CSE4006: Software Engineering

# Lab 8: Requirements Modeling : Behavior, Patterns, and Web/Mobile Apps

## Software Engineering Lab



lab(se);

# Create Behavior Model

It can represent the behavior of the system as a function of specific events and time.

1. Evaluate all use-cases to understand the sequence of interaction within the system.
2. Identify events that drive the interaction sequence and how these events relate to specific objects.
3. Create a sequence or event-trace for each use-case.
4. Build a state transition diagram for the system.
5. Review the behavior model to verify accuracy and consistency.

# Create Behavior Model (cont)

## Identifying Events With The Usecase

The homeowner uses the keypad to key in a four-digit password. The password is compared with the valid password stored in the system. If the password is incorrect, the control panel will beep once and reset itself for additional input. If the password is correct, the control panel awaits further action.

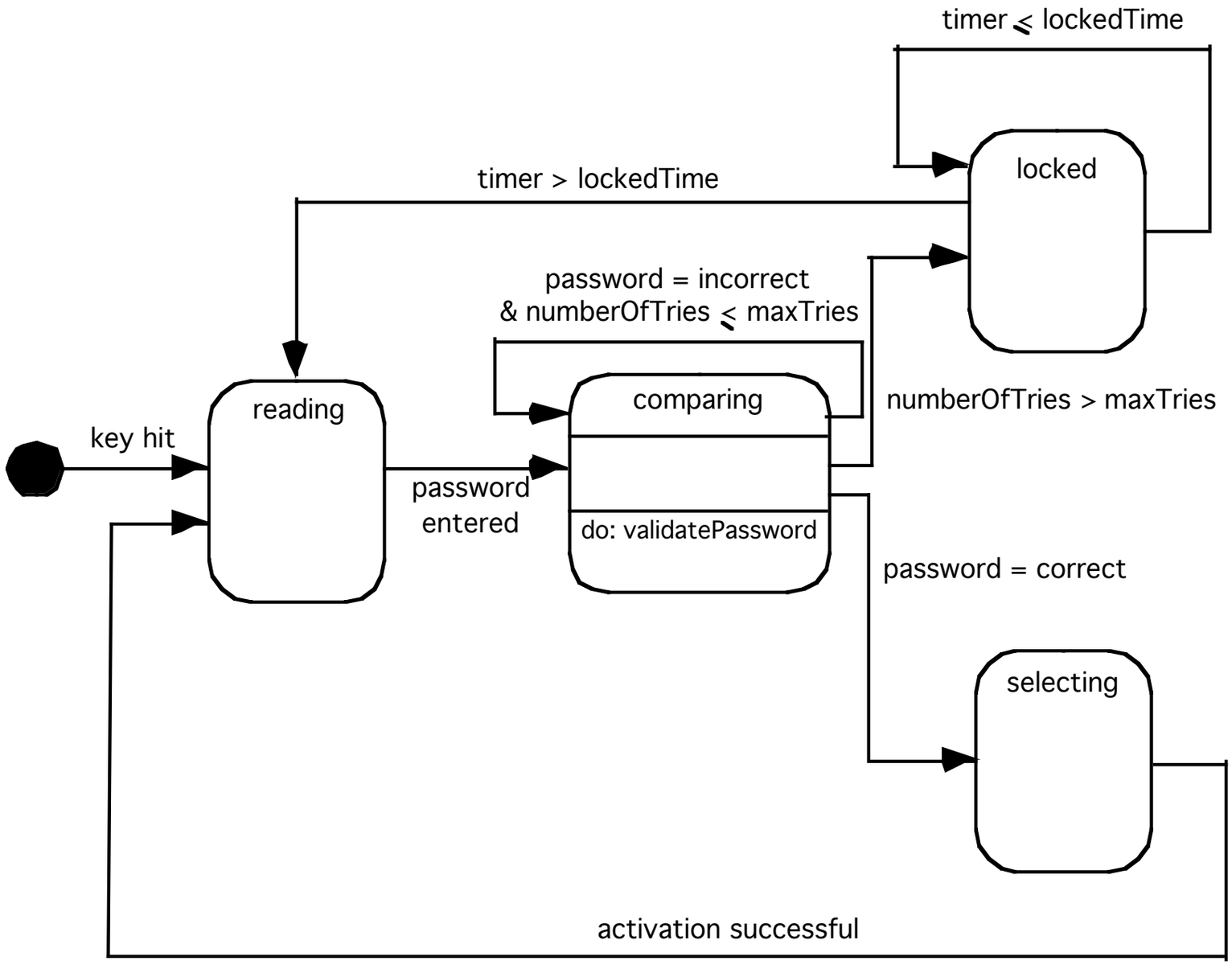
# Create Behavior Model
# - State Representations

- A state transition diagrams (STD) represents the system states and events that trigger state transitions.

