UCEd Use Cases development approach

Stéphane S. Somé
SITE, University of Ottawa,
ssome@site.uottawa.ca

Abstract

This document presents an approach for specifying use cases with the Use Case Editor (UCEd). We propose an iterative approach where use cases are defined in conjunction with a domain model. Each iteration involves validation by inspection and simulation.

1. Overview

The following flowchart shows the first part of a suggested approach for developing use cases using UCEd. The process starts with requirements including the overall business objective, description of product intent, preliminary version of use cases elicited from users, etc.

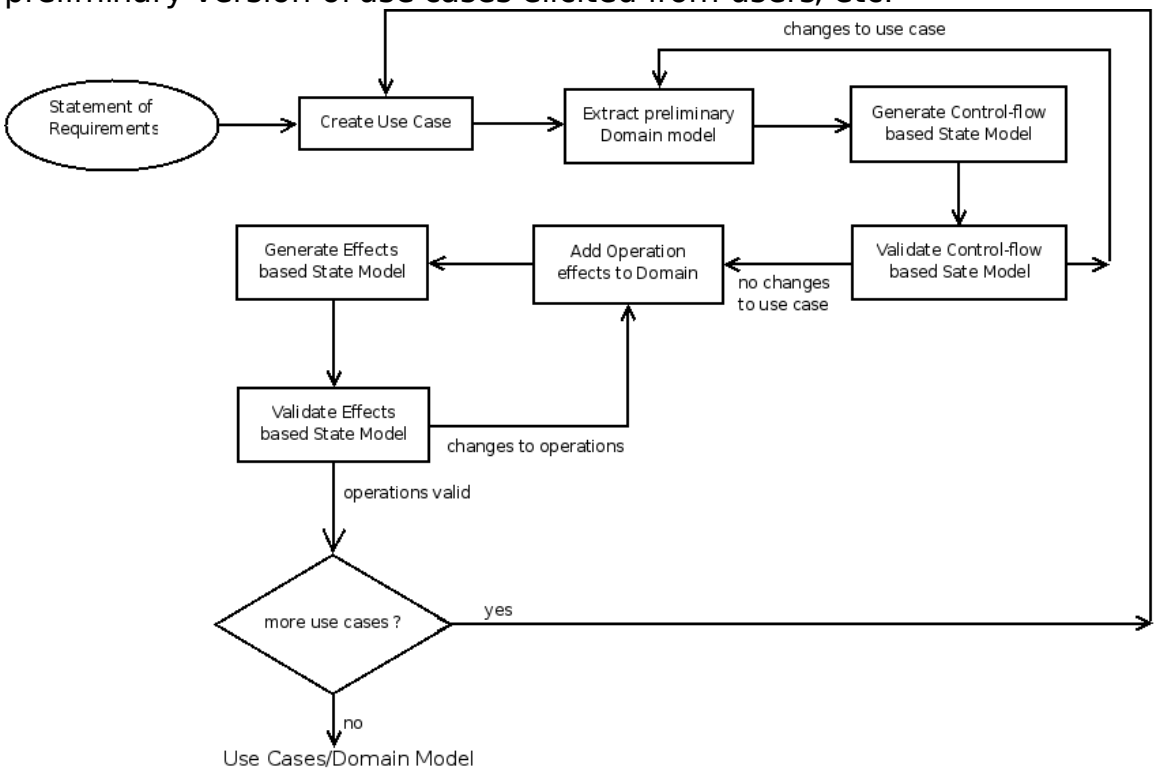


Figure 1: Use Cases development approach with UCEd

The end result is a requirements specification consisting of use cases and a preliminary domain model. In the second part of the process (discussed in Section 9) sequencing between related use cases is specified and the domain model updated.

We use an Automated Teller Machine (ATM) system to illustrate the approach. The initial requirements should state information such as what the system is about, who are the users, what are the boundaries and so on. We suppose a classical ATM system which goal is to provide banking facilities to customers. For sake of simplicity, we suppose the system includes the ATM interface itself as well as all the back treatment necessary for banking. At this stage, it is possible that preliminary sketching of the main functions and transactions are known.

2. Use Cases Creation

We suggest an iterative approach for use cases creation, where use cases are defined, edited, and validated one at the time.
As an example, suppose our first use case for the ATM system is a use case describing how users log themselves in the system in order to use it for their transactions.

The first task in a use case definition is to come up with a meaningful name. A use case name should capture the high-level goal of the described activity. In our example, a good name would be “log in”. The use case is then added to the use case model.

- Open the ***use case editing tool*** (**File -> Open -> Use Case Editor**), and Specify a new name for the use case by replacing the default name (*New*) by the use case name (“*log in*”). Click on the new use case in

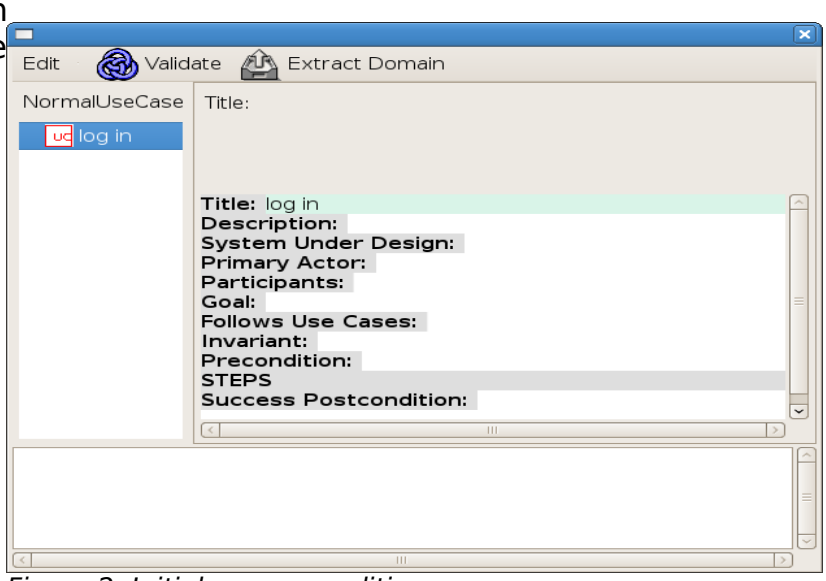


Figure 2: Initial use case edition

the use case model tree to open a description editor.

- Once a use case has been identified in the use case model,

1. Define the use case description elements. The mandatory elements are: *title, system under design, primary actor,* and *precondition.* The title is the same as the use case name. The following figure shows the ***use case writing tool*** after the first stage of use case "log in" creation.

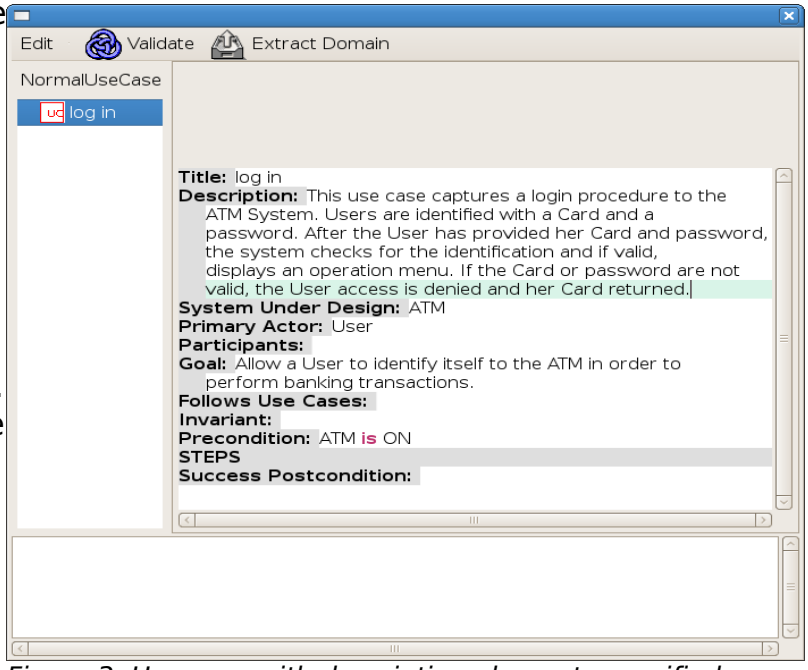


Figure 3: Use case with description elements specified

In the above example, the system under design is *ATM*, the primary actor is *User* (the one who initiate the use case and have a goal that needs to be satisfied) and the precondition is "*ATM is ON*". Recall that conditions are in the form <Entity> *verb* <Value>
Where *verb* is a conjugated form of *"to be"* in the present tense (e.g: is, are, is not, ...)[1]. A *description* and a *goal* should also be specified for the use case. The description is a short paragraph on what the use case is about. The goal specifies in one or two sentences the expectation of the use case actors from the execution of the use case.

2. Define use case primary scenario.
   A primary scenario describes the normal course of events in a use case. The use case *goal* is fulfilled at the end of this scenario.

   Create a sequence of actor actions, system reactions to fulfill the goal and to realize the postcondition (if already defined).

   In order to add a first step to a use case:
   - right-click on **STEPS**, and

1 Other allowed verbs are **have** and **can**.

- select ***Add Step***

A new line numbered ***1.*** will be added to the use case description. Type in the use case description ***edition area*** to edit a line,

The following figure shows a primary scenario definition to use case “log in”.

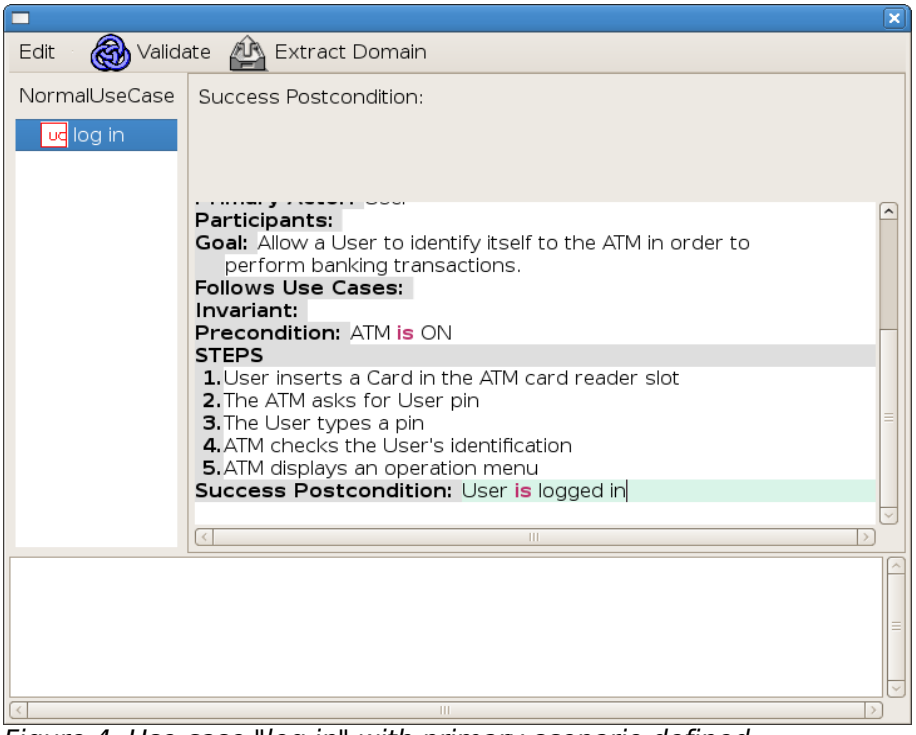


Figure 4: Use case "log in" with primary scenario defined

The scenario is a sequence of steps each being either an actor action or a system reaction. The first step in the primary scenario should be an action performed by the primary actor.
It is also advised to define *postconditions* to scenarios. Condition “*User is logged in*” has been specified as the postcondition of the main scenario of use case “log in”.

3. Add alternative courses of events (secondary scenarios) to use case.

Alternative courses of events may be created by systematically analyzing each step asking questions as

- what error situations are possible after the step ?
- what variations to the normal behavior are possible after the step ?

Some secondary scenarios end-up with the use case goal being abandoned (*failure scenarios*). While other secondary scenarios branch

back at the primary scenario leaving the possibility that the goal becomes fulfilled (*recovery scenarios*). The last step in a recovery scenario is a branching statement (*GOTO*) back to the primary scenario. An alternative postcondition should be specified for each failure scenario.

Right-click on a step and select ***Add Alternative to Step*** to create an alternative scenario from a step.

Figure 5 shows use case “log in” with some alternative scenarios.
Note that not all (even not any) alternative course of events need to be defined before performing the next stages of domain extraction and validation. Several iterations may be used for a single use case.

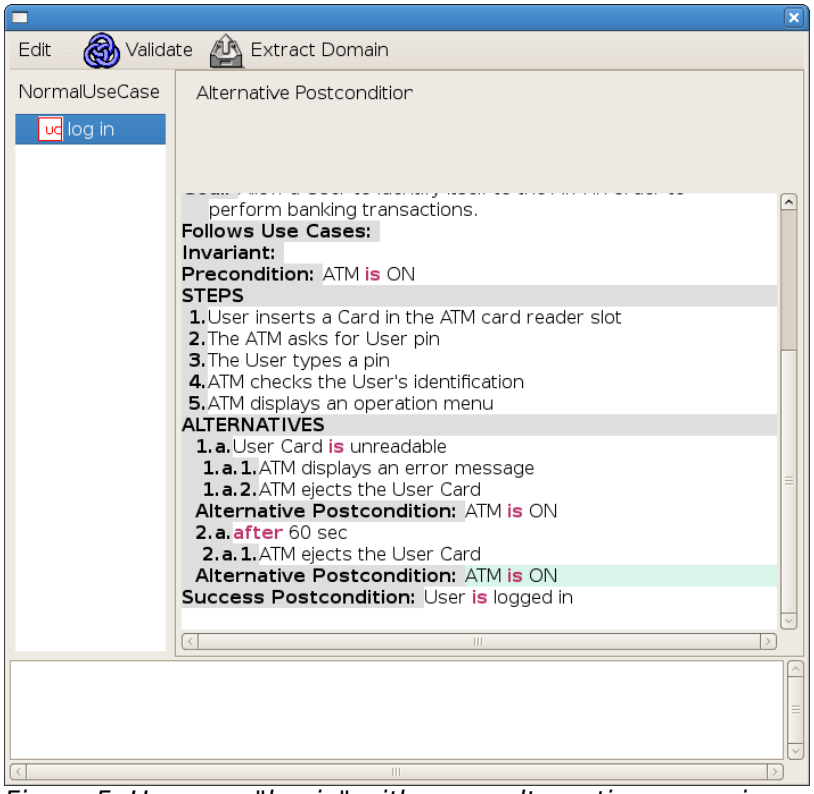


Figure 5: Use case "log in" with some alternative scenarios

3. Domain elements extraction

Use case “log in” cannot be validated nor a state model generated at this stage for a lack of domain model. A domain model can be created manually. Alternatively, some of the domain elements may be extracted from the use case using the ***domain model extraction wizard***.

Double-click on ***Extract Domain*** (or select ***Validate -> Extract Domain From Use Cases***), to start the ***domain model extraction wizard.***

The following figure is a view of a session with the wizard.

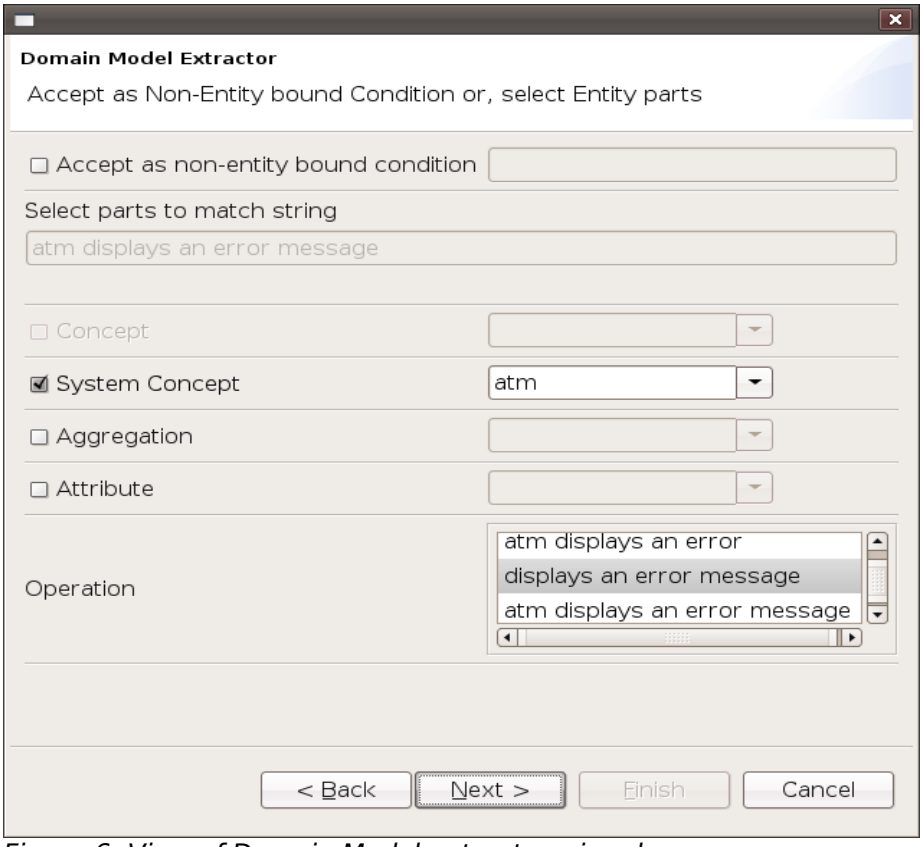


Figure 6: View of Domain Model extractor wizard

Select parts to match string shows a part of a use case text that can not be parsed because of insufficient information. A selection of the appropriate types for each part of the string needs to be made. For instance, in the above example, *ATM* was chosen as a ***System Concept*** and the remainder of the string as an operation of *ATM*.

