
TNO ESI
Powered by industry
and academia

STATECHARTS:
AN ENHANCEMENT OF FINITE-STATE MACHINES TO MODEL COMPLEX SYSTEMS

Software Systems (Computer & Embedded System Engineering)

Rosilde Corvino

07-01-2025

TU Delft Delft University of Technology

TNO ESI Powered by industry and academia

AGENDA OF THE COURSE

	Week 6 (17-12)	Week 6 (19-12)	Week 7 (7-1)	Week 7 (9-1)	Week 8 (14-1)	Week 8 (16 -1)	Week 9 (21-1)	Week 10
Lectures on Tuesdays (2 hours)	Introduction		StateChart 1		DSL 1			
	UML 1		StateChart 2		DSL 2			
Labs on Thursdays (4 hours)		1 UML Lect 3 labs		StateChart		1 DSL lec + conclusion 3 DSL labs		
Assignment due on Friday				UML		Statechart		DSL + Reflection

QUIZ GAMES

- Identified by the banner: **Think/Write → Pair → Share**

- Instructions:
  1. Divide into teams of two students
  2. Discuss the solution to the quiz together
  3. Volunteer or be asked to share
  4. Points will be awarded for participation:
     1. Every time you share during a game, you earn 0.3 points
     2. You can earn up to a maximum of 1 additional point on the final note for this part of the course
     3. Do not forget to write your name on the winners' sheet after the lecture

OBJECTIVES

- At the end of the course, you should be able to:
  - Explain the purpose of State Charts, including several application areas
  - Explain the concepts and notations of State Charts
  - Create basic State Charts to model software systems

- Assessment:
  - Modeling assignment using State Charts (in groups of 2 students)
  - (Talk of State Charts in the) Reflection document (individual)

WHAT DO YOU ALREADY KNOW ABOUT FSM?

- **What is an FSM?**
  - It is a mathematical model of computation.
  - It is an abstract machine that can be in exactly one of a finite number of states at any given time.
- **Can you give an example of the use of an FSM?**

- **There can be two types of deterministic FSM (Mealy and Moore). What is the main difference between them?**

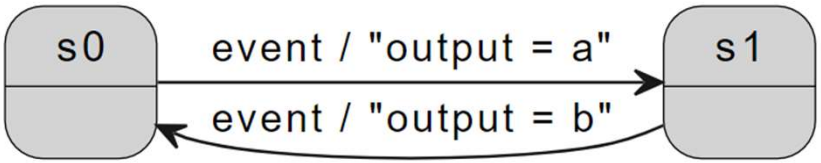
- **What are possible problems with FSM representations of complex systems?**

- Duration: 3 minutes discussion in pair

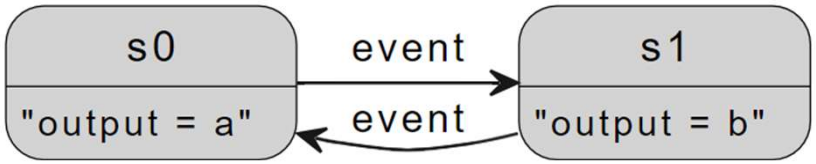
Think/Write → Pair → Share

MEALY VERSUS MOORE

- **Mealy**



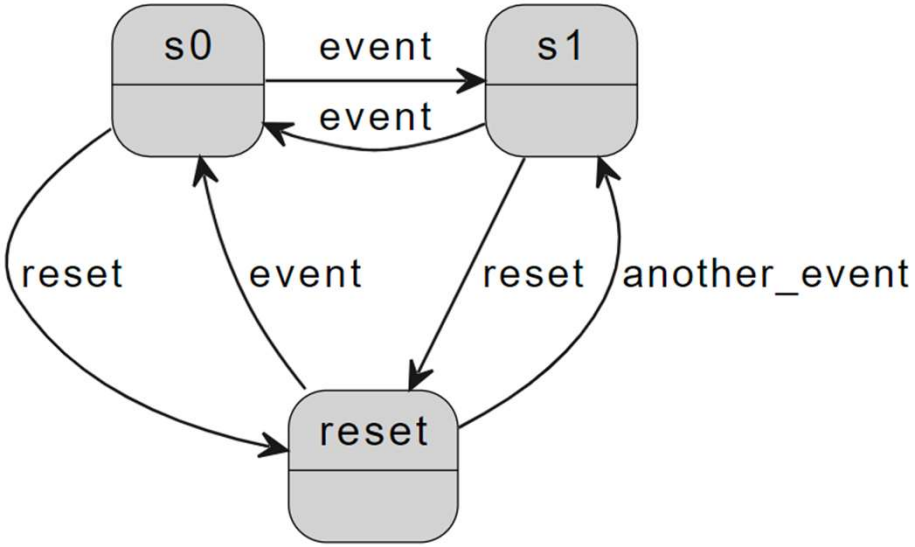
- **Moore**

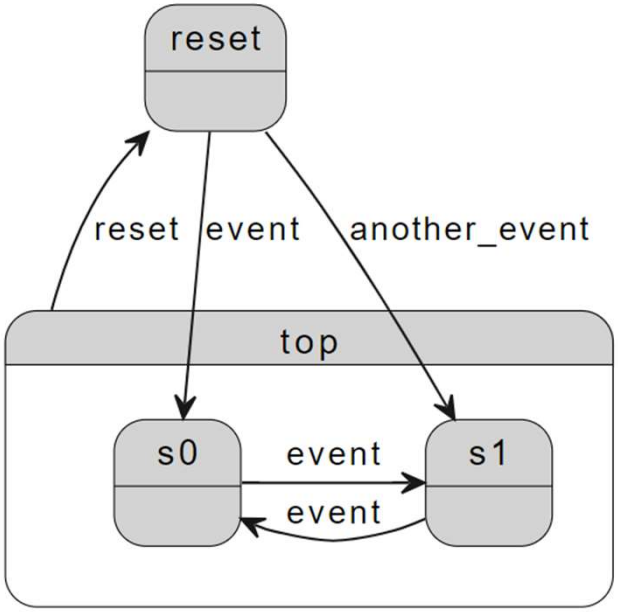


	Mealy Machine	Moore Machine
Output Glitches	More prone to glitches in output due to immediate reaction to input changes.	Less prone to glitches as outputs change only on state transitions.
State Complexity	Generally, it requires fewer states for the same functionality.	May require more states to achieve the same functionality.
Design Complexity	Can be more complex to design and understand due to the input dependency.	Simpler to design and understand as outputs are state-dependent only.

POSSIBLE PROBLEMS WITH FSM REPRESENTATIONS OF COMPLEX SYSTEMS

- Explosion of the number of states and transitions





"Statecharts: A Visual Formalism for Complex Systems" by David Harel. 1986

EXAMPLE OF USE OF FINITE-STATE MACHINES (AND STATE CHARTS)

- FSM are a very practical way to describe behaviour:
  - User workflow
    - In which environment will the system be used?
    - E.g., passport renewal (submit an 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)

AGENDA FOR FINITE-STATE MACHINES

(EACH WEEK THE SOFTWARE SYSTEMS COURSE HAS 2 LECTURE HOURS + 4 LAB HOURS)

- Week 7 Lecture
  - 5 minutes Introduction
  - 15 minutes Basic Notation and simulation
  - 25 minutes Modeling skills
  - 15 minutes Break
  - 20 minutes Advanced notation and simulation
  - 20 minutes Modeling skills
  - 5 minutes Conclusions

- Week 7 Lab
  - Notation and simulation
  - Modeling skills

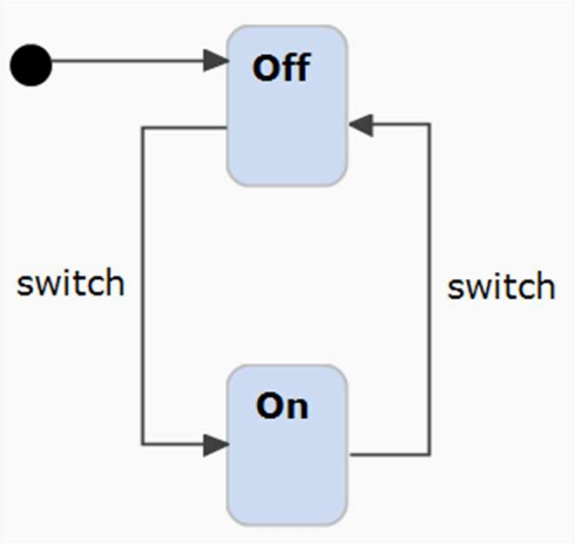
BASIC NOTATION AND SIMULATION

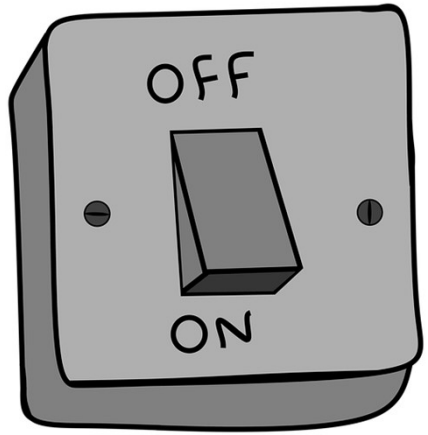
- State Charts

An initiative of industry, academia and TNO

ESI

WHAT DO THE ELEMENTS IN THIS STATE CHART MEAN?



