

**Dr. Hazel Campbell**  
Slides originally by Dr. Ken Wong  
Department of Computing Science  
University of Alberta



# **REQUIREMENTS**

Images reproduced in these slides have been included under section 29 of the Copyright Act, as fair dealing for research, private study, criticism, or review. Further distribution or uses may infringe copyright.

# Importance of Requirements

- Link:
  - <http://www.projectsmart.co.uk/docs/chaos-report.pdf>

| Reason for Project Failure | % of Responses |
|----------------------------|----------------|
| Incomplete requirements    | 13.1           |
| Lack of user involvement   | 12.4           |
| Lack of resources          | 10.6           |
| Unrealistic expectations   | 9.9            |
| Lack of management support | 9.3            |
| Changing requirements      | 8.7            |
| Lack of planning           | 8.1            |
| System no longer needed    | 7.5            |

# Requirements

- Types:
  - user requirements
    - what tasks the user can do with the system
  - functional requirements (features)
    - what behaviors the system does or supports
  - non-functional requirements (qualities)
    - how well the system should do what it does
    - e.g., response time, resource usage, availability

# Requirements

- **Types:**
  - **external interfaces**
    - e.g., interfaces to other hardware and software, data sources and sinks, formats, protocols
  - **physical setting**
    - e.g., location, workspace, lighting, noise, temperature
  - **developer constraint**
    - e.g., implementation technology, documentation

# Requirements

- Types:
  - business requirements
    - why the system is needed
  - business constraint
    - what the system or process must comply with
    - e.g., corporate policy, industry standard, government regulation

# Requirements

- Requirements should be:
  - correct
    - requirements properly represent user needs
  - complete
    - all possible scenarios are described
  - consistent
    - requirements do not contradict each other
  - clear
    - no ambiguities
  - realistic
    - can be achieved by “mere mortals”

# Requirements

- Also desired:
  - traceable
    - can trace functionality and tests to the requirement being satisfied
  - verifiable
    - repeatable test(s) can be designed to show that the system fulfills the requirement



# Verifiable Requirements

- Verifiable?
  - “The system shall have a good user interface.”

# Verifiable Requirements

- Verifiable?
  - “The system shall respond to the user in under one second for most tasks.”



# Verifiable Requirements

- Verifiable?
  - “When the output state changes, it is logged in the event log.”



# Verifiable Requirements

- Verifiable?
  - “The system shall be free of defects.”

# Requirements Activities

- Done iteratively:
  - requirements elicitation
    - discover user needs
  - requirements analysis
    - decide scope and priorities
    - study feasibility
  - requirements specification
    - detail the requirements in terms the *users can understand*

# Users

- Who is the “user”?
  - primary
    - end user
    - with frequent hands-on use
  - secondary
    - manager of end users
    - with occasional use, or via an assistant
  - tertiary
    - owner of the system
    - uses output, influences or makes funding decisions

# Users

- Some characteristics to consider:
  - background
    - literacy and language
    - motivation to learn
  - domain knowledge
  - task familiarity
  - computer skills
  - attitude to computers and technology

# Users

- Some characteristics to consider:
  - perceptual, motor, and tactile abilities
    - seeing and hearing difficulties
    - fine motor skills with input devices
  - physical
    - height and strength (for kiosk design)
    - hand/finger size (for mobile device design)
    - health, age, and gender
  - social
    - relationships with peers
    - culture

# Users

- Kinds of use:
  - infrequent use / novice user
    - need wizards
    - need clear prompts, error handling
  - frequent use / expert user
    - need keyboard shortcuts
    - need customization, programmability

# Users

- Some issues to consider:
  - users cannot always express what they want
    - but they often know what they do not like
  - users may not know what is possible
    - what is technically and economically feasible?
  - users stick to what they ...
    - know already works, or have always done
  - users may fear job losses
    - leads to non-constructive participation

# Users

- “Innovator’s dilemma”:
  - [Clayton Christensen]
    - as the user base for an application grows, there is a tendency for developers to focus on this increasingly expert (and vocal) group of users
    - the system becomes more sophisticated
    - development becomes “optimized” for them



# Users

- “Innovator’s dilemma”:
  - potential new users need “less”
  - experts don’t want to lose advanced features
  - competitor attracts the new users with a simpler “good enough” app
  - original app loses market share due to disruption from the low end

# Understanding

- Tips:
  - manage expectations
    - be clear and honest about claims
    - avoid surprises, disappointments, hype
  - involve the user
    - build tangible prototypes to gain feedback
    - more likely to forgive problems if they are involved
  - establish a glossary
    - terminology used in the application domain  
(not programming domain)



# **USER STORIES**

# Specifying Needs

- Old (waterfall) method:

**Write down all the requirements.**



**Users only get what *was written*.**

- Agile method: less writing, more talking

# Specifying Needs

- User story:
  - written description of what a user wants to achieve with the system

As a guest, I want to reserve a hotel room.

As a guest, I want to see a list of room amenities.

As a conference planner, I want to see meeting room capacities.

*on index cards*

*Typical forms:*

*As a «user role»,  
I want «goal».*

*As a «user role»,  
I want «goal»,  
so that «reason».*

# Defining User Stories

- Tips:
  - describe *what not how*
    - avoid technical details or choices of technologies, unless it is a development constraint
  - avoid epics for near-term needs
    - better to split up huge stories into more, smaller stories (but not too small)

# Defining User Stories

- **Tips:**
  - prioritize user stories
    - discuss with the user what they find of most value, and stage development on that first
  - can attach an effort estimate to complete
    - normally sized to take days, not many weeks
  - use stories to plan development tasks
    - create work items in the iteration plan

# “SMART Work Items”

|          |            |
|----------|------------|
| <b>S</b> | Specific   |
| <b>M</b> | Measurable |
| <b>A</b> | Achievable |
| <b>R</b> | Relevant   |
| <b>T</b> | Time Boxed |

