Finite State Machine

machine à états finis

Julien Deantoni
http://www.i3s.unice.fr/~deantoni/

problème

- En pseudo code, implémentez une lampe de bureau tactile. Cette lampe possède un interrupteur permettant d'être consciente de la luminosité ambiante ou non.
  - *Lorsque lampe est éteinte:*
    - *Si l'interrupteur est sur ON,*
      - *la toucher l'allume en lumière forte si la luminosité est comprise entre 0 et 400 lux, sinon la lampe s'allume en lumière douce.*
      - *Si un deuxième touché est effectué dans les 2 secondes qui suivent l'allumage, la lampe passe en lumière forte.*
    - *Si l'interrupteur est sur OFF, la toucher l'allume en lumière faible et un deuxième touché dans les 2 secondes l'allume en lumière forte.*
  - *Lorsque la lampe est allumée*

    un touché l'éteint.

```
API: isTouched(); switchOff(); switchSoft(); switchStrong()
```

Getting the feeling

- Mettez vous par 3 avec vos voisin, allez sur http://codeshare.io/

1) aJYZZE
2) G6z88k
3) 5QYxxx
4) 5w6MM7
5) 29woo7
6) 5eelD9
7) 2KYVz8
8) ad9mve
9) G86b0J
10) anm1x4
11) 5ol7RL
12) a3Jer1
13) Gko4Ew
14) amkAWo
15) al30MY
16) G860LZ
17) GLYPQ6
18) 2jbNqD

problème

- Permutation cyclique des groupes, dites moi si l'implémentation de vos collègues est bonne

Pourquoi ce cours

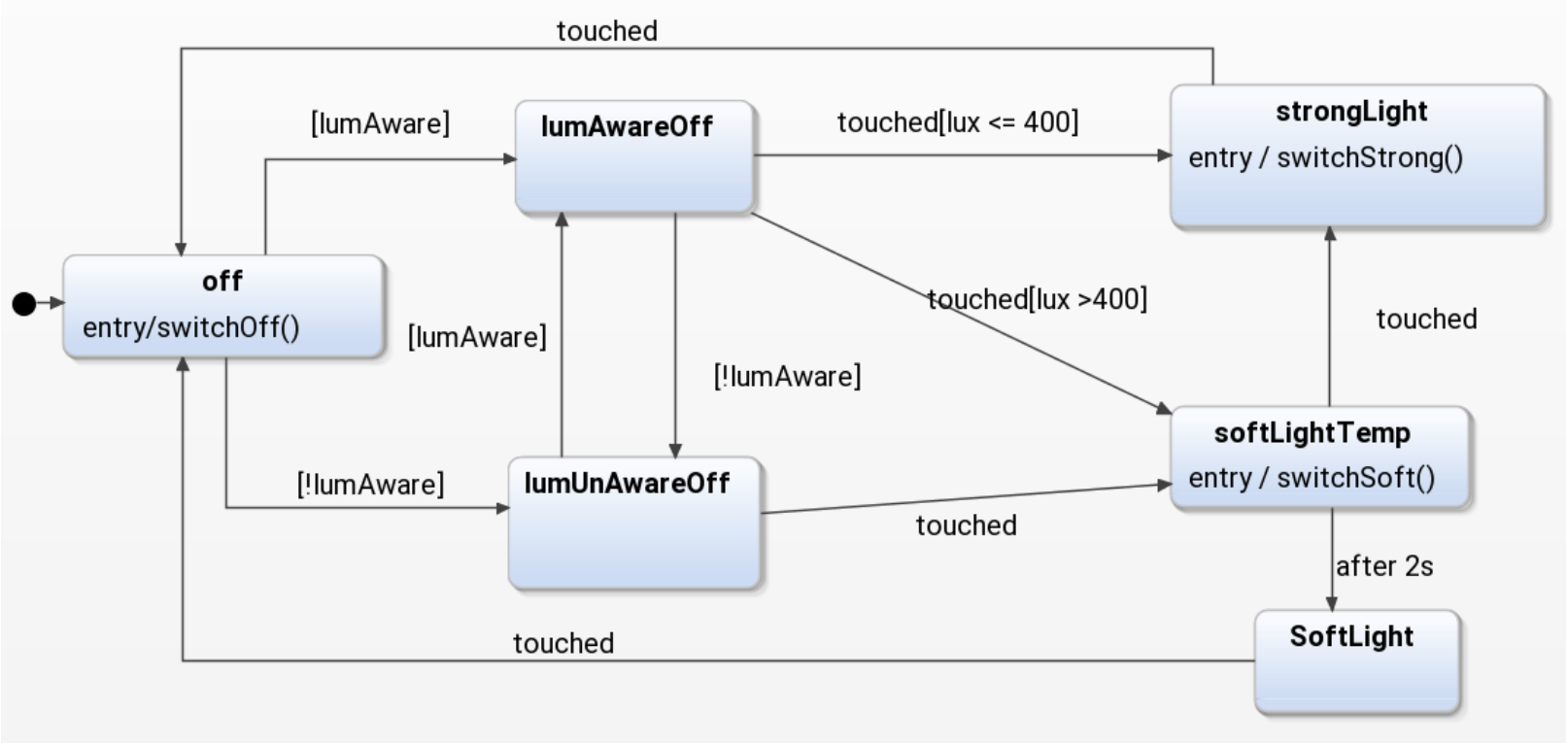
- **Vous faire connaître la notion de FSM**
- Vous faire comprendre les intérêts des machines à états pour un ingénieur
  - Pour structurer de votre code (génération de code)
  - Pour l'évolution de votre code
  - Pour la communication avec d'autres personnes
- Vous faire comprendre certains problèmes inhérents (différentes sémantiques, plus ou moins expressives)
- Vous ouvrir des perspectives sur les notions de V&V (composition d'automates, Labelled Transition Systems et logiques temporelles)

Out of scope

- Vous faire comprendre tous les problèmes théoriques sous-jacents
- Vous donnez un panel exhaustifs des différents dialectes de FSM
- Vous donnez toutes les manières d’utiliser les FSMs.
- Faire de vous des pros du model checking

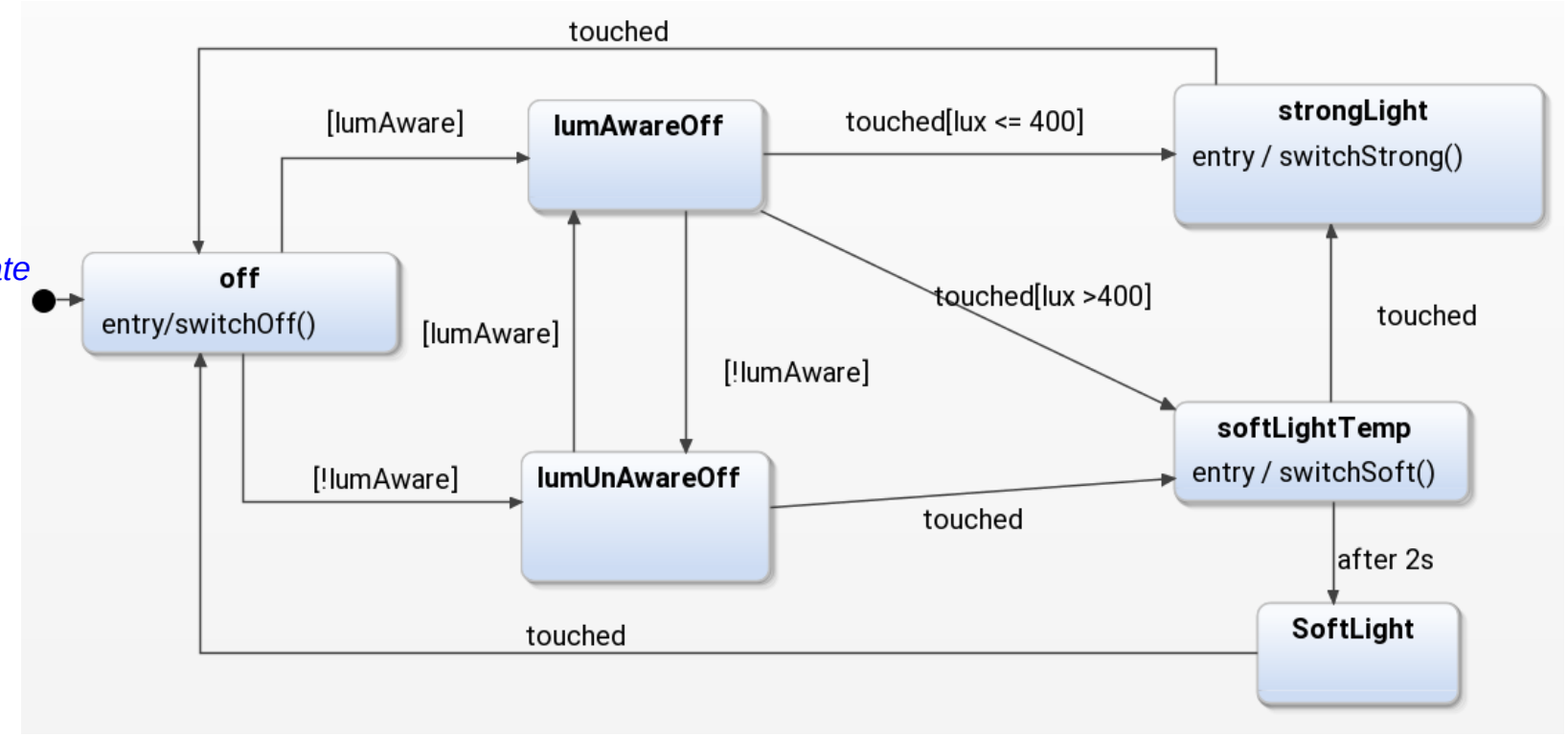
FSM

- C'est une abstraction permettant de structurer du code de contrôle.
  - Exemple simple de la lampe de bureau fait dans YAKINDU (https://www.itemis.com/en/yakindu/statechart-tools/)



