Finite-State Machines (FSM)

Based on YAKINDU material from Itemis.

Software Systems (Computer & Embedded Systems Engineering)

Arjan Mooij

December 2022 (week 6)

An initiative of industry, academia and TNO

Dimensions for each model type

- **Motivation** When/where to apply the model type?
- **Concepts** What elements are used in the model type?
- **Notation** How to represent these concepts in a textual/graphical way?
- **Tool** How to create models using this notation?
- **Skill** How to determine which concepts to use for your models?

Objectives

At the end of the course, you should be able to:

- Explain the purpose of Finite-State Machines, including several application areas
- Explain the concepts and notations of Finite-State Machines
- Create basic Finite-State Machines to model software-intensive systems

Assessment:

- Modeling assignment using Finite-State Machines (in groups of 2 students)
- Reflection document on Model-Based Development (individual)

Agenda for Finite-State Machines
(Each week the Software Systems course has 2 lecture hours + 4 lab hours)

- **Week 6 Lecture**
  - 30 minutes Notation and simulation
  - 15 minutes Modeling skills
- **Week 6 Lab**
  - Notation and simulation
- **Week 7 Lecture**
  - 15 minutes Notation and simulation
  - 15 minutes Modeling skills
  - 15 minutes Application areas
- **Week 7 Lab**
  - Modeling skills

Motivation

Finite-State Machines are a very practical way to describe behavior:

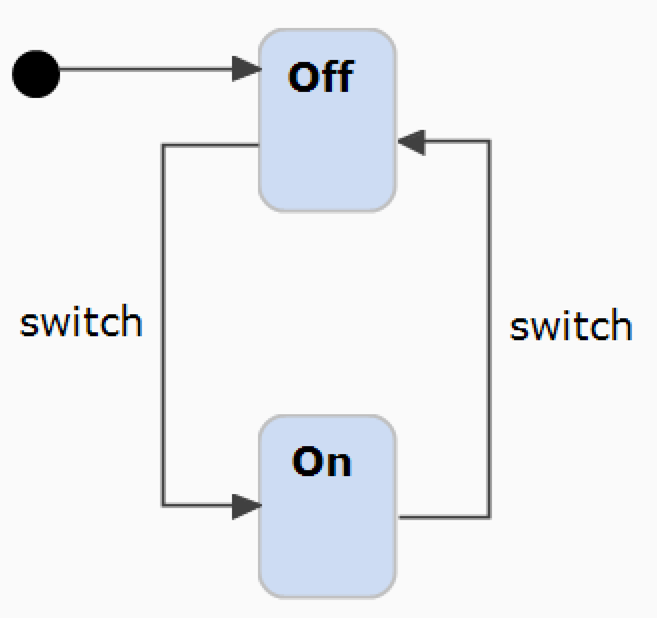
- User workflow
  - In which environment will the system be used?
  - E.g., passport renewal (submit application, background check, printing process, delivery, etc.)
- System behavior
  - What is the logic that the system should implement?
  - E.g., guarantee the safety of traffic lights
- Communication protocols on interfaces
  - How should concurrent components interact with each other?
  - E.g., only send messages (or call methods) in a specific order (e.g., after initialization)

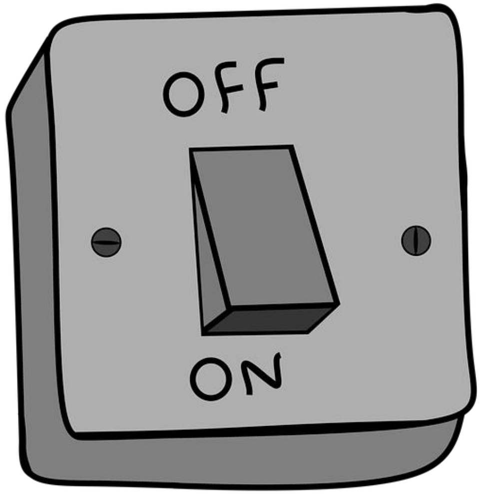
Note: Finite-State Machines are called State Machine Diagram in UML

