Project 

- Situations may get tense when working on a project as a team
- However, **be respectful towards your teammates and other teams**

  It is the instructor's expectation that ALL students experience this classroom as a safe environment. (Syllabus)

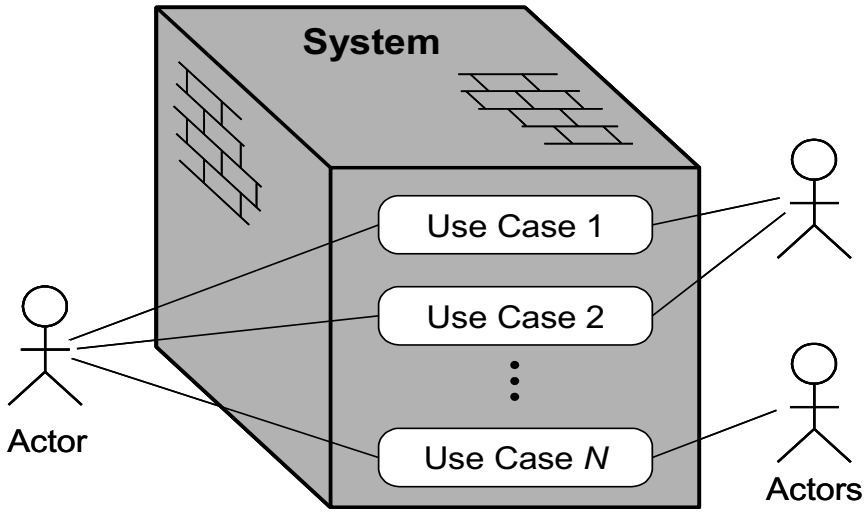
- “Everyone knew exactly what had to be done until someone wrote it down!” (Software Engineering: A Practitioner’s Approach, 8/e)
- **Use the *Development Artifacts* to keep track and sync your actions**
- Don’t create rival sub-teams within your team – nobody wins

Requirements Modeling: Domain Models

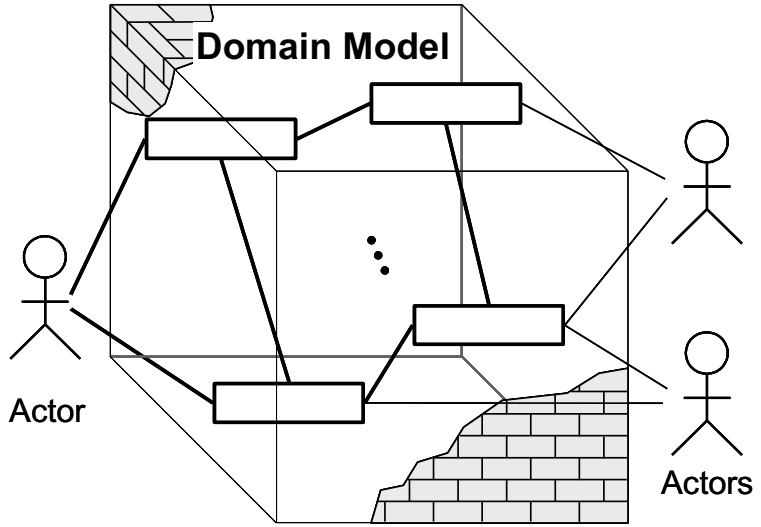
Prof. Alex Bardas

Shifting from Use-Case Modeling

- **Use Cases** focus on the system's environment and external behavior.



System is viewed as a "black box"



