Week 4

CS2212

Introduction to Software Engineering

Behavioural Modeling

Ask Questions Live

cs1.ca/ask

Behavioural Modeling

- A **behavioural model** indicates how software will **respond to internal or external events** or stimuli.

- This information is useful in the creation of an effective design for the system to be built.

- **UML activity diagrams** can be used to model how system elements **respond to internal events**.

- **UML state diagrams** can be used to model how system elements **respond to external events**.

Identifying Events

- A **use case** represents a sequence of activities that involves **actors** and **the system**.
- An **event** occurs whenever the **system and an actor exchange information**.
- An **event** is **not the information** that has been exchanged, but rather the **fact that information has been exchanged**.
- A **use case** needs to be examined for **points of information exchange**.
- **Events** are used to **trigger state transitions**.

Identifying Events: SafeHome Example 1

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.

An **event** occurs whenever the **system and an actor exchange information**.

Identifying Events: SafeHome Example 1

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.

Identifying Events: SafeHome Example 2

Use Case:	Access Camera Surveillance via the Internet and Display Camera Views
Primary Actor:	Homeowner
Secondary Actor:	Cameras System Administrator
Goal in Context:	To view output of cameras placed throughout the house from any remote location via the Internet.
Preconditions:	System must be fully configured; appropriate user ID and passwords must be obtained.
Trigger:	The homeowner decides to take a look inside the house while away.
Scenario:	1. The homeowner loads the SafeHome Products website. 2. The homeowner enters his or her user ID. 3. The homeowner enters their passwords (at least eight characters in length). 4. The system displays all major function buttons. 5. The homeowner selects the “surveillance” button from the major function buttons. 6. The homeowner selects “pick a camera”. 7. The system displays the floor plan of the house. 8. The homeowner selects a camera icon from the floor plan. 9. The homeowner selects the “view” button. 10. The system displays a viewing window that is identified by the camera ID. 11. The system displays video output within the viewing window at one frame per second.

⋮

Identifying Events: SafeHome Example 2

⋮

Alternatives:	1. Homeowner selects "View thumbnail snapshots for all camera" rather than "view a camera" in the surveillance menu – see use case **View thumbnail snapshots for all cameras**. 2. Homeowner selects a different major function button from the main menu – see use case for that major function.
Exceptions:	1. ID or passwords are incorrect or not recognized – see use case **Validate ID and passwords.** 2. Surveillance function not configured for the system – system displays appropriate error message; see use case **Configure surveillance function**. 3. A floor plan is not available or has not been configured – display appropriate error message and see use case **Configure floor plan**. 4. An alarm condition is encountered – see use case **Alarm condition encountered**.
Priority:	Moderate priority, to be implemented after basic functions.
When Available:	Third increment.
Frequency of use:	Infrequent.
Channel to actor:	Via PC-based browser and Internet connection.
Channels to secondary actors:	**System administrator:** PC-based system **Cameras:** wireless connectivity.
Open Issues:	……..

Identifying E

Identify all of the **events** in this **use case**.

Use Case:	Access Camera Surveillance via the Internet and Display Camera Views
Primary Actor:	Homeowner
Secondary Actor:	Cameras System Administrator
Goal in Context:	To view output of cameras placed throughout the house from any remote location via the Internet.
Preconditions:	System must be fully configured; appropriate user ID and passwords must be obtained.
Trigger:	The homeowner decides to take a look inside the house while away.
Scenario:	1. The homeowner loads the SafeHome Products website. 2. The homeowner enters his or her user ID. 3. The homeowner enters their passwords (at least eight characters in length). 4. The system displays all major function buttons. 5. The homeowner selects the “surveillance” button from the major function buttons. 6. The homeowner selects “pick a camera”. 7. The system displays the floor plan of the house. 8. The homeowner selects a camera icon from the floor plan. 9. The homeowner selects the “view” button. 10. The system displays a viewing window that is identified by the camera ID. 11. The system displays video output within the viewing window at one frame per second.

⋮

State Representations

- The state of a system is **a set of observable circumstances** that **characterizes the behaviour of the system** at a given time.

- In the context of behavioural modeling, two different characterizations of states must be considered:

    1. The state of **each class** as the system performs its function.
    2. The state of the system **as observed from the outside** as the system performs its function.

State Representations

- The state of a class takes on both **passive** and **active** characteristics
  - A **passive** state is simply the **current status of all of an object's attributes.**
  - The **active** state of an object indicates **the current status of the object as it undergoes a continuing transformation** or processing.
- An **event**, also sometimes called a **trigger**, must occur to **force an object to make a** state **transition**; in other words, to move from one **active** state to another.
- **Actions** might also occur as a consequence of making a transition.

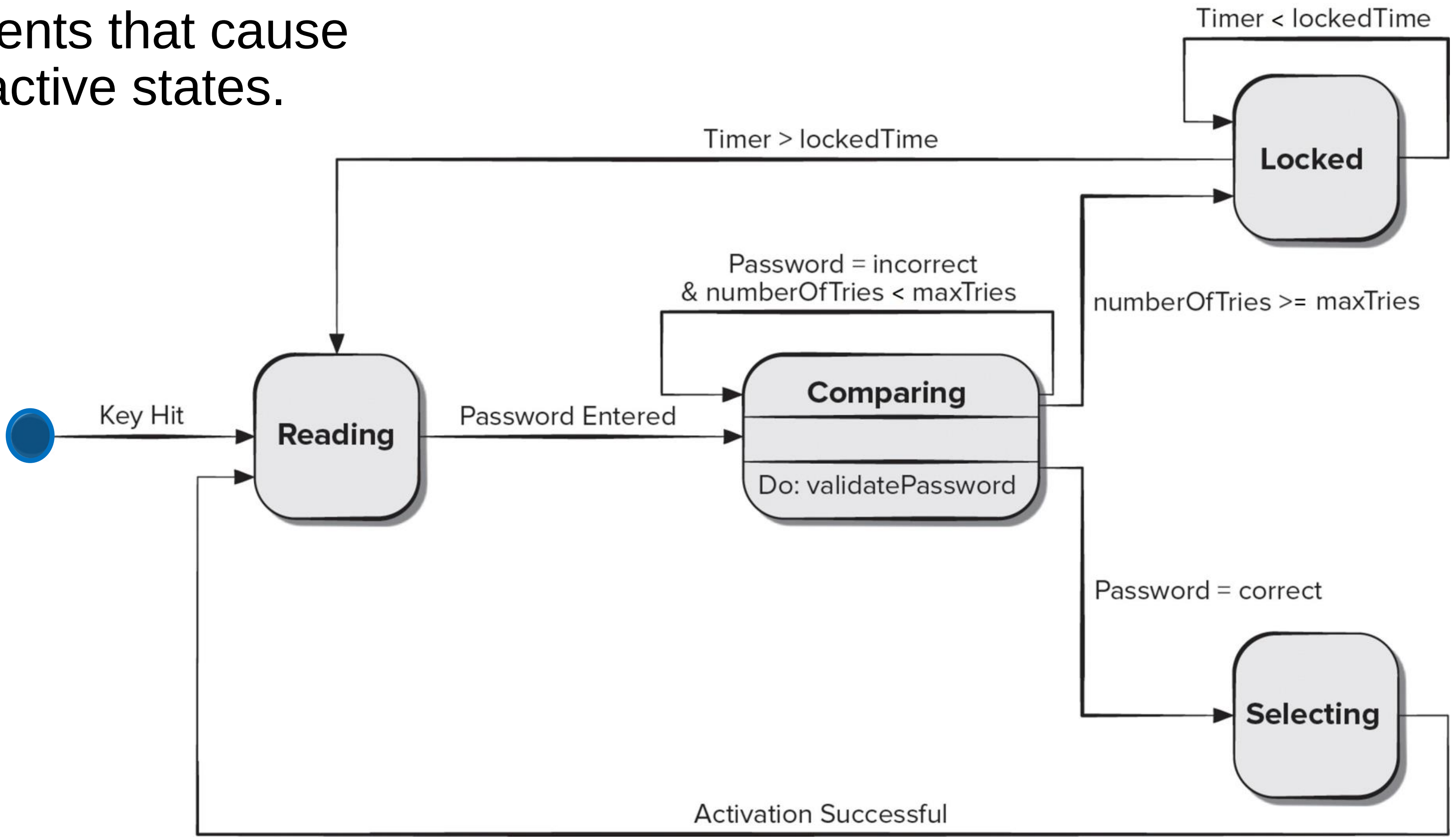
State Representations

- There are several behavioural representations that can be used; we will focus on two here that are part of UML:
  - **State Diagram:** Indicates how an individual class changes state **based on external events**.
  - **Sequence Diagram:** Shows the **behaviour of the software as a function of time**.

State Diagrams

State diagrams represent the active states for each class and the events that cause changes between these active states.

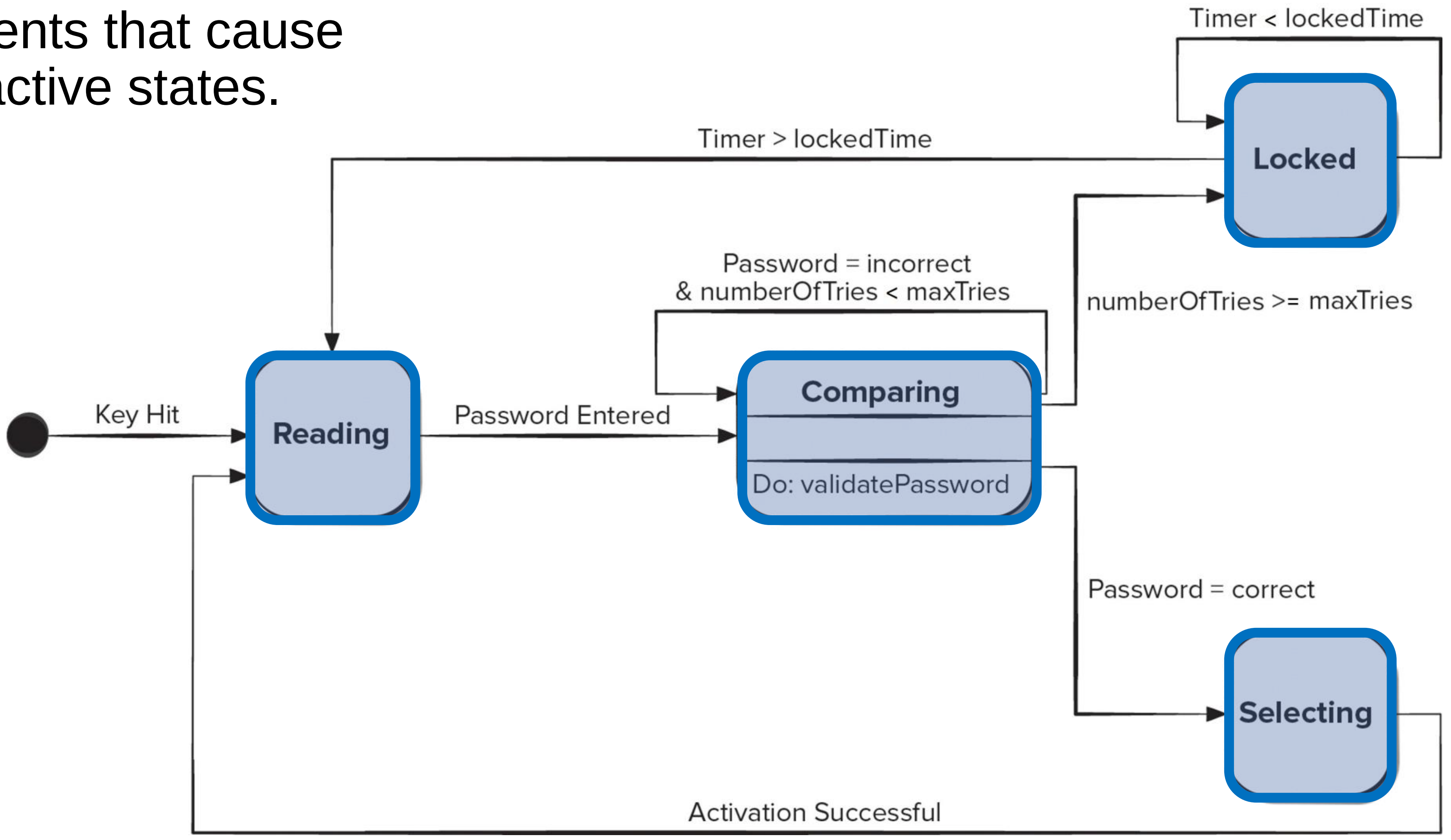
- States
- Transitions
- Guards
- Activity



