Pseudo Code

Contents

Propositions

No.	Propositions
1	Agents operate in an organizational environment where they perform actions on objects to achieve goals.
2	An action is always performed by one agent. Agents can perform many actions.
3	An action is always performed on one object. Many actions can be performed on an object.
4	An object has one state at any given time.
5	An action performed on an object may cause a state transition for that object.
6	A postcondition of an action is comprised of one or more objects and their corresponding states after the action is performed. The postcondition should at least include the object on which the action is performed and the intended state change of that action.
7	A precondition of an action is comprised of zero or more objects and their corresponding states before the action can be performed.
8	An object in a state can be part of many preconditions and many postconditions.

Use Case Model

Map

Set of Related Extended User Stories	**Use Case Model**
Agent	Actor
Action - Object	Use Case

Rules

- **Mapping rule 1.1:** For each *unique* agent in the set of related extended user stories, add an actor to the use case diagram.
- **Mapping rule 1.2:** For each *unique* action – object combination (i.e., the action is performed on the object) in the set of related extended user stories, add a use case to the use case diagram.
- **Mapping rule 1.3:** For each agent that performs an action on an object in the set of related extended user stories, draw an association between the actor that represents the agent and the use case that represents the action-object combination.

Algorithm

Input: Structured representation of a set of BDD scenarios that document related user stories
Output: Use case diagram

```
1:  for each row in the structured representation
2:      if the agent is not already represented as an actor then
3:          create a new actor
4:      end if
5:      if the action – object is not already represented as a use case then
6:          create a new use case
7:      end if
8:      if the actor representing the agent and the use case representing the action -
        object are not already connected by an association then
9:          create an association between the actor and the use case
10:     end if
11: end for
```

Domain Model

Map

Set of Related Extended User Stories	Domain Model
Agent	Entity
Object	Entity
Action	Relationship

Rules

- **Mapping rule 2.1:** For each *unique* agent in the set of related extended user stories, add an entity to the domain model.
- **Mapping rule 2.2:** For each *unique* object in the set of related extended user stories, add an entity to the domain model.
- **Mapping rule 2.3:** For each agent that performs an action on an object in the set of related extended user stories, draw a relationship between the entity that represents the agent and the entity that represents the object. Label the relationship with the action performed by the agent on the object.

Algorithm

```
Input: Structured representation of a set of BDD scenarios that document related user
stories
Output: class diagram
1:    for each row in the structured representation
2:        if the agent is not already represented as a class then
3:            create a new class
4:        end if
5:        if the object is not already represented as a class then
6:            create a new class
7:        end if
8:        if the class representing the agent and the class representing the object
          are not already connected by an association representing the action
          then
9:            create an association between the two classes and label the
              association with the name of the action
10:       end if
11:   end for
```

State Machine

Map

Set of Related Extended User Stories	**State Machine for Object$_i$ (i ∈{1, .., n}, with n = number of unique objects on which actions are performed in the set of related extended user stories)**
Precondition State of Precondition Object$_i$	State
Postcondition State of Postcondition Object$_i$	State
Action on Object$_i$	Transition

Rules

- **Mapping rule 3.1:** For each *unique* object on which an action is performed in the set of related extended user stories, create a state machine.
- **Mapping rule 3.2:** For each *unique* precondition or postcondition state of a specific object in the set of related extended user stories, add a state to the corresponding state machine.
- **Mapping rule 3.3:** If a state of an object is in the precondition of an extended user story and a different state of the same object is in the postcondition of the same user story, draw a transition from the state that represents the precondition state to the state that represents the postcondition state in the state machine for that object. The label of the transition is the action performed on the object.
- **Mapping rule 3.4:** If a state of an object is in the precondition of an extended user story and this state is not in the postcondition of any other extended user story, then indicate in the state machine for the object that this is an initial state.
- **Mapping rule 3.5:** If a state of an object is in the postcondition of an extended user story, and this state is not in the precondition of any other extended user story or it is in the precondition of another extended user story but not in the postcondition of that other extended user story, then indicate in the state machine for the object that this is a final state.
- **Mapping rule 3.6:** If a state of an object is in the postcondition of an extended user story and this object is not in the precondition of the same extended user story, then create a ‘unknown’ state in the state machine of the object and draw a transition from this ‘unknown’ state to the state representing the postcondition state in the state machine for the object. The label of the transition is the action of the extended user story