- Link:
  - <http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>

# “INVEST in Good Stories”

|          |             |
|----------|-------------|
| <b>I</b> | Independent |
| <b>N</b> | Negotiable  |
| <b>V</b> | Valuable    |
| <b>E</b> | Estimable   |
| <b>S</b> | Small       |
| <b>T</b> | Testable    |

- Link:
  - <http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>

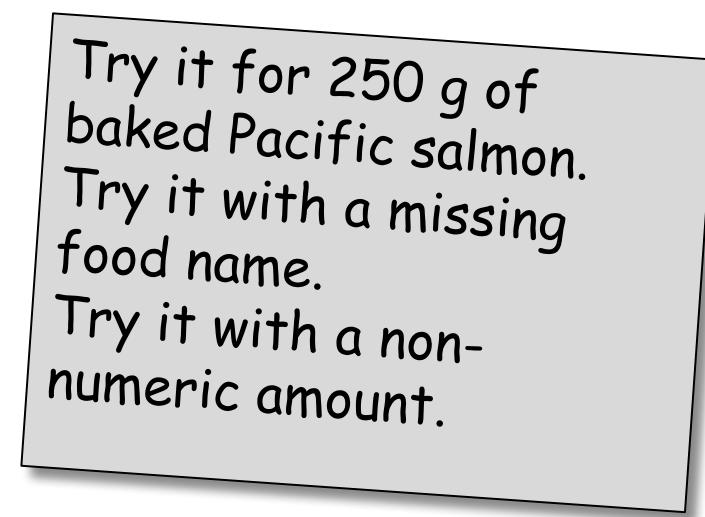
# Testable User Stories

- Front of the card:

As a meal planner, I  
want to see nutrition  
information for a given  
amount of a given food.

# Testable User Stories

- Back of the card:



*acceptance tests*

- Link:

- <http://xprogramming.com/articles/expcardconversationconfirmation/>



# USE CASES

# Identifying Tasks

- Study what tasks users do:
  - what is the goal and context?
  - what information is needed?
  - what are the steps?
  - who does the user work with?
  - why is it done this way?

# Identifying Tasks

- Scenario:
  - an informal narrative
  - personal and concrete, but not particularly general
  - use the scenario to understand existing goals, task flow, and possible irritants

# Scenario

- Example:
  - “I want to track the calories for a meal, so I consult the USDA Nutrient database. I want to look up ‘Pacific salmon’ so I enter that as the keywords. Item not found! So I enter ‘salmon’ and try again. That works, but I get 46 items, including salmonberries and even cloudbERRIES. Why? I choose ‘fish, salmon, sockeye, cooked, dry heat’, then figure  $2.5 \times 100$  g units for my item, and scan the table to see 422 kcal in the energy row.”

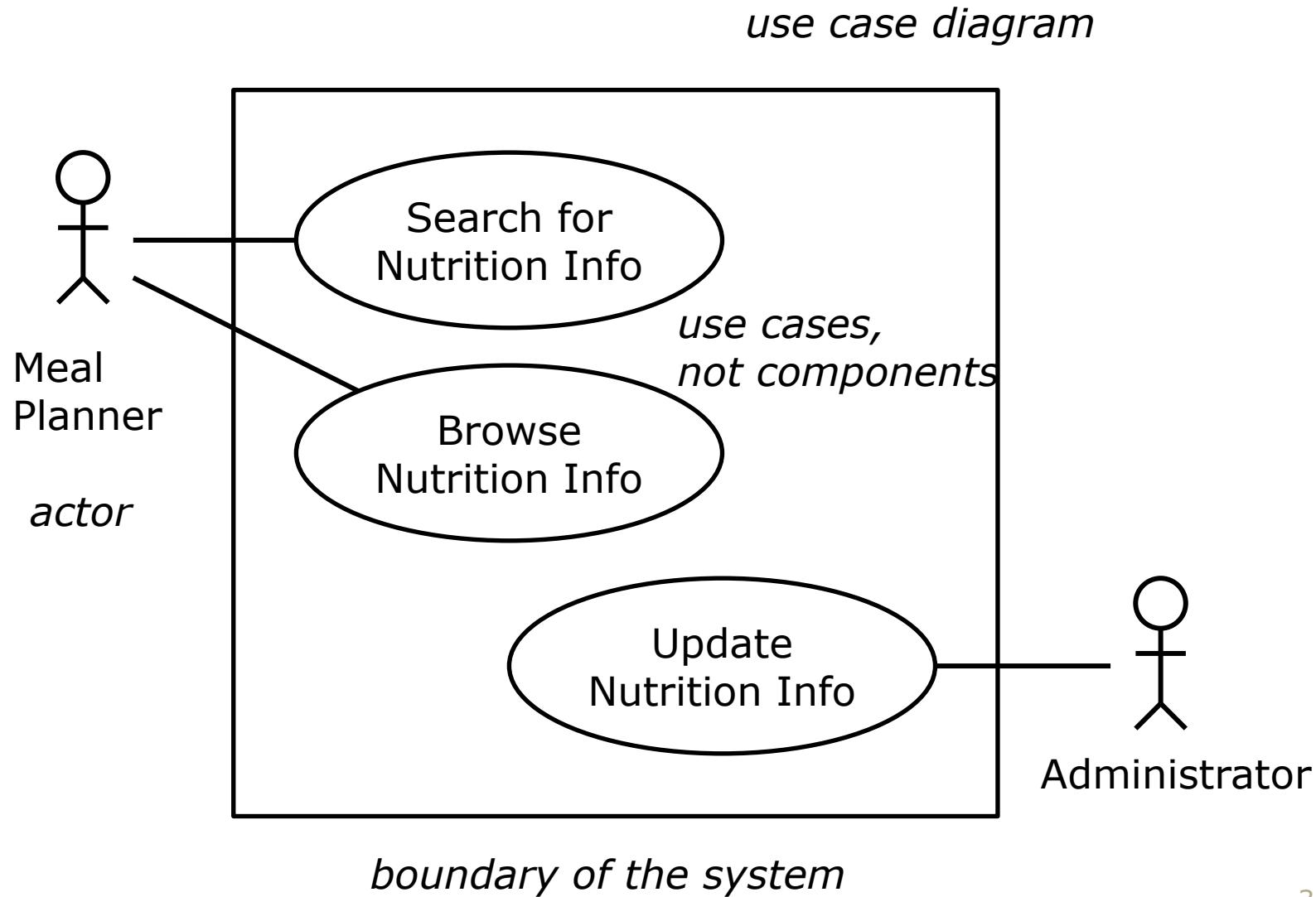
# Specifying Tasks

- Use Cases:
  - capture the goal, conditions, and steps of a coherent interaction between the actor(s) and the software system
  - more general than a specific scenario
  - written from a “user” point-of-view

