CS6473 : Software Architecture Laboratory (Autumn 2022)

Bishwajit Prasad Gond
222CS3113

Master of Technology
222cs3113@nitrkl.ac.in

Department of Computer Science & Engineering
NIT, Rourkela

November 13, 2022

Contents

1 UML DIAGRAMS (STAR UML)

1.1 USE CASE DIAGRAM

A use case diagram is a graphic depiction of the interactions among the elements of a system.

A use case is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system to achieve a goal. The actor can be a human or other external system.

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

- It gathers the system's needs.
- It depicts the external view of the system.
- It recognizes the internal as well as external factors that influence the system.
- It represents the interaction between the actors.

Use case diagram for ATM machine

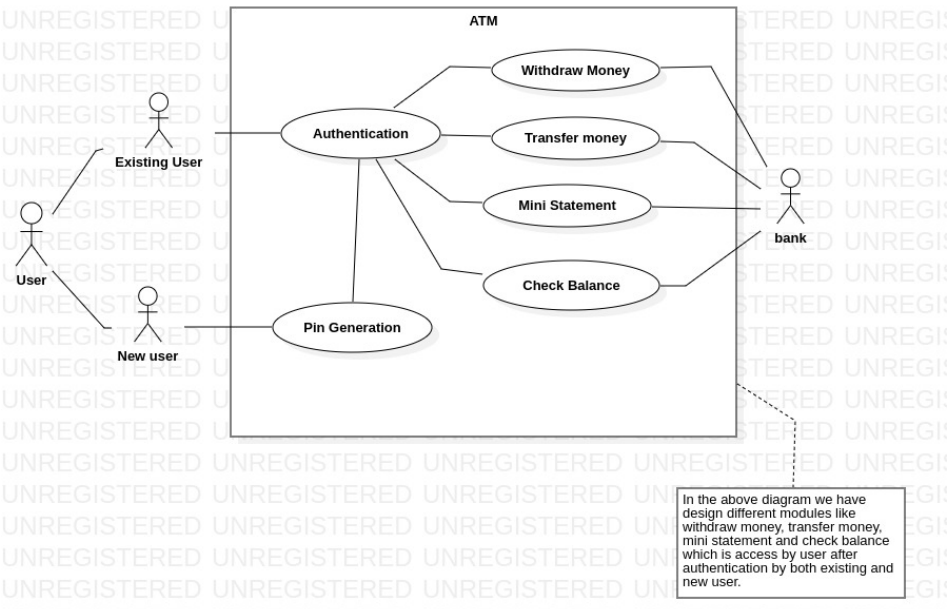

Figure 1: ATM USE Case

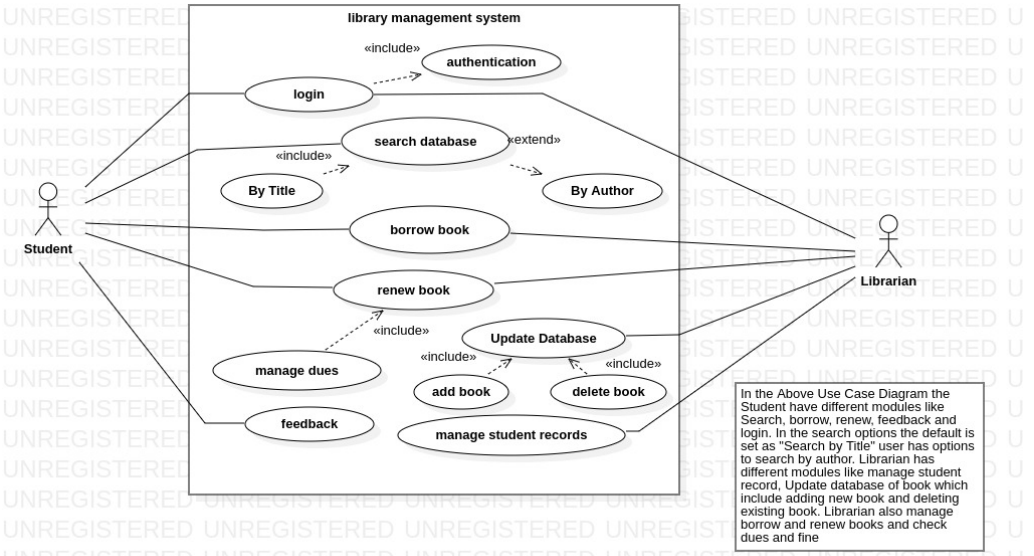