After going through the wizard, use ***File -> Open -> Domain Editor*** to open the ***domain editor***.

The resulting domain model shown below includes all the necessary elements for use cases validation.

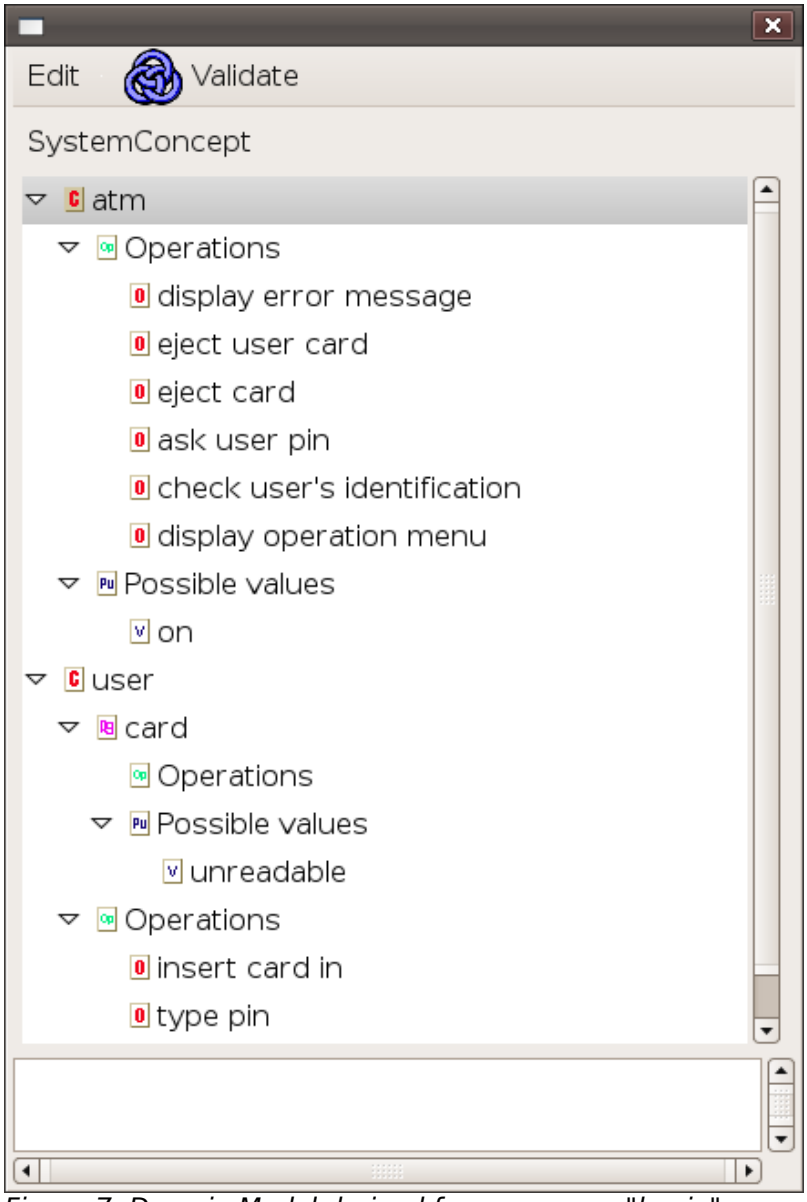


Figure 7: Domain Model derived from use case "log in"

4. Domain Model Validation

A domain model needs to be validated to ensure correctness and completeness. A validated domain model is also needed before use cases can be validated.

Double-click on ***Validate*** (or select ***Validate -> Validate Domain***) to validate the domain. Results of domain model validation are shown in the ***domain editor*** tool message area.

5. Use Cases validation

Use cases validation ensures that a correct syntax is used for use cases and that all use case elements are defined in the domain model.

Select ***Validate -> Validate Use Cases*** to launch use cases validation. Results of use cases validation are shown in the ***use cases editing*** tool message area.

In the ATM example, the validation results are as in Figure 8.

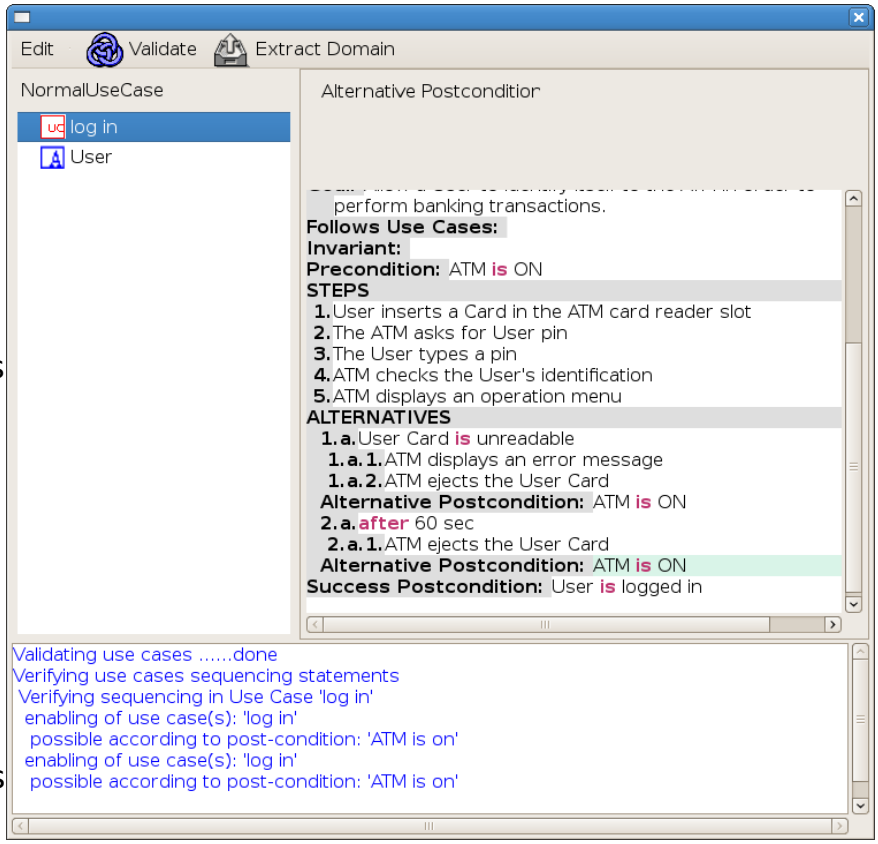


Figure 8: Use case "log in" validation results

The use case syntax is fine. The validation results include warnings related to use case sequencing. We discuss sequencing in Section 9.

6. Control-flow based state model generation

A control flow-based state model for use case "*log in*" is generated by right-clinking on the use case in the ***use case model***, and selecting ***State Machine -> (Re)generate Control Flow State Machine for log in*** (alternatively, menu option ***Generation -> Generation based on Use Case flow -> Generate for: log in*** could be used).

The resulting state model can be viewed by right-clicking on use case "*log in*" and selecting ***State Machine -> View State Machine*** (alternatively, menu option ***State Machine -> State Machine Obtained from Use Case flow -> View Use Case log in*** could be used).

The following Figure shows use case "*log in*" control-flow based state model as shown by the state model viewer.

Transition events are abbreviated for convenience. They can be revealed by hovering over the graph.

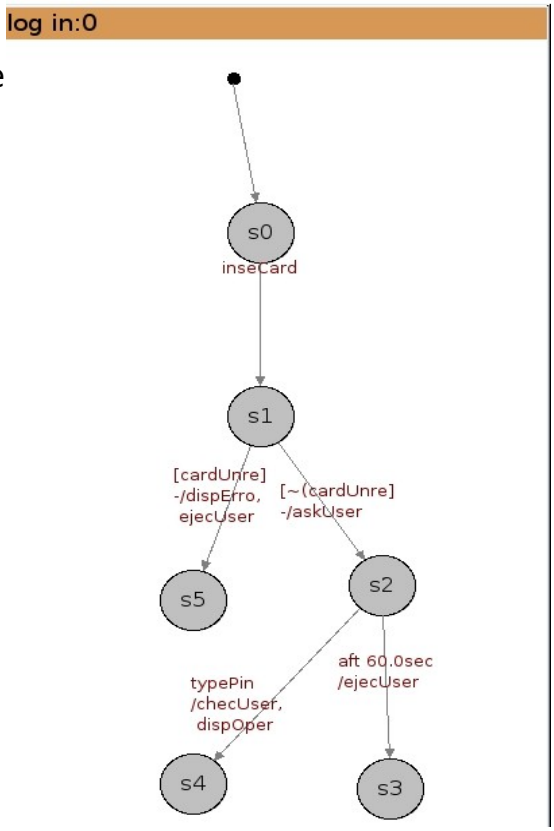


Figure 9: Control-flow based state model generated from "log in"

State models may also be viewed by generating a *Graphviz dot*[2] file and by using a dot viewer. Right-click on use case "*log in*" and select ***State Machine -> Export State Machine in Graphviz dot format*** to generate a *dot* file.

The following shows the resulting dot file rendered with *dotty*.

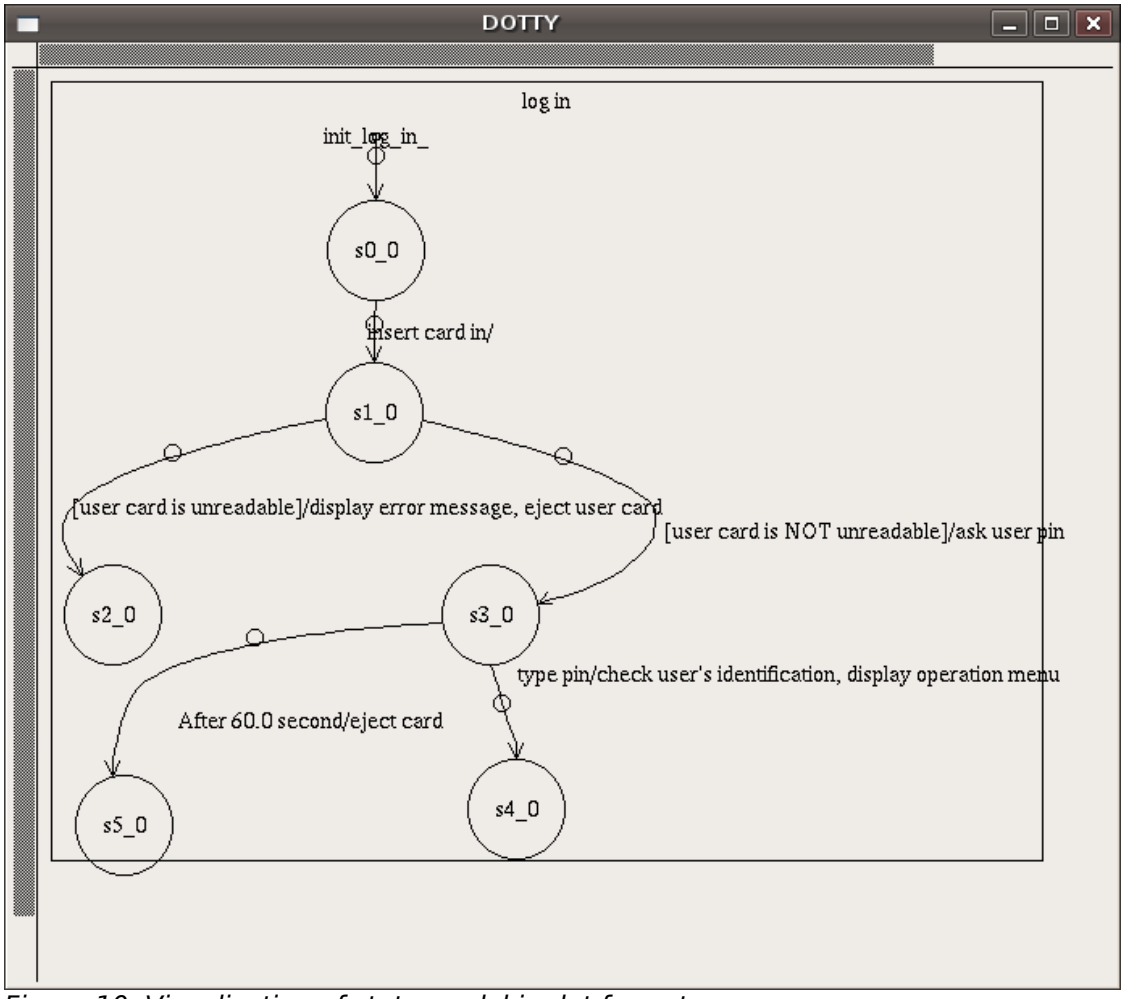


Figure 10: Visualization of state model in dot format

Visualization allows validating that a state model conforms to the intent of a use case. Validation is also possible using simulation.

2 http://www.graphviz.org/

7. State model simulation

Simulation of the control-flow based state model generated from use case "*log in*" is launched by right-clicking on the use case and selecting ***State Machine -> Simulate Use Case log in*** (alternatively, menu option ***Simulation -> State Machine Obtained from Use Case flow -> Simulate Use Case log in*** could be used). UCEd simulation tool is opened using menu option ***File -> Open -> Simulator***.

The following shows a view of the simulation tool given the *ATM* example.

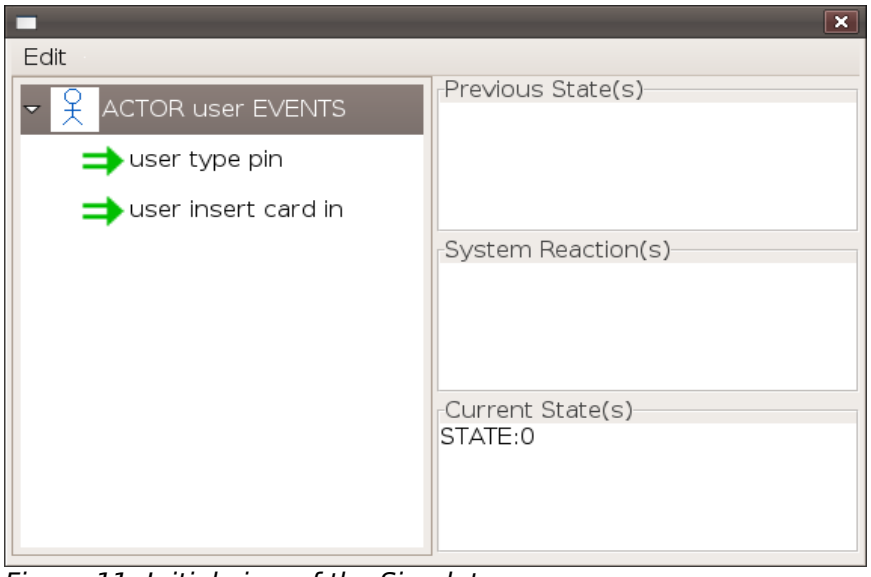


Figure 11: Initial view of the Simulator

Only one actor (*User*) has been defined so far and only two operations are defined for that actor. The current state is the state model initial state.

Events are simulated by clicking on them in the Actor Events pane. For instance clicking on event *user type pin* results in the following message.

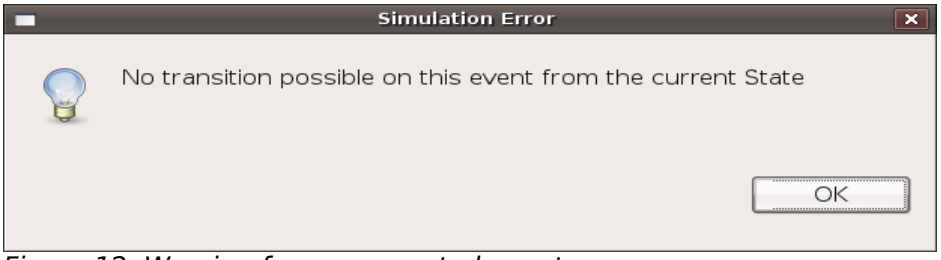


Figure 12: Warning for un-supported event

This is normal according to the use case since the first event should be *User inserts a card*.