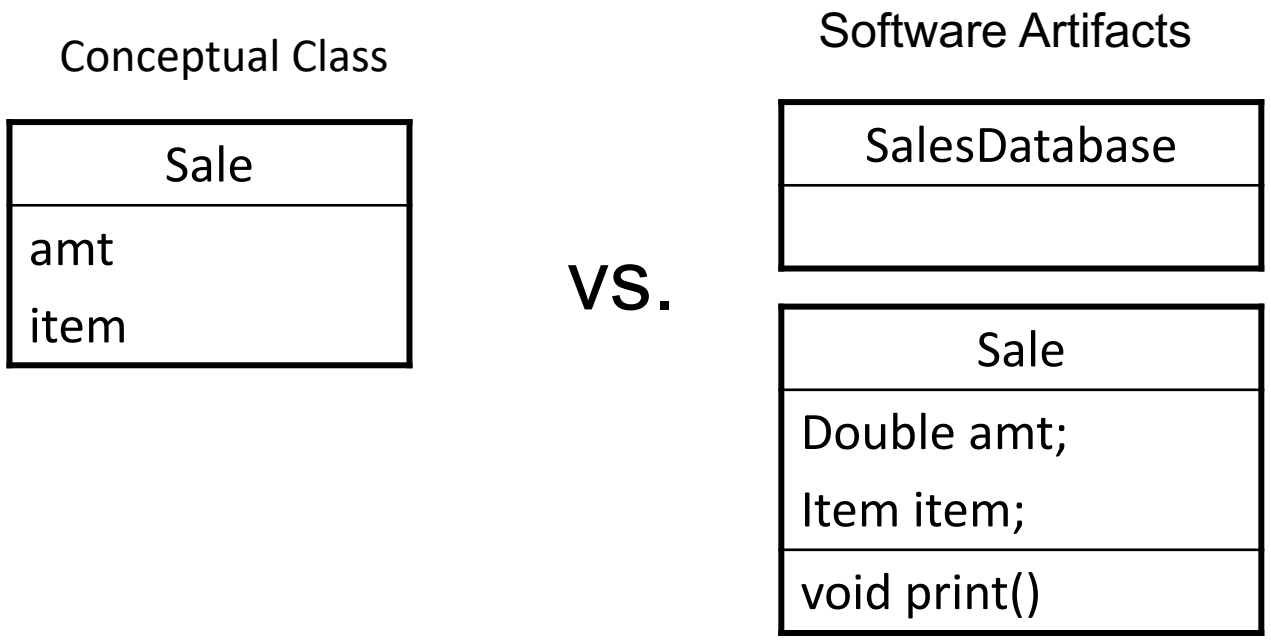
A domain model is used to understand the system as a "transparent box"

Domain Model

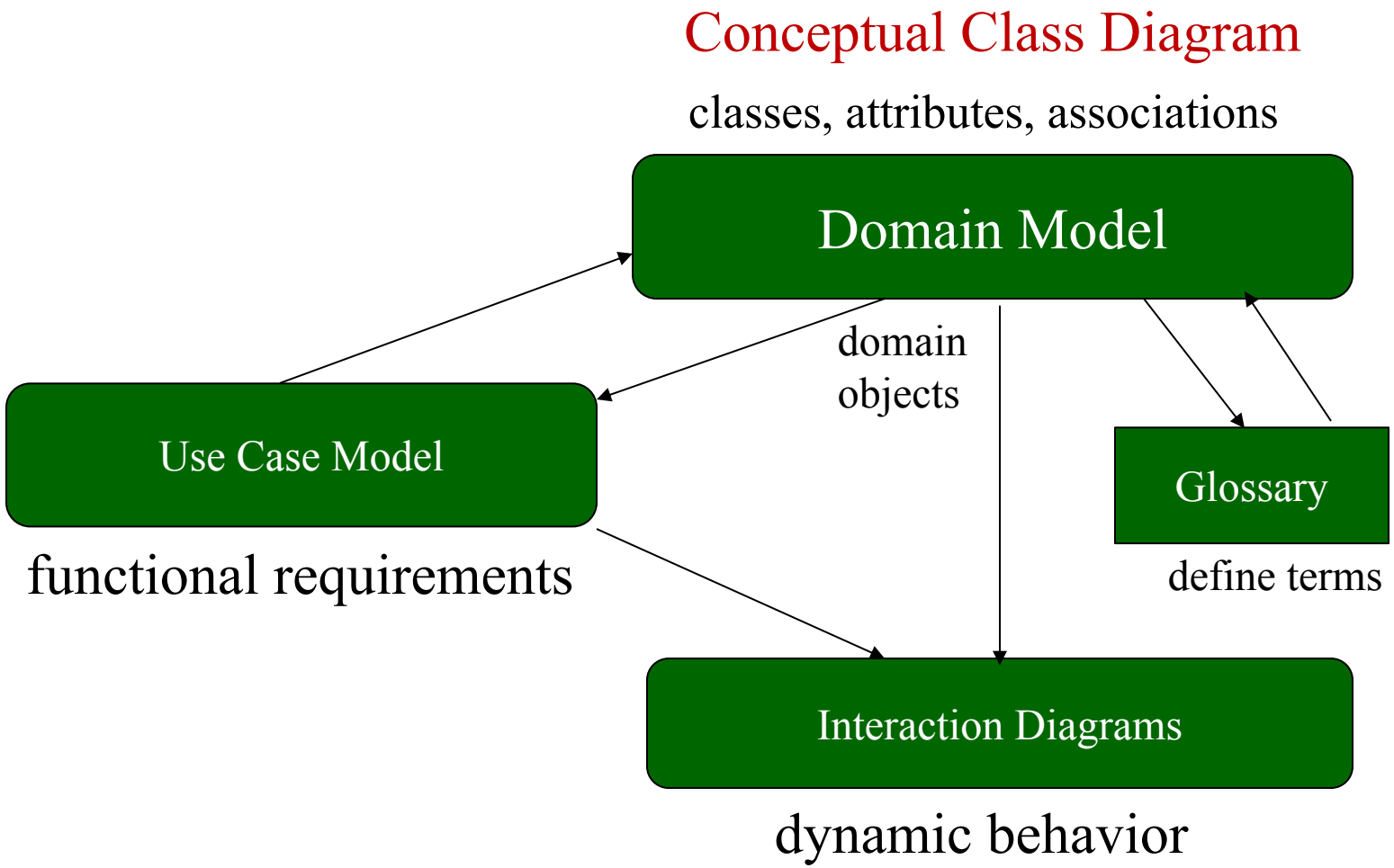
- A **domain model** is a conceptual framework of elements in the problem space
  - Uncovers entities and their static relations that make the black box behave as described by use cases
  - Starts from the “periphery” (or boundary) of the system
    - A boundary object translates information from an actor into a form that can be used by “internal” objects