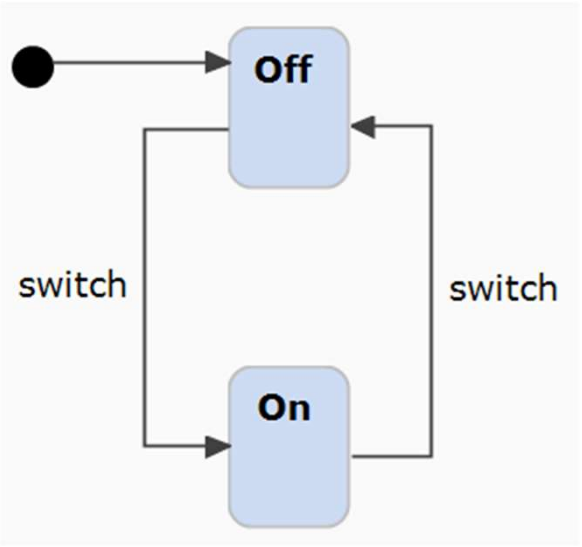


Duration: 1 minute discussion in pair

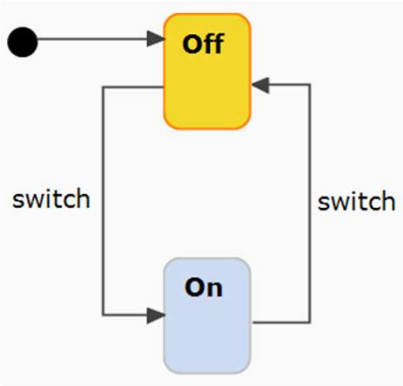
Think → Pair → Share

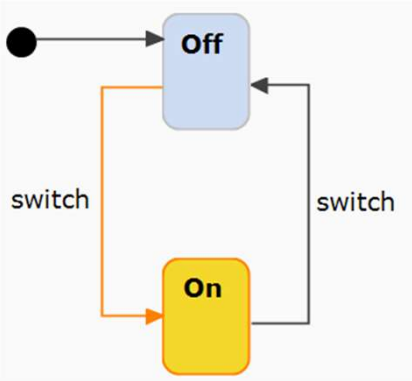
STATES, TRANSITIONS AND EVENTS

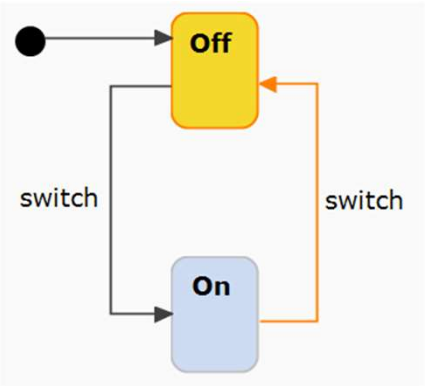
- **State**
  - Represents a possible mode of a system where the system executes an activity or waits for an event.
  - Each state can be active or inactive
  - Type:
    - Normal state: Rounded rectangle (with a name)
    - Initial state: Indicated by an entry point (Filled black circle)
- **Transition**
  - Represents a possible state change triggered by an event
  - Visualization: Arrow from the source state to the target state (with an event trigger)
- **Event**
  - Triggers a state change
  - An input event represents a possible element on the interface of the system



SIMULATION

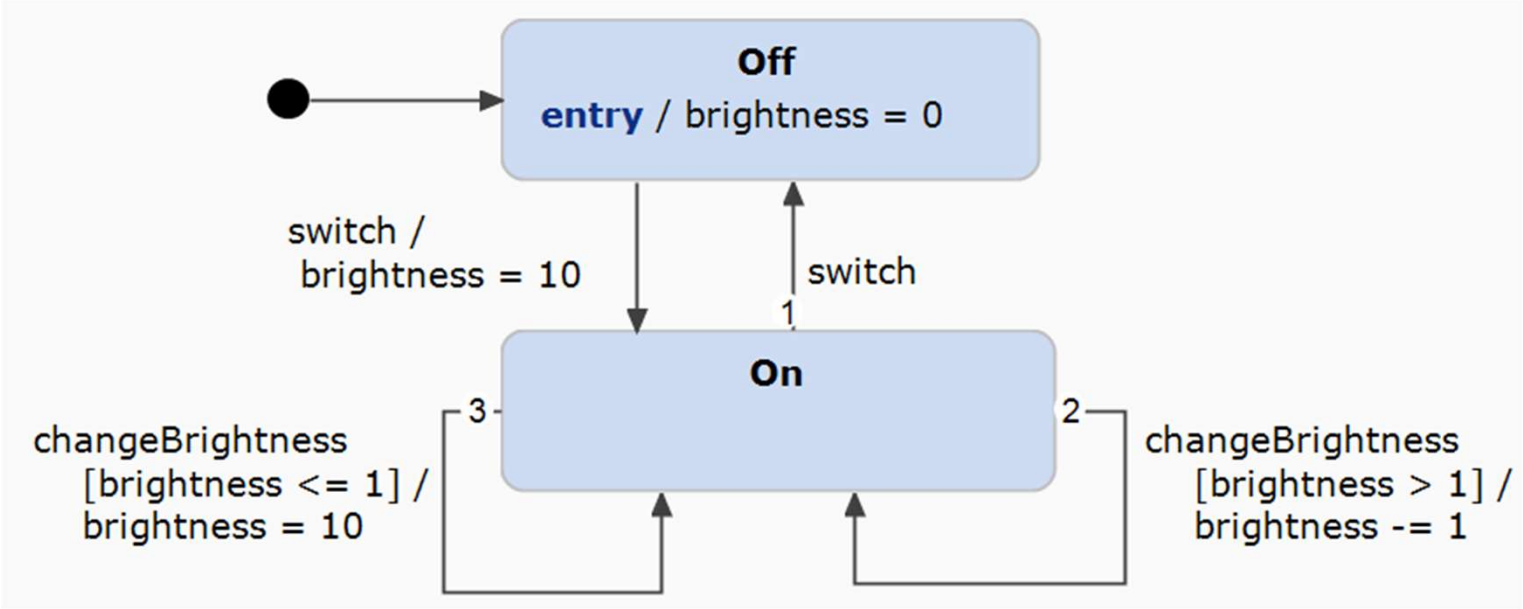






Live demo

WHAT DO THE ELEMENTS IN THIS STATE CHART MEAN?

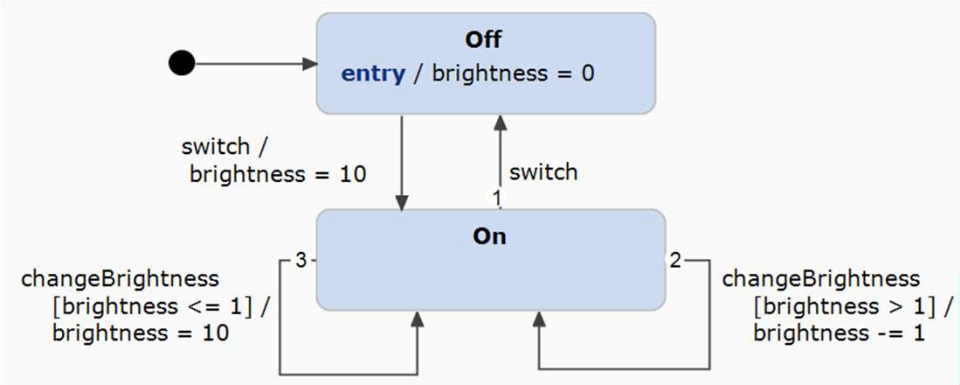


Duration: 3 minutes discussion in pair

Think → Pair → Share

VARIABLES, GUARDS, AND EFFECTS

- Variable
  - Stores some data that can be changed
- 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
    - **every 2**s / effect Executed every 2 seconds
    - **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



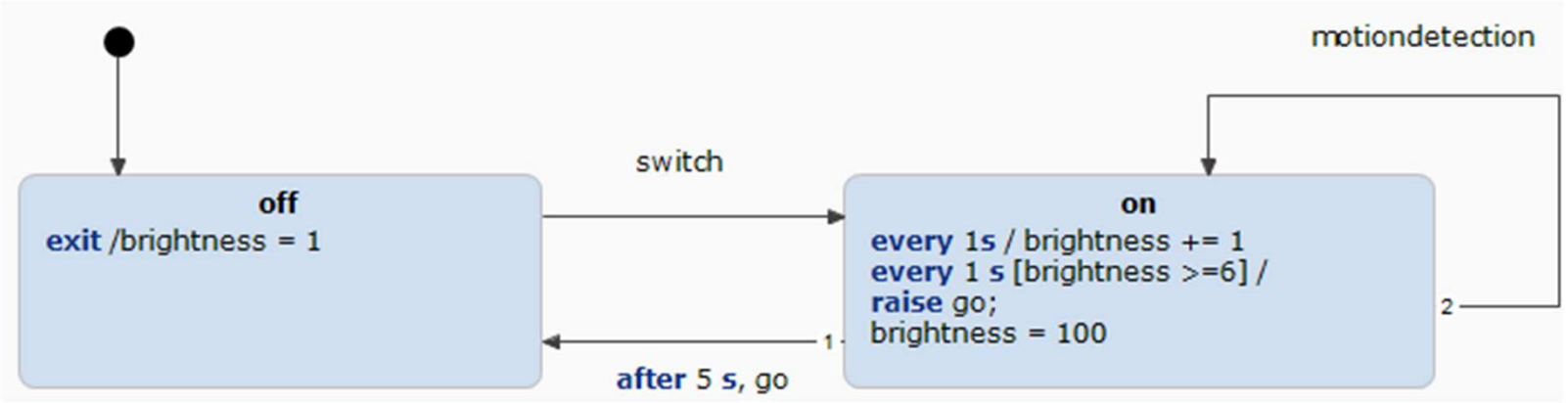
SIMULATION

Live demo

WHAT WOULD THE ELEMENTS IN THIS FINITE-STATE MACHINE MEAN?

Statechart

interface:
  in event motiondetection
  var brightness: integer
internal:
  event go

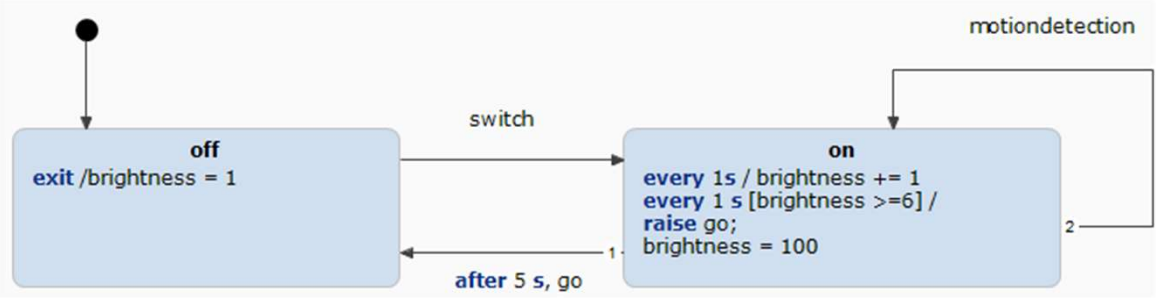


Think → Pair → Share

TRIGGERS

- Single event trigger (e.g., switch)
  - Trigger when the event is raised and can have a condition
  - Syntax: ev1 [condition]
- Multiple event triggers (e.g. after 5s, go)
  - Trigger when one of the two events is raised
  - Syntax: ev1, ev2
- Time trigger (e.g., after 5s)
  - Trigger after a given amount of time
  - Syntax: after 30s

```
Statechart
interface:
  in event switch
  in event motiondetection
  var brightness: integer
internal:
  event go
```