Notation and simulation

Finite-State Machines (FSM)

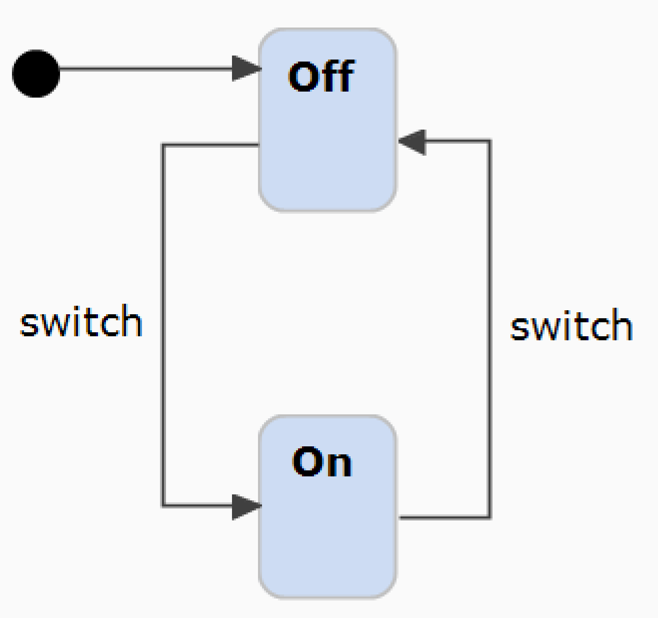
What would the elements in this Finite-State Machine mean?