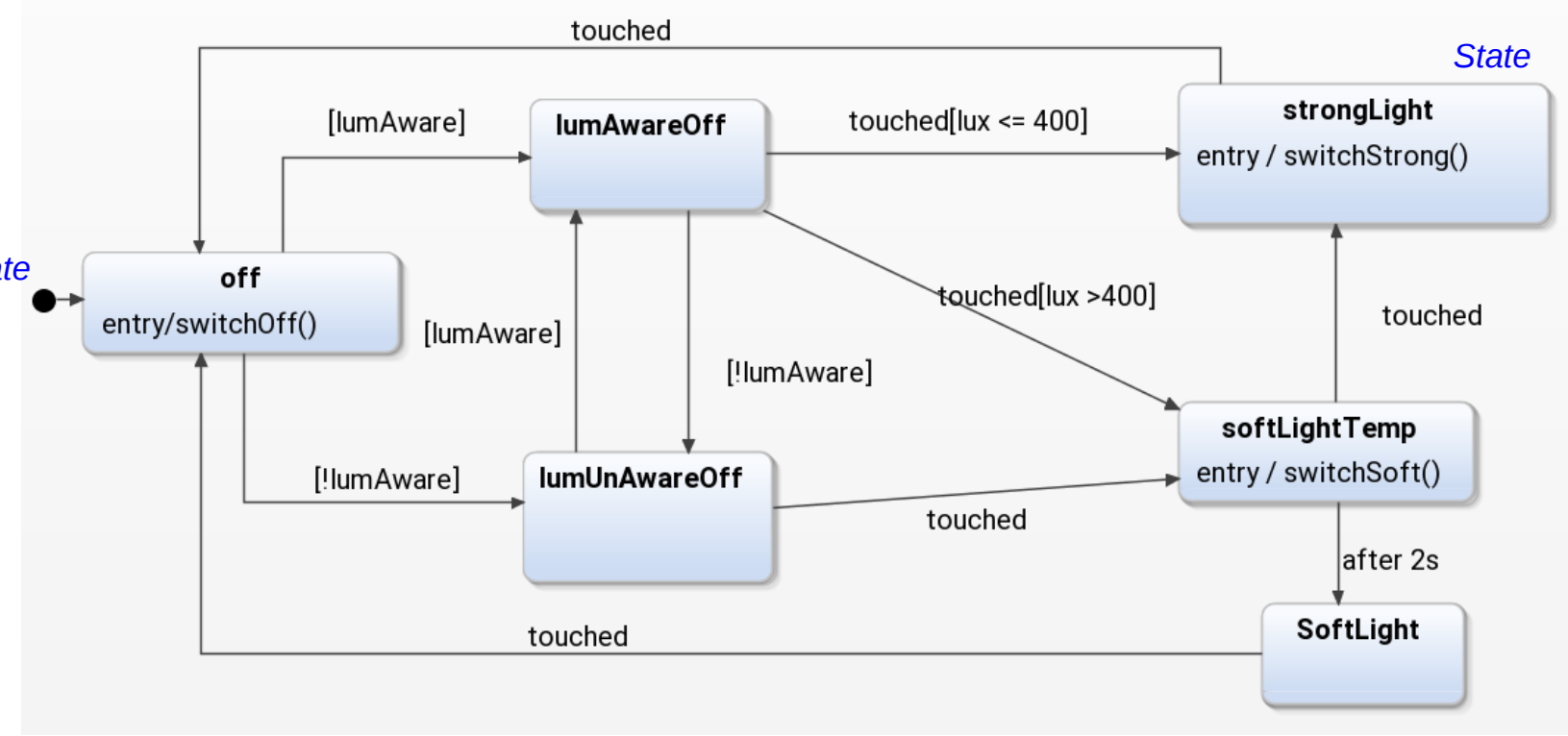
FSM

- C'est une abstraction permettant de structurer du code de contrôle.



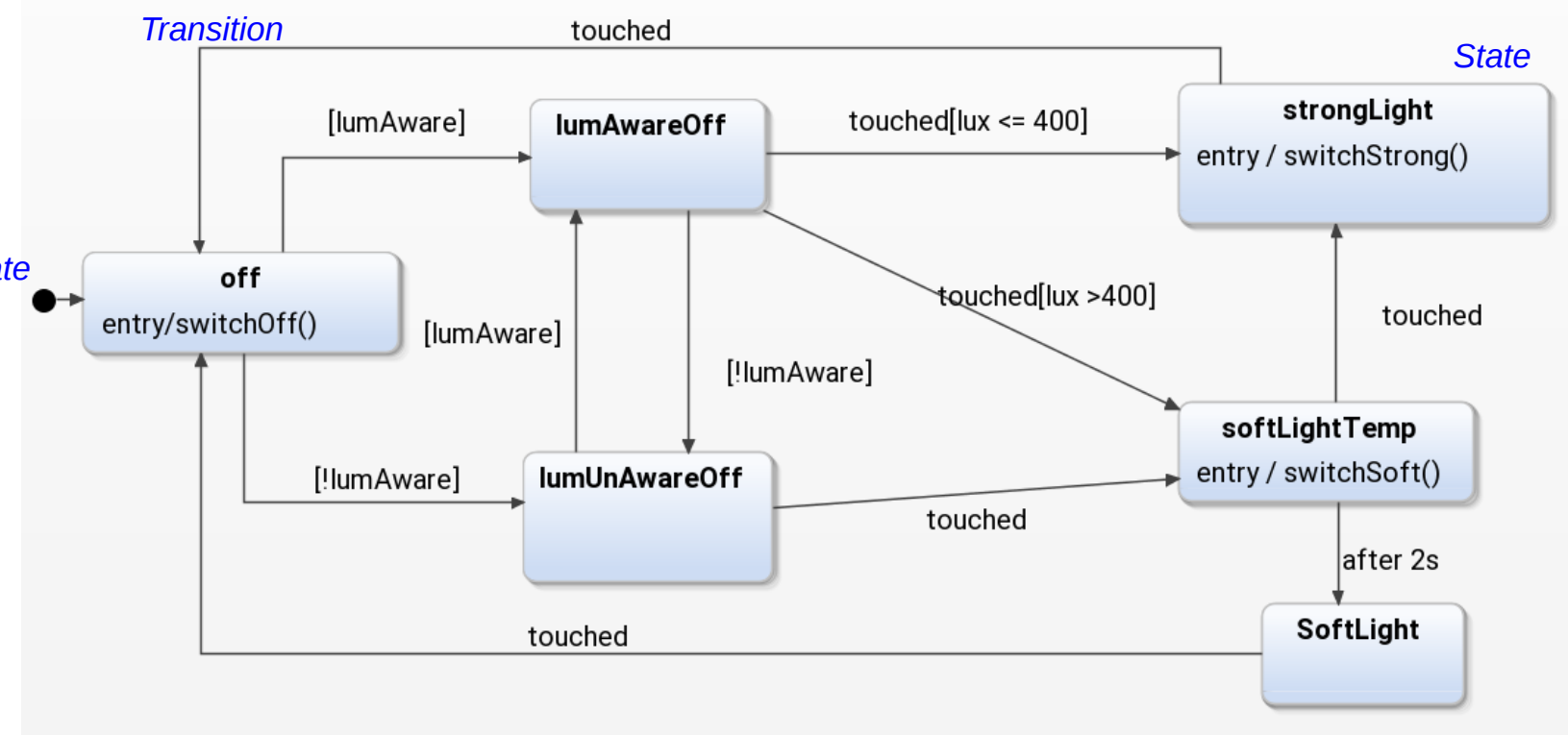
FSM

- C'est une abstraction permettant de structurer du code de contrôle.



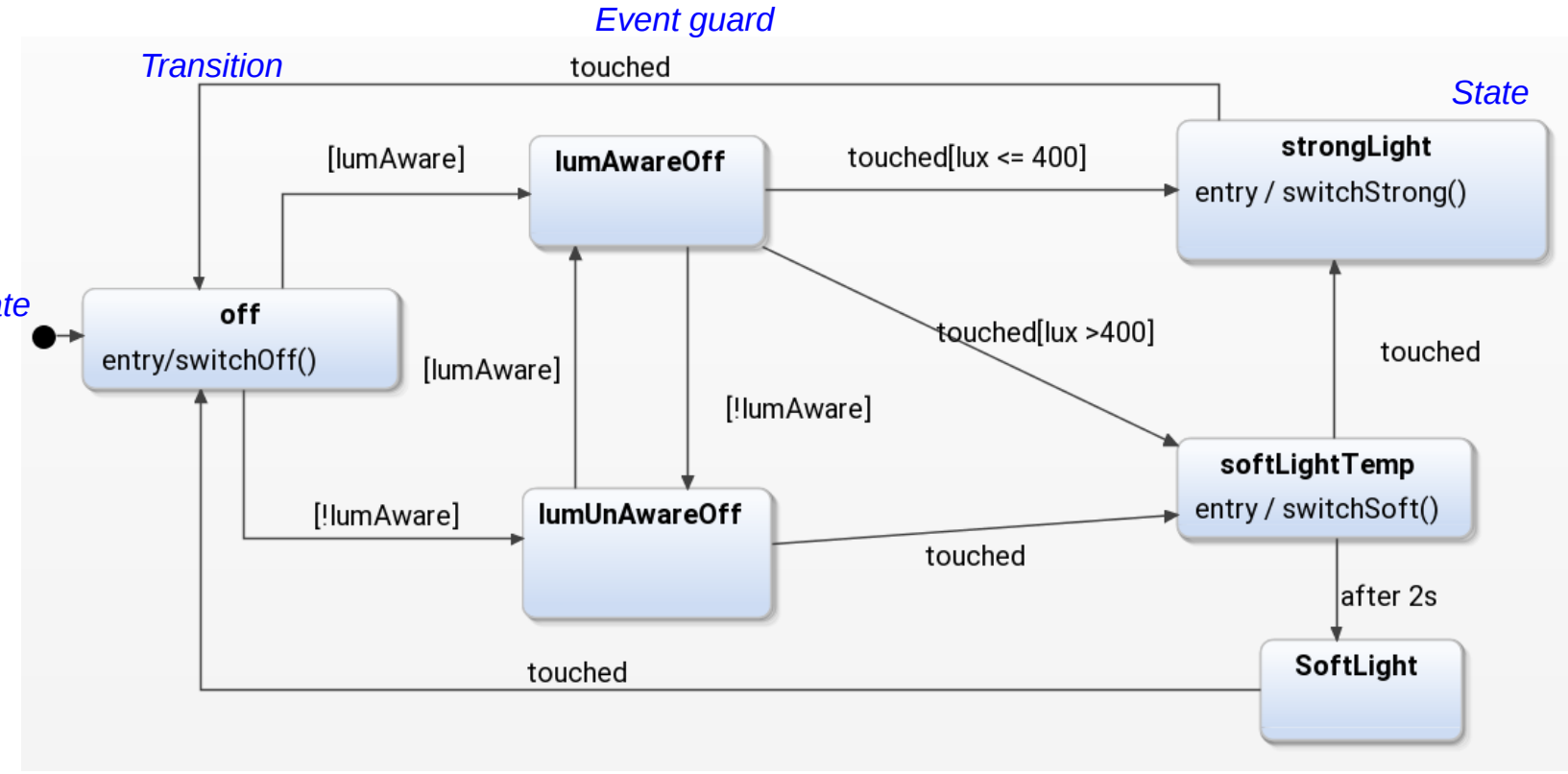
FSM

- C'est une abstraction permettant de structurer du code de contrôle.



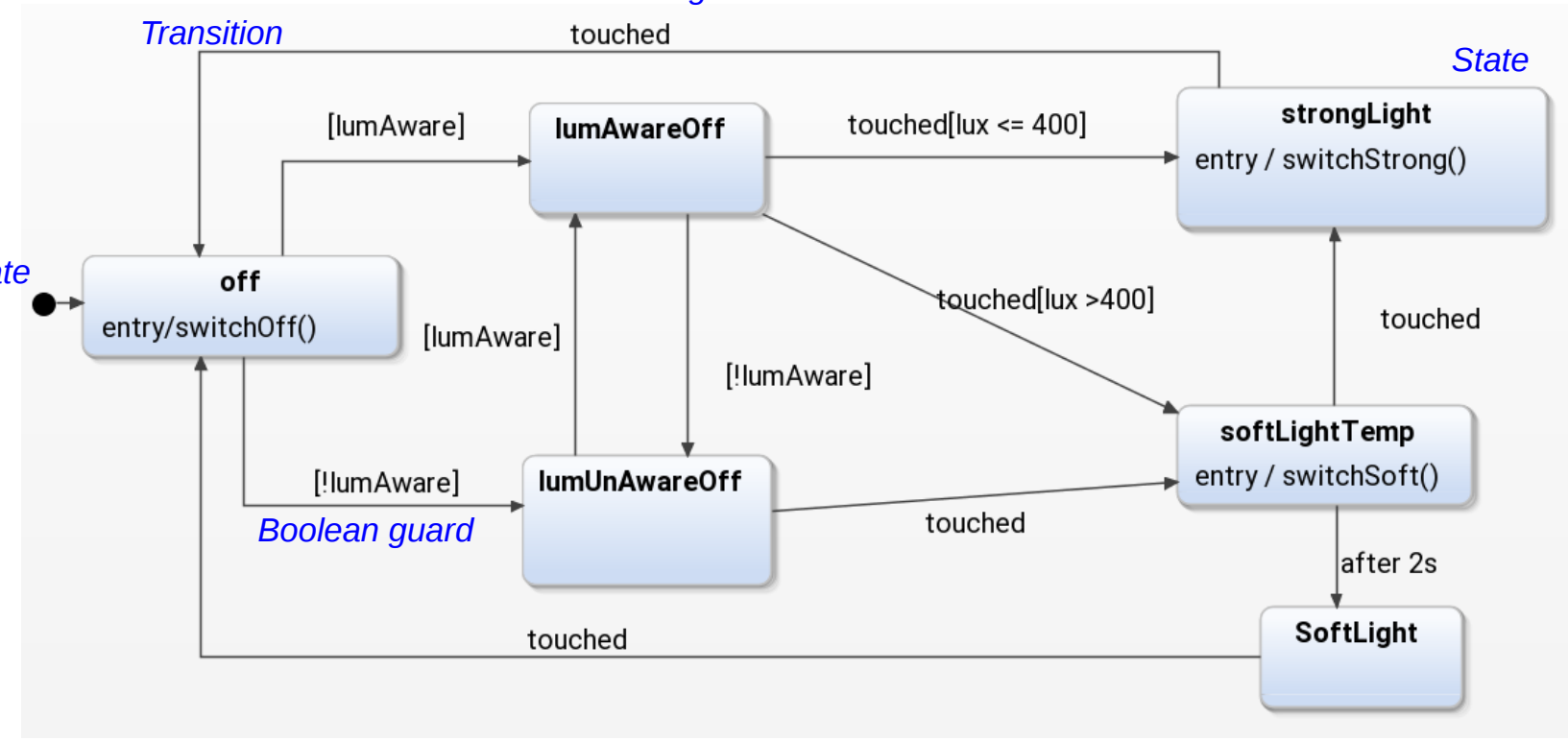
FSM

- C'est une abstraction permettant de structurer du code de contrôle.