SIMULATION

Live demo

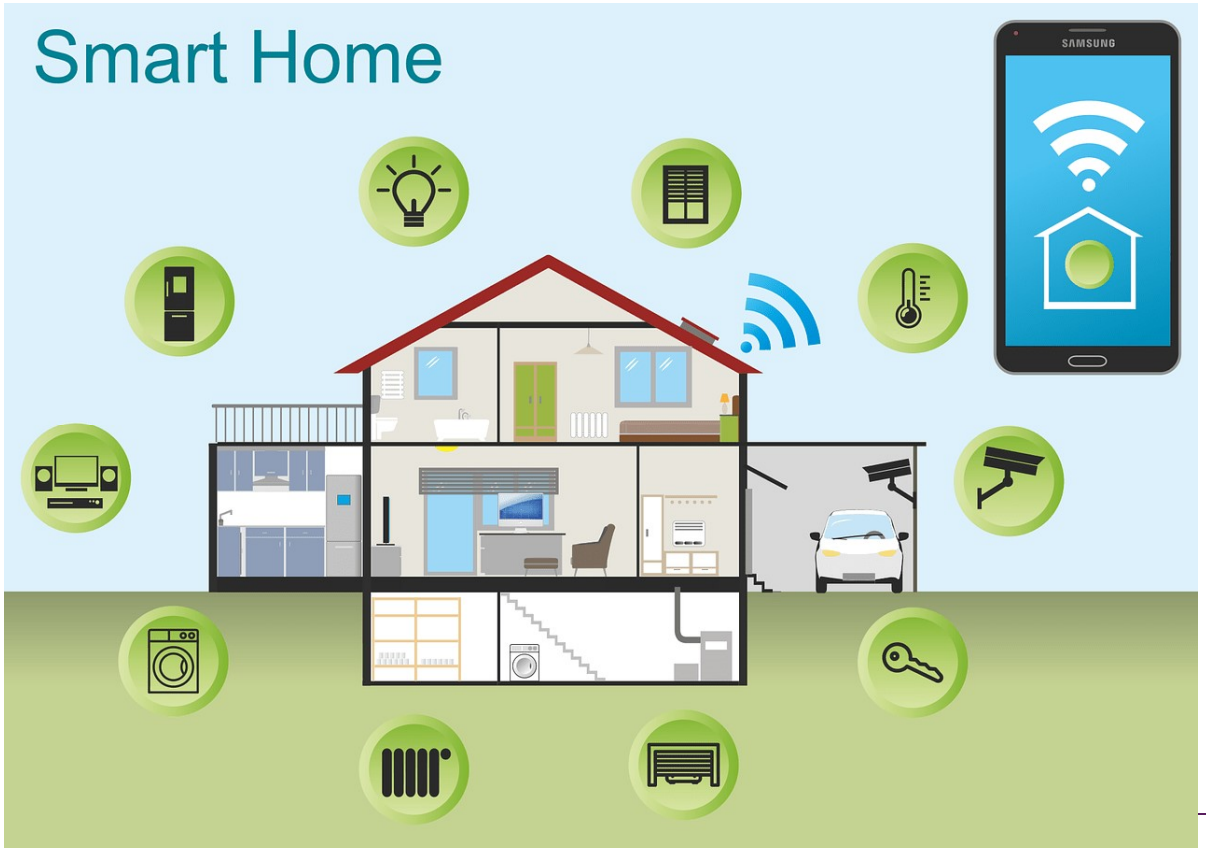
MODELING SKILLS

- State Charts

An initiative of industry, academia and TNO

ESI

CASE STUDY: HOME AUTOMATION – SECURITY SYSTEM



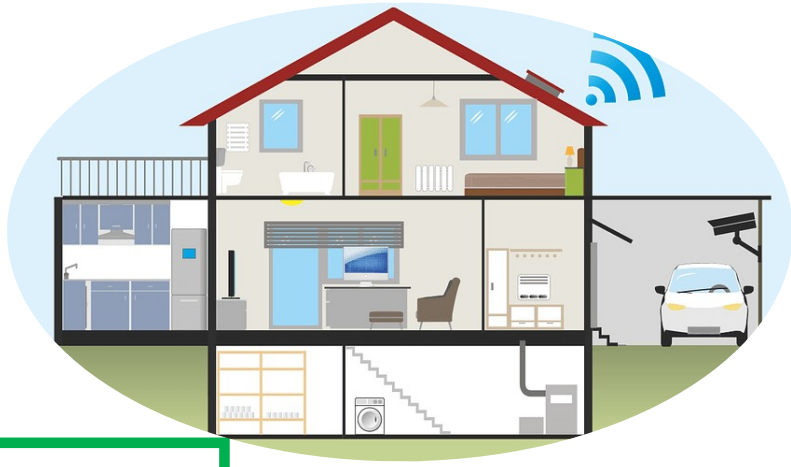
Cameras

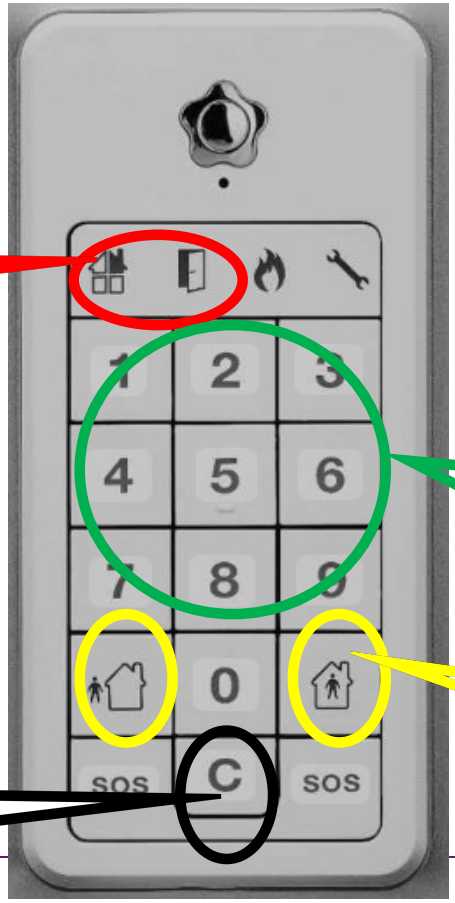
Sensors

Alarm

Keypad

USER INTERACTION WITH THE SECURITY SYSTEM KEYPAD





Alarm on

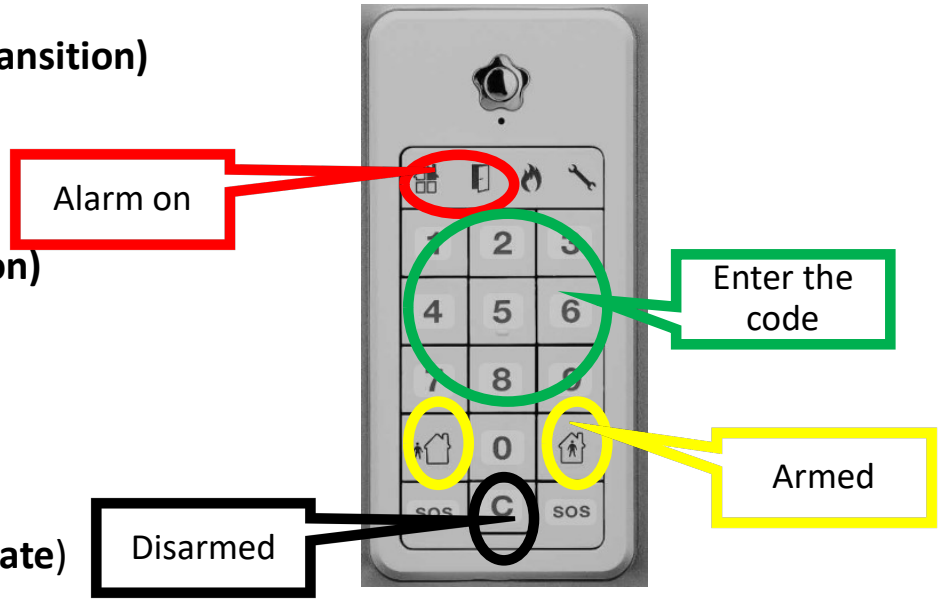
Enter the code

Armed

Disarmed

CREATE A STATE CHART FOR A SECURITY SYSTEM

- Goal of the model: describe how the user interacts with the system
  - The user can **arm the security system (disarmed → armed transition)**
    - *First, select* the mode "armed" and then *dial the code*
    - (for now, in-house or out-of-house are one single state)
  - The user can **disarm the system (armed → disarmed transition)**
    - *First, select* the mode "C" and then *dial the code*
  - If the alarm is on (because a sensor detected an intrusion), the user can **deactivate the alarm** by selecting "C" and then, *dialling the security code* (**the system returns to the armed state**)



Duration: 10 minutes designing + 5 minutes reporting

Think → Pair → Share

A POSSIBLE SOLUTION (1)

?

What is the problem here?

HSS

interface:

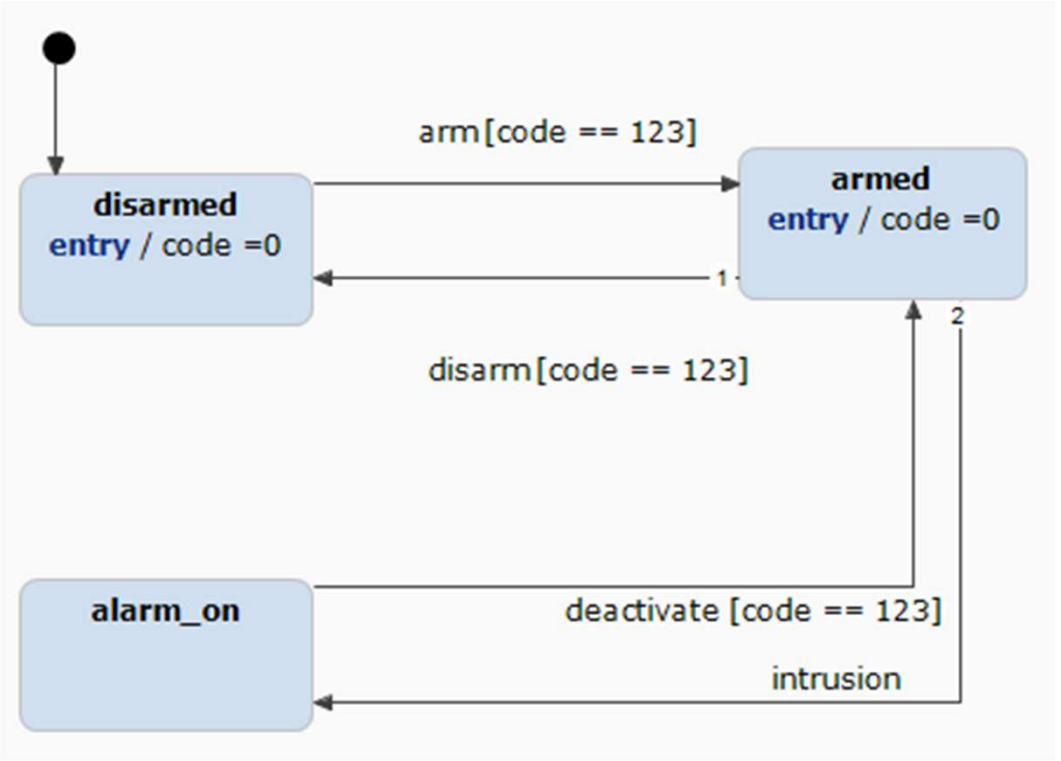
in event arm

in event disarm

in event intrusion

in event deactivate

var code:integer



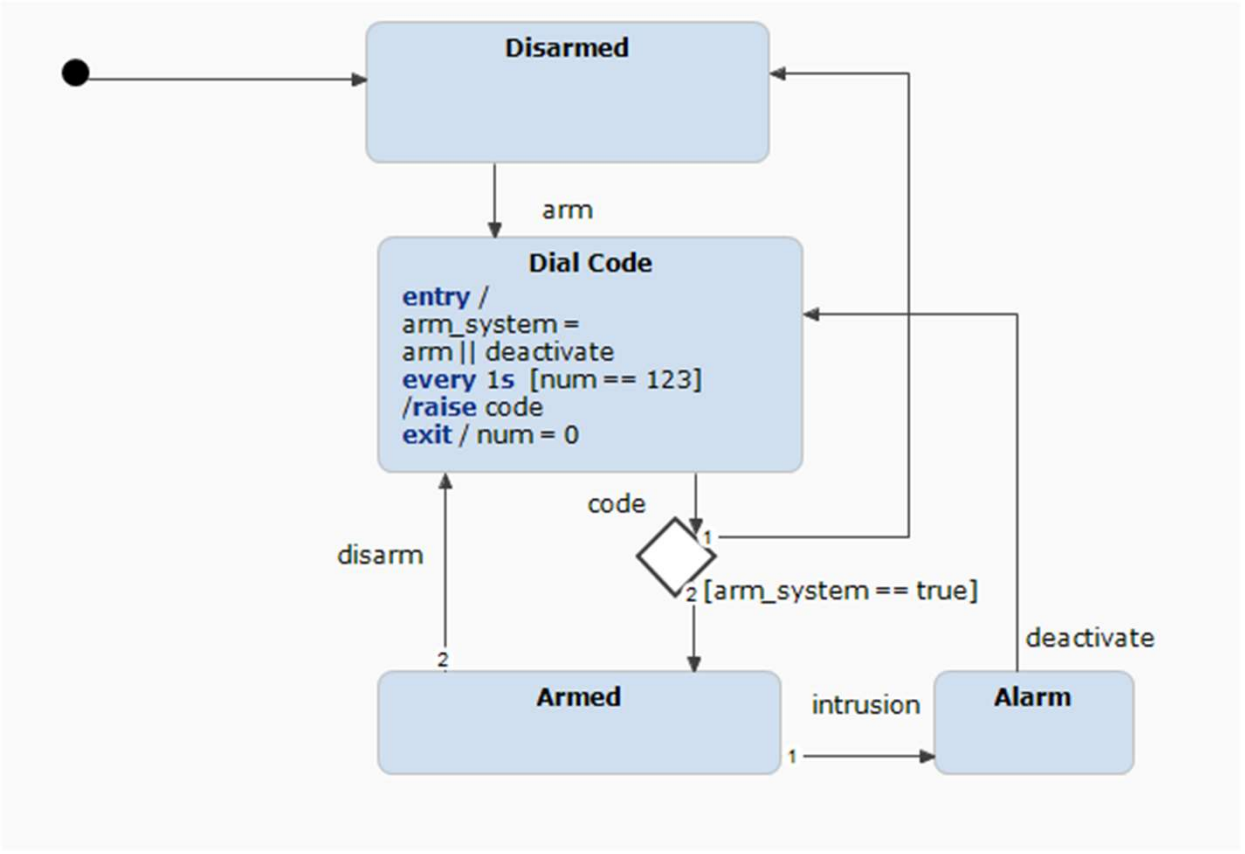
A POSSIBLE SOLUTION (2)

Statechart

interface:
- **in event** arm
- **in event** disarm
- **in event** intrusion
- **in event** deactivate

- **var** num:integer
- **var** arm_system: boolean

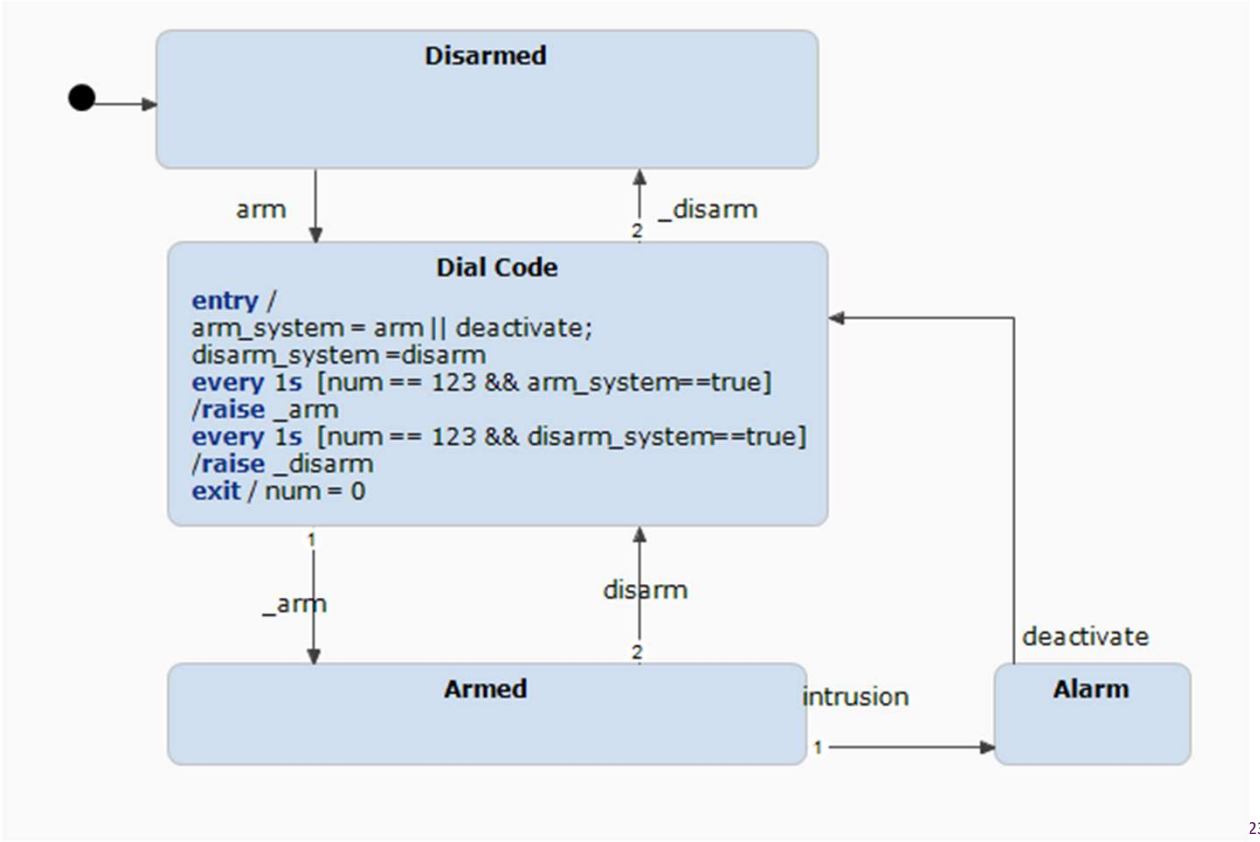
internal:
- **event** code



A POSSIBLE SOLUTION (3)

Statechart

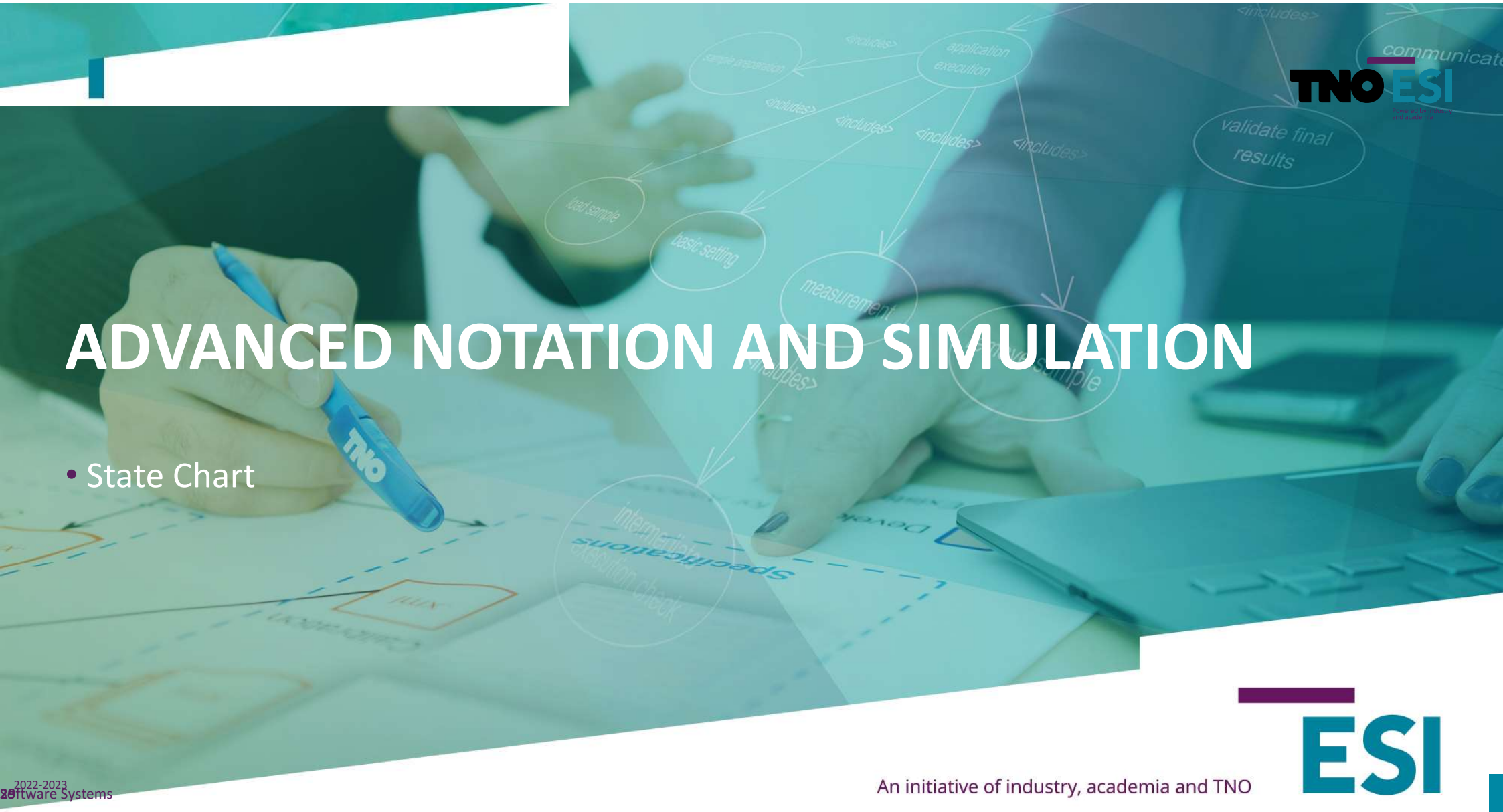
```
interface:
    in event arm
    in event disarm
    in event intrusion
    in event deactivate

    var num:integer
    var arm_system: boolean
    var disarm_system: boolean

internal:
    event _arm
    event _disarm
```