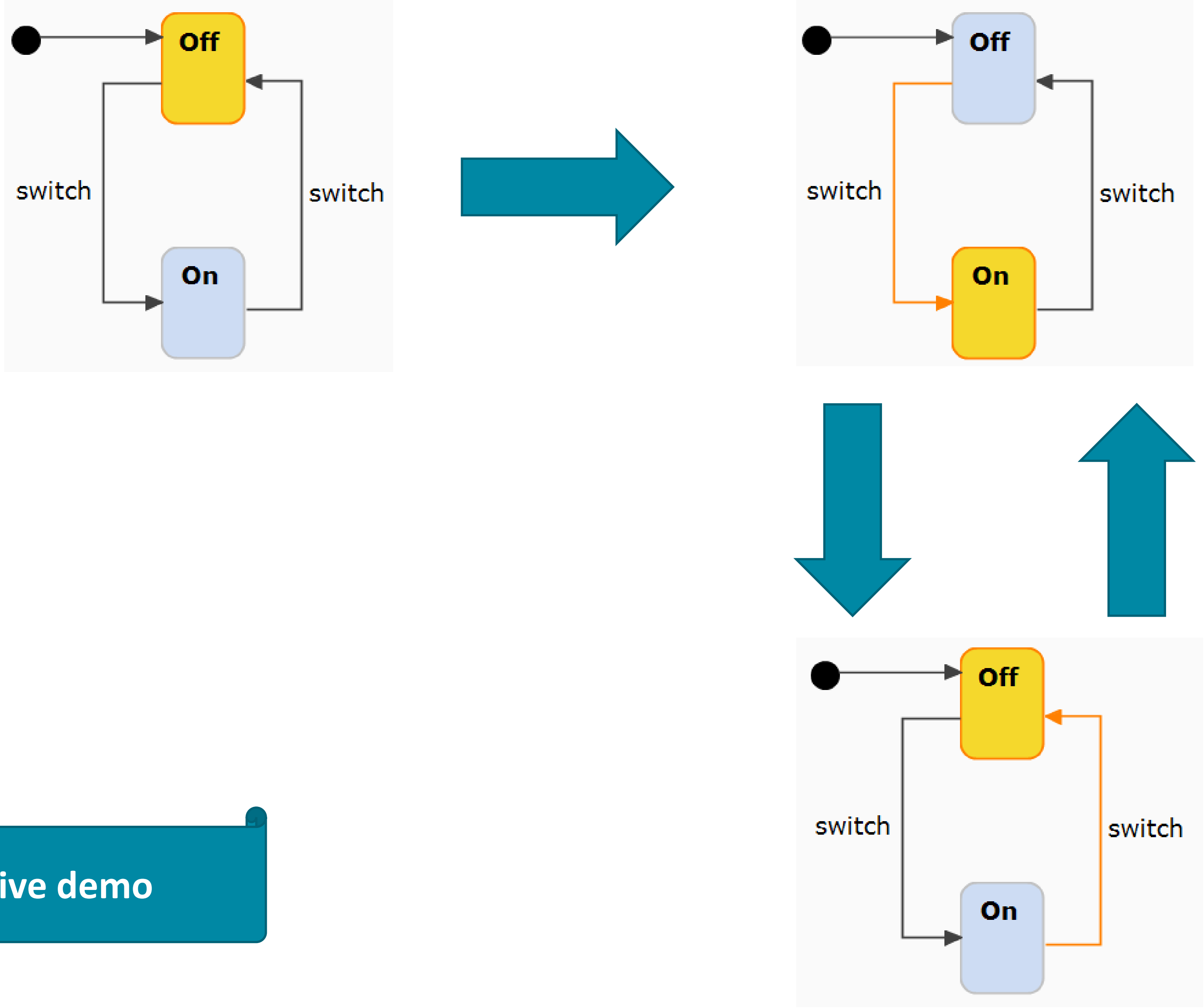
Think → Pair → Share

States, transitions and events



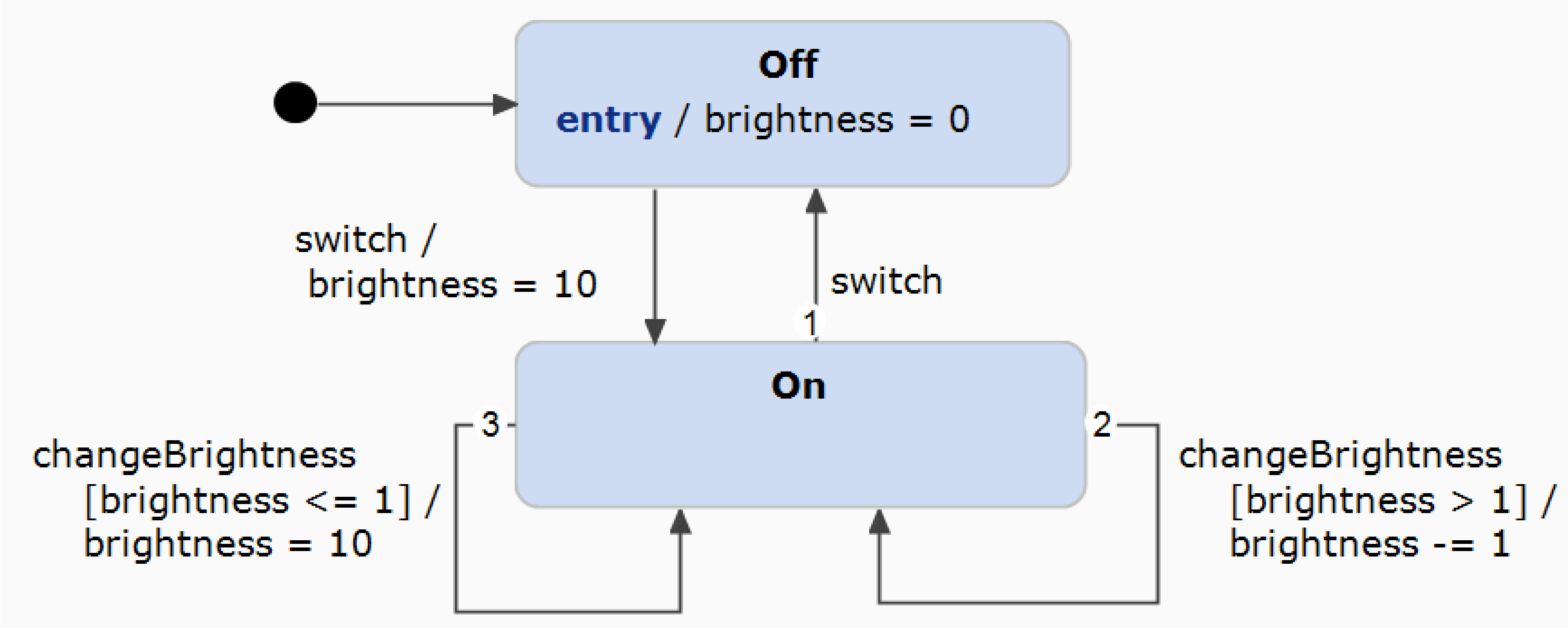
- **State**
  - Represents a possible mode of a system
    - Where the system is executing an activity or waits for an event.
  - Each state can be active or inactive
  - Visualization:
    - Normal state: Rounded rectangle (with a name)
    - Initial state: Filled black circle (without a name)
- **Transition**
  - Represents a possible state change
  - Visualization: Arrow from the source state to the target state (with an event trigger)
- **Event**
  - Represents a possible element on the interface of the system