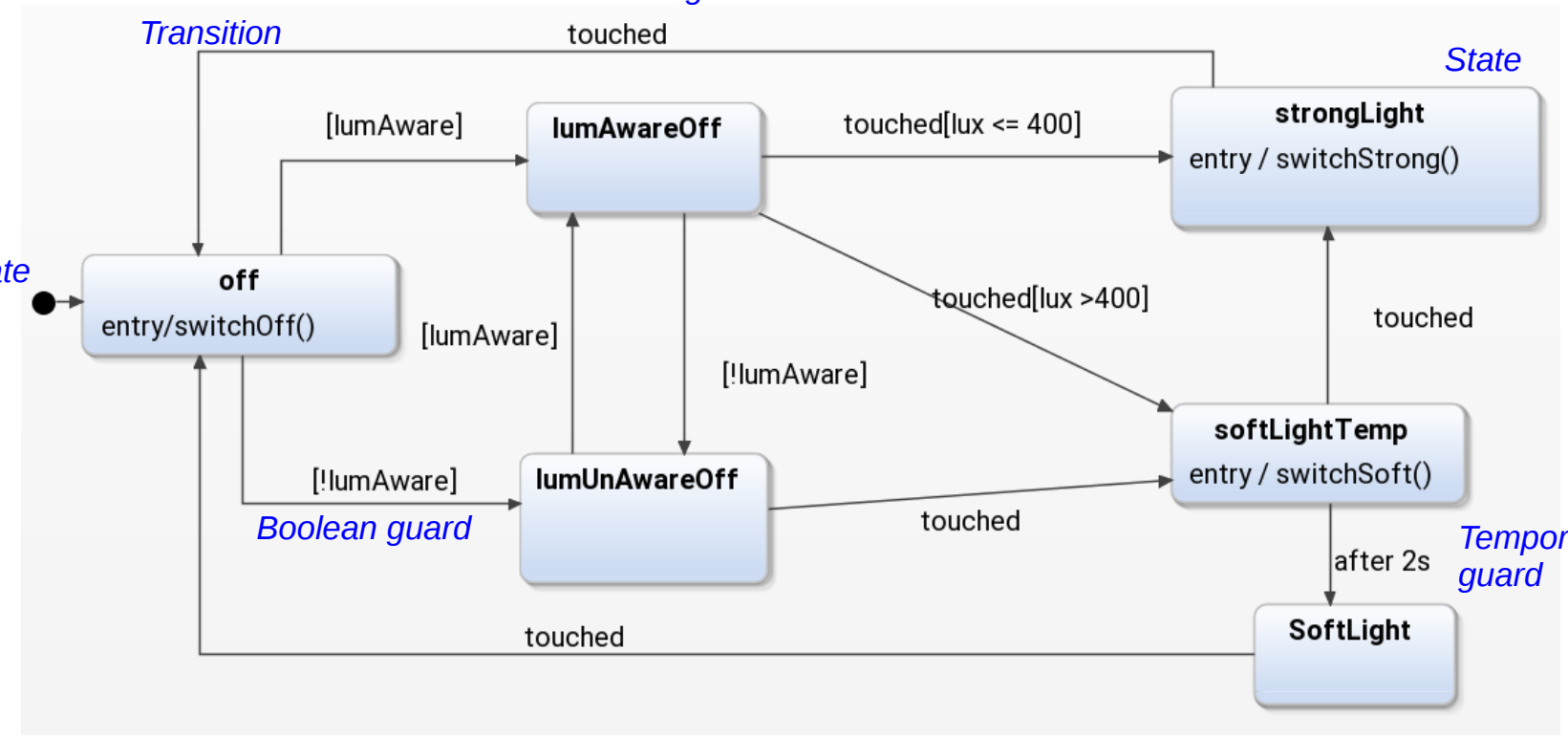
FSM

- C'est une abstraction permettant de structurer du code de contrôle.



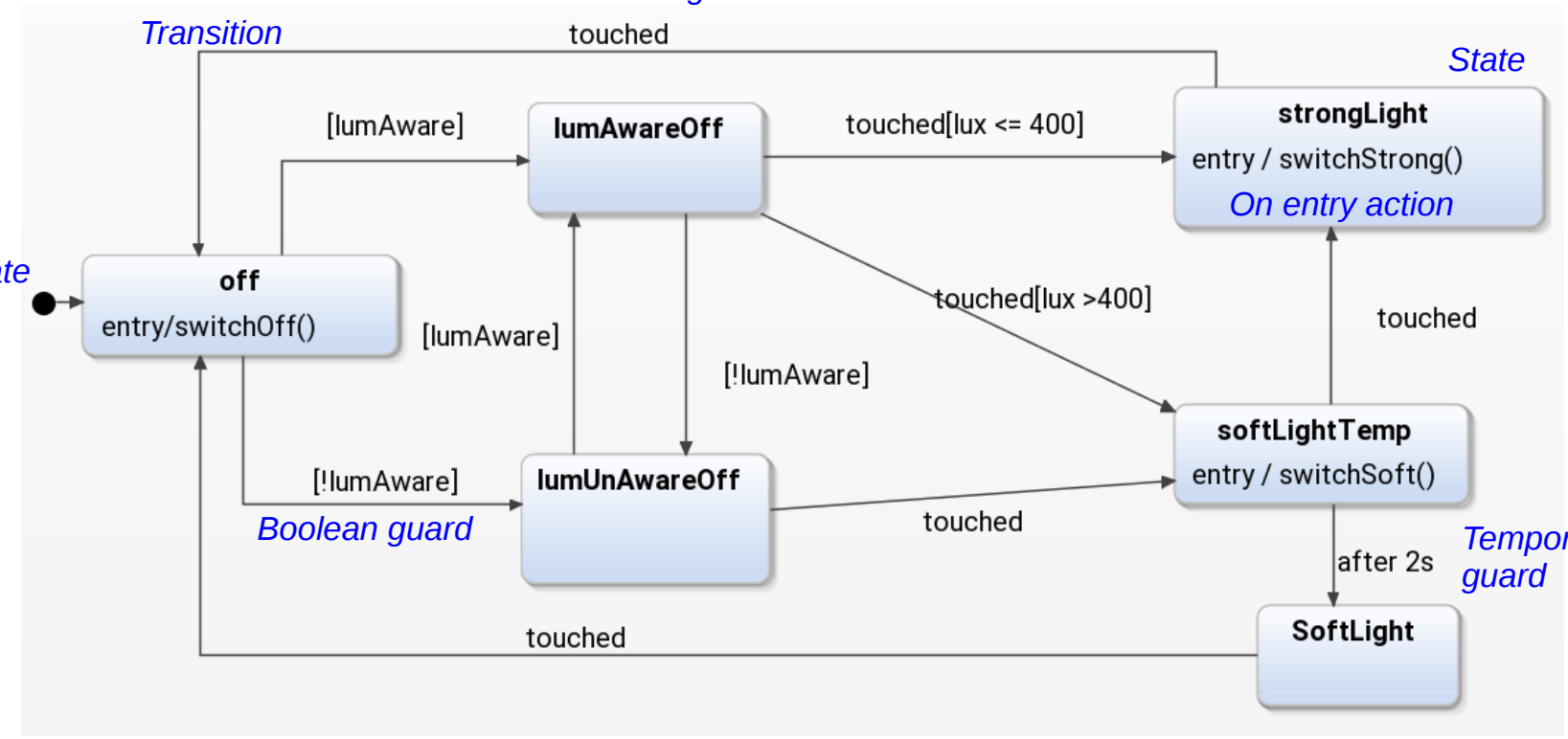
FSM

- C'est une abstraction permettant de structurer du code de contrôle.



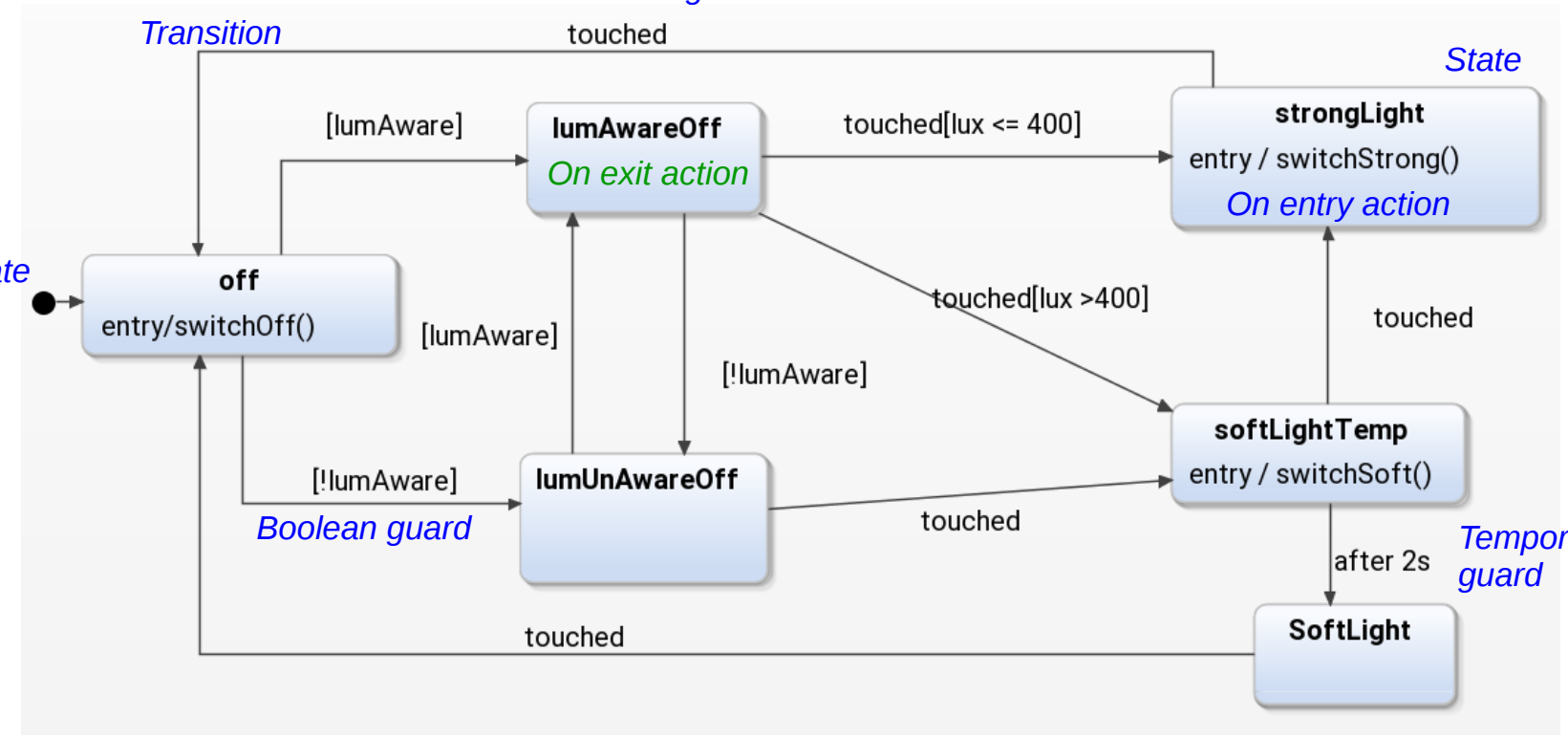
FSM

- C'est une abstraction permettant de structurer du code de contrôle.