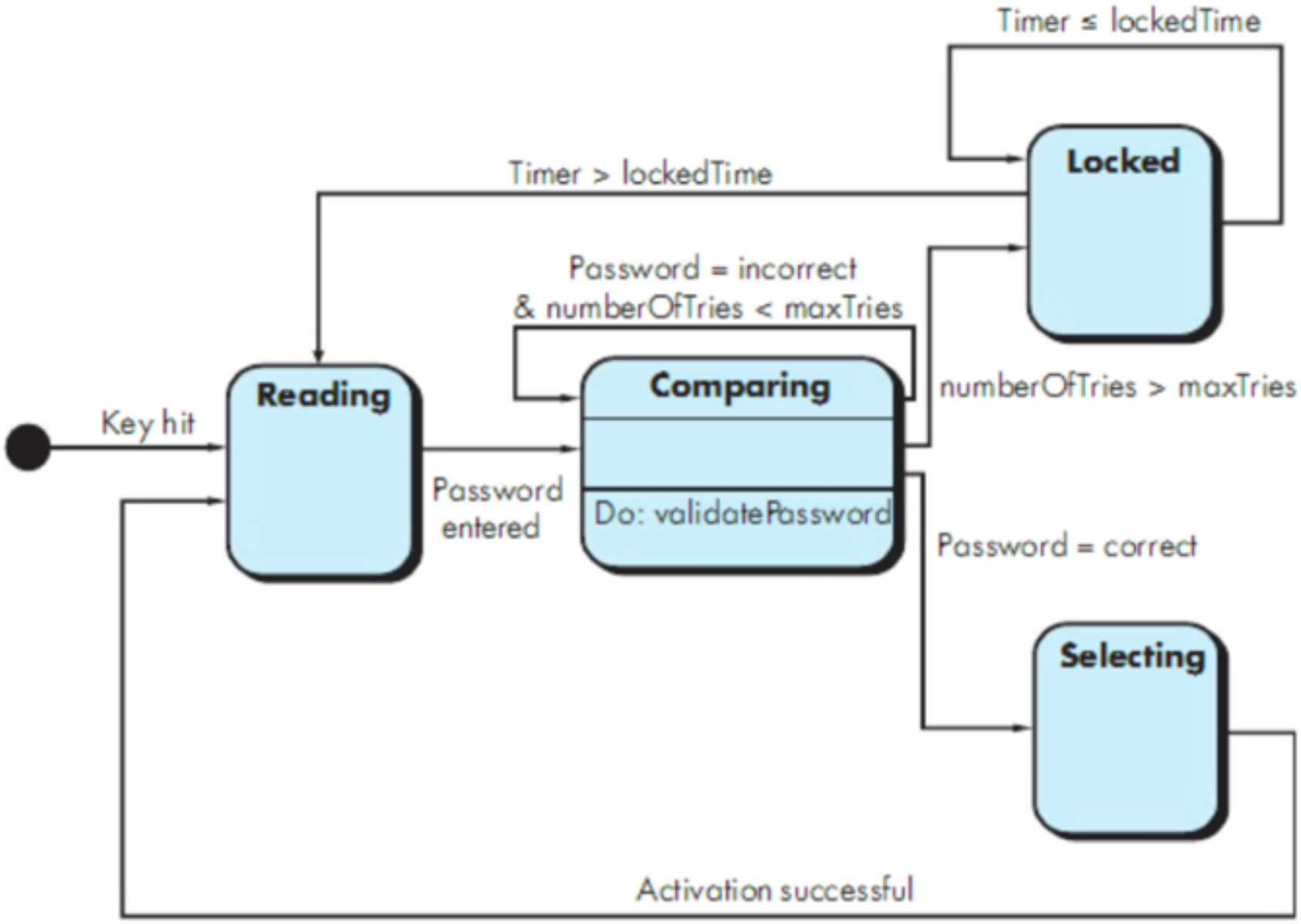
- The state of each class as the system performs its function.

