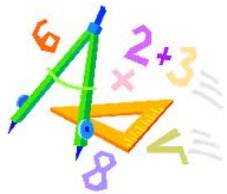
Formal Verification of UML Statechart Diagrams with COSIDE®

Karin Greimel

16, October, 2014

Formal Verification



Def.: Includes all mathematical techniques to verify security and/or correctness of software or hardware.

- Possible techniques:
  - refinement,
  - theorem proving,
  - model checking,
  - equivalence checking
- Possible application fields:
  - security protocol verification,
  - software verification,
  - hardware verification

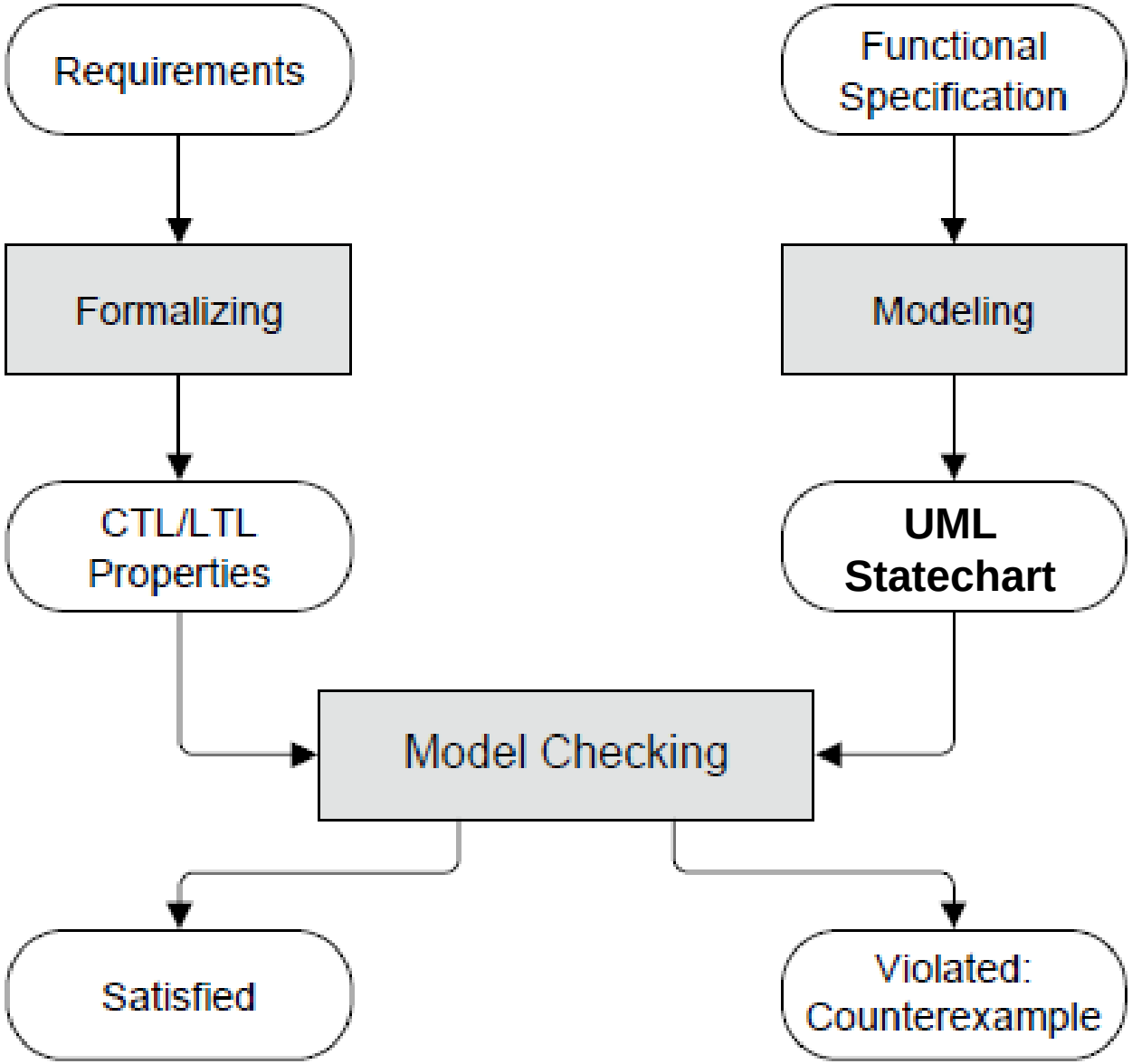
Why Formal Verification

- Security Products:
  - eGovernment (e.g. electronic Passport)
  - Bank Cards (e.g. Credit Cards)
  - Smart Mobility & Access Management Cards