Algorithm

Input: Structured representation of a set of BDD scenarios that document related user stories
Output: A set of state machine diagrams

```
1:      for each row in the structured representation
2:          if the object on which the action is performed is not already
            represented as a state machine diagram then
3:              create a new state machine diagram for this object
                (confer mapping rule 3.1)
```

```
4:          end if
5:      end for
6:      for each row i in the structured representation
7:          if for the precondition object a state machine diagram was created
            then
8:              if the precondition state is not already represented as a state in
                that state machine diagram then
9:                  create a new state in the state machine diagram for the
                    object
                    (confer mapping rule 3.2)
10:             end if
11:             set initial state to true
12:             for each row j in the structured representation (action row i ≠
                action row j)
13:                 if postcondition object row j = precondition object row i
                    AND postcondition state in row j = precondition state
                    in row i then
14:                     set initial state to false
15:                 end if
16:             end for
17:             if initial state is true then
18:                 indicate that the state representing the precondition
                    state in the state machine diagram for the object is an
                    initial state
                    (confer mapping rule 3.4)
19:             end if
20:         end if
21:         if for the postcondition object a state machine diagram was created
            then
22:             if the postcondition state is not already represented as a state in
                that state machine diagram then
23:                 create a new state in the state machine diagram for the
                    object
                    (confer mapping rule 3.2)
24:             end if
25:             set final state to true
26:             for each row j in the structured representation (action row i ≠
                action row j)
27:                 if precondition object row j = postcondition object row i
                    AND precondition state in row j = postcondition state
                    in row i AND postcondition object in row j =
                    postcondition object in row i then
28:                     set final state to false
29:                 end if
30:             end for
31:             if final state is true then
```

```
32:                 indicate that the state representing the postcondition
                    state in the state machine diagram for the object is a
                    final state
                    (*confer mapping rule 3.5*)
33:             **end if**
34:         **end if**
35:         **if** precondition object = postcondition object AND postcondition state
            $\neq$ precondition state **then**
                create in the state machine diagram for the object, a transition
                from the state that represents the precondition state to the state
                that represents the postcondition state and label the transition
                with the name of the action
                (*confer mapping rule 3.3*)
36:         **end if**
37:         set not in precondition to true
38:         **for each** row *j* in the structured representation (action row *i* = action
            row *j*)
39:             **if** postcondition object row *i* = precondition object row *j* **then**
40:                 set not in precondition to false
41:             **end if**
42:         **end for**
43:         **if** not in precondition is true **then**
44:             create in the state machine diagram for the object, a state
                labelled ‘unknown’ and create a transition from this ‘unknown’
                state to the state that represents the postcondition state and
                label the transition with the name of the action
                (*confer mapping rule 3.6*)
45:         **end if**
46:     **end for**
```

Process Model

Map

Set of Related Extended User Stories	**Process Model**
Agent	Swimlane
Action - Object	Activity
Postcondition State of Postcondition Object in an extended user story *is equal to* Precondition State of Precondition Object in *some other* extended user story	Sequence Flow
Precondition State of Precondition Object in an extended user story *is not equal to* Postcondition State of Postcondition Object in *any other* extended user story	Start Event
Postcondition State of Postcondition Object in an extended user story *is not equal to* Precondition State of Precondition Object in *any other* extended user story	End Event
Multiple Object – State pairs in the Precondition of an extended user story	Gateway
Multiple Object – State pairs in the Postcondition of an extended user story	Gateway

Rules

- **Mapping rule 4.1:** For each *unique* agent in the set of related extended user stories, add a swimlane to the process model.
- **Mapping rule 4.2:** For each *unique* action – object pair (i.e., the action is performed on the object) in the set of related extended user stories, add an activity to the process model in the swimlane for the agent who performs the action on the object.
- **Mapping rule 4.3:** If the state of an object in the postcondition of an extended user story is equal to the state of that object in the precondition of *another* user story, then add a sequence flow from the activity corresponding to the action – object pair of the first user story to the activity corresponding to the action – object pair of the second user story.
- **Mapping rule 4.4:** If the state of an object in the precondition of an extended user story is not equal to the state of that object in the postcondition of *any other* extended user story, then add a start event to the process model and connect it with a sequence flow to the activity representing the action described in the extended user story which precondition was considered.
- **Mapping rule 4.5:** If the state of an object in the postcondition of an extended user story is not equal to the state of that object in the precondition of *any other* extended user story, then add an end event to the process model and connect the activity representing the action described in the extended user story which postcondition was considered with a sequence flow to this end event.
- **Mapping rule 4.6:** If there is a disjunction of object-state pairs in the precondition of an extended user story, then add an OR gateway *before* the activity representing the action-object pair of the user story. This OR gateway merges the incoming sequence flows of the activity.
- **Mapping rule 4.7:** If there is a conjunction of object-state pairs in the precondition of an extended user story, then add an AND gateway *before* the activity representing the action-

object pair of the user story. This AND gateway merges the incoming sequence flows of the activity.