- A state is any observable mode of behavior.

# Create Behavior Model
# - State Representations, Example

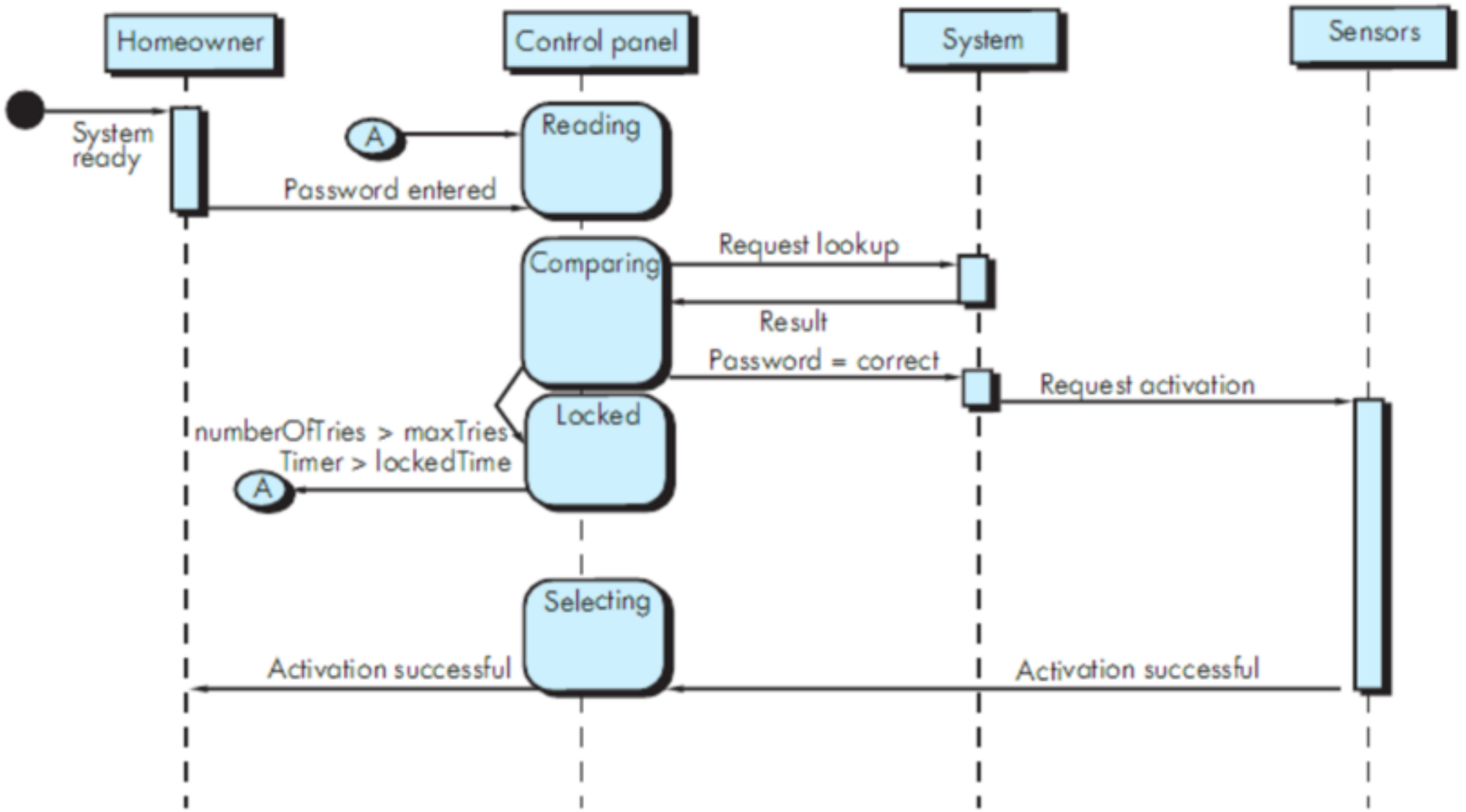


# Create Behavior Model
# - State Representations, Example




lab(se);