Simulation



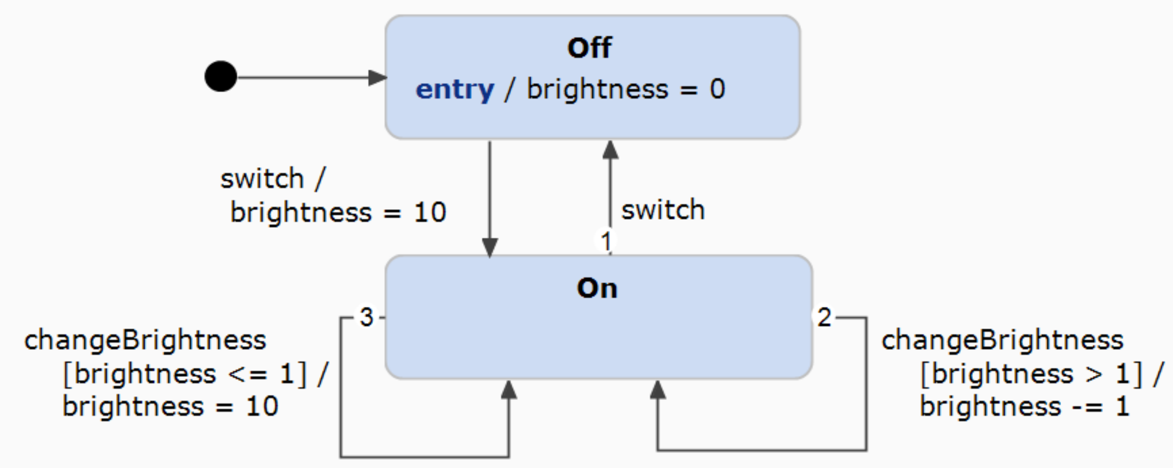
Live demo

What would the elements in this Finite-State Machine mean?