Figure 2: Online Shopping USE Case

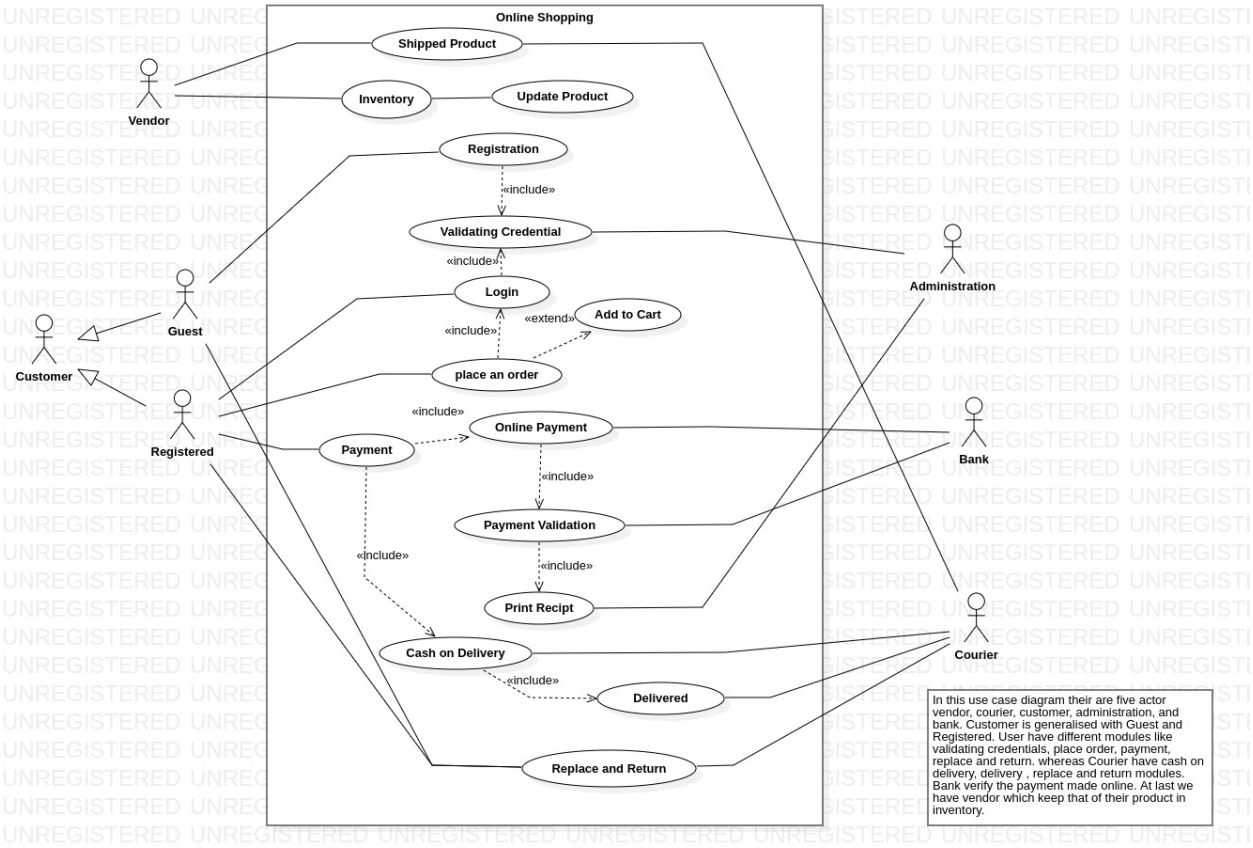

Figure 3: ATM USE Case

1.2 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

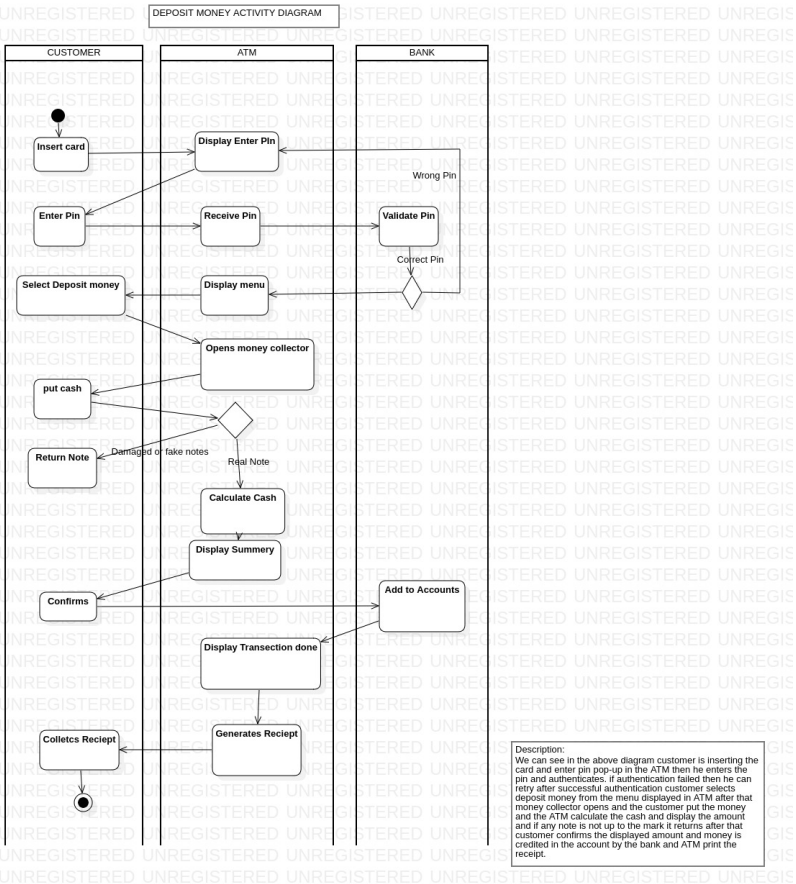

Figure 4: Deposit Activity Diagram

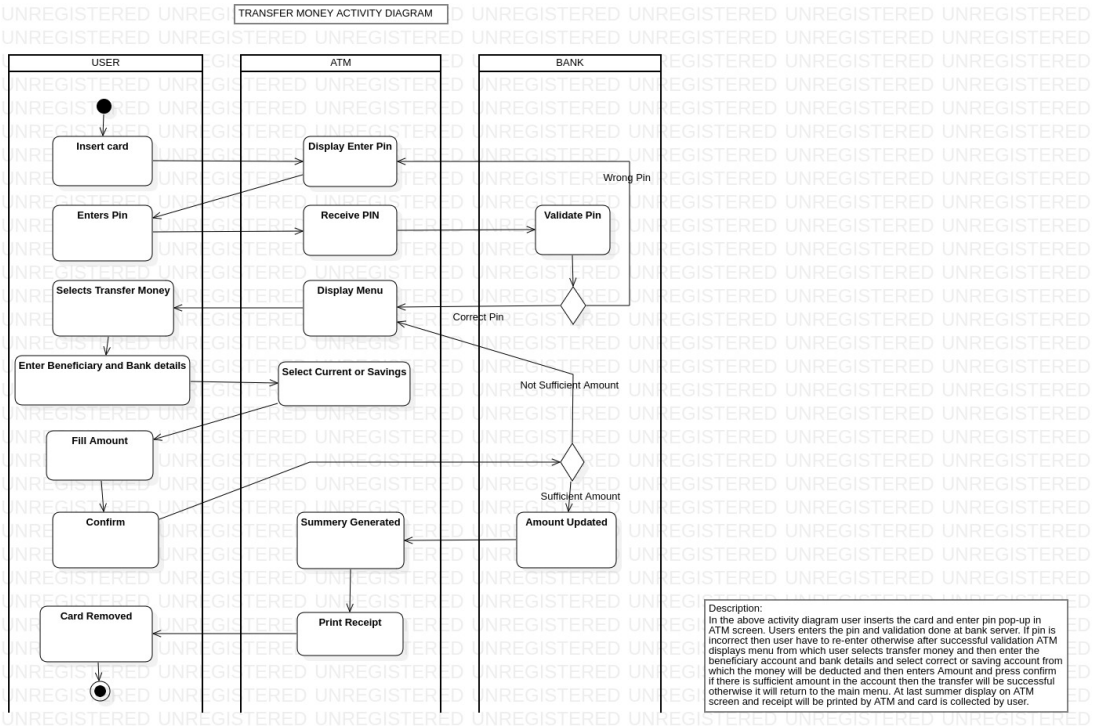

Figure 5: Transfer Activity Diagram

1.3 STATE CHART DIAGRAM

A state diagram is a diagram used in computer science to describe the behavior of a system considering all the possible states of an object when an event occurs. This behavior is represented and analyzed in a series of events that occur in one or more possible states.

Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using Statechart diagrams

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object.

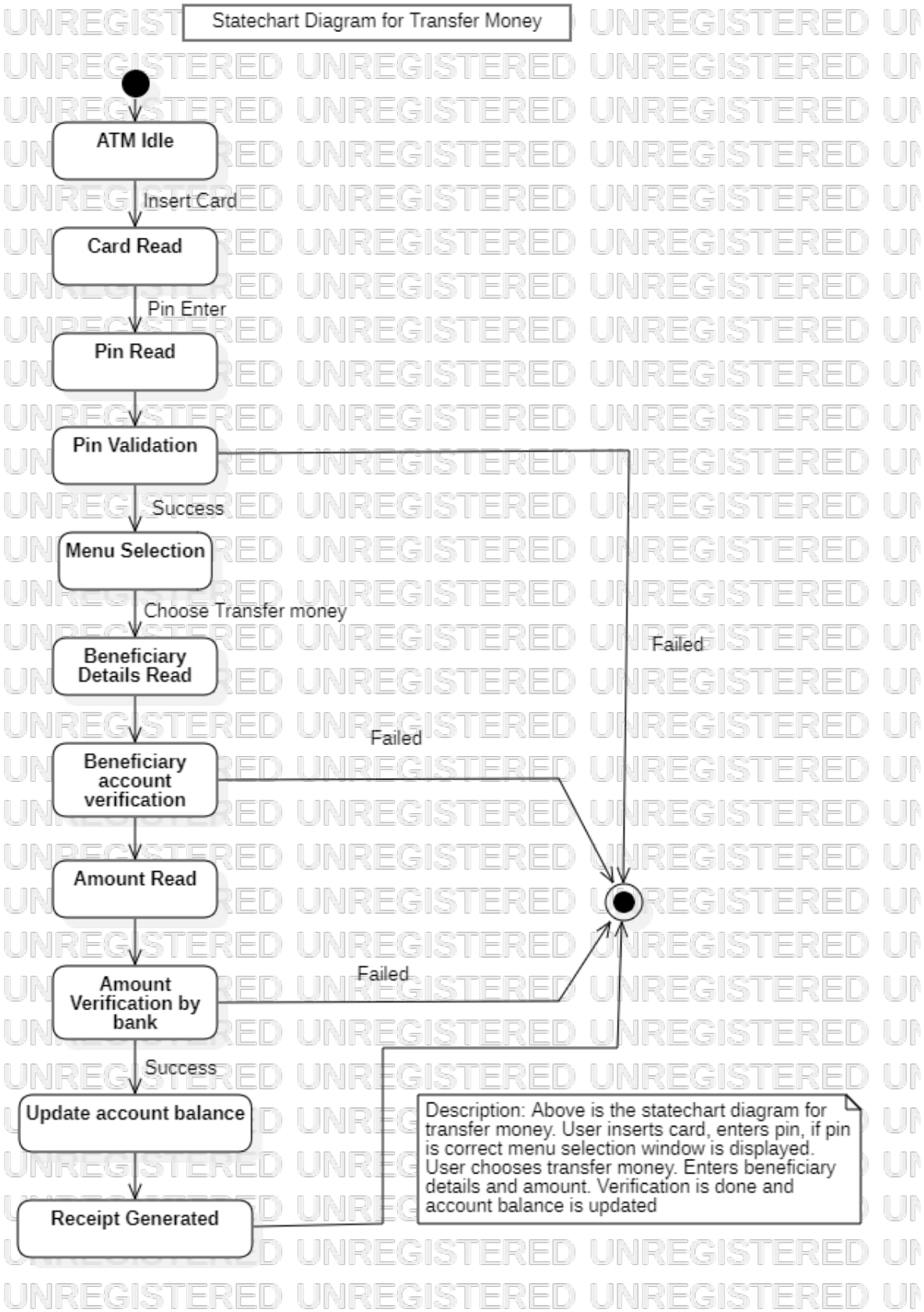

Figure 6: Transfer Money State chart Diagram.

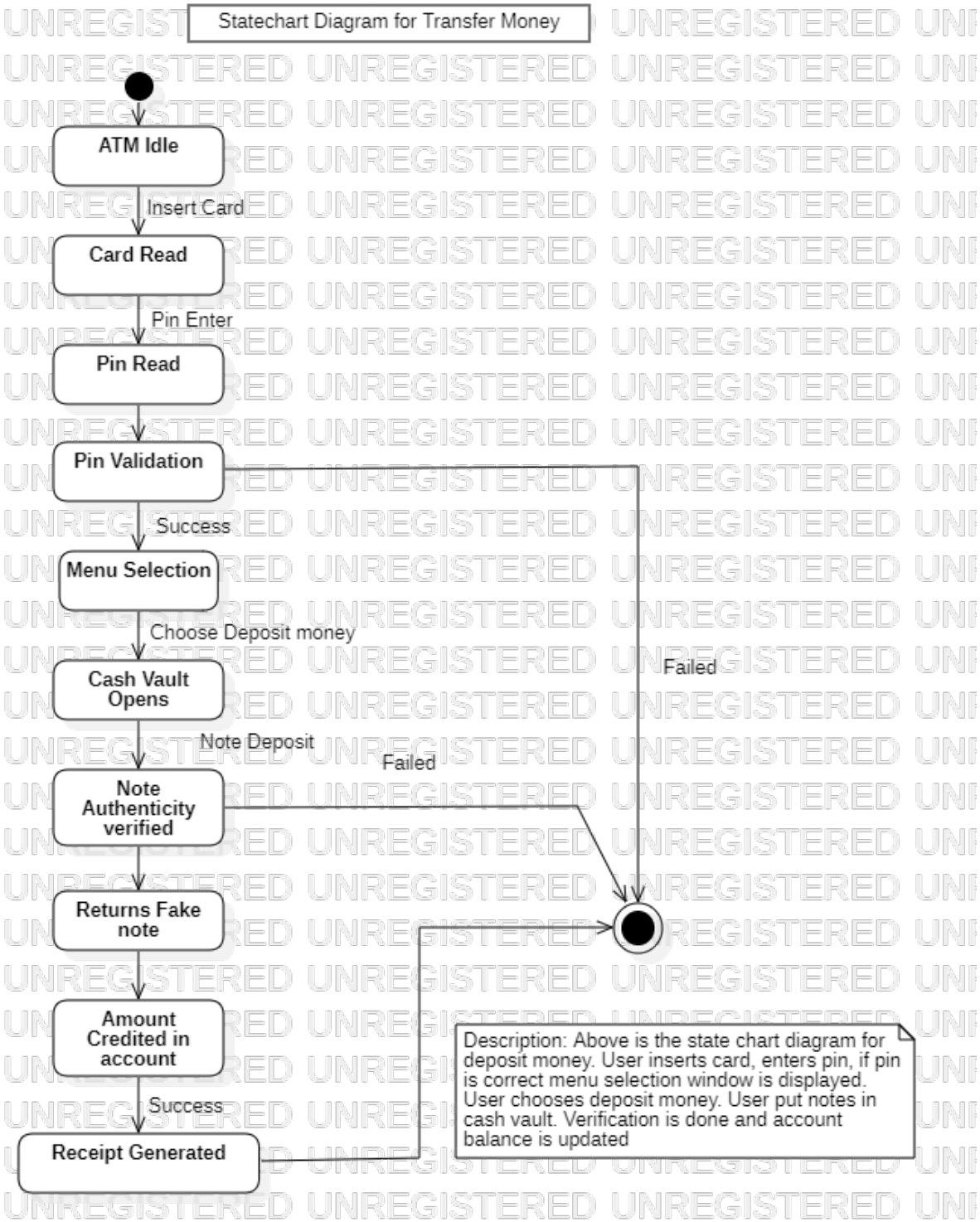

Figure 7: Deposit Money State chart Diagram.