# Create Behavior Model
# - State Representations, Sequence Diagram




lab(se);

# Patterns for Requirements Modeling

- Software patterns are a mechanism for capturing domain knowledge in a way that allows it to be reapplied when a new problem is encountered.
- domain knowledge can be applied to a new problem within the same application domain
- the domain knowledge captured by a pattern can be applied by analogy to a completely different application domain.
- The original author of an analysis pattern does not “create” the pattern, but rather, discovers it as requirements engineering work is being conducted.
- Once the pattern has been discovered, it is documented

# Patterns for Requirements Modeling - Exmaple

**Use case: Monitor reverse motion**

**Description**: When the vehicle is placed in reverse gear, the control software enables a video feed from a rear-placed video camera to the dashboard display. The control software superimposes a variety of distance and orientation lines on the dashboard display so that the vehicle operator can maintain orientation as the vehicle moves in reverse. The control software also monitors a proximity sensor to determine whether an object is inside 10 feet of the rear of the vehicle. It will automatically break the vehicle if the proximity sensor indicates an object within 3 feet of the rear of the vehicle.

# Patterns for Requirements Modeling - Exmaple

**Pattern Name: Actuator-Sensor**

**Intent**: Specify various kinds of sensors and actuators in an embedded system.

Motivation: Embedded systems usually have various kinds of sensors and actuators. These sensors and actuators are all either directly or indirectly connected to a control unit. Although many of the sensors and actuators look quite different, their behavior is similar enough to structure them into a pattern. The pattern shows how to specify the sensors and actuators for a system, including attributes and operations. The Actuator-Sensor pattern uses a pull mechanism (explicit request for information) for PassiveSensors and a push mechanism (broadcast of information) for the ActiveSensors.

**Constraints:**

Each passive sensor must have some method to read sensor input and attributes that represent the sensor value.

Each active sensor must have capabilities to broadcast update messages when its value changes.

Each active sensor should send a life tick, a status message issued within a specified time frame, to detect malfunctions.

Each actuator must have some method to invoke the appropriate response determined by the ComputingComponent.

Each sensor and actuator should have a function implemented to check its own operation state.

Each sensor and actuator should be able to test the validity of the values received or sent and set its operation state if the values are outside of the specifications.

# Requirements Modeling for Web and Mobile Apps

## How much analysis is enough ?

- the size and complexity of the application increment.
- the number of stakeholders(analysis can help to identify  conflicting requirements coming from different sources)
- the size of the app development team
- the degree to which members of the team have worked together before(analysis can help develop a common understanding of there project)
- the degree to which the organization’s success is directly dependent on the success of the application.

# Requirements Modeling for Web and Mobile Apps

- Inputs
  - Any information collected during communication activity.
- Outptus
  - Models for WebApp content, function, user interaction, environment, infrastructure.

# Requirements Modeling for Web and Mobile Apps

## Content Model

- The full spectrum of content to be provided by the WebApp is identified, including text, graphics and images, video, and audio data.
- Content objects can be determined directly from useless by examine the scenario description for direct and indirect references to content.
- Attributes of each content object are identified.
- The relationships among content objects and/or the hierarchy of content maintained by a WebApp
  - Relationships : entity-relationship diagram or UML
  - Hierarchy : data tree or UML