Think → Pair → Share

Variables, guards, and effects

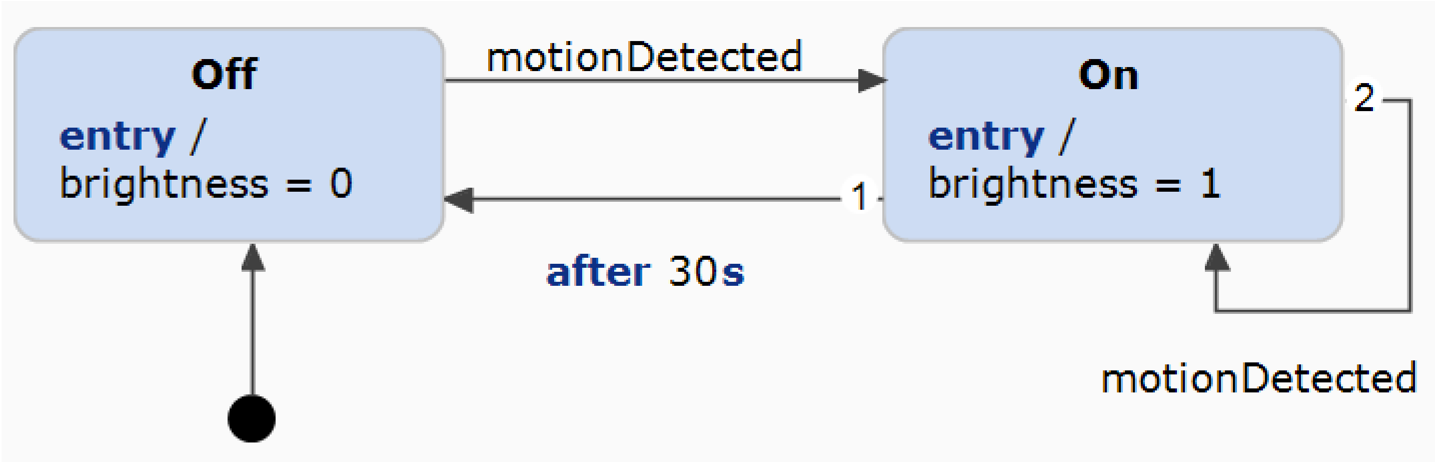


- **Variable**
  - Stores some data that can be changed
    - (Model may no longer be finite state)
- **Effects:**
  - Assignment to a variable
  - Raise an event (syntax: **raise** event)
  - Sequential composition(syntax: effect1 ; effect2)
- **Transition reaction:**
  - Executed when the transition is taken
  - Syntax: trigger [guard] / effect
    - Guard is a condition that enables the transition
- **State reaction:**
  - Syntax:
    - **entry** / effect — Executed when the state is entered
    - **exit** / effect — Executed when the state is exited
    - event / effect — Executed when no outgoing transition can be taken
- **Priorities on the outgoing transitions of a state**

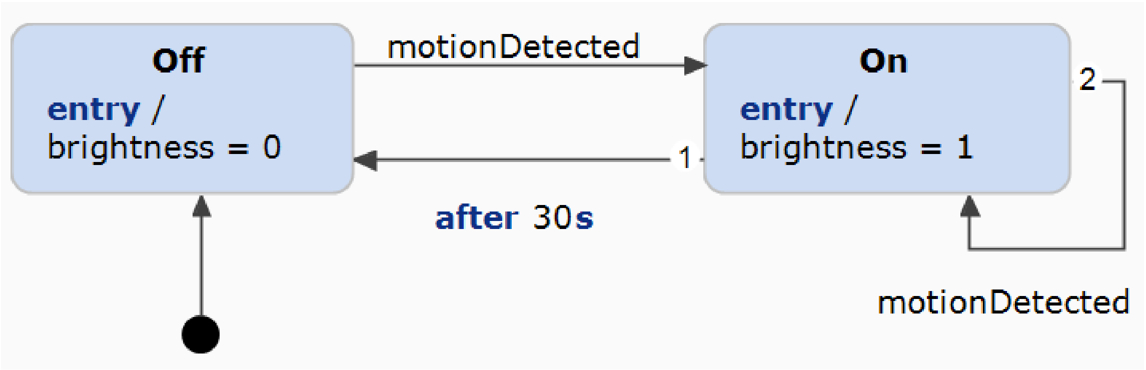
Simulation

Live demo

What would the elements in this Finite-State Machine mean?