1.4 SEQUENCE DIAGRAM

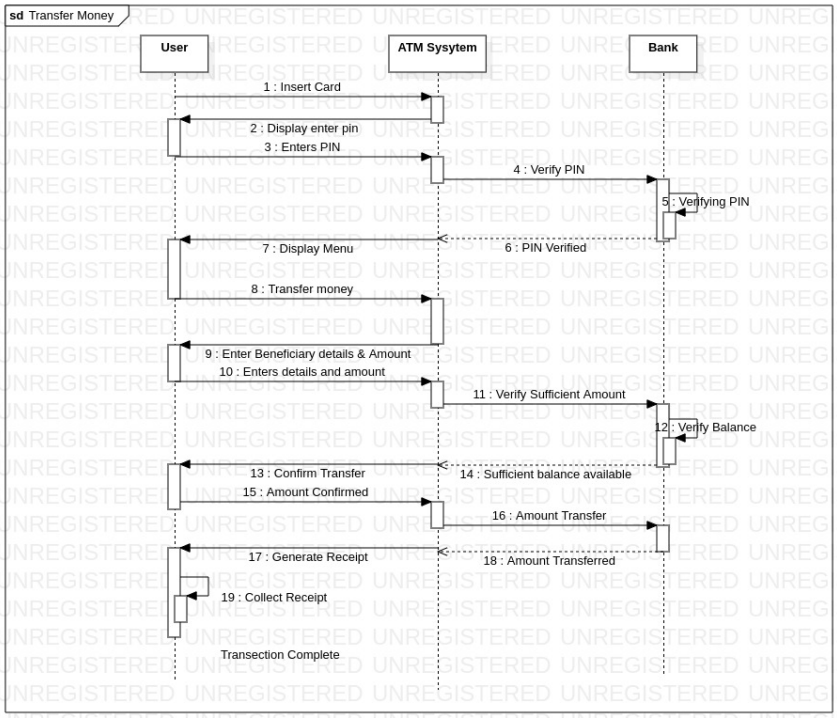

Figure 8: Transfer Money Sequence Diagram.

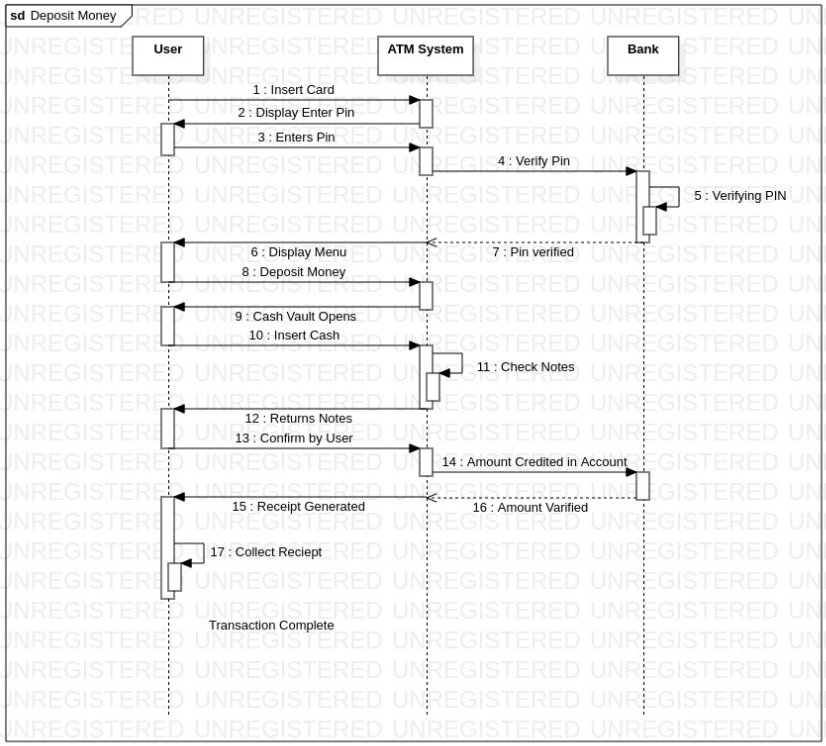

Figure 9: Deposit Money Sequence Diagram.

1.5 COMPONENT DIAGRAM

In Unified Modeling Language (UML), a component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.

UML Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

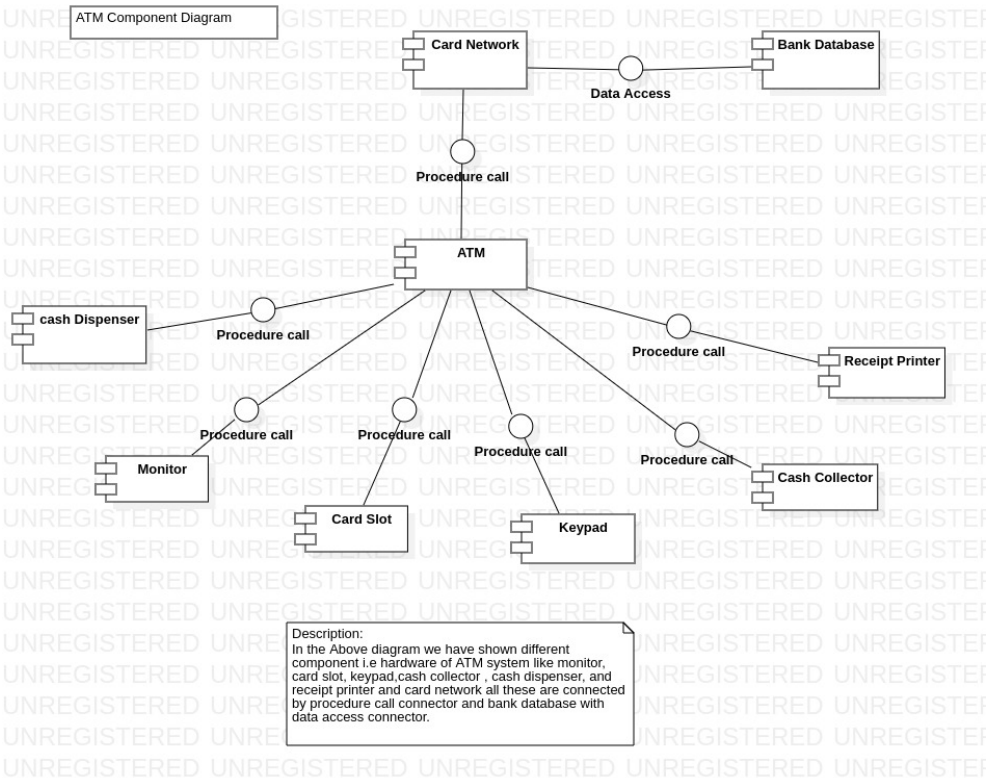

Figure 10: ATM Component Diagram.

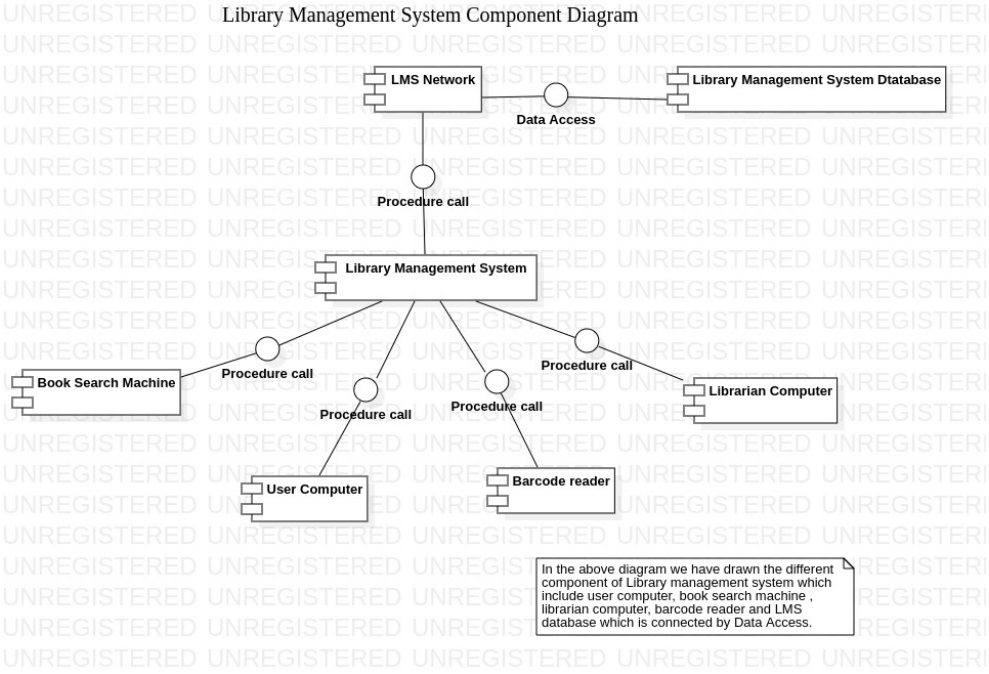

Figure 11: Library Management System Component Diagram.

1.6 CLASS DIAGRAM

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Purpose of Class Diagrams

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

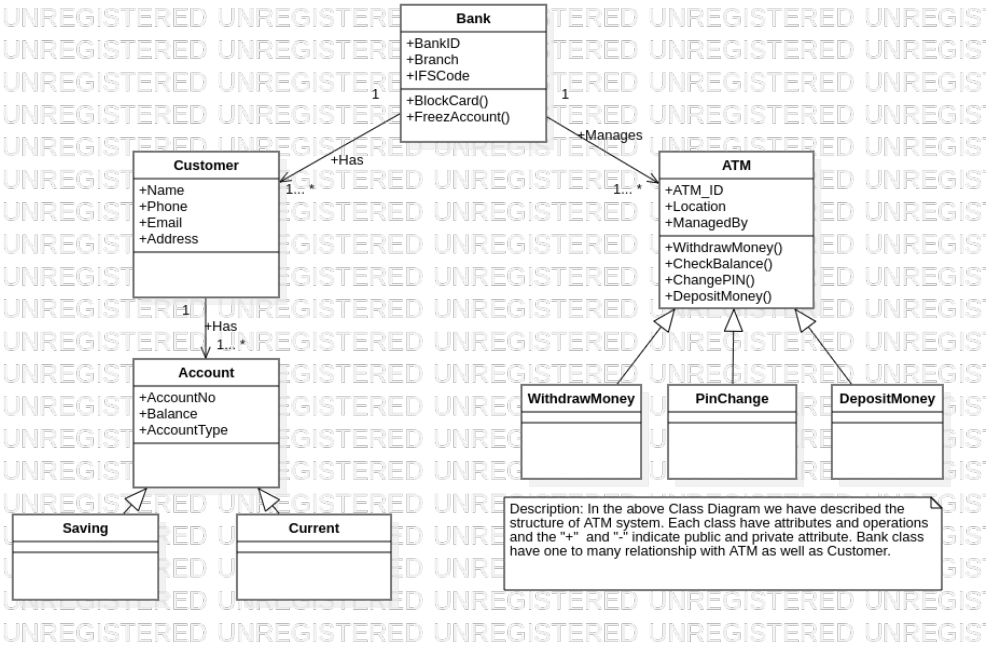

Figure 12: ATM Class Diagram.

