

Modelling IV

State Machines

Ketil Stølen

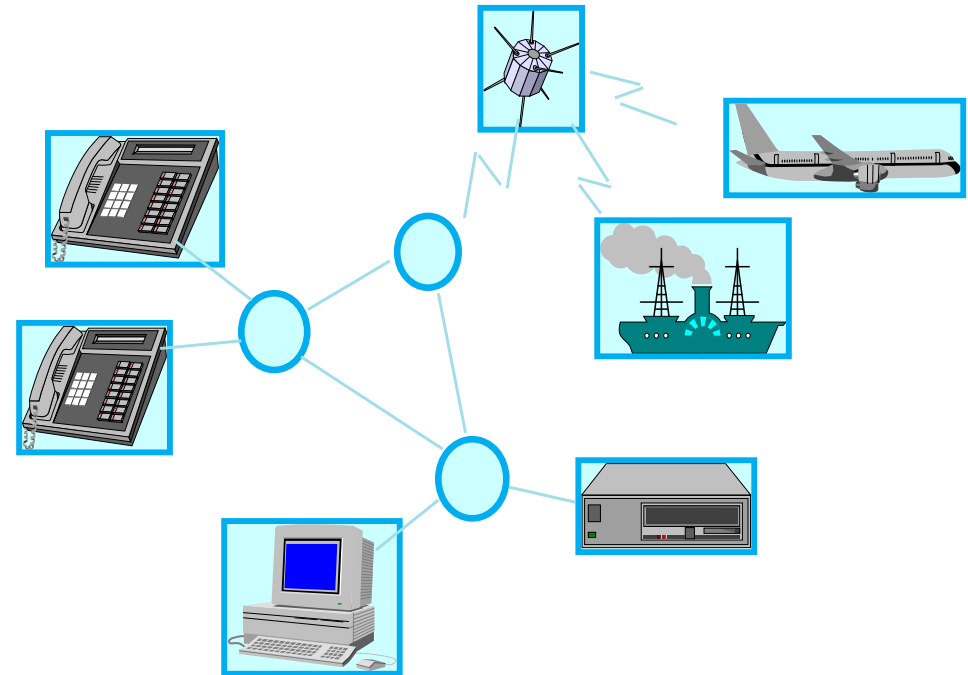
Based on slides prepared by Prof. Øystein Haugen, HiØ & SINTEF

Overview of lecture

- State machines
- Consistency wrt lifeline
- One versus several control states
- Robustness

Suitability of UML state machines

- reactive
- concurrent
- real-time
- distributed
- heterogeneous



Main notions

Finite

- a finite number of control states

Control state

- a stable situation where the process awaits stimuli
- represents the control pointer within program execution

Machine

- only stimulus in the form of a message triggers behavior
- the behavior consists of executing transitions
- may also have local variables (not to be confused with control states)

Exercise

- What is a *control state* in a dataprogram?
- What is a *machine* in a programming language?

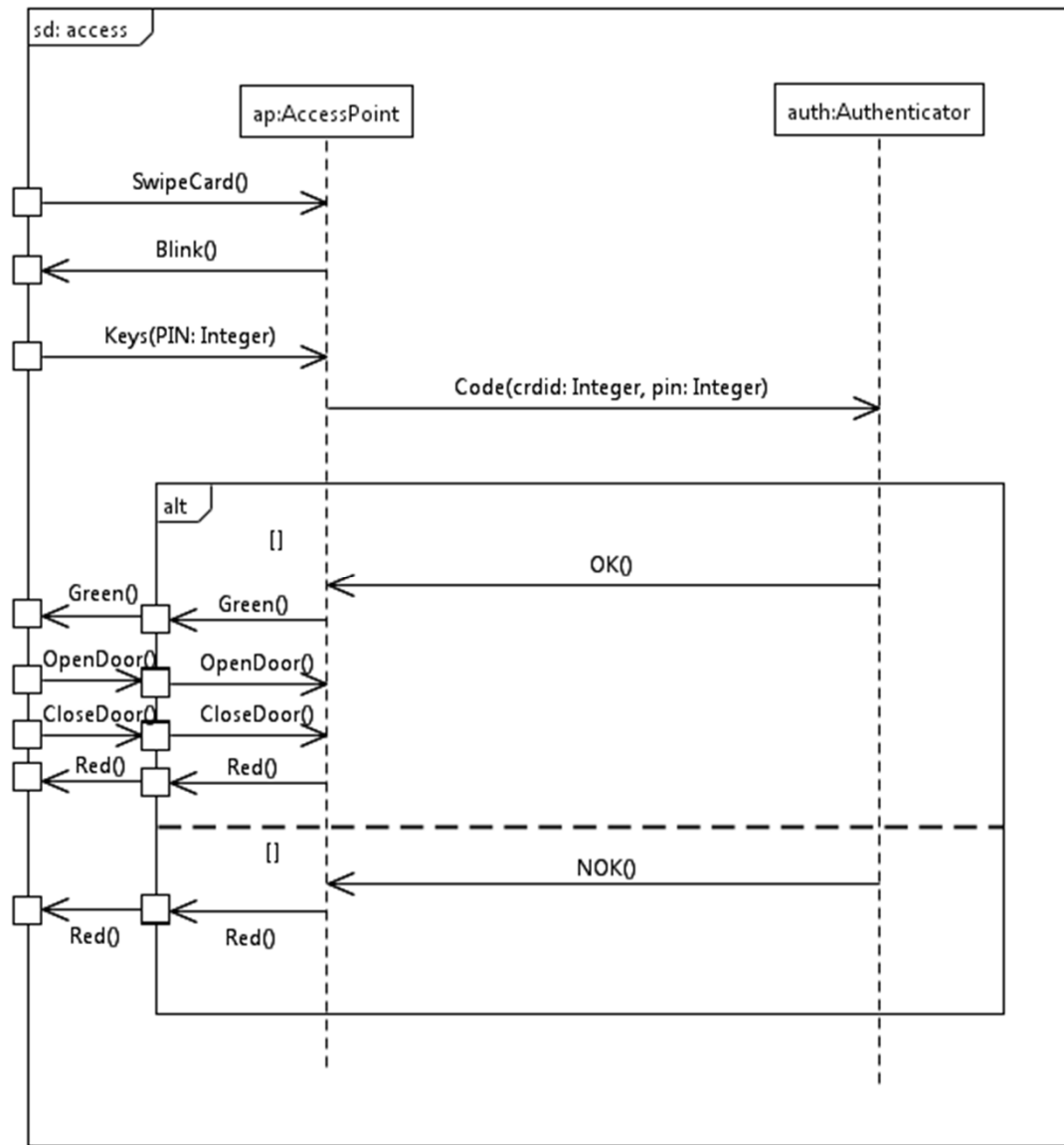
Our example today



Access control system

- A set of Access Points are established to control the access to an area
- The Access Points controls the locking of a door
 - in a more abstract sense, access control systems may control bank accounts or any other asset that one wants to protect
- The Access Point access is granted when two pieces of correct identification is presented
 - a card
 - a PIN (Personal Identification Number)
- The access rights are awarded by a central Authentication service