Domain Model

- A domain model is conceptual, not a software artifact
- What's the difference?

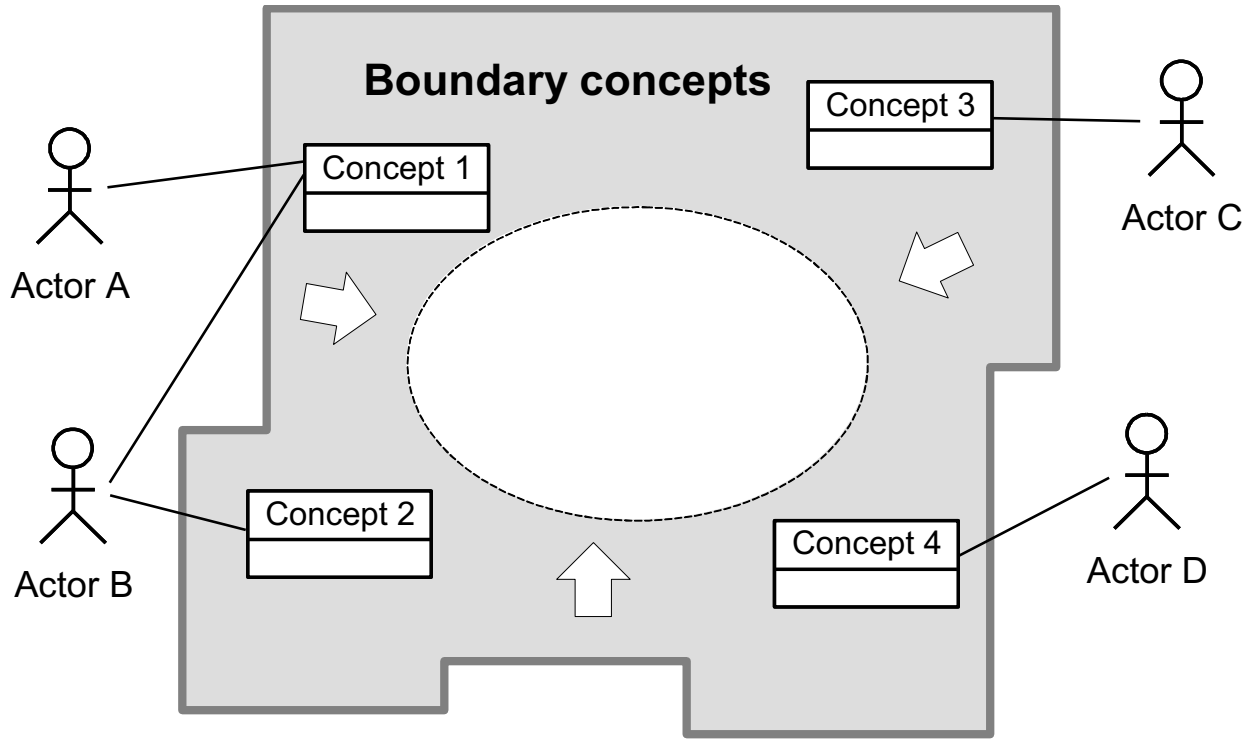


Domain Model Relationships



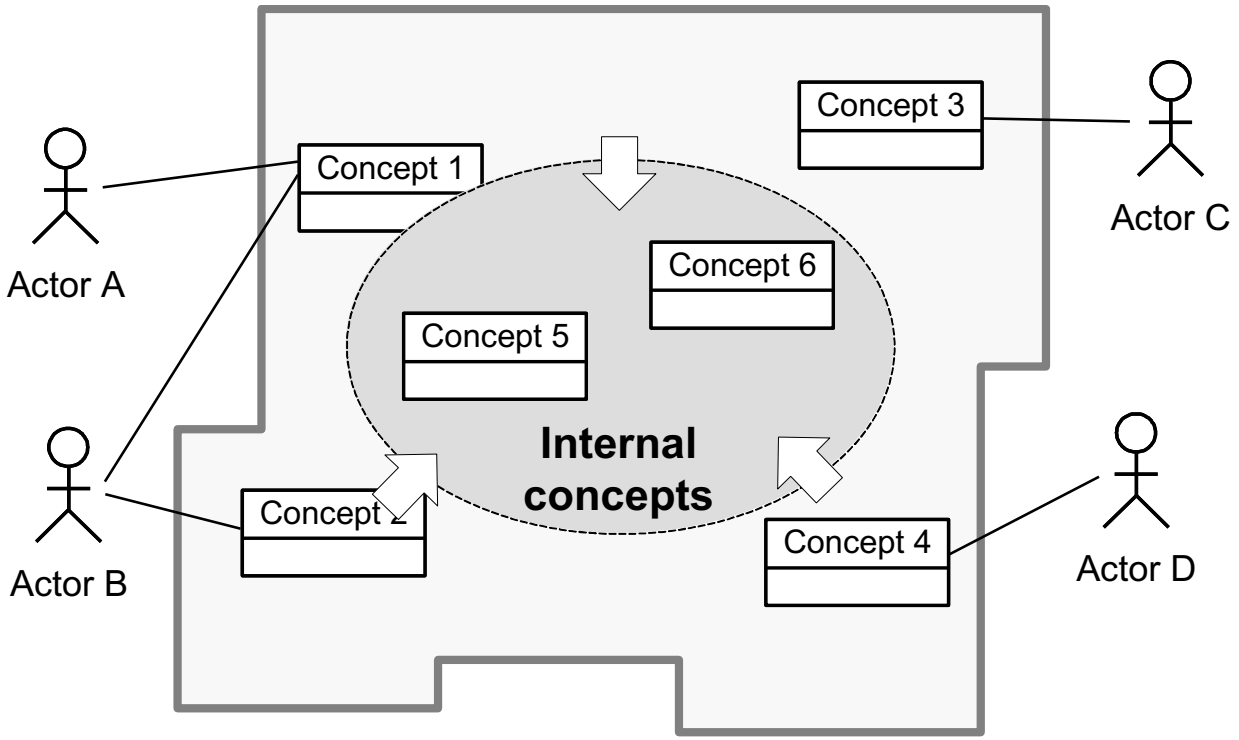
Building a Domain Model

- **Step 1:** Identify boundary concepts



Building a Domain Model

- **Step 2:** Identify internal concepts



Identifying Concepts (1/2)

- Identify conceptual classes from noun phrases
  - Linguistic analysis: vision and scope, glossary, use cases
  - However,
    - Words may be ambiguous or synonymous
    - Noun phrases may also be attributes or parameters rather than classes

  e.g.,
    - If it stores state information or it has multiple behaviors, then it's a class
    - If it's just a number or a string, then it's probably an attribute
- Responsibilities can also be studied

Identifying Concepts (2/2)

The ATM verifies whether the customer's card number and PIN are correct.
S**C** V O**R** O**A** O **A**
If it is, then the customer can check the account balance, deposit cash, and withdraw cash.
S **R** V O**A** V O**A** V O**A**
Checking the balance simply displays the account balance.
S **M** O **A** V O**A**
Depositing asks the customer to enter the amount, then updates the account balance.
S **M** V S **R** V O **A** V O **A**
Withdraw cash asks the customer for the amount to withdraw; if the account has enough cash,
S **M** O**A** V S **R** O **A** V S **C** V O **A**
the account balance is updated. The ATM prints the customer's account balance on a receipt.
O **A** V S**C** V O**A** O

- Analyze each **subject** and **object**
- Identify actor, method, attribute, conceptual class (concept)
- Verbs can also be classes
  - e.g., Deposit is a class if it retains state information

Case Study: Secure Home Access

Use Case UC-1:	**Unlock**
Related Requirements:	REQ1, REQ2, REQ3 and REQ4
Primary Actor:	Any of: Tenant, Landlord
Actor's Goal:	To disarm the lock and get space lighted up automatically
Secondary Actors:	LockDevice, LightSwitch, Timer
Preconditions:	• The set of valid keys stored in the system database is non-empty • The system displays the menu of available functions • At the door keypad the menu choices are "Lock" and "Unlock"
Post conditions:	The auto-lock timer has started count down from autoLockInterval

Flow of Events for Main Success Scenario:

→ 1. **Tenant/Landlord** arrives at the door and selects the menu item "Unlock"

2. include::*AuthenticateUser* (UC-7)