State Diagrams

State diagrams represent the active states for each class and the events that cause changes between these active states.

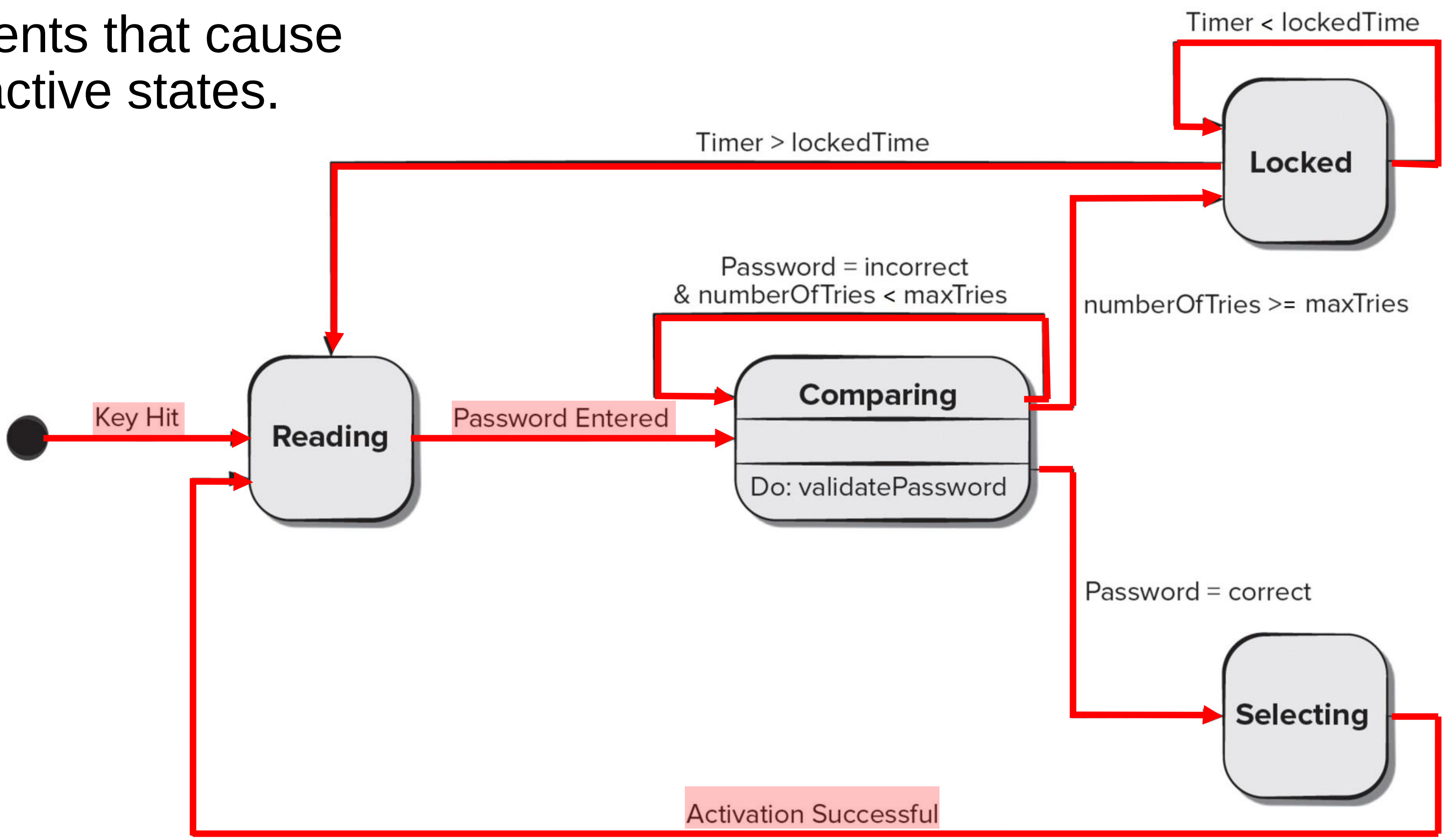
- **States**
- Transitions
- Guards
- Activity



State Diagrams

State diagrams represent the active states for each class and the events that cause changes between these active states.

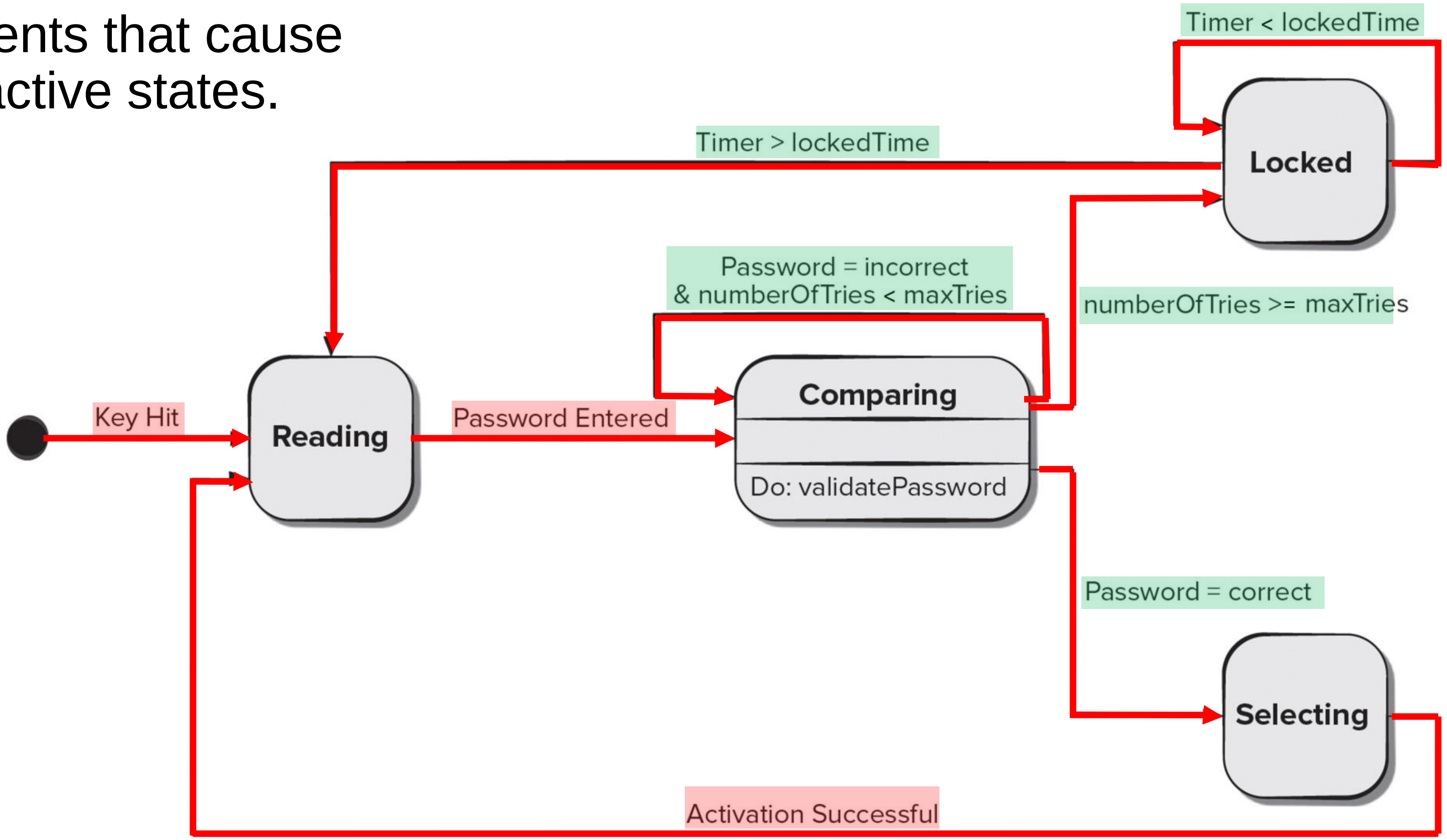
- States
- **Transitions**
- Guards
- Activity



State Diagrams

State diagrams represent the active states for each class and the events that cause changes between these active states.

