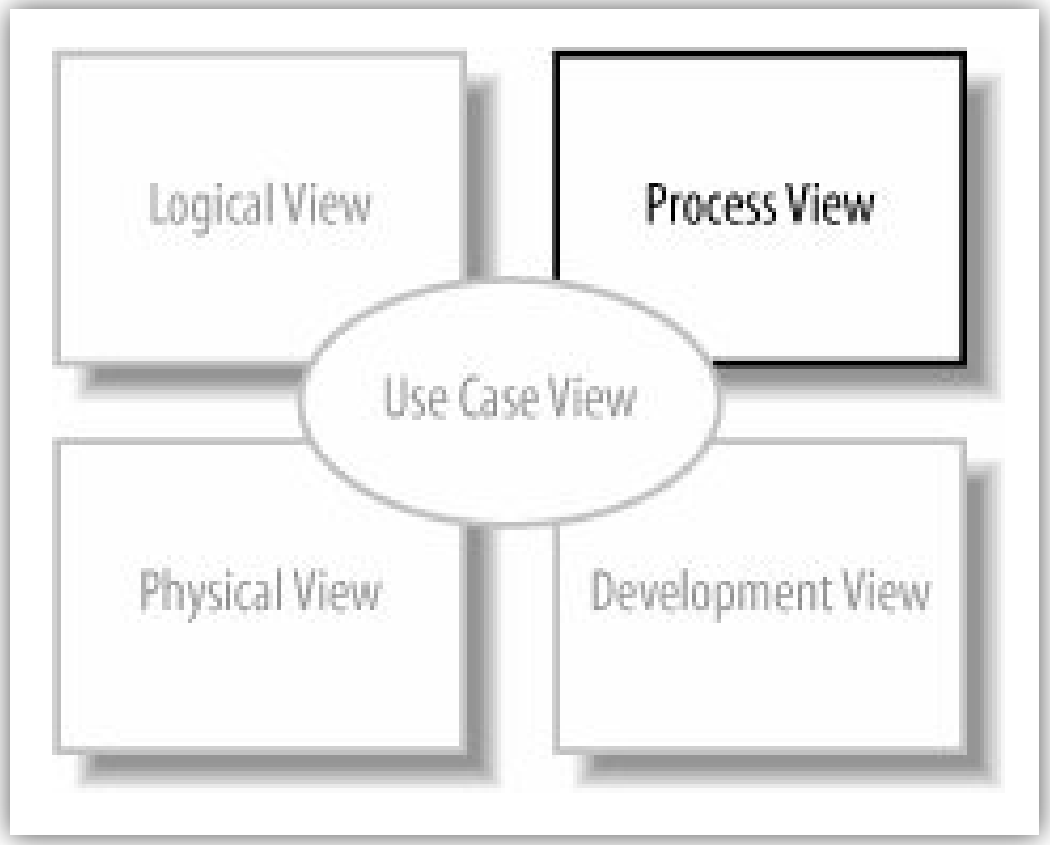
Activity diagrams

- Use cases show what your system should do. 做什么
- Activity diagrams allow you to specify how your system will accomplish its goals. 怎么做
- Activity diagrams show high-level actions chained together to represent a process occurring in your system.
  - For example, you can use an activity diagram to model the steps involved with creating a blog account. 步骤

Business processes

- Activity diagrams are particularly good at modeling *business processes* . 业务过程
- A business process is a set of coordinated tasks that achieve a business goal, such as shipping customers‘ orders. 业务目标
- Some business process management (BPM) tools allow you to define business processes using activity diagrams, or a similar graphical notation, and then execute them.
  - This allows you to define and execute, for example, a payment approval process where one of the steps invokes a credit card approval web service using an easy graphical notation such as activity diagrams.

Process View

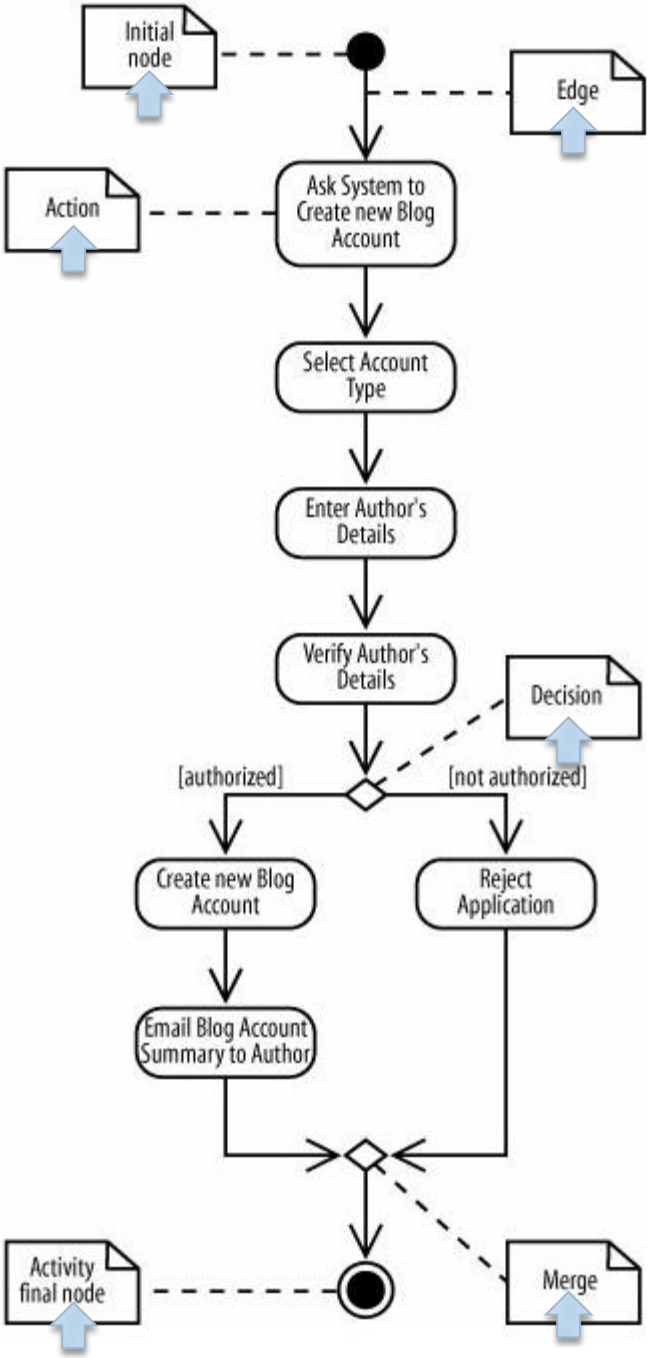


3.1. Activity Diagram Essentials

Table 3-1. Create a new Blog Account use case description

Use case name	**Create a new Blog Account**	
Related Requirements	Requirement A.1.	
Goal In Context	A new or existing author requests a new blog account from the Administrator.	
Preconditions	The system is limited to recognized authors, and so the author needs to have appropriate proof of identity.	
Successful End Condition	A new blog account is created for the author.	
Failed End Condition	The application for a new blog account is rejected.	
Primary Actors	Administrator.	
Secondary Actors	Author Credentials Database.	
Trigger	The Administrator asks the Content Management System to create a new blog account.	
Main Flow	Step	Action
	1	The Administrator asks the system to create a new blog account.
	2	The Administrator selects an account type.
	3	The Administrator enters the author's details.
	4	The author's details are verified using the Author Credentials Database.
	5	The new blog account is created.
	6	A summary of the new blog account's details are emailed to the author.
Extensions	Step	Branching Action
	4.1	The Author Credentials Database does not verify the author's details.
	4.2	The author's new blog account application is rejected.

Figure 3-2. del
dynamic bel n processes;
the basic ele grams are
shown in th ion process



3.2. Activities and Actions

- Actions are active steps in the completion of a process.
  - An action can be a calculation, such as Calculate Tax, or a task, such as Verify Author's Details. 操作 》 步骤
- An activity is the process being modeled, such as washing a car. 活动》 过程
- An action is a step in the overall activity, such as Lather (涂肥皂), Rinse (清洗), and Dry (擦干).

Figure 3-3. Capturing the three actions Lather, Rinse, and Dry that make up washing a car in an activity diagram

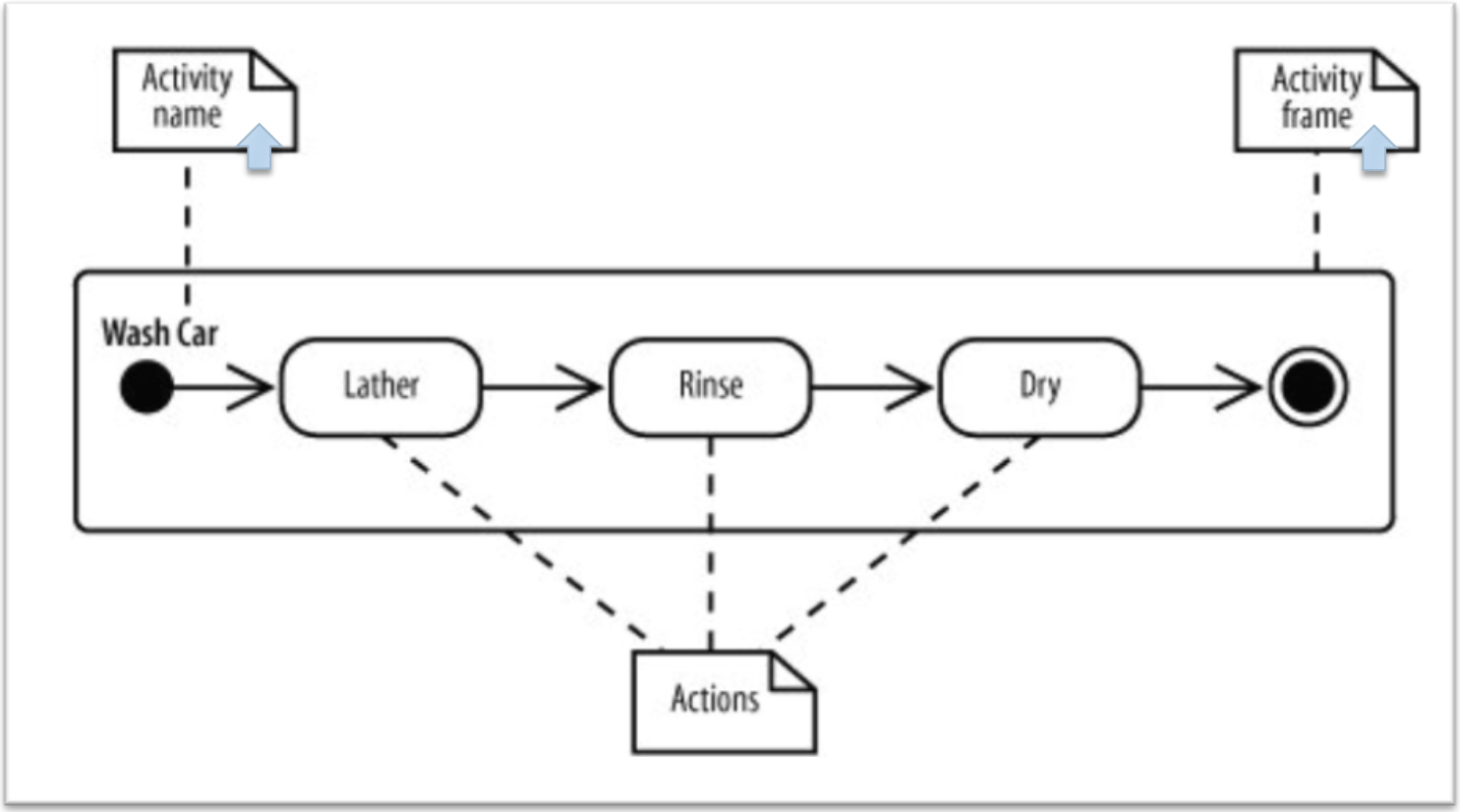
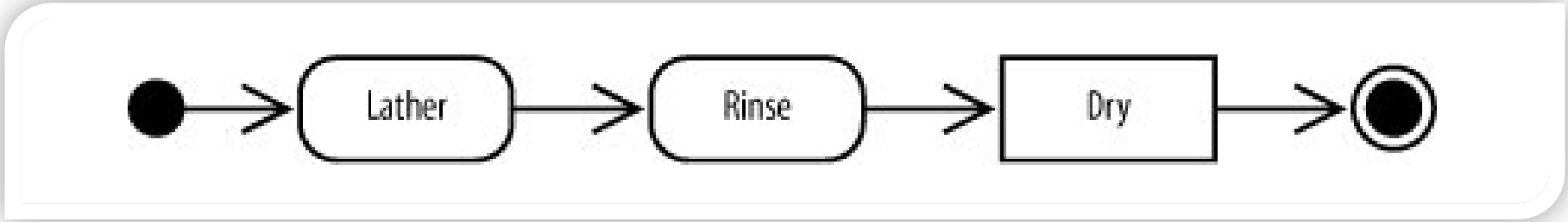