When event *inserts a card* is selected, the behavior as specified by use case *log in* depend on whether the inserted card is unreadable or not (alternative **1.a**). Accordingly, the simulator asks which possibility should be considered by displaying the following

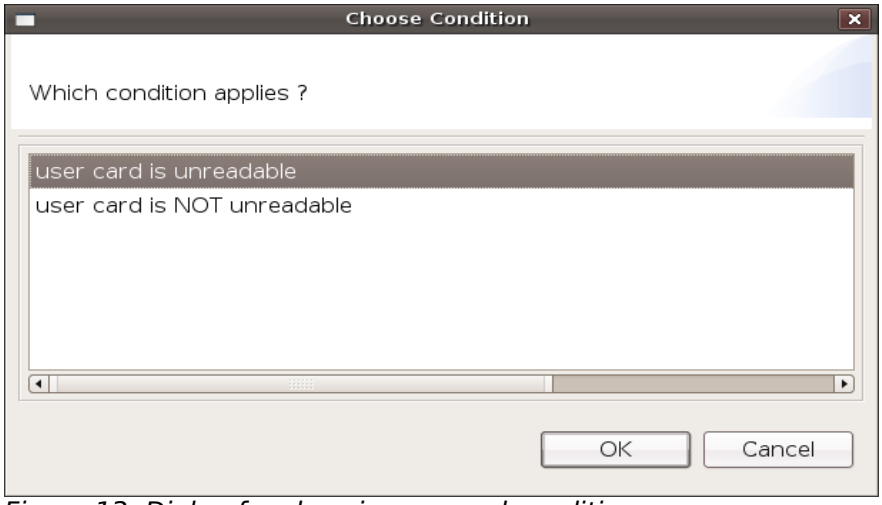


Figure 13: Dialog for choosing a guard condition

Choosing the first condition results in the following.

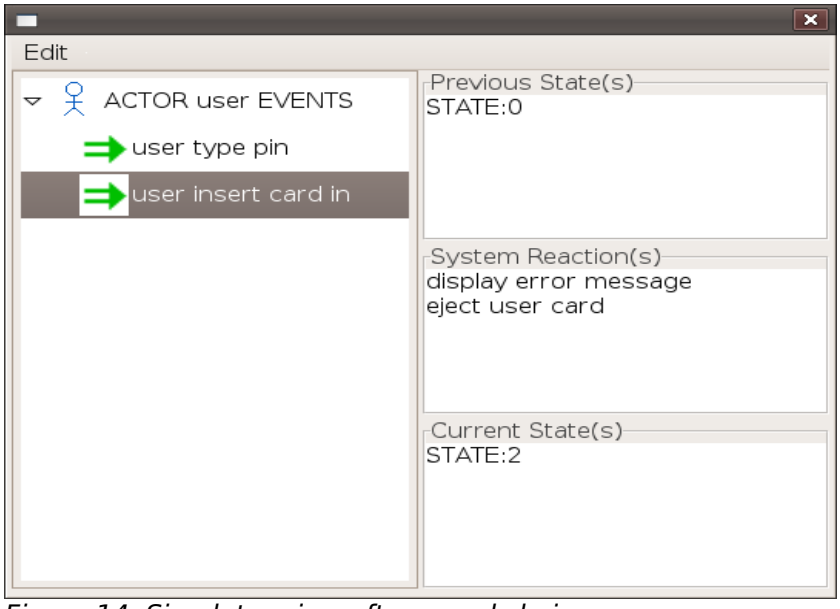


Figure 14: Simulator view after guard choice

8. Specification of operations

A part from use cases, an objective of UCEd is to help specify operations such that their implementation (according to the specification) would allow the realization of the use cases.
An operation specification includes:

- *pre-conditions*: a set of conditions that need to hold prior to the operation,
- *post-conditions* which include in turn
  - *added-conditions*: a set of conditions that become true after the operation,
  - *withdrawn-conditions*: a set of conditions that are removed after the operation.

Preconditions specify the necessary state for an operation. Added-conditions specify new state information and withdrawn-conditions specify parts of the state that stop being relevant after an operation. For instance, withdrawn-conditions may be used to express the fact that an entity state is re-initialized.

8.1 Operation effects identification

Some of the operations effects follow from an analysis of what operations are supposed to do. As an example, the ATM operation *display error message* intuitively should result in a modification of what is displayed to the user such at an error message is shown. The effect can be specified by:

1. introducing a new attribute to the ATM called ***display***[3]***,***
2. adding ***error message*** as a possible value of display,
3. adding condition ***ATM display is error message*** as an added condition of operation d*isplay error message*.

Similar effects can be added to operations *ask User pin* and *display operation menu.* The resulting domain model is as follow.

3 A sub-component is probably more appropriate.

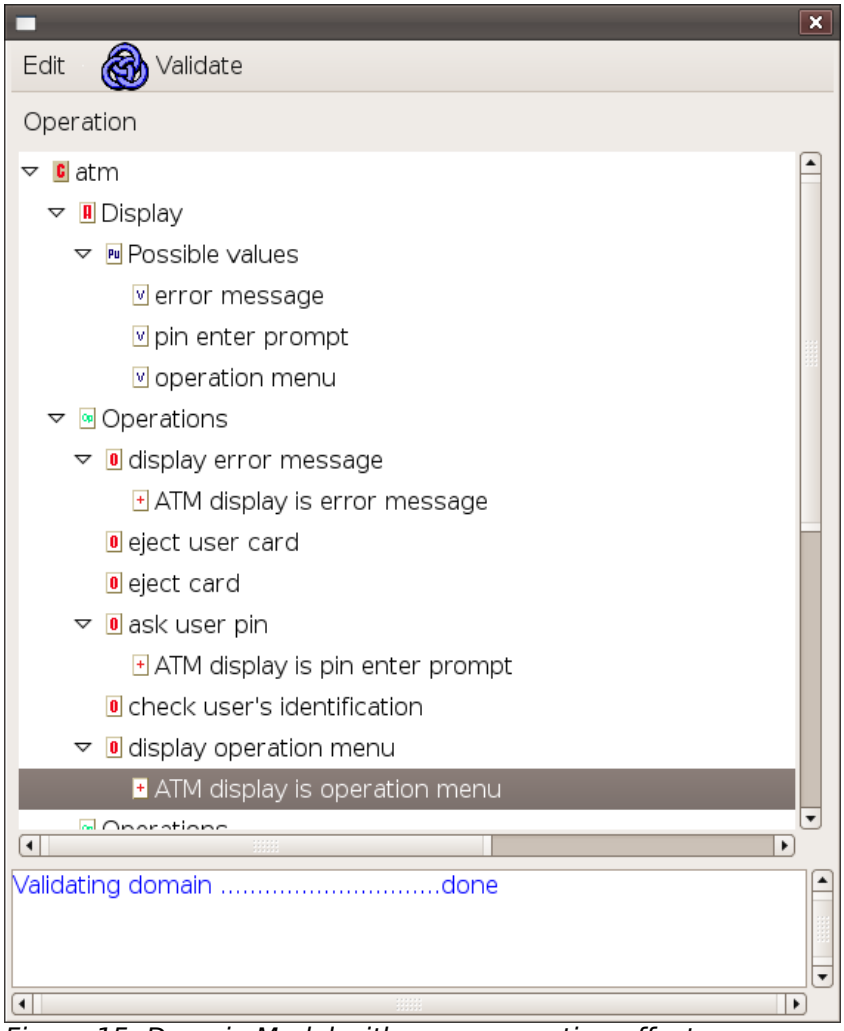


Figure 15: Domain Model with some operation effects

8.2 Operation effects validation

The validation of specified operation effects proceeds by (1) the generation of an operation-effect based state model and (2) the manual inspection and/or simulation of that state model.

To generate an operation-effect based state model for the ATM example with the modified domain, validate the use case model, then select ***Generation -> Generation based on Operation effects -> Add: log in***. Then, selects ***State Machine -> State Machine obtained from Operation effects -> View Global StateChart*** to view the generated StateChart.

The generation given the operations as specified results in an inconsistent state model (a StateChart can not be created) and the following message is displayed.

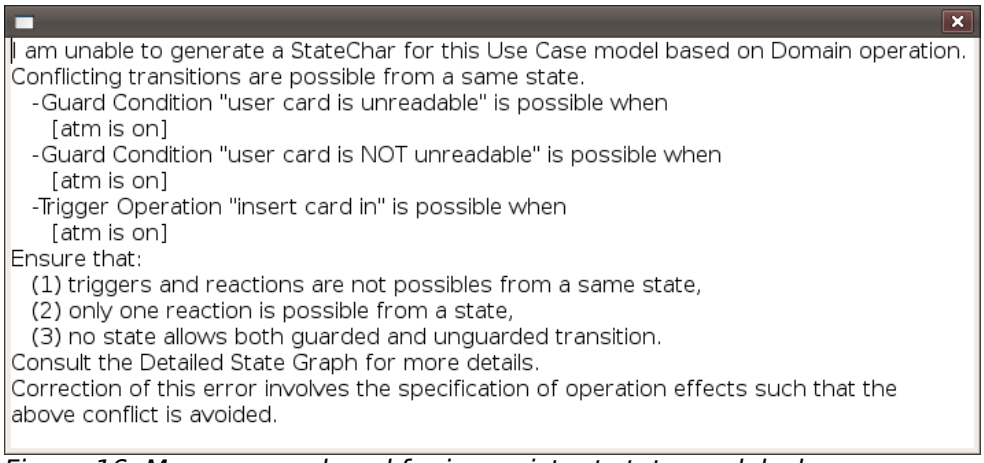


Figure 16: Message produced for inconsistent state model when attempting StateChart generation

A consistent state model should be such that:

(1) triggers and reactions are not possible from a same state,
(2) only one reaction is possible from a state,
(3) no state allows both guarded and unguarded transitions.

Situation (3) applies in the ATM example. Typically, a trigger event *insert card* is possible from the same state as guarded transitions.

Notice that a state model was generated but due to the inconsistency, a StateChart representation of this state model is not possible. The produced state model (in text form) is shown in Figure 17. It is displayed using menu option ***State Machine -> State Machine From Operation Effects -> View Detailed StateGraph***.

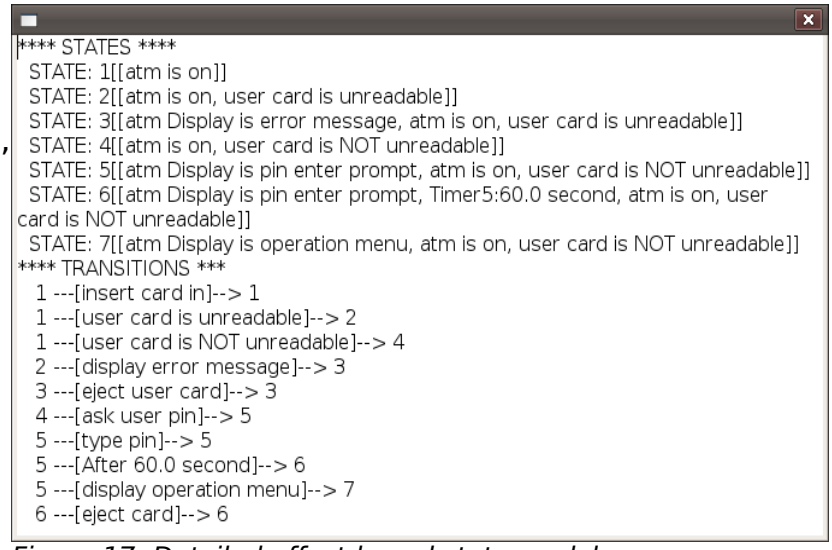


Figure 17: Detailed effect-based state model

The problem is due to lack of effects to User's operation *insert card* that translates in the above detailed state graph to the three ***non-deterministic*** transitions from state *1*.
We need to specify operation *insert card* such that its execution produces a change of state. The guards conditions *user card is unreadable* and *user card is NOT unreadable* would then be applied from that state and not conflict with operation *insert card****.***

As a general rule, actor's operations such as User's operation *insert card,* should have added-conditions specified to capture a ***transaction state***. In the ATM example, we will specify transaction states for the operations of actor User as follow.

1. Add an attribute to User, say ***transaction status.***
2. Specify possible values to the attribute such that different states of Users' transactions are captured. We define possible values ***card inserted*** and ***pin entered*** for that purpose.
3. Specify added conditions to User operations. Here we specify condition ***User transaction status is card inserted*** to operation ***insert Card,*** and condition ***User transaction status is pin entered*** to operation ***type pin.***

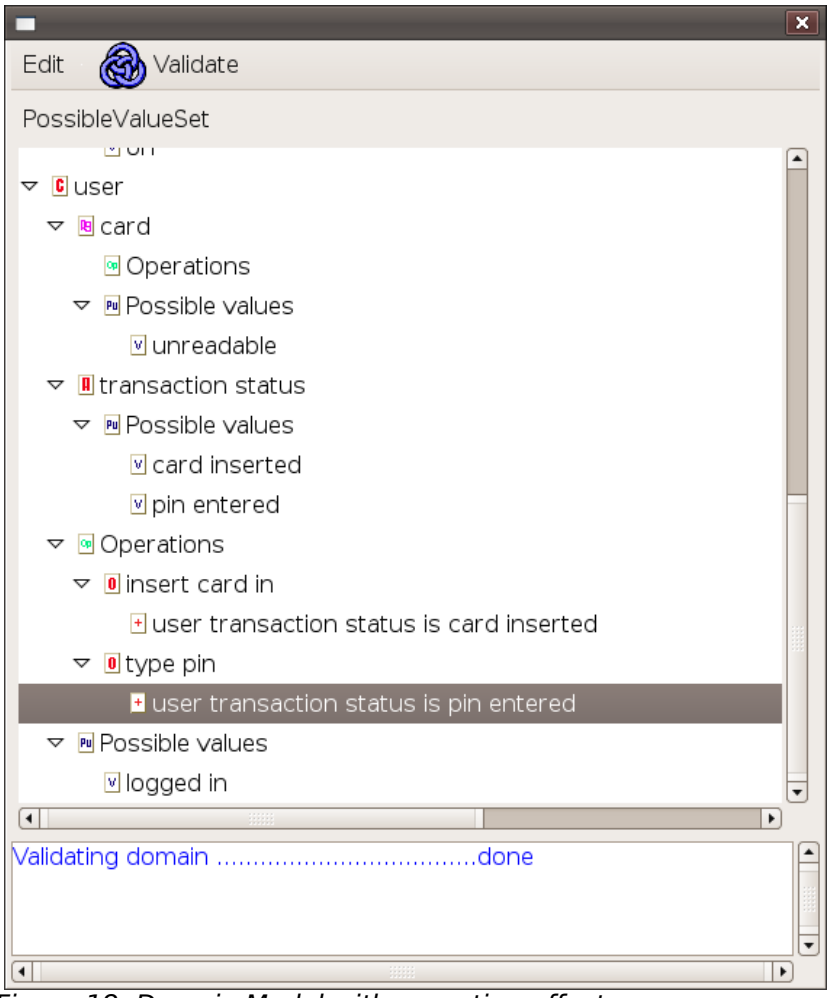


Figure 18: Domain Model with operation effects

The resulting domain model is shown in Figure 18.

We can attempt to generate a state model with the modified domain model. Reset the state model first to remove use case *log in* from the sate model by selecting ***Generation -> Generation based on Operation effects -> Reset State Machine***.
The resulting state model is still inconsistent as shown by the following message, which is displayed when attempting to view the StateChart.

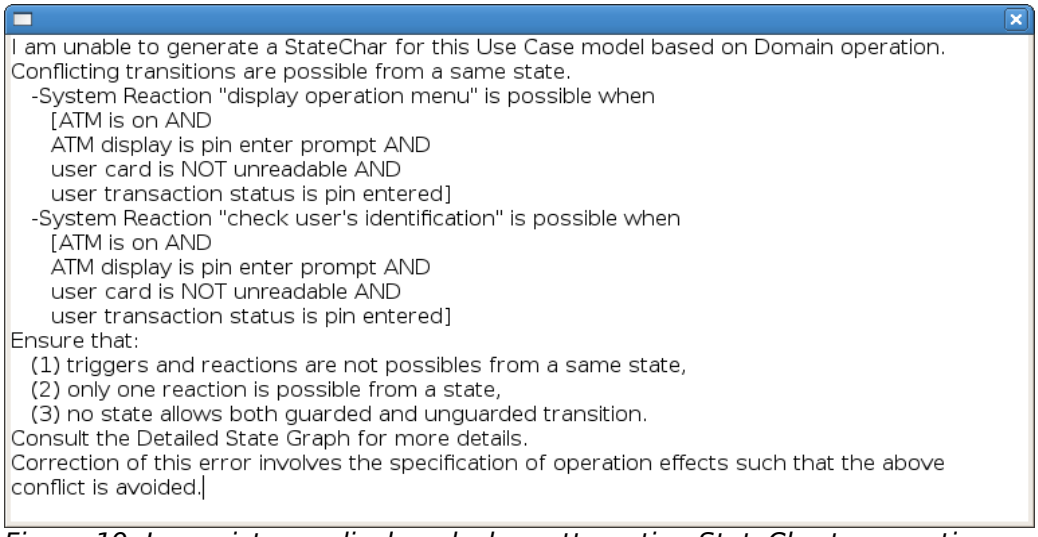


Figure 19: Inconsistency displayed when attempting StateChart generation

Below is the detailed state model.