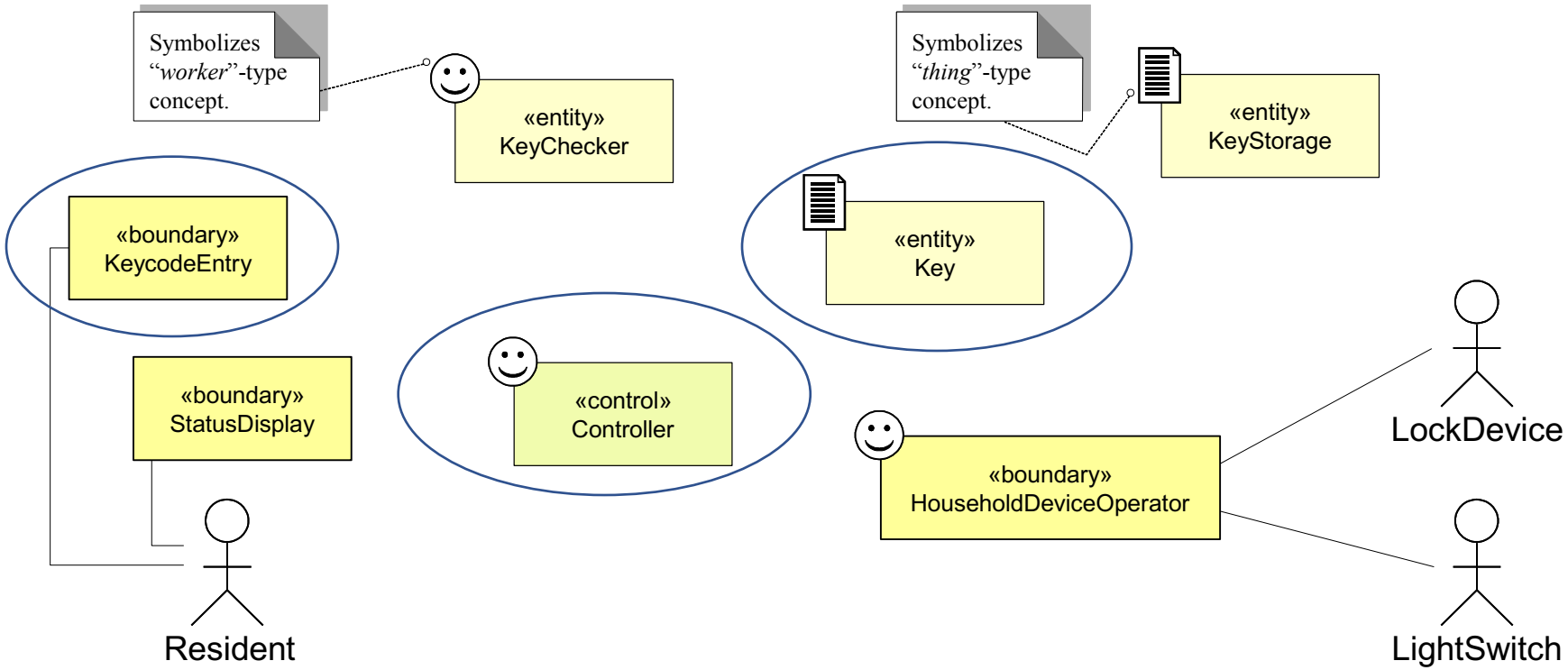
← 3. **System** (a) signals to the **Tenant/Landlord** the lock status, e.g., "disarmed," (b) signals to **LockDevice** to disarm the lock, and (c) signals to **LightSwitch** to turn the light on

← 4. **System** signals to the **Timer** to start the auto-lock timer countdown

→ 5. **Tenant/Landlord** opens the door, enters the home [and shuts the door and locks]