# Defining Use Cases

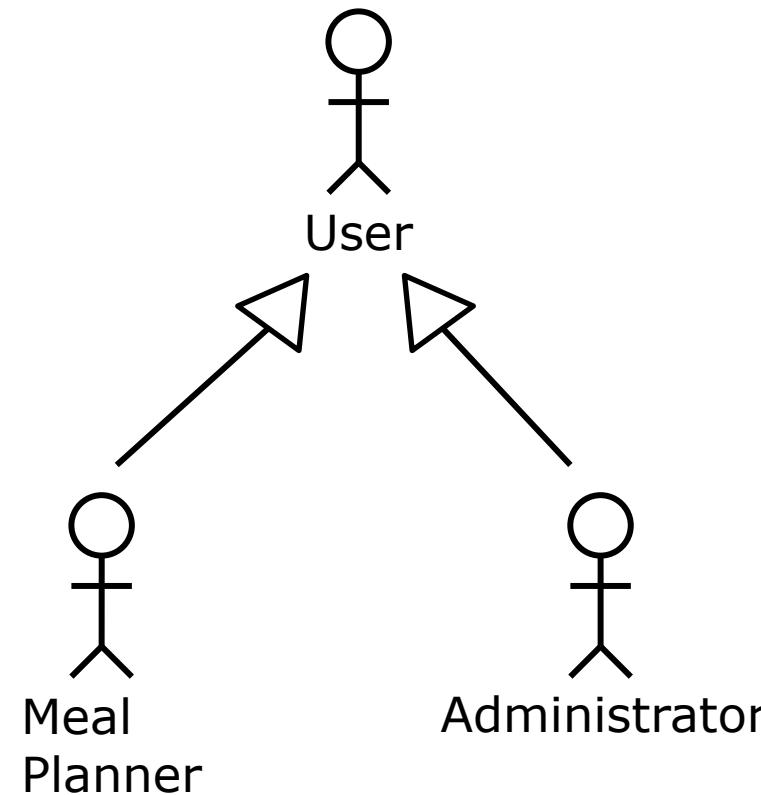
- **Stages:**
  - **identify the actors**
    - consider different user roles and external systems
  - **define use cases**
    - include all cases of use
  - **refine use cases**
    - consider exceptional conditions and qualities
  - **relate use cases**
    - consider inclusion and extension dependencies

# Identify the Actors



# Identify the Actors

- Actor generalization:



# Define/Refine Use Cases

- Example:

|                      |  |
|----------------------|--|
| Use Case Name        | <b>SearchForNutritionInfo</b>                |
| Participating Actors | Meal Planner (primary)                       |
| Goal                 | Meal Planner finds nutrition information     |
| Trigger              | Meal Planner chooses the Search option       |
| Precondition         | Meal Planner knows food name and amount.     |
| Postcondition        | On success, nutrition information displayed. |
|                      | ...  |

# Define/Refine Use Cases

- Example: *user point of view*

|              |   |
|--------------|---|
| Basic Flow I | System prompts Meal Planner to enter keywords.          |
| 2            | Meal Planner submits keywords.                          |
| 3            | System lists matching foods, prompting for a selection. |
| 4            | Meal Planner browses and selects a food.                |
| 5            | System prompts for food weight in units of 100 g.       |
| 6            | Meal Planner enters food units.                         |
| 7            | System presents nutrition data for the amount of food.  |
|              | ...   |

*avoid implementation specifics*

# Define/Refine Use Cases

- Example:

|              |   |
|--------------|---|
| Exceptions 3 | If there are no matching foods                        |
| 3.1          | System displays an error                              |
| 3.2          | System returns to step 1                              |
| 7            | If given food units is non-numeric, use 0 and proceed |
|              |   |

# Define/Refine Use Cases

- Example:

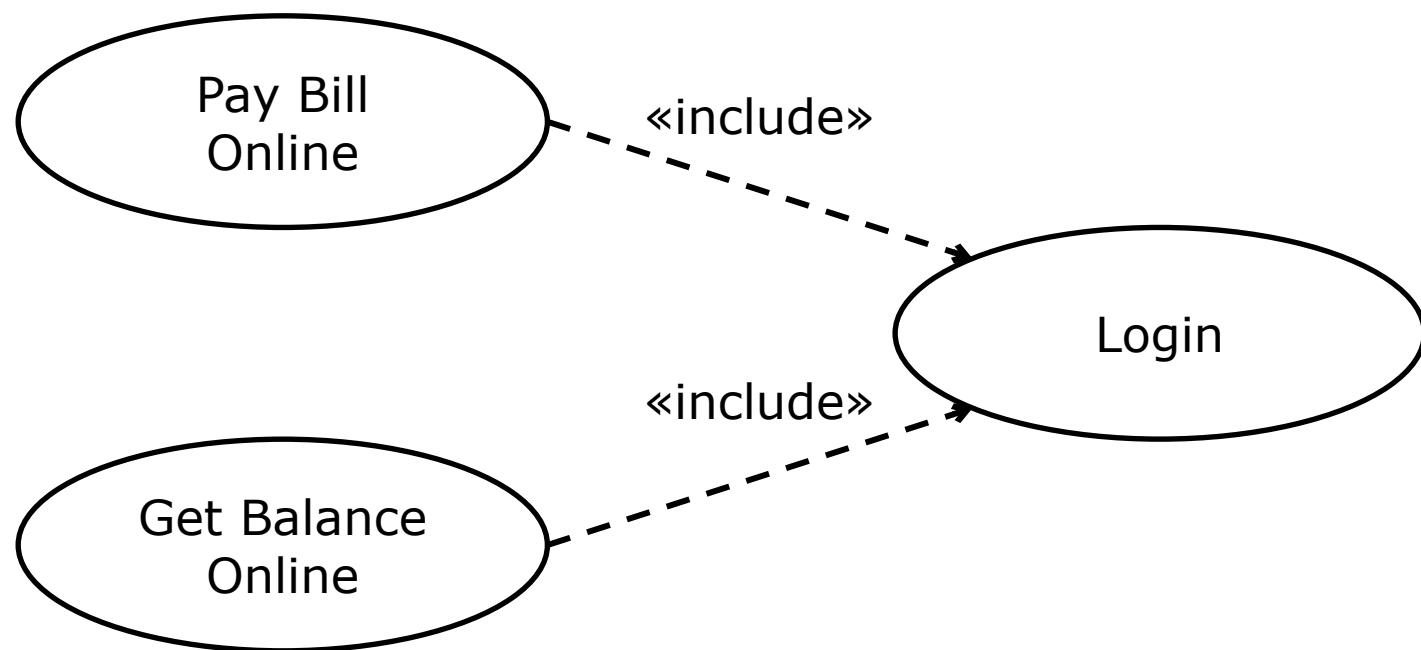
|                   |  |
|-------------------|--|
| Qualities         | System responds in under 2 s for list of matching foods and for nutrition data on a specific food. |
| Constraints       | Use USDA nutrition data.   |
| Includes          |  |
| Extends           |  |
| Related Artifacts |  |
| Notes             |  |
| Open Issues       |  |

# Essential Use Case

| <b>SearchForNutritionInfo</b>        |                                |
|--------------------------------------|--------------------------------|
| <i>User Intention (Meal Planner)</i> | <i>System Responsibility</i>   |
| Initiate search.                     |                                |
|                                      | Request keywords.              |
| Submit keywords.                     |                                |
|                                      | List matching foods to select. |
| Select a food.                       |                                |
|                                      | Request food units.            |
| Enter food units.                    |                                |
|                                      | Present nutrition data.        |

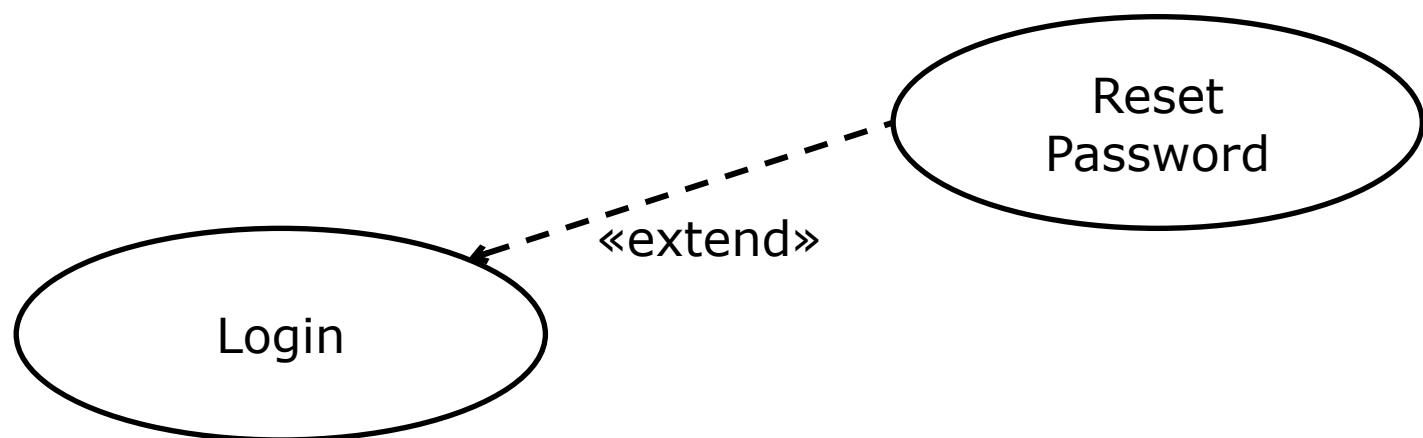
# Relate Use Cases

- Inclusion:
  - a use case may include another use case (for necessary, shared behavior)