SIMULATION

TNO ESI
Powered by industry and academia

Live demo



ADVANCED NOTATION AND SIMULATION

- State Chart

ADVANCED CONCEPTS OF STATECHARTS

- Composite state
- Named entries and exits
- History nodes

A STATE CHART FOR A SECURITY SYSTEM

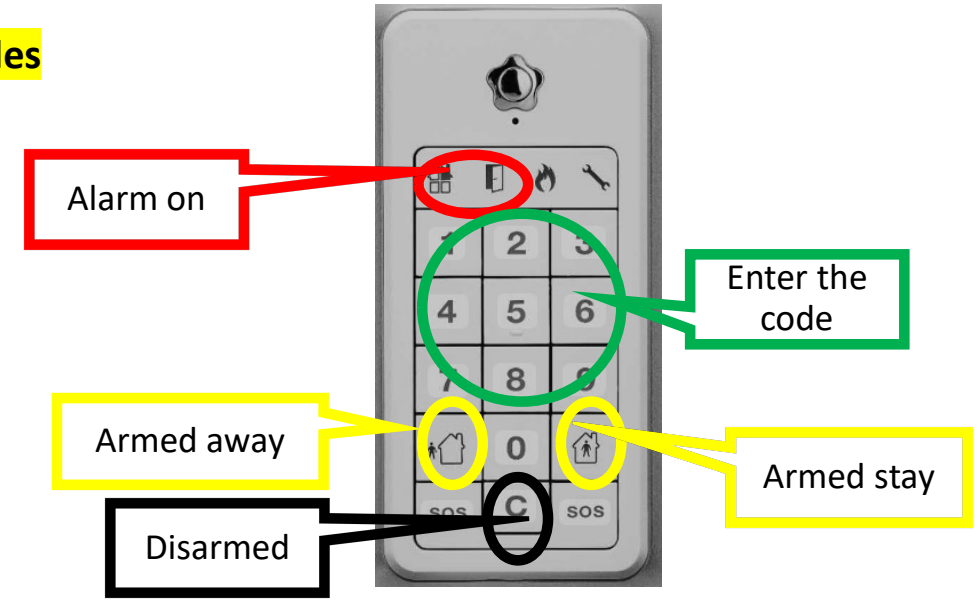
- Goal of the model: describe how the user interacts with the system

RQ1

- The user can **arm the security system in STAY or AWAY modes**

  (disarmed → away/stay or away ←→stay transitions)

  - *First, select* the mode and then *dial the code*
  - (mode armed away waits 5s before being activated)
- The user can **disarm the system**
  - *First, select* the mode "C" and then *dial the code*
- If the alarm is on (because a sensor detected an intrusion), the user can **deactivate the alarm** by selecting "C" and then, *dialling the security code*
  - After deactivating the alarm, the system returns to the exact same armed state it was before the alarm was turned on.



WE START FROM THIS SOLUTION

RQ1

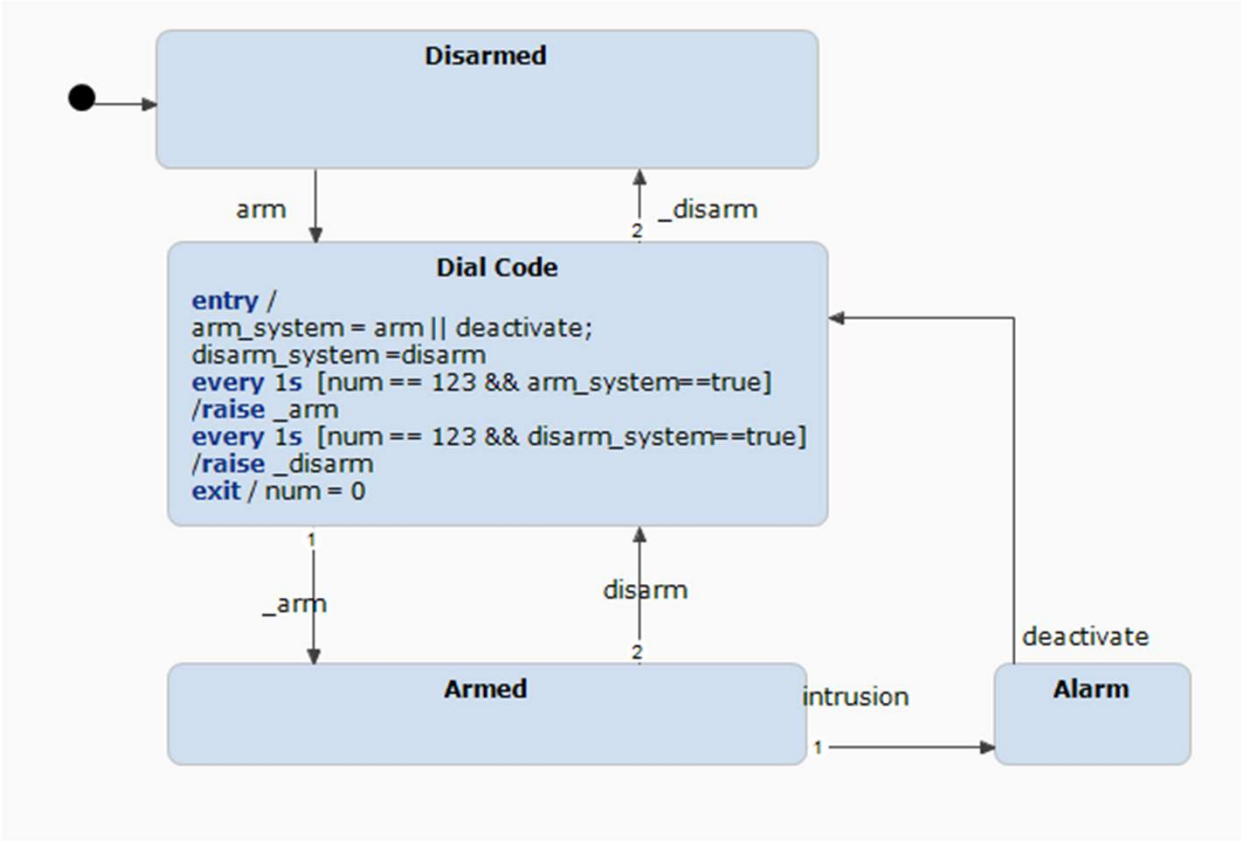
The user can **arm the security system in STAY or AWAY modes (disarmed → away/stay or away ←→stay transitions)**

Statechart

interface:
- **in event** arm
- **in event** disarm
- **in event** intrusion
- **in event** deactivate

- **var** num:integer
- **var** arm_system: boolean
- **var** disarm_system: boolean

internal:
- **event** _arm
- **event** _disarm



RQ1

The user can **arm the security system in STAY or AWAY modes**

armed_hierarchy

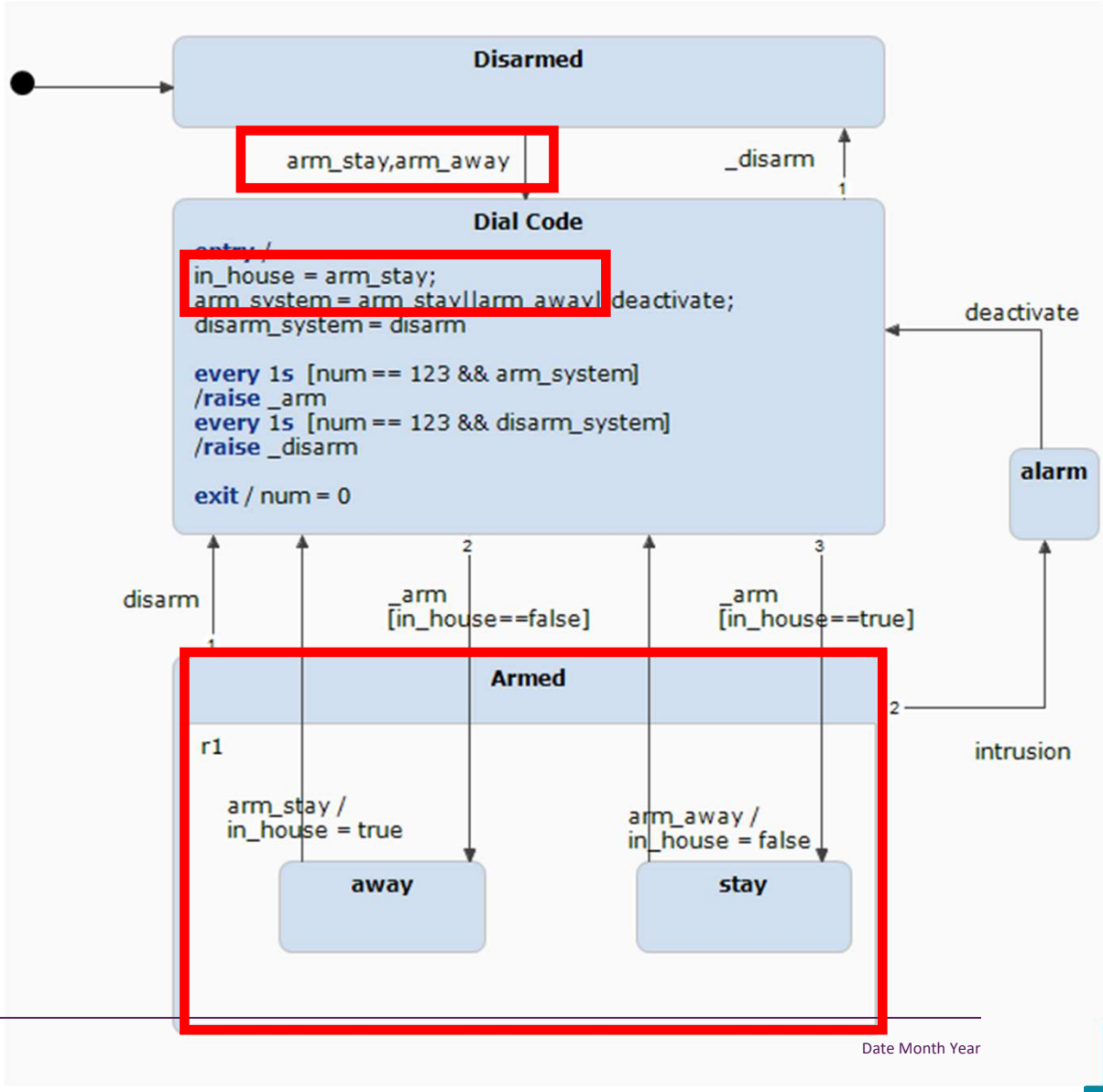
interface:

in event arm_stay
in event arm_away
in event disarm
in event intrusion
in event deactivate

var in_house: boolean
var num: integer
var arm_system: boolean
var disarm_system: boolean