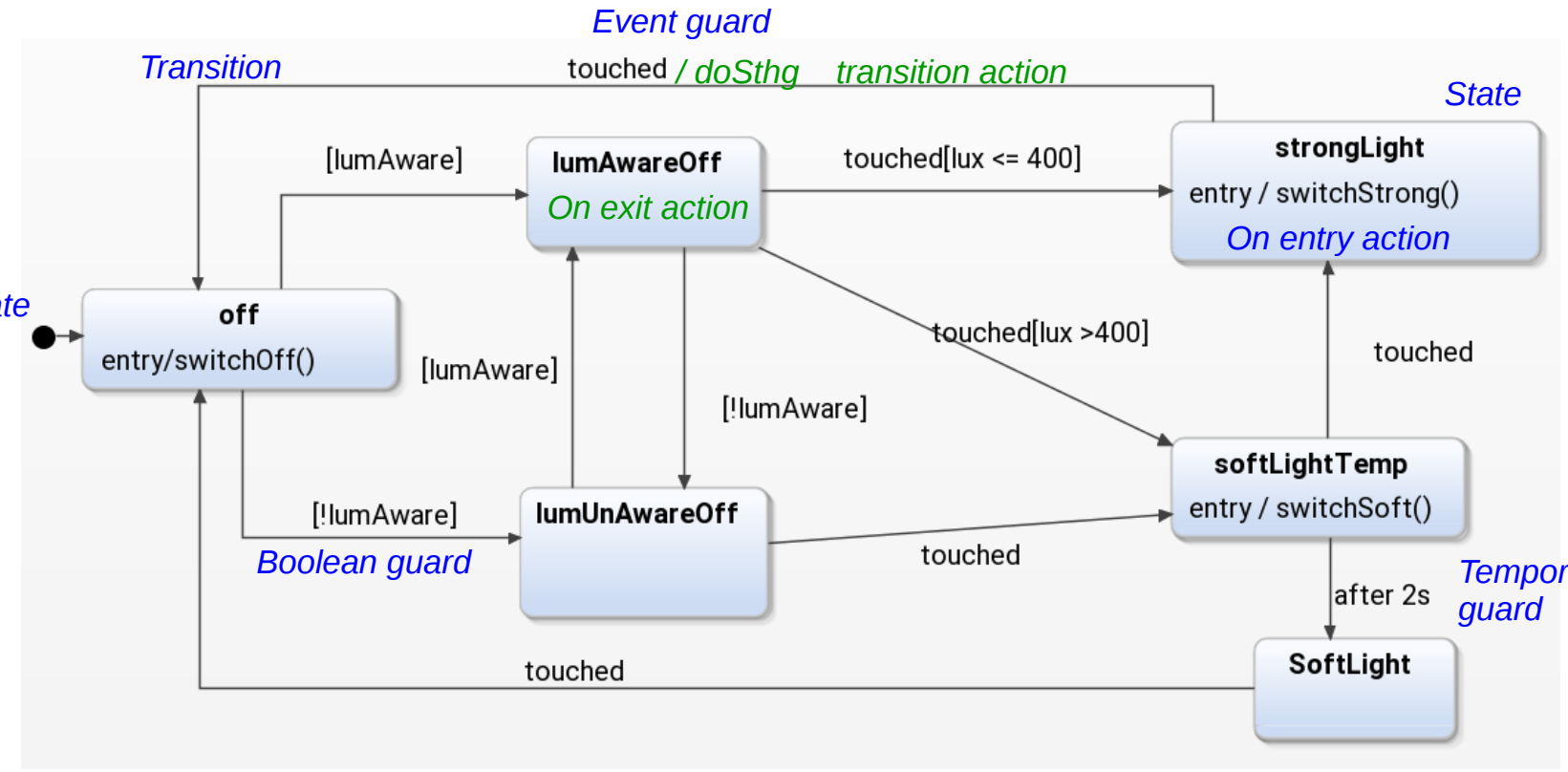
FSM

- C'est une abstraction permettant de structurer du code de contrôle.



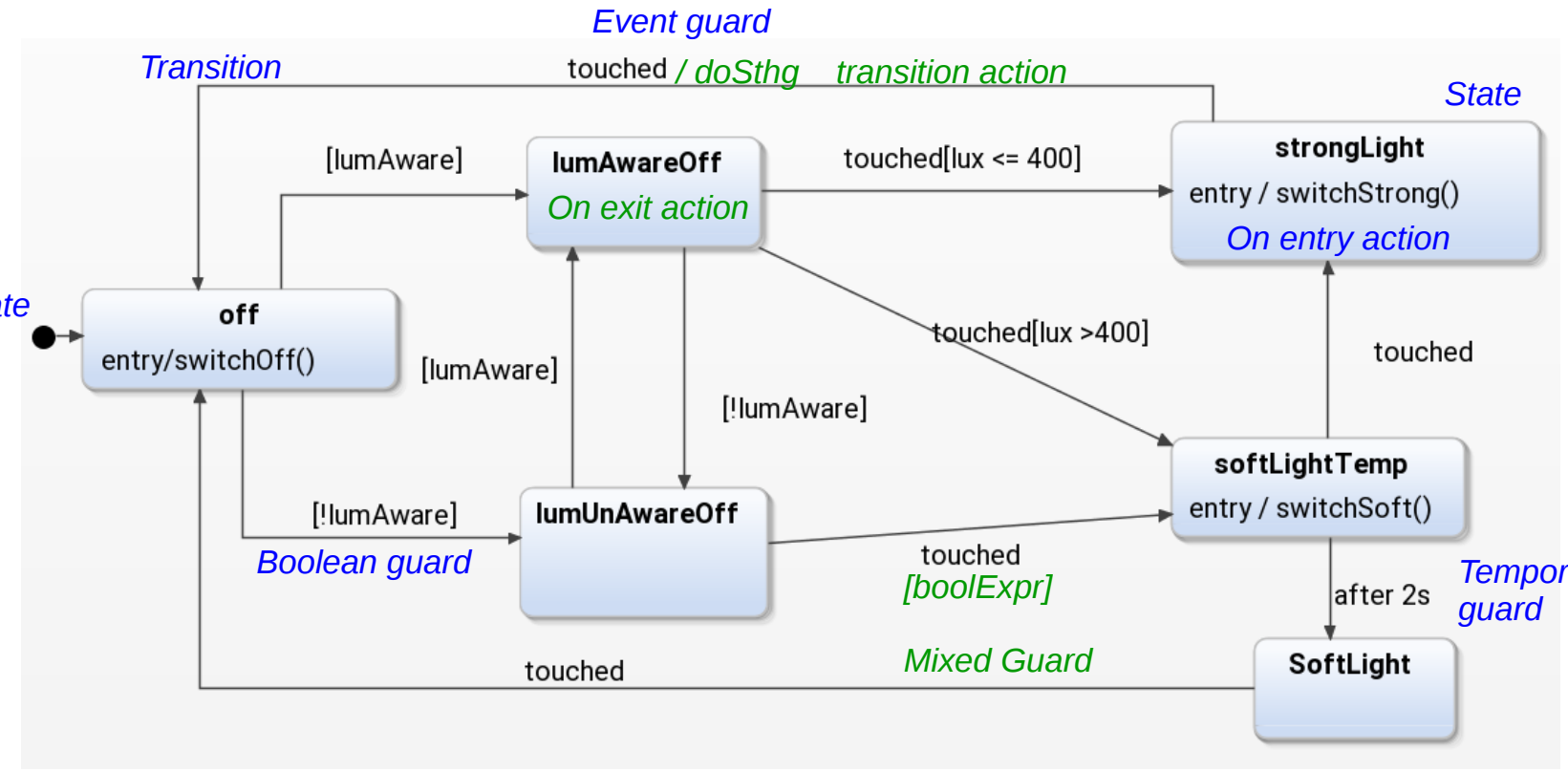
FSM

- C'est une abstraction permettant de structurer du code de contrôle.



FSM

- C'est une abstraction permettant de structurer du code de contrôle.



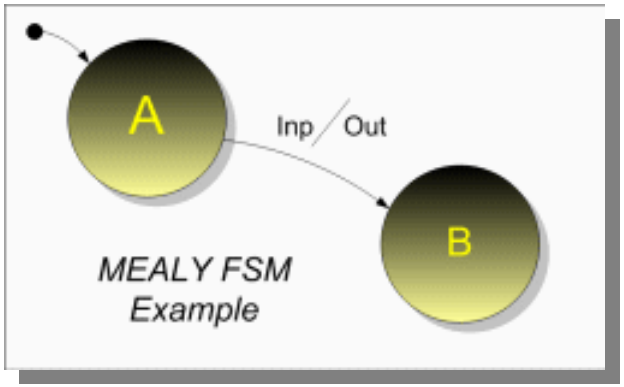
Historique court

The first people to consider the concept of a finite-state machine shared a common interest: to model the human thought process, whether in the brain or in a computer.

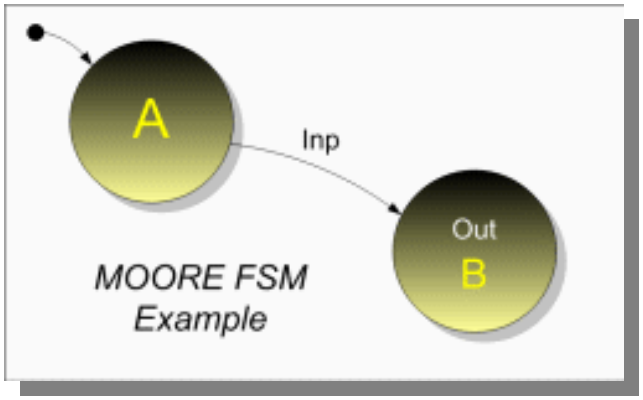
Warren McCulloch and Walter Pitts, two neurophysiologists, were the first to present a description of finite automata in 1943. Their paper, entitled, "A Logical Calculus Immanent in Nervous Activity", made significant contributions to the study of neural network theory, theory of automata, the theory of computation and cybernetics.

http://www-cs-faculty.stanford.edu/~eroberts/courses/soco/projects/2004-05/automata-theory/basics.html

Later, two computer scientists, G.H. Mealy and E.F. Moore, generalized the theory to much more powerful machines in separate papers, published in 1955-56. The finite-state machines, the Mealy machine and the Moore machine, are named in recognition of their work.