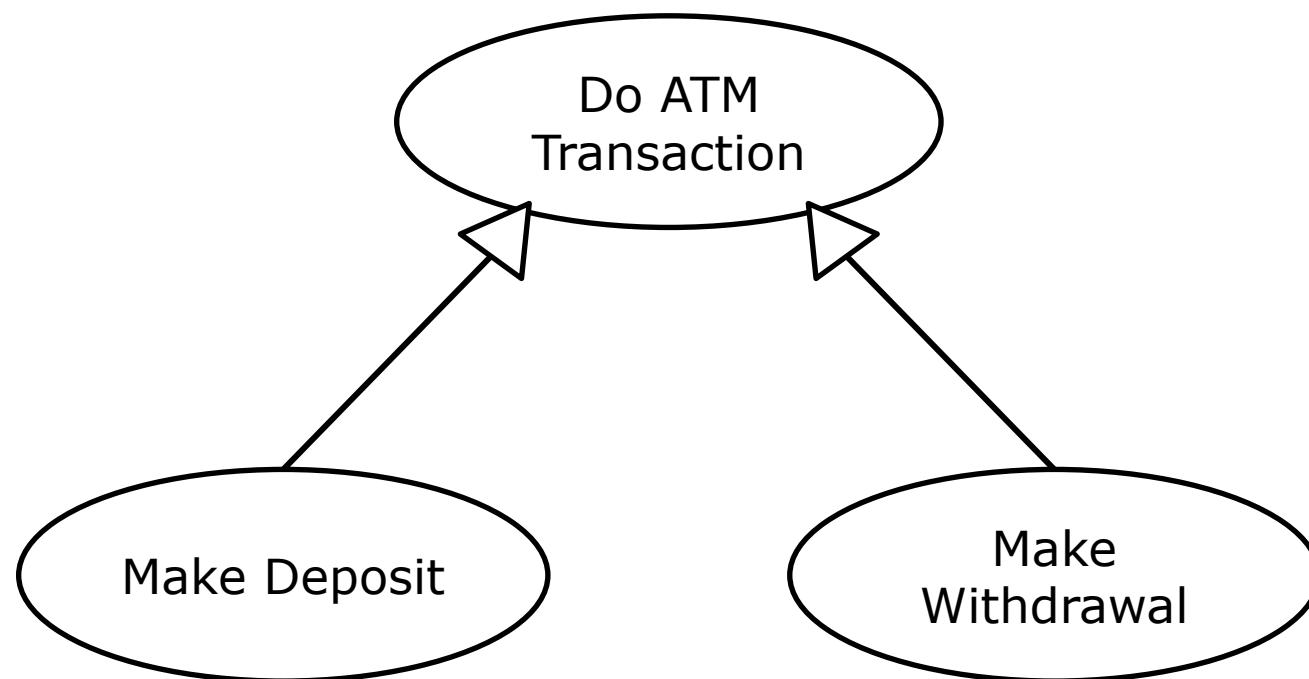
# Relate Use Cases

- Extension:
  - a use case may be extended by another use case (for optional or exceptional behavior)



# Relate Use Cases

- Use case generalization:
  - a use case may be a specialization of a more general use case

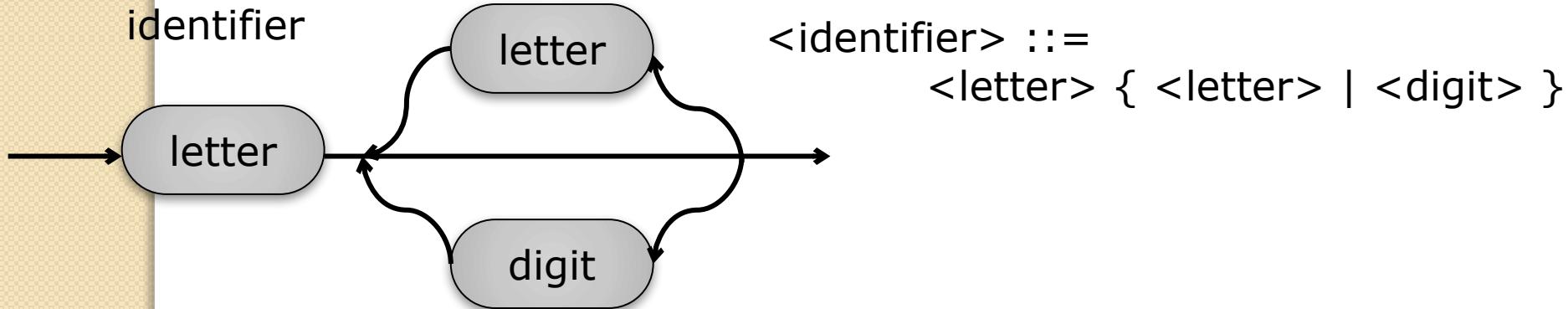




# **AUGMENTING REQUIREMENTS**

# Augmenting Requirements

- Can add other descriptions, for example:
  - use cases to user stories
  - data schemas
  - sample input and output
  - user interface mockups and storyboards
  - grammars (language syntactic/lexical structure)