Why Formal Verification of UML Statechart Diagrams

- Mathematical proof that the functional specification satisfies the requirements
  - Requirements – what
  - Specification – how
- Feasible but still useful:
  - Find errors early – before they are implemented
  - Generate precise/unambiguous understanding of the specification
  - Increases assurance as required for e.g. Common Criteria certification

How

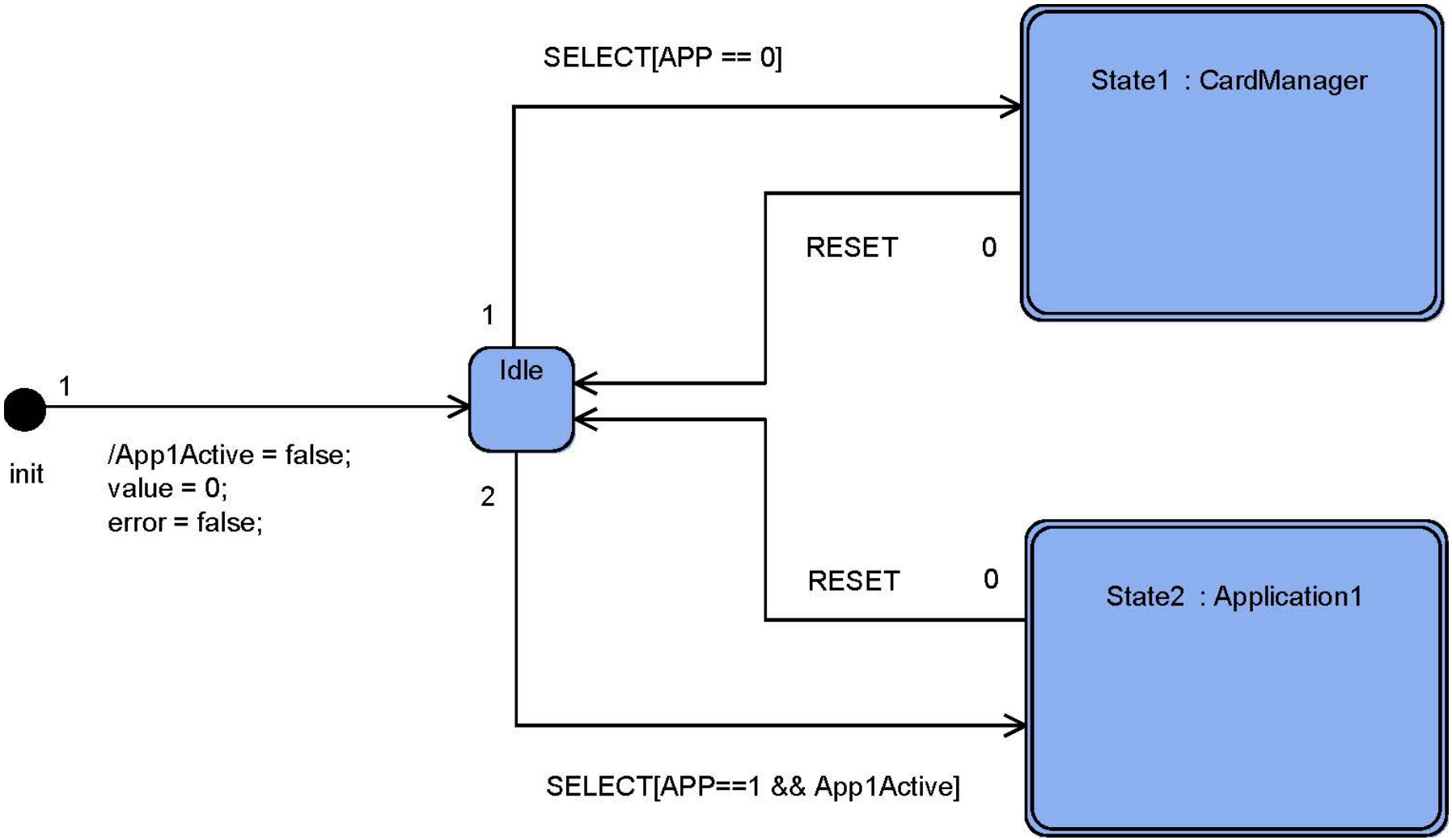

Requirements
Functional Specification
Formalizing
Modeling
CTL/LTL Properties
UML Statechart
Model Checking
Satisfied
Violated: Counterexample