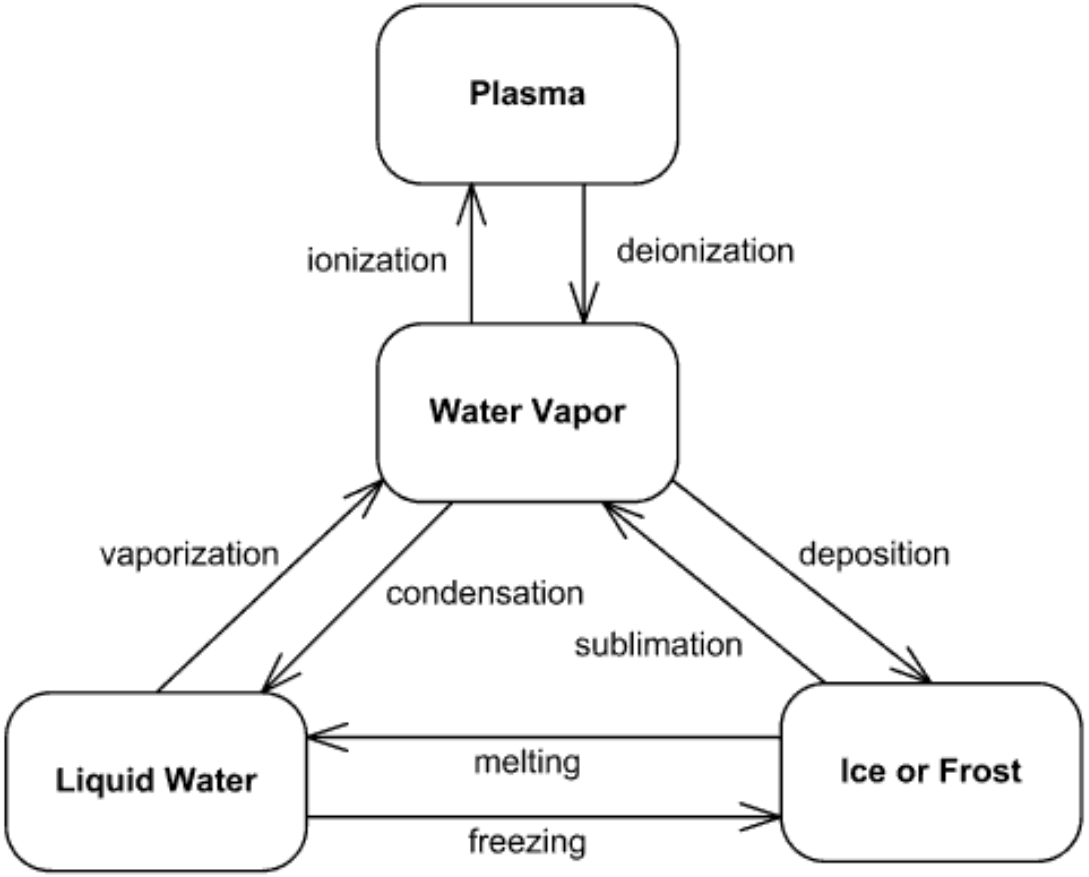
Mealy machine determines its outputs through the current state and the input



the Moore machine's output is based upon the current state alone.

Idée intuitive

- Les états du système, par exemple ceux de l'eau
- Des transitions avec des conditions entre les états



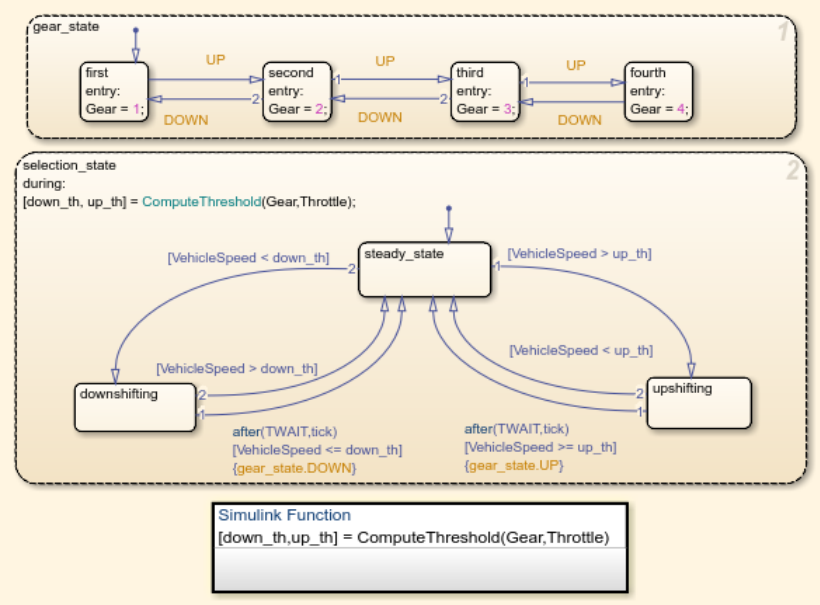
Domaine d'utilisation (en info)

- Pour ainsi dire, tous !
  - **Embedded systems (depuis toujours)**
    - Stateflow (mathworks),
  - **IHM et Web depuis plusieurs années**
    - SCXML (W3C)
  - **Réseau de communication**
    - SDL
  - **COO**:
    - Harel state charts dans UML
  - **Parser, reconnaisseur de langage, voir LFA**
  - **CPS: automate hybride**
  - …

Domaine d'utilisation (en info)

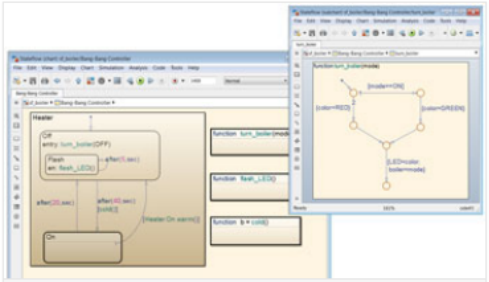
Embedded systems

- Stateflow (mathworks)



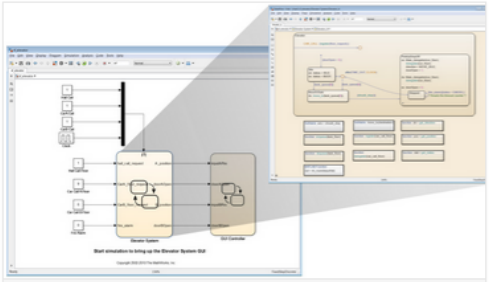
Solution commerciale professionelle pour la conception de systèmes embarqués

Fonctionnalités

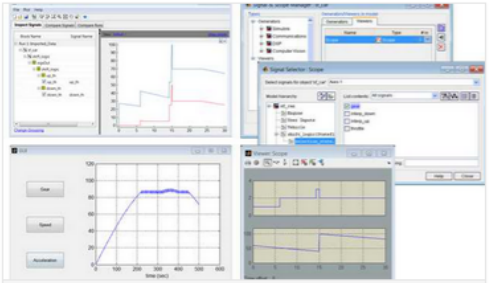


Conception de logique

Modélisez une logique de système à l'aide de machines à états.

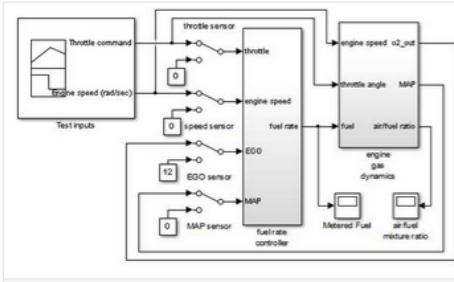


Intégration de composants et algorithmes de planification



Simulation d'un modèle et analyse des résultats

Analysez le comportement de



Validation de la conception et génération de code

Domaine d'utilisation (en info)

Embedded systems

- SMCube (Scicos)

SMCube is a tool for modeling, simulation, and code generation of discrete time state machines.

