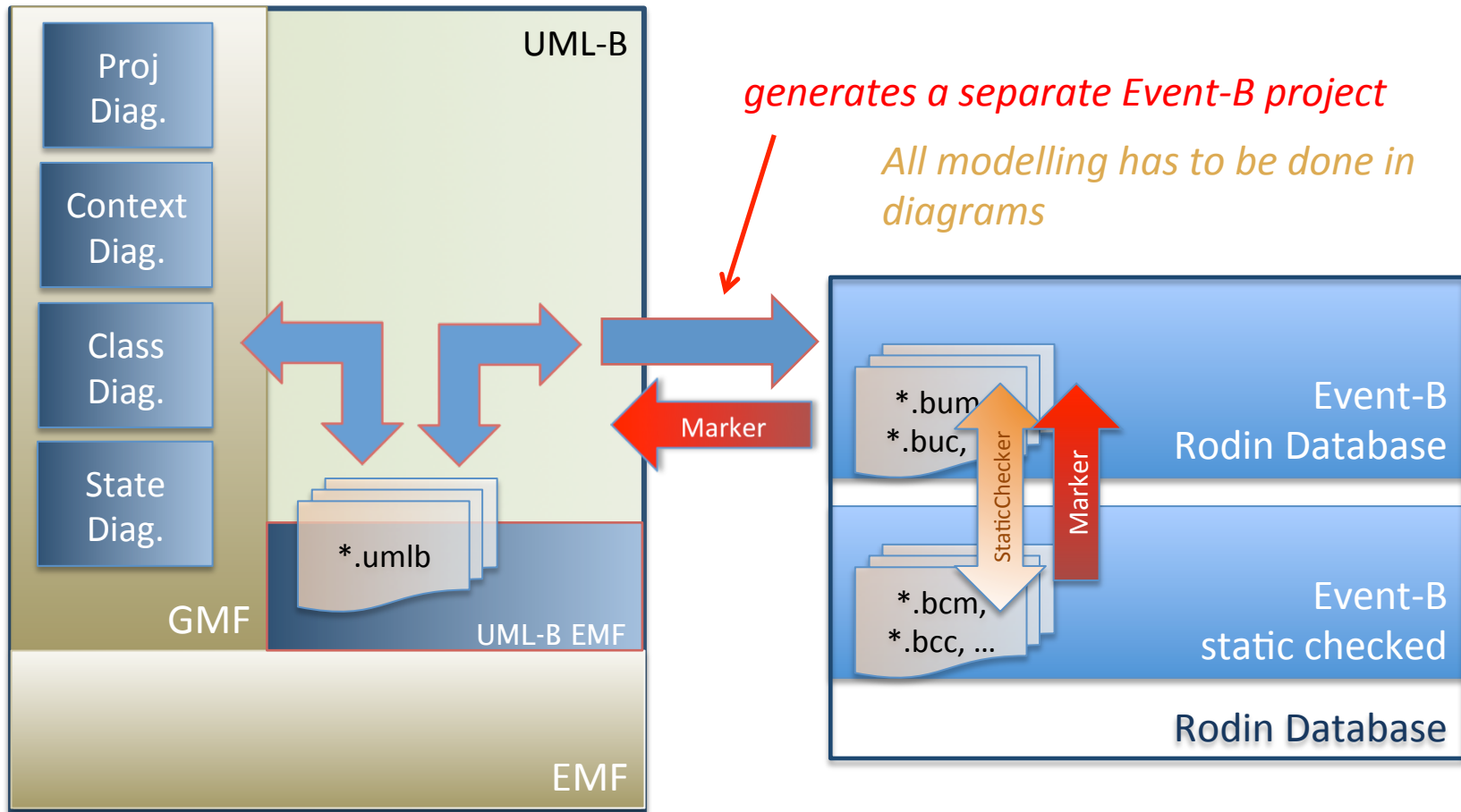


iUML-B Statemachines, a new approach to UML-B

(old) UML-B Architecture



Motivation for (new) iUML-B

Experienced Event-B users

would like diagrams

without being distanced from Event-B

Integrated UI

Mix notations within one machine

Event-B Text editor +

Class diagrams +

State machine diagrams +

... and others

Solution:

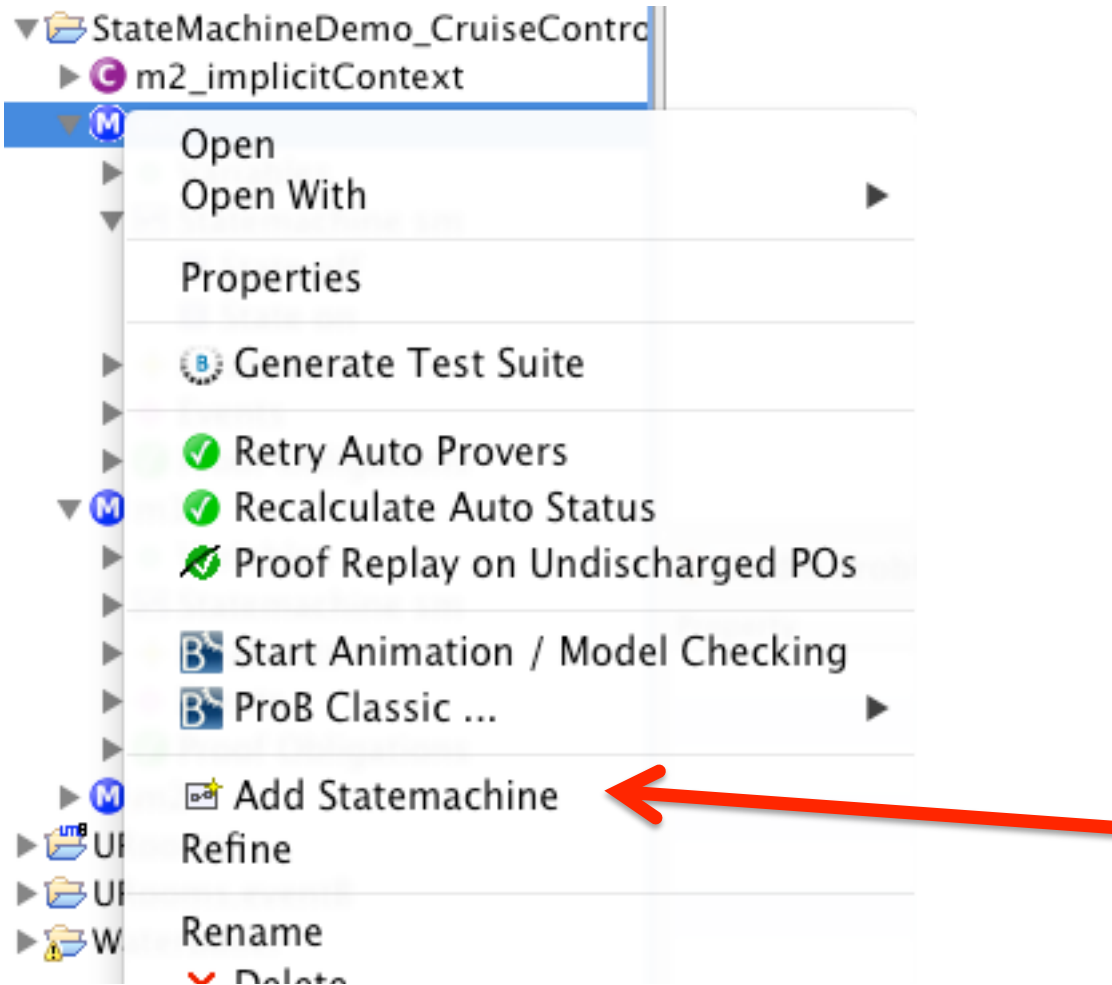
Re-implement UML-B as an extension to Event-B

e.g. a state machine is an element inside an Event-B machine

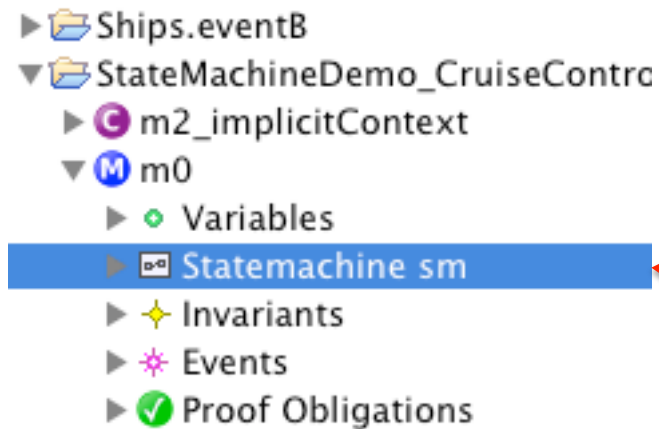
Contributes guards and actions to hand written events