Simplified Example – Access Control Policy

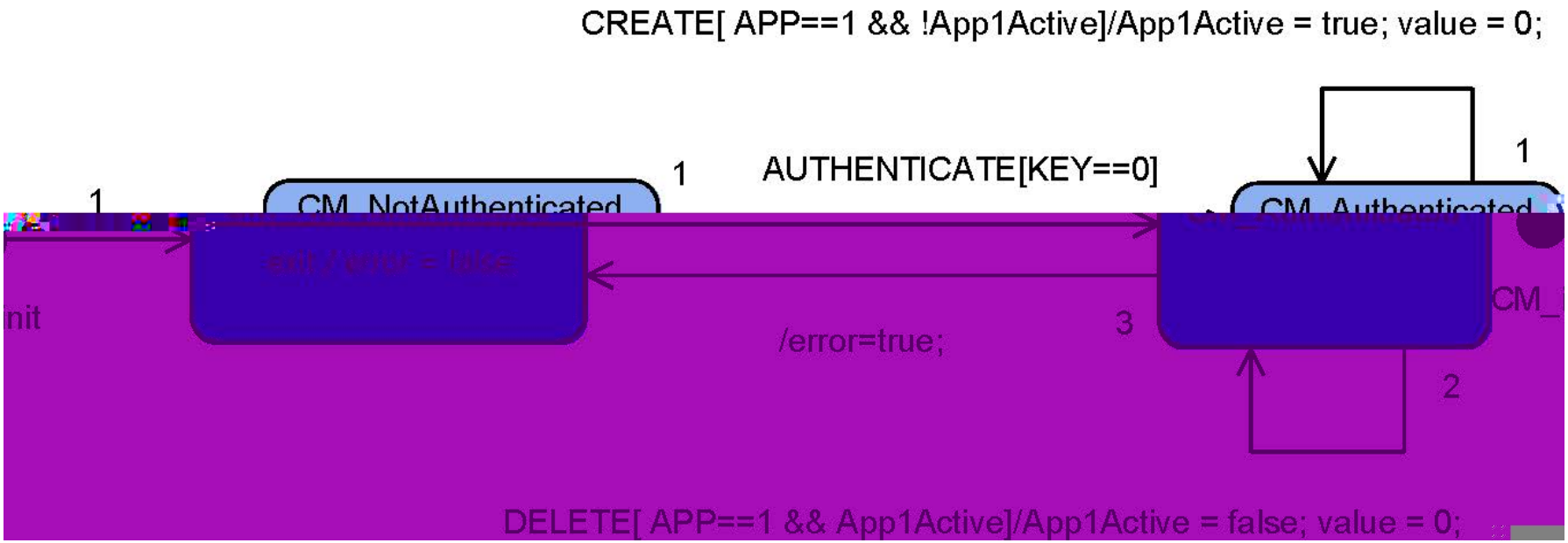
- 2 Features
- A public transport company can create/delete an application on the card (has to be authenticated with KEY = 0).
- A customer can incremented and decremented the value stored in the application (has to be authenticated with KEY = 1).