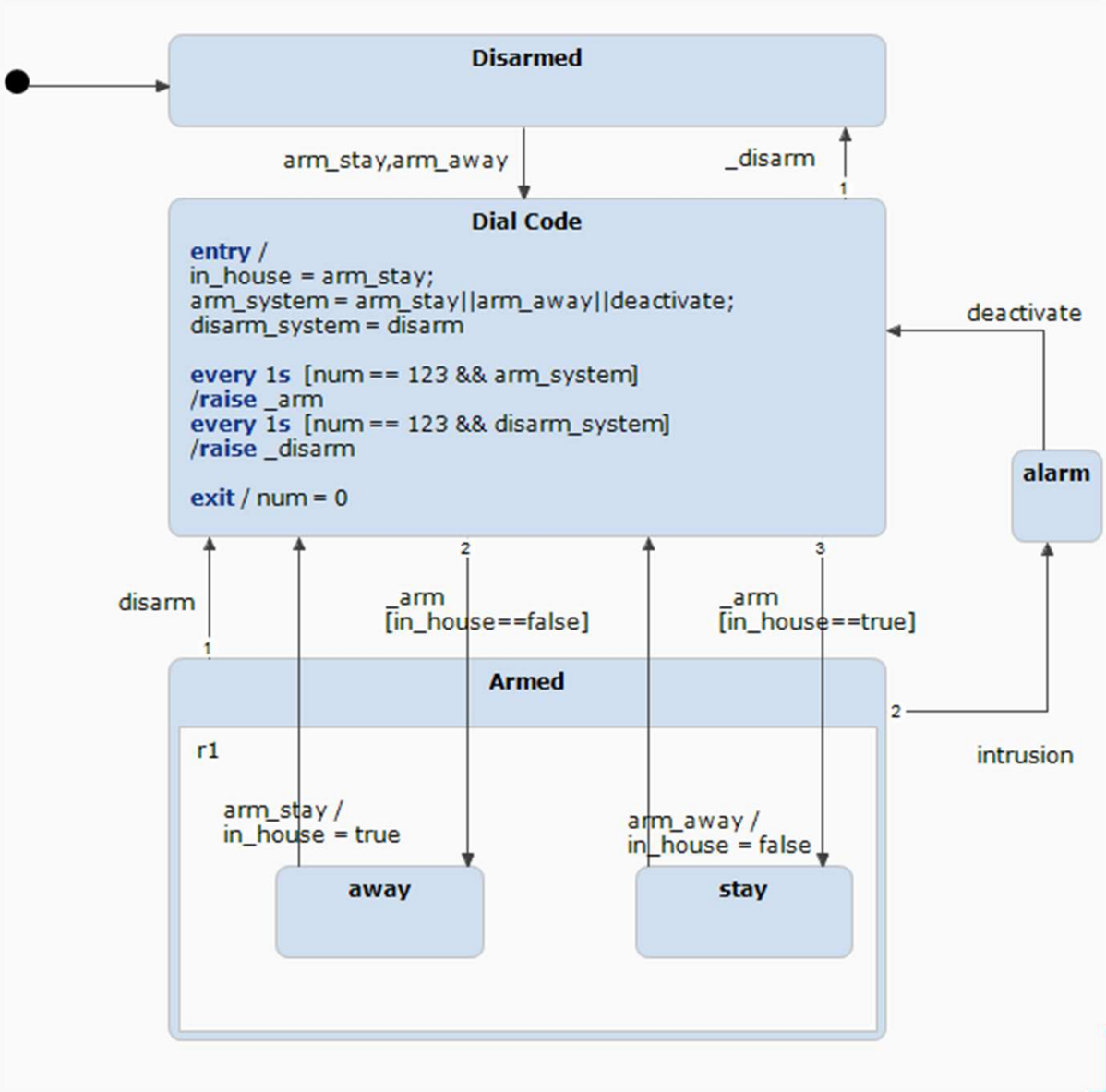
internal:

event _arm
event _disarm



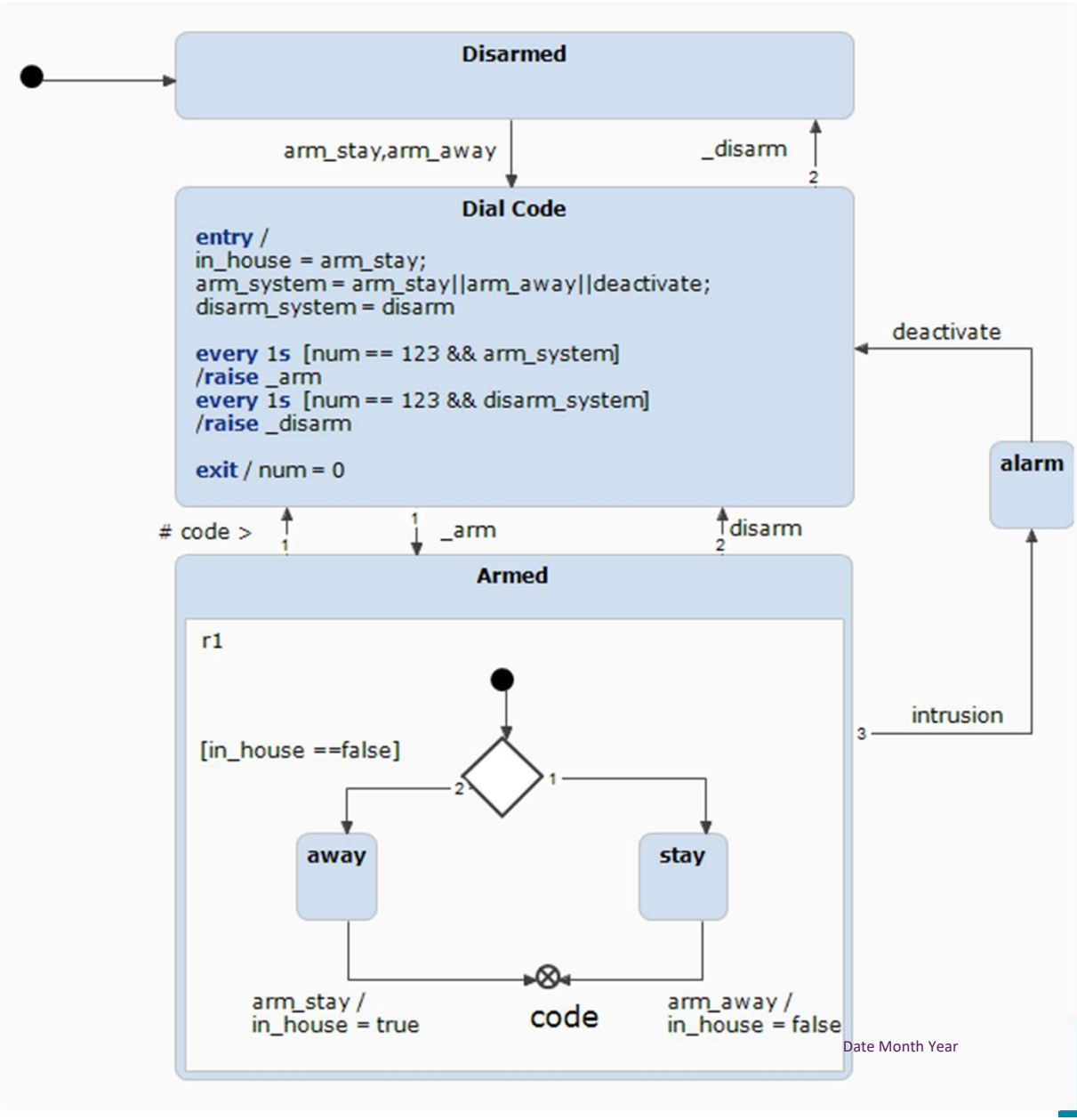
COMPOSITE STATES

- State that contains one or more substates (Armed)
- Outgoing transitions from a state apply to all its substates
- Transitions can point to either a state or a substate



(MULTIPLE) NAMED ENTRIES AND EXITS

- Whenever entering a composite state, the default (i.e., unnamed) entry node is activated
- It is possible to have (multiple) named entry nodes (with unique names)
- It is possible to have (multiple) named exit nodes (with unique names)



SIMULATION

Live demo

A STATE CHART FOR A SECURITY SYSTEM

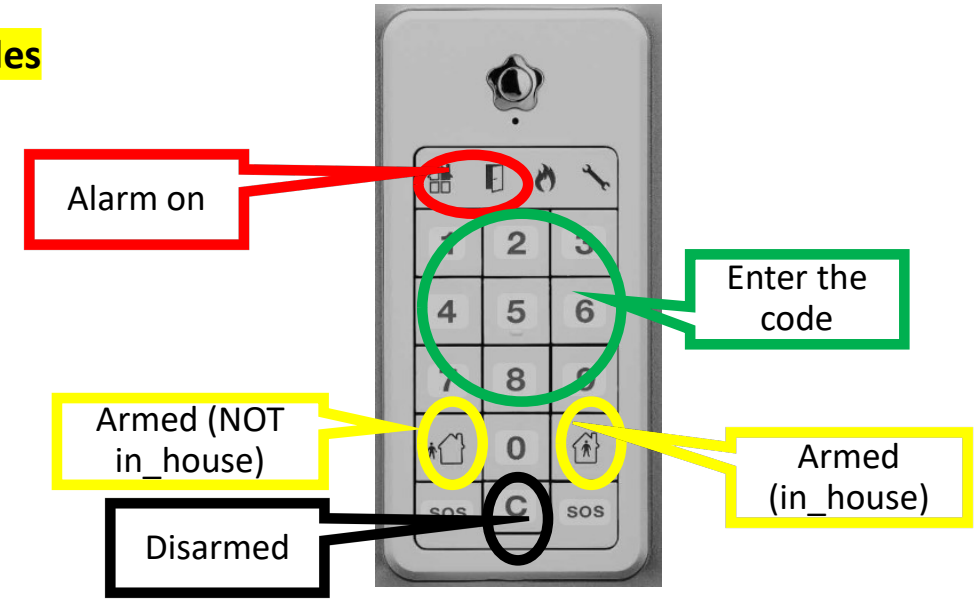
- Goal of the model: describe how the user interacts with the system

RQ1

- The user can **arm the security system in STAY or AWAY modes**

  (disarmed → away/stay or away ←→stay transitions)

  - *First, select* the mode and then *dial the code*
  - (mode armed away waits 5s before being activated)
- The user can **disarm the system**
  - *First, select* the mode "C" and then *dial the code*
- If the alarm is on (because a sensor detected an intrusion), the user can **deactivate the alarm** by selecting "C" and then, *dialling the security code*