Figure 3-4. The activity frame can be omitted



3.3. Decisions and Merges

- *Decisions* are used when you want to execute a different sequence of actions depending on a condition.
  - 决策》条件判断
- Decisions are drawn as diamond-shaped nodes with one incoming edge and multiple outgoing edges

Figure 3-5. Only one edge is followed after a decision node

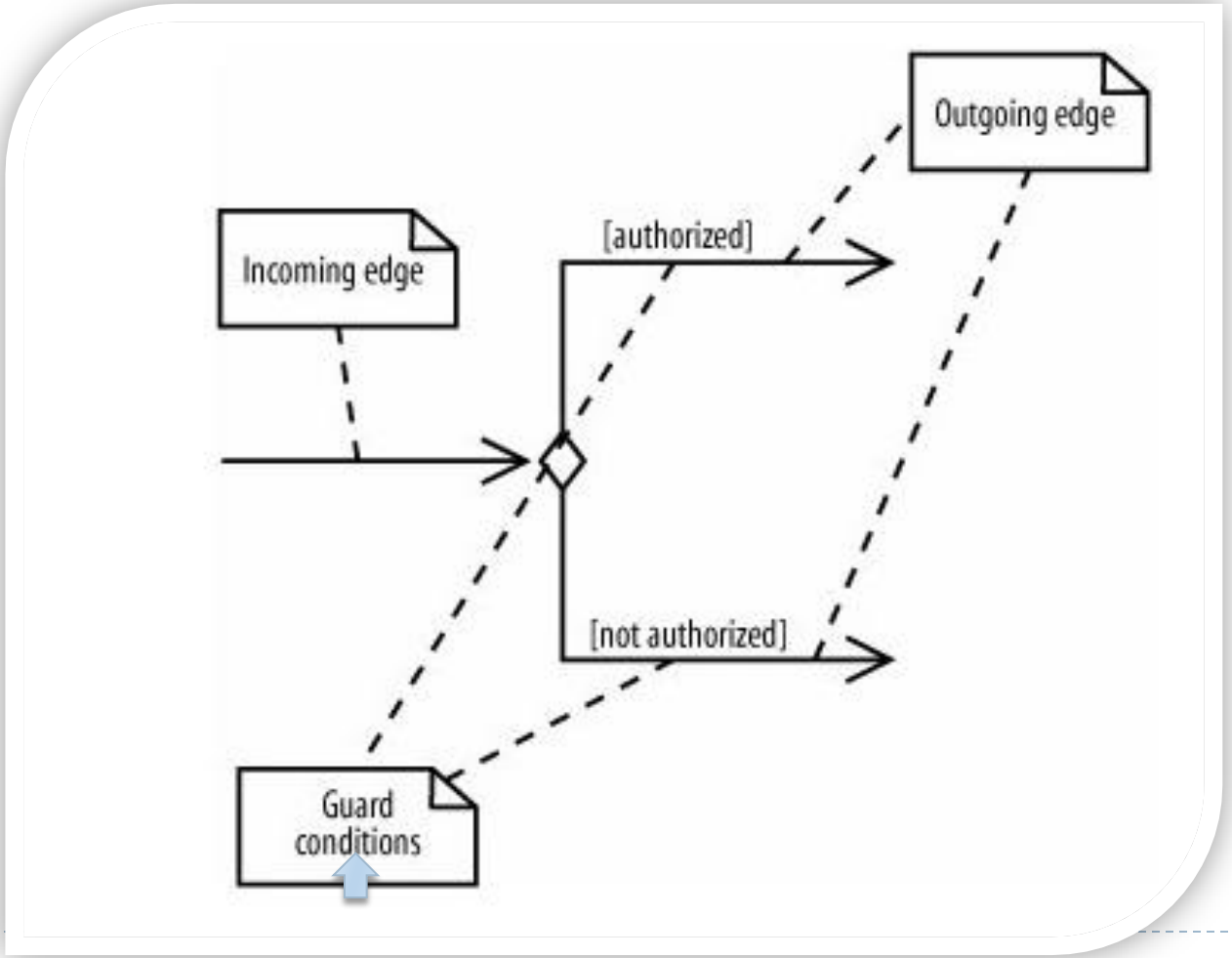


Figure 3-6. If the input value of age is 1200, then the Notify Blog Entry too long action is performed

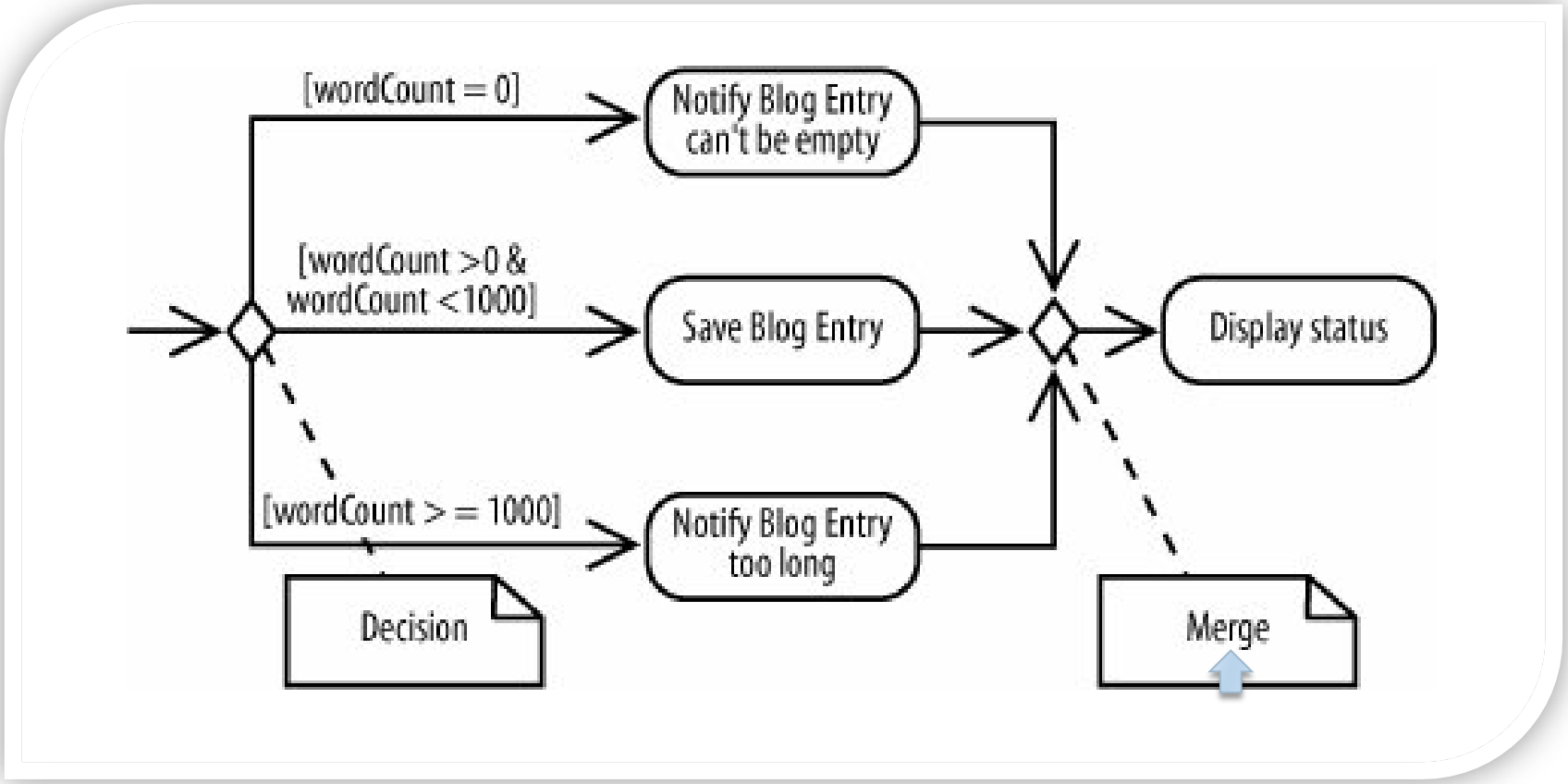
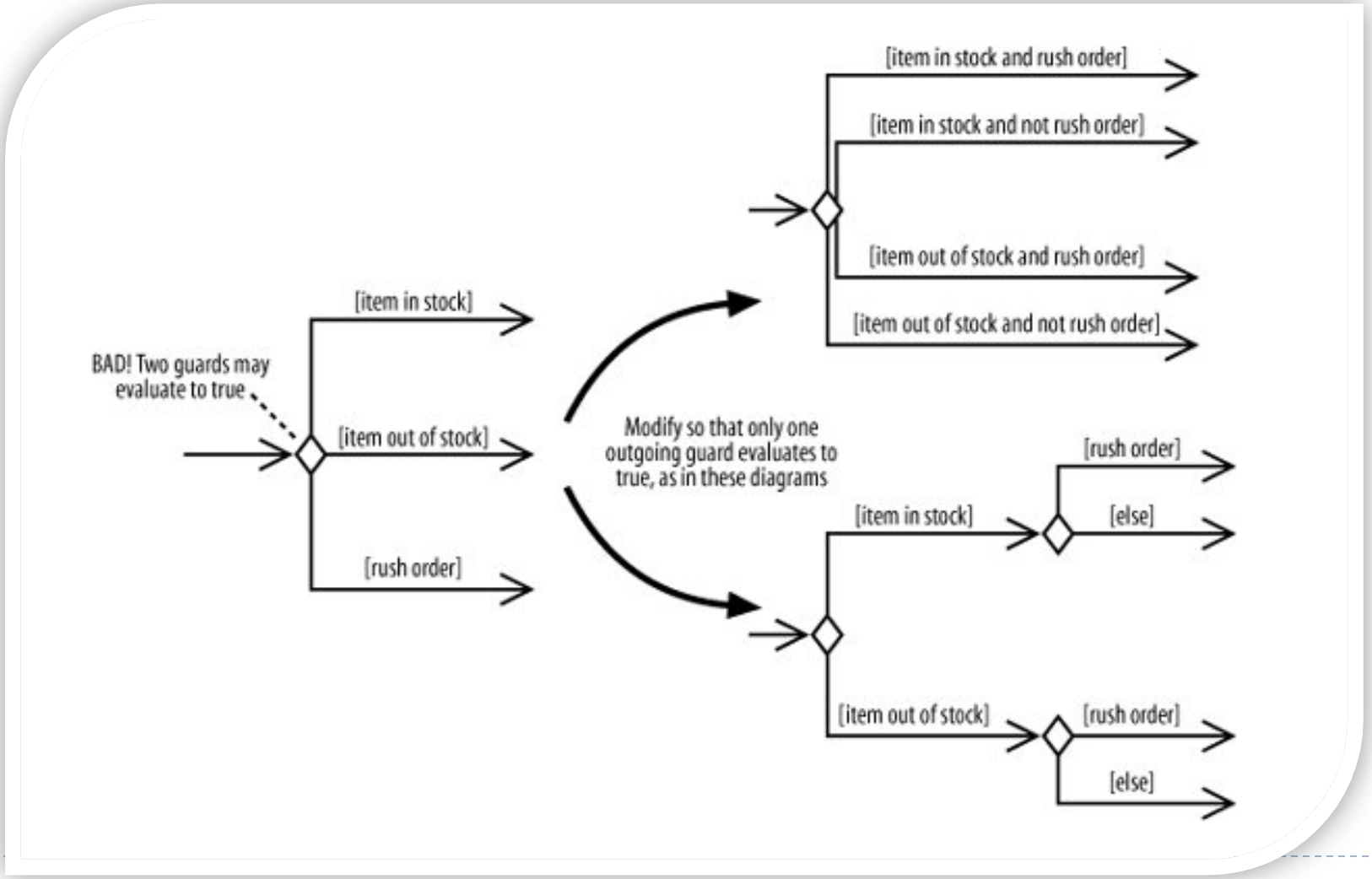