Domain Model

- Partial domain model for UC-1



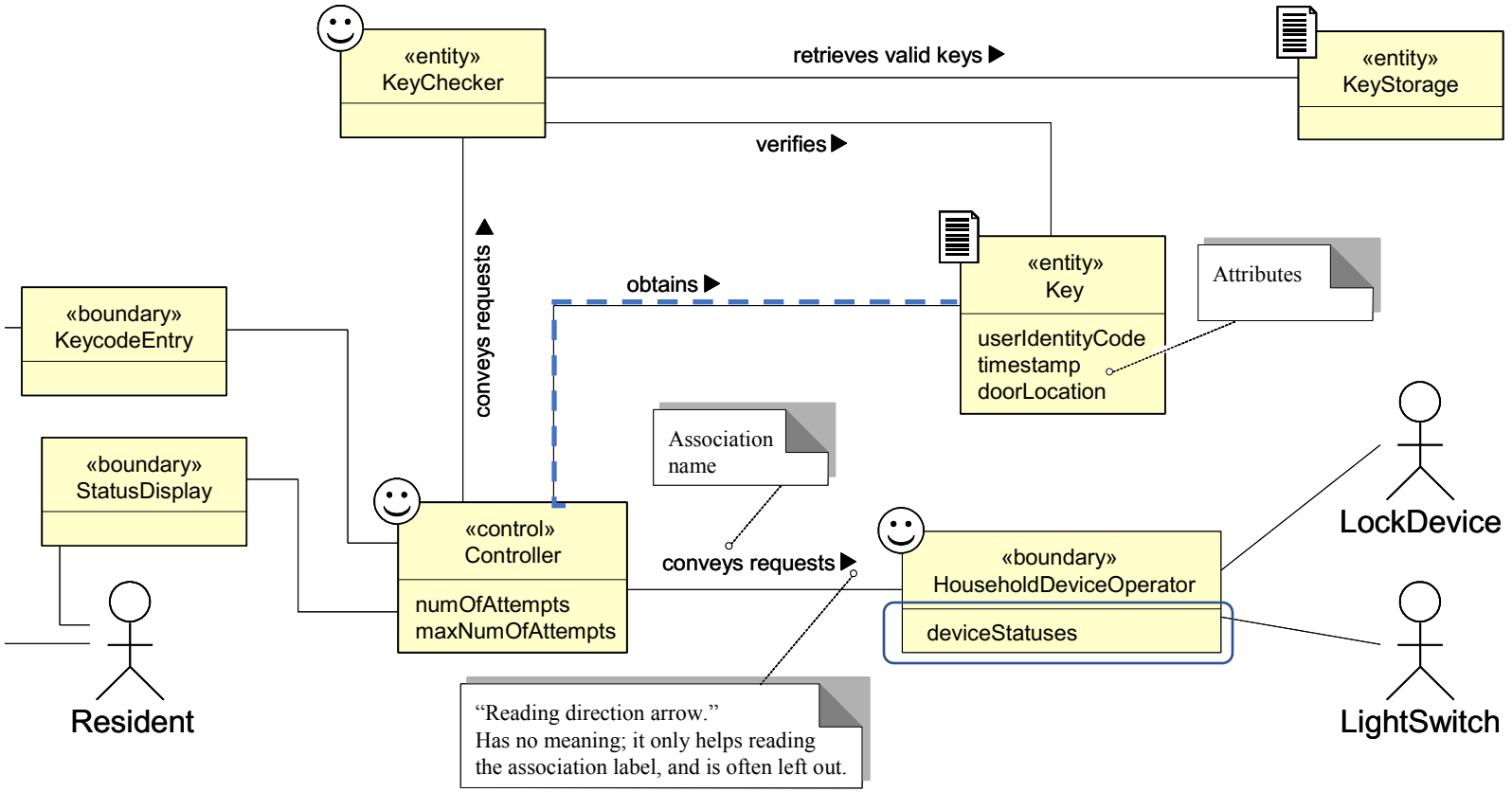
Concepts Association

- Concepts can be associated to describe
  - **Who needs to work together and why**
  - *Not how they work together*

- Concepts are linked by lines
  - Each line has a name
    - Indicate collaboration anticipated between the linked concepts
  - Optional “reading direction arrow” shown as ► (don’t consider this a function call)
  - Logically, associated objects belong to the same package