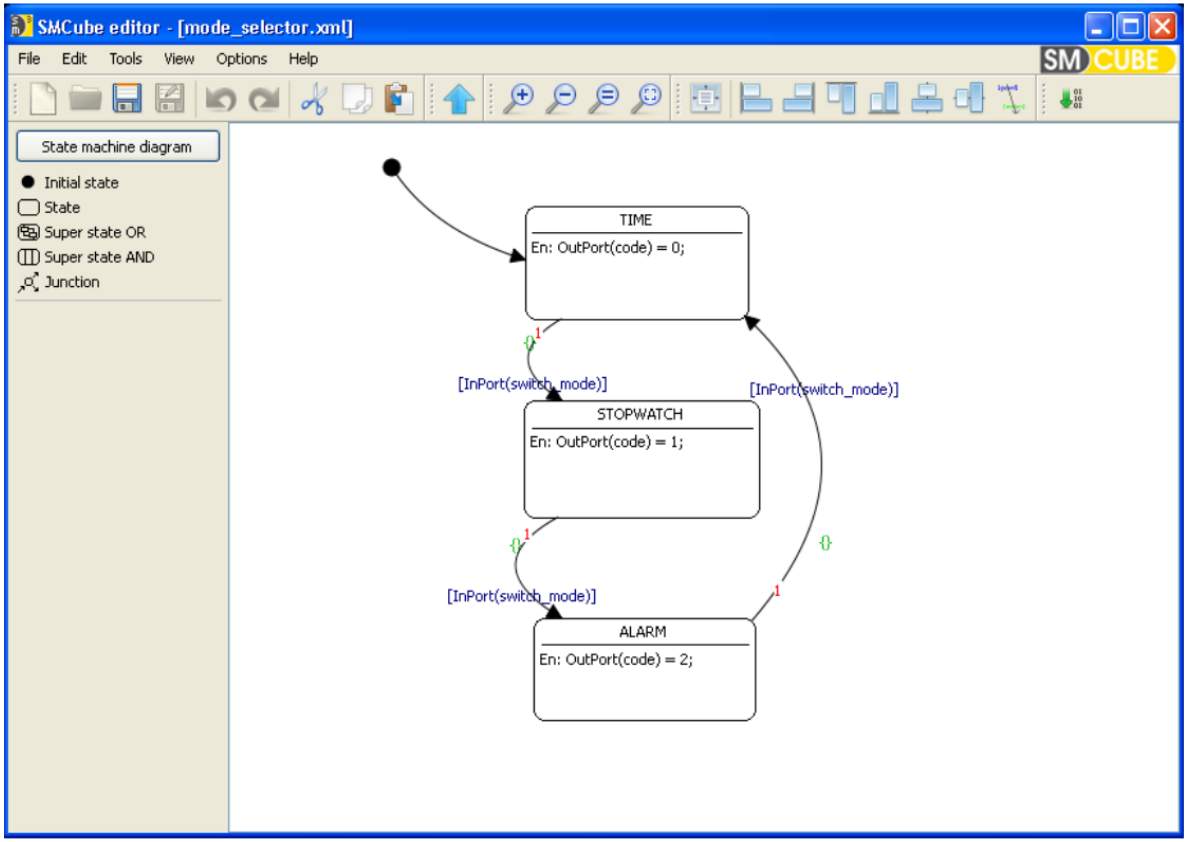


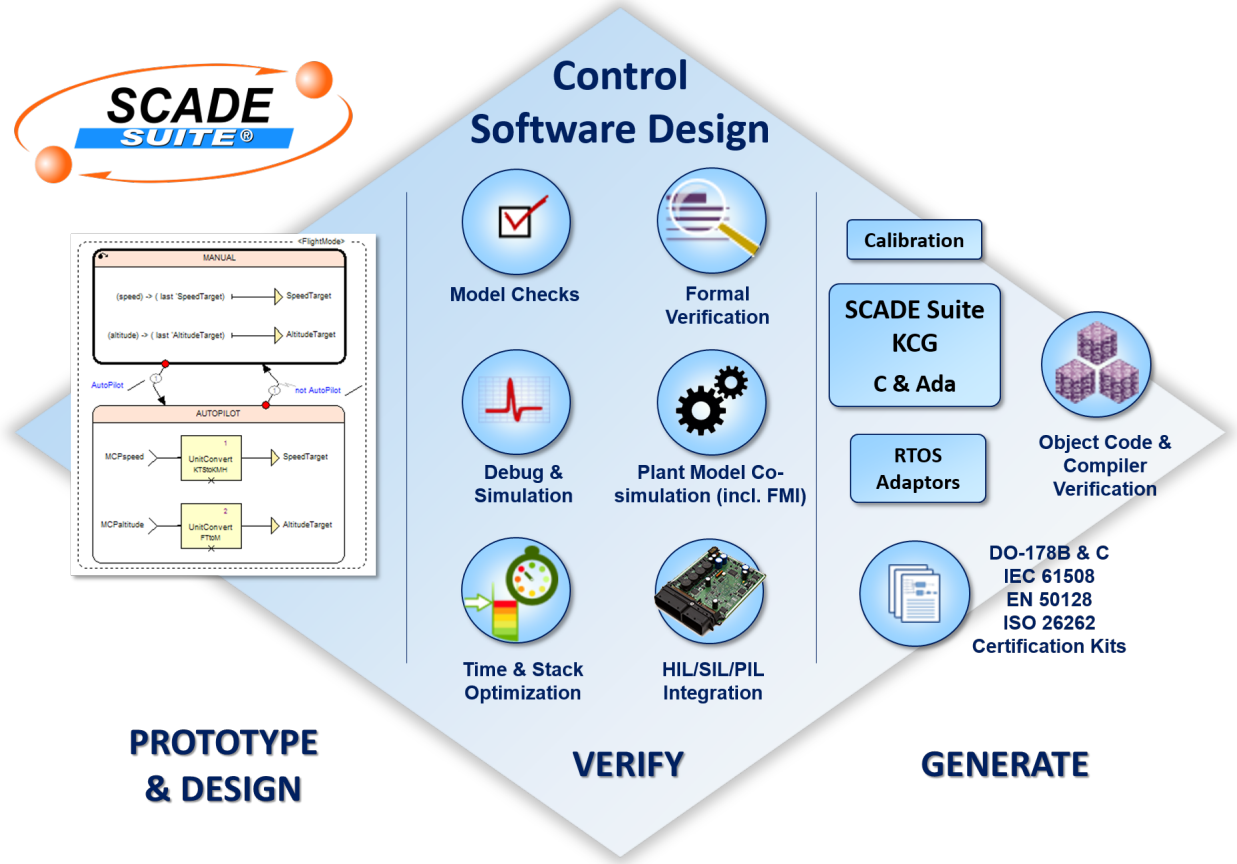
Figure: an example of a SMCube state machine. This state machine is used to model the various states of a Digital Clock.

Solution commerciale professionelle pour
la conception de systèmes embarqués
(sémantique synchrone)

Domaine d'utilisation (en info)

Embedded systems

- Scade suite (Ansys (ex Esterel))

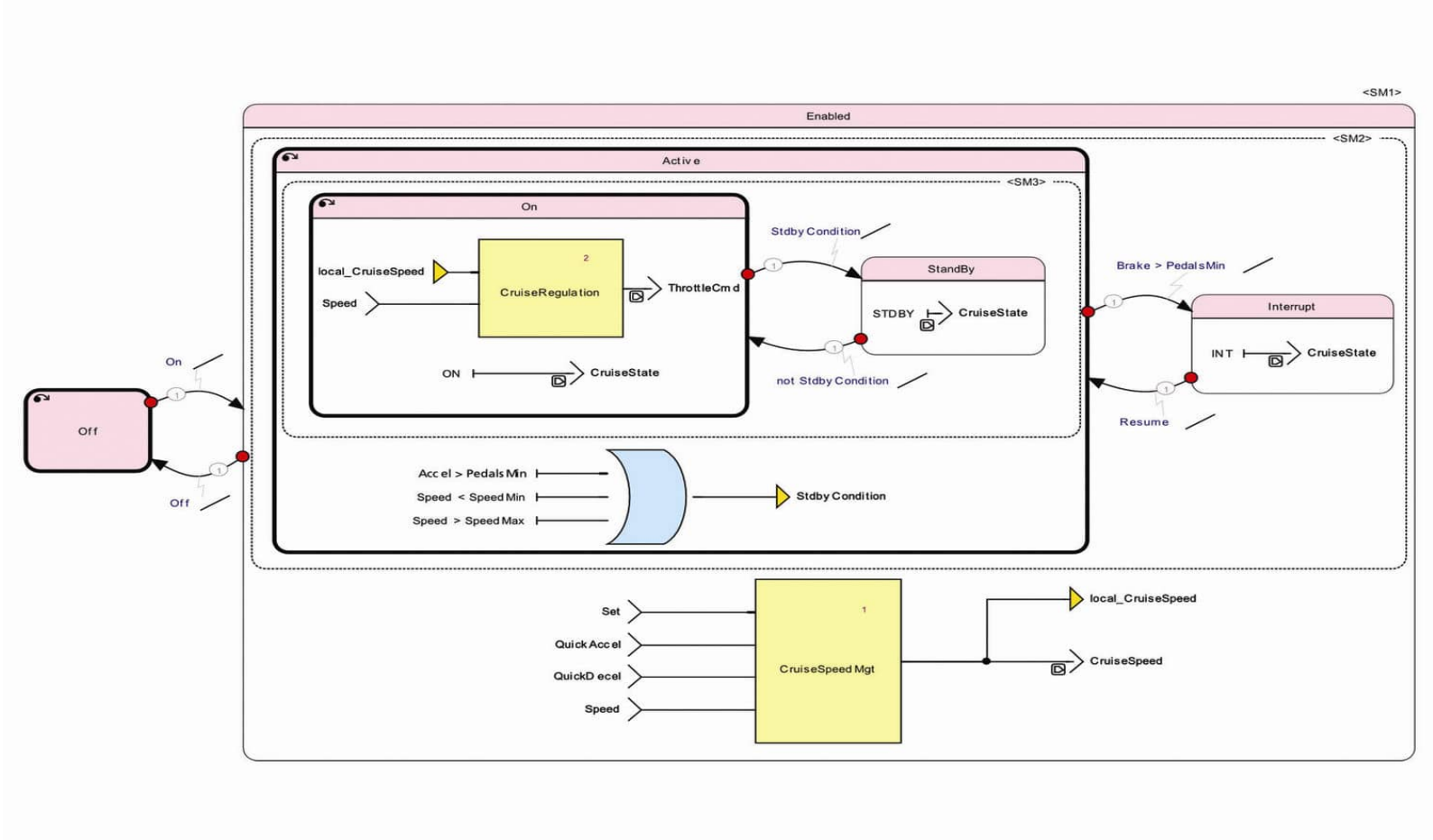


Solution commerciale professionelle pour la conception de systèmes embarqués CRITIQUES

Domaine d'utilisation (en info)

Embedded systems

- Scade suite (Esterel)

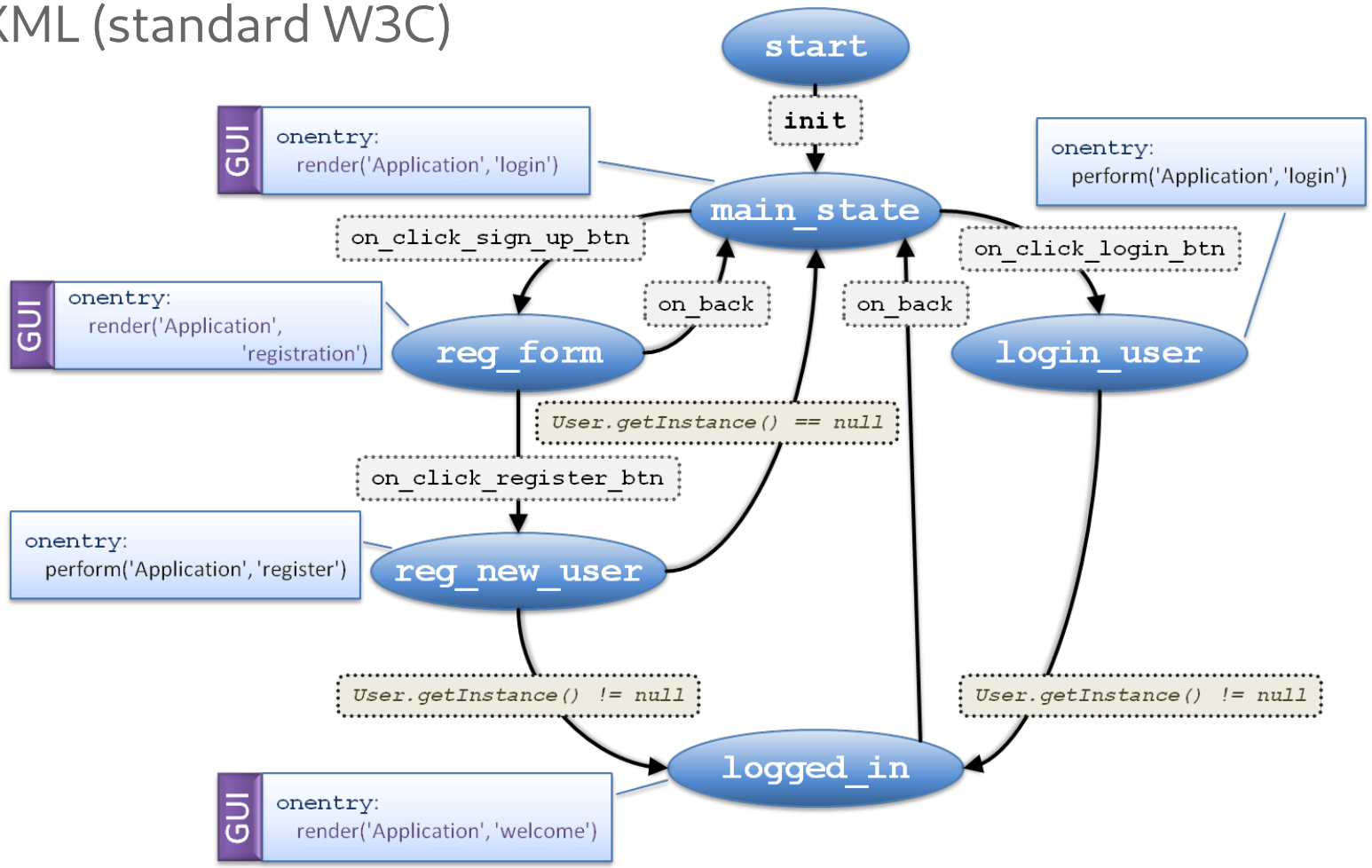


Solution commerciale professionelle pour
la conception de systèmes embarqués
CRITIQUES

Domaine d'utilisation (en info)

IHM et Web

- SCXML (standard W3C)