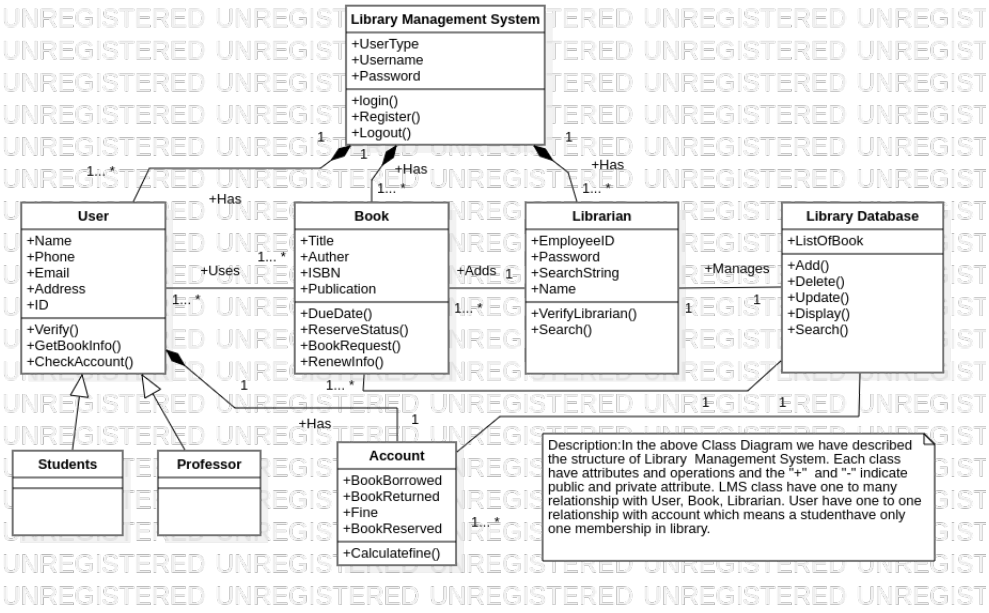

Figure 13: Library Management System Class Diagram.

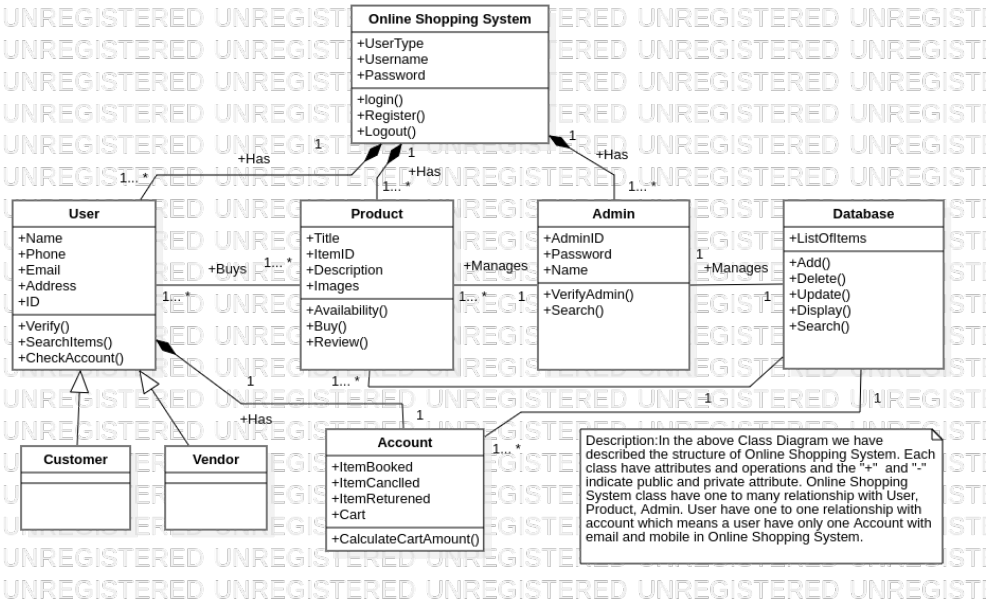

Figure 14: Online Shopping System Class Diagram.

2 ACME STUDIO

AcmeStudio is a customizable editing environment and visualization tool for software architectural designs based on the Acme architectural description language (ADL).

2.1 ATM

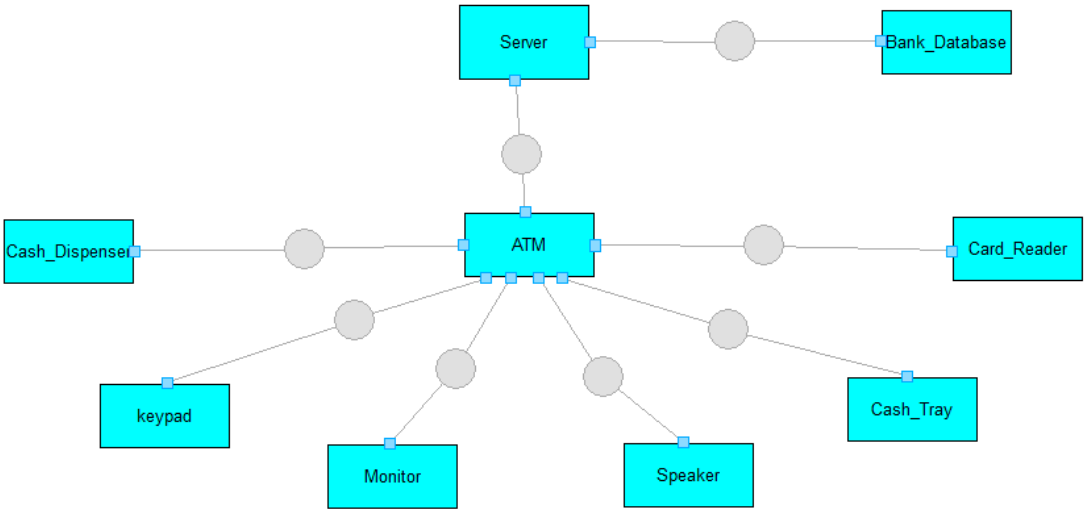

Figure 15: ATM ACME Diagram.

2.2 ATM SOURCE CODE

```
System ATM = {

    Component ATM  = {
        Port procedurecall  = {

        }
        Port procedurecall0  = {

        }
        Port procedurecall1  = {

        }
        Port procedurecall2  = {

        }
        Port procedurecall3  = {

        }
```

```
19          Port procedurecall4  = {
20
21          }
22          Port procedurecall5  = {
23
24          }
25
26      }
27      Component keypad  = {
28          Port procedurecall1  = {
29
30          }
31
32      }
33      Component Server  = {
34          Port procedure_call  = {
35
36          }
37          Port procedure_call0  = {
38
39          }
40
41      }
42      Component Monitor  = {
43          Port procedurecall1  = {
44
45          }
46
47      }
48      Component Cash_Tray  = {
49          Port procedurecall1  = {
50
51          }
52
53      }
54      Component Cash_Dispenser  = {
55          Port procedurecall1  = {
56
57          }
58
59      }
60      Component Card_Reader  = {
61          Port procedurecall1  = {
62
63          }
64
65      }
66      Component Speaker  = {
```

```
67          Port procedurecall1  = {
68
69          }
70
71      }
72      Component Bank_Database  = {
73          Port procedure_call0  = {
74
75          }
76
77      }
78      Connector procedure_call  = {
79          Role role0  = {
80
81          }
82          Role role1  = {
83
84          }
85
86      }
87      Connector procedure_call0  = {
88          Role role0  = {
89
90          }
91          Role role1  = {
92
93          }
94
95      }
96      Connector procedure_call1  = {
97          Role role0  = {
98
99          }
100         Role role1  = {
101
102         }
103
104     }
105     Connector procedure_call2  = {
106         Role role0  = {
107
108         }
109         Role role1  = {
110
111         }
112
113     }
114     Connector procedure_call3  = {
```

```
115         Role role0  = {
116
117         }
118         Role role1  = {
119
120         }
121
122     }
123     Connector procedure_call4  = {
124         Role role0  = {
125
126         }
127         Role role1  = {
128
129         }
130
131     }
132     Connector procedure_call5  = {
133         Role role0  = {
134
135         }
136         Role role1  = {
137
138         }
139
140     }
141     Connector Data_Access  = {
142         Role role0  = {
143
144         }
145         Role role1  = {
146
147         }
148
149     }
150     Attachment Server.procedure_call to procedure_call.role1;
151     Attachment ATM.procedurecall to procedure_call.role0;
152     Attachment ATM.procedurecall0 to procedure_call0.role1;
153     Attachment keypad.procedurecall1 to procedure_call0.role0;
154     Attachment Cash_Tray.procedurecall1 to procedure_call2.role1;
155     Attachment ATM.procedurecall2 to procedure_call2.role0;
156     Attachment Monitor.procedurecall1 to procedure_call1.role0;
157     Attachment ATM.procedurecall1 to procedure_call1.role1;
158     Attachment ATM.procedurecall5 to procedure_call3.role1;
159     Attachment Card_Reader.procedurecall1 to procedure_call5.
          role0;
160     Attachment ATM.procedurecall3 to procedure_call5.role1;
161     Attachment ATM.procedurecall4 to procedure_call4.role1;
```

```
    Attachment Cash_Dispenser.procedurecall1 to procedure_call4.
       role0;
    Attachment Speaker.procedurecall1 to procedure_call3.role0;
    Attachment Bank_Database.procedure_call0 to Data_Access.role0
       ;
    Attachment Server.procedure_call0 to Data_Access.role1;
}
```

2.3 LIBRARY MANAGEMENT SYSTEM

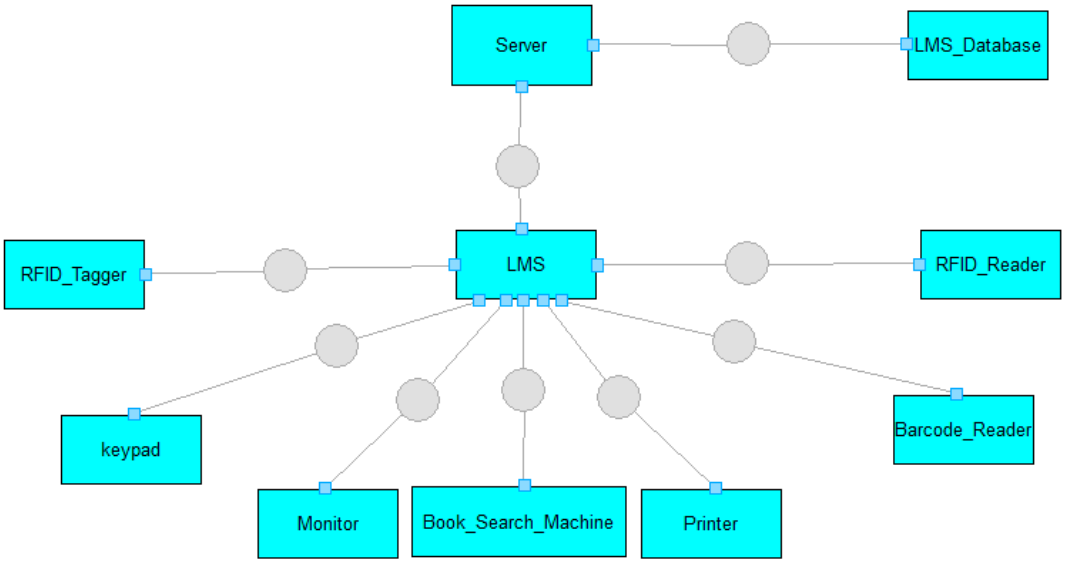

Figure 16: Library Management System ACME Diagram.