- **Mapping rule 4.8:** If there is a conjunction of object-state pairs in the postcondition of an extended user story, then add an AND gateway *after* the activity representing the action-object pair of the user story. This AND gateway splits the outgoing sequence flows of the activity.

Algorithm

Part 1

Input: Structured representation of a set of BDD scenarios that document related user stories
Output: A partial activity diagram (Stage 1)

```
1:  for each row in the structured representation
2:      if the agent is not already represented as a swimlane then
3:          create a new swimlane for the agent
4:      end if
5:      if the action-object pair is not already represented as an activity then
6:          create a new activity for the action-object pair in the swimlane
            representing the agent performing the action on the object
7:      end if
8:  end for
```

Part 2

Input: Structured representation of a set of BDD scenarios that document related user stories & Stage 1 activity diagram generated from that structured representation
Output: A partial activity diagram (Stage 2)

```
9:   for each row i in the structured representation
10:      for each row j in the structured representation (action row i ≠ action row j)
11:          if postcondition object of row i = precondition object of row j AND
             postcondition state of row i = precondition state of row j then
12:              draw a control flow from the activity representing the action-
                 object pair in row i to the activity representing the action-
                 object pair in row j
13:          end if
14:      end for
15:  end for
```

Part 3

Input: Structured representation of a set of BDD scenarios that document related user stories & Stage 2 activity diagram generated from that structured representation
Output: A partial activity diagram (Stage 3)

```
16:  for each row i in the structured representation
17:      set unmatched precondition to true
```

```
18:         set unmatched postcondition to true
19:         for each row $j$ in the structured representation (action row $i \neq$ action row $j$)
20:             if precondition object of row $i$ = postcondition object of row $j$ AND
                precondition state of row $i$ = postcondition state of row $j$ then
21:                 set unmatched precondition to false
22:             end if
23:             if postcondition object of row $i$ = precondition object of row $j$ AND
                postcondition state of row $i$ = precondition state of row $j$ then
24:                 set unmatched postcondition to false
25:             end if
26:         end for
27:         if unmatched precondition is true AND no start node has already been
            connected to the activity representing the action-object pair in row $i$ then
28              create a start node and draw a control flow from the start node to the
                activity representing the action-object pair in row $i$
29:         end if
30:         if unmatched postcondition is true AND no end node has already been
            connected to the activity representing the action-object pair in row $i$ then
31:             create an end node and draw a control flow to the end node from the
                activity representing the action-object pair in row $i$
32:         end if
33:     end for
```

Part 4

Input: Structured representation of a set of BDD scenarios that document related user stories & Stage 4 activity diagram generated from that structured representation
Output: An activity diagram (Stage 4)

```
34:     for each row $i$ in the structured representation
35:         if an XOR logical operator was added to the precondition object then
36:             create a merge node in the activity diagram
37:             connect all control flows directed to the activity that represents the
                action of row $i$, to this merge node as incoming control flows
38:             create a control flow going out of the merge node and direct it to the
                activity that represents the action of row $i$
39:         end if
40:         Skip all rows $j$ (action row $i$ = action row $j$)
41:     end for
42:     for each row $i$ in the structured representation
43:         if an AND logical operator was added to the precondition object then
45:             create a join node in the activity diagram
46:             connect all control flows directed to the activity that represents the
                action of row $i$, to this join node as incoming control flows
47:             create a control flow going out of the join node and direct it to the
                activity that represents the action of row $i$
48:         end if
```

```
49:         Skip all rows j (action row i = action row j)
50:     end for
51:     for each row i in the structured representation
52:         if an AND logical operator was added to the postcondition object then
53:             create a fork node in the activity diagram
54:             connect all control flows leaving from the activity that represents the
                action of row i, to this fork node as outgoing control flows
55:             create a control flow from the activity that represents the action of
                row i, to the fork node
56:         end if
57:         Skip all rows j (action row i = action row j)
58:     end for
```