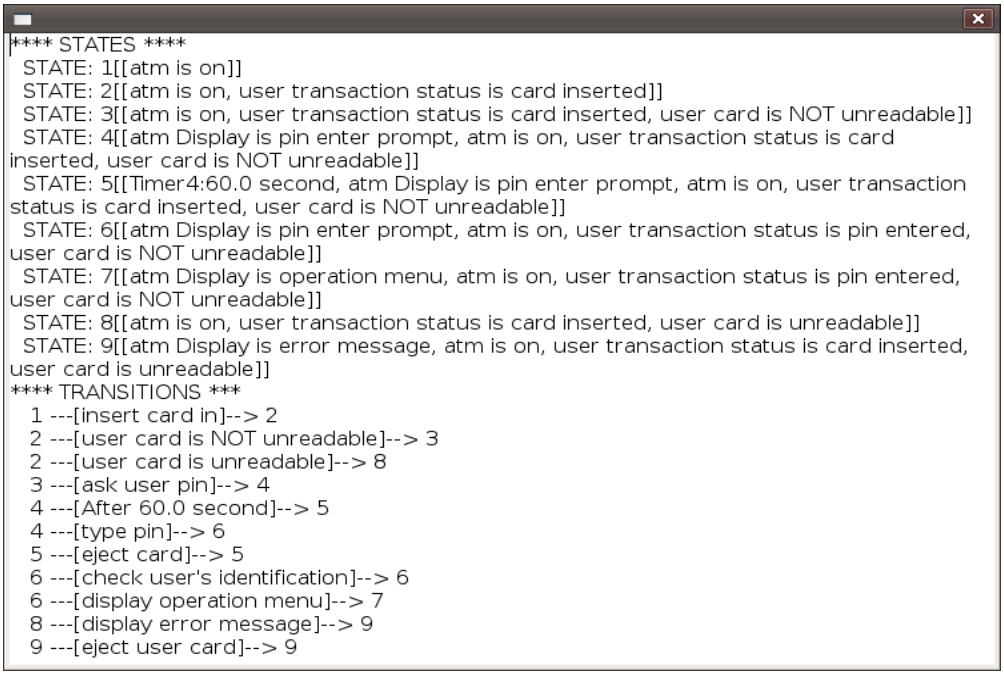


Figure 20: Detailed effect-based state model

We can observe non-deterministic transitions from state *6* where two system reactions (*check user's identification* and *display operation menu*) are conflicting. Operation *check user's identification* should produce a state change.
Intuitively checking of a user identification would result in a state where the user's identification is known as *valid* or *not valid*. We can

1. add an attribute *identification* to User,
2. add *valid* as a possible value of attribute *identification* of User,
3. add added-condition ***user identification is valid OR user identification is not valid*** to operation *check user's identification*

The modified domain model and use case are as follow.

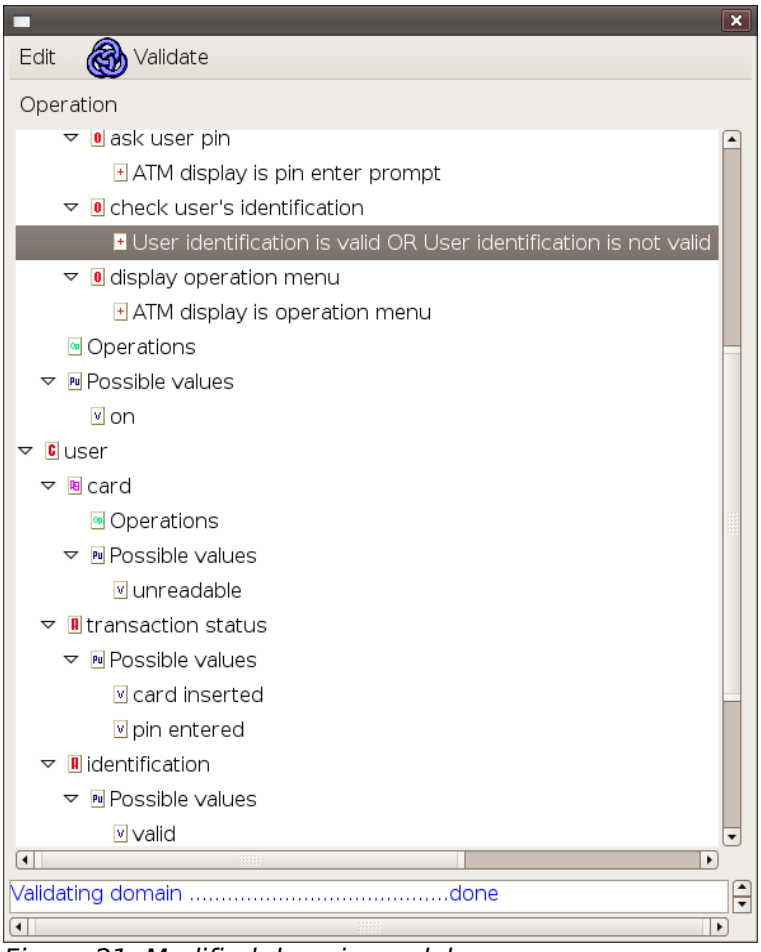


Figure 21: Modified domain model

Additionally, a guard needs to be added to step 5 to ensure operation *display operation menu* is executed only when the User identification is valid.

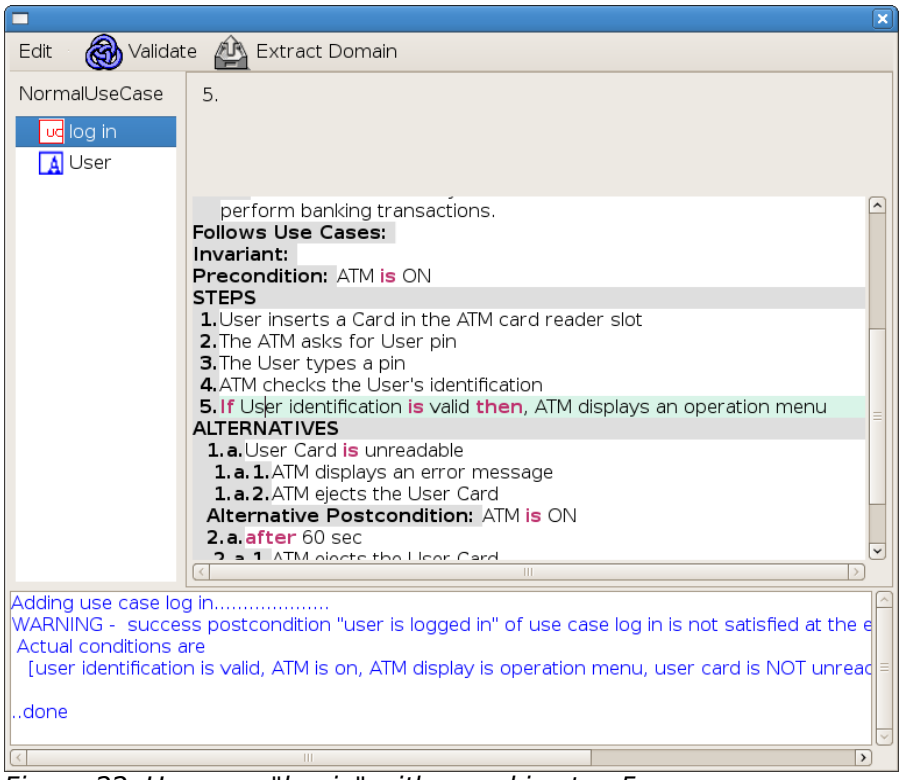


Figure 22: Use case "log in" with guard in step 5

The generated state model is now free of inconsistencies. StateChart generation from operation effects is now possible. Below is the StataChart obtained from use case "*log in*".

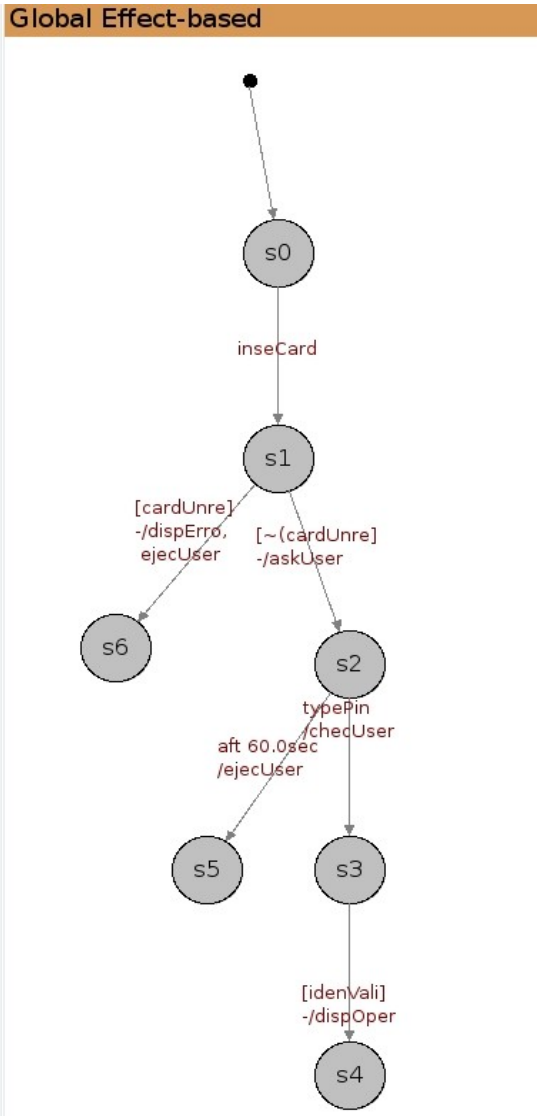


Figure 23: State model corresponding to use case in Figure 22

We notice the following warning when generating a state model based on operation effects.

WARNING - success postcondition "user is logged in" of use case log in is not satisfied at the end of scenario.
Actual conditions are
[user identification is valid, atm Display is operation menu, atm is on, user transaction status is pin entered, user card is NOT unreadable]

Effects need to be specified such that the use case success postcondition holds at the end of the primary scenario. Recall that use

case *log in* success postcondition is "*User is logged in".* State *S5* is the last state obtained from the main scenario. The conditions corresponding to a state may be displayed by hovering over it or by double clicking on the state when viewing the state model.

State *S5* corresponds to condition [*User identification is valid AND User card is NOT unreadable ATM is ON AND User transaction status is pin entered AND ATM display is operation menu*] (). we can ensure use case *log in* postcondition by adding condition "*User is logged in"* as an added condition to operation *display operation menu*. The following shows the state model generation results after this change.

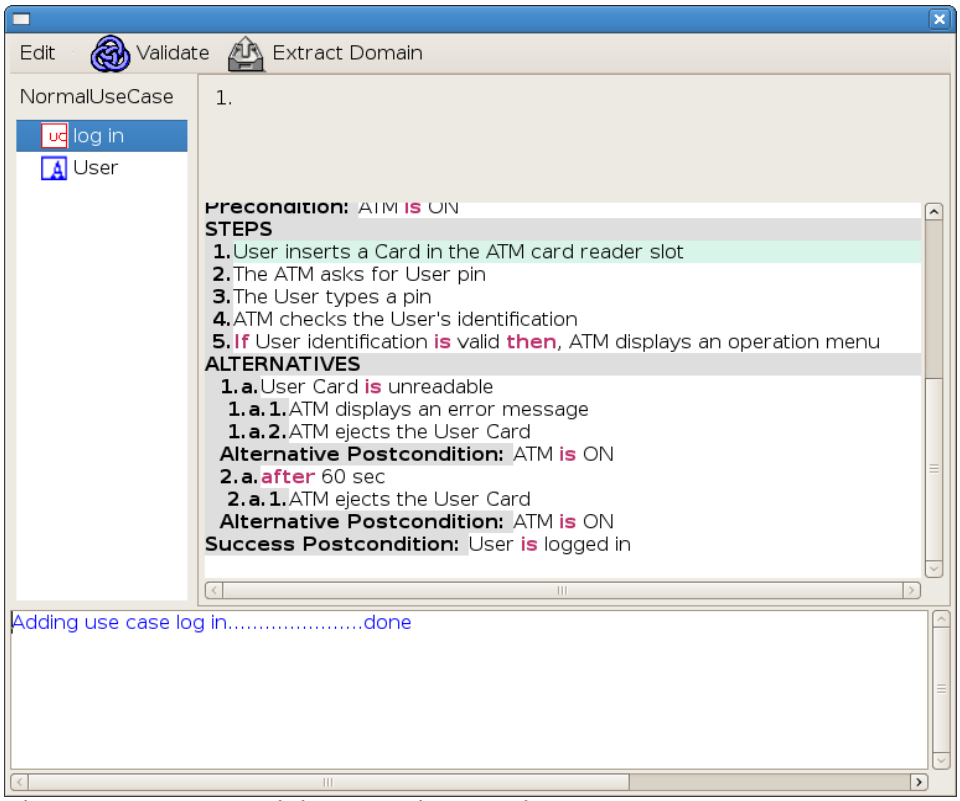


Figure 24: State model generation result

The state model inspection and simulation reveals other problems with operation effects.
For instance, when simulating the operation-effect based state machine, the following shows the simulator screen after selecting operation *user insert card*, condition *user card is NOT unreadable* and operation *user type pin*.

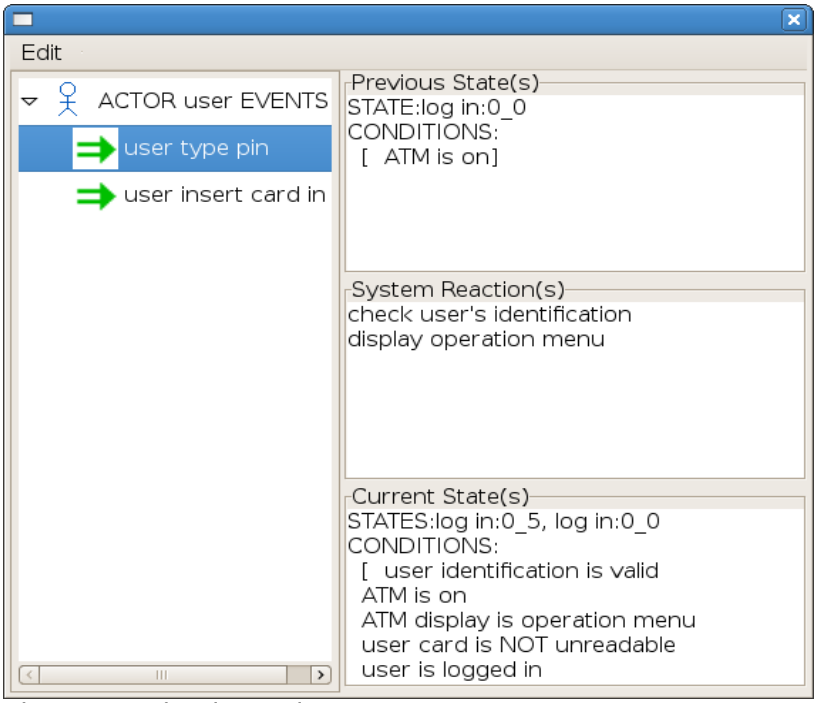


Figure 25: Simulator view

Figure 26 is obtained after the selection of operation *user insert card.*

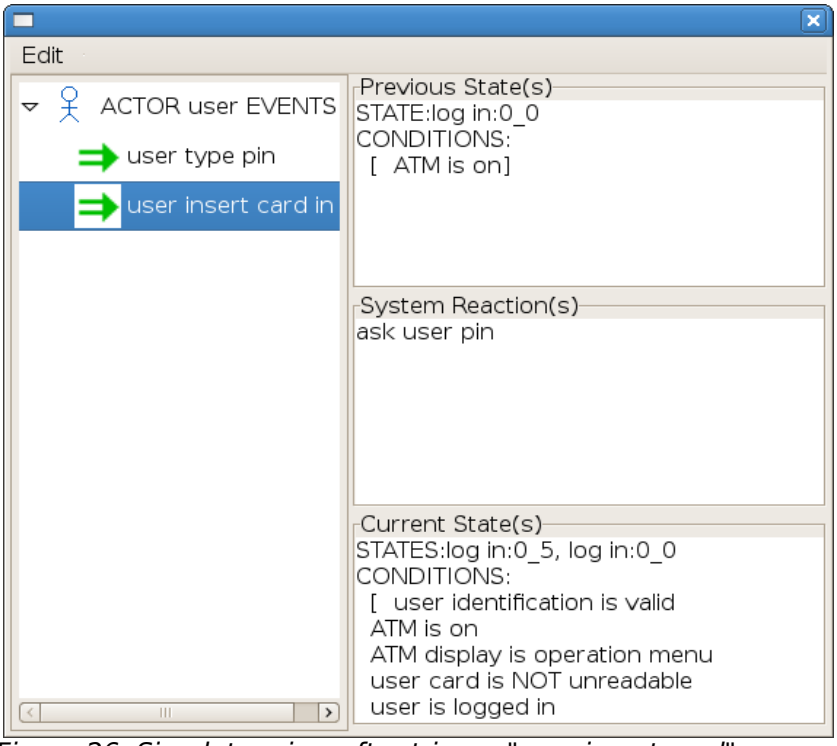


Figure 26: Simulator view after trigger "user insert card"

The operation shouldn't be accepted at this stage since the User's card has already been inserted and has not been ejected yet. The control-flow based state machine exhibits the correct behavior by

not allowing operation *user insert card* after operation *user insert card*, condition *user card is NOT unreadable* and operation *user type pin*.