2.4 LIBRARY MANAGEMENT SYSTEM SOURCE CODE

```
System Library = {

    Component keypad  = {
        Port procedurecall1  = {

        }

    }
    Component Server  = {
        Port procedure_call  = {

        }
        Port procedure_call0  = {

```

```
15          }
16
17      }
18      Component Monitor  = {
19          Port procedurecall1  = {
20
21          }
22
23      }
24      Component LMS  = {
25          Port procedurecall  = {
26
27          }
28          Port procedurecall0  = {
29
30          }
31          Port procedurecall1  = {
32
33          }
34          Port procedurecall2  = {
35
36          }
37          Port procedurecall3  = {
38
39          }
40          Port procedurecall4  = {
41
42          }
43          Port procedurecall5  = {
44
45          }
46          Port procedurecall6  = {
47
48          }
49
50      }
51      Component Barcode_Reader  = {
52          Port procedurecall1  = {
53
54          }
55
56      }
57      Component Book_Search_Machine  = {
58          Port procedurecall1  = {
59
60          }
61
62      }
```

```
63      Component LMS_Database  = {
64          Port procedure_call0  = {
65
66          }
67
68      }
69      Component RFID_Reader  = {
70          Port procedurecall4  = {
71
72          }
73
74      }
75      Component RFID_Tagger  = {
76          Port procedurecall4  = {
77
78          }
79
80      }
81      Component Printer  = {
82          Port procedurecall1  = {
83
84          }
85
86      }
87      Connector procedure_call3  = {
88          Role role0  = {
89
90          }
91          Role role1  = {
92
93          }
94
95      }
96      Connector procedure_call0  = {
97          Role role0  = {
98
99          }
100         Role role1  = {
101
102         }
103
104     }
105     Connector procedure_call  = {
106         Role role0  = {
107
108         }
109         Role role1  = {
110
```

```
        }

    }
    Connector procedure_call1  = {
        Role role0  = {

        }
        Role role1  = {

        }

    }
    Connector procedure_call2  = {
        Role role0  = {

        }
        Role role1  = {

        }

    }
    Connector Data_Access  = {
        Role role0  = {

        }
        Role role1  = {

        }

    }
    Connector procedure_call4  = {
        Role role0  = {

        }
        Role role1  = {

        }

    }
    Connector procedure_call5  = {
        Role role0  = {

        }
        Role role1  = {

        }

    }
```

```
159     Connector procedure_call6  = {
160         Role role0  = {
161 
162         }
163         Role role1  = {
164 
165         }
166 
167     }
168     Attachment Server.procedure_call0 to Data_Access.role1;
169     Attachment LMS_Database.procedure_call0 to Data_Access.role0;
170     Attachment Book_Search_Machine.procedurecall1 to
          procedure_call3.role0;
171     Attachment LMS.procedurecall0 to procedure_call0.role1;
172     Attachment LMS.procedurecall to procedure_call.role0;
173     Attachment LMS.procedurecall5 to procedure_call3.role1;
174     Attachment LMS.procedurecall2 to procedure_call2.role0;
175     Attachment LMS.procedurecall1 to procedure_call1.role1;
176     Attachment Barcode_Reader.procedurecall1 to procedure_call2.
          role1;
177     Attachment LMS.procedurecall4 to procedure_call4.role1;
178     Attachment RFID_Tagger.procedurecall4 to procedure_call4.
          role0;
179     Attachment LMS.procedurecall3 to procedure_call5.role1;
180     Attachment RFID_Reader.procedurecall4 to procedure_call5.
          role0;
181     Attachment LMS.procedurecall6 to procedure_call6.role1;
182     Attachment Printer.procedurecall1 to procedure_call6.role0;
183     Attachment Monitor.procedurecall1 to procedure_call1.role0;
184     Attachment keypad.procedurecall1 to procedure_call0.role0;
185     Attachment Server.procedure_call to procedure_call.role1;
186 }
```

2.5 ONLINE SHOPPING CENTER

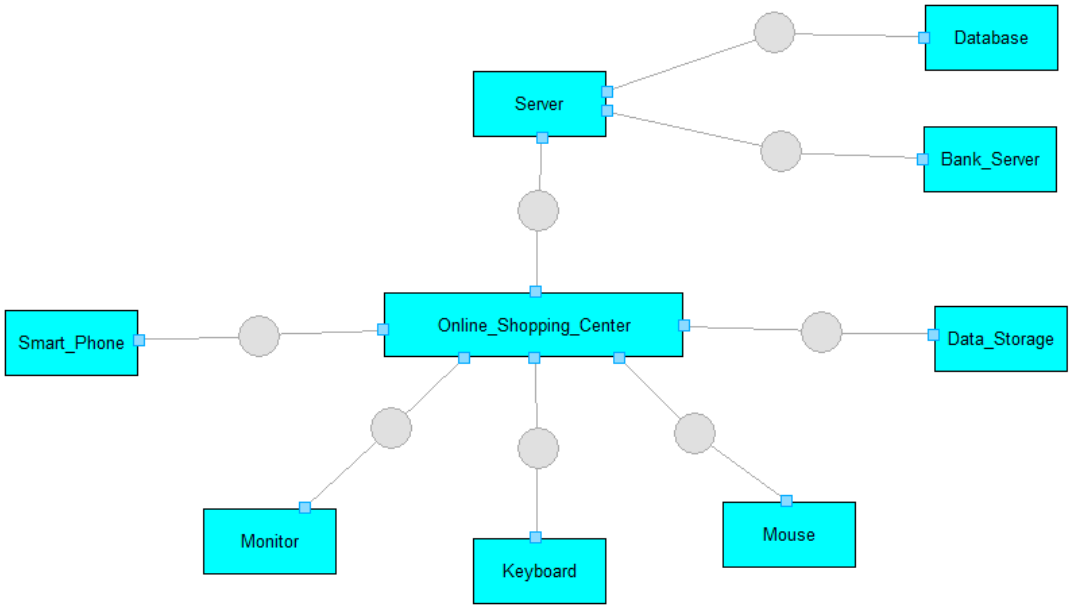

Figure 17: Online Shopping Center ACME Diagram.

2.6 ONLINE SHOPPING CENTER SOURCE CODE

```
System OSC = {

    Component Online_Shopping_Center  = {
        Port port1  = {

        }
        Port port0  = {

        }
        Port port2  = {

        }
        Port port4  = {

        }
        Port port5  = {

        }
        Port port3  = {

        }

```

```
23      }
24      Component Server  = {
25          Port port1  = {
26
27          }
28          Port port0  = {
29
30          }
31          Port port2  = {
32
33          }
34
35      }
36      Component Database  = {
37          Port port1  = {
38
39          }
40
41      }
42      Component Smart_Phone  = {
43          Port port0  = {
44
45          }
46
47      }
48      Component Monitor  = {
49          Port port0  = {
50
51          }
52
53      }
54      Component Data_Storage  = {
55          Port port0  = {
56
57          }
58
59      }
60      Component Keyboard  = {
61          Port port0  = {
62
63          }
64
65      }
66      Component Mouse  = {
67          Port port0  = {
68
69          }
70
```

```
71      }
72      Component Bank_Server  = {
73          Port port1  = {
74
75          }
76
77      }
78      Connector Data_Access  = {
79          Role role0  = {
80
81          }
82          Role role1  = {
83
84          }
85
86      }
87      Connector Procedure_Call  = {
88          Role role0  = {
89
90          }
91          Role role1  = {
92
93          }
94
95      }
96      Connector Procedure_Call0  = {
97          Role role0  = {
98
99          }
100         Role role1  = {
101
102         }
103
104     }
105     Connector Procedure_Call1  = {
106         Role role0  = {
107
108         }
109         Role role1  = {
110
111         }
112
113     }
114     Connector Procedure_Call2  = {
115         Role role0  = {
116
117         }
118         Role role1  = {
```

```
119
120         }
121
122     }
123     Connector Procedure_Call3  = {
124         Role role0  = {
125
126         }
127         Role role1  = {
128
129         }
130
131     }
132     Connector Procedure_Call4  = {
133         Role role0  = {
134
135         }
136         Role role1  = {
137
138         }
139
140     }
141     Connector Data_Access0  = {
142         Role role0  = {
143
144         }
145         Role role1  = {
146
147         }
148
149     }
150     Attachment Server.port1 to Data_Access.role1;
151     Attachment Database.port1 to Data_Access.role0;
152     Attachment Server.port0 to Procedure_Call4.role1;
153     Attachment Online_Shopping_Center.port3 to Procedure_Call4.
            role0;
154     Attachment Online_Shopping_Center.port0 to Procedure_Call3.
            role1;
155     Attachment Smart_Phone.port0 to Procedure_Call3.role0;
156     Attachment Online_Shopping_Center.port1 to Procedure_Call2.
            role0;
157     Attachment Data_Storage.port0 to Procedure_Call2.role1;
158     Attachment Online_Shopping_Center.port4 to Procedure_Call.
            role1;
159     Attachment Monitor.port0 to Procedure_Call.role0;
160     Attachment Online_Shopping_Center.port5 to Procedure_Call1.
            role1;
161     Attachment Keyboard.port0 to Procedure_Call1.role0;
```

```
162     Attachment Online_Shopping_Center.port2 to Procedure_Call0.
            role1;
163     Attachment Mouse.port0 to Procedure_Call0.role0;
164     Attachment Bank_Server.port1 to Data_Access0.role0;
165     Attachment Server.port2 to Data_Access0.role1;
166 }
```

3 PETRINET DIAGRAMS

Petri net is a graphical programming language for modeling concurrent systems. It has been mainly used to model artificial systems such as manufacturing systems and communication protocols.