Happy Day Scenario



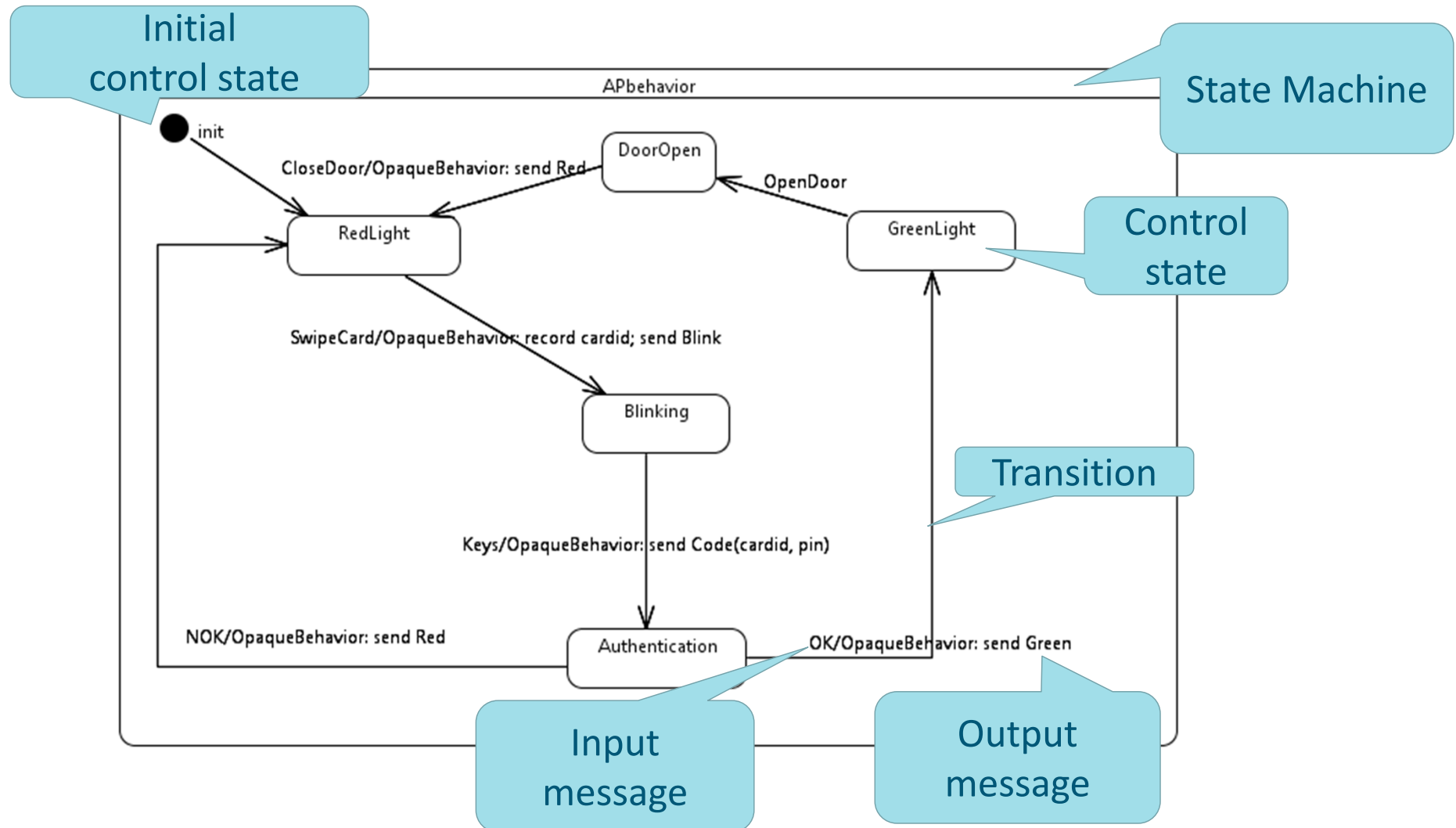
Exercises for sequence diagram Access

- What is the length of the shortest trace?
- What is the length of the longest trace?
- Describe the set of traces representing the semantics of the sequence diagram?

OpaqueBehavior is a UML behavior defined in another language

In this course we are flexible wrt how behaviors are expressed
Hence, using the OpaqueBehavior construct is not important

The behaviour of the AccessPoint



Exercises for state machine APbehaviour

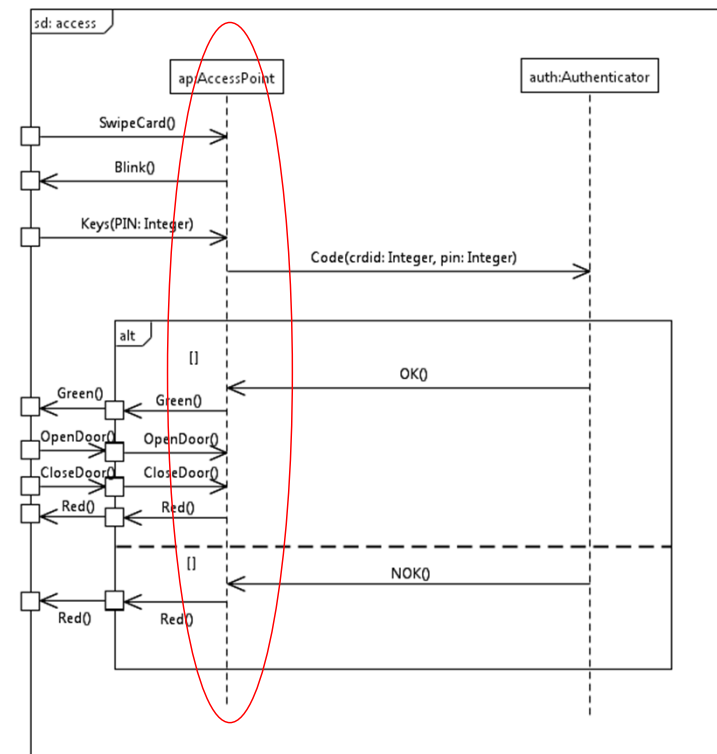
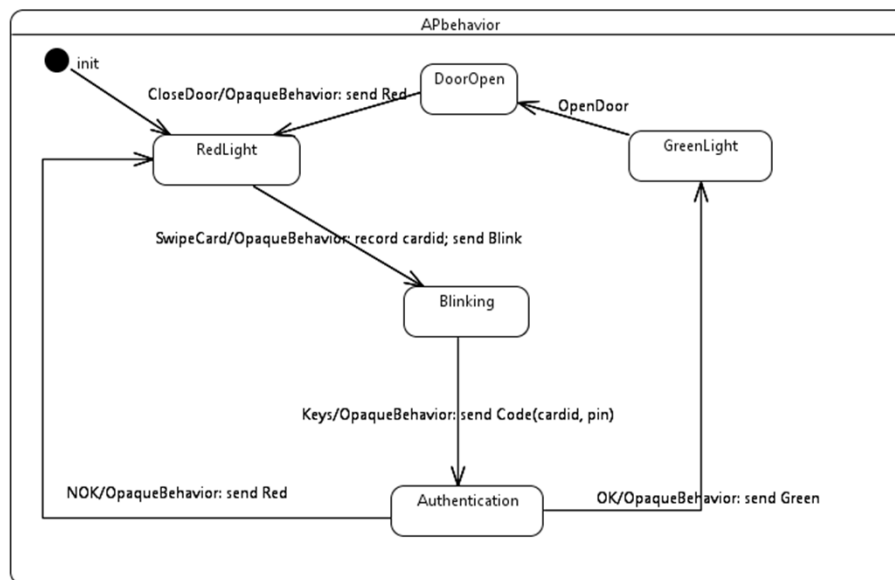
- What is the length of the shortest trace?
- What is the length of the longest trace?
- What is the size of the set of traces representing the semantics of the state machine?