We can note that state *S5* in the operation-effects based state machine (Figure 23) is a *sub-state* of state *S0*. State *S0* corresponds to condition [*ATM is on*] while state *S5* corresponds to [*user identification is valid AND ATM is on AND ATM display is operation menu AND user is logged in AND user card is NOT unreadable AND user transaction status is pin entered*]. Because of this relation, all transitions possible from state *S0* are also possible from state *S5***.**

In order to avoid the above mentioned behavior, the state obtained after *user insert card* followed by *user type pin* shouldn't be sub-state of *S0*. The specification can be corrected based on the observation that the pre-condition do not reflect the fact the User card is not inserted at the start of the use case. After a modification of the pre-condition to *ATM is ON AND User transaction status is not card inserted* followed by validation and generation of operation-effect based state machine, state *S0* now corresponds to [A*TM is on AND user transaction status is NOT card inserted*] and state *S5* is not sub-state of *S0* anymore. Operation *user insert card* is now denied after operation *user insert card*, condition *user card is NOT unreadable* and operation *user type pin*.

9. Use Case sequencing

Use cases are not always independent one from the other. There are sequential dependencies between use cases such that a use case execution may need that other use cases have been completed first. Sequencing may also concern the ability to repeat a use case after a particular scenario. As in the preceding discussion, use case sequencing should be elaborated first based on control-flow, before equivalent operation effects are introduced.

9.1 Use Case repetition

It should be possible to start use case "*log in*" over after User card ejection in steps ***1.a.2*** and ***2.a.1***. We use the **resume** statement to specify that the use case may repeat after steps **1.a.2** and **2.a.1.**

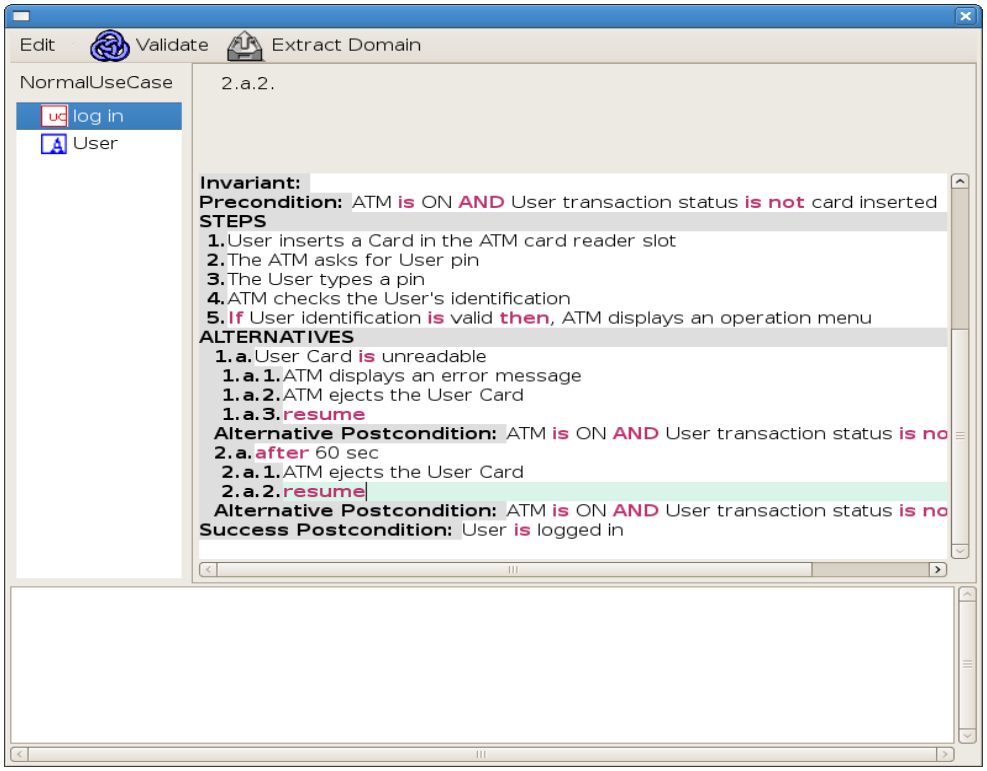


Figure 27: Use case "log in" with resume statements

Use case sequencing is reflected in control-flow generated StateChart-Charts as transitions between use case nodes.

The StateChart-Chart generated from use case “log in” includes transitions from state *S3* and *S6* to the use case state border corresponding to the two resume statements.

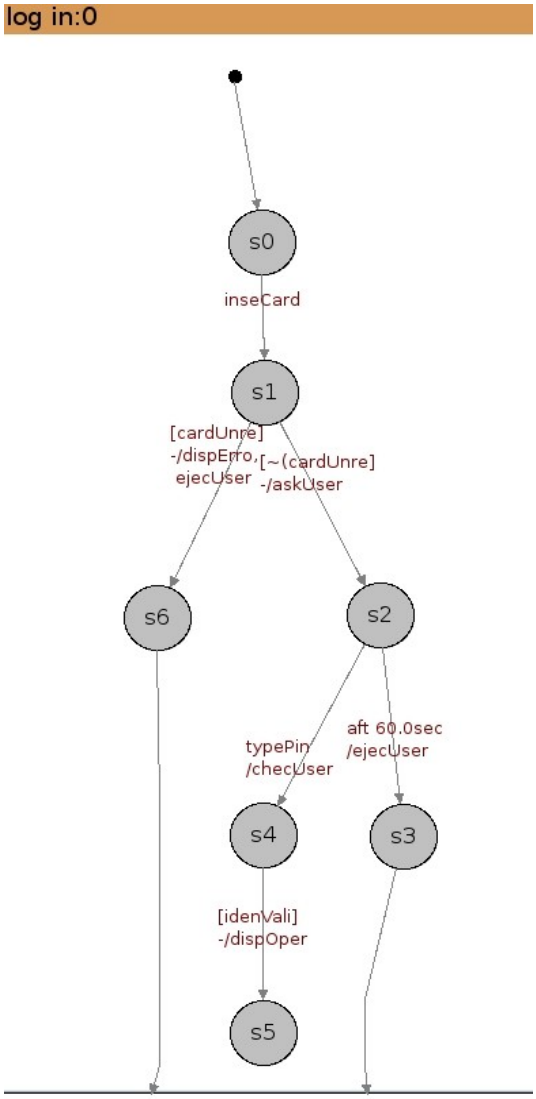


Figure 28: StateChart corresponding to use case "log in" showing transitions corresponding to 'resume' statements

In order that the operation-effect based StateChart corresponds to the control-flow based StateChart in Figure 28, the control-flow based state model in Figure 23 should be such that the system returns to state *S0* rather than going to states *S6* and *S3,* after user card ejection (operation *user eject card*).

Recall that state *S0* corresponds to condition [*ATM is on AND user transaction status is NOT card inserted*], state *S3* corresponds to condition [*ATM Display is pin enter prompt AND ATM is on AND User*

transaction status is card inserted AND User card is NOT unreadable] and state *S6* corresponds to [*ATM Display is error message, ATM is on AND User transaction status is card inserted, User card is unreadable*]. Operation *eject User Card* post-conditions should be such that the condition corresponding to *S0* is obtained.
Therefore, the operation needs to withdraw conditions ***ATM Display is pin enter prompt, User transaction status is card inserted, User card is NOT unreadable, User Card is unreadable and ATM display is error message*** and needs to add condition ***User transaction status is not card inserted***.

We specify ***User transaction status is not card inserted*** as an added-condition and conditions ***ANY ON User**** and ***ANY ON ATM display*** as withdrawn-conditions to operation *eject User Card.* The withdrawn-conditions state that all conditions on entity *User* (as well as sub-entities of User), and all conditions on entity *ATM display* are to be removed after the operation execution.

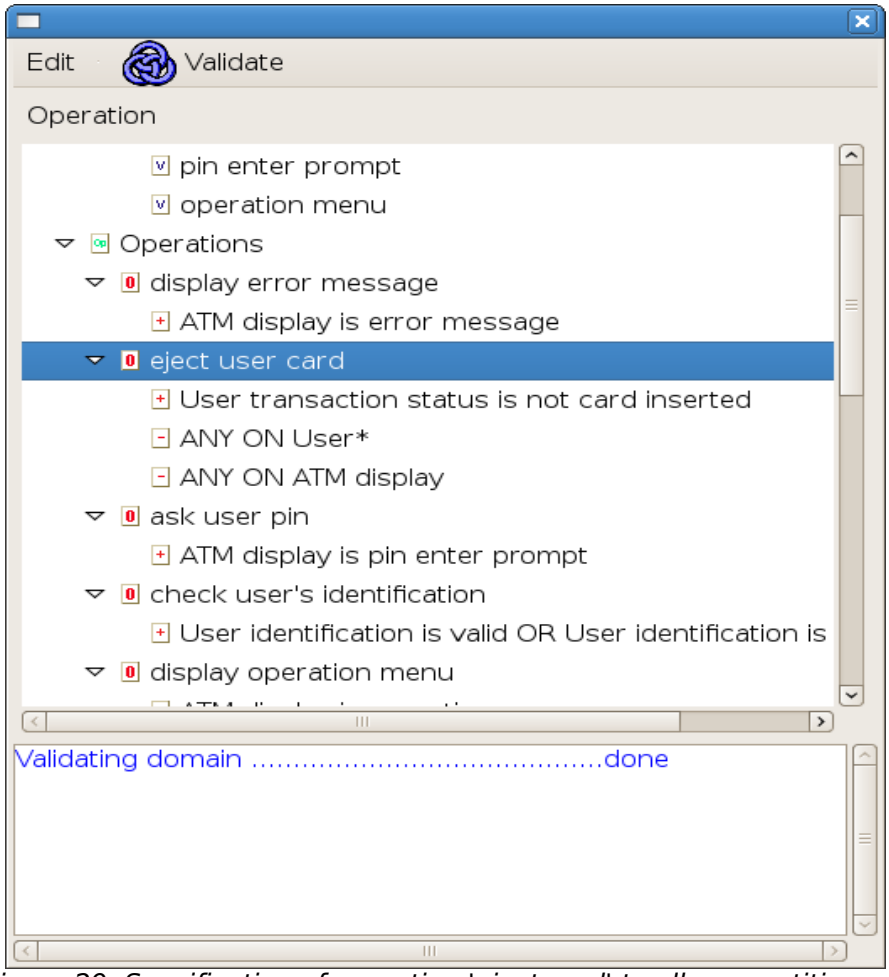


Figure 29: Specification of operation 'eject card' to allow repetition

The following is the resulting state chart. The system returns now to state *S0* after operation *eject Card*.

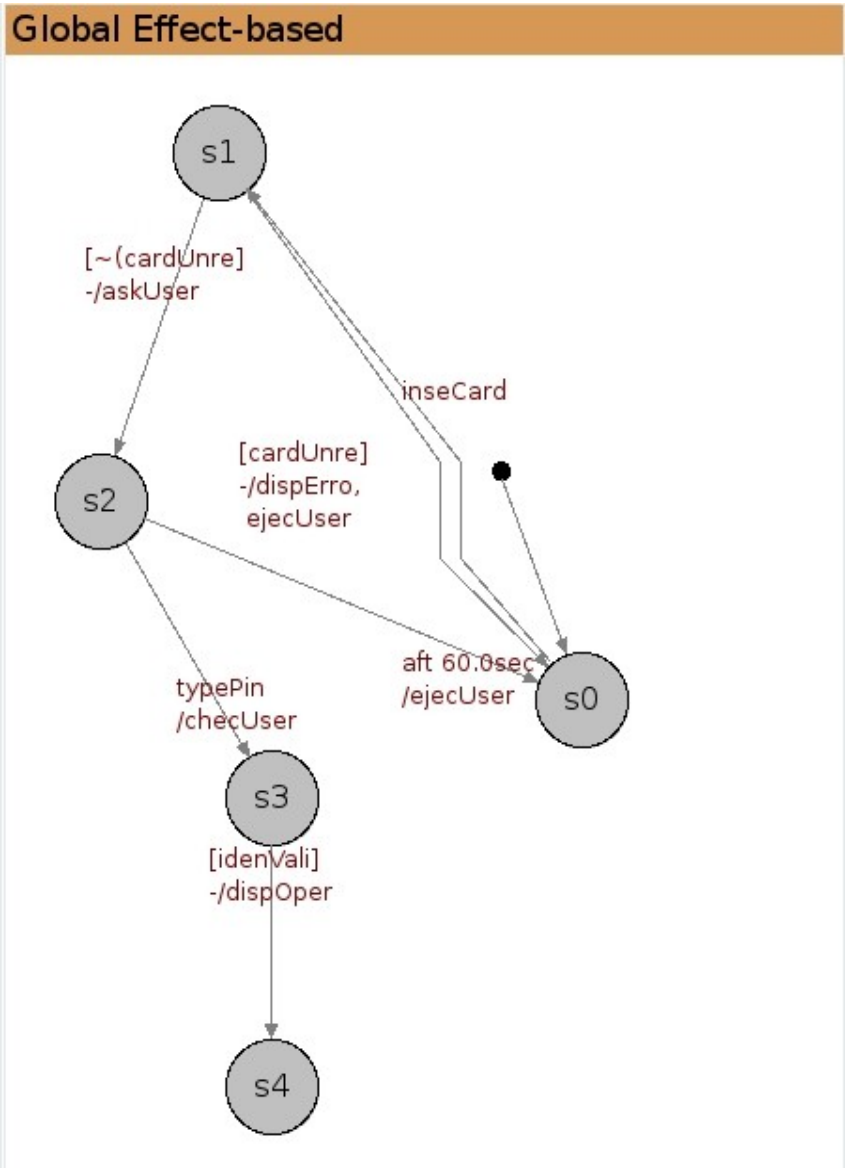


Figure 30: Effect-based state model

9.2 Use Cases integration

Once a use case definition is satisfactory (i.e the use case is deemed valid by inspection/simulation), use cases development process may proceed with definition of additional use cases.
In the ATM example, there are several other use cases such as *withdraw cash*, *make deposit*, *transfer funds*, ...
We suggest that each use case be defined independently prior to integration.

1. Define each use case from a fresh state model following the approach discussed .
2. Integrate the use cases. We suggest an *incremental* integration approach where use cases are integrated one at the time or in small subsets.

The following flowchart describes the use case integration process starting from a set of use cases and a domain model.

The process begins with a set of use cases and a domain model. Control-flow sequencing between the use cases are first specified using use cases ***follow lists*** and ***enabling directives***.