3.1 CONCATENATION OF STRING

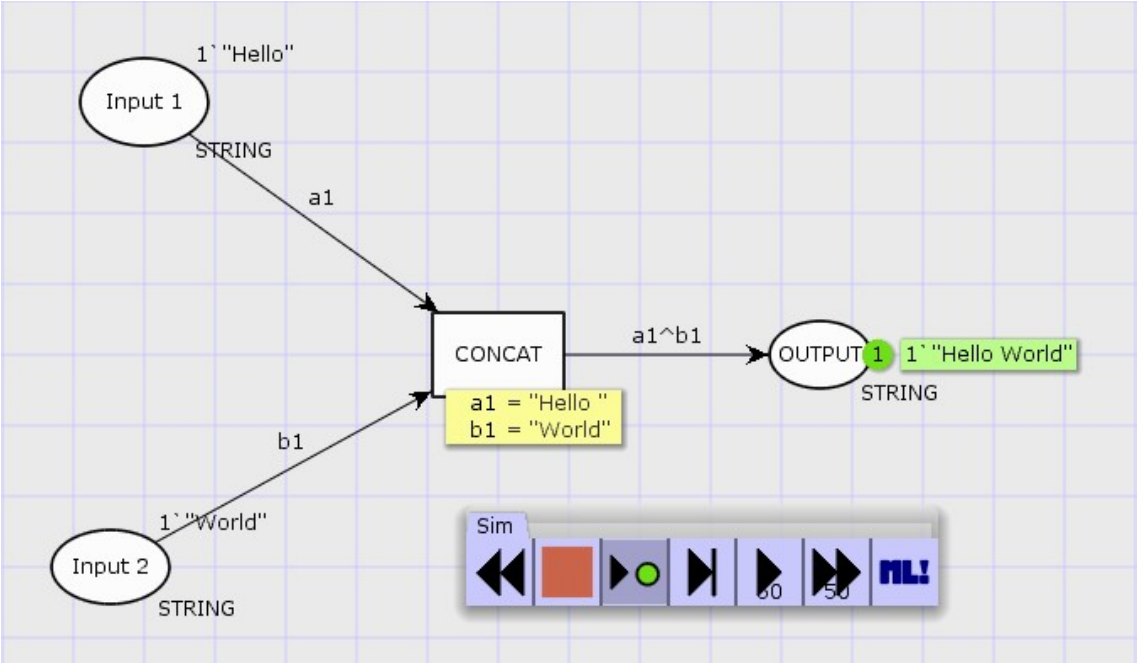

Figure 18: Concatenation of String

3.2 ODD EVEN NUMBER

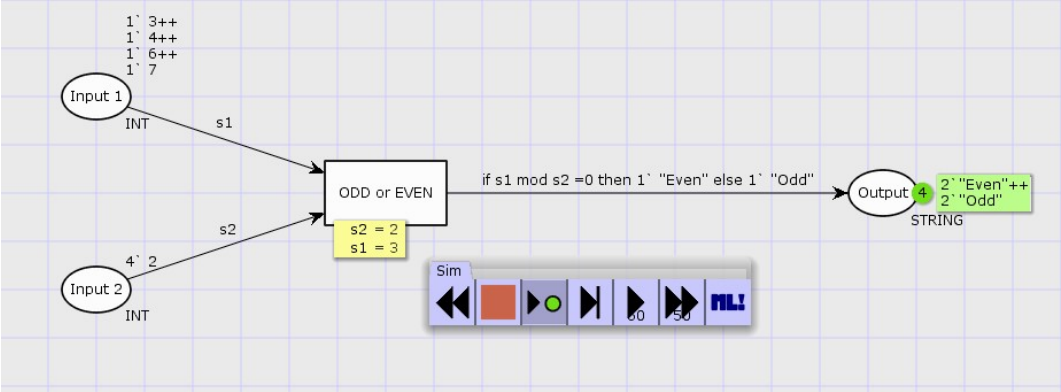

Figure 19: Odd Even Number

3.3 PRINT IF NUMBER IS LESS THAN 10 OR GREATER

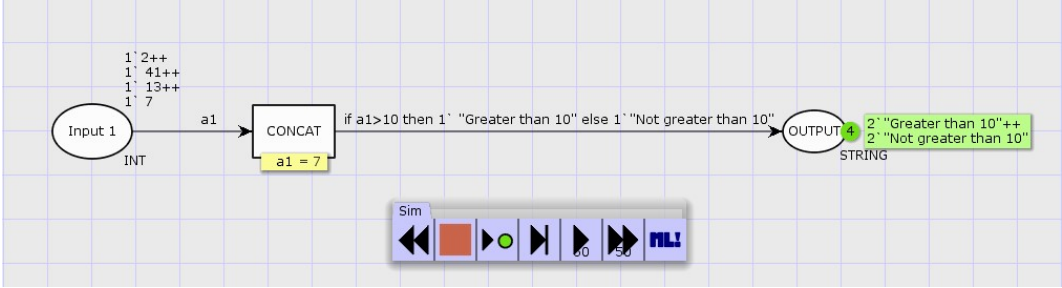

Figure 20: Print if number is less than 10 or greater

3.4 SUM OF TWO NUMBERS

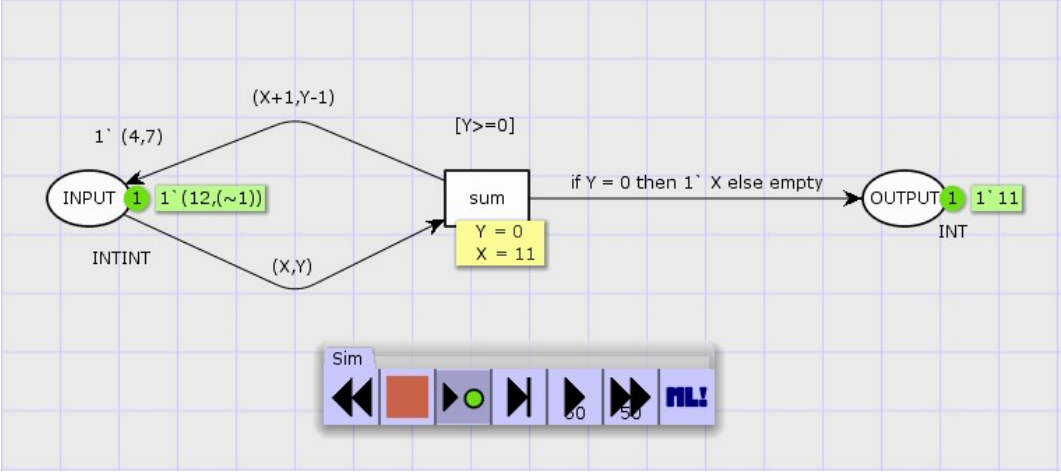

Figure 21: Sum of two numbers

3.5 FACTORIAL OF A NUMBER

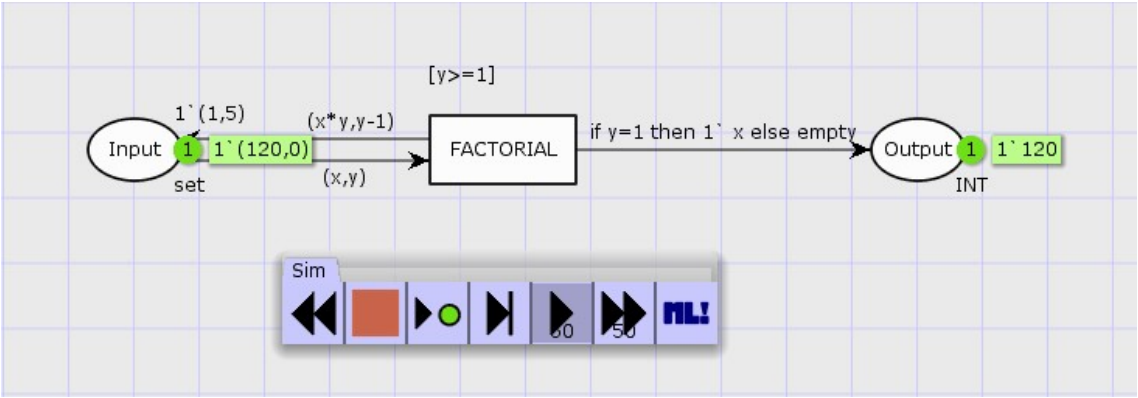

Figure 22: Factorial of a number

3.6 CHECK FOR PRIME NUMBER

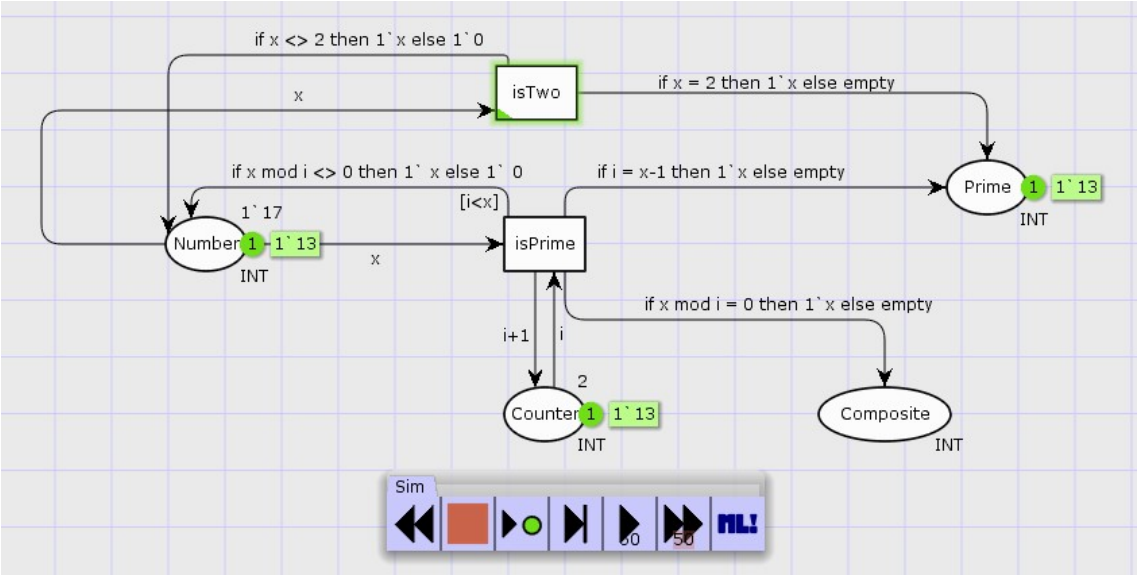

Figure 23: Check for prime number

3.7 CHECK FOR LEAP YEAR

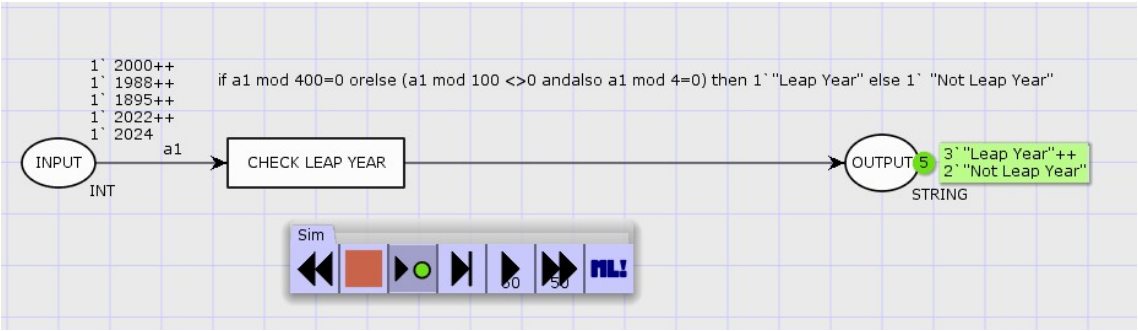

Figure 24: Check for Leap year

3.8 ATM SYSTEM LOGIN

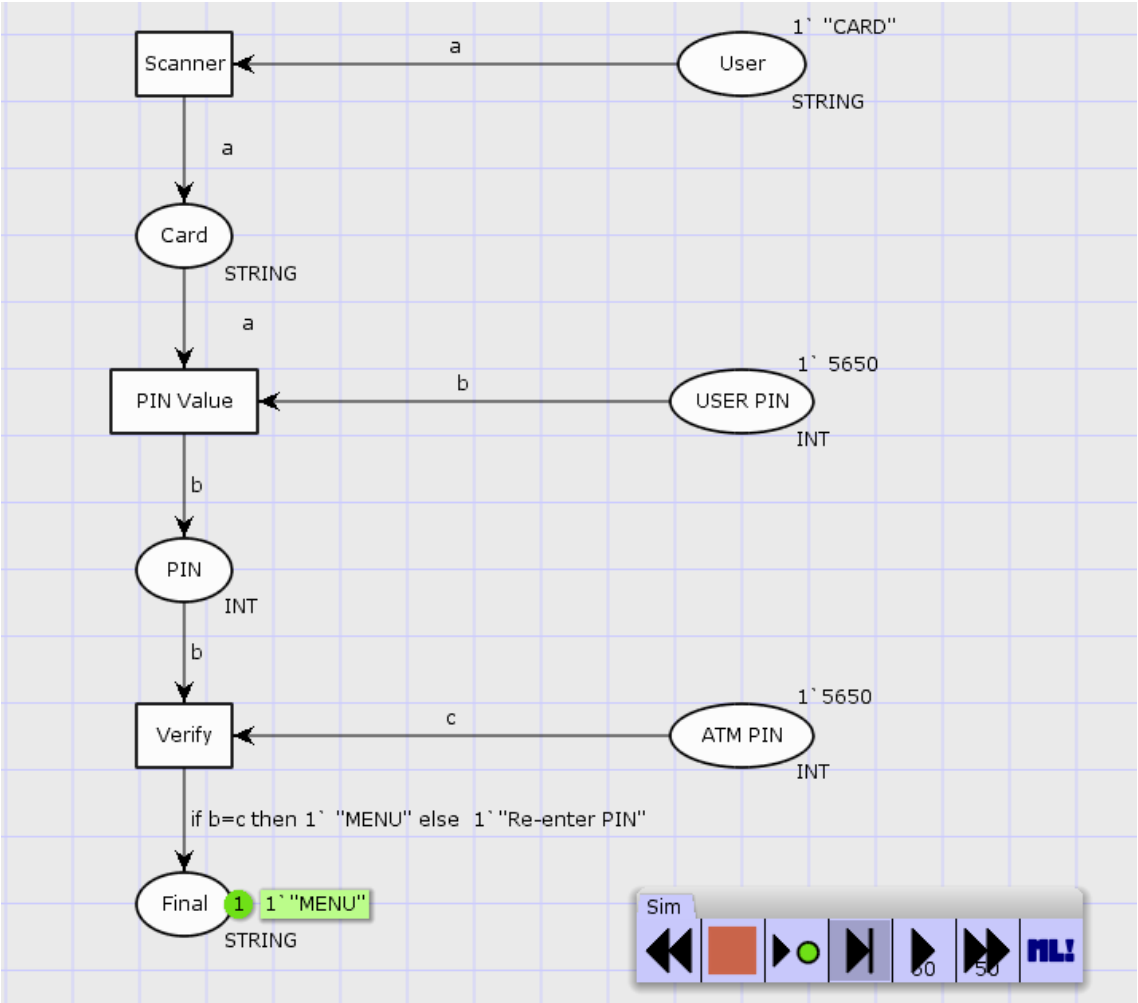

Figure 25: ATM System

3.9 ATM SYSTEM: DISPLAY MONEY

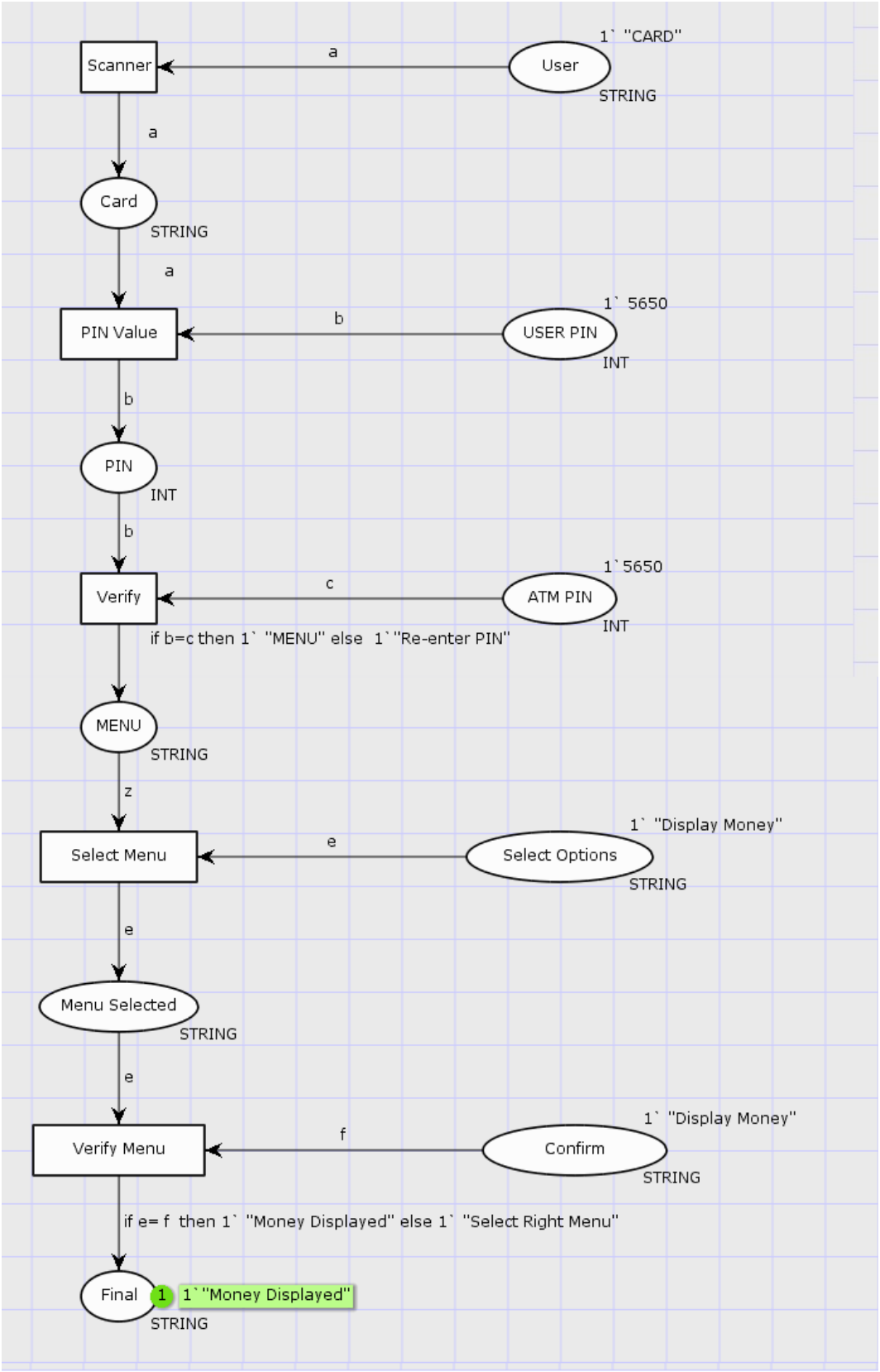

Figure 26: Money

3.10 ATM SYSTEM: WITHDRAW MONEY

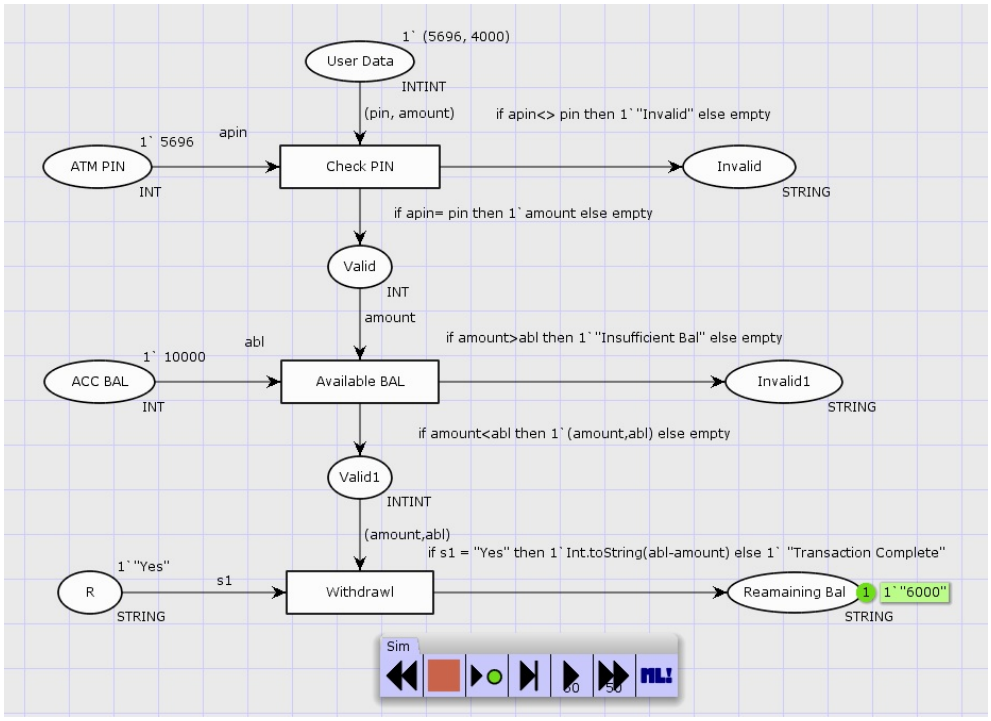