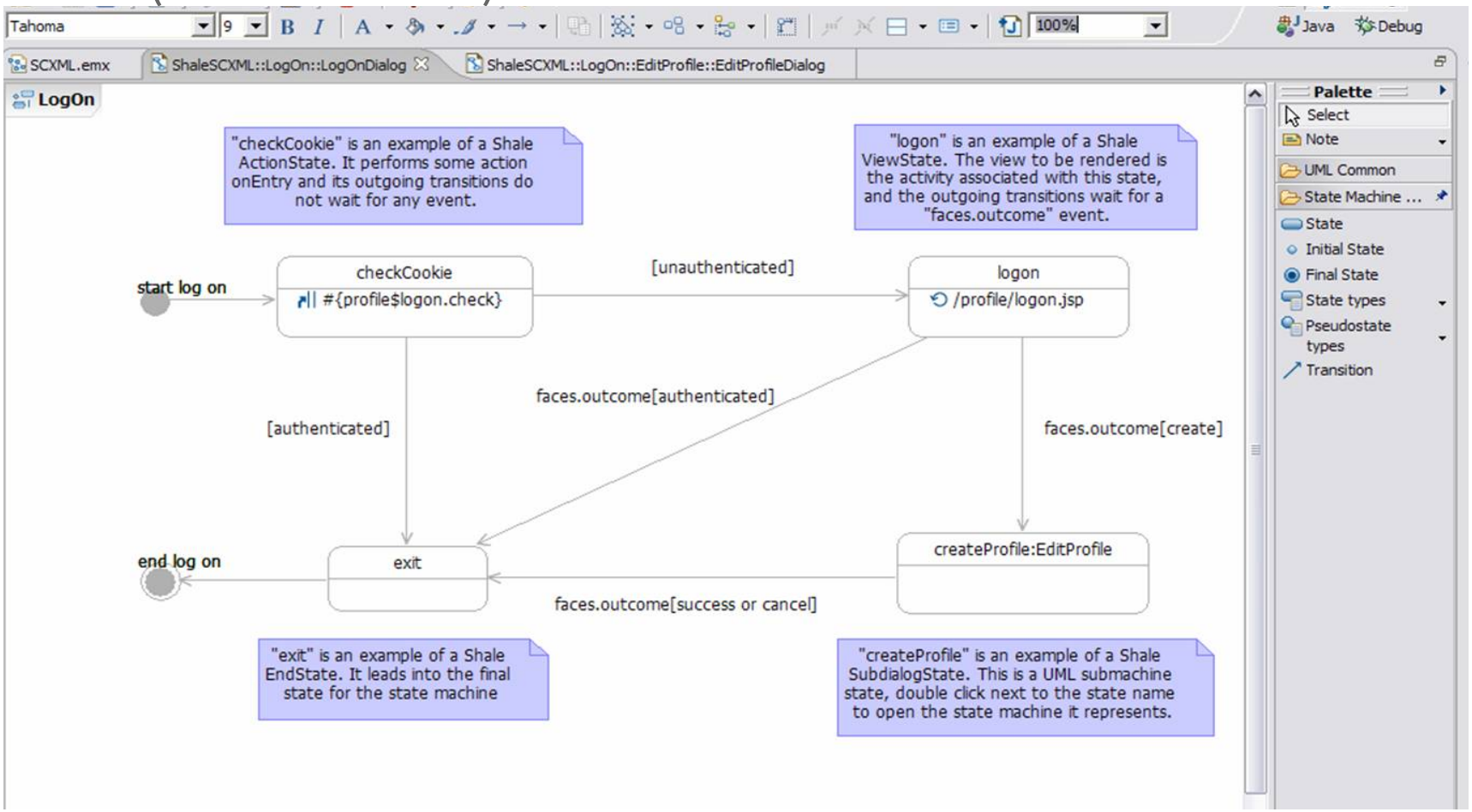


Multimodal Mobile Interaction and Rendering (MMIR) defines the dialog flow of an application by specifying transitions between application states using SCXML

Domaine d'utilisation (en info)

IHM et Web

- SCXML (standard W3C)



Shale Dialog Manager (SCXML Implementation) […] defines a generic API by which an application may utilize a Dialog Manager implementation to manage conversations with the user of that application […] This module contains the SCXML (State Chart XML) Implementation of the Shale Dialog Manager facilities

Domaine d'utilisation (en info)

IHM et Web

- SCXML (standard W3C)

Developing User Interfaces using SCXML Statecharts

Gavin Kistner
NVIDIA, Inc.
1350 Pine St.
Boulder, CO
gkistner@nvidia.com

Chris Nuernberger
NVIDIA, Inc.
1350 Pine St.
Boulder, CO
chrisn@nvidia.com

ABSTRACT

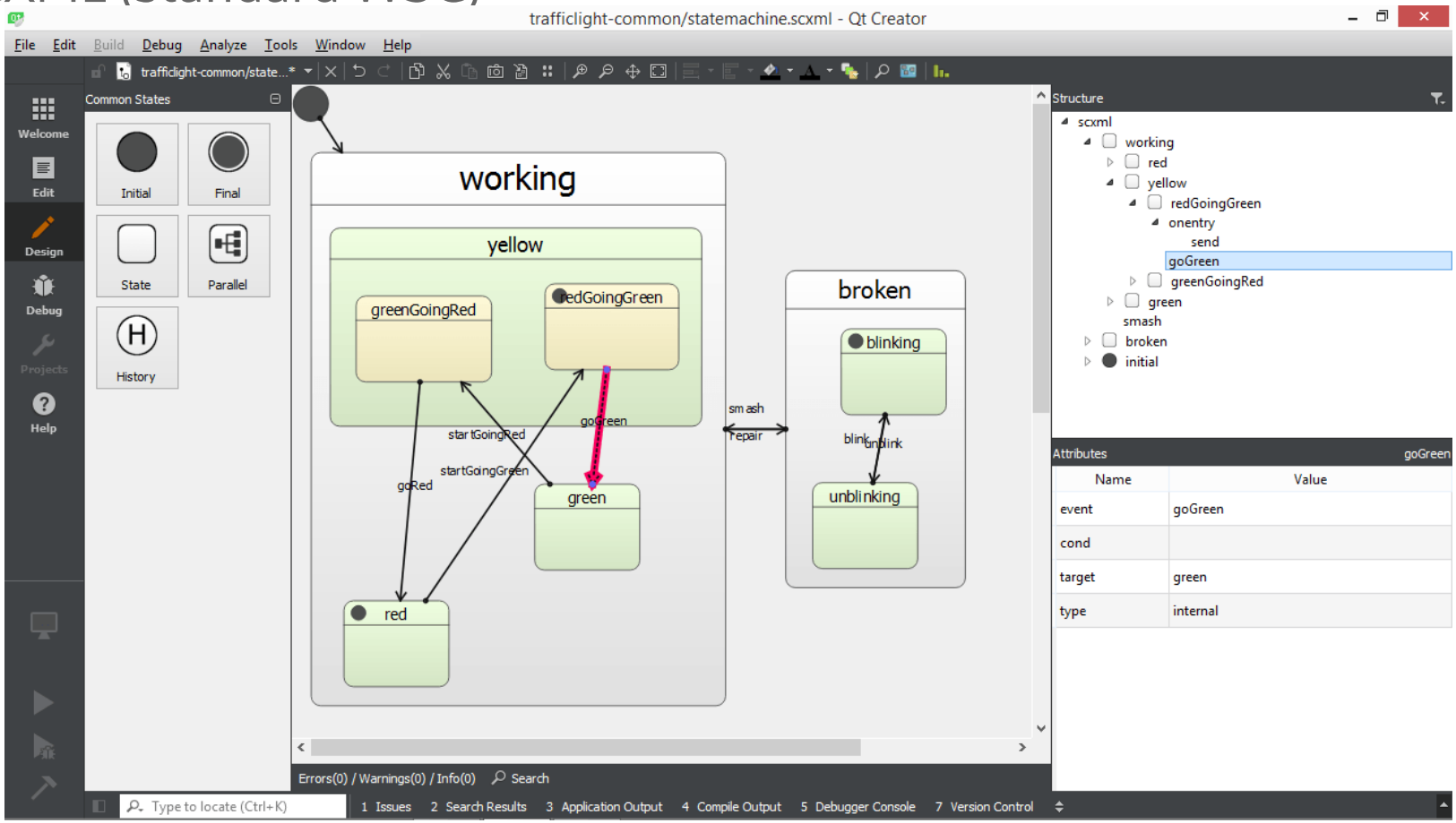
In this paper we describe NVIDIA Corporation's implementation of an editor and runtime for the SCXML statechart standard. The editor and runtime are used for both prototyping and production of user interfaces, targeted primarily for automotive in-vehicle interfaces. We show how state machines improve the simplicity and stability of application development, particularly when using the hierarchical and parallel states available in SCXML. We investigate the usefulness of statecharts in user interaction design. We further describe subtle additions and deviations from the SCXML standard, the motivations for these changes, and their benefits compared to a strictly standards-compliant implementation.

http://phrogz.net/developing-user-interfaces-using-scxml-statecharts

Domaine d'utilisation (en info)

IHM et Web

- SCXML (standard W3C)



QtCreator SCXML editor available in QT 5.8. The Qt SCXML module provides classes for embedding state machines created from State Chart XML (SCXML) files in Qt applications. […] Parts of the application logic can be replaced with an encapsulated SCXML file. This enables creating a clear division between the application logic and the user interface implementation by using Qt Quick or Qt Widgets

Domaine d'utilisation (en info)

IHM et Web

- Javascript Finite State Machine

```
var fsm = StateMachine.create({
  initial: 'first', final: 'fourth',
  events: [
    { name: 'hop',  from: 'first',  to: 'second' },
    { name: 'skip', from: 'second', to: 'third'  },
    { name: 'jump', from: 'third',  to: 'fourth' },
  ]
});

fsm.isFinished(); // false

fsm.hop();
fsm.isFinished(); // false

fsm.skip();
fsm.isFinished(); // false

fsm.jump();
fsm.isFinished(); // true
```

A standalone library for finite state machines.
https://github.com/jakesgordon/javascript-state-machine/

Domaine d'utilisation (en info)

IHM et Web

- Machina.js

```
states: {
    uninitialized: {
        // Input handlers are usually functions. They can
        // take arguments, too (even though this one doesn
        // The "*" handler is special (more on that in a b
        "*": function() {
            this.deferUntilTransition();
            // the `transition` method takes a target state
            // and transitions to it. You should NEVER dire
            // state property on an FSM. Also - while it's
            // call `transition` externally, you usually er
            // cleanest approach if you endeavor to transi
            // and just pass input to the FSM.
            this.transition( "green" );
        }
    },
```