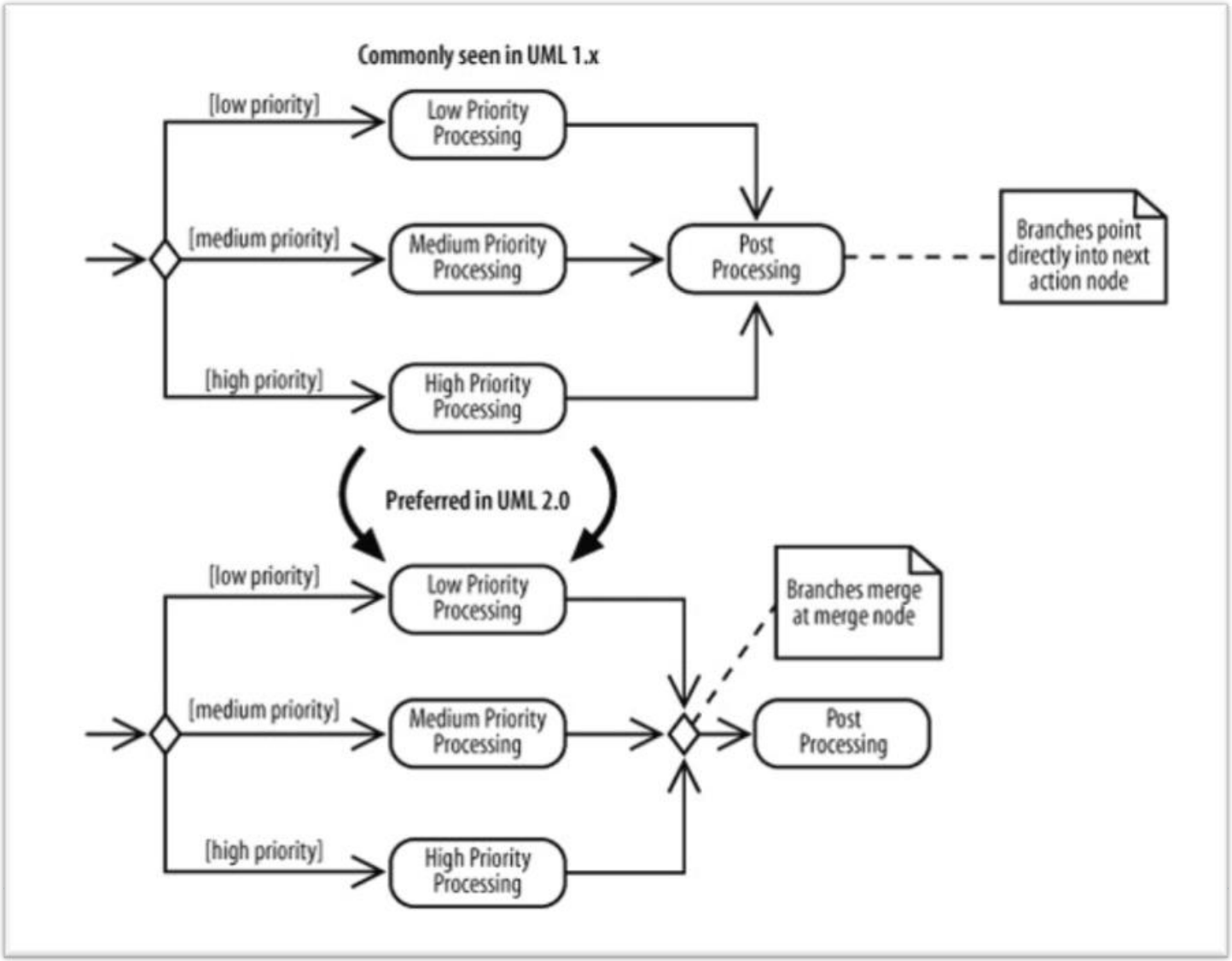
Figure 3-7. Beware of diagrams where multiple guards evaluate to true



3.4. Doing Multiple Tasks at the Same Time

- Consider a computer assembly workflow that involves the following steps: 组装电脑
  1. Prepare the case.
  2. Prepare the motherboard.
  3. Install the motherboard.
  4. Install the drives.
  5. Install the video card, sound card, and modem.
- Steps that occur at the same time are said to occur concurrently or in parallel. 并行步骤

Figure 3-8. In UML 2.0, it's better to be as clear as possible and to show merge nodes



forks and joins for parallel actions

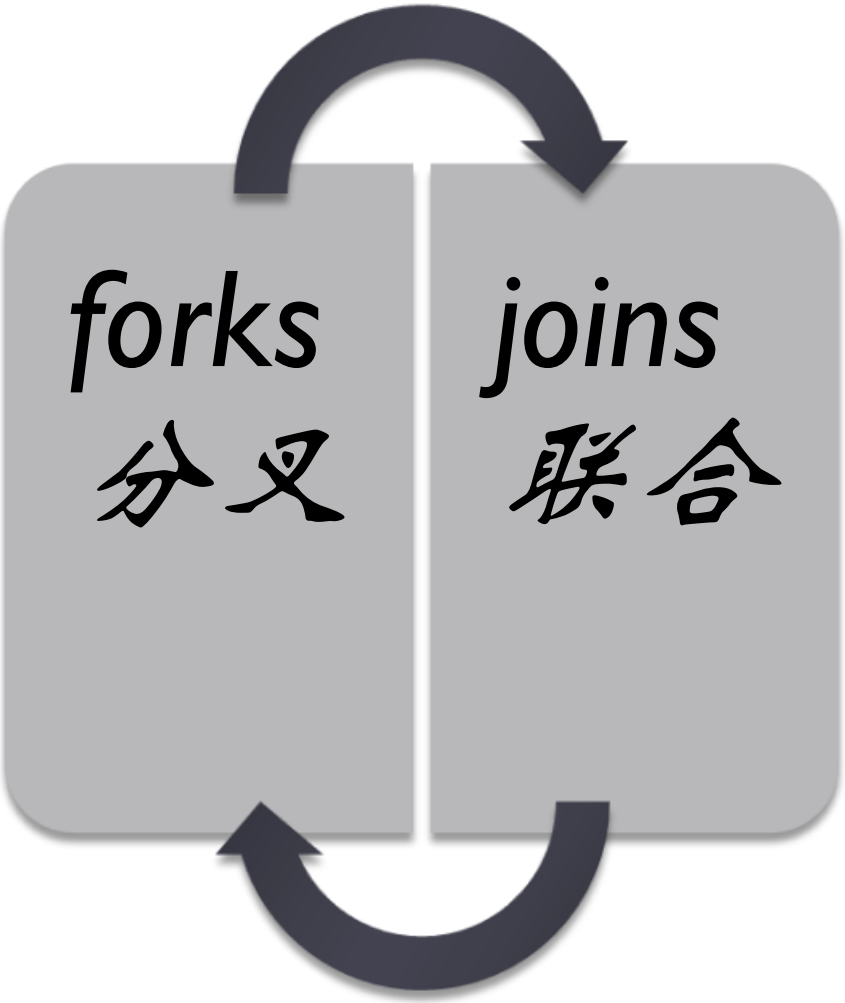


Figure 3-9. Both outgoing paths are followed at the fork, in contrast with decision nodes, where only one outgoing path is taken