Exercise

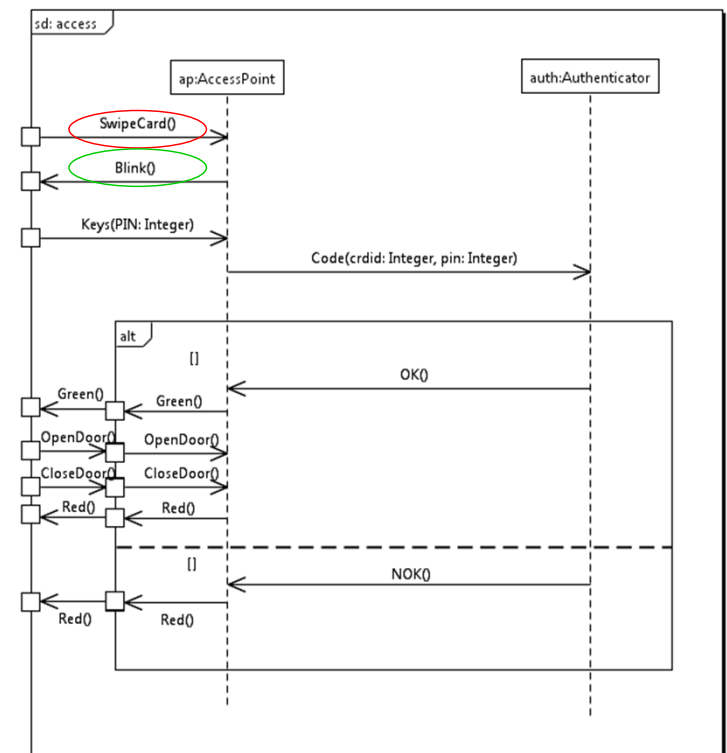
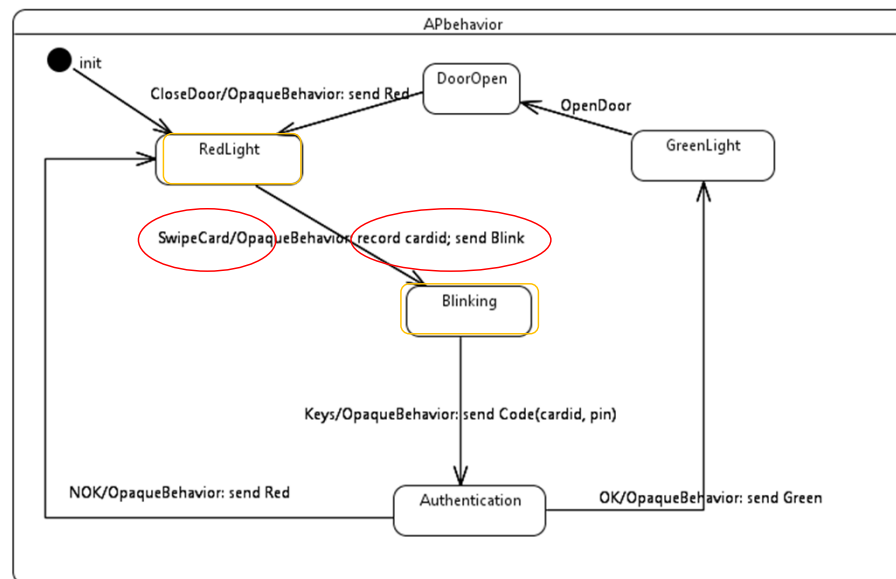
- Make a state machine for the Authenticator

Consistency wrt lifeline

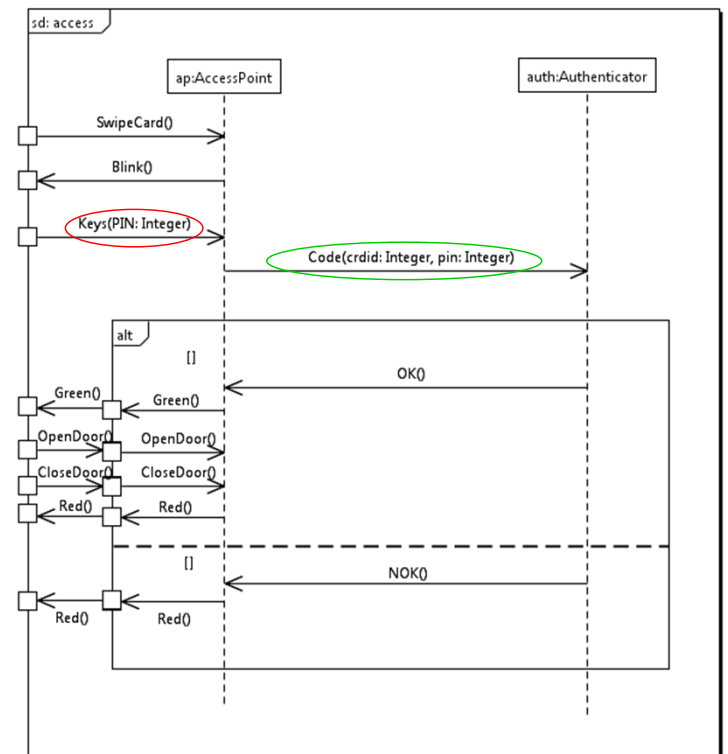
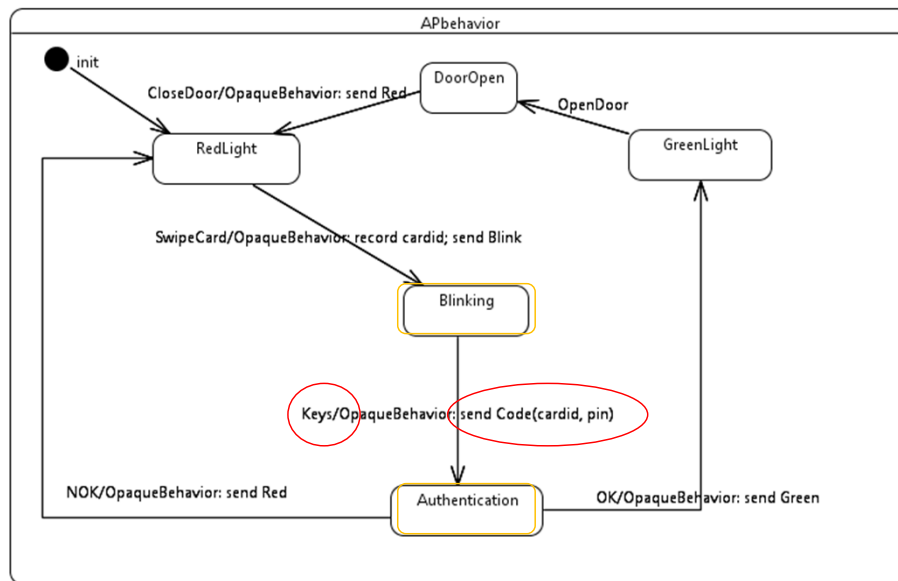
Runtime consistency check