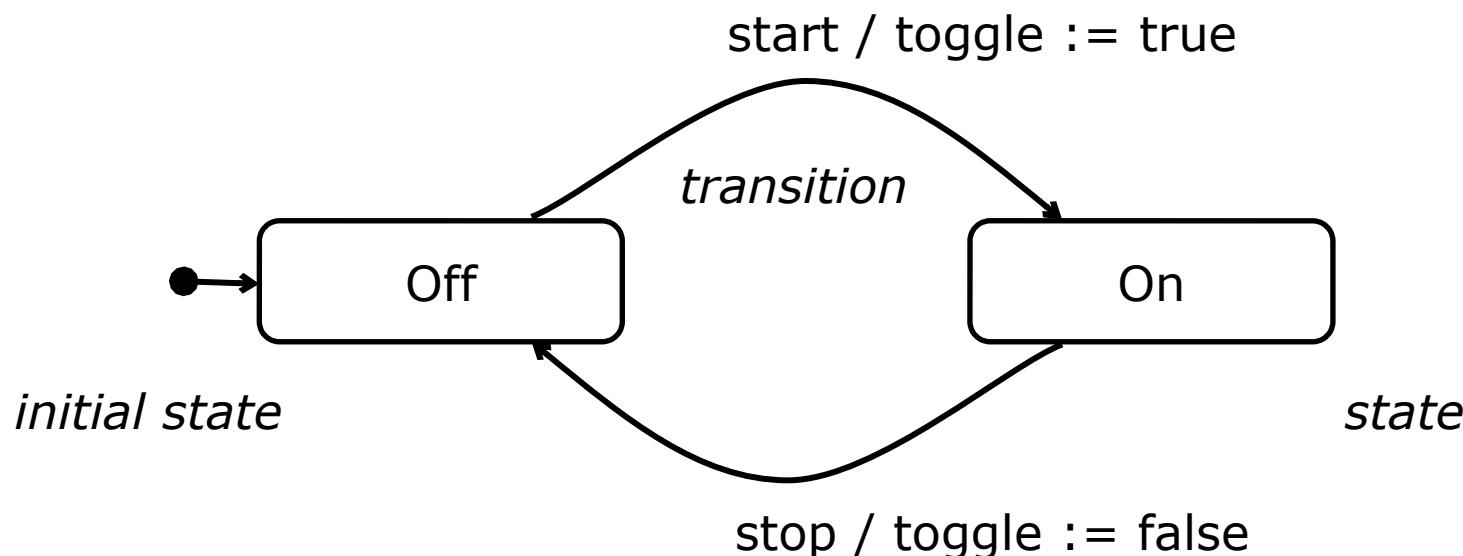


# State Models

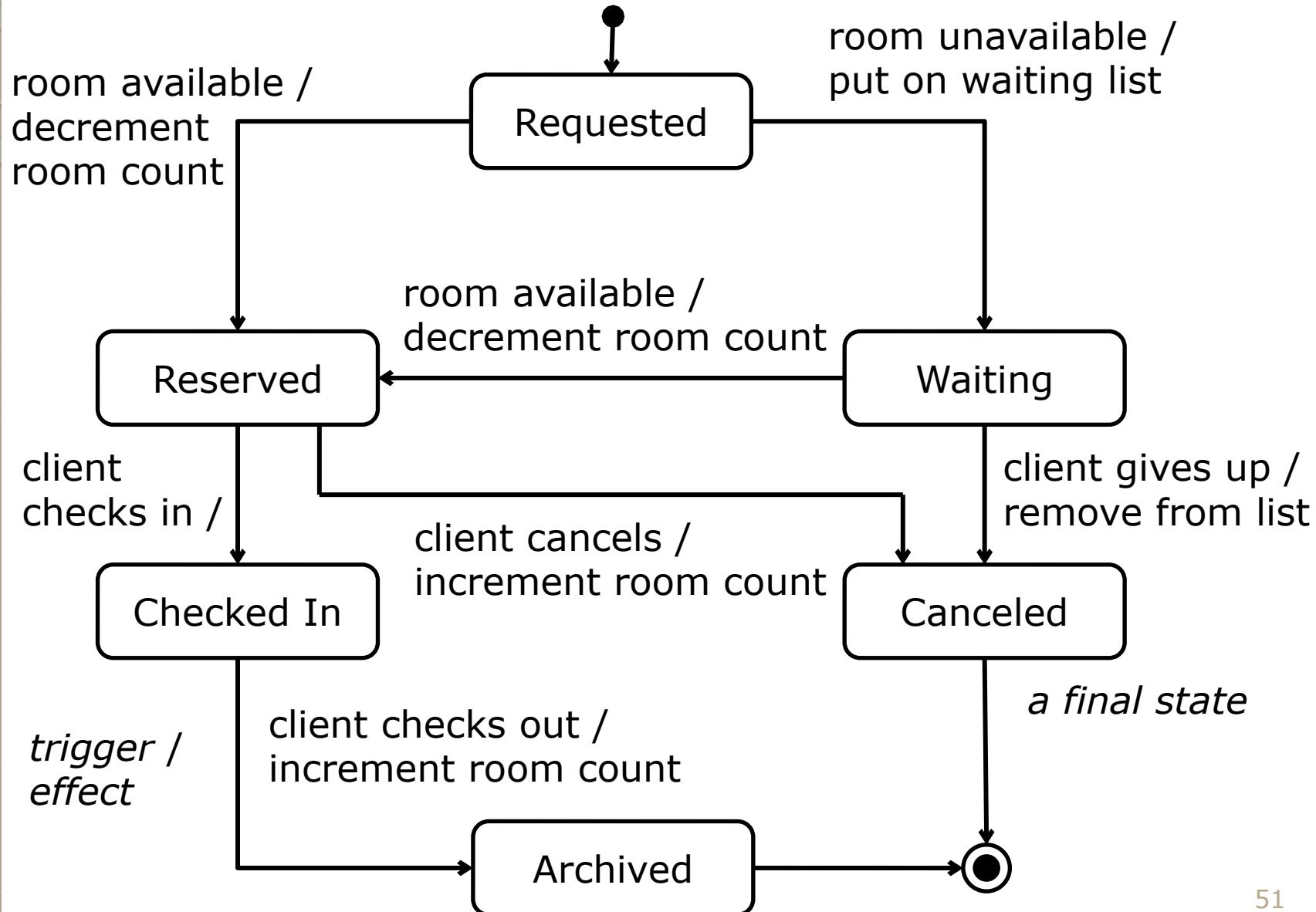
- Modeling behavior:
  - used in formally modeling the behavior of a specific object in response to external events

# UML State Diagram

- Modeling behavior:
  - states in which something can be in
    - a situation represented by attribute values
  - directed *transitions* between states
    - triggered by events, input, time, messages, etc.



# UML State Diagram



# UML State Diagram

- States:

*state name*

*state name*

*activities*

*state name*

*variables*

*activities*

# UML State Diagram

- **Transitions:** general form of transition label

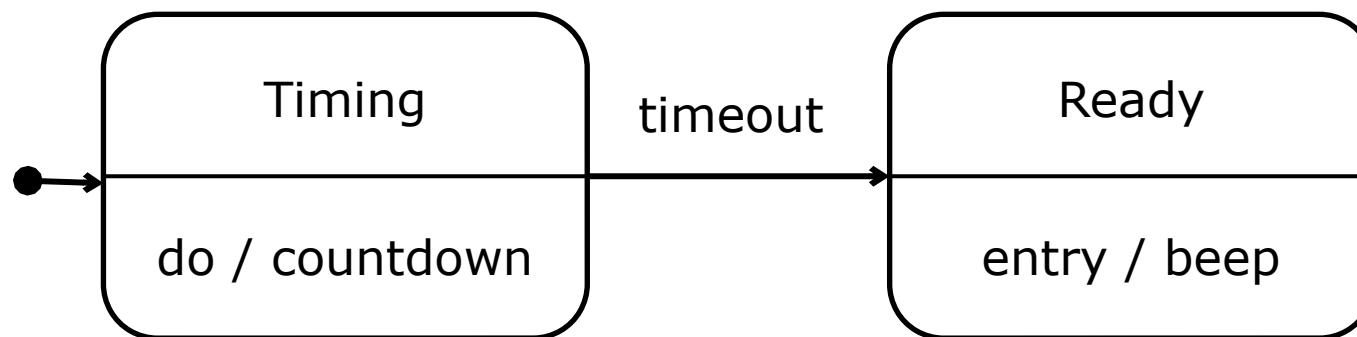
*trigger [ guard ] / effect*

- if in a current state,  
and *trigger* occurs,  
and *guard* constraint (if any) is true ...
- then perform state exit actions (if any),  
perform corresponding transition effect (if any),  
perform new state entry actions (if any);
- otherwise, stay at current state