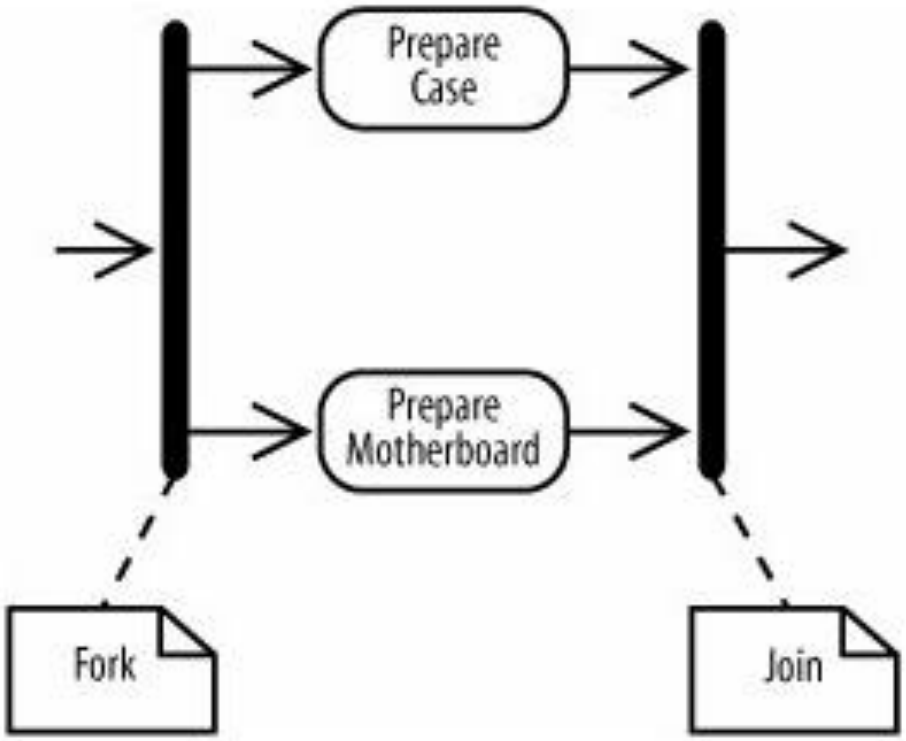
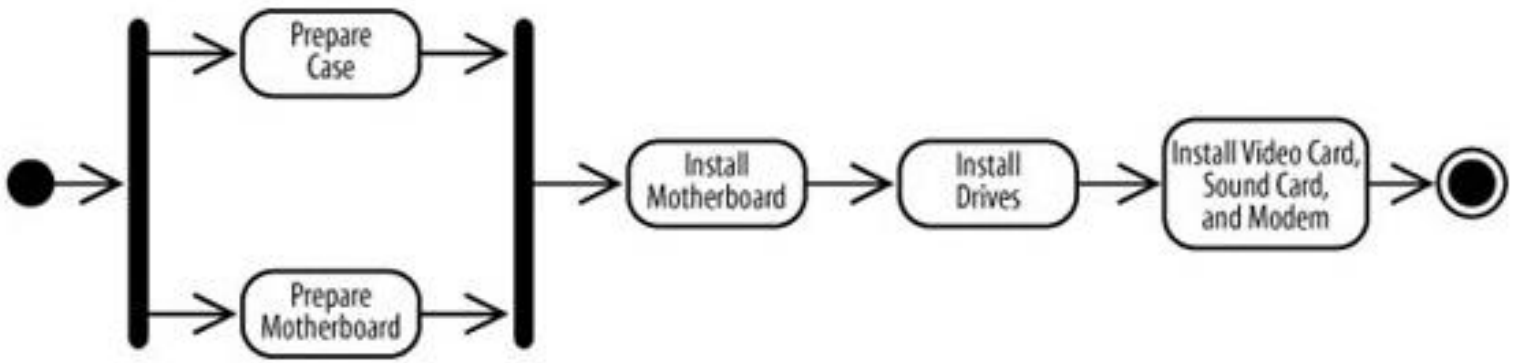


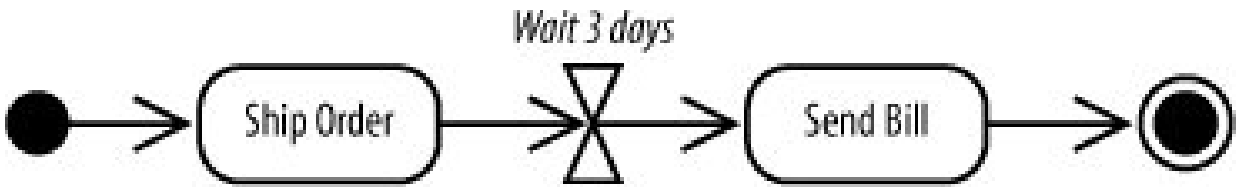
Figure 3-10. The computer assembly workflow demonstrates how forks and joins work in a complete activity diagram



3.5. Time Events

- Sometimes time is a factor in your activity. 时间
- You may want to model a wait period, such as 等待时间
  - waiting three days after shipping an order to send a bill.
- You may also need to model processes that kick off at a regular time interval, such as 轮循时间
  - a system backup that happens every week.
- *Time events* are drawn with an hourglass (沙漏) symbol.

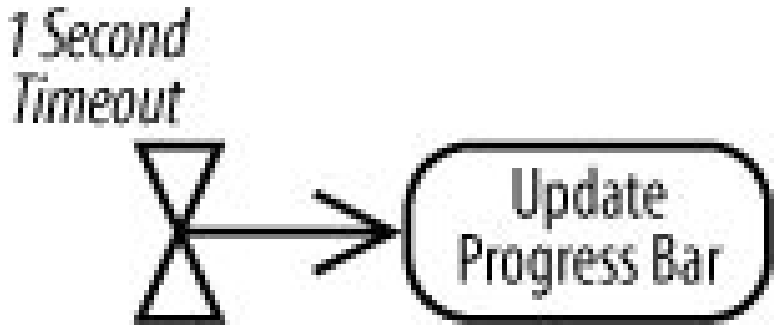
Figure 3-11. A time event with an incoming edge represents a timeout



recurring time event (循环时间事件)

- A time event with no incoming flows is a recurring time event, meaning it's activated with the frequency in the text next to the hourglass.
  - 没有到达流
- Notice that there is no initial node in Figure 3-12; a time event is an alternate way to start an activity.
  - 没有初始节点。时间事件也是启动活动的一种方式。
  - Use this notation to model an activity that is launched periodically. 周期性执行的活动

Figure 3-12. A time event with no incoming flows models a repeating time event



3.6. Calling Other Activities 调用其他活动

- As detail is added to your activity diagram, the diagram may become too big, or the same sequence of actions may occur more than once.
- When this happens, you can improve readability by providing details of an action in a separate diagram, allowing the higher level diagram to remain less cluttered.
- A call activity node calls the activity corresponding to its node name. This is similar to calling a software procedure.

Figure 3-13. Rather than cluttering up the top-level diagram with details of the Prepare Motherboard action, details are provided in another activity diagram

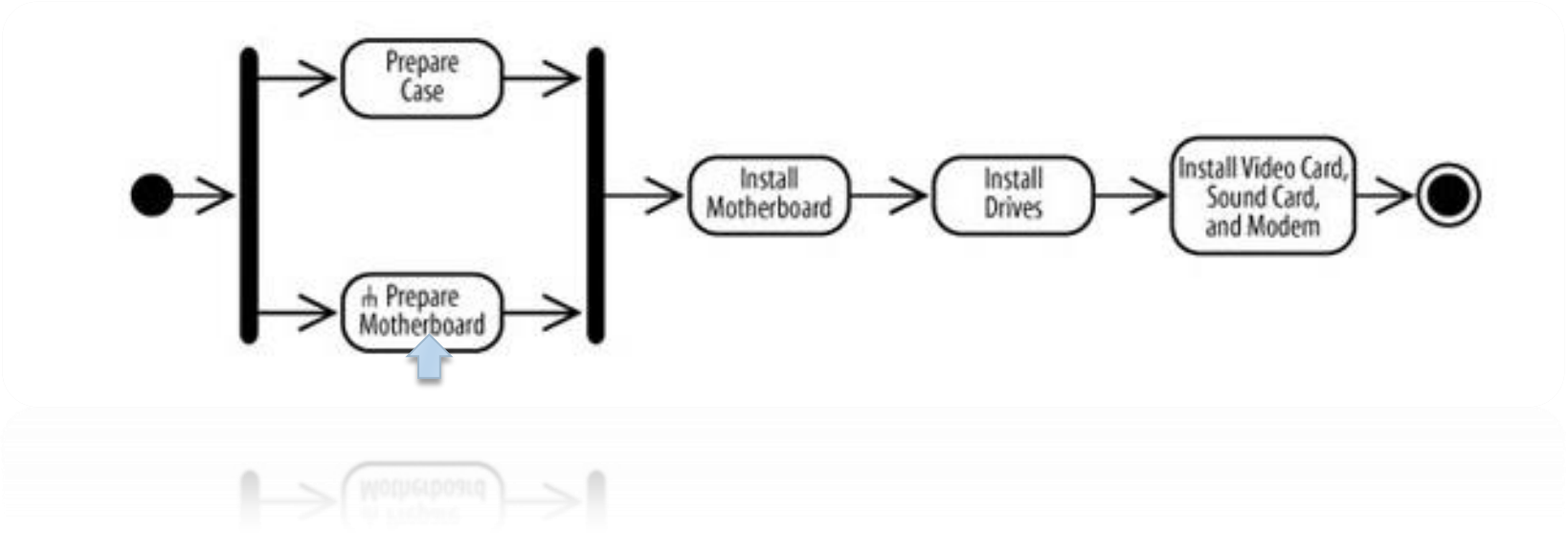
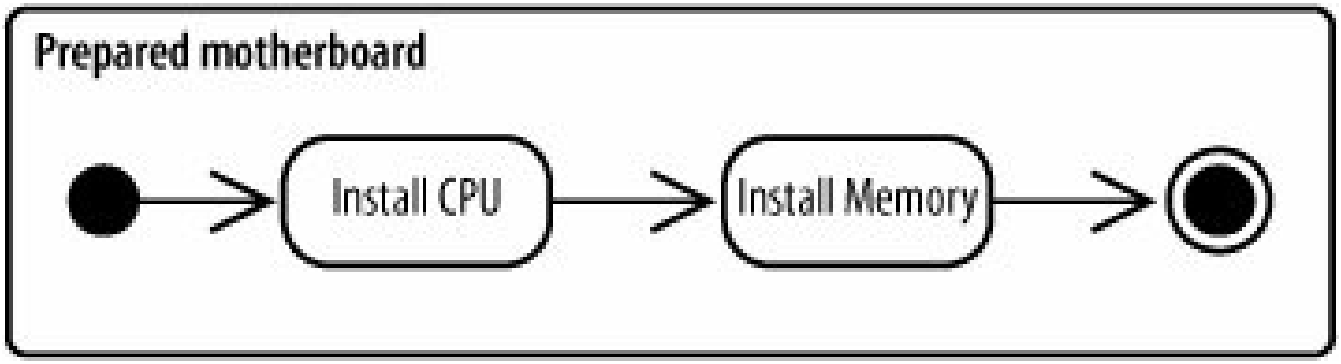


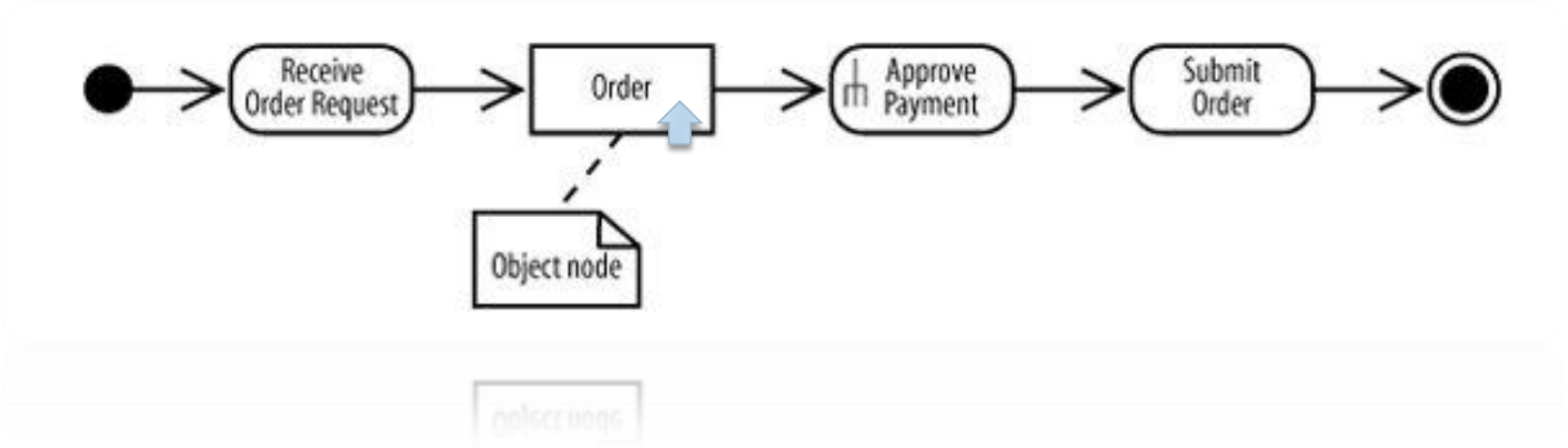
Figure 3-14. The Prepare Motherboard activity elaborates on the motherboard preparation process