Figure 27: Money Withdraw

3.11 ATM SYSTEM: DEPOSIT MONEY

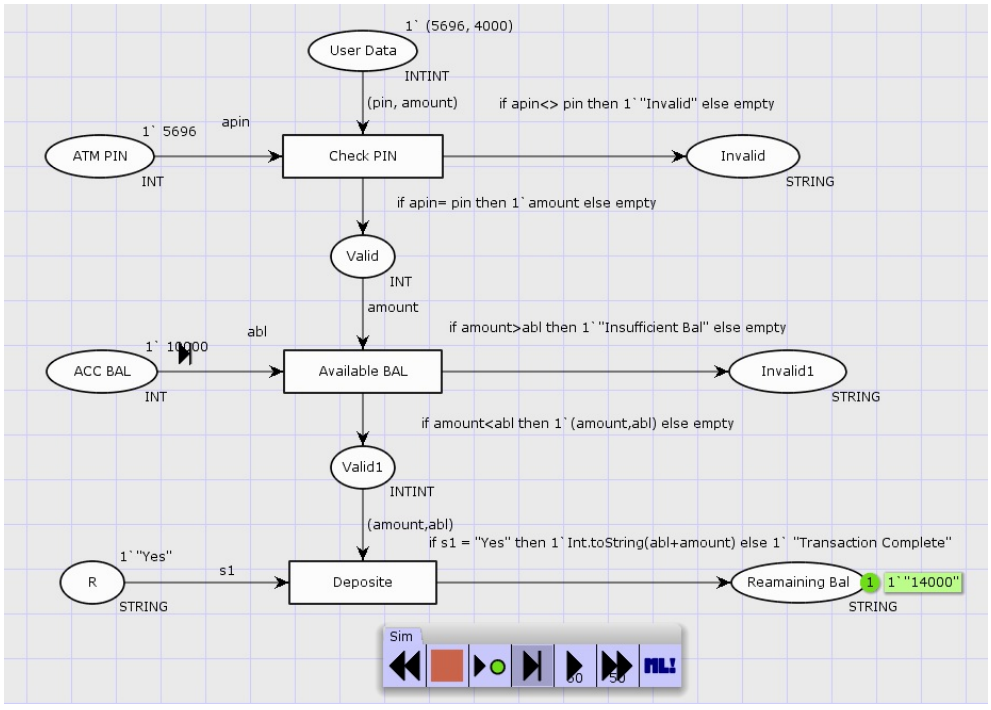

Figure 28: Deposit Money in ATM

3.12 DINNING PHILOSOPHER

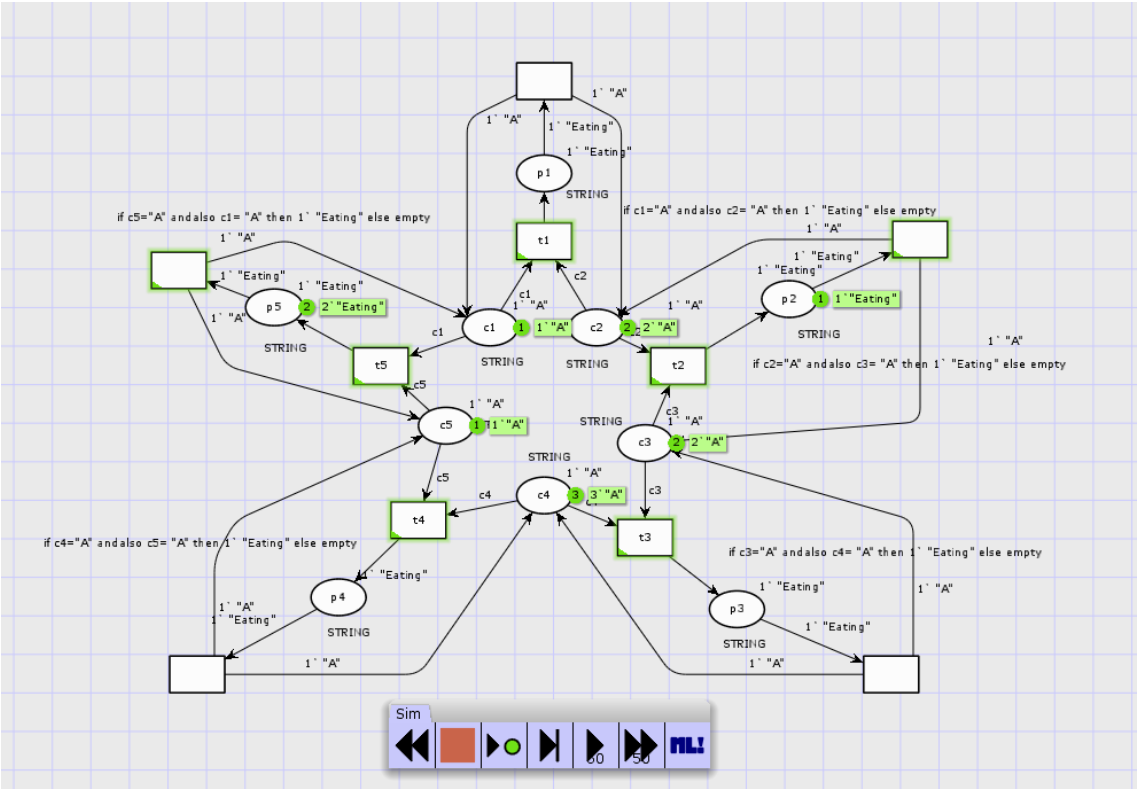

Figure 29: Dinning Philosopher

4 LTSA

LTSA is a verification tool for concurrent systems. It mechanically checks that the specification of a concurrent system satisfies the properties required of its behaviour. In addition, LTSA supports specification animation to facilitate interactive exploration of system behaviour.

A system in LTSA is modelled as a set of interacting finite state machines. The properties required of the system are also modelled as state machines. LTSA performs compositional reachability analysis to exhaustively search for violations of the desired properties. More formally, each component of a specification is described as a Labelled Transition System (LTS), which contains all the states a component may reach and all the transitions it may perform.