iUML-B Tool demo

Adding a statemachine to an Event-B machine

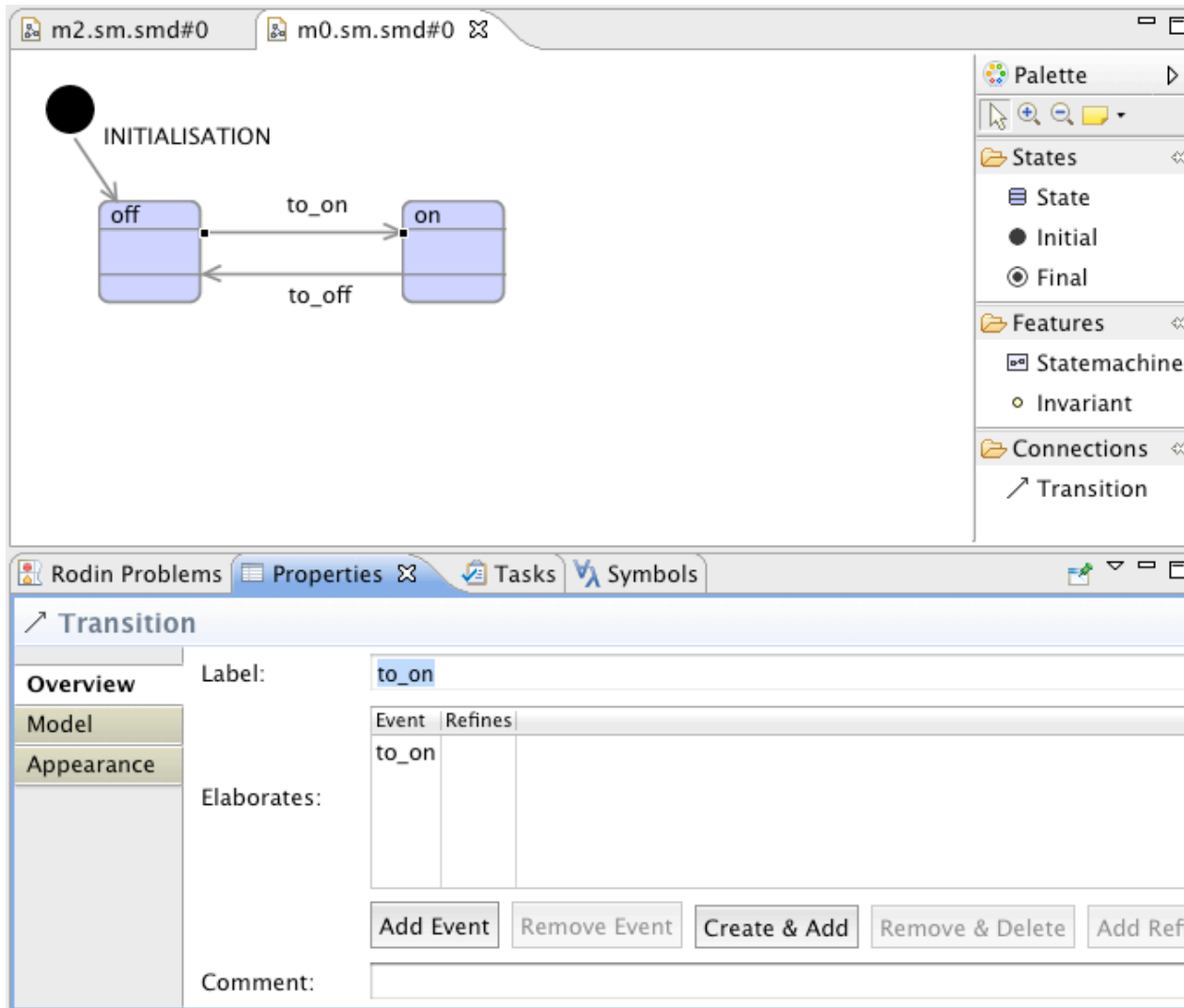


Open state machine diagram editor



Double click to open

Editor i/f

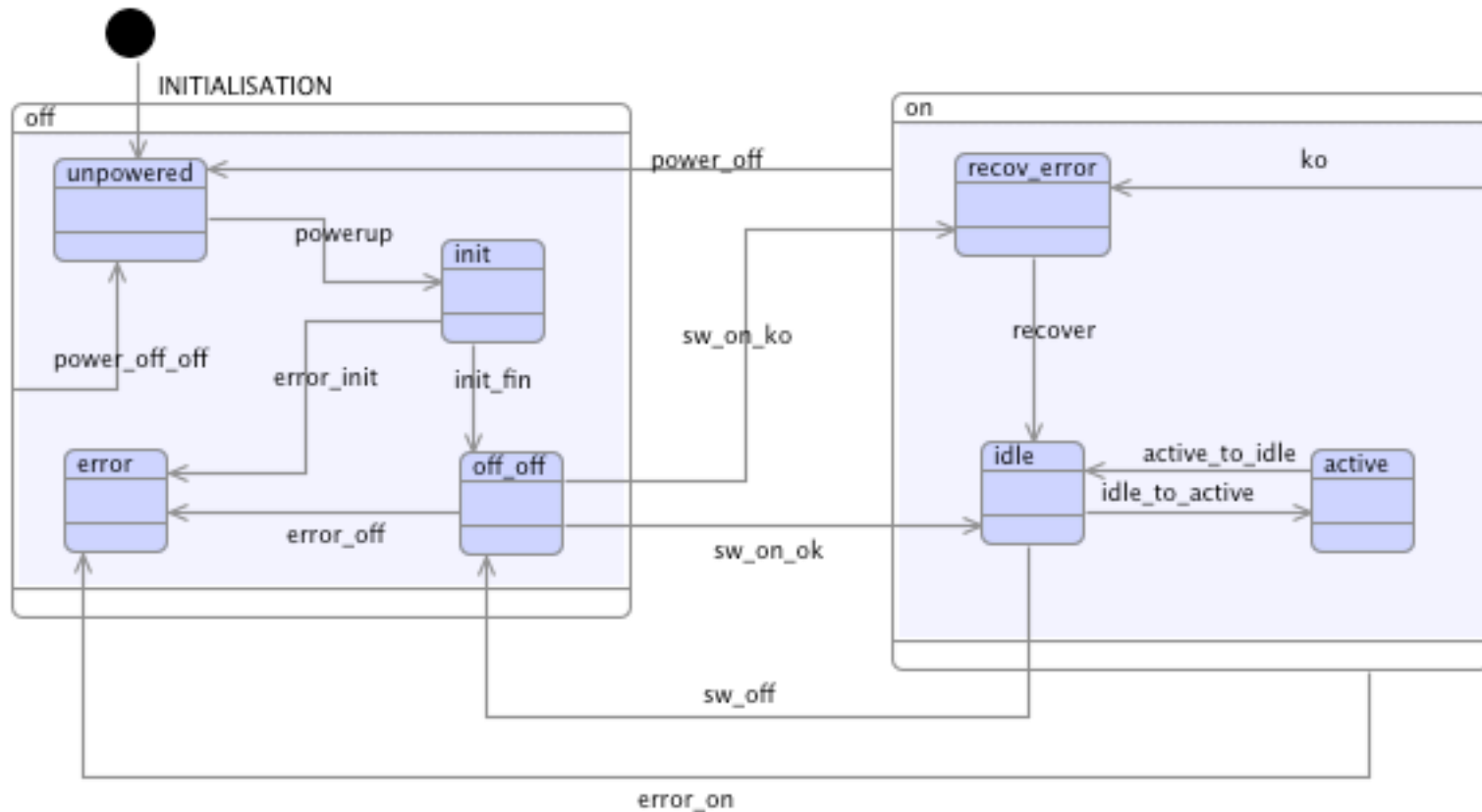


Draw diagram,

Link to existing events using properties sheet
add event button

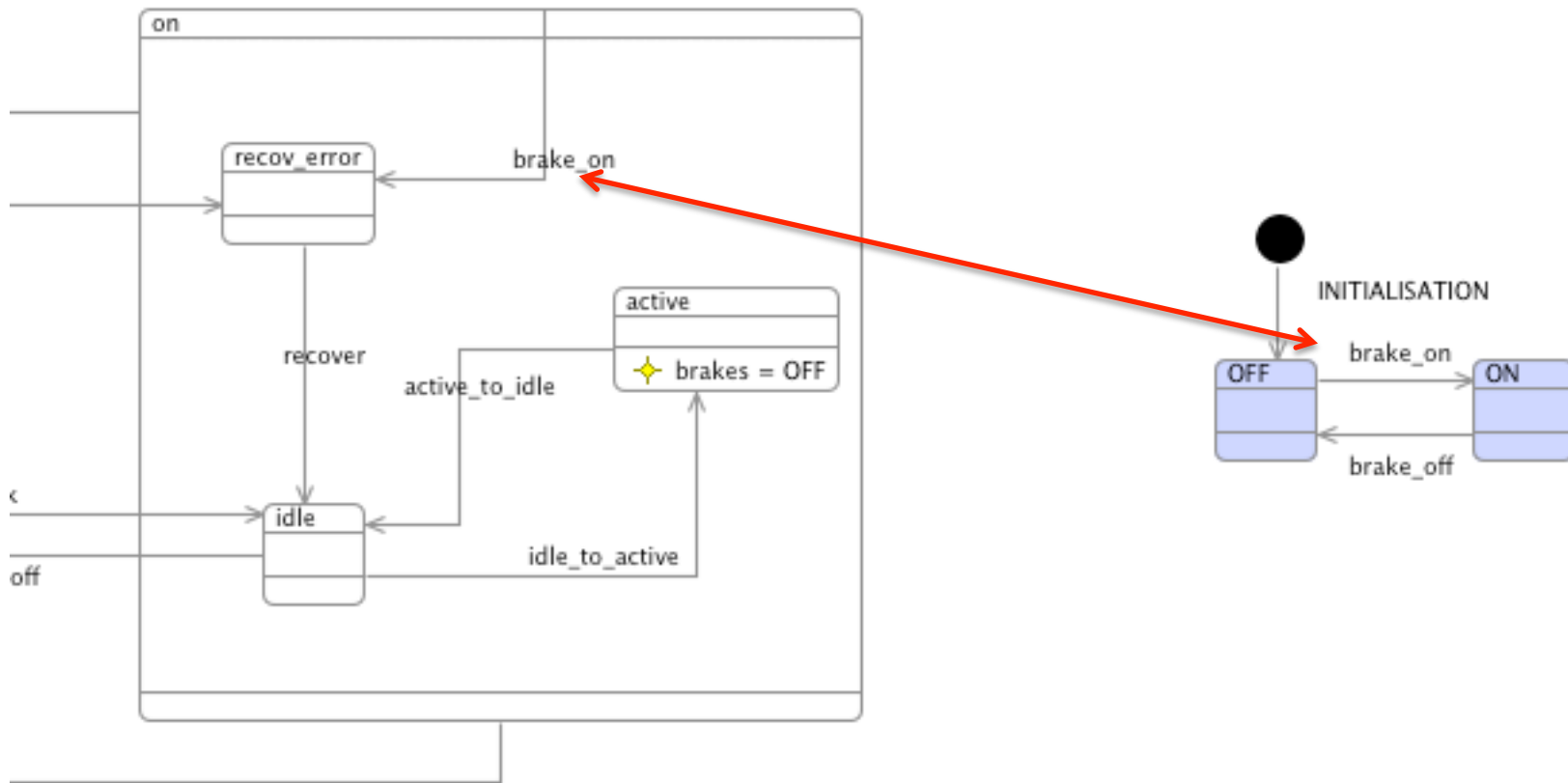
Create new events and and link to them using create & add button

Refinement – add nested statemachines



ko represents the occurrence of some conditions where the cruise control shouldn't be active (or idle?)

Synchronise state machines via event elaboration



Animation/model checker

*m2.sm.smd#0

m0.sm.smd#0

m1.sm.smd#0