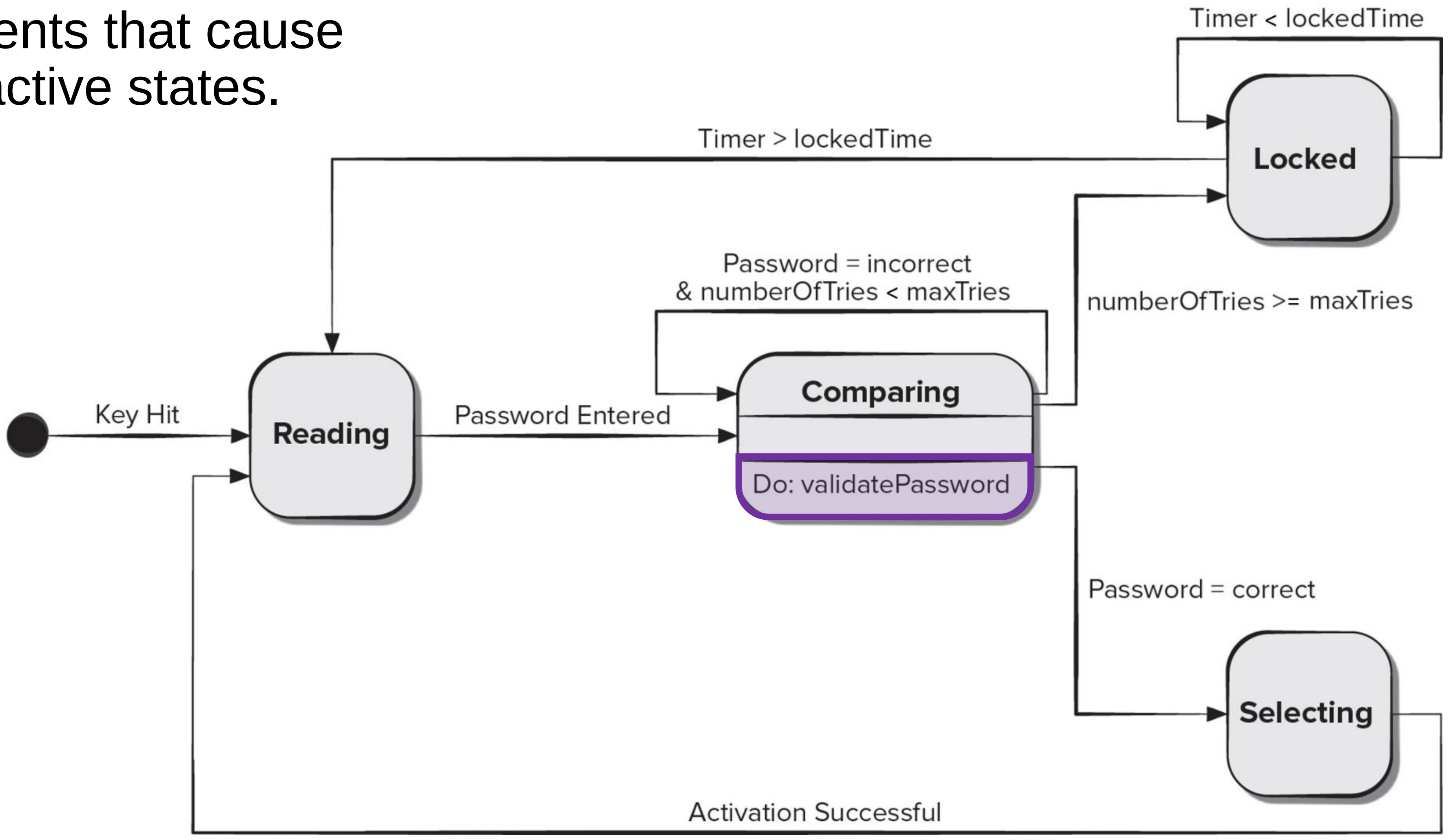
- States
- **Transitions**
- **Guards**
- Activity



State Diagrams

State diagrams represent the active states for each class and the events that cause changes between these active states.

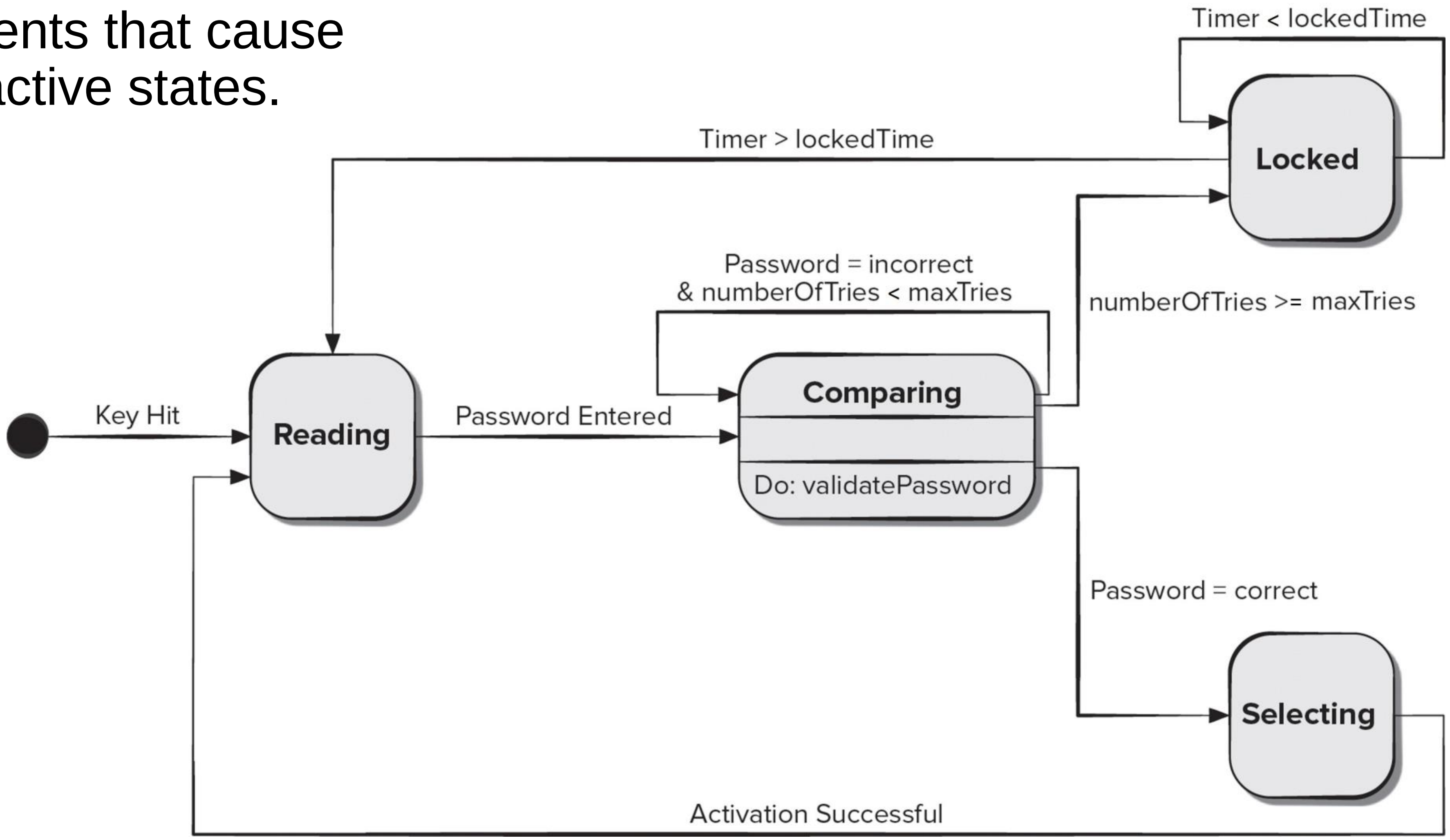
- States
- Transitions
- Guards
- **Activity**



State Diagrams

State diagrams represent the active states for each class and the events that cause changes between these active states.

- States
- Transitions
- Guards
- Activity

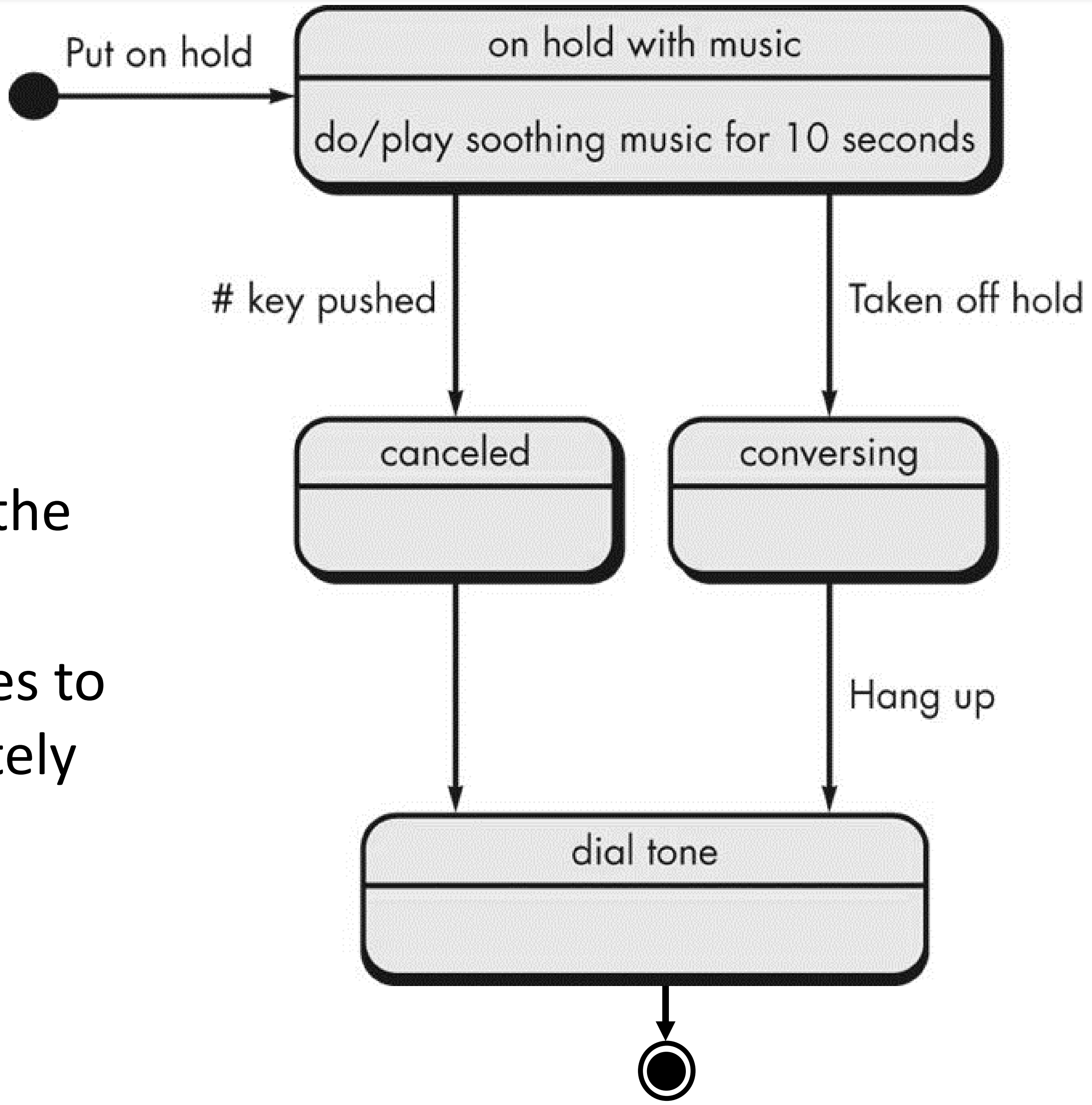


State Diagrams

State diagram for an hold system that plays music when you're on hold.

Notice the lack of trigger/event from the canceled state.