# UML State Diagram

- Activities in states:

|                       |   |
|-----------------------|---|
| <i>entry / action</i> | perform <i>action</i> when entering state |
| <i>do / action</i>    | perform <i>action</i> while in state      |
| <i>exit / action</i>  | perform <i>action</i> when exiting state  |

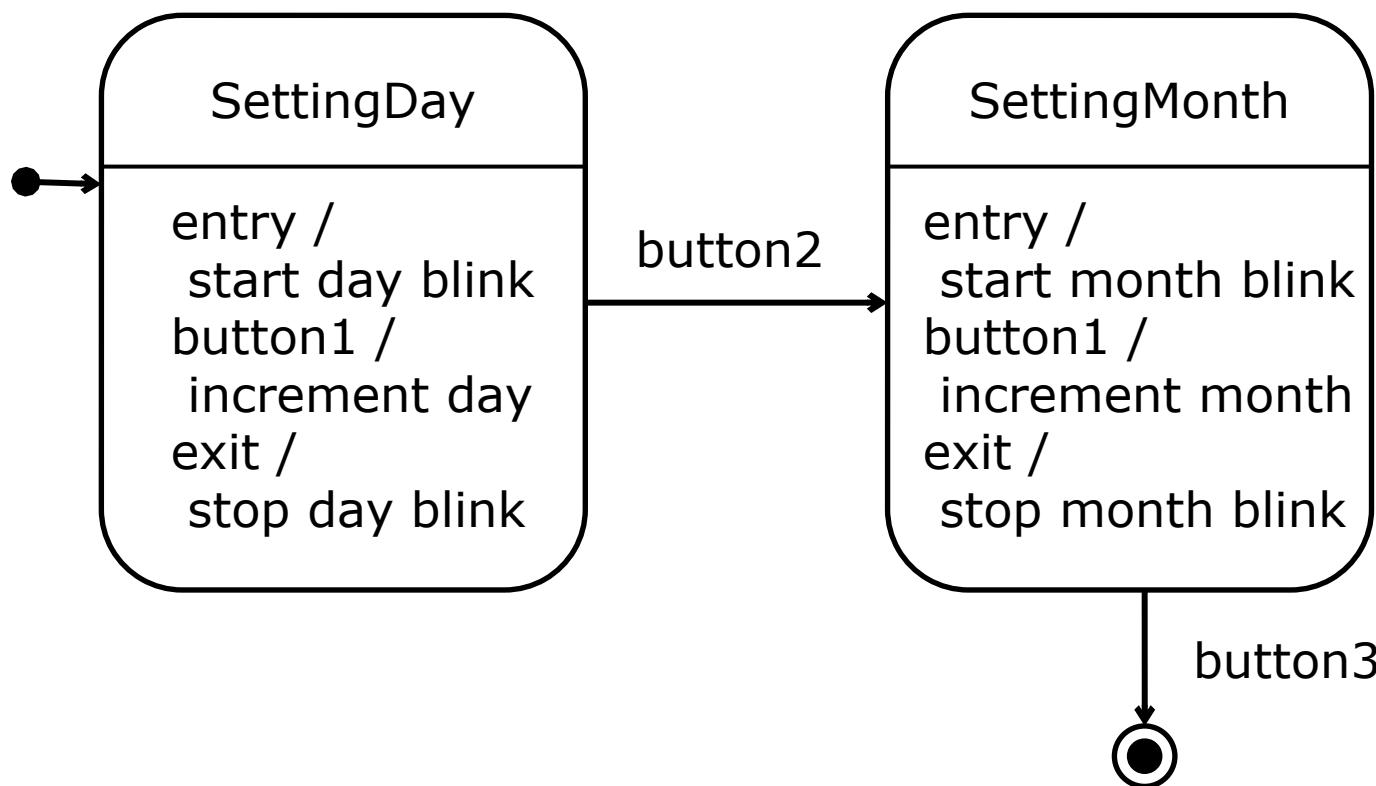


# UML State Diagram

- Activities in states:

*also called an internal transition*

|                         |  |
|-------------------------|--|
| <i>trigger / action</i> | <i>when trigger occurs, perform action</i> |
|-------------------------|--|