- Modeled with COSIDE®

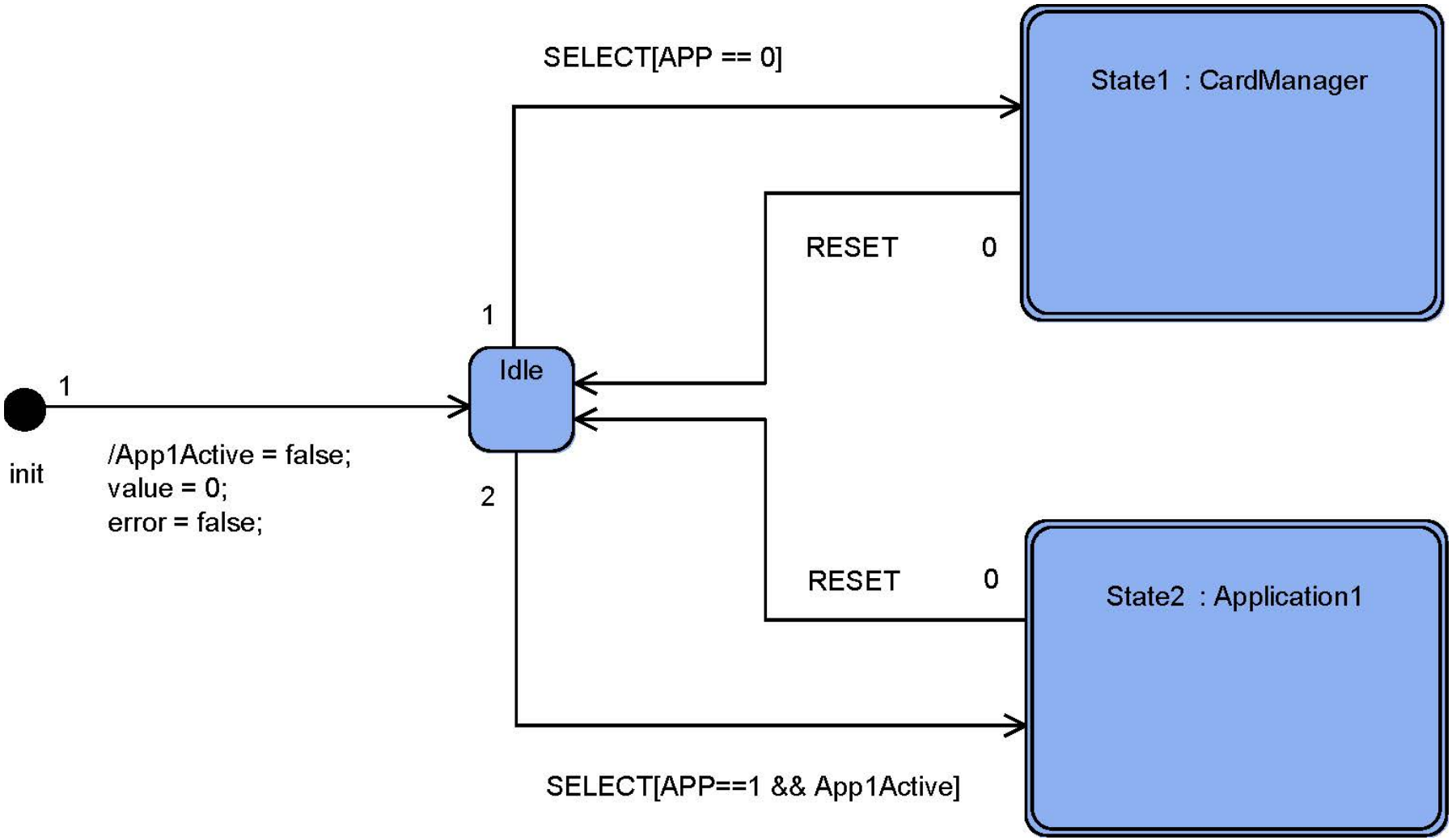
Example


SELECT[APP == 0]
State1 : CardManager
RESET
0
1
Idle
1
init
/App1Active = false;
value = 0;
error = false;
2
RESET
0
State2 : Application1
SELECT[APP==1 && App1Active]