When the # key is pushed, the call goes to the canceled state and then immediately transitions to the dial tone state.



State Diagrams

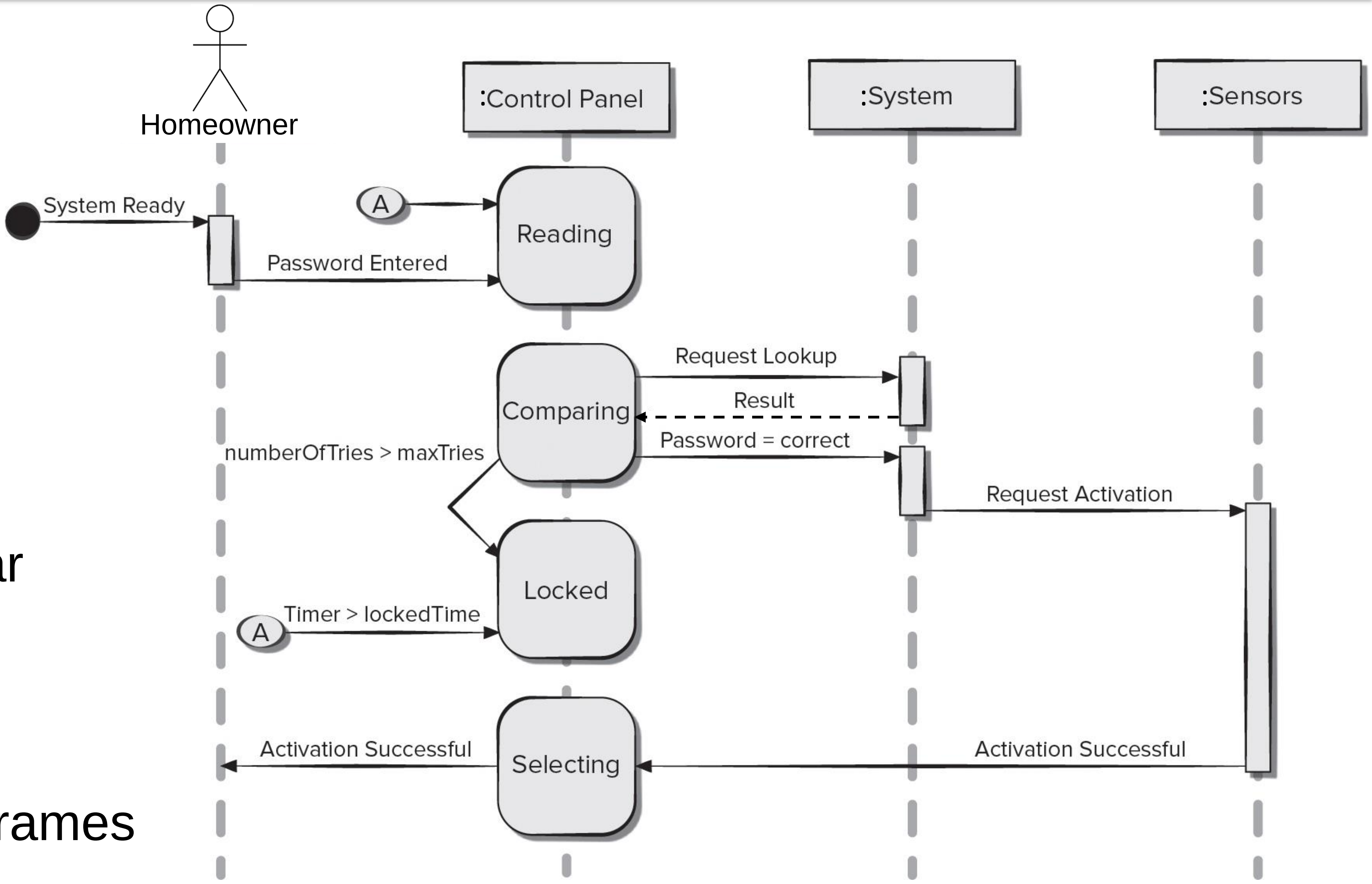
When to Use State Diagrams:

- Used for describing the **behaviour of an object across one or more use cases**.
- Not good at describing the behavior that involves several different objects collaborating.
- Good for showing how an **individual object changes state** based on **external events**.
- **Don’t try to draw them for every class in the system.** Only for classes that exhibiting interesting, complex, or noteworthy behaviour where a state diagram might help with understanding the class.

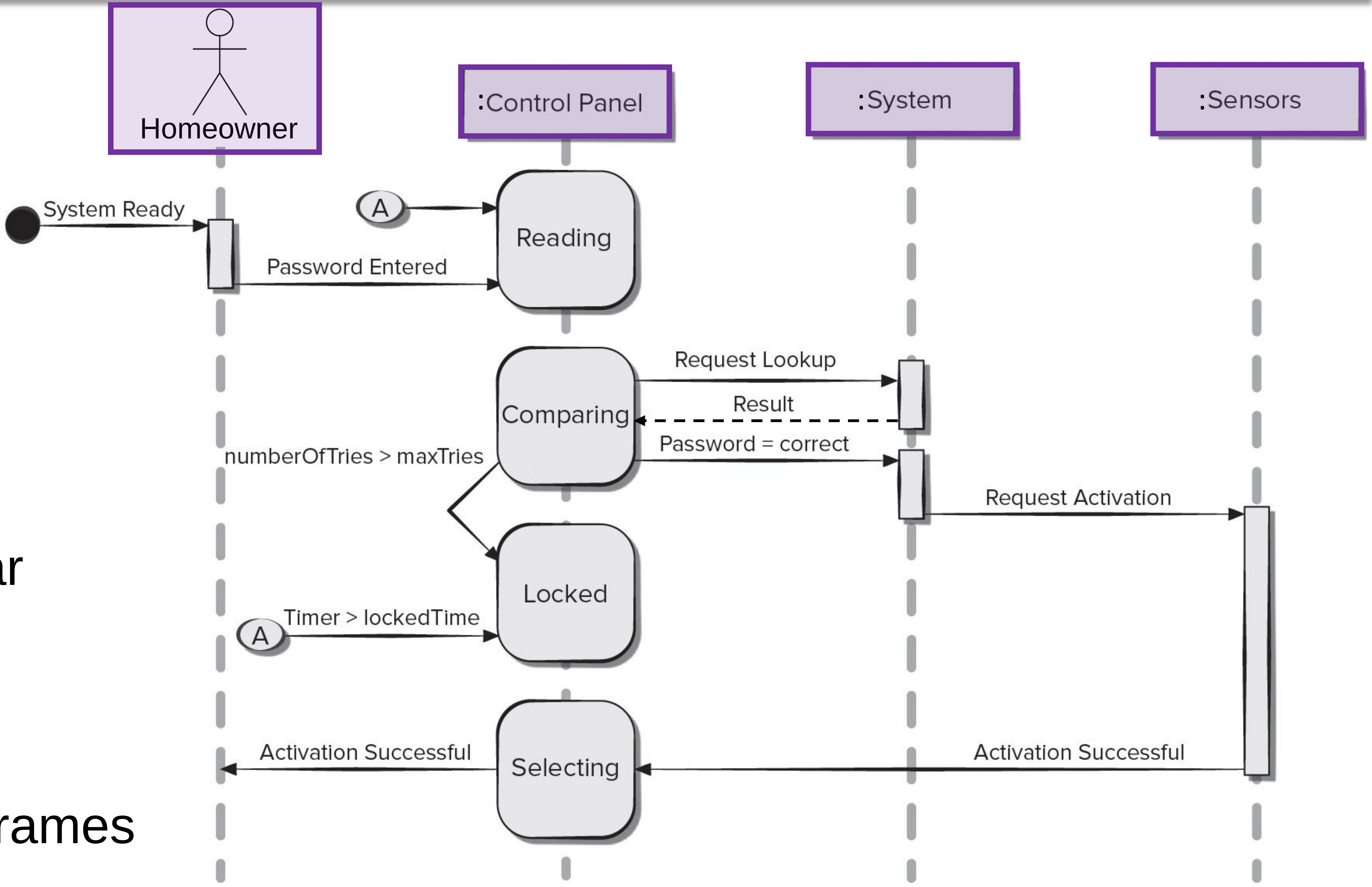
Sequence Diagrams

- The UML **sequence diagram** can be used for **behavioural modeling**.
- **Sequence diagrams** can also be used to show **how events cause transitions** from object to object.
- Once **events** have been identified by examining a **use case**, the modeler creates a **sequence diagram**—a representation of how events cause flow from one object to another as a function of time.
- **Sequence diagram** is a shorthand version of a **use case**.

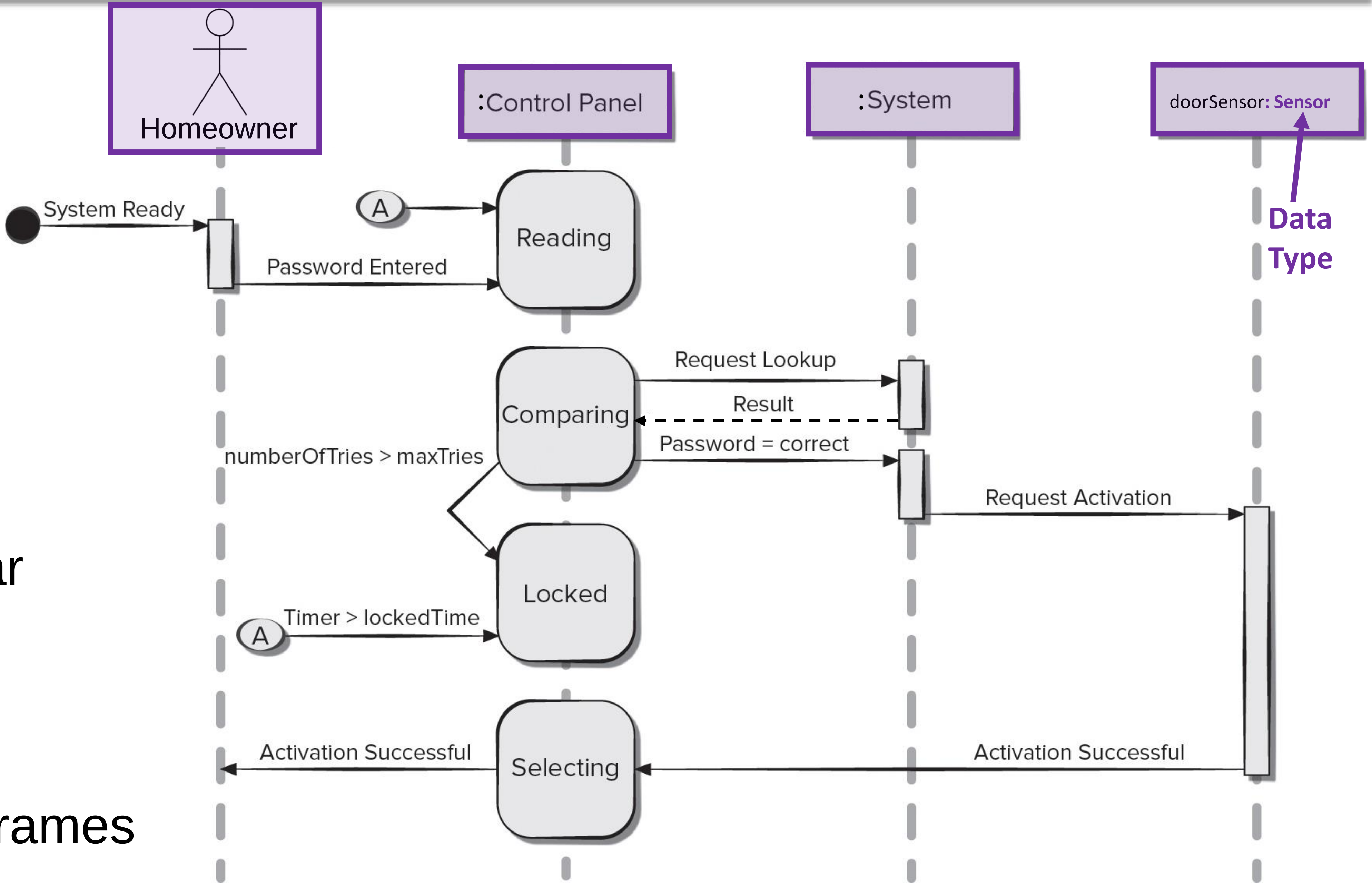
- Objects
- Lifeline
- Activation Bar
- Messages
- Interaction Frames