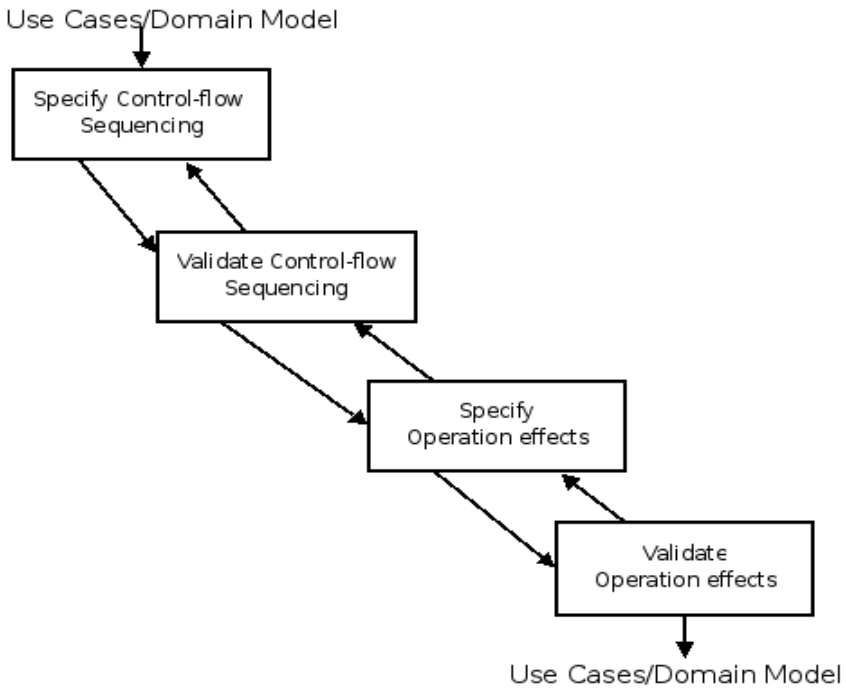


Figure 31: Use Case sequencing elaboration process

The control-flow sequencing is validated by inspection/simulation of a generated control-flow based StateChart-Chart. Following control-flow validation, operation effects are specified such that a generated effect-based state-model is deemed satisfactory by comparison to the control-flow based StateChart-Chart.

For instance suppose use cases "*turn ATM on*", "*turn ATM off*", "*withdraw cash*" and "*make deposit*" are defined for the ATM application in addition to use case "log in".

Title: turn ATM on **Description**: **System Under Design**: ATM **Primary Actor**: Operator **Participants**: **Goal**: Allows an Operator to start the ATM up so that it could provide transaction services to Users. **Follows Use Cases**: **Invariant**: **Precondition**: ATM is OFF **STEPS** 1.The Operator turns the system ON 2.The ATM asks the amount in the cash dispenser 3.The Operator enters the amount of money currently in cash dispenser 4.The ATM displays a welcome message **Success Postcondition**:	**Title**: turn ATM off **Description**: **System Under Design**: ATM **Primary Actor**: Operator **Participants**: **Goal:** Allows an Operator to switch the ATM off. Transaction services are not provided anymore following the use case. **Follows Use Cases**: **Invariant**: **Precondition**: ATM is ON **STEPS** 1.The Operator turns the system off 2.The ATM clears the system **Success Postcondition**:
Title: withdraw cash **Description**: **System Under Design:** ATM **Primary Actor**: User **Participants**: **Goal**: Allow a User to get a cash amount by deduction from his/her account. **Follows Use Cases**: **Invariant**: **Precondition**: ATM is ON AND ATM Display is operation menu **STEPS** 1.The User selects cash withdrawal 2.The ATM asks the withdrawal amount 3.The User enters an amount 4.The ATM asks the customer account update 5.ATM provides cash in the cash compartment 6.The User takes the cash from the cash compartment 7.The ATM ejects the user card **Success Postcondition**:	**Title:** make deposit **Description:** **System Under Design**: ATM **Primary Actor**: User **Participants**: **Goal**: Allows a User to make a money deposit to his/her account. **Follows Use Cases**: **Invariant:** **Precondition**: ATM is ON AND ATM Display is operation menu **STEPS** 1.The User selects cash deposit 2.The ATM asks for a deposit amount 3.The User specifies a deposit amount 4.The ATM asks the user to insert a deposit 5.The User inserts a deposit 6.The ATM updates the User's account 7.The ATM ejects the user card **Success Postcondition**:

Table 1: Use cases in the ATM System without sequencing constructs

For sake of simplicity we are restricting these use cases to their main scenario.

9.2.1 Control-flow sequencing

Control-flow sequencing is specified using use case *follow lists* and *enabling directives*.

A use case follow list specifies which use cases precede that use case and how these use cases are synchronized. Two operators: AND and OR are used.

- If a use case *uc0* follow list is expressed as "*uc1 AND uc2 AND ... ucN*". All of the use cases *uc1, uc2, ... ucN* need to reach a point where they enable use case *uc0* in order for uc0 to be executed.
- If a use case *uc0* follow list is expressed as "*uc1 OR uc2 OR ... ucN*". Use case *uc0* can execute as soon as any of use cases *uc1, uc2, ... ucN* reaches a point where *uc0* is enabled.

An enabling directive specifies which use cases may execute from a given point of a use case scenario and whether or not these use cases execute concurrently.

- After enabling directive "*enable uc1, uc2, ... ucN*", one and only one of use cases among *uc1, uc2, ... ucN* may execute.
- After enabling directive "*enable in parallel uc1, uc2, ... ucN*", all of use cases *uc1, uc2, ... ucN* may execute concurrently with the others.

In the ATM example, suppose an analysis determined the following sequencing constraints:

(1) Use case "log in" may execute after the primary scenario of use cases "turn ATM on", "withdraw cash" or "make deposit".

(2) The primary scenario of use case "log in" must be completed before use cases "withdraw cash" and "make deposit".

(3) Use cases "withdraw cash" and "make deposit" execute alternatively. Meaning only one of these 2 use cases execute at a time.

(4) Use case "turn ATM off" may follow any of "turn ATM on", "withdraw cash" and "make deposit".

(5) Use case "turn ATM on" may follow the primary scenario of "turn ATM off".

Following are use cases "*turn ATM on*", "*turn ATM off*", "*withdraw cash*" and "*make deposit*" with follow lists and enabling directives to reflects the above sequencing constraints.

Title: turn ATM on **Description**: **System Under Design**: ATM **Primary Actor**: Operator **Participants**: **Goal**: Allows an Operator to start the ATM up so that it could provide transaction services to Users. **Follows Use Cases**: turn ATM off **Invariant**: **Precondition**: ATM is OFF **STEPS** 1.The Operator turns the system ON 2.The ATM asks the amount in the cash dispenser 3.The Operator enters the amount of money currently in cash dispenser 4.The ATM displays a welcome message 5.enable log in, turn ATM off **Success Postcondition**:	**Title**: turn ATM off **Description**: **System Under Design**: ATM **Primary Actor**: Operator **Participants**: **Goal:** Allows an Operator to switch the ATM off. Transaction services are not provided anymore following the use case. **Follows Use Cases**: turn ATM on OR withdraw cash OR make deposit **Invariant**: **Precondition**: ATM is ON **STEPS** 1.The Operator turns the system off 2.The ATM clears the system 3.enable turn ATM on **Success Postcondition**:
Title: withdraw cash **Description**: **System Under Design:** ATM **Primary Actor**: User **Participants**: **Goal**: Allow a User to get a cash amount by deduction from his/her account. **Follows Use Cases**: log in **Invariant**: **Precondition**: ATM is ON AND ATM Display is operation menu **STEPS** 1.The User selects cash withdrawal 2.The ATM asks the withdrawal amount 3.The User enters an amount 4.The ATM asks the customer account update 5.ATM provides cash in the cash compartment 6.The User takes the cash from the cash compartment 7.The ATM ejects the user card 8.enable log in, turn ATM off **Success Postcondition**:	**Title:** make deposit **Description:** **System Under Design**: ATM **Primary Actor**: User **Participants**: **Goal**: Allows a User to make a money deposit to his/her account. **Follows Use Cases**: log in **Invariant:** **Precondition**: ATM is ON AND ATM Display is operation menu **STEPS** 1.The User selects cash deposit 2.The ATM asks for a deposit amount 3.The User specifies a deposit amount 4.The ATM asks the user to insert a deposit 5.The User inserts a deposit 6.The ATM updates the User's account 7.The ATM ejects the user card 8.enable log in, turn ATM off **Success Postcondition**:

Table 2: Use Cases in the ATM System with sequencing constructs

Use case “log in” is shown in Figure 32. A StateChart-Chart is generated from the use cases using ***Generation -> Generation based on Use Case flow -> Generate StateChart Chart***.The generation proceeds fine as shown in Figure 32.

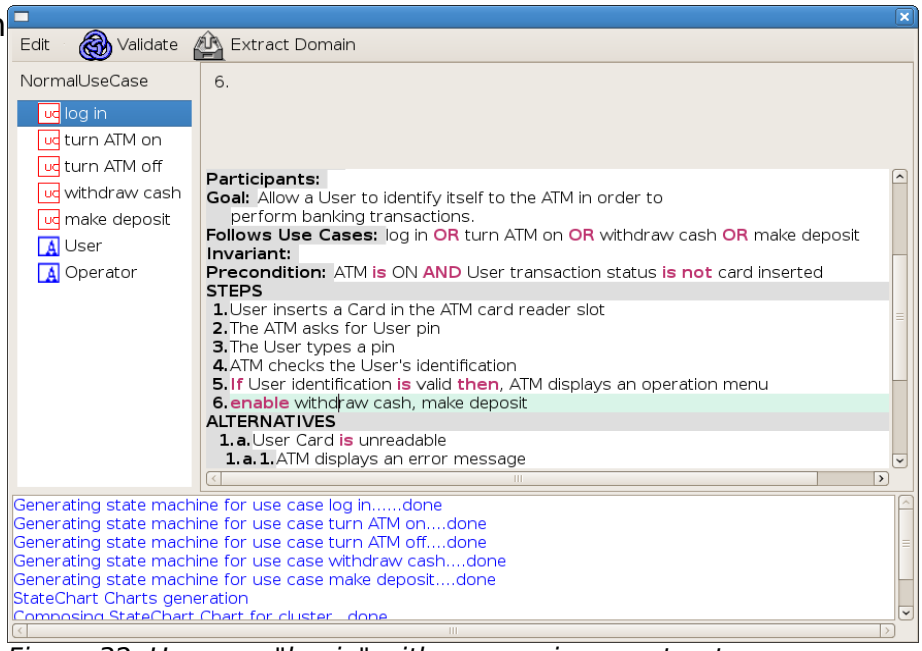


Figure 32: Use case "log in" with sequencing constructs

The generated StateChart Chart is displayed using ***State Machine -> State Machine obtained from Use Case flow -> View StateChart Chart***.

Figure 33 shows the generated StateChart-Chart. Each use case corresponds to a node and these nodes are connected either

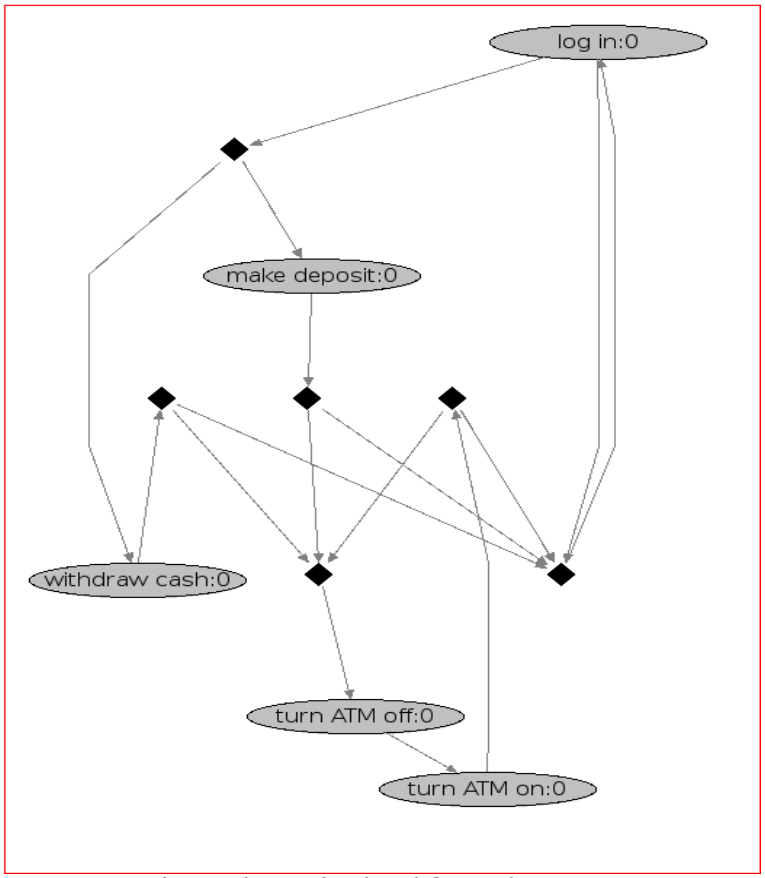
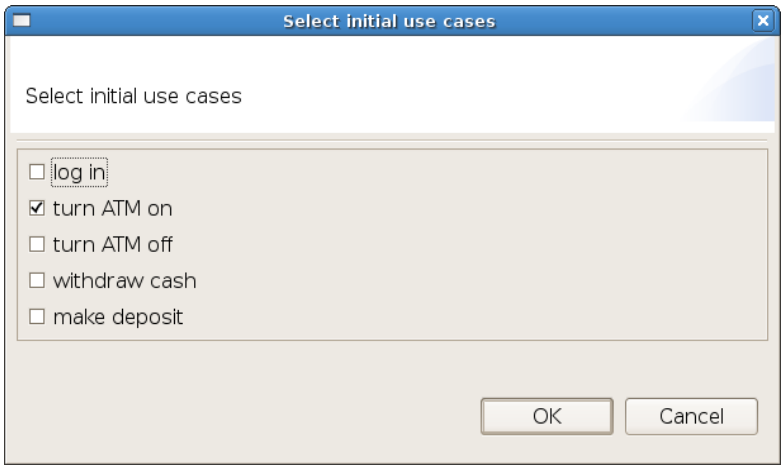


Figure 33: StateChart-Chart obtained from the ATM use cases

directly as “turn ATM off” and “turn ATM on” or through flow nodes. In this example, only decision/merge nodes appear.

Sequencing can be validated by simulating the generated StateChart-Chart. Prior to simulation, the default initial use case may be changed using ***Simulation -> State Model obtained from Use Case flow -> Simulate StateChart Chart -> Set Initial State***.



Use Case “turn ATM on” is selected here as initial use case. The simulation will start with this use case enabled.

9.2.2 Operation-effects based sequencing

Once control-flow sequencing has been validated, the domain model is updated such that state models based on operation effects allow the same sequencing between use cases as state models generated based on control-flow.

We suppose the process described in the previous sections has been followed for each use case. A domain model has been developed such that the operation effect-based state model corresponding to each use case in isolation is deemed valid by inspection/simulation. Table 3 shows an example of domain model with the required properties.

```
domain
  System Concept:ATM
      Attribute:display
          Possible Values Set
              Value:error message
              Value:pin enter prompt
              Value:operation menu
              Value:amount in dispenser prompt
              Value:welcome message
              Value:withdrawal amount
              Value:deposit amount prompt
              Value:deposit insertion prompt
      Attribute:transaction status
          Possible Values Set
              Value:customer account update
              Value:user account updated
              Value:user cash provided
      Operation Set
          Operation:display error message
              AddedCondition:ATM display is error message
          Operation:eject user card
              AddedCondition:User transaction status is not card
                             inserted
              WithdrawCondition:ANY ON User*
              WithdrawCondition:ANY ON ATM display
          Operation:ask user pin
              AddedCondition:ATM display is pin enter prompt
          Operation:check user's identification
              AddedCondition:User identification is valid or User
                             identification is not valid
          Operation:display operation menu
              AddedCondition:ATM display is operation menu
              AddedCondition:User is logged in
          Operation:ask amount in cash dispenser
              AddedCondition:ATM display is amount in dispenser prompt
          Operation:display welcome message
              AddedCondition:ATM display is welcome message
          Operation:clear system
              AddedCondition:ATM is OFF
              WithdrawCondition:ANY ON Operator*
              WithdrawCondition:ANY ON ATM*
              WithdrawCondition:ANY ON User*
          Operation:ask withdrawal amount
              AddedCondition:ATM display is withdrawal amount
          Operation:ask customer account update
              AddedCondition:ATM transaction status is customer account
                             update
          Operation:ask deposit amount
              AddedCondition:ATM display is deposit amount prompt
          Operation:ask user insert deposit
              AddedCondition:ATM display is deposit insertion prompt
          Operation:update user's account
```

```
            AddedCondition:ATM transaction status is user account
                           updated
        Operation:provide cash
            AddedCondition:ATM transaction status is user cash
                           provided
    Possible Values Set
        Value:on
        Value:off
Concept:user
    Sub Component:card
        Operation Set
        Possible Values Set
            Value:unreadable
    Attribute:transaction status
        Possible Values Set
            Value:card inserted
            Value:pin entered
            Value:cash withdrawal selected
            Value:deposit amount specified
            Value:cash deposit selected
            Value:deposit inserted
            Value:amount entered
            Value:cash taken
    Attribute:identification
        Possible Values Set
            Value:valid
    Operation Set
        Operation:insert card in
            AddedCondition:User transaction status is card inserted
        Operation:type pin
            AddedCondition:User transaction status is pin entered
        Operation:select cash withdrawal
            AddedCondition:User transaction status is cash withdrawal
                           selected
        Operation:enter amount
            AddedCondition:User transaction status is amount entered
        Operation:take cash
            AddedCondition:User transaction status is cash taken
        Operation:select cash deposit
            AddedCondition:User transaction status is cash deposit
                           selected
        Operation:specifie deposit amount
            AddedCondition:User transaction status is deposit amount
                           specified
        Operation:insert deposit
            AddedCondition:User transaction status is deposit inserted
    Possible Values Set
        Value:logged in
Concept:operator
    Attribute:transaction status
        Possible Values Set
            Value:system turned on
```

```
            Value:amount in cash entered
            Value:system turned off
    Operation Set
        Operation:turn system on
            Precondition:ATM is OFF
            AddedCondition:ATM is ON
            AddedCondition:Operator transaction status is system
                           turned on
        Operation:enter amount of money currently in cash dispenser
            AddedCondition:Operator transaction status is amount in
                           cash entered
        Operation:turn system off
            AddedCondition:Operator transaction status is system
turned off
```

Table 3: Sample domain model allowing operation-effect validation of the Use Cases in the ATM System

Operation-effects based sequencing is achieved by specifying use cases pre and post-conditions such that when a use case *uc1* is followed by a use case *uc0* after a given scenario *sc0*, the post-condition at the end of *sc0* implies use case *uc1* pre-condition.