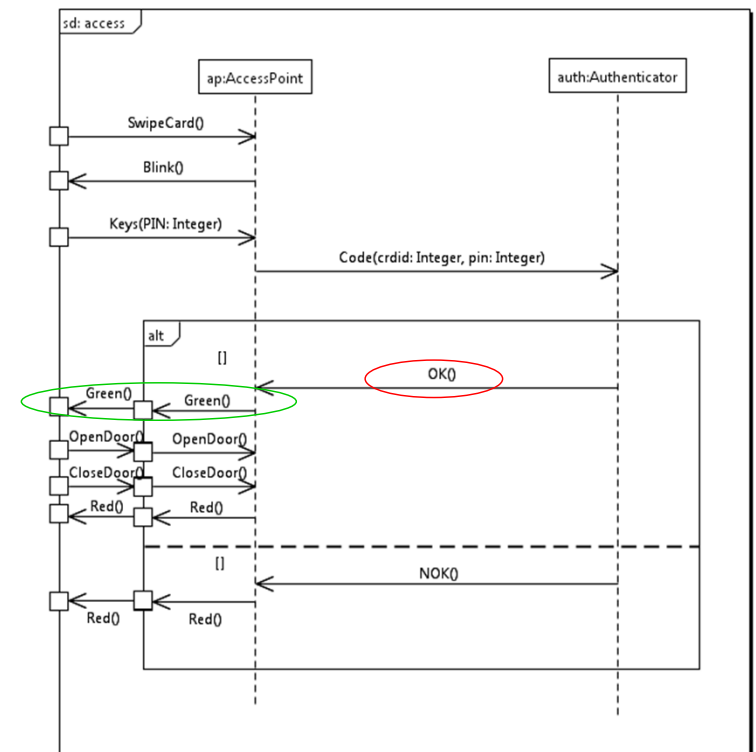
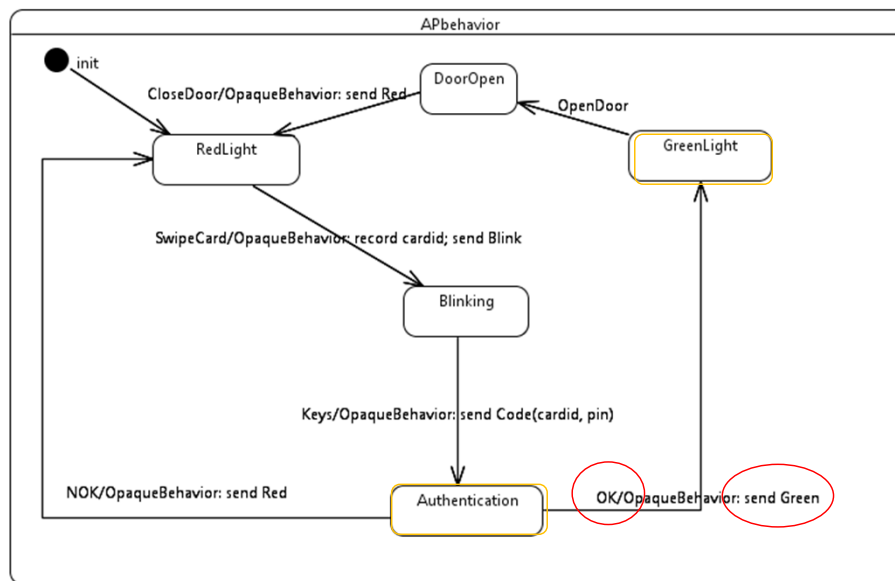
Let's execute the state machine according to the sequence diagram



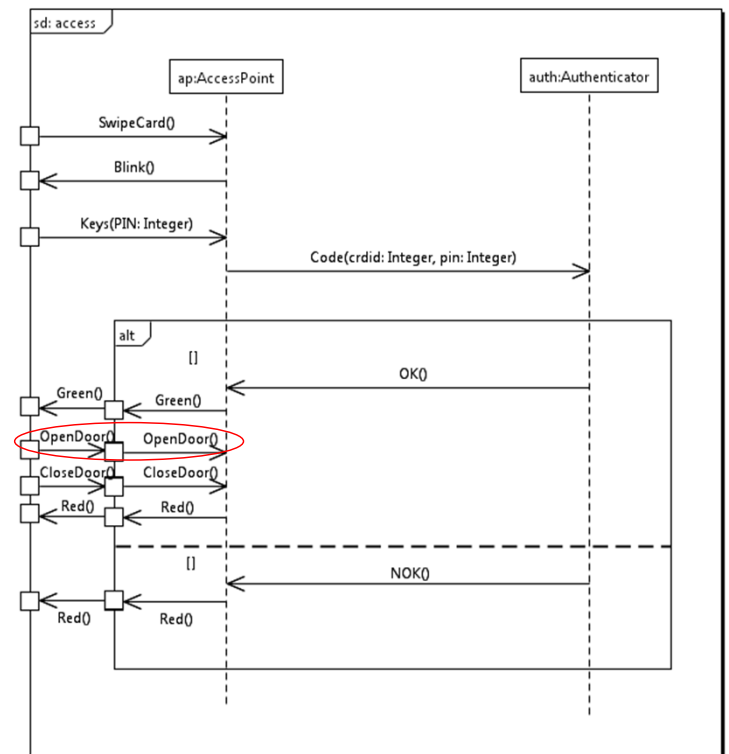
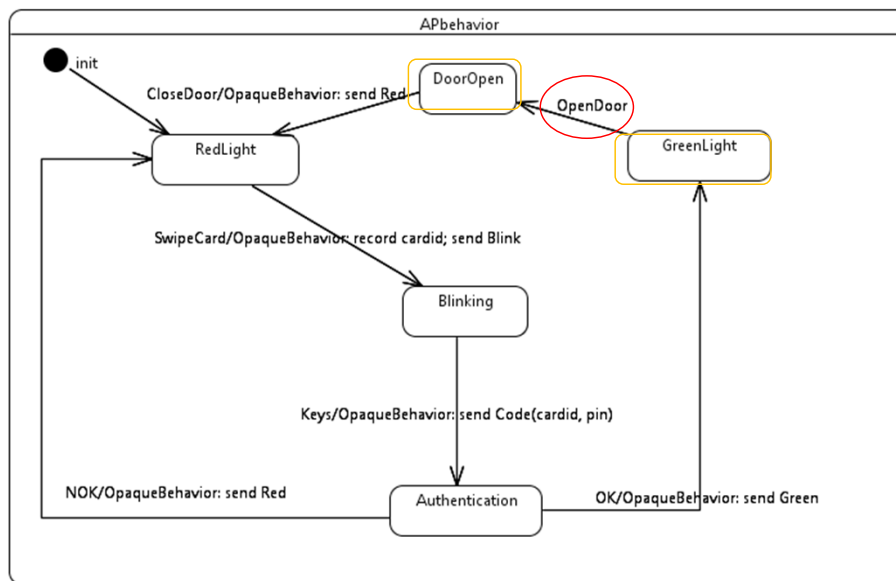
Play it again Sam