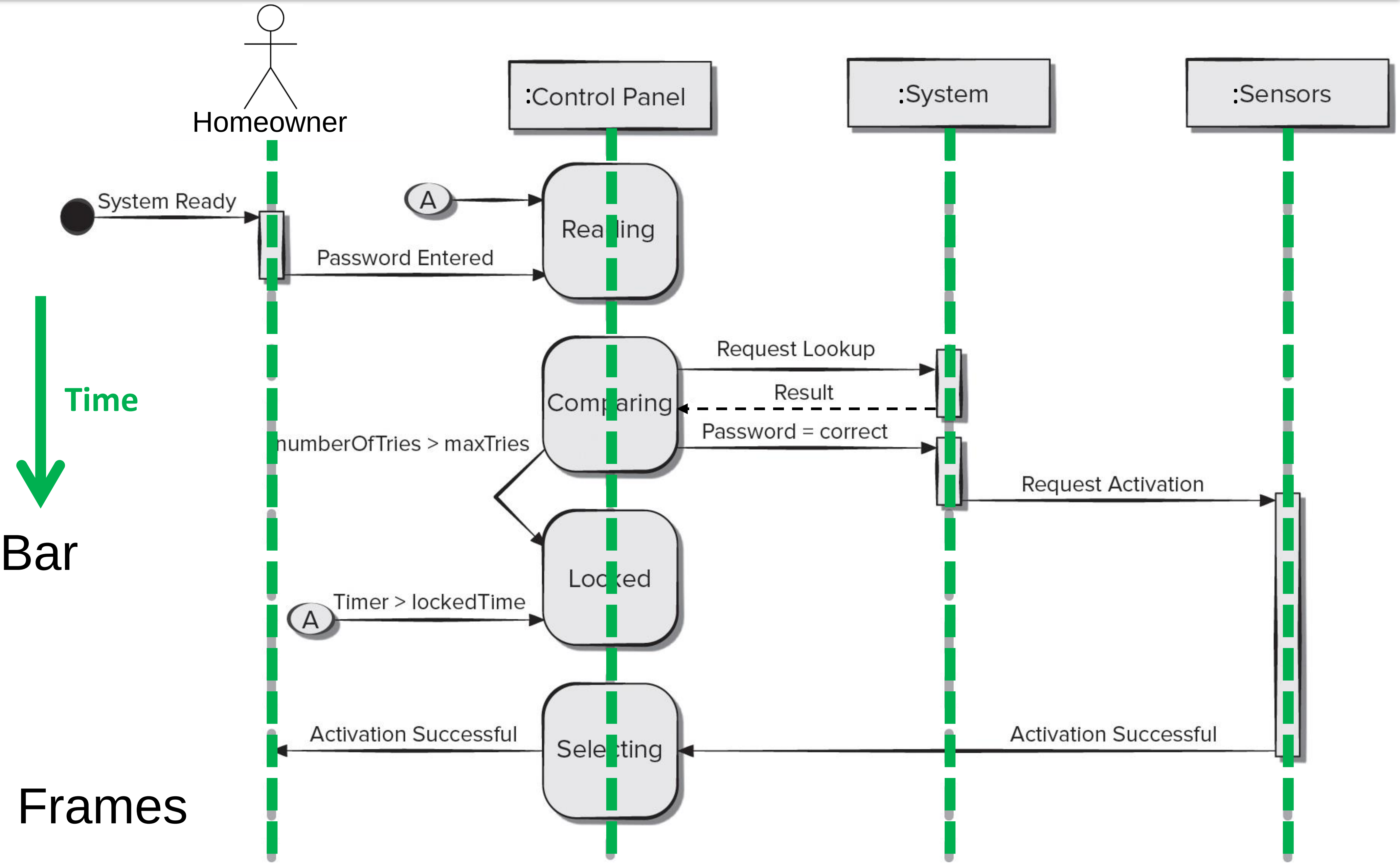
- **Objects**
- Lifeline
- Activation Bar
- Messages
- Interaction Frames



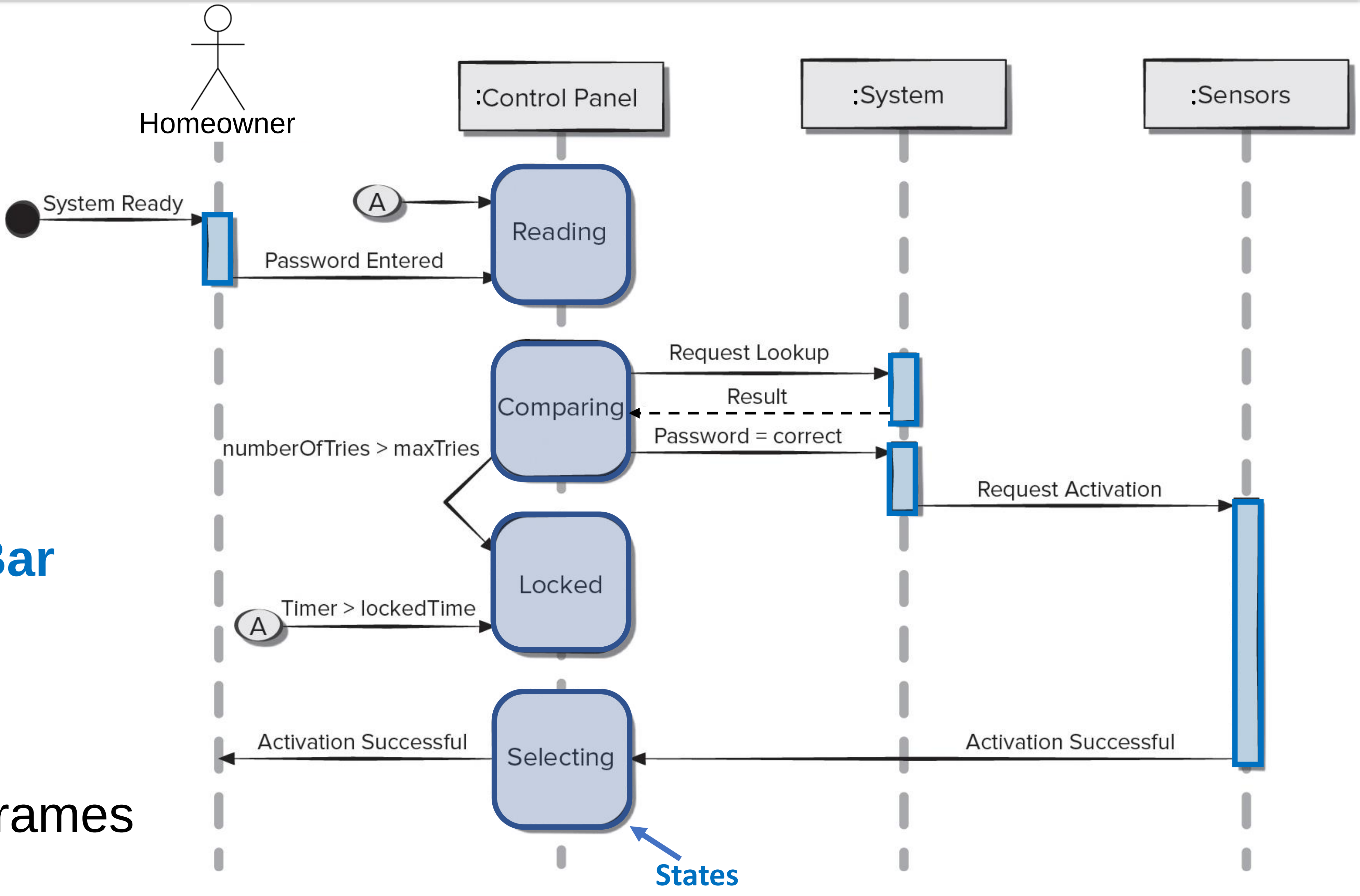
- **Objects**
- Lifeline
- Activation Bar
- Messages
- Interaction Frames