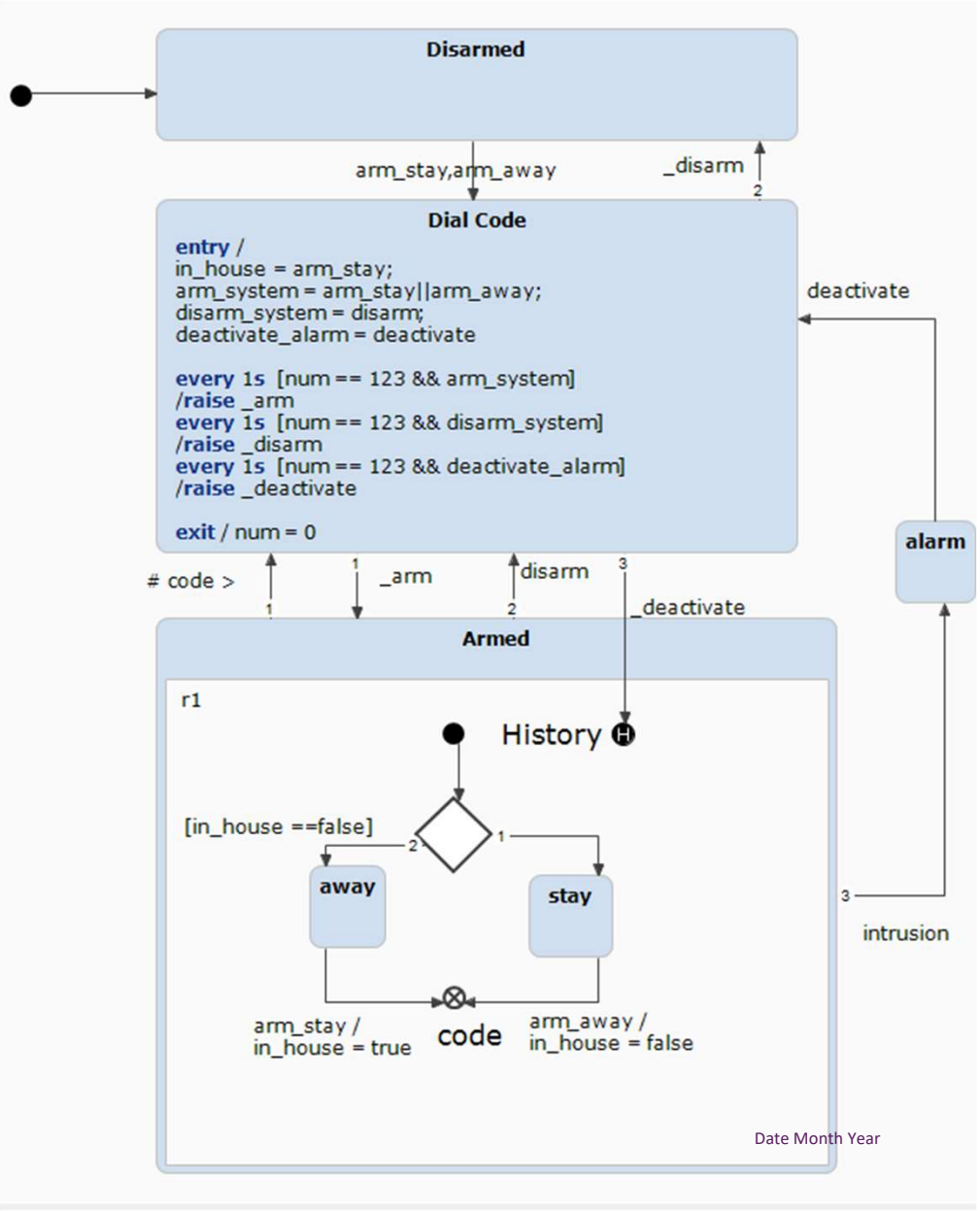


RQ2

- After deactivating the alarm, the system returns to the exact same armed state it was before the alarm was turned on.

HISTORY NODES

- (Default: don’t remember the state that was active when the composite state was left)
- Shallow: remember the state that was active when the composite state was left
- Deep: remember all nested states when the composite state was left



SIMULATION

Live demo

MODELING SKILLS

- State Chart

An initiative of industry, academia and TNO

ESI

CREATE A STATE CHART FOR A SECURITY SYSTEM

- Goal of the model: describe how the user interacts with the system

RQ1

- The user can **arm the security system in STAY or AWAY modes**

  (disarmed → away/stay or away ←→stay transitions)

  - *First, select* the mode and then *dial the code*

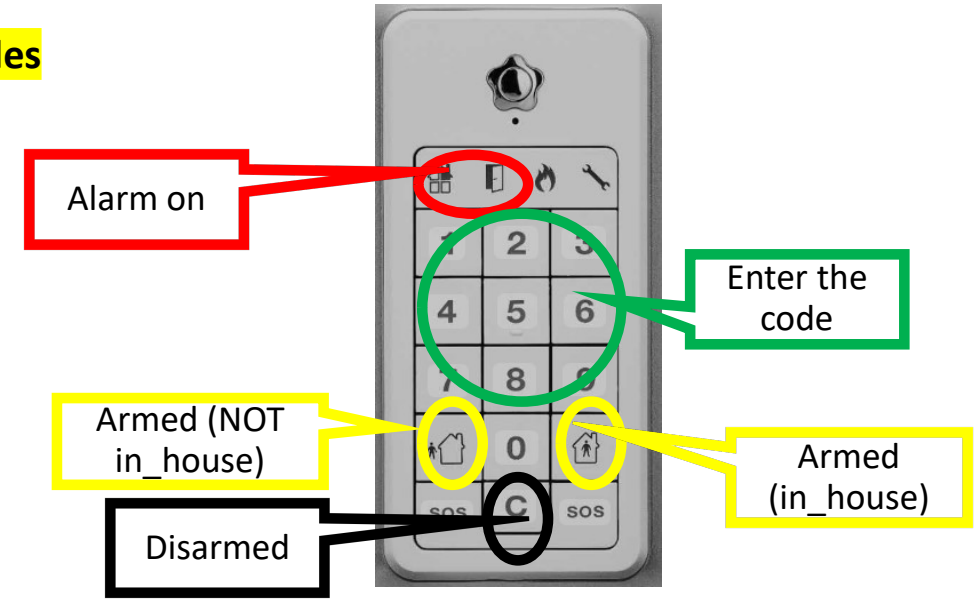
RQ3

  - (mode armed away waits 5s before being activated)

- The user can **disarm the system**
  - *First, select* the mode "C" and then *dial the code*
- If the alarm is on (because a sensor detected an intrusion), the user can **deactivate the alarm** by selecting "C" and then, *dialling the security code*

RQ2

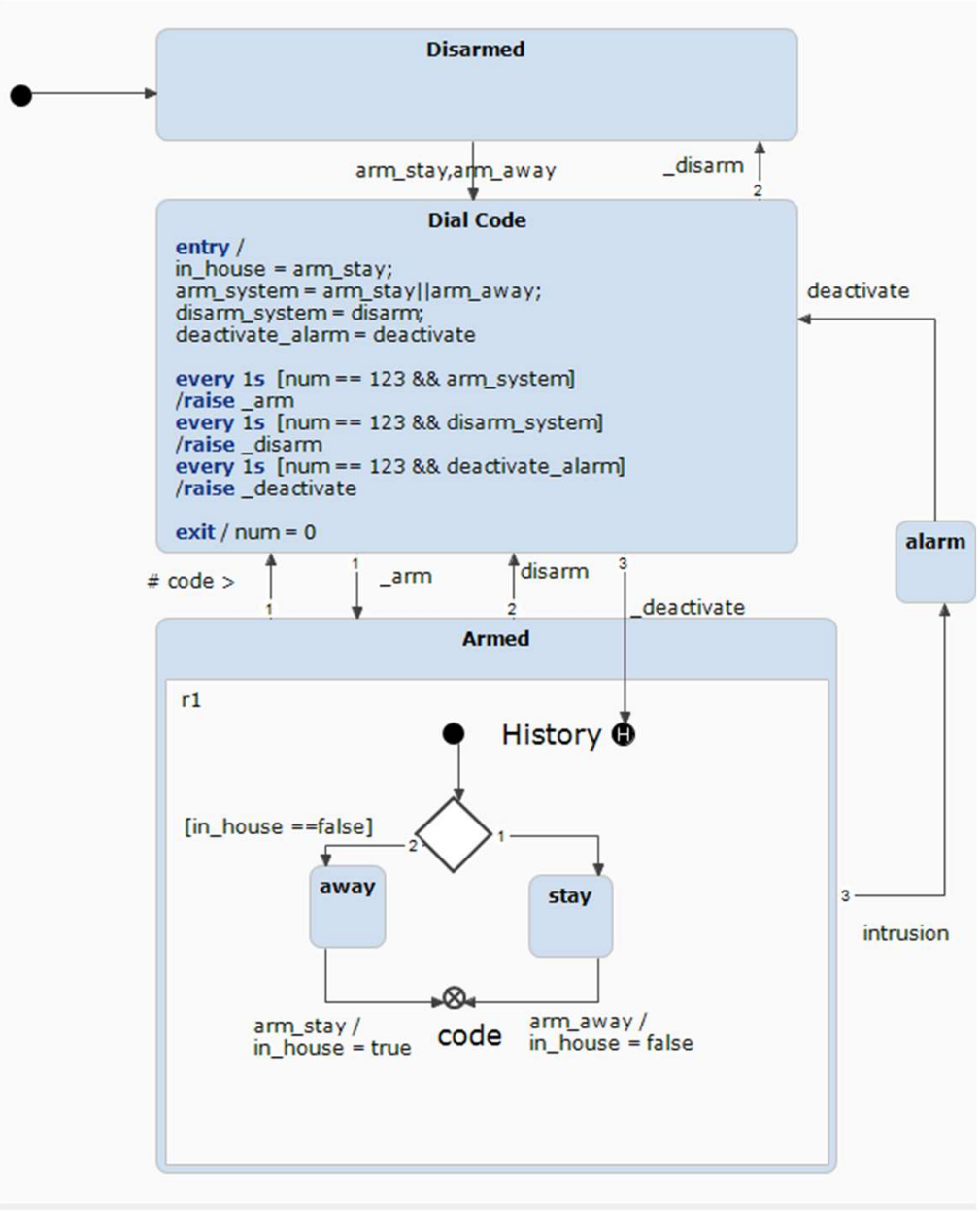
  - After deactivating the alarm, the system returns to the exact same armed state it was before the alarm was turned on.