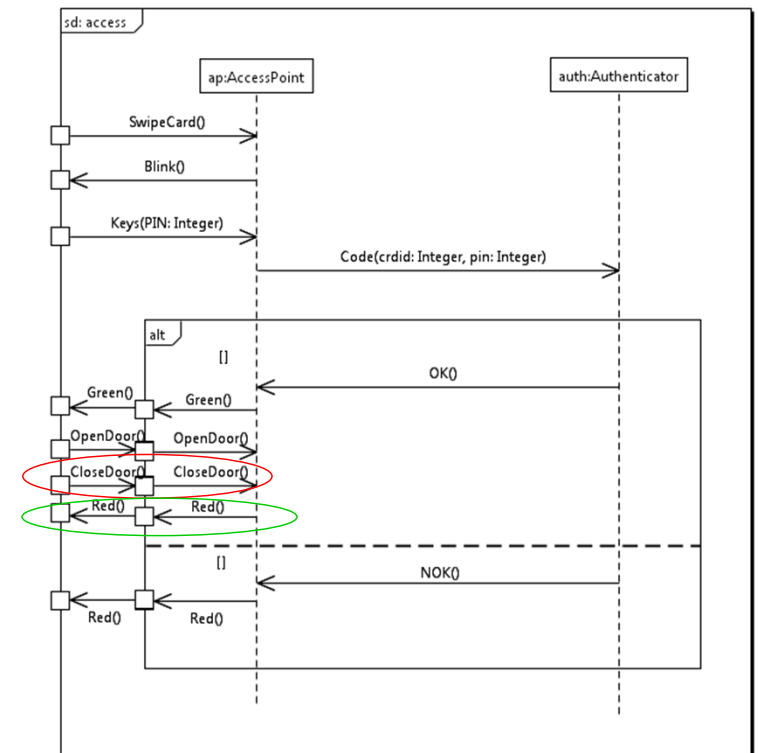
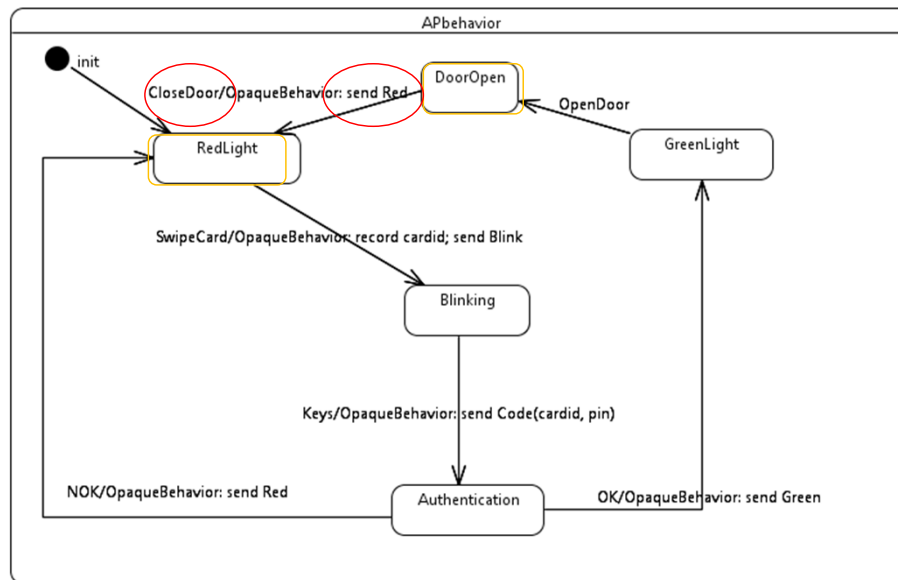
Access granted (one out of two alternatives)



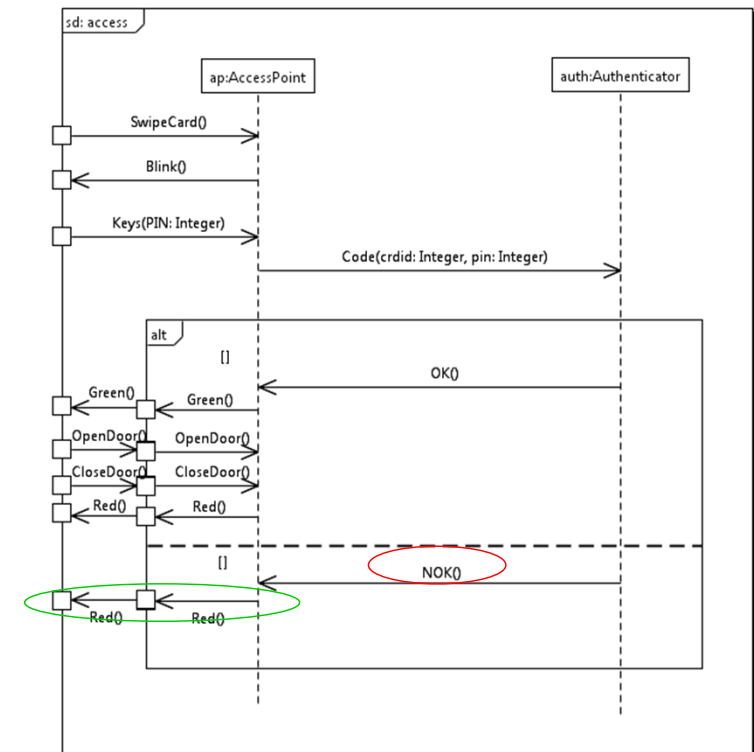
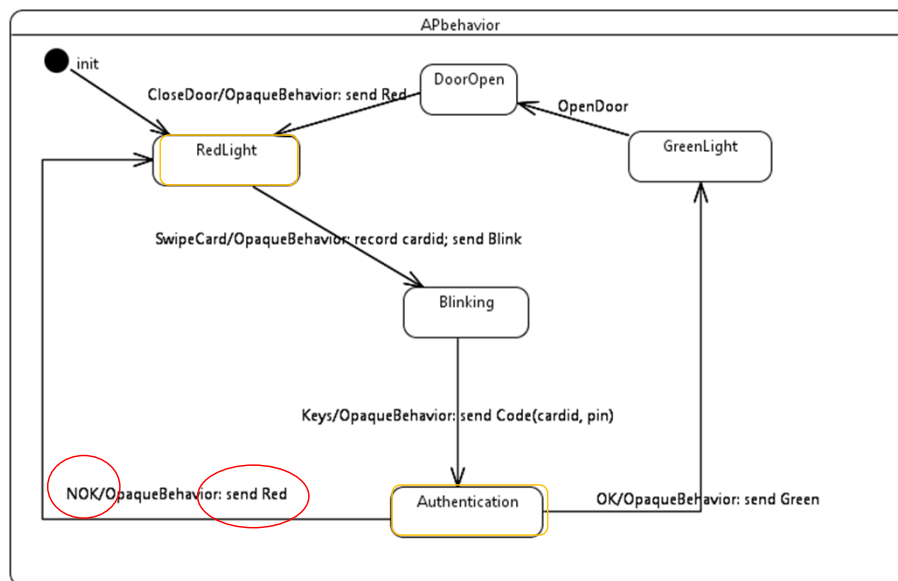
User opens the door