3.7. Objects 对象

- 3.7.1. Showing Objects Passed Between Actions
  - An object node represents an object that is available at a particular point in the activity,
    - 活动过程中特定时刻出现的对象
  - and can be used to show that the object is used, created, or modified by any of its surrounding actions.
  - An object node is drawn with a rectangle. 矩形

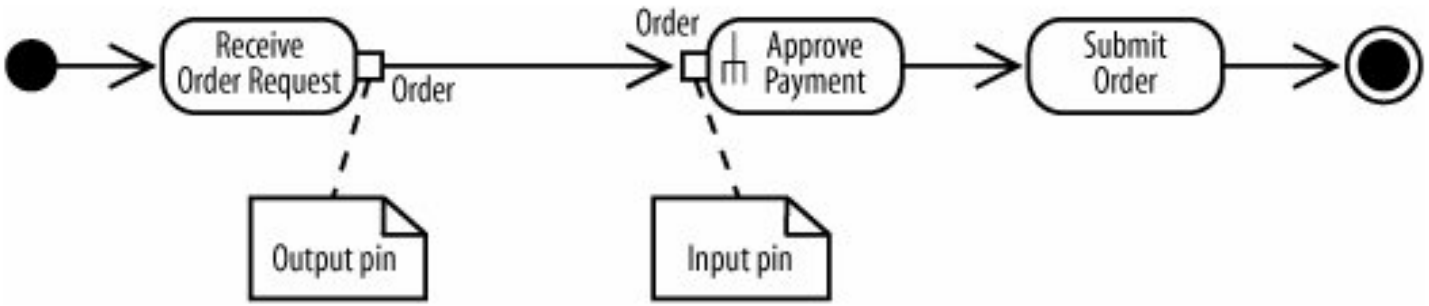
Figure 3-15. The Order object node emphasizes that it is important data in this activity and shows which actions interact with it



3.7. Objects

- 3.7.2. Showing Action Inputs and Outputs
  - Pins show that an object is input to or output from an action.
  - An *input pin* means that the specified object is input to an action.
  - An *output pin* means that the specified object is output from an action.

Figure 3-16. Pins in this change request approval process allow finer-grained specification of input and output parameters



Comparison

Object nodes

- object nodes are good at emphasizing the flow of data through an activity.

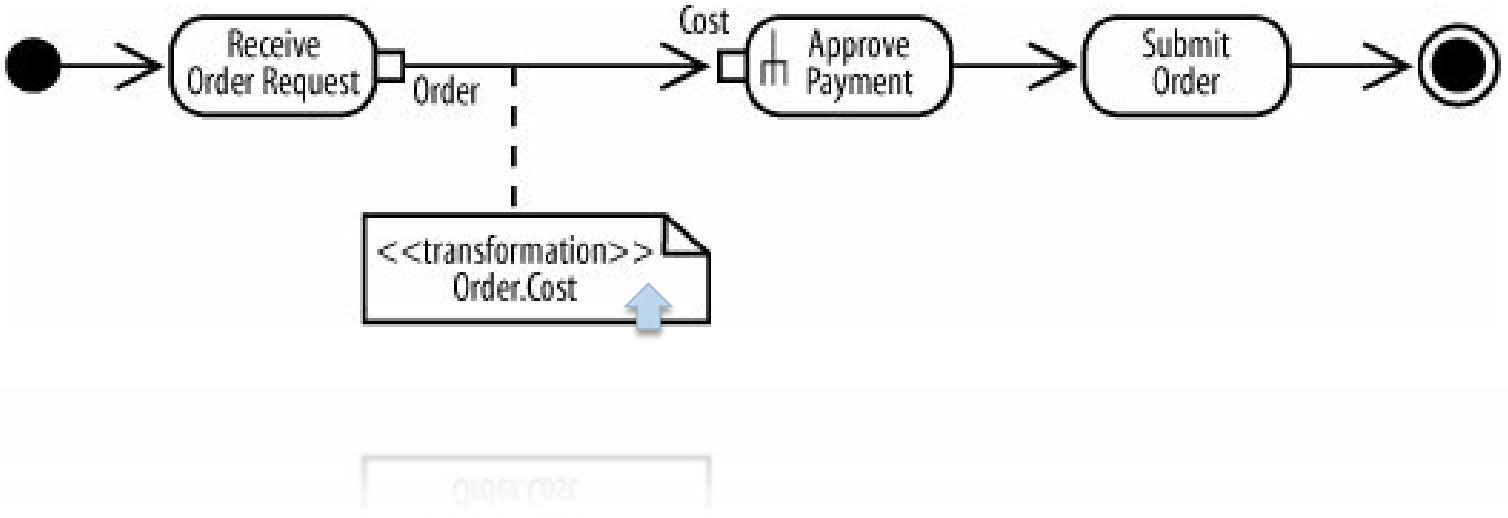
Pins

- pins are good at emphasizing that an object is required input and output

Transformation

- If the Approve Payment action needs only parts of the Order object not the whole object, you can use a *transformation* to show which parts are needed.
  - 对象的转换
- Transformations allow you to show how the output from one action provides the input to another action.
  - 一个操作的输出将作为另一个操作的输出

Figure 3-17. Transformations show where input parameters come from



3.7. Objects

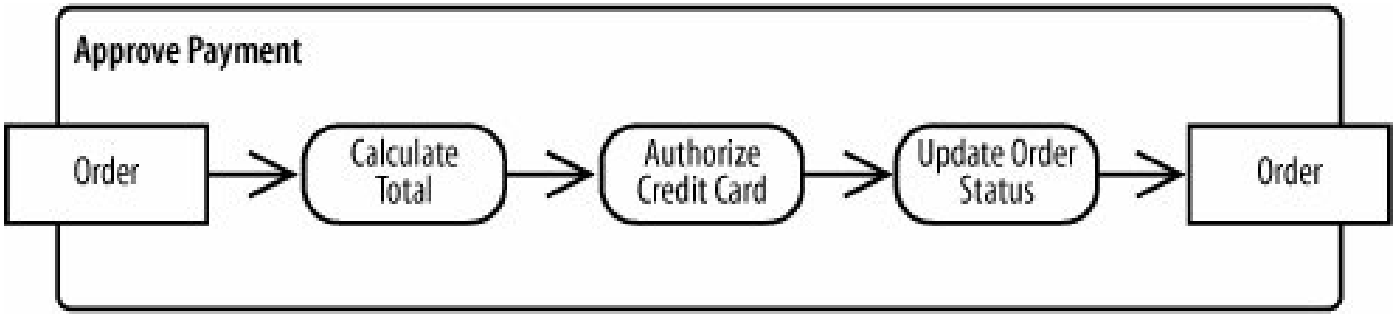
- 3.7.3. Showing How Objects Change State During an Activity 活动中对象状态的变化

Figure 3-18. The focus of this diagram is the change of state of the Order object throughout the order approval process

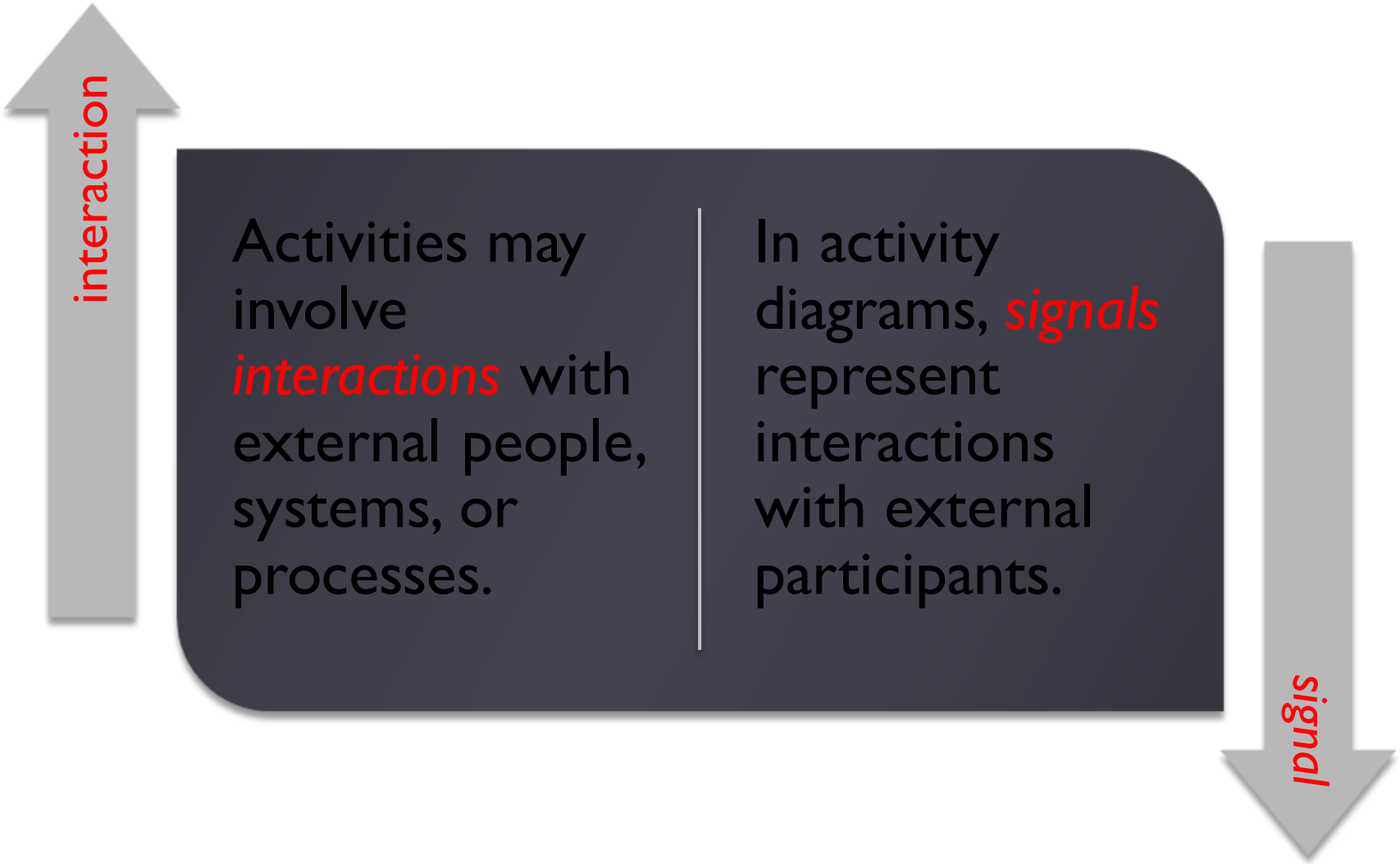
3.7. Objects

- 3.7.4. Showing Input to and Output from an Activity
  - In addition to acting as inputs to and outputs from actions, object nodes can be inputs to and outputs from an activity.
  - Activity inputs and outputs are drawn as object nodes straddling (跨) the boundary of the activity frame, as shown in Figure 3-19.
  - This notation is useful for emphasizing that the entire activity requires input and provides output.