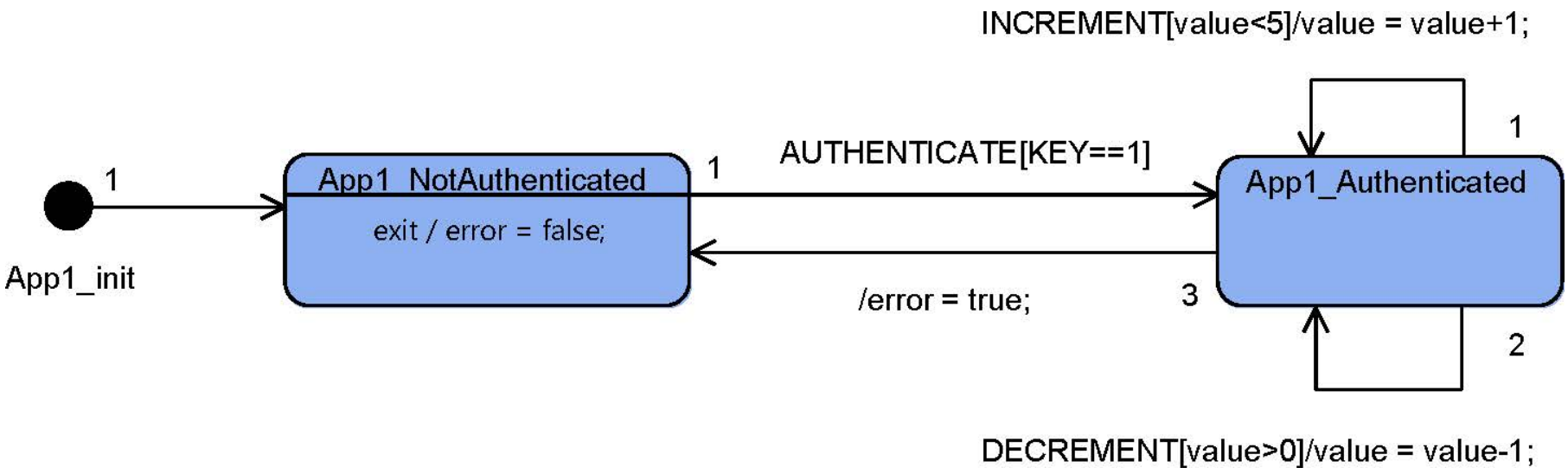
Example


CREATE[APP==1 && !App1Active]/App1Active = true; value = 0;
1
1
AUTHENTICATE[KEY==0]
1
CM_NotAuthenticated
CM_Authenticated
exit / error = false
init
/error=true;
3
2
DELETE[APP==1 && App1Active]/App1Active = false; value = 0;

Example


SELECT[APP == 0]
State1 : CardManager
RESET
0
1
Idle
1
init
/App1Active = false;
value = 0;
error = false;
2
RESET
0
State2 : Application1
SELECT[APP==1 && App1Active]

Example


INCREMENT[value<5]/value = value+1;
1
1
App1_init
App1_NotAuthenticated
exit / error = false;
1
AUTHENTICATE[KEY==1]
App1_Authenticated
/error = true;
3
2
DECREMENT[value>0]/value = value-1;