Triggers



- **Single event trigger**
  - Trigger when the event is raised
  - Syntax: ev1
- **Multiple event trigger**
  - Trigger when one of the event is raised
  - Syntax: ev1, ev2
- **Time trigger**
  - Trigger after given amount of time
  - Syntax: after 30s

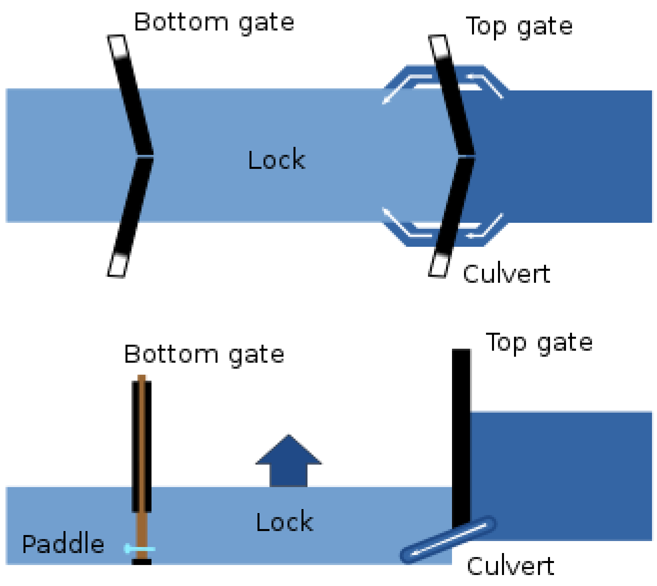
Modeling skills

Finite-State Machines (FSM)

Interactive creation of an FSM

Ship lock:

- Two gates: bottom and top
- Two valves: bottom (paddle) and top (culvert)



https://en.wikipedia.org/wiki/Lock_(water_navigation)

Why would it be interesting to model this?

Safety constraints:

- At most one gate or valve open at a time
- Gates can only be opened when the water levels match

Let's model the behavior of the lock in terms of the gates and valves!