User closes the door again



Access not granted (second of two alternatives)

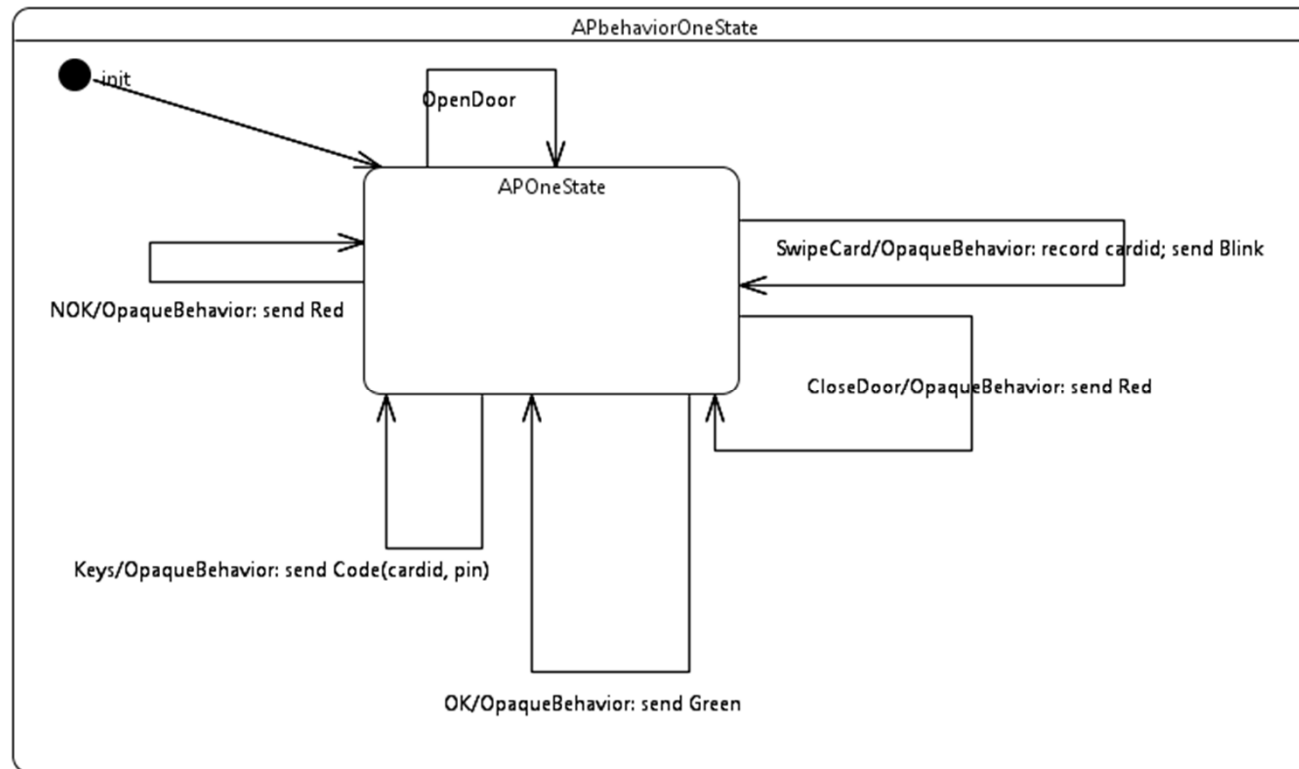


Concluding the runtime consistency check

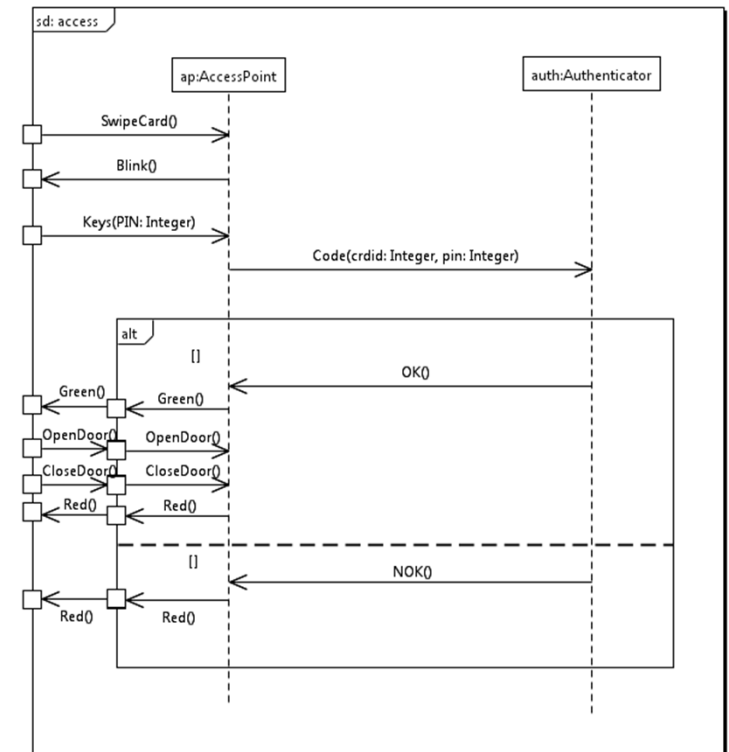
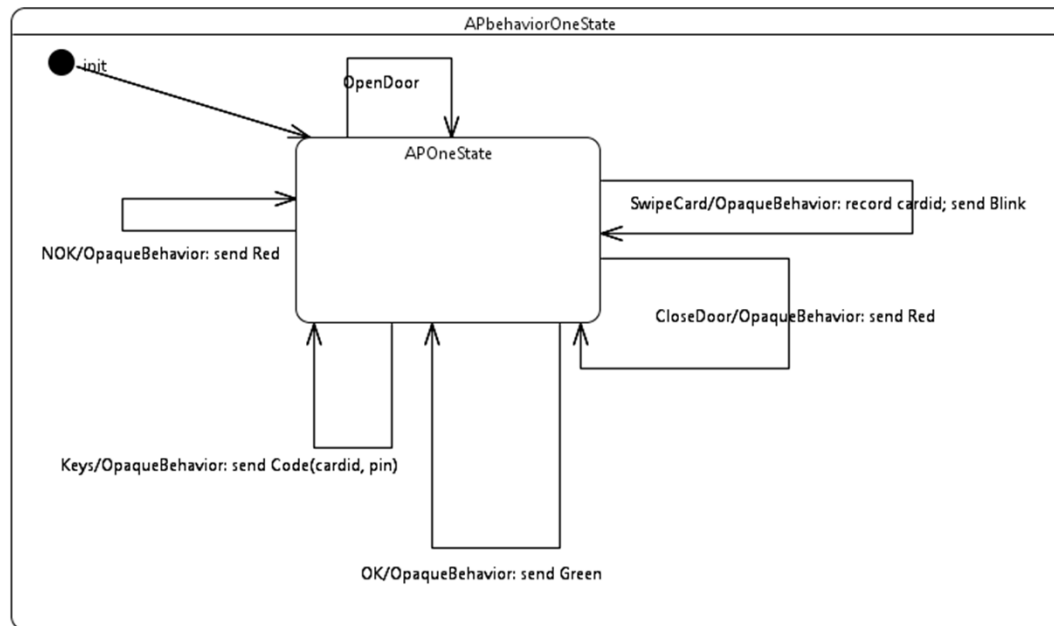
The state machine APbehavior allows all traces of the sequence diagram Access