MODIFY THE STATE CHART TO MEET RQ3

- (mode armed out-of-house waits 5s before being activated)
- Duration: 5 minutes designing and 3 minutes reporting

Think → Pair → Share



HSS_final

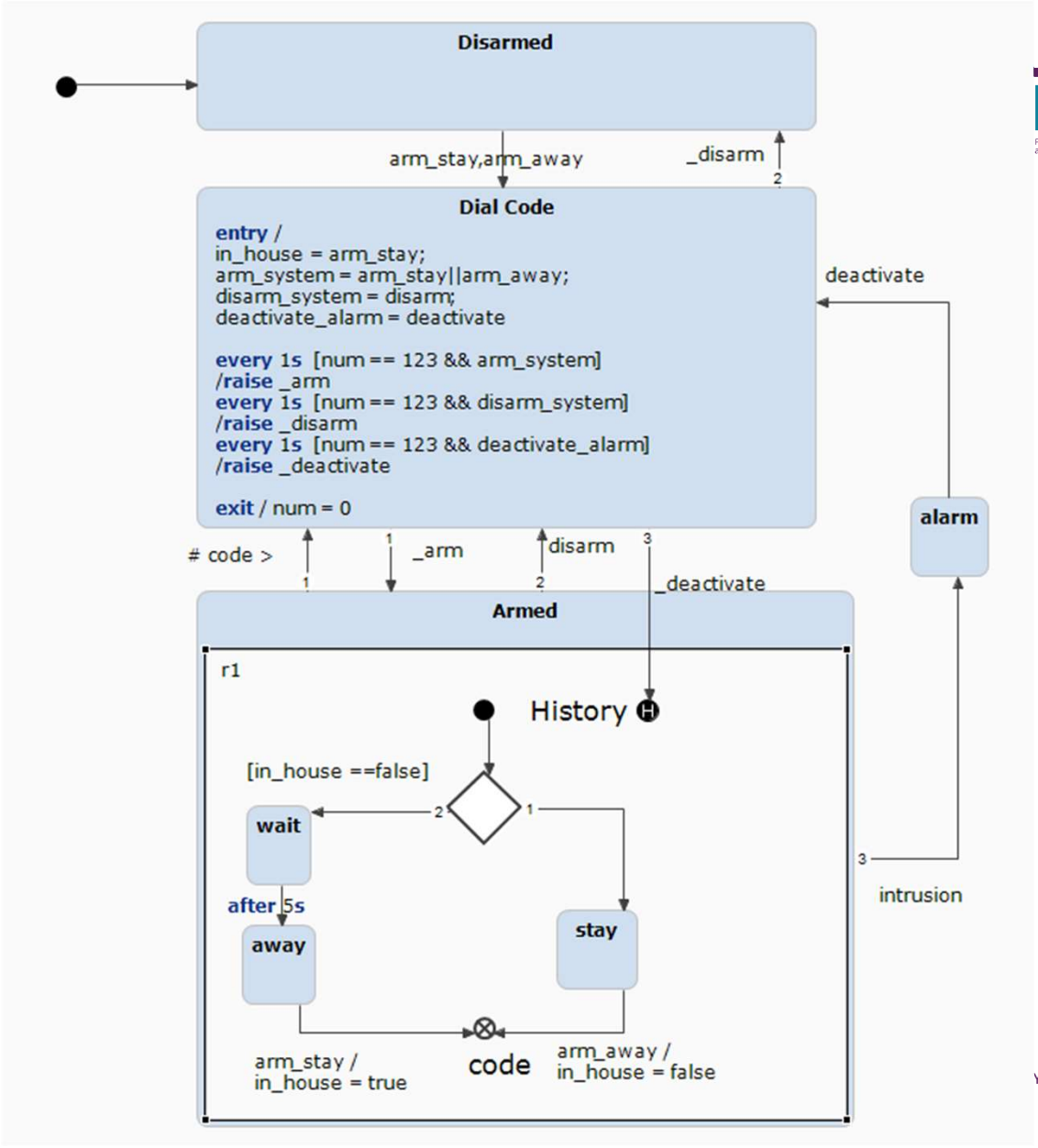
interface:

in event arm_stay
in event arm_away
in event disarm
in event intrusion
in event deactivate

var in_house: boolean
var num: integer
var arm_system: boolean
var disarm_system: boolean
var deactivate_alarm: boolean

internal:

event _arm
event _disarm
event _deactivate



SIMULATION

TNO ESI
Powered by industry and academia

Live demo

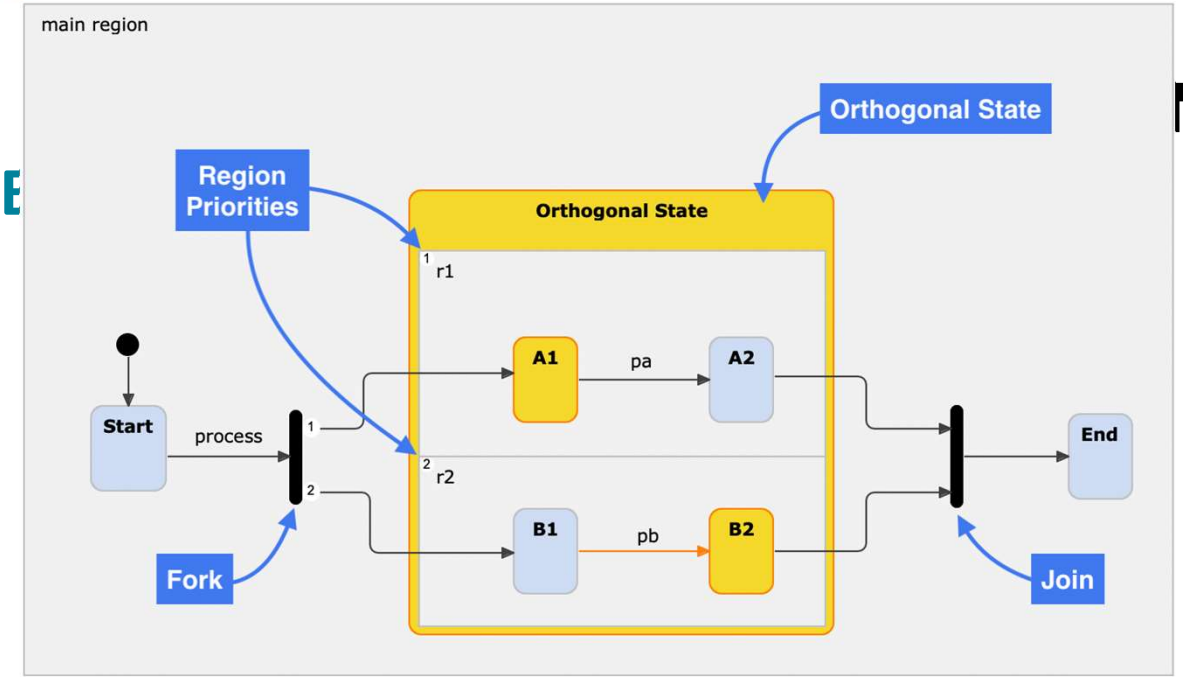
OPTIONAL ADVANCED CONCEPTS

- State Charts

An initiative of industry, academia and TNO

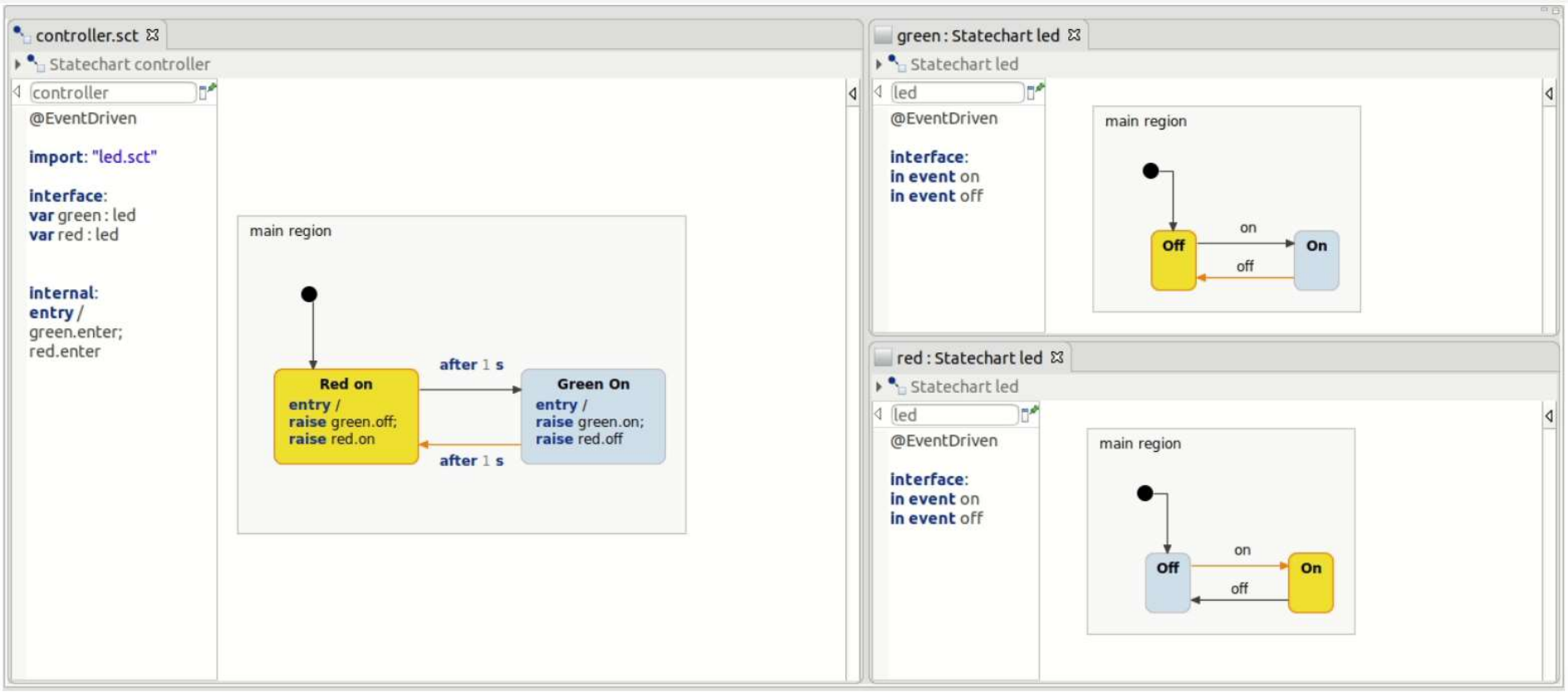
ESI

ORTHOGONAL STATE



- Orthogonal state
  - Surrounded by Fork and Join nodes
  - System can be in multiple states simultaneously
    - Transitions are executed sequentially
    - Orthogonal regions have priorities
  - Orthogonal regions can communicate via internal events

MULTI STATE-MACHINE MODELLING



CLOSING REMARKS

- State Charts

An initiative of industry, academia and TNO

ESI

HAVE YOU REACHED THE OBJECTIVES?

- How many of you knew about State Charts before today?
- Do you understand the purpose and application areas of State Charts?
- Do you feel capable of explaining the concepts and notations of State Charts?
- Could you start designing basic State Charts to model software systems?

CLOSING REMARKS

- Strongly suggested: work on “Notation and simulation” before the lab session
  - Download and install the Itemis CREATE
  - Eclipse → Help → Help Contents → Itemis CREATE documentation
    - Tutorials → Comprehensive tutorial
- Optional (if you are interested in test-driven development):
  - Eclipse → Help → Help Contents → Itemis CREATE documentation
    - User guide → Testing state machines
      - **NOTE: Up to section 1.3 only! So stop before “1.4 The SCTUnit language”**