4.1 ATM SYSTEM

```
1 ATM=CARD,
2 CARD=(card->PIN),
3 PIN=(pin->MENU | invalidcard-> CARD),
4 MENU=(wrongpin->PIN | option-> OPTIONS),
5 OPTIONS=(balanceCheck-> BALCHECK | withdraw-> WITHDRAW | deposite
     -> DEPOSIT | pinchange-> PINCHANGE),
6 BALCHECK=(balance->ATM),
7 WITHDRAW=(collect->ATM),
8 DEPOSIT=(submit->ATM),
9 PINCHANGE=(pinchanged->ATM).
```

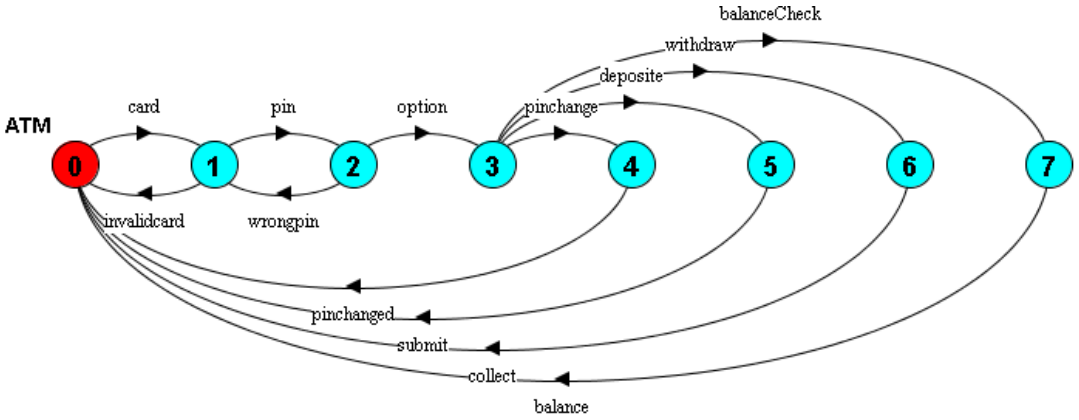

Figure 30: ATM System

4.2 NITRIS SYSTEM

```
NITRIS=LOGIN,
LOGIN=(credentials->AUTH),
AUTH=(vallidcredential->DASHBOARD | invalid-> LOGIN),
DASHBOARD=(studentdetails-> STUDENT_DETAILS | examdetails->
   EXAMINATION |  register->REGISTRATION | check_fee->FEEPAYMENT
   | feedback->FEEDBACK |logout->NITRIS ),
STUDENT_DETAILS=( details_updated->DASHBOARD),
EXAMINATION= ( checked_timing->DASHBOARD),
REGISTRATION= ( registration_done->DASHBOARD),
FEEPAYMENT= ( payment_done->DASHBOARD),
FEEDBACK= (feedback_submitted ->DASHBOARD).
```

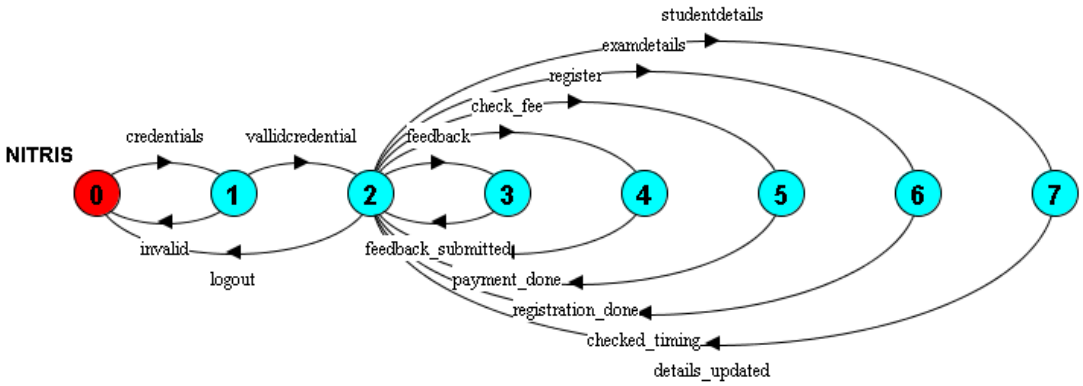

Figure 31: NITRIS System