All traces of the sequence diagram are consistent with the state machine

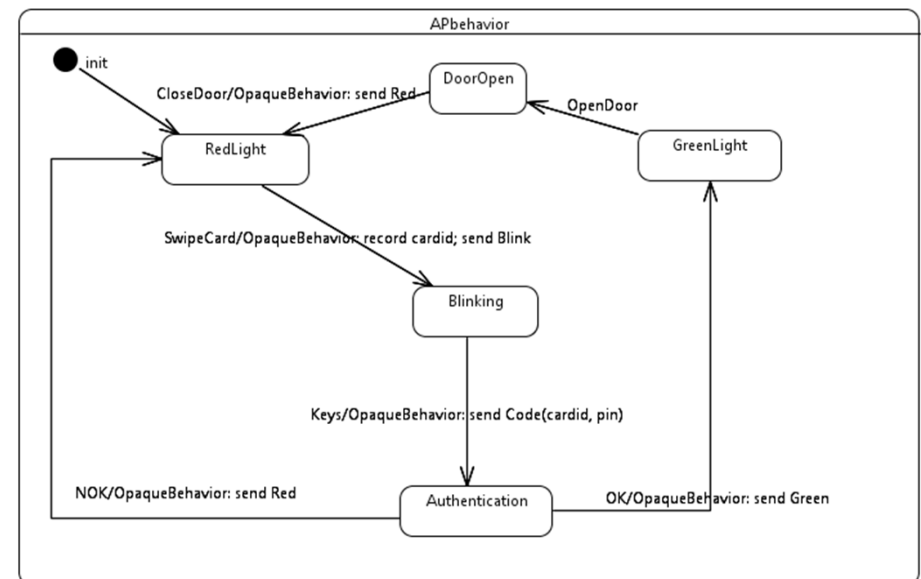
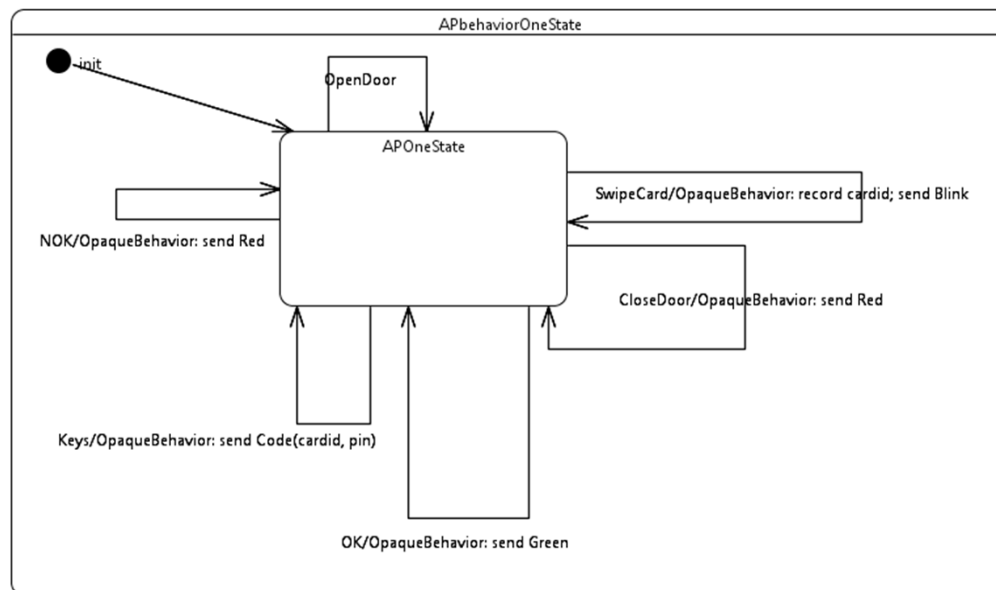
Another attempt to define the state machine