UCEd verification includes checking that sequencing statements are consistent with pre/post-conditions. Figure 34 shows use case sequencing warnings resulting from the ATM system.

(1) The enabling of use cases “withdraw cash” and “make deposit” at step *6* of use case “log in” is not consistent with pre/post-conditions as use case “log in” primary scenario post-condition is “*user is logged in*” and both “withdraw cash” and “make deposit” have “*(ATM is on AND ATM display is operation menu)*” as pre-condition.

(2) No post-condition is defined for the main scenario of use case “turn ATM on” however, the enabling of use case “log in” and “turn ATM off” at the end of the scenario is such that the post-condition should be “*(ATM is on AND user transaction status is NOT card inserted)*”.

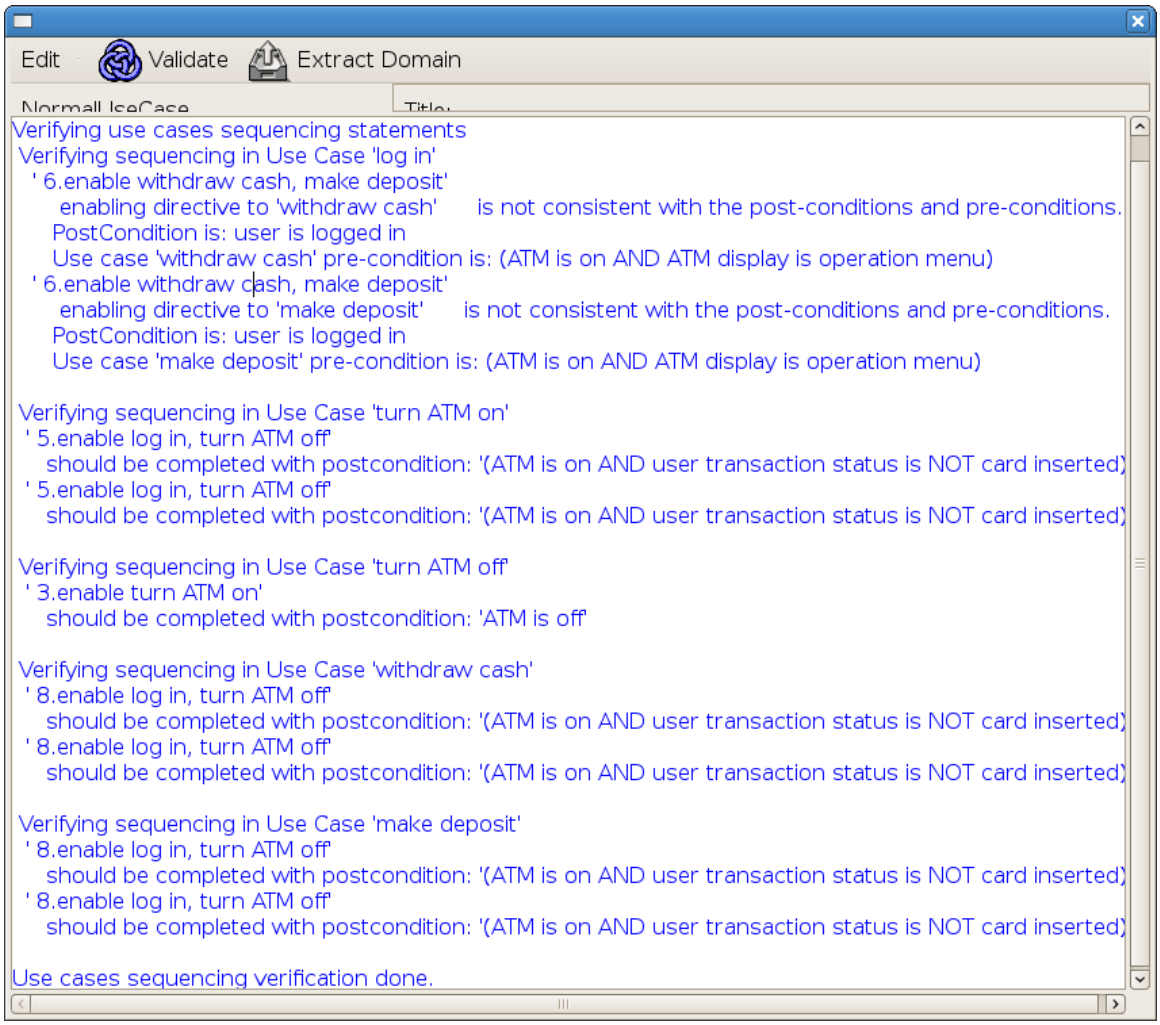


Figure 34: Use Case sequencing verification results for the ATM system

(3) Similarly, no post-condition is defined for the primary scenario of use case "turn ATM off" but, in accordance with the enabling of use case "turn ATM on", the post-condition should be "*ATM is off*".

(4) Finally, no post-condition are defined for the primary scenarios of use cases "withdraw cash" and "make deposit". However, because each of these use cases enable use cases "log in" and "turn ATM off", they should both have "(*ATM is on AND user transaction status is NOT card inserted)*" as post-condition.

In order to remove these validation messages:

- We change use case "*log in*" success post-condition to "*User is logged in AND ATM is ON AND ATM display is operation menu*" and we add condition "*User is logged in*" to use cases "withdraw cash" and "make deposit" pre-condition.

- We add "*ATM is on AND user transaction status is NOT card inserted*" as success post-condition to use case "turn ATM on".
- We add "*ATM is OFF"* as success post-condition to use case *"turn ATM off*".
- And we add "ATM is on AND user transaction status is NOT card inserted" as success post-condition to use cases "withdraw cash" and "make deposit".

Figure 35 shows the sequencing validation results after performing these changes. No warning is produced.

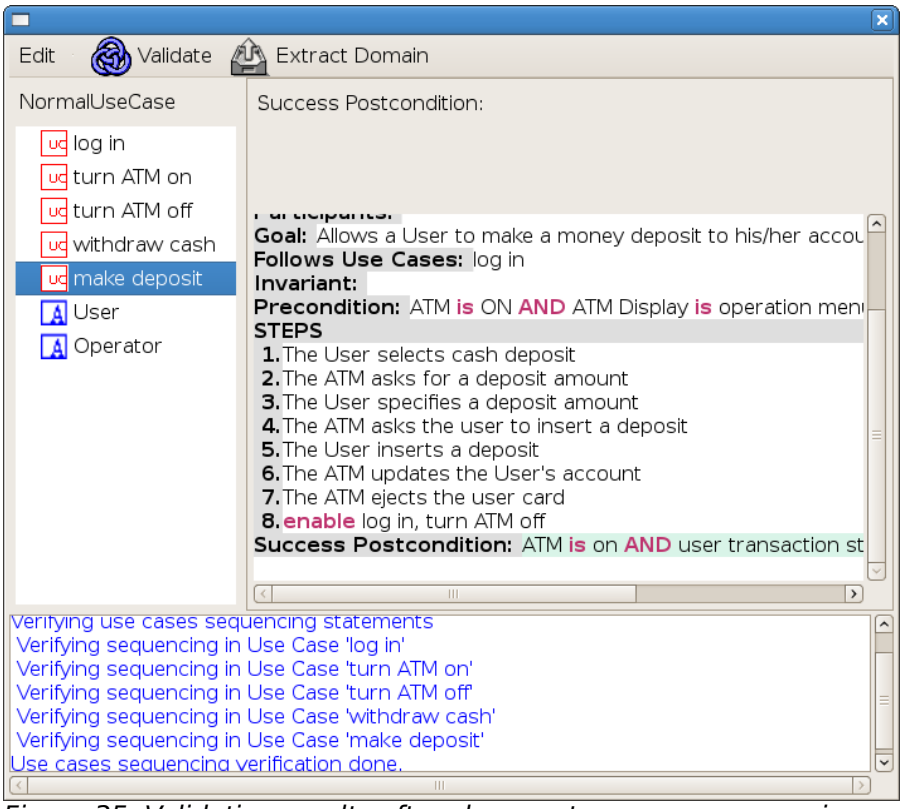


Figure 35: Validation results after changes to remove sequencing warnings

Figure 36 shows the global effect-based state model generated from the ATM use cases by considering the domain model in Table 3. States *S0* to *S4* correspond to use case "log in", states *S9* to *S15* correspond to use cases "withdraw cash" and "make deposit", and states *S9* to *S7* correspond to "turn ATM on" and "turn ATM off".
State *S0* corresponds to use case "*log in"* pre-condition, state *S4* corresponds to use case "log in" success post-condition, state *S9* corresponds to use cases "withdraw cash" and "make deposit" pre-conditions, state *S12* corresponds to "withdraw cash" success post-condition, state *S15* corresponds to "make deposit" success post-condition, state *S8* corresponds to "turn ATM off" pre-condition, state *S5* corresponds to "turn ATM off" success post-condition and "turn ATM on"

pre-condition, and finally, state *S7* corresponds to "*turn ATM on*" success post-condition.

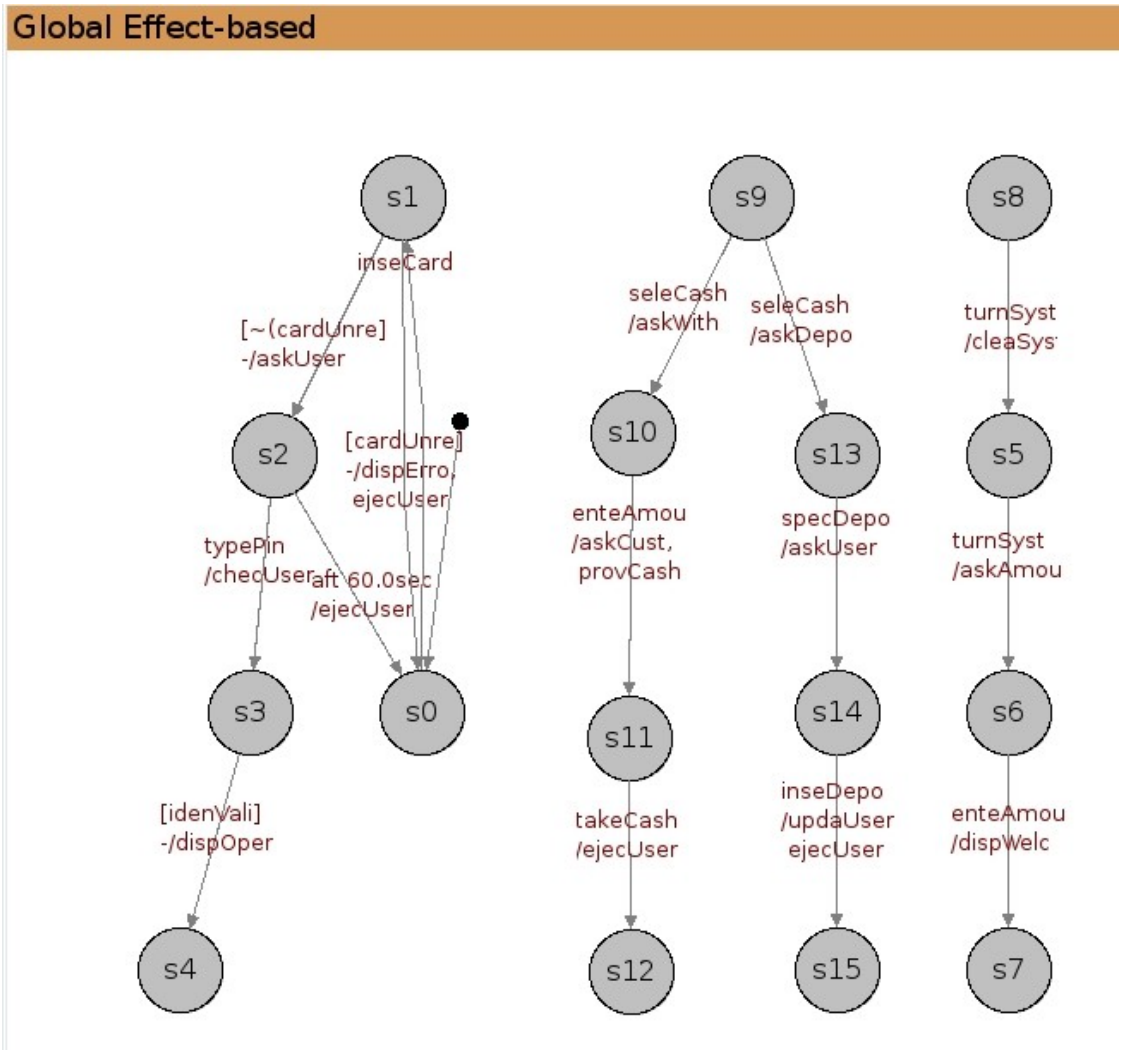


Figure 36: Global effect-based state model obtained with domain model in Table 3

Although the graphical depiction shows an unconnected state model, there is actually a connection when we consider the conditions corresponding to states corresponding to pre/post-conditions.
For instance, state *S4* is a sub-state of *S9* (the condition corresponding to *S4* subsumes the condition corresponding to *S9*). Therefore, all transitions from *S9* are possible from *S4* and consequently use case "log in" is followed by "withdraw cash" and "make deposit" as expected. Additionally there is an alternative choice between the use cases because of the two transitions from *S9*.
Similarly, states *S12* and *S15* are both sub-states of *S8*. The sequencing

from “withdraw cash” and “make deposit” to “*turn ATM off*” is therefore possible.

State	Condition
S0	ATM is on AND user transaction status is NOT card inserted
S4	user identification is valid AND ATM is on AND ATM display is operation menu AND user is logged in AND user card is NOT unreadable AND user transaction status is pin entered
S5	ATM is off
S7	ATM display is welcome message AND operator transaction status is amount in cash entered AND ATM is on
S8	ATM is on
S9	ATM is on AND ATM display is operation menu AND user is logged in
S12	ATM is on AND ATM transaction status is user cash provided AND user transaction status is NOT card inserted
S15	ATM is on AND ATM transaction status is user account updated AND user transaction status is NOT card inserted

Table 4: Condition corresponding to use case connecting states

The global state model includes extra behaviors from the strict interpretation of control-flow relations. For instance, condition “*ATM is on*” is included in all states conditions but *S5*. As a consequence, operation “turn system off” would be accepted from all these states. This extra behavior might be accepted as a valid generalization of the use case model. In case extra behaviors such as the above are deemed unacceptable, operations effects need to be modified to obtain the required behavior.

In order to have a strict correspondence between states *S4* and *S9,* operation *display operation menu* should be such that condition “*user is*

logged in AND ATM display is operation menu AND ATM is on" is obtained rather than "*user identification is valid AND user is logged in AND ATM display is operation menu AND ATM is on AND user transaction status is pin entered AND user card is NOT unreadable"*. Therefore, the extra conditions "*user identification is valid", "user transaction status is pin entered"* and "*user card is NOT unreadable"* need to be withdrawn by the operation. We specify operation *display operation menu* withdrawn-condition as "*ANY ON User**".
Similarly we add "*ANY ON ATM transaction status*" as withdrawn-condition to operation eject user card such that states *S12* and *S15* are replaced by *S0*.
In order that use case "log in" follows "*turn ATM on*", state *S0* conditions need to be obtained at the end of the primary scenario of "turn ATM on". Note that the following warning message is displayed when generating an effect-based state model

WARNING - success postcondition "user transaction status is NOT card inserted" of use case turn ATM on is not satisfied at the end of scenario.
Actual conditions are
[ATM display is welcome message, operator transaction status is amount in cash entered, ATM is on]

We add "*user transaction status is NOT card inserted*" as added-condition to operation *turn system ON*, and we add condition "*ATM display is welcome message*" to use case "*log in*" pre-condition and specify "*ANY ON Operator**" as withdrawn-condition to operation *display welcome message*. We also note that use cases "*withdraw cash*" and "*make deposit*" need to execute operation display welcome message in order that condition "*display is welcome message*" hold and perform the necessary modifications.