*m2.brakes.smd#0

INITIALISATION

```
stateDiagram-v2
    state off {
        [*] --> unpowered : powerup
        unpowered --> off : power_off
        unpowered --> error : error_init
        error --> off : error_off
    }
    state on {
        state recov_error {
            --> idle : recover
        }
        state idle {
            --> active : idle_to_active
            active --> idle : active_to_idle
        }
        state active {
            invariant brakes = OFF
        }
        recov_error --> idle : recover
        idle --> recov_error : brake_on
    }
    state init
    state off_off
    state error
    state error_off

    init --> off_off : init_fin
    off_off --> off : power_off
    off_off --> error : error_init
    error --> error_off : error_off
    error_off --> off : power_off
    error_off --> idle : sw_on_ok
    idle --> error_off : sw_off
    idle --> recov_error : sw_on_ko
    recov_error --> idle : recover
    active --> idle : active_to_idle
    idle --> active : idle_to_active
    error_on --> error
```

State

Ltl Counter-Example

Name

▼ m0

off

on

▼ ★ m1

★ active

error

★ idle

init

off_off

recov_error

unpowered

off

on

▼ ★ m2

brakes

★ active

error

★ idle

init

off_off

recov_error

unpowered

off

on

▼ Formulas

► ★ invariants

► guards

Invariant violated!