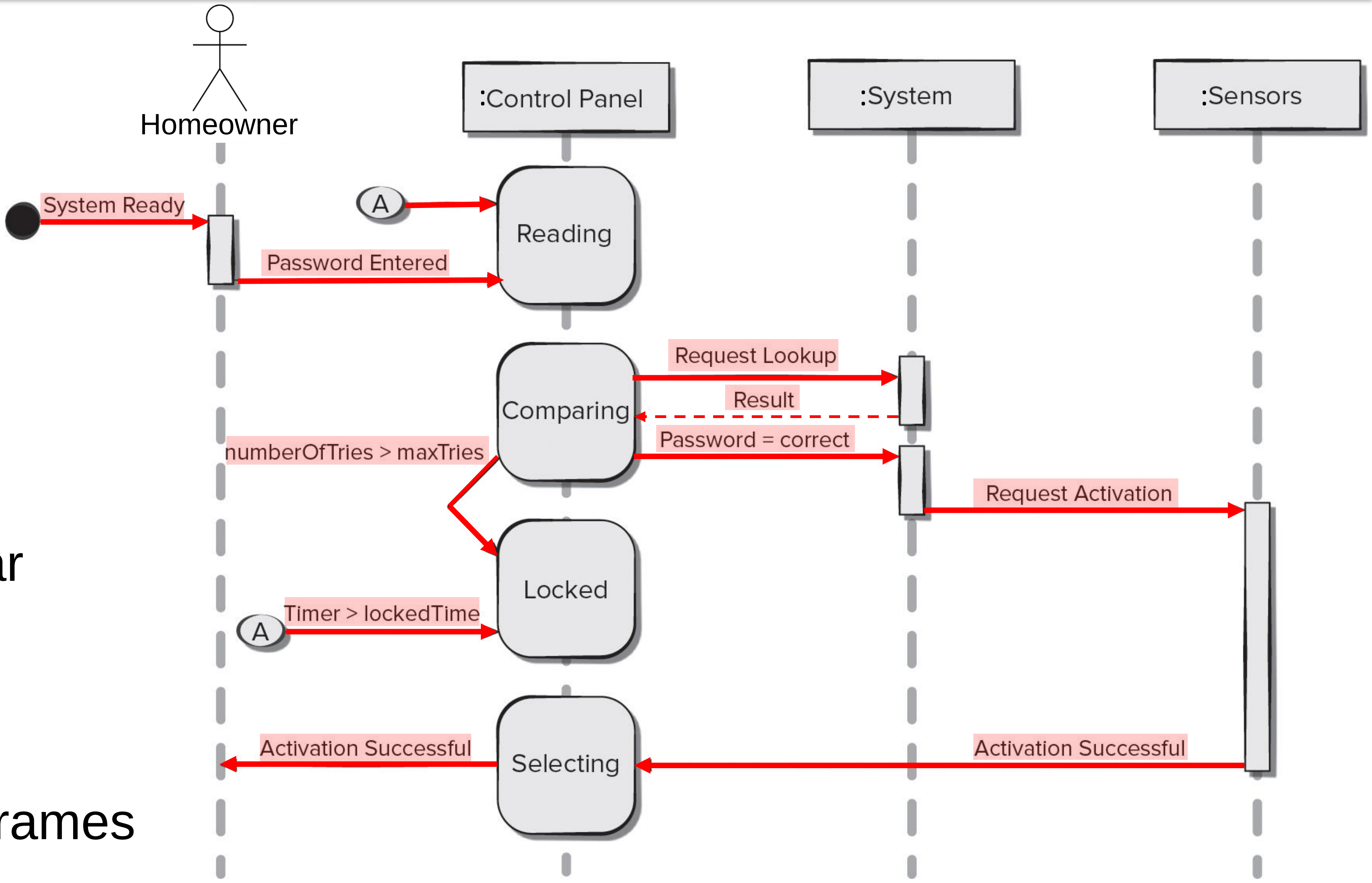
- Objects
- **Lifeline**
- Activation Bar
- Messages
- Interaction Frames



- Objects
- Lifeline
- **Activation Bar**
- Messages
- Interaction Frames

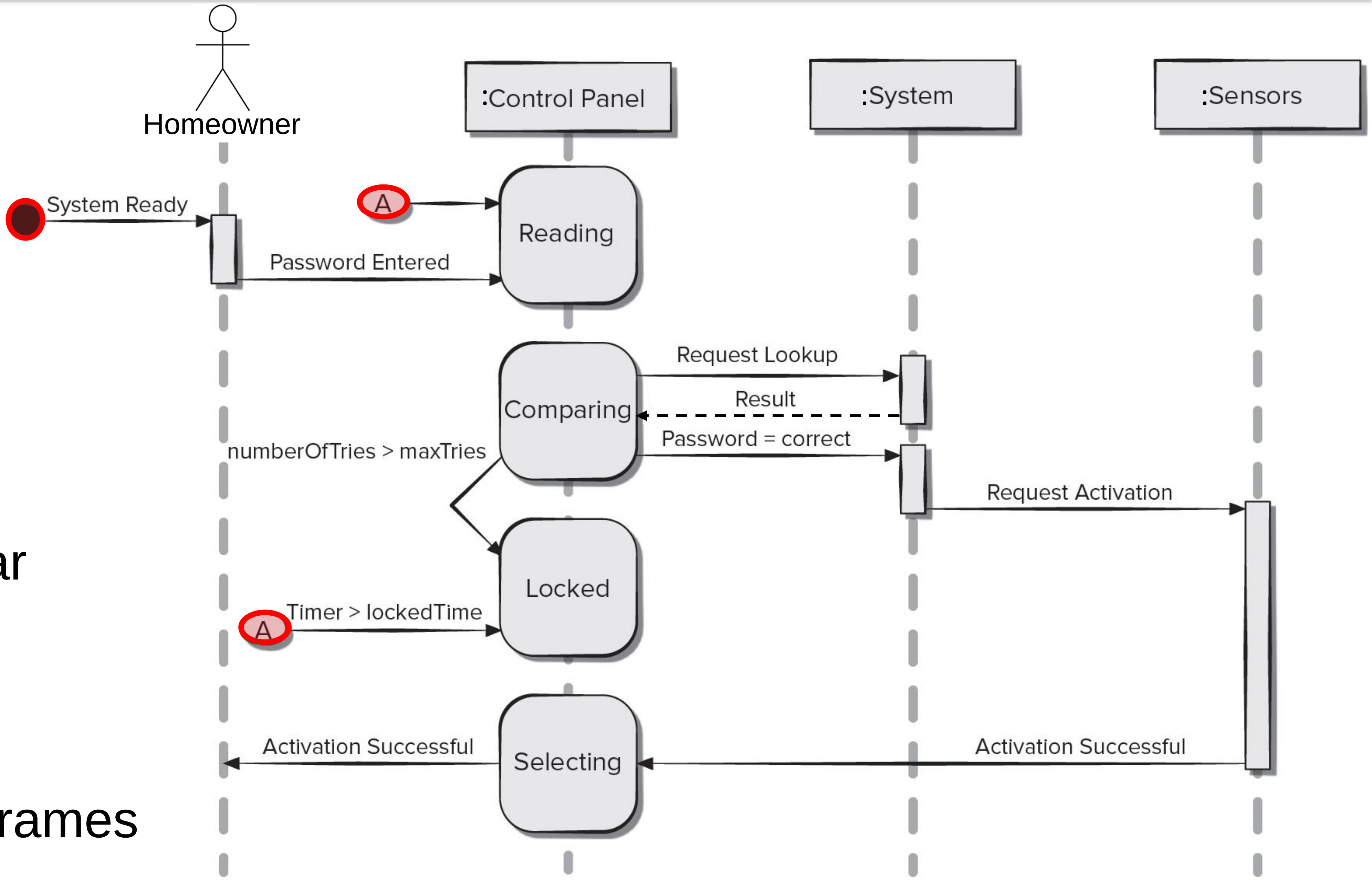


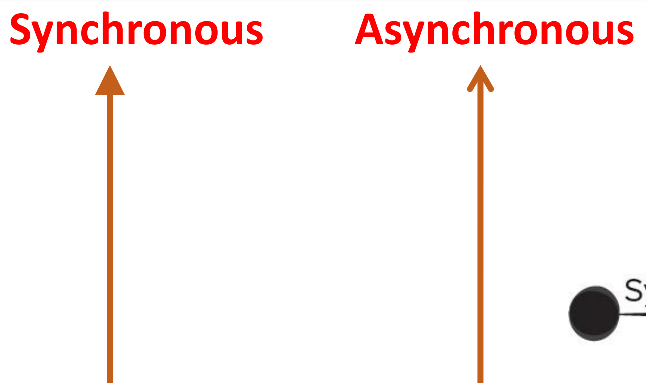
- Objects
- Lifeline
- Activation Bar
- **Messages**
- Interaction Frames



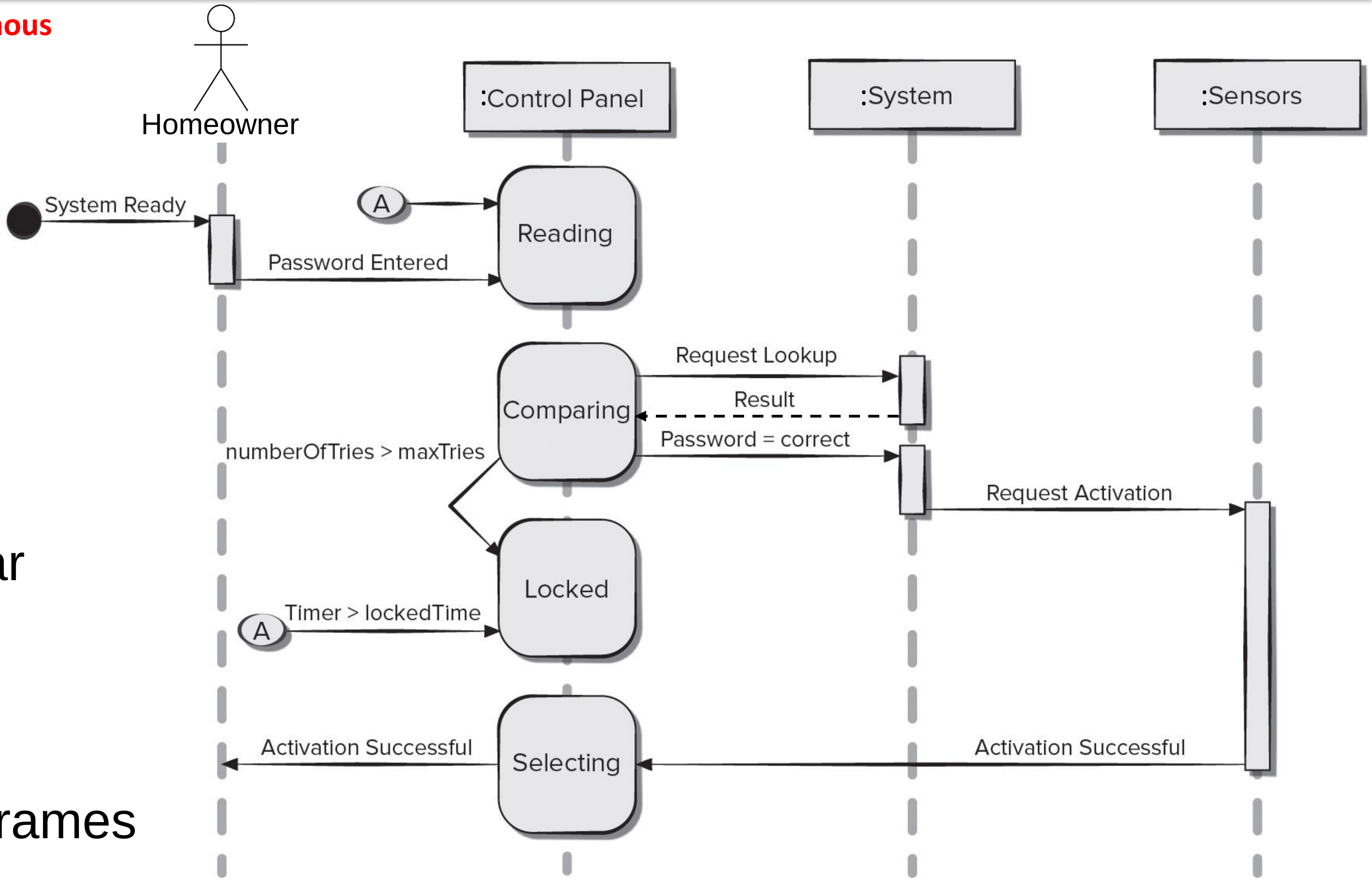
Found Message

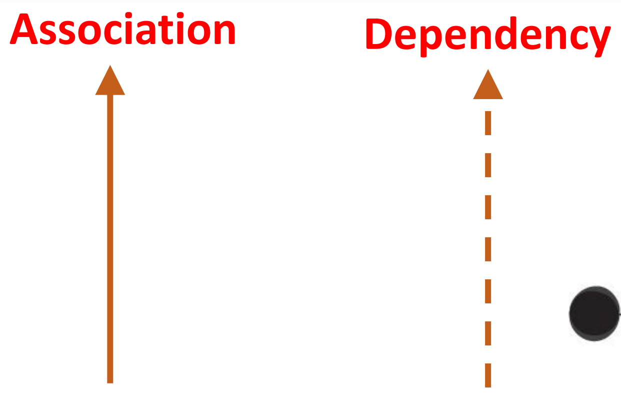
- Objects
- Lifeline
- Activation Bar
- **Messages**
- Interaction Frames