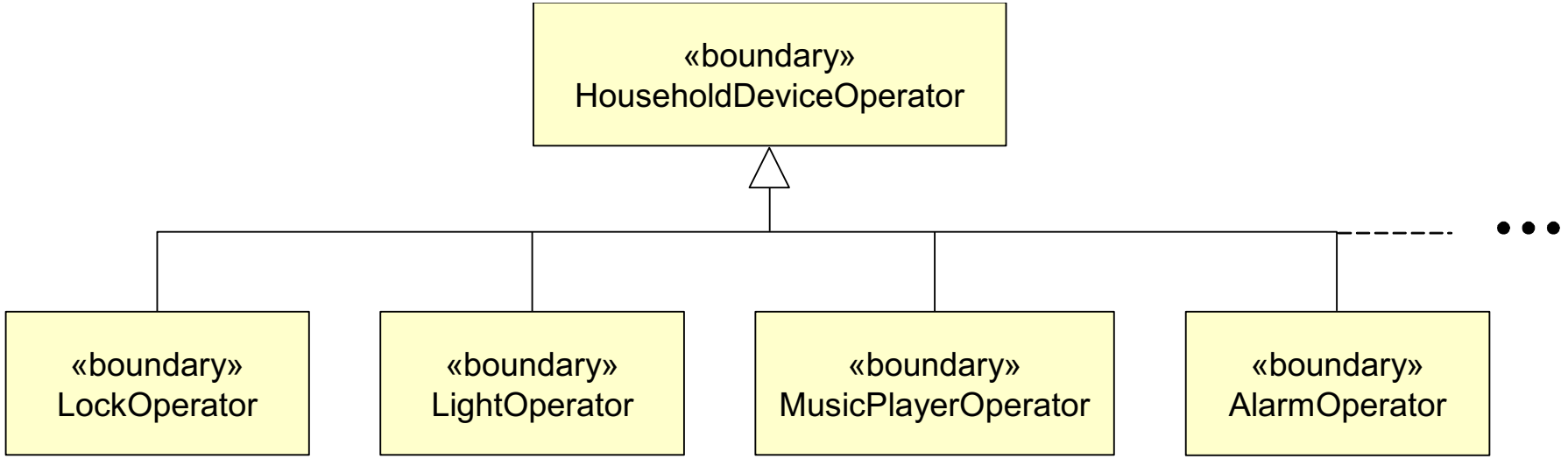
Concepts Association

- Concept association of UC-1

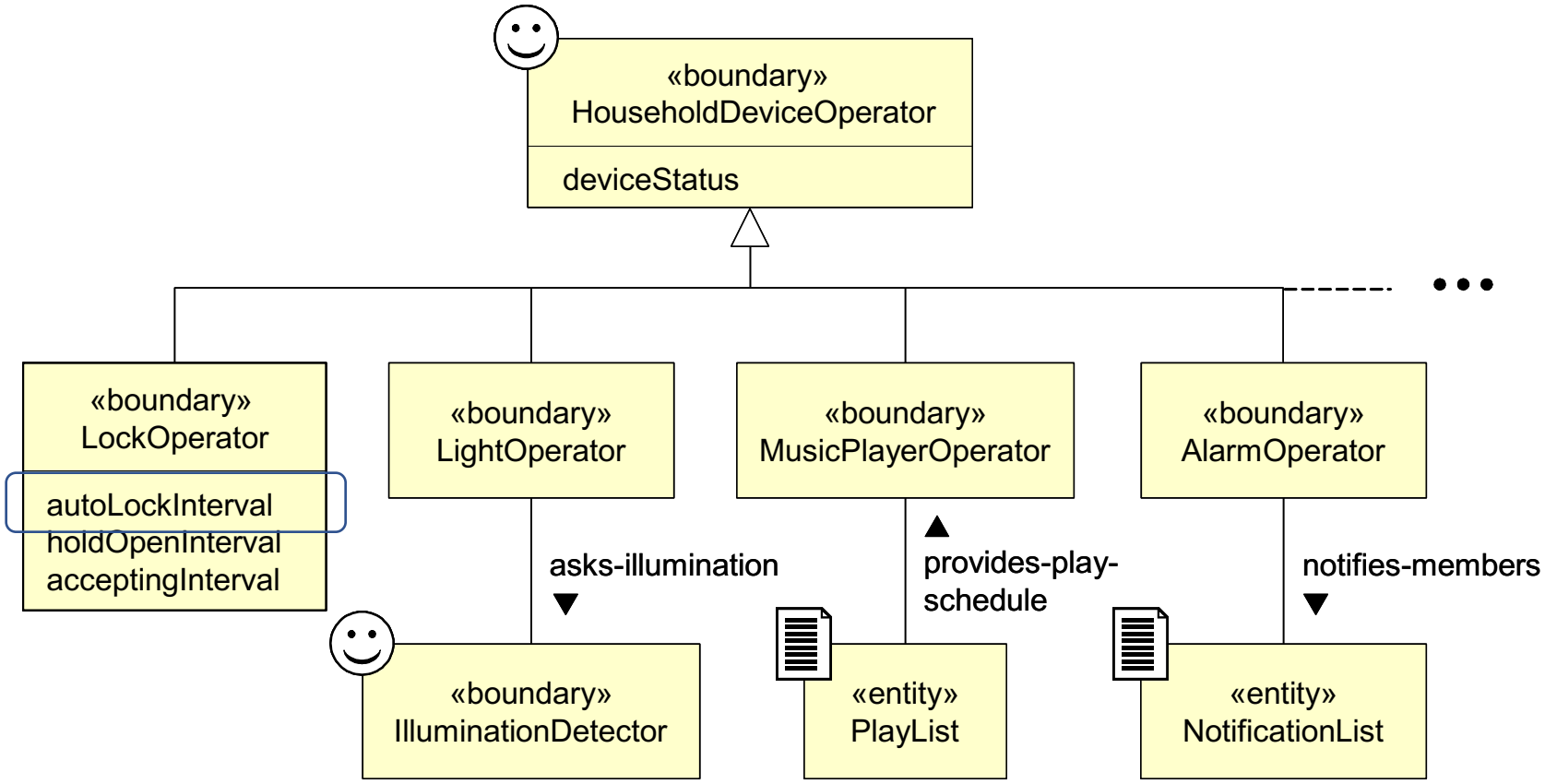


Generalization of Concept

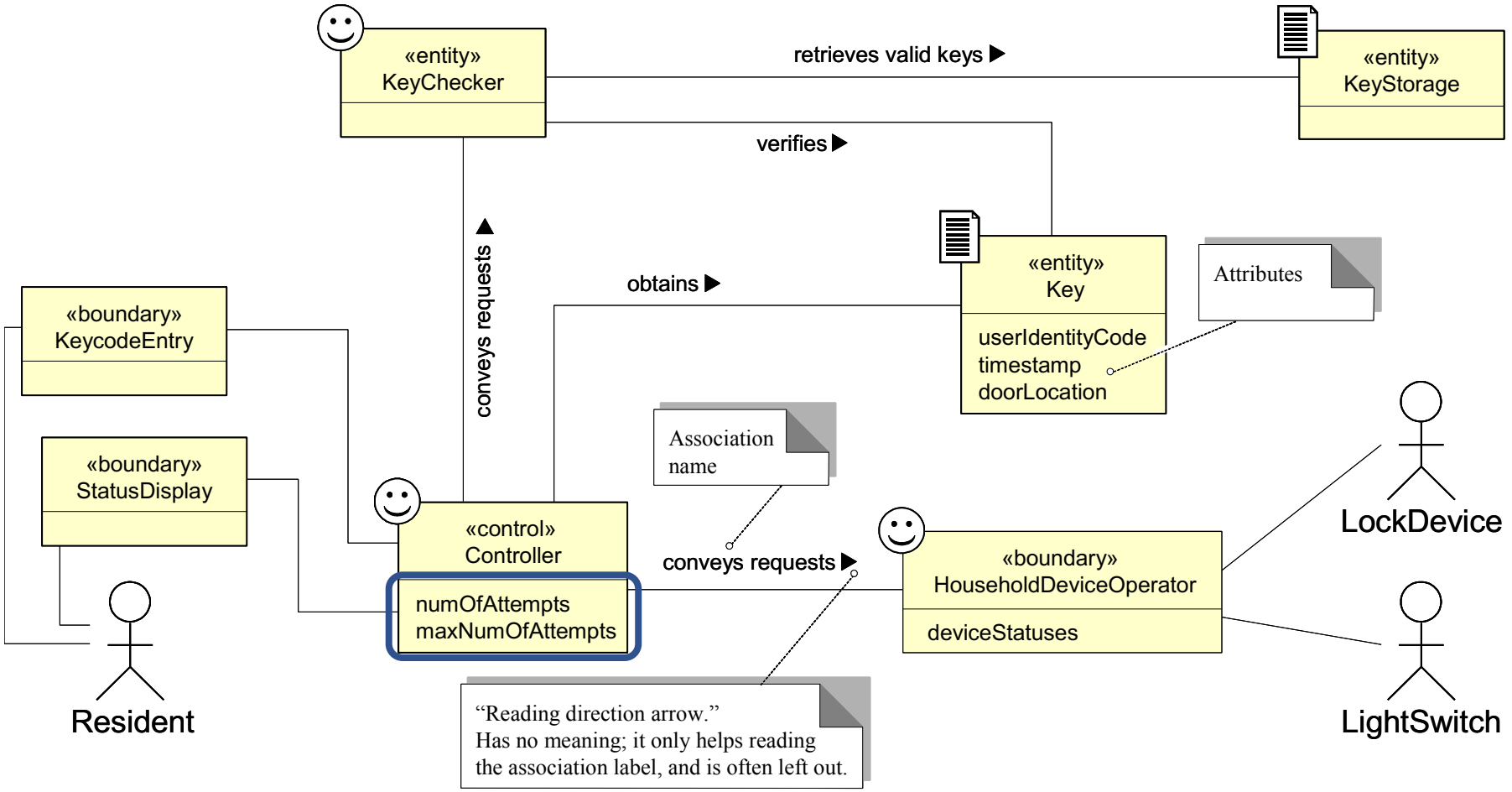
- Device operators may share common properties
- Different devices have specific needs



Need more concepts



Assigning Attributes Carefully



Assigning Attributes Carefully

- Consider how likely two users will input keys in two minutes
- Define a new concept: maxAttemptPeriod
- *Needs an attempts-counting worker* to check the number of attempts and take actions:
  - if numOfAttempts ≥ maxNumOfAttempts, sound the alarm bell and reset numOfAttempts = 0
  - reset numOfAttempts = 0 after a specified amount of time (if the user discontinues the attempts before reaching maxNumOfAttempts)
  - reset numOfAttempts = 0 after a valid key is presented

References

- Prof. Fengjun Li's EECS 448 Fall 2015 slides

- This slide set has been extracted and updated from the slides designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014) by Roger Pressman