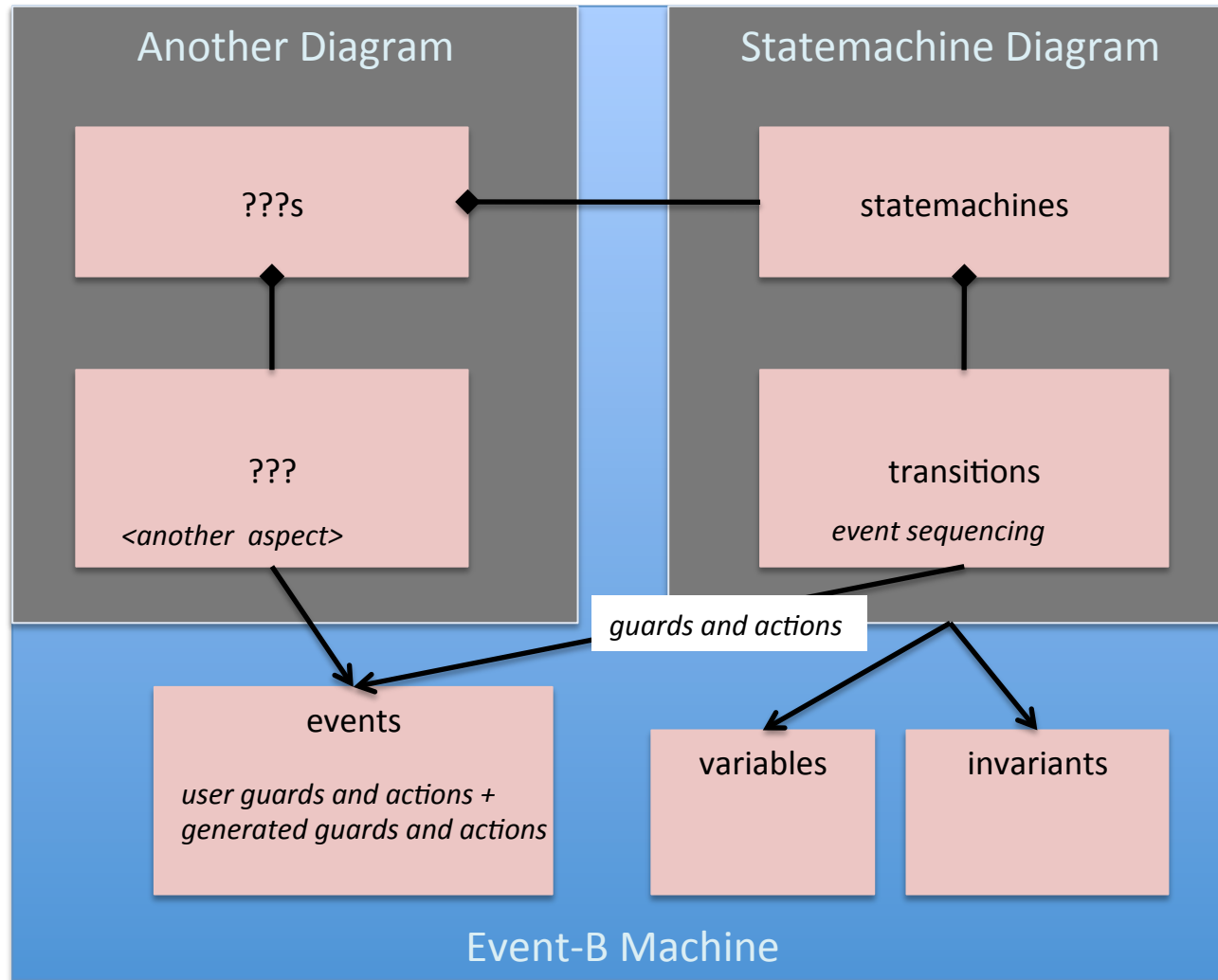
no event errors detected

Now we give the conditions for ko in more detail – it means the brakes have been pressed. (see brakes state machine).

We also add an invariant that the brakes are off if the cruise control is active.

Try animating the statemachine to find a path that breaks the invariant in state 'active' (you can also run the model checker to find a path automatically).. Then correct the model.

Multiple Diagrams



Architecture/persistence