Figure 3-19. Object nodes can be used to emphasize input to and output from an activity



3.8. Sending and Receiving Signals



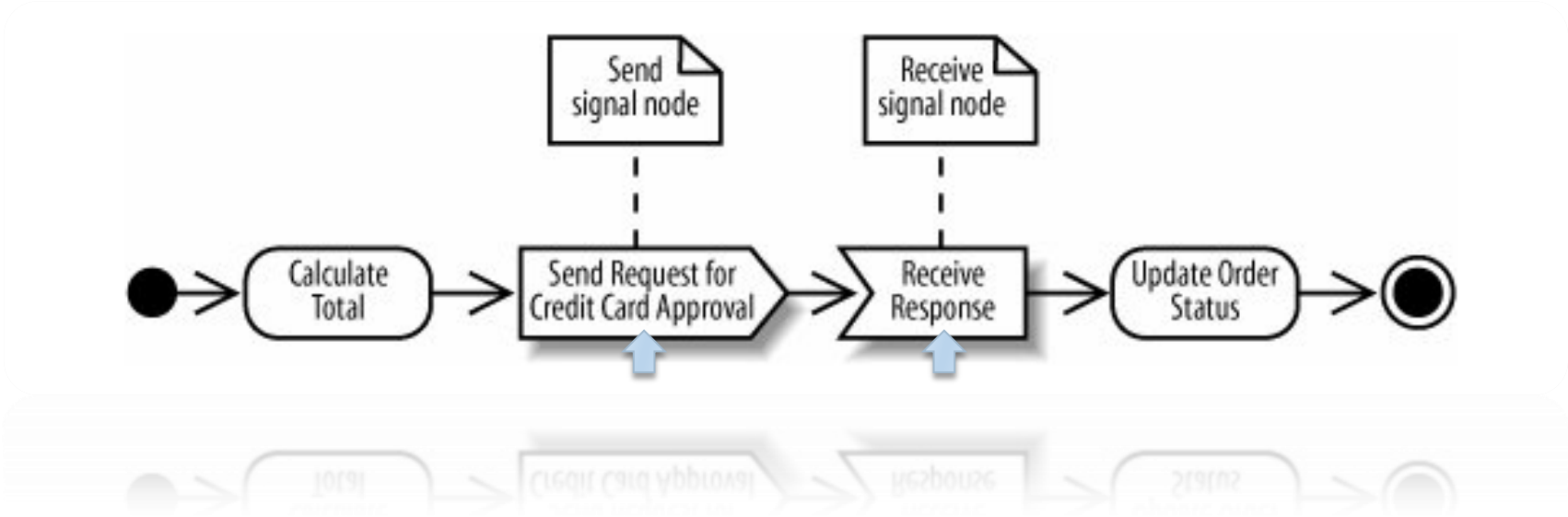
3.8. Sending and Receiving Signals

- Signals are messages that can be sent or received, as in the following examples: 信号就是消息
  - Your software sends a request to the credit card company to approve a credit card transaction, and your software receives a response from the credit card company (sent and received, from the perspective of your credit card approval activity).
  - The receipt of an order prompts an order handling process to begin (received, from the perspective of the order handling activity).
  - The click of a button causes code associated with the button to execute (received, from the perspective of the button event handling activity).
  - The system notifies a customer that his shipment has been delayed (sent, from the perspective of the order shipping activity).

3.8. Sending and Receiving Signals

- A *receive signal* has the effect of waking up an action in your activity diagram.
  - 接收信号
  - The recipient of the signal knows how to react to the signal and expects that a signal will arrive at some time but doesn't know exactly when.
- *Send signals* are signals sent to an external participant.
  - 发送信号
  - When that external person or system receives the message, it probably does something in response, but that isn't modeled in your activity diagram.

Figure 3-20. Send and receive signal nodes show interactions with external participants



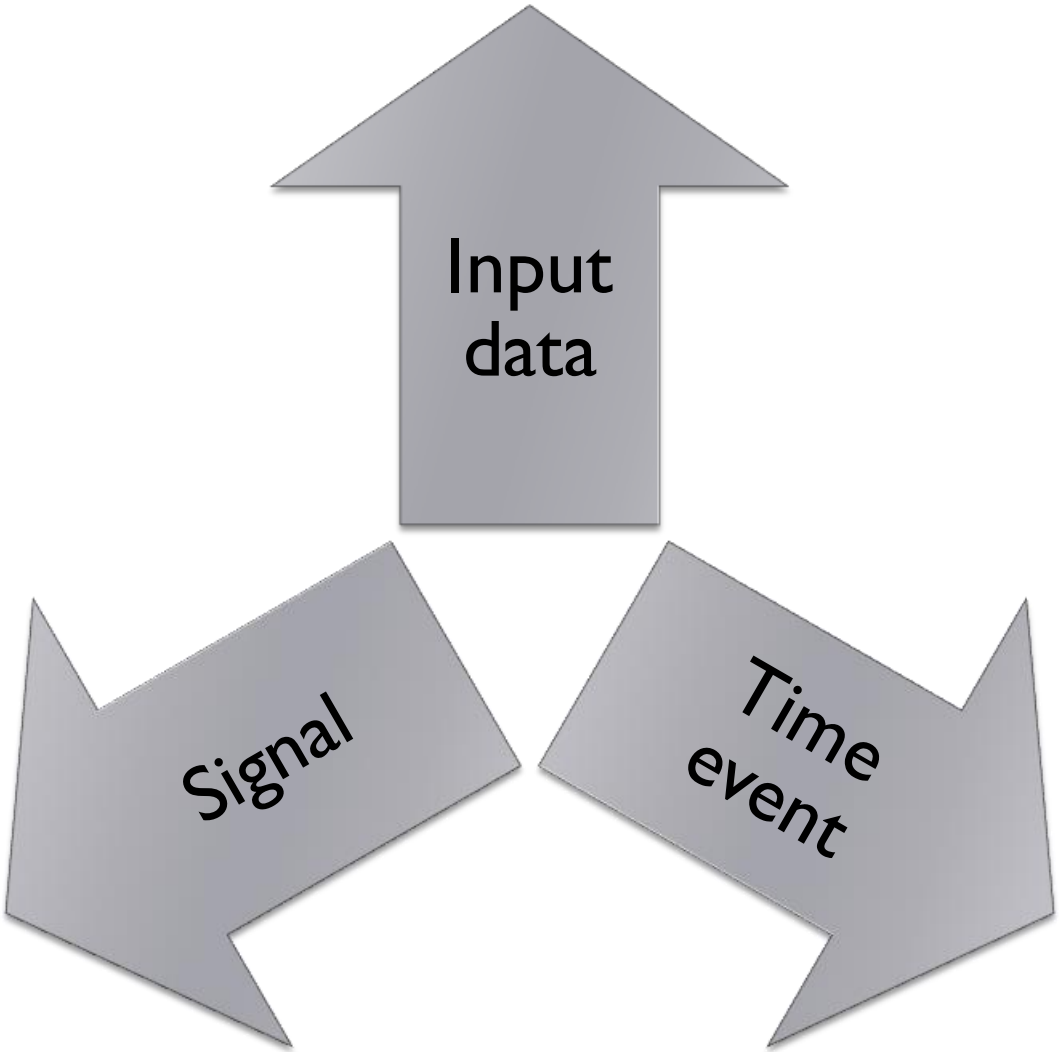
a receive signal node is always waiting

Figure 3-21. Starting an activity with a receive signal node: the receive signal node replaces the usual initial node

3.9. Starting an Activity

- The simplest and most common way to start an activity is with a single initial node; most of the diagrams you've seen so far in this chapter use this notation.
  - 初始节点是启动一个活动的最常用最简单的方法

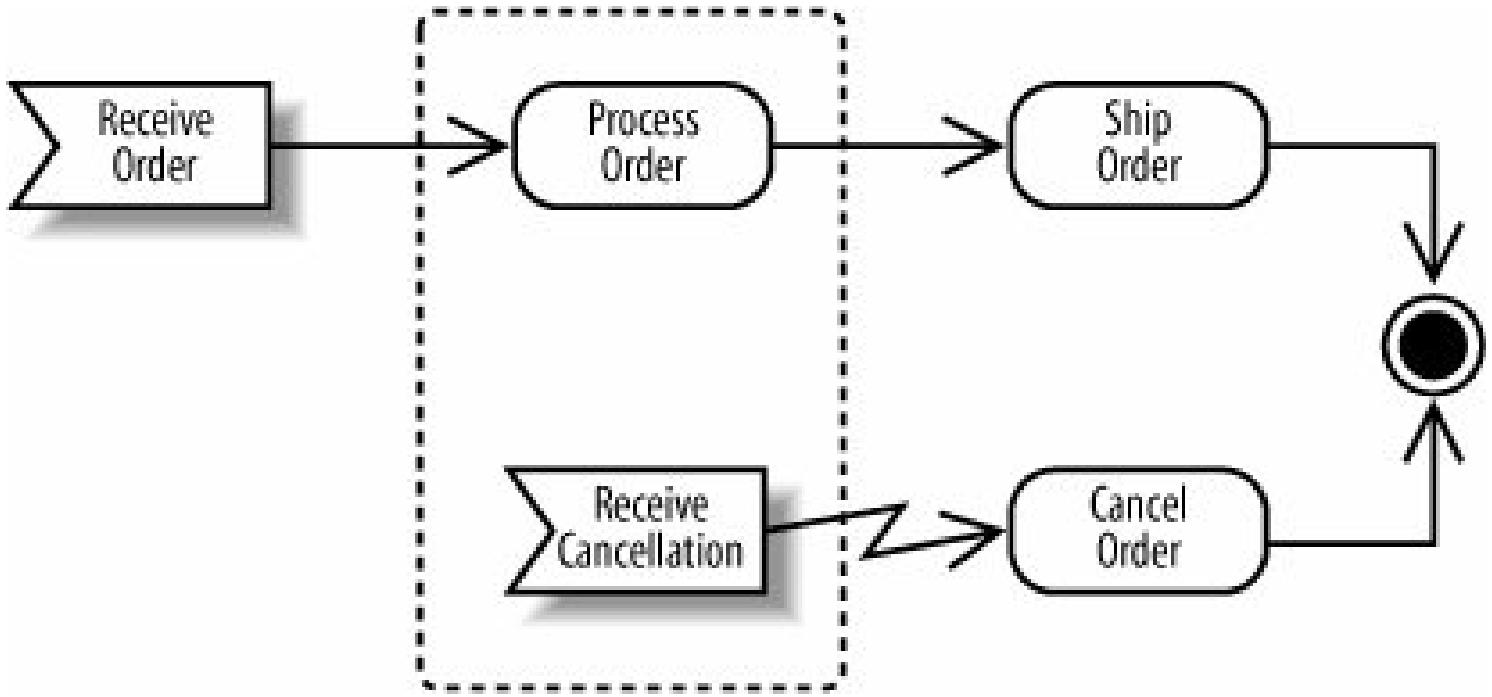
other ways to represent the start of an activity