Figure 37 shows the global effect-based state model obtained after the above changes. The initial state has been set to *S10* using ***Simulation -> State Machine obtained from Operation effects -> Set initial State***. State *S9* is sup-set of all states but *S10*. Therefore the transition from *S9* applies to all the states except state *S10*.

According to the strict interpretation of the control-flow sequencing constraints, use case "*turn ATM OFF*" might be executed only after use case "*turn ATM ON*", "*withdraw cash*" and "*make deposit*". This typically corresponds to state *S0*. By adding "*ATM display is welcome message AND user transaction status is NOT card inserted*" to the pre-

condition of use case "*turn ATM OFF*", the generated effect-based state model shown in Figure becomes compliant to the required sequencing. The final use case and domain model are shown in the Appendix.

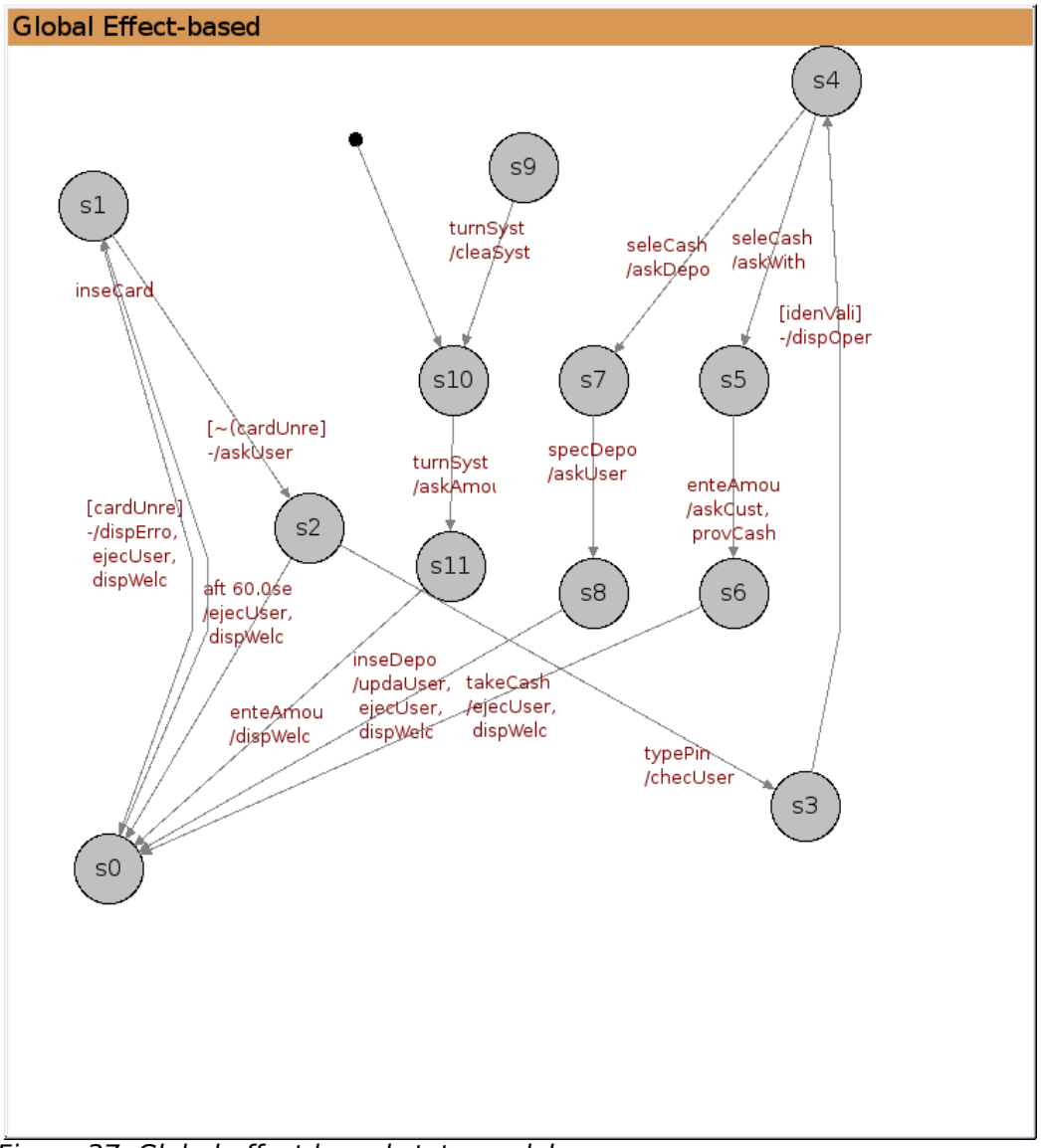


Figure 37: Global effect-based state model

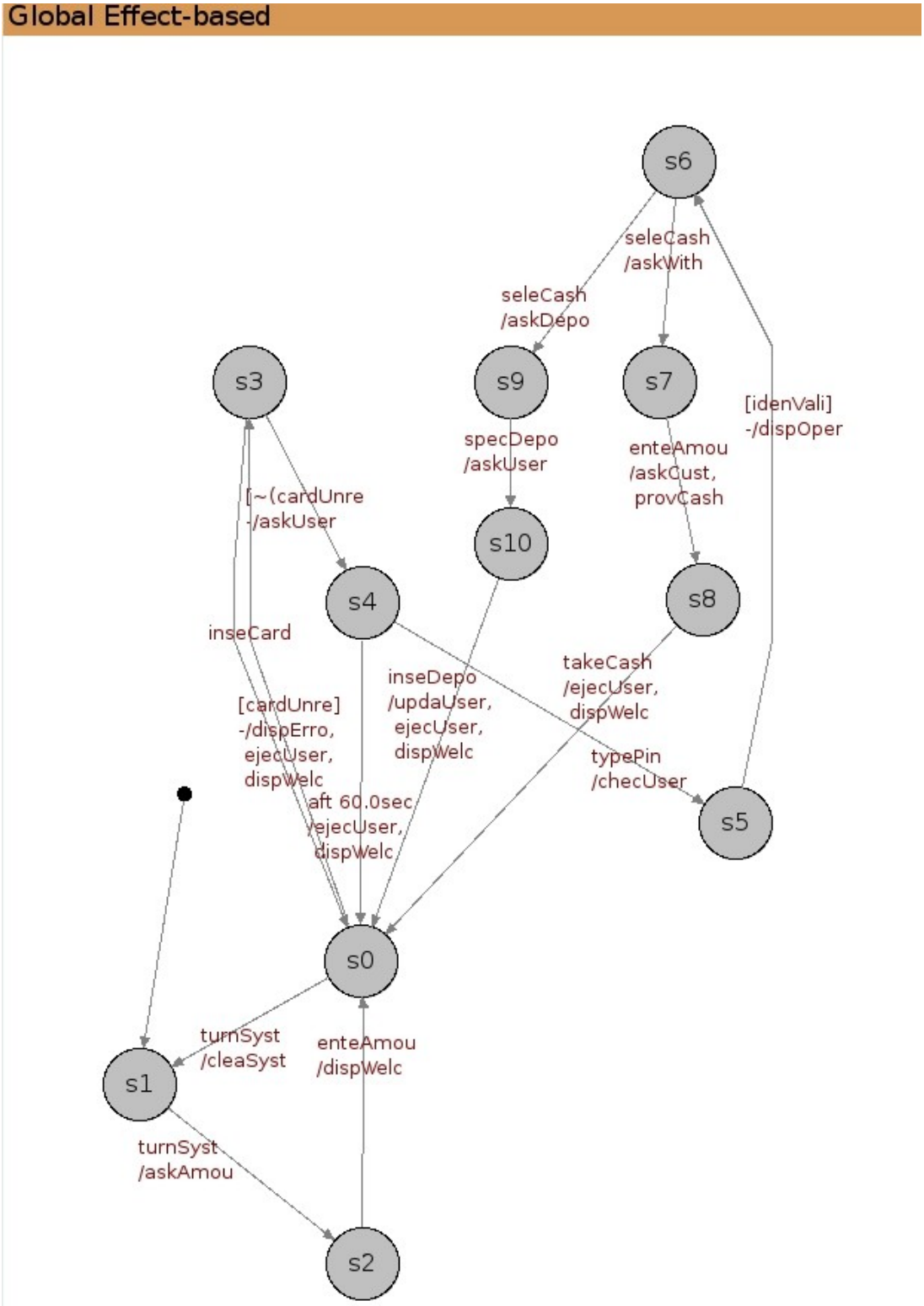


Figure 38: Global effect-based state model

10. Conclusion

This document presented an iterative approach for use cases elaboration in conjunction with domain elements using UCEd. The approach is based on a strong relation between use cases and specification of operations. Use cases state required events sequencing. However, these requirements are possible only given specific transformation performed by operations. We capture these transformations in a contractual form as preconditions and postconditions. In this document, we only focused on a subset of use case description capabilities supported by UCEd. Other capabilities include use case «*include*» and «*extend*» relations as well as *scenarios* for automating use case simulation.

Appendix

Final use case and domain model for the ATM example.

Domain model

```
domain
  System Concept:ATM
      Attribute:display
          Possible Values Set
              Value:error message
              Value:pin enter prompt
              Value:operation menu
              Value:amount in dispenser prompt
              Value:welcome message
              Value:withdrawal amount
              Value:deposit amount prompt
              Value:deposit insertion prompt
      Attribute:transaction status
          Possible Values Set
              Value:customer account update
              Value:user account updated
              Value:user cash provided
      Operation Set
          Operation:display error message
              AddedCondition:ATM display is error message
          Operation:eject user card
              AddedCondition:User transaction status is not card inserted
              WithdrawCondition:ANY ON User*
              WithdrawCondition:ANY ON ATM display
              WithdrawCondition:ANY ON ATM transaction status
          Operation:ask user pin
              AddedCondition:ATM display is pin enter prompt
          Operation:check user's identification
              AddedCondition:User identification is valid or User
                             identification is not valid
          Operation:display operation menu
              AddedCondition:ATM display is operation menu
              AddedCondition:User is logged in
              WithdrawCondition:ANY ON User*
          Operation:ask amount in cash dispenser
              AddedCondition:ATM display is amount in dispenser prompt
          Operation:display welcome message
              AddedCondition:ATM display is welcome message
              WithdrawCondition:ANY ON Operator*
          Operation:clear system
              AddedCondition:ATM is OFF
              WithdrawCondition:ANY ON Operator*
              WithdrawCondition:ANY ON ATM*
              WithdrawCondition:ANY ON User*
          Operation:ask withdrawal amount
              AddedCondition:ATM display is withdrawal amount
```

Operation:ask customer account update
AddedCondition:ATM transaction status is customer account update
Operation:ask deposit amount
AddedCondition:ATM display is deposit amount prompt
Operation:ask user insert deposit
AddedCondition:ATM display is deposit insertion prompt
Operation:update user's account
AddedCondition:ATM transaction status is user account updated
Operation:provide cash
AddedCondition:ATM transaction status is user cash provided
Possible Values Set
Value:on
Value:off
Concept:user
Sub Component:card
Operation Set
Possible Values Set
Value:unreadable
Attribute:transaction status
Possible Values Set
Value:card inserted
Value:pin entered
Value:cash withdrawal selected
Value:deposit amount specified
Value:cash deposit selected
Value:deposit inserted
Value:amount entered
Value:cash taken
Attribute:identification
Possible Values Set
Value:valid
Operation Set
Operation:insert card in
AddedCondition: User transaction status is card inserted
Operation:type pin
AddedCondition: User transaction status is pin entered
Operation:select cash withdrawal
AddedCondition: User transaction status is cash withdrawal selected
Operation:enter amount
AddedCondition: User transaction status is amount entered
Operation:take cash
AddedCondition: User transaction status is cash taken
Operation:select cash deposit
AddedCondition: User transaction status is cash deposit selected
Operation:specifie deposit amount
AddedCondition: User transaction status is deposit amount specified
Operation:insert deposit
AddedCondition: User transaction status is deposit inserted

```
    Possible Values Set
        Value:logged in
Concept:operator
    Attribute:transaction status
        Possible Values Set
            Value:system turned on
            Value:amount in cash entered
            Value:system turned off
    Operation Set
        Operation:turn system on
            Precondition:ATM is OFF
            AddedCondition:ATM is ON
            AddedCondition:Operator transaction status is system turned
                           on
            AddedCondition:User transaction status is not card inserted
        Operation:enter amount of money currently in cash dispenser
            AddedCondition:Operator transaction status is amount in
                           cash entered
        Operation:turn system off
            AddedCondition:Operator transaction status is system turned
                           off
```

Use case model

Use Case: log in

Title: log in
Description: This use case captures a login procedure to the ATM System. Users are identified with a Card and a password. After the User has provided her Card and password, the system checks for the identification and if valid, displays an operation menu. If the Card or password are not valid, the User access is denied and her Card returned.
System Under Design: ATM
Primary Actor: User
Participants:
Goal: Allow a User to identify itself to the ATM in order to perform banking transactions.
Follows Use Cases: log in OR turn ATM on OR withdraw cash OR make deposit
Invariant:
Precondition: ATM is ON AND User transaction status is not card i inserted AND ATM display is welcome message
STEPS
1.User inserts a Card in the ATM card reader slot
2.The ATM asks for User pin
3.The User types a pin
4.ATM checks the User's identification

5.If User identification is valid then, ATM displays an operation menu
6.enable withdraw cash, make deposit

ALTERNATIVES

1.a.User Card is unreadable
1.a.1.ATM displays an error message
1.a.2.ATM ejects the User Card
1.a.3.ATM displays welcome message
1.a.4.resume

Alternative Postcondition: ATM is ON AND User transaction status is not card inserted AND ATM display is welcome message

2.a.after 60 sec
2.a.1.ATM ejects the User Card
2.a.2.ATM displays welcome message
2.a.3.resume

Alternative Postcondition: ATM is ON AND User transaction status is not card inserted AND ATM display is welcome message
Success Postcondition: User is logged in AND ATM is ON AND ATM display is operation menu

Use Case: turn ATM on

Title: turn ATM on
Description:
System Under Design: ATM
Primary Actor: Operator
Participants:
Goal: Allows an Operator to start the ATM up so that it could provide transaction services to Users.
Follows Use Cases: turn ATM off
Invariant:
Precondition: ATM is OFF

STEPS

1.The Operator turns the system ON
2.The ATM asks the amount in the cash dispenser
3.The Operator enters the amount of money currently in cash dispenser
4.The ATM displays a welcome message
5.enable log in, turn ATM off

Success Postcondition: ATM is on AND user transaction status is NOT card inserted AND ATM display is welcome message

Use Case: turn ATM off

Title: turn ATM off
Description:
System Under Design: ATM

Primary Actor: Operator
Participants:
Goal: Allows an Operator to switch the ATM off. Transaction services are not provided anymore following the use case.
Follows Use Cases: turn ATM on OR withdraw cash OR make deposit
Invariant:
Precondition: ATM is ON AND ATM display is welcome message AND user transaction status is NOT card inserted
STEPS
1.The Operator turns the system off
2.The ATM clears the system
3.enable turn ATM on
Success Postcondition: ATM is OFF

Use Case: withdraw cash

Title: withdraw cash
Description:
System Under Design: ATM
Primary Actor: User
Participants:
Goal: Allow a User to get a cash amount by deduction from his/her account.
Follows Use Cases: log in
Invariant:
Precondition: ATM is ON AND ATM Display is operation menu AND User is logged in
STEPS
1.The User selects cash withdrawal
2.The ATM asks the withdrawal amount
3.The User enters an amount
4.The ATM asks the customer account update
5.ATM provides cash in the cash compartment
6.The User takes the cash from the cash compartment
7.The ATM ejects the user card
8.The ATM displays a welcome message
9.enable log in, tum ATM off
Success Postcondition: ATM is on AND user transaction status is NOT card inserted AND ATM display is welcome message

Use Case: make deposit

Title: make deposit
Description:
System Under Design: ATM

Primary Actor: User
Participants:
Goal: Allows a User to make a money deposit to his/her account.
Follows Use Cases: log in
Invariant:
Precondition: ATM is ON AND ATM Display is operation menu AND User is logged in
STEPS
1.The User selects cash deposit
2.The ATM asks for a deposit amount
3.The User specifies a deposit amount
4.The ATM asks the user to insert a deposit
5.The User inserts a deposit
6.The ATM updates the User's account
7.The ATM ejects the user card
8.The ATM displays a welcome message
9.enable log in, tum ATM off
Success Postcondition: ATM is on AND user transaction status is NOT card inserted AND ATM display is welcome message