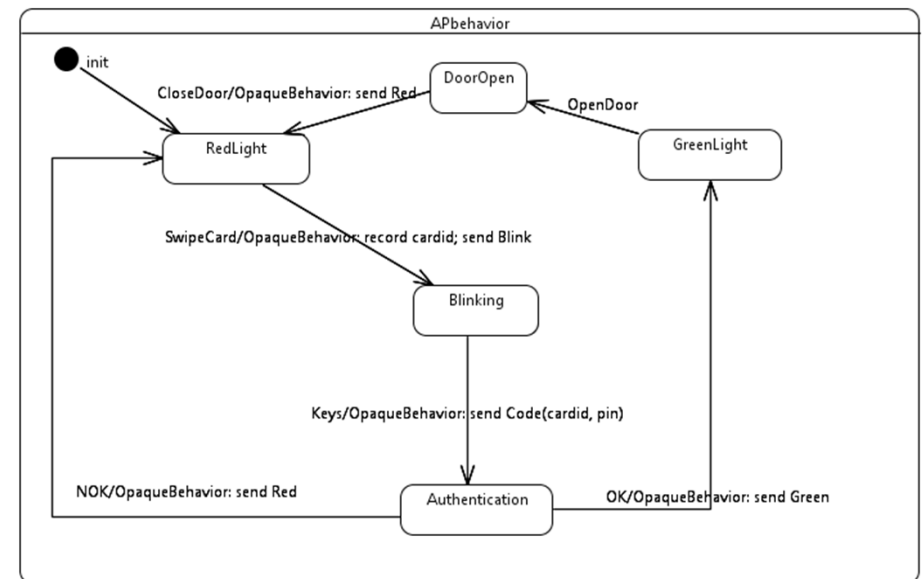
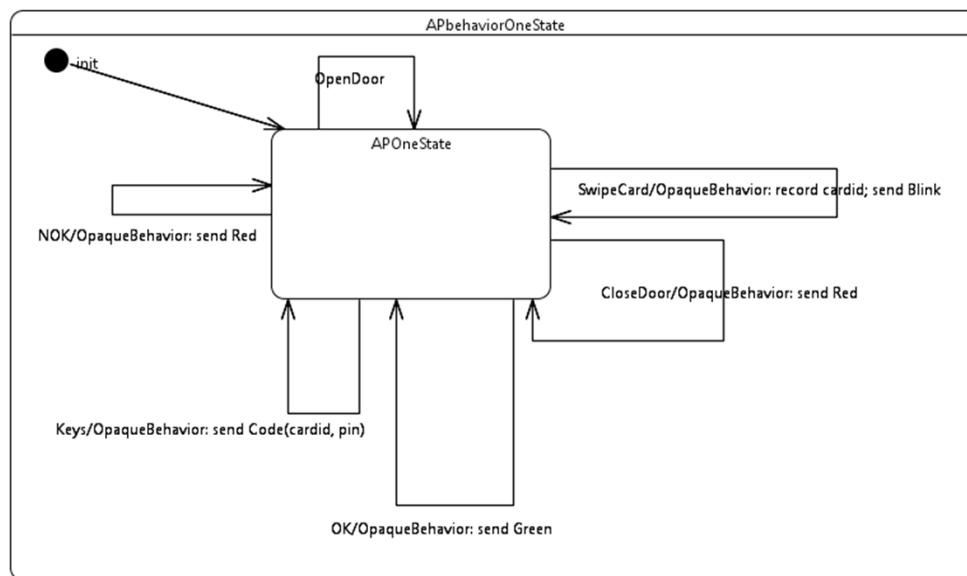
Do we still have consistency?



Which state machine is the better description?



What if the user started keying the PIN at once?



APbehavior may spot the problem
APbehaviorOneState will go on in error

Why use several control states?

- Different control states distinguishes between different situations
- In different situations, different reactions may be desirable to the same trigger

Exercise

- Explain the difference between the two machines in terms of a dataprogram

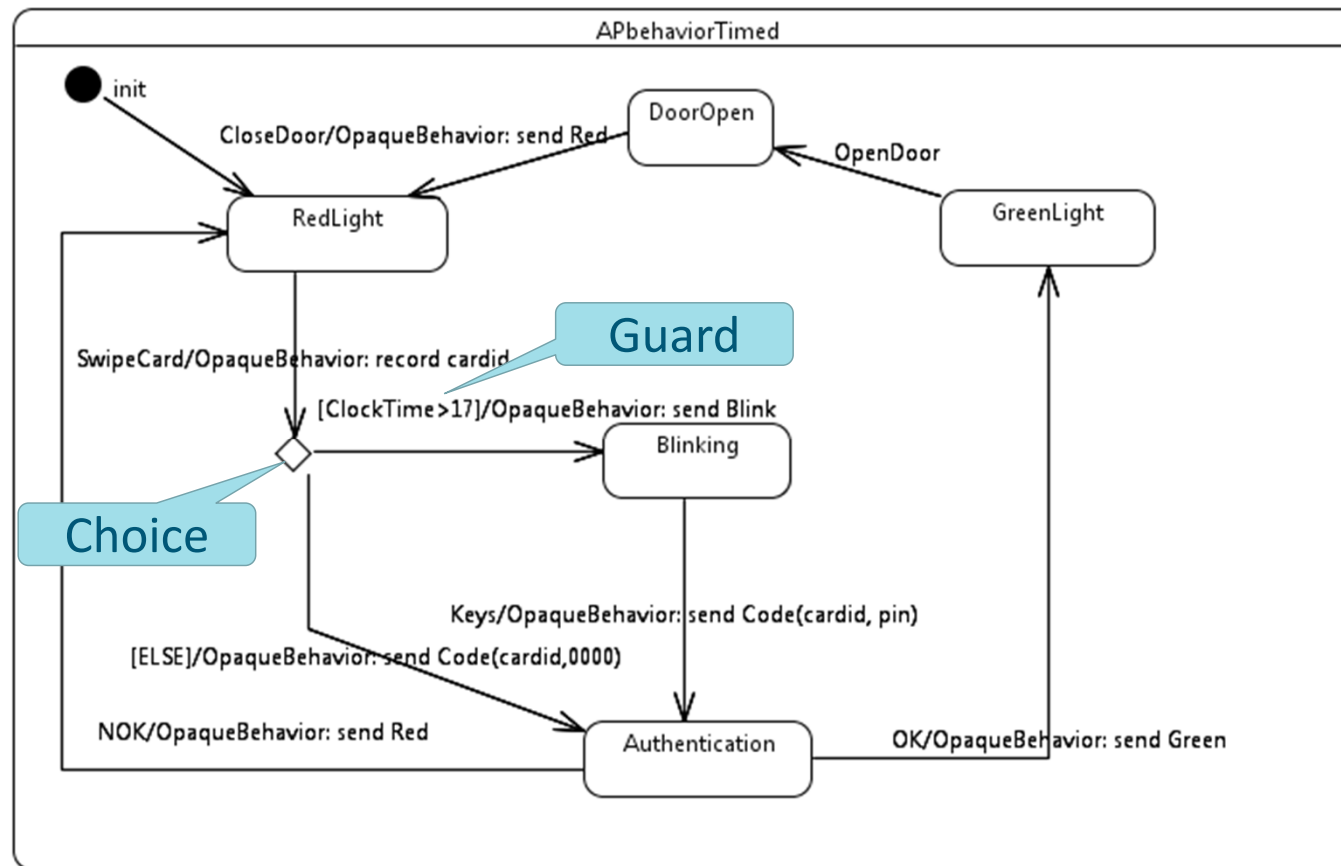
Guidelines and Reminders

- Even though the one control state machine was consistent with the sequence diagram, the state machine was flawed
 - sequence diagrams are only partial descriptions
 - state machines are complete descriptions
- Use several control states if you can
 - each control state represents a recognizable situation
- We should supplement our state machine with all possible transitions
 - this helps us consider and handle most error situations

What if we need to modify a state machine?

- Our access control system should possibly be acting differently during working hours than at other times
- How well do state machines cope with modifications?

Enhancing the state machine



Summary

- State machines describe behaviour of independently acting components
- Reactive systems are suitable for state machines
- Consistency checks between state machines and their respective lifelines are very useful, but not sufficient
- State machines are robust in as much as additional functionality can often be included without ripple effects on other parts of the behaviour