3.10. Ending Activities and Flows

- 3.10.1. Interrupting an Activity
  - 中断一个活动
- 3.10.2. Ending a Flow
  - 停止一个流

Figure 3-22. Interruption region showing a process that can be interrupted



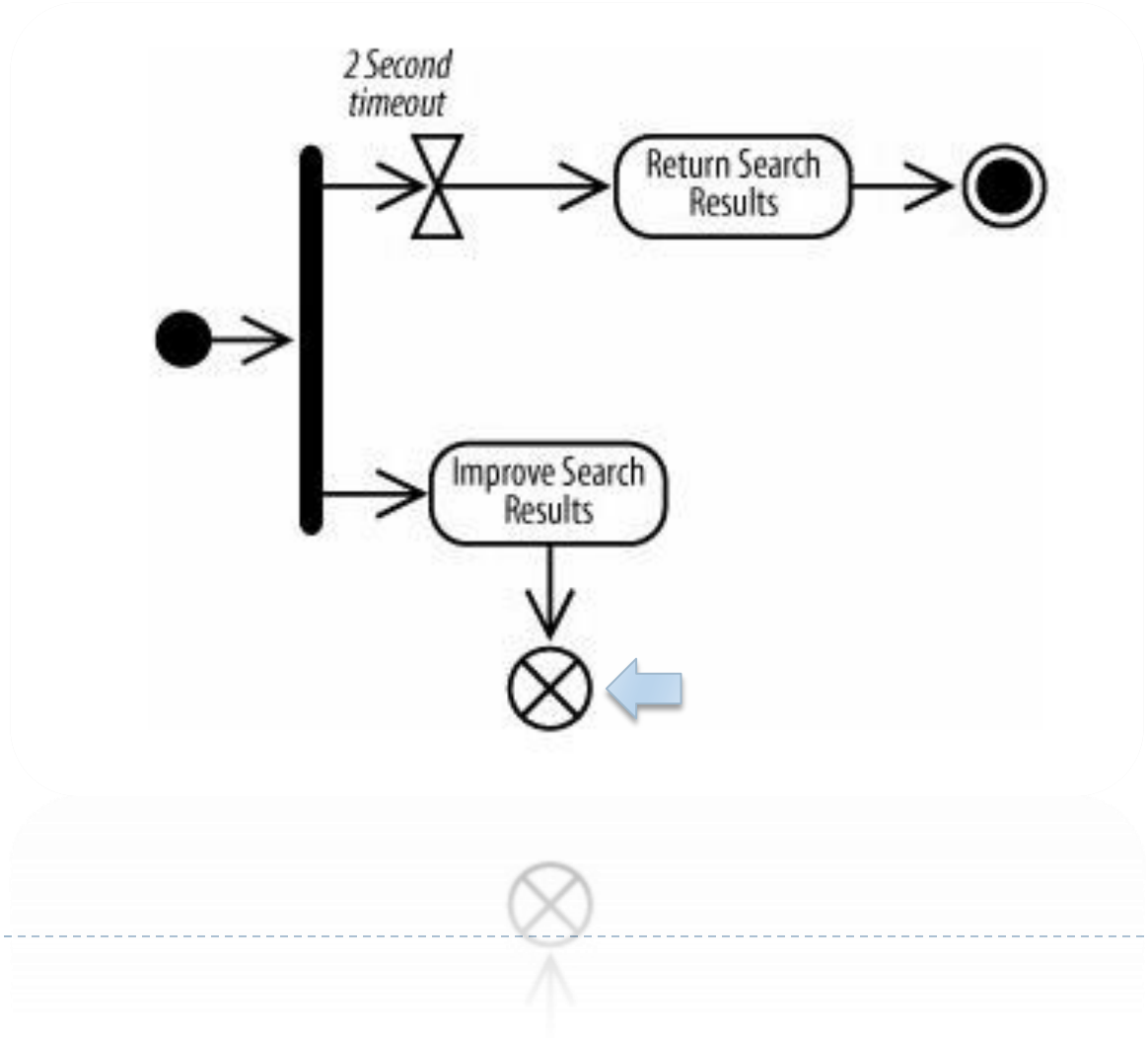
3.10.1. Interrupting an Activity

- Notice there's only one path leading into the activity final node; every action in this diagram gets a chance to finish.
  - 只有一条路径可以到达终止节点
- Sometimes you need to model that a process can be terminated by an event. 一个过程可由一个事件结束
  - This could happen if you have a long running process that can be interrupted by the user. 长过程
  - Or, in the CMS order handling activity, you may need to account for an order being canceled.
- You can show interruptions with *interruption regions* .

3.10.2. Ending a Flow

- A new feature of UML 2.0 is the ability to show that a flow dies without ending the whole activity.
  - 终止某一个流
- A *flow final* node terminates its own path not the whole activity.

Figure 3-23. A flow final node terminates only its own path not the whole activity



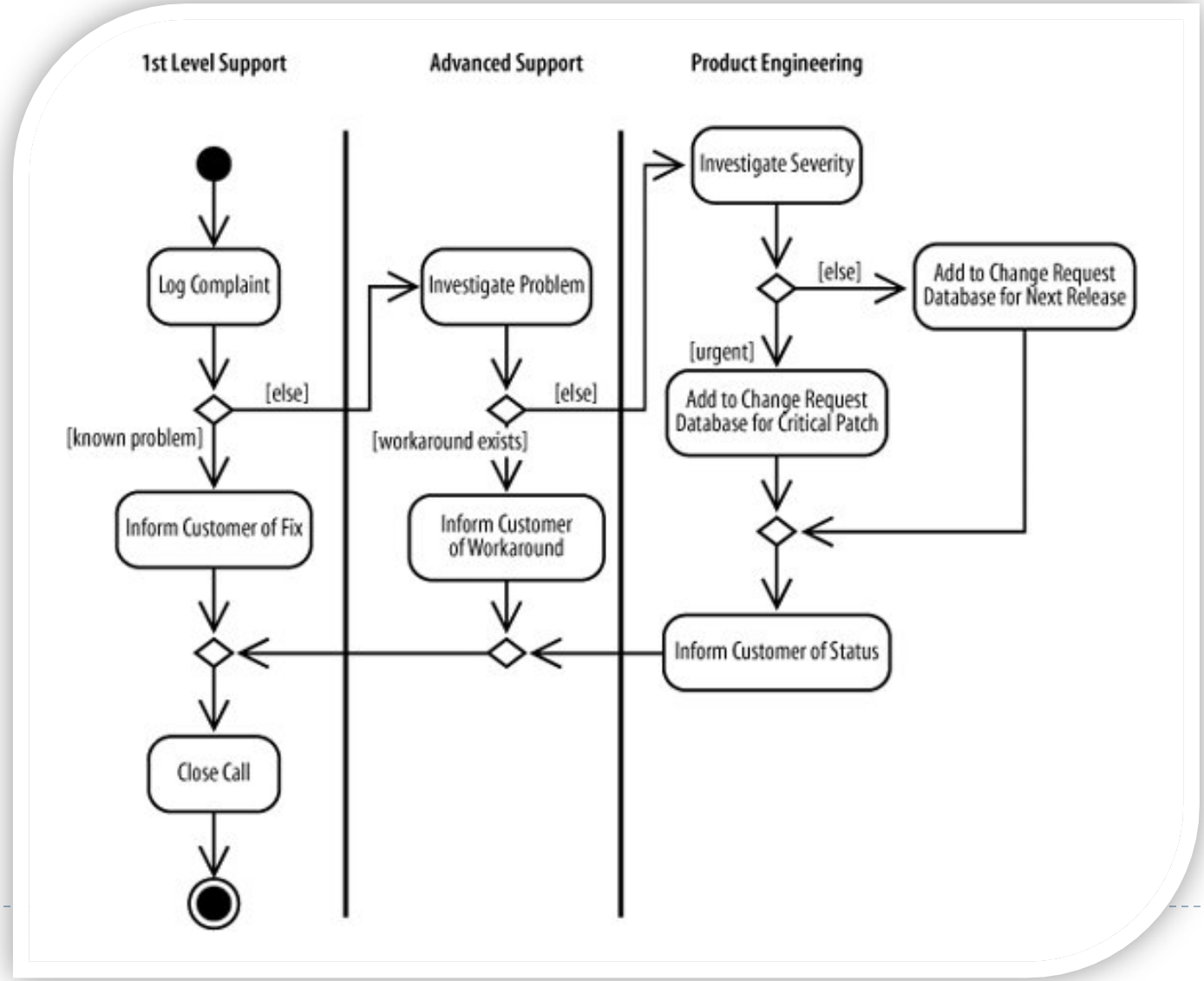
3.11. Partitions (or Swimlanes)

- Activities may involve different participants, such as different groups or roles in an organization or system.
  - 不同分组、不同角色
- The following scenarios require multiple participants to complete the activity：多种参与者
  - *An order processing activity*
    - Requires the shipping department to ship the products and the accounts department to bill the customer.
  - *A technical support process*
    - Requires different levels of support, including 1st level Support, Advanced Support, and Product Engineering.

Swimlanes (泳道)

- You use *partitions* to show which participant is responsible for which actions. 分区
- Partitions divide the diagram into columns or rows (depending on the orientation of your activity diagram) and contain actions that are carried out by a responsible group. 纵向划分 或 横向划分
- The columns or rows are sometimes referred to as *swimlanes*.