Interactive creation of an FSM

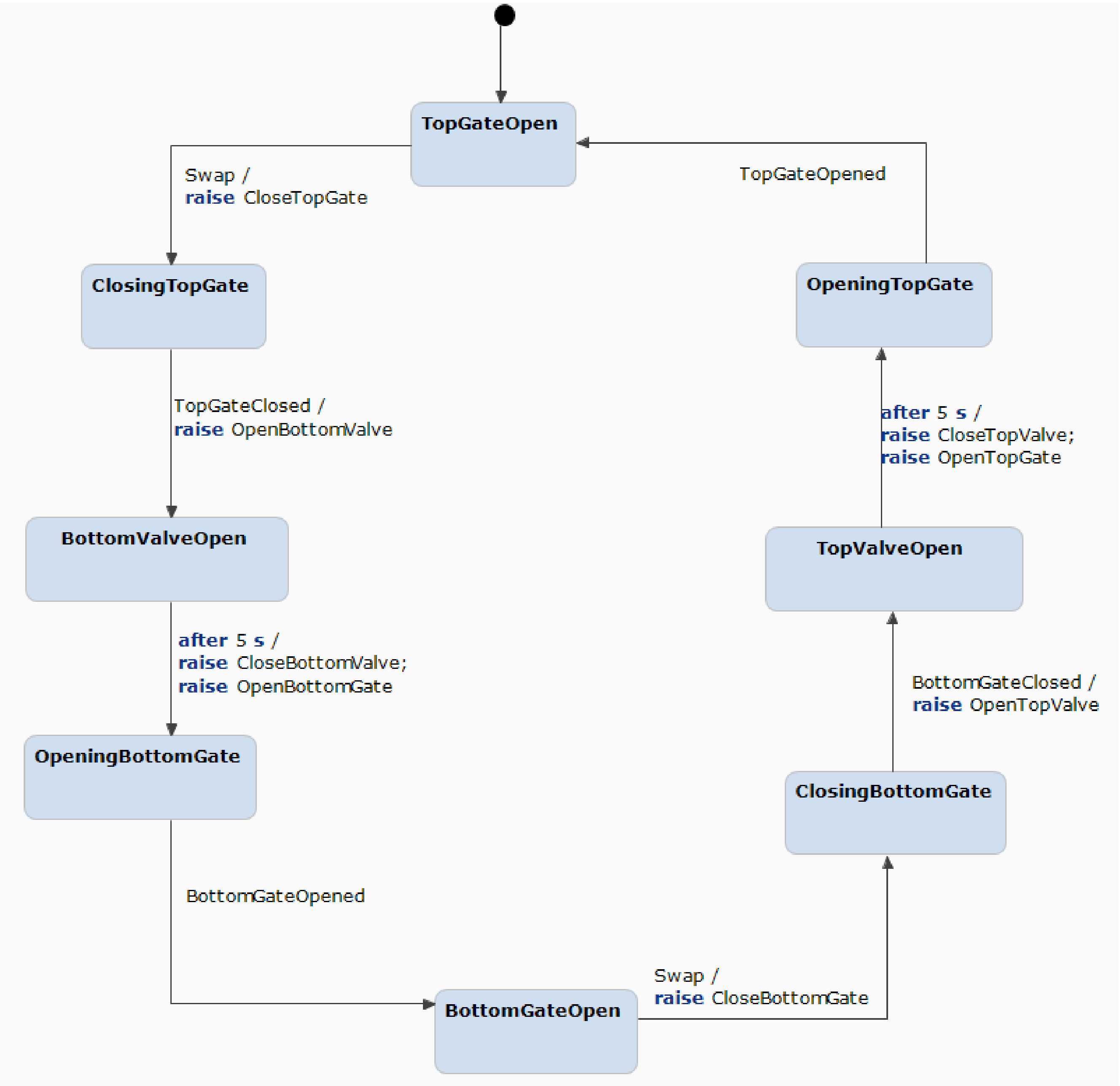
Goal of the model: **safe operating procedure of the gates and valves**

Events:

- **Depend on the gate/valve interface** → **perhaps first model their interface behavior!**
    - For the valves we rely on time
    - For the gates we rely on sensors that confirm certain positions
- **User interactions:**
    - Start the next swap
    - (Possible extension: interrupt a swap?)

Two kinds of state:

- **Stable system situation: e.g., gate open**
- **Instable stable situation: e.g., valve open**


TopGateOpen
Swap /
raise CloseTopGate
TopGateOpened
ClosingTopGate
OpeningTopGate
TopGateClosed /
raise OpenBottomValve
after 5 s /
raise CloseTopValve;
raise OpenTopGate
BottomValveOpen
TopValveOpen
after 5 s /
raise CloseBottomValve;
raise OpenBottomGate
BottomGateClosed /
raise OpenTopValve
OpeningBottomGate
ClosingBottomGate
BottomGateOpened
Swap /
raise CloseBottomGate
BottomGateOpen

Closing remarks

Finite-State Machines (FSM)

Closing remarks

- **Lab session work on “Notation and simulation” <u>before</u> the next lecture**
  - Download and install the YAKINDU StateChart tools
  - Eclipse → Help → Help Contents → YAKINDU Statechart Tools documentation
    - Tutorials → Comprehensive tutorial
      - **NOTE: Section 2 only!** **So stop before “3. Statechart concepts tutorial”**

- **Optional (if you are interested in test-driven development):**
  - Eclipse → Help → Help Contents → YAKINDU Statechart Tools documentation
    - User guide → Testing state machines
      - **NOTE: Up to section 1.3 only!** **So stop before “1.4 The SCTUnit language”**

- **Think about the relation between FSM and UML use case diagram**