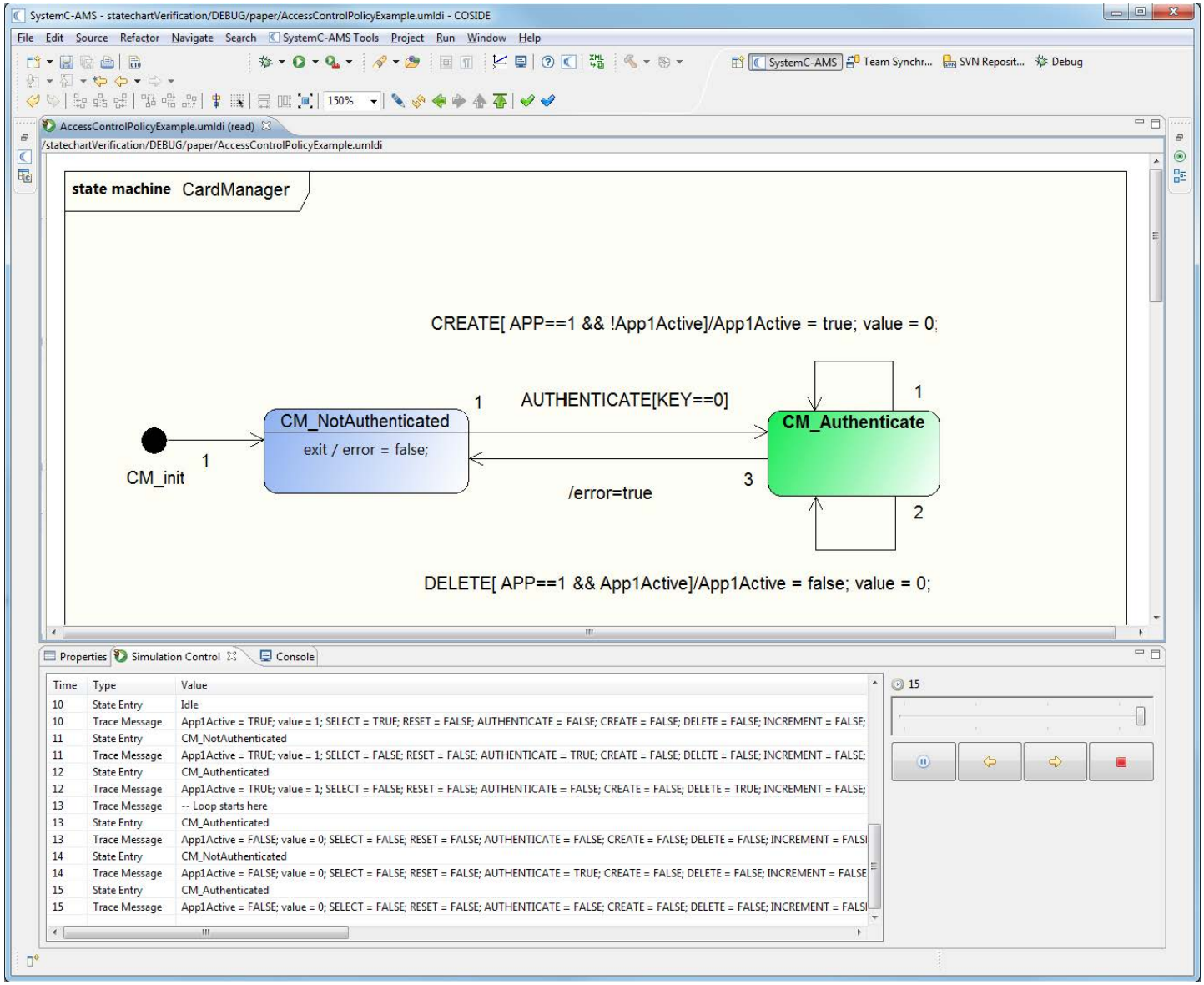
Example

- It is only possible to create an application when authenticated with the card manager key.

$$\mathbf{G}(\ !\text{App1Active} \ \&\ \mathbf{X}(\ \text{App1Active}\ ) \rightarrow \text{state} = \text{CM\_Authenticated}\ )$$

create an application

- Model checker proves the property.
- For properties that can not be proven a counter example is given.
- Counter examples can be visualized.


SystemC-AMS - statechartVerification/DEBUG/paper/AccessControlPolicyExample.umldi - COSIDE
File Edit Source Refactor Navigate Search SystemC-AMS Tools Project Run Window Help
AccessControlPolicyExample.umldi (read)
/statechartVerification/DEBUG/paper/AccessControlPolicyExample.umldi
state machine CardManager
CREATE[APP==1 && !App1Active]/App1Active = true; value = 0;
CM_init
1
CM_NotAuthenticated
exit / error = false;
1
AUTHENTICATE[KEY==0]
CM_Authenticate
1
/error=true
3
2
DELETE[APP==1 && App1Active]/App1Active = false; value = 0;
Properties
Simulation Control
Console
Time Type Value
10 State Entry Idle
10 Trace Message App1Active = TRUE; value = 1; SELECT = TRUE; RESET = FALSE; AUTHENTICATE = FALSE; CREATE = FALSE; DELETE = FALSE; INCREMENT = FALSE;
11 State Entry CM_NotAuthenticated
11 Trace Message App1Active = TRUE; value = 1; SELECT = FALSE; RESET = FALSE; AUTHENTICATE = TRUE; CREATE = FALSE; DELETE = FALSE; INCREMENT = FALSE;
12 State Entry CM_Authenticated
12 Trace Message App1Active = TRUE; value = 1; SELECT = FALSE; RESET = FALSE; AUTHENTICATE = FALSE; CREATE = FALSE; DELETE = TRUE; INCREMENT = FALSE;
13 Trace Message -- Loop starts here
13 State Entry CM_Authenticated
13 Trace Message App1Active = FALSE; value = 0; SELECT = FALSE; RESET = FALSE; AUTHENTICATE = FALSE; CREATE = FALSE; DELETE = FALSE; INCREMENT = FALSE
14 State Entry CM_NotAuthenticated
14 Trace Message App1Active = FALSE; value = 0; SELECT = FALSE; RESET = FALSE; AUTHENTICATE = TRUE; CREATE = FALSE; DELETE = FALSE; INCREMENT = FALSE
15 State Entry CM_Authenticated
15 Trace Message App1Active = FALSE; value = 0; SELECT = FALSE; RESET = FALSE; AUTHENTICATE = FALSE; CREATE = FALSE; DELETE = FALSE; INCREMENT = FALSE
15

Summary

- We formally prove that the **functional specification** (UML state diagram) satisfies the **requirements** (temporal logic formula).
- Modeling the specification and using an input language that is understood by engineers, helps to
  - avoid errors at the specification phase
  - generate a common understanding of the specification
- Ensure high quality and security – enabling certification