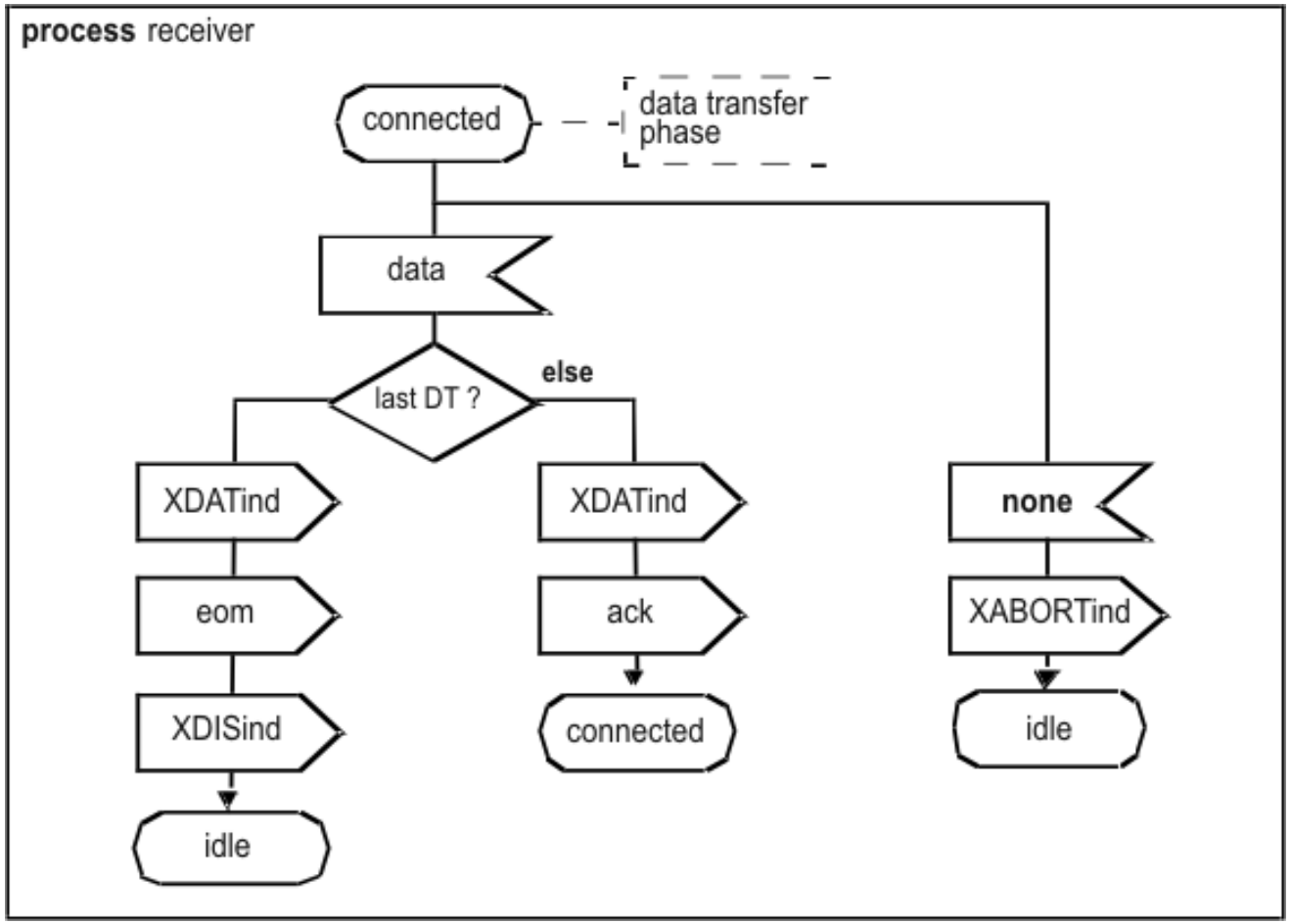
```
    green: {
        // _onEnter is a special handler that is invoked
        // immediately as the FSM transitions into the new state
        _onEnter: function() {
            this.timer = setTimeout( function() {
                this.handle( "timeout" );
            }.bind( this ), 30000 );
            this.emit( "vehicles", { status: GREEN } );
        },
        // If all you need to do is transition to a new state
        // inside an input handler, you can provide the string
        // name of the state in place of the input handler function.
        timeout: "green-interruptible",
        pedestrianWaiting: function() {
            this.deferUntilTransition( "green-interruptible" );
        },
        // _onExit is a special handler that is invoked just before
        // the FSM leaves the current state and transitions to another
        _onExit: function() {
            clearTimeout( this.timer );
        }
    },
```

Machina.js is a JavaScript framework for highly customizable finite state machines (FSMs)
http://machina-js.org/

Domaine d'utilisation (en info)

réseau

- SDL Specification Description Language

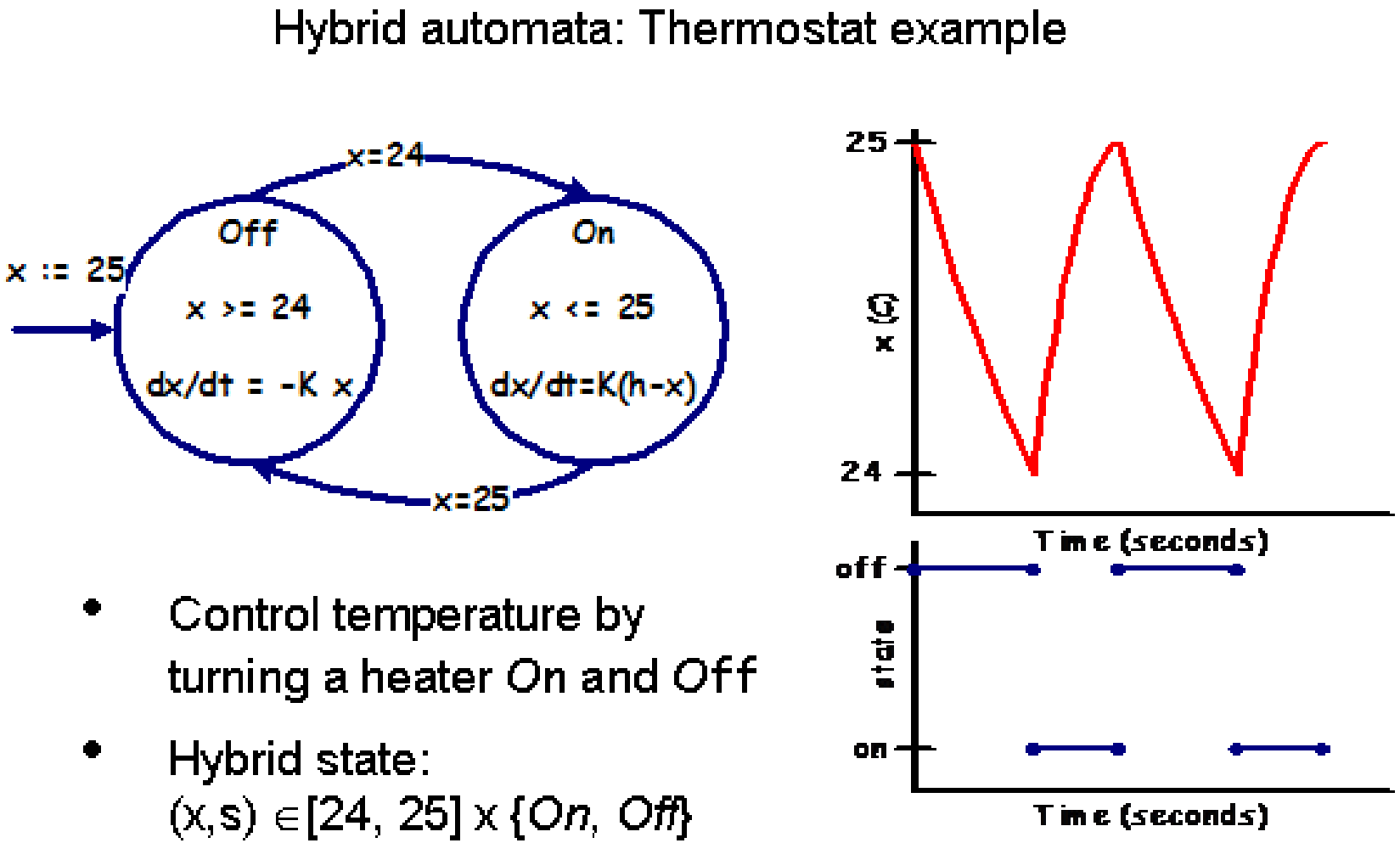


Specification and Description Language (SDL) is a specification language targeted at the unambiguous specification and description of the behaviour of reactive and distributed systems. The ITU-T has defined SDL in Recommendations Z.100 to Z.106. SDL originally focused on telecommunication systems

Domaine d'utilisation (en info)

Cyber-Physical System

- Hybrid automaton



- Control temperature by turning a heater *On* and *Off*
- Hybrid state:
  $(x,s) \in [24, 25] \times \{On, Off\}$

http://paginas.fe.up.pt/~jtasso/HS10.htm

Domaine d'utilisation (en info)

V&V

- Fiacre

```
process sender [inp: data, outp: msg,
      timeout, ack: none] (ssn: bool) is
  states idle, send, wait, resend
  var data: data
  from idle inp? data; to send
  from send
    outp! ssn,data; ssn := not ssn; to wait
  from wait
    select ack; to idle
     []    timeout; to resend
    end
  from resend outp! ssn,data; to wait
```

Academic format for formal verification

Des variantes /dialectes

- Mealy, moore,
- temporisés ou non,
- Temps continus, temps discret
- Guardes booléennes ou non,
- Hierarchiques,
- Communicants
- …

SCXML et UML couvre une grosse combinatoire des possibilités

Les variations peuvent être syntaxique mais aussi sémantiques… influant sur l'expressivité et la facilité de vérification

SCXML

State Chart XML (SCXML): State Machine Notation for Control Abstraction

W3C Recommendation 1 September 2015

https://www.w3.org/TR/scxml/

- Standard W3C récent aligné sur les state machines UML.
- Syntaxe XML
- Parser + simulateur
  - Java chez Apache
  - C++ chez Qt
  - Python : pyscxml

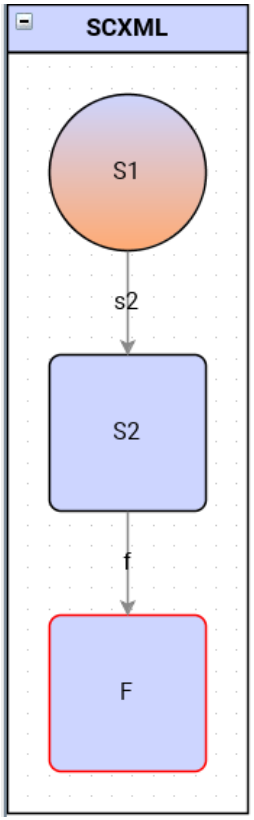
SCXML

State Chart XML (SCXML): State Machine Notation for Control Abstraction

W3C Recommendation 1 September 2015

https://www.w3.org/TR/scxml/

- Parser + simulateur
  - Python : pyscxml




```
from scxml.pyscxml import StateMachine
import logging
logging.basicConfig(level=logging.NOTSET)

sm = StateMachine("fsm1.scxml")
sm.start_threaded()
sm.send("s2")
sm.send("f")
```

```
hello S1
INFO:pyscxml.pyscxml_session_139883684041040.interpreter:external event found: s2
bye S1
transition s2 from S1 to S2
hello S2
INFO:pyscxml.pyscxml_session_139883684041040.interpreter:new config: {S2}
INFO:pyscxml.pyscxml_session_139883684041040.interpreter:external event found: f
bye S2
transition f from S2 to F
hello F
INFO:pyscxml.pyscxml_session_139883684041040.interpreter:new config: {F}
INFO:pyscxml.pyscxml_session_139883684041040.interpreter:Exiting interpreter
DEBUG:pyscxml.multisession:The session 'pyscxml_session_139883684041040' finished
```

SCXML

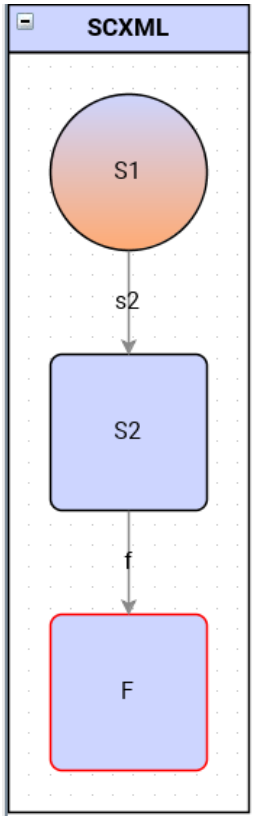
State Chart XML (SCXML): State Machine Notation for Control Abstraction

W3C Recommendation 1 September 2015

https://www.w3.org/TR/scxml/

- Viewer or text

```
<?xml version="1.0" encoding="UTF-8"?>
<scxml xmlns="http://www.w3.org/2005/07/scxml" version="1.0" datamodel="python" initial="S1">
    <state id="S1">
        <onentry>
           <log expr="'hello S1'"/>
        </onentry>
        <transition event="s2" target="S2">
           <log expr="'transition s2 from S1 to S2'" />
        </transition>
        <onexit>
           <log expr="'bye S1'"/>
        </onexit>
    </state>
    <state id="S2">
        <onentry>
           <log expr="'hello S2'"/>
        </onentry>
        <transition event="f" target="F">
           <log expr="'transition f from S2 to F'" />
        </transition>
        <onexit>
           <log expr="'bye S2'"/>
        </onexit>
    </state>
    <final id="F">
        <onentry>
           <log expr="'hello F'"/>
        </onentry>
    </final>
</scxml>
```



scxmlgui