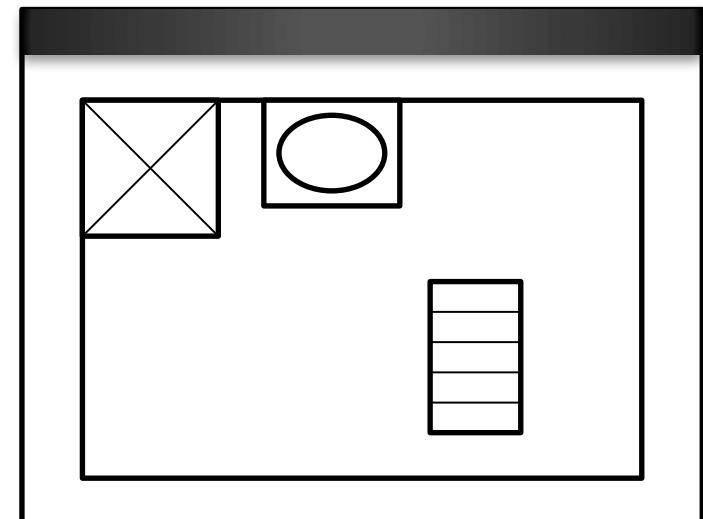


# Room Planner

- Canvas:
  - to place and move items

- Mouse events:
  - click
  - press/drag/release

- Item type menu:
  - choices of fixtures and furniture



# Room Planner

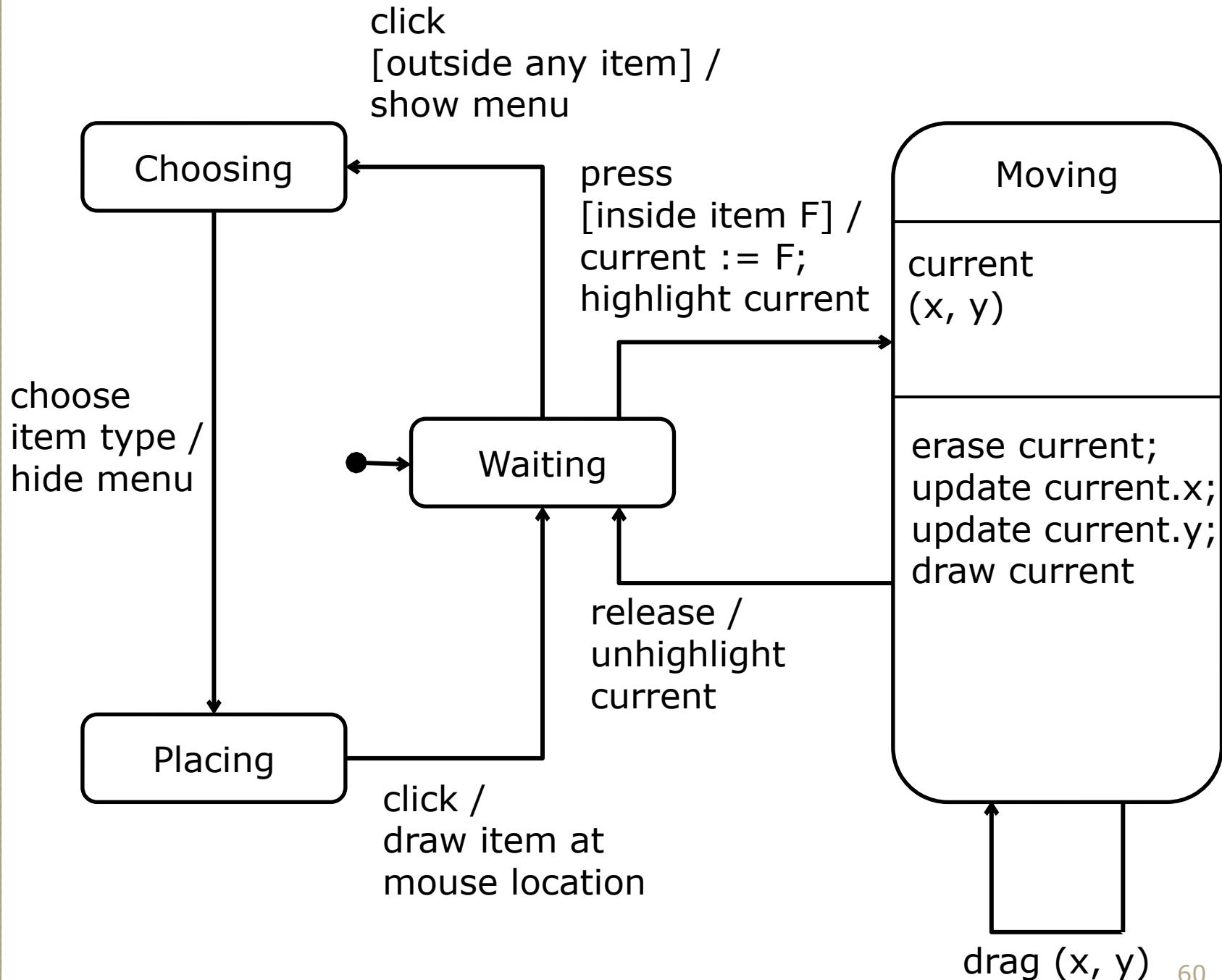
- Placing an item:
  - user clicks on canvas outside any item
    - system shows the item type menu
  - user chooses an item type from the menu
    - system hides the item type menu
  - user clicks on the canvas
    - system draws item of the chosen type at the mouse location

# Room Planner

- Moving an item:
  - user presses inside an item
    - system highlights item
  - user drags item
    - system shows moving item
  - user releases mouse
    - system puts item at new location
    - system removes item highlighting

# Room Planner

- States:
  - waiting
    - nothing happening
  - choosing
    - choosing item type from menu
  - placing
    - placing chosen shape
  - moving
    - moving item to new position



# UML State Diagram

- Tips:
  - check for completeness
    - states reachable?
    - missing transitions?
    - events not considered?
    - unforeseen situations?
  - check for dangerous situations
    - e.g., exiting without having saved edits

# UML State Diagram

- Tips:
  - check for consistency
    - similar interactions have similar effects?
    - effects are visible and give good feedback?
  - aid the user
    - is undo appropriate, in a given state?
    - is cancel or escape appropriate?
    - is invoking help appropriate?

# More Information

- Books:
  - The Essence of Object-Oriented Programming with Java and UML
    - B. Wampler
    - Addison-Wesley, 2002
  - UML Distilled
    - M. Fowler
    - Addison-Wesley, 2003

# More Information

- Books:
  - User Stories Applied
    - M. Cohn
    - Addison-Wesley, 2004
  - More About Software Requirements
    - K. Wiegers
    - Microsoft, 2006

# More Information

- Books:
  - Software Engineering: Theory and Practice
    - S.L. Pfleeger
    - Prentice-Hall, 2009

# More Information

- Links:
  - Use Cases, Ten Years Later
    - <http://alistair.cockburn.us/Use+cases%2c+ten+years+later>
  - Effective User Stories for Agile Requirements
    - <http://www.mountaingoatsoftware.com/presentations/52-effective-user-stories-for-agile-requirements>