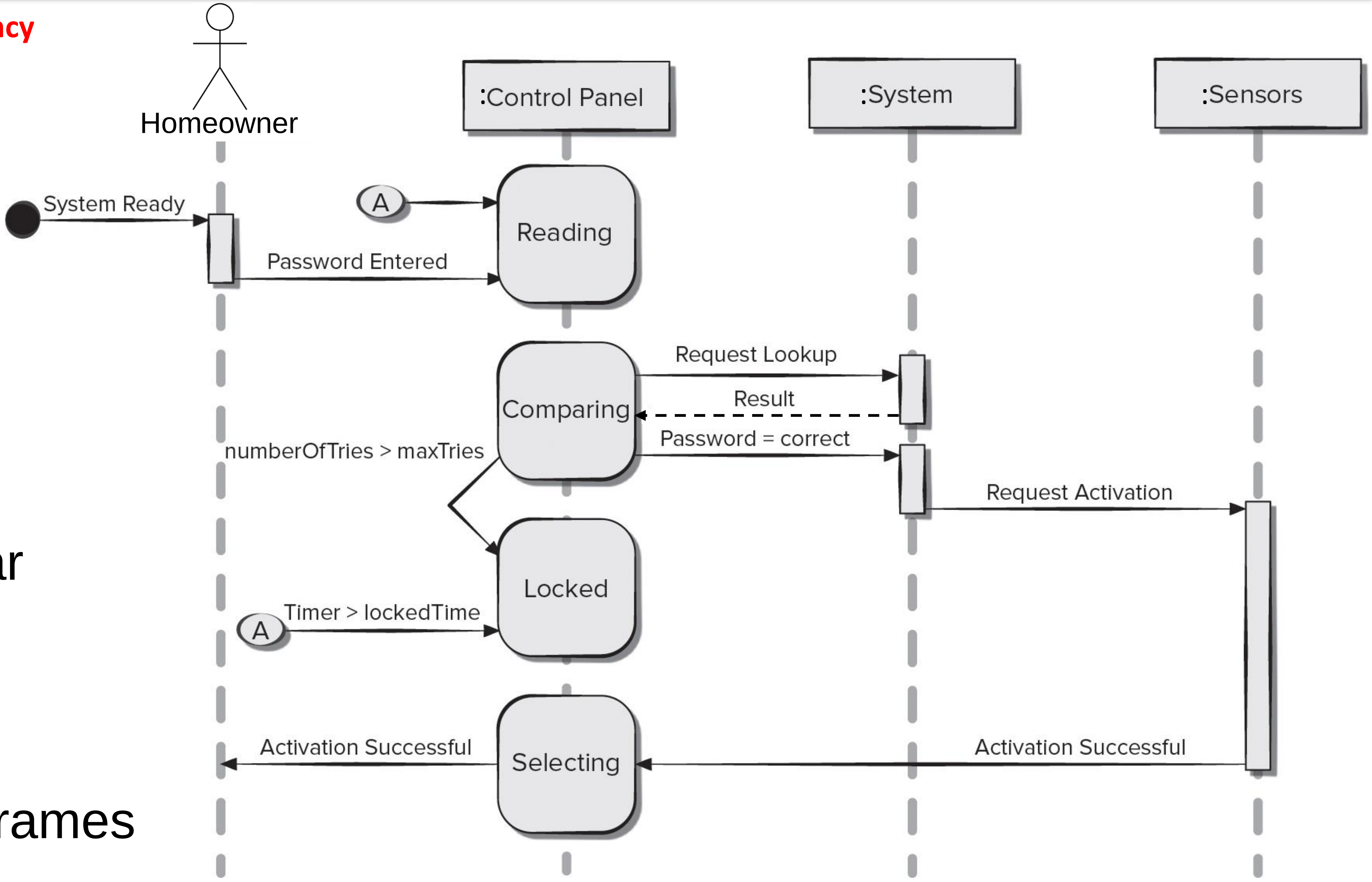


- Objects
- Lifeline
- Activation Bar
- **Messages**
- Interaction Frames

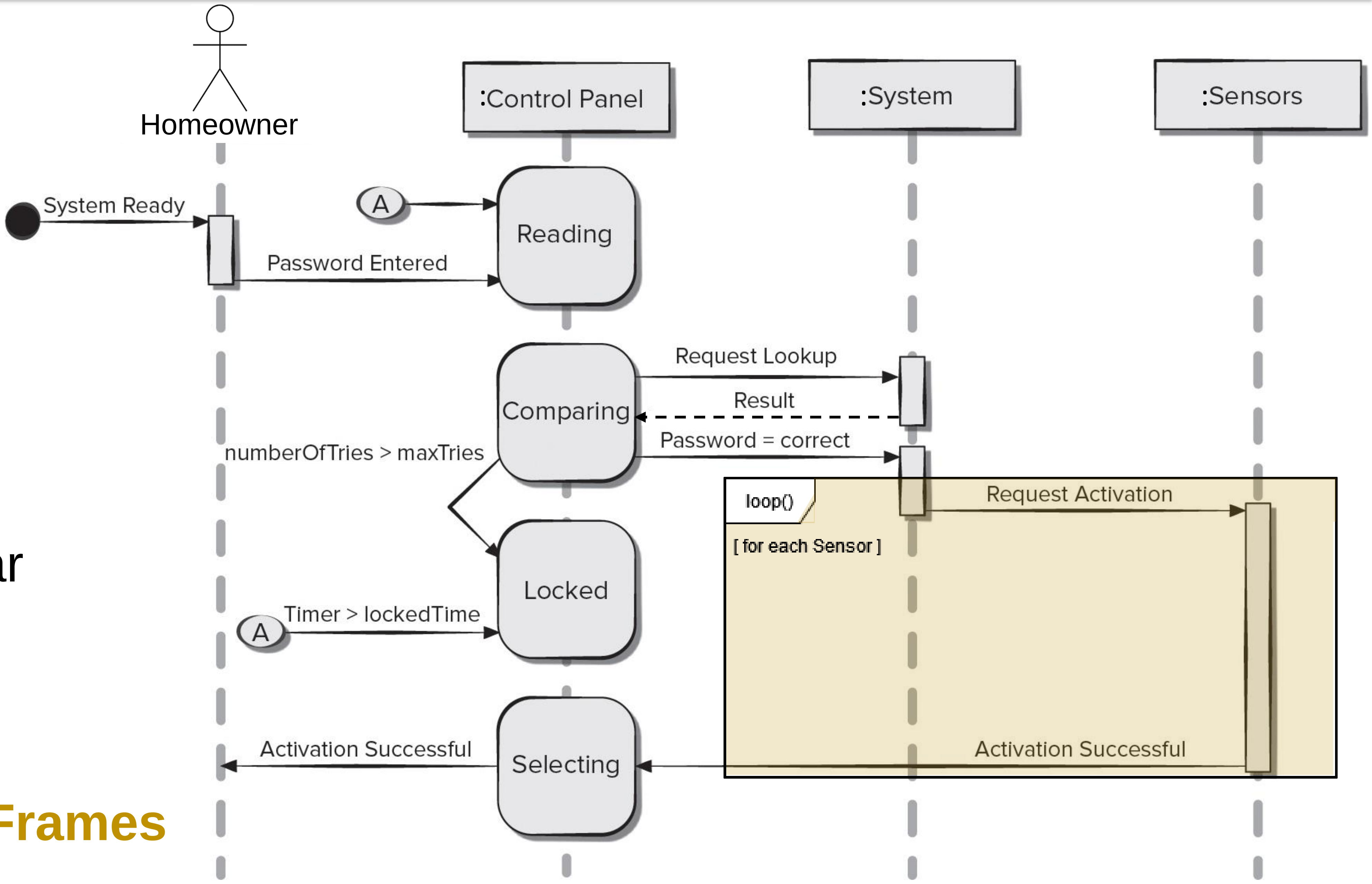




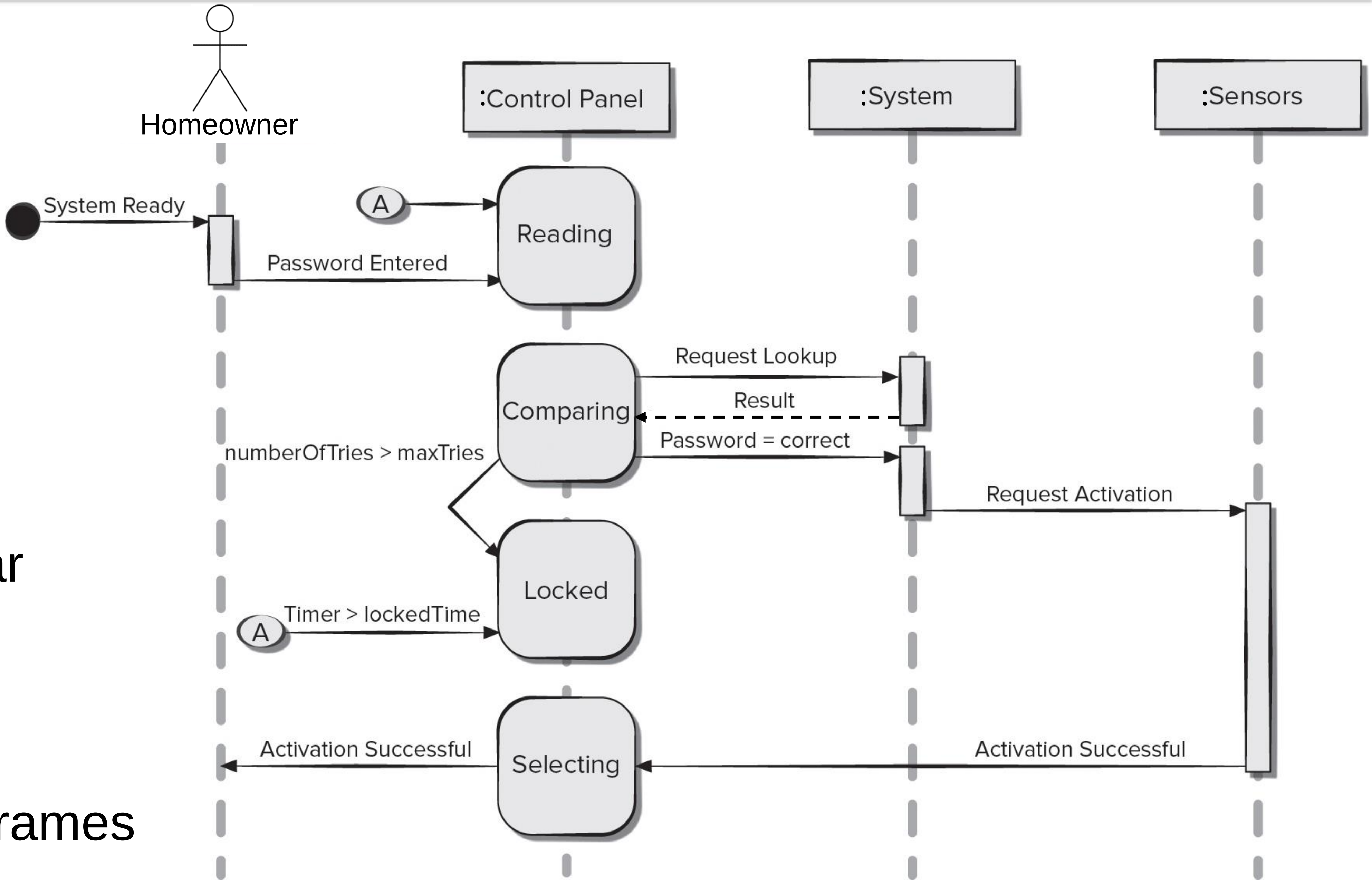
- Objects
- Lifeline
- Activation Bar
- **Messages**
- Interaction Frames



- Objects
- Lifeline
- Activation Bar
- Messages
- **Interaction Frames**

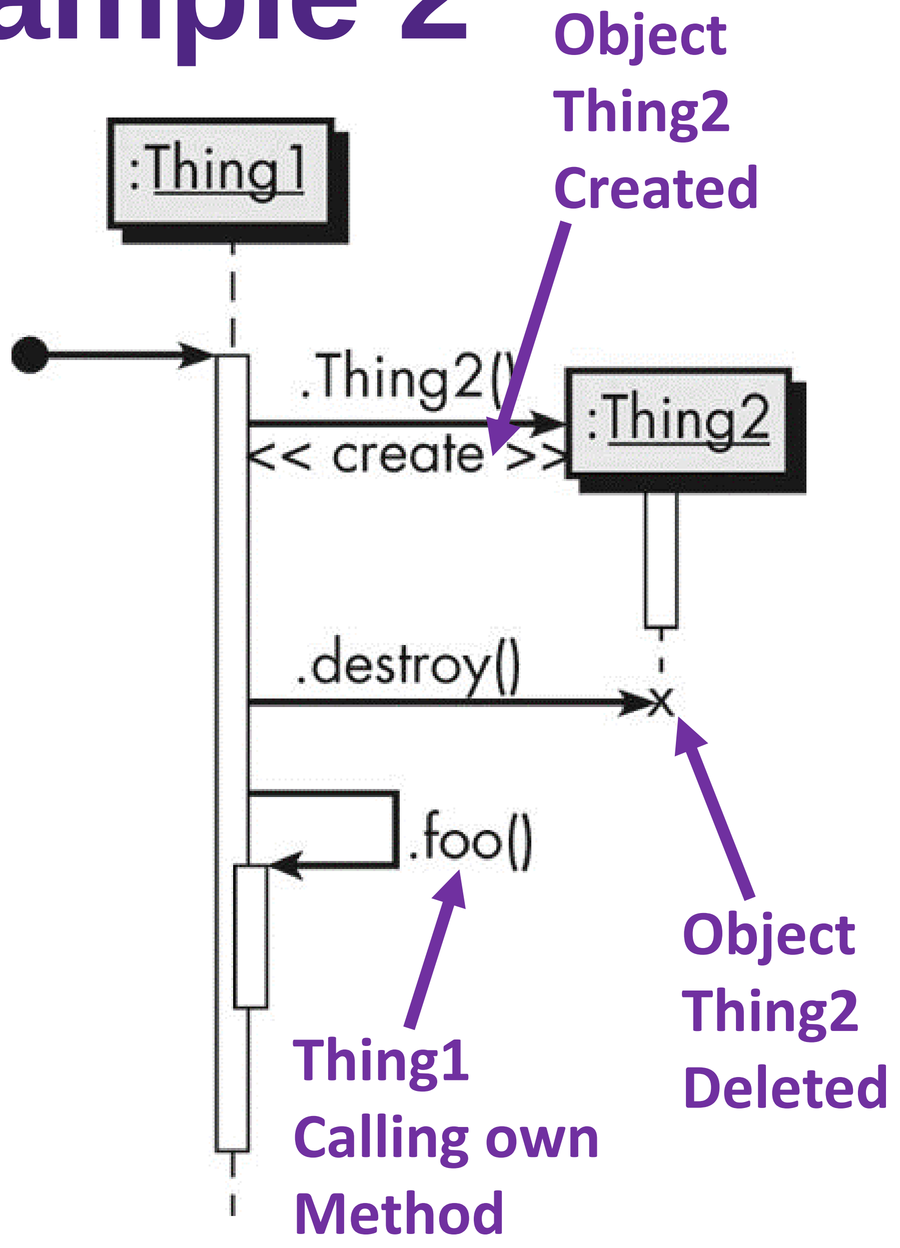


- Objects
- Lifeline
- Activation Bar
- Messages
- Interaction Frames

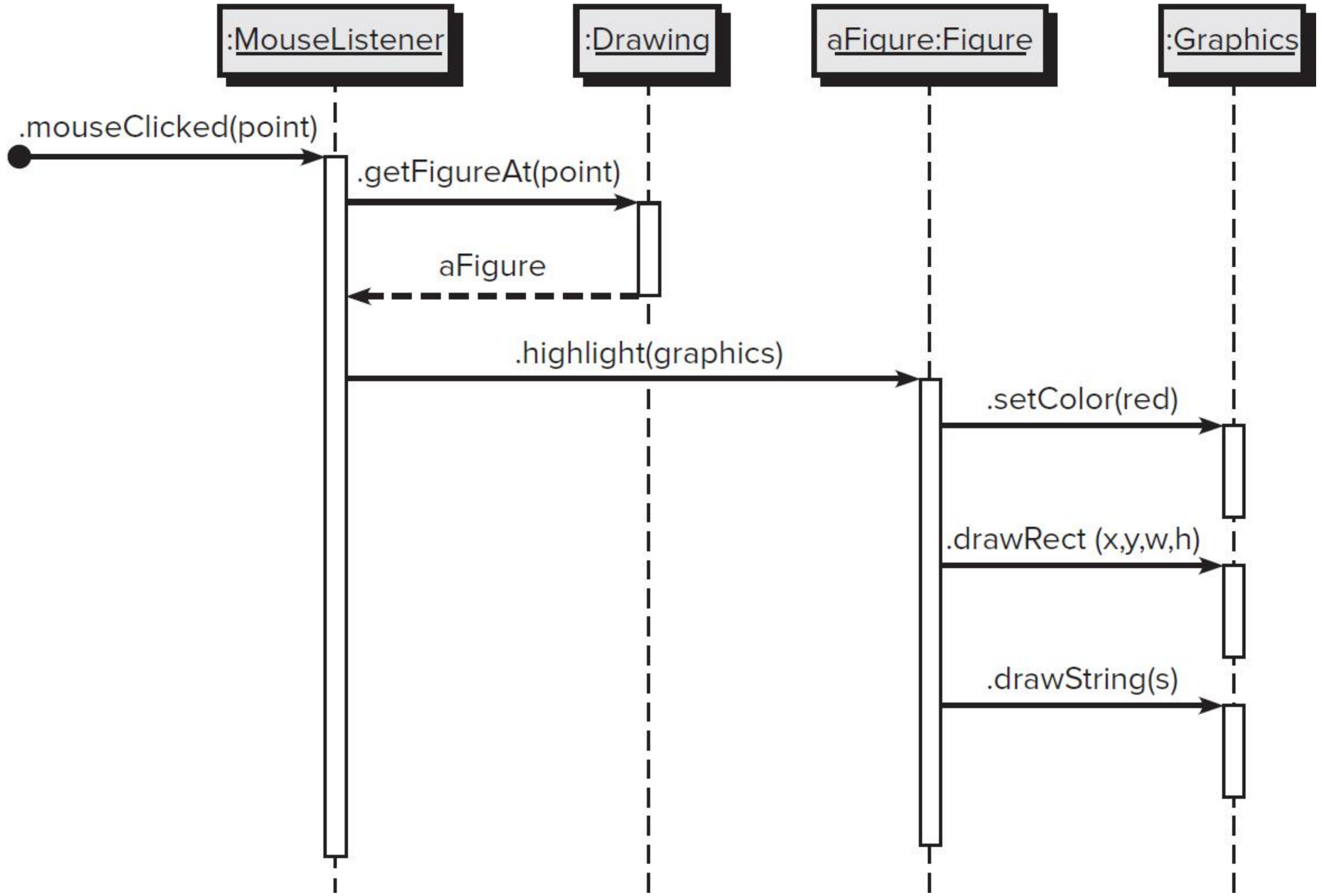


Sequence Diagram: Example 2

Sample sequence diagram showing the creation and deletion of objects, as well as message loops.



Sequence Diagram: Example 3



Sequence Diagram

When to Use Sequence Diagram:

- Used for describing the **behaviour of several objects within a single use case.**
- Good at showing **collaborations among objects**.
- **Not** good at showing precise definition of behaviour.
- Use **state diagrams** for behaviour of a single object, **activity diagrams** for concurrency or more complex behaviour, **CRC cards** for quick exploration of alternative interactions.

Behavioural Modeling in Summary

The general process is as follows:

1. Make a **list of the different states** of the software system *(how does the system behave?).*
2. Indicate **how the system makes a transition** from one state to another *(how does the system change state?).*
3. As part of this, **indicate all events and actions** that occur as a result.
4. Draw a **state diagram** and/or a **sequence diagram** using this information.
5. Draw an **activity diagram** (or optionally a **swimlane diagram**) to denote activity flow and expand on **use cases**.