# Requirements Modeling for Web and Mobile Apps

## Interaction Model

### Composed of four elements:

- use-cases
- sequence diagrams
- state diagrams
- a user interface prototype

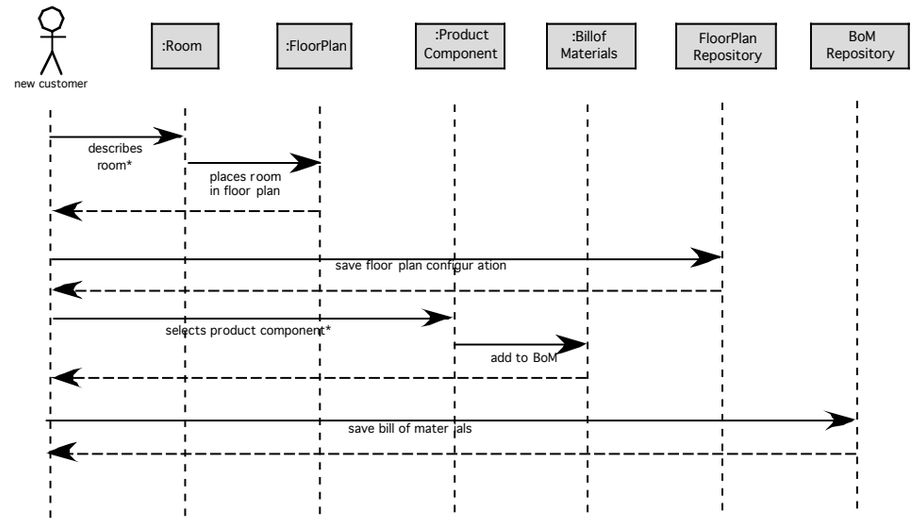



Figure 18.5 Sequence diagram for use-case: *select SafeHome components*

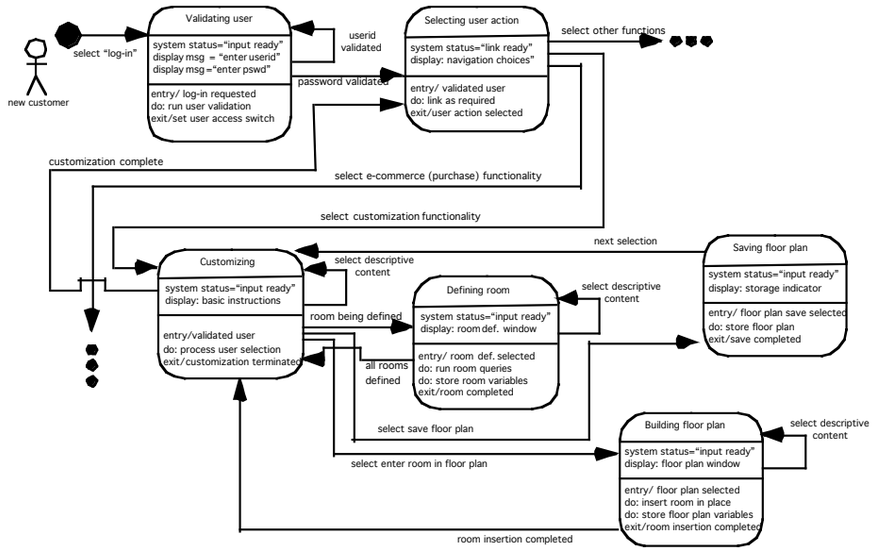



Figure 18.6 Partial state diagram for **new customer** interaction

# Requirements Modeling for Web and Mobile Apps

## Functional Model

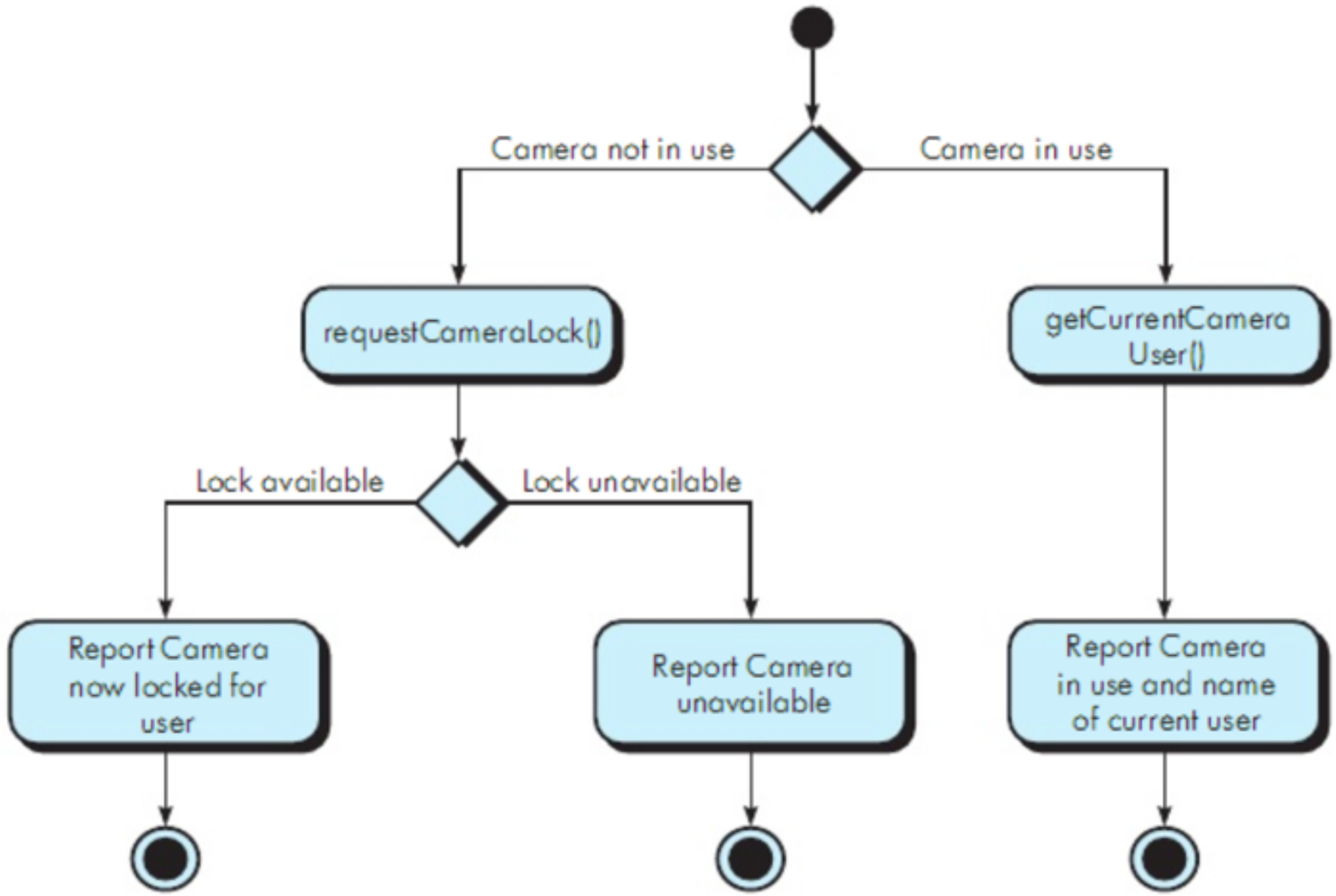
The functional model addresses two processing elements of the WebApp

- user observable functionality that is delivered by the WebApp to end-users
- the operations contained within analysis classes that implement behaviors associated with the class.

An activity diagram can be used to represent processing flow

# Requirements Modeling for Web and Mobile Apps

## Functional Model - Activity Diagram




lab(se);

# The Configuration Model

## Server-side

- Server hardware and operating system environment must be specified
- Interoperability considerations on the server-side must be considered
- Appropriate interfaces, communication protocols and related collaborative information must be specified

## Client-side

- Browser configuration issues must be identified
- Testing requirements should be defined

# Navigation Model

- Software engineers consider requirements that dictate how each type of user will navigate from one content object to another.
- Navigation mechanics are defined as part of design.
- Software engineers and stakeholders must determine navigation requirements.

# Requirements Modeling for Web and Mobile Apps

## Navigation Modeling

- Should certain elements be easier to reach (require fewer navigation steps) than others? What is the priority for presentation?
- Should certain elements be emphasized to force users to navigate in their direction?
- How should navigation errors be handled?
- Should navigation to related groups of elements be given priority over navigation to a specific element.
- Should navigation be accomplished via links, via search-based access, or by some other means?
- Should certain elements be presented to users based on the context of previous navigation actions?
- Should a navigation log be maintained for users?

# Requirements Modeling for Web and Mobile Apps

## Navigation Modeling

- Should a full navigation map or menu (as opposed to a single “back” link or directed pointer) be available at every point in a user’s interaction?
- Should navigation design be driven by the most commonly expected user behaviors or by the perceived importance of the defined WebApp elements?
- Can a user “store” his previous navigation through the WebApp to expedite future usage?
- For which user category should optimal navigation be designed?
- How should links external to the WebApp be handled? overlaying the existing browser window? as a new browser window? as a separate frame?

# References

Software Engineering: A Practitioner's Approach, 8/e, by Roger Pressman, Bruce Maxim