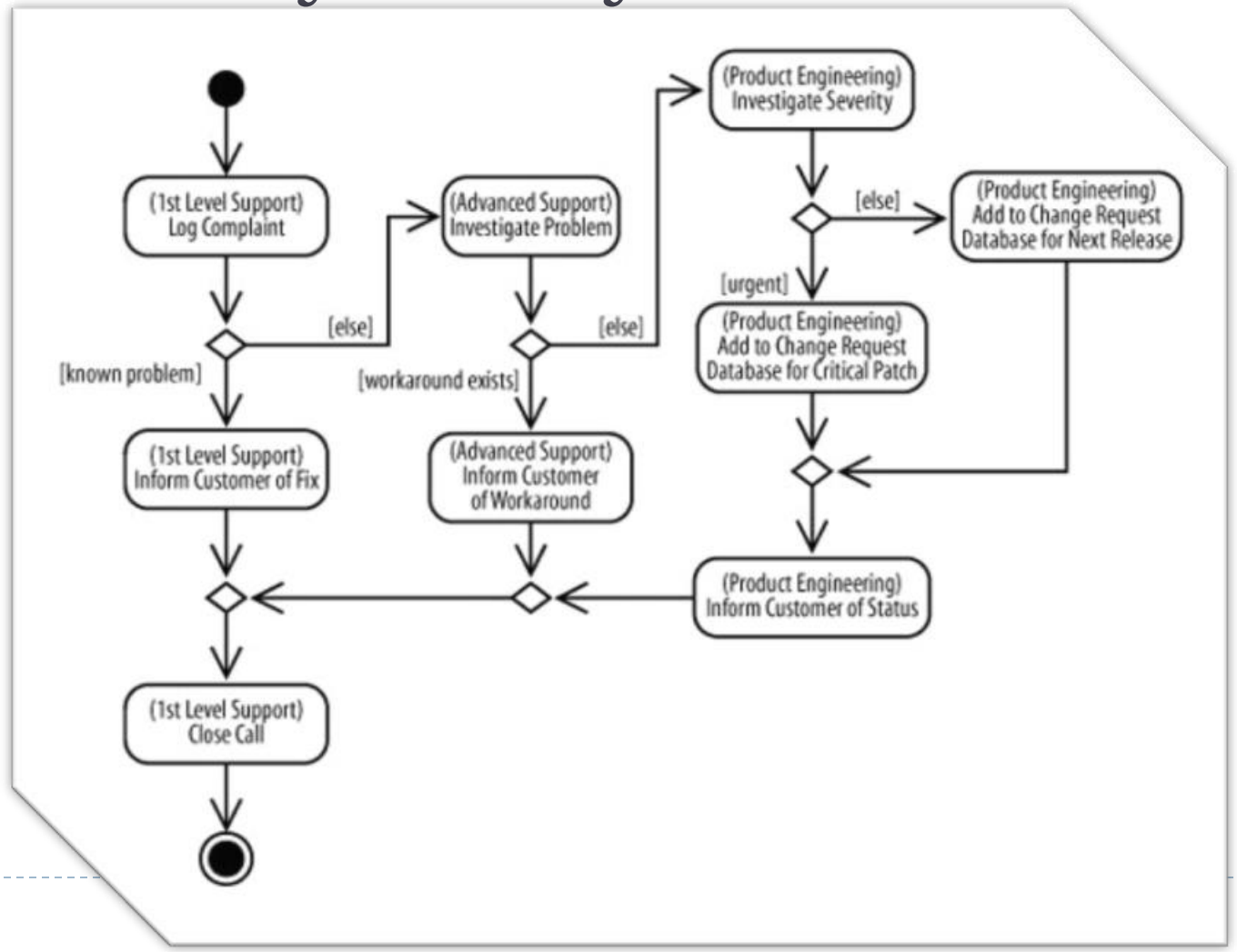
Figure 3-24. Partitions help organize this activity diagram by clarifying responsible parties



Annotations 标注

- You can also show responsibility by using *annotations*.
- Notice that there are no swimlanes; instead, the name of the responsible party is put in parentheses in the node.
- This notation typically makes your diagram more compact, but it shows the participants less clearly than swimlanes.

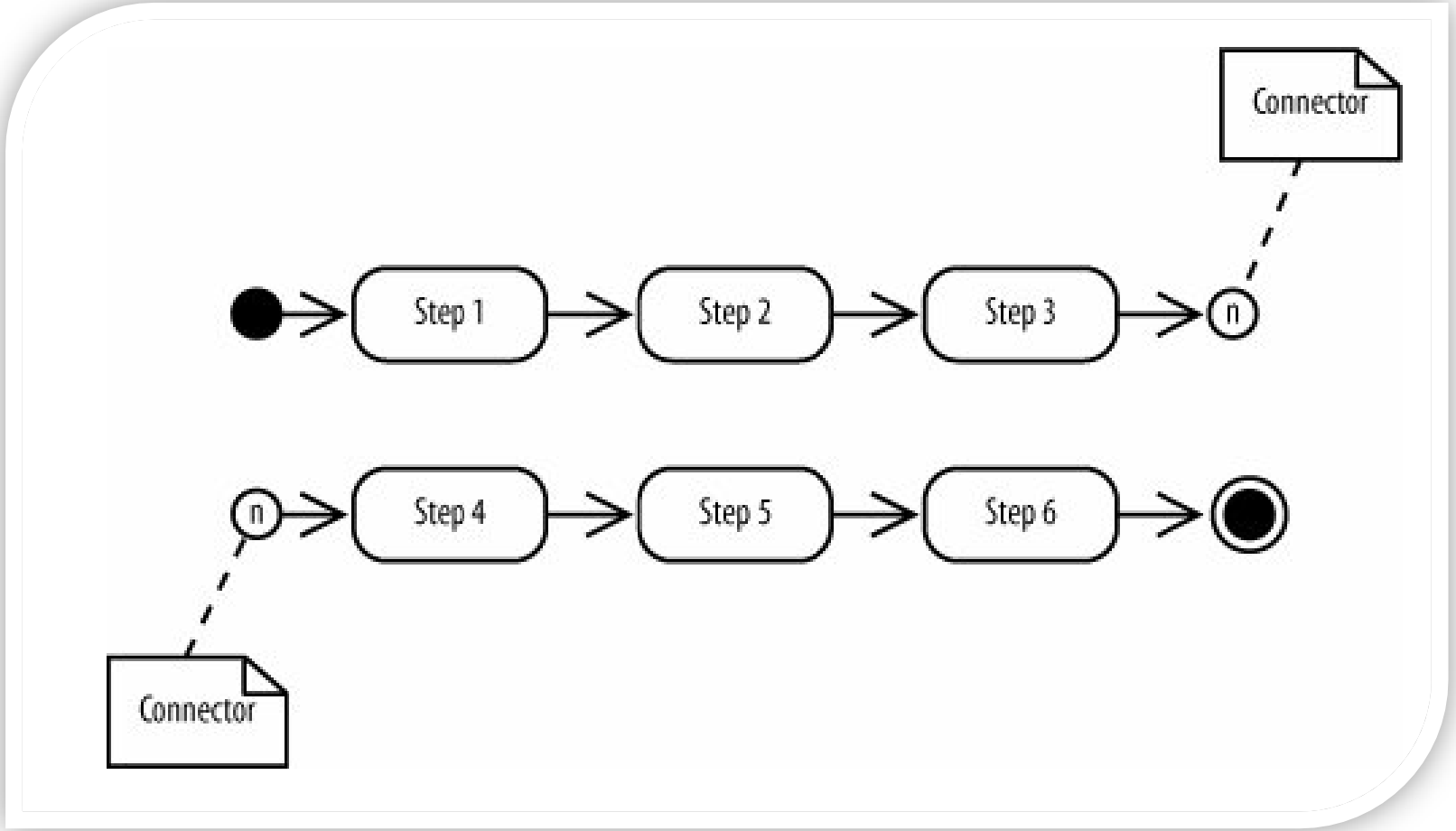
Figure 3-25. Annotations can be used instead of swimlanes as a way of showing responsibility directly in the action



3.12. Managing Complex Activity Diagrams

- 3.12.1. Connectors 连接器
  - If your activity diagram has a lot of actions, you can end up with long, crossing lines, which make the diagram hard to read. This is where connectors can help you out.
  - *Connectors* help untangle your diagrams, connecting edges with symbols instead of explicit lines.
    - A connector is drawn as a circle with its name written inside. 圆圈
    - Connectors are typically given single character names.
  - Connectors come in pairs: 成对出现
    - one has an incoming edge and the other has an outgoing edge.
    - The second connector picks up where the first connector left off.

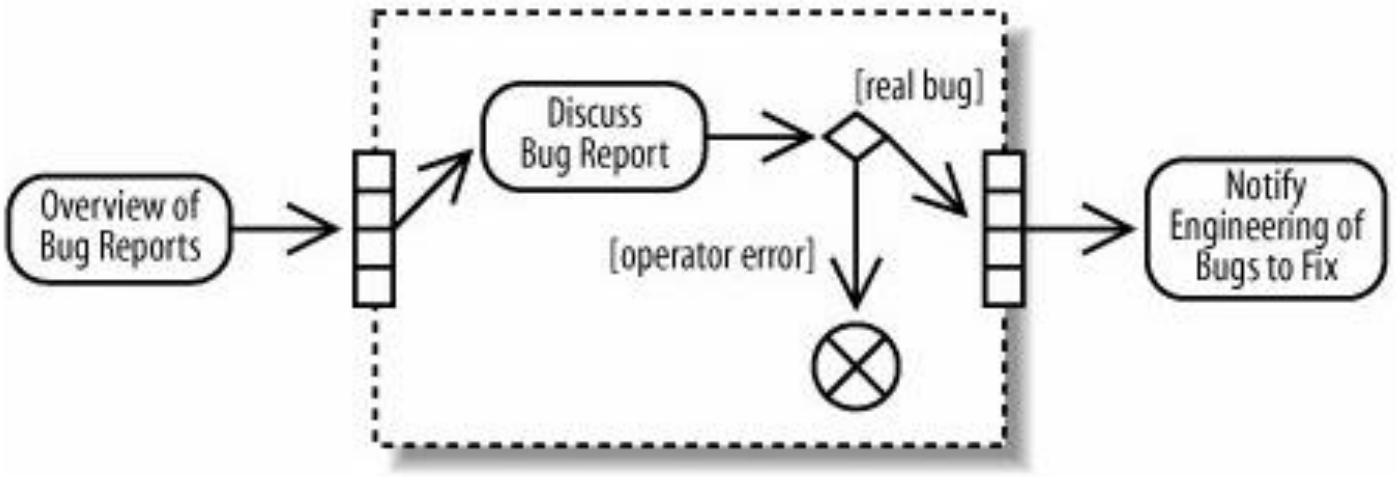
Figure 3-26. Connectors can improve the readability of a large activity diagram



3.12. Managing Complex Activity Diagrams

- 3.12.2. Expansion Regions 扩展区域
  - *Expansion regions* show that actions in a region are performed for each item in an input collection.
    - For example, an expansion region could be used to model a software function that takes a list of files as input and searches each file for a search term.
  - Draw an expansion region as a large rounded rectangle with dashed lines and four aligned boxes on either side.
    - The four boxes represent input and output collections (but they don't imply that the collection size is four).

Figure 3-27. The actions in an expansion region are performed for each item in a collection



Summary

- 3. Modeling System Workflows: Activity Diagrams
  - 3.1. Activity Diagram Essentials
  - 3.2. Activities and Actions
  - 3.3. Decisions and Merges
  - 3.4. Doing Multiple Tasks at the Same Time (forks and joins)
  - 3.5. Time Events
  - 3.6. Calling Other Activities
  - 3.7. Objects
  - 3.8. Sending and Receiving Signals
  - 3.9. Starting an Activity
  - 3.10. Ending Activities and Flows
  - 3.11. Partitions (or Swimlanes)
  - 3.12. Managing Complex Activity Diagrams

What's Next?

- Chapter 4. Modeling a System's Logical Structure: Introducing Classes and Class Diagrams
  - 4.1. What Is a Class?
  - 4.2. Getting Started with Classes in UML
  - 4.3. Visibility
  - 4.4. Class State: Attributes
  - 4.5. Class Behavior: Operations
  - 4.6. Static Parts of Your Classes
- Chapter 5. Modeling a System's Logical Structure: Advanced Class Diagrams
  - 5.1. Class Relationships
  - 5.2. Constraints
  - 5.3. Abstract Classes
  - 5.4. Interfaces
  - 5.5. Templates

See you …