| **Use Case 1.1** | **Add RateConstant** |
|---|---|
| *Primary Actor:* | End-User |
| *Preconditions:* | • An instance of MarkovModel is initialized<br>• A RateConstant is initialized with valid parameters and stored in a smart_ptr |
| *Postconditions:* | • The RateConstant is added to MarkovModel's map_of_rates with its name as the key<br>• MarkovModel is set to unvalidated |

*Main Success Scenario:*

1. The user calls MarkovModel's method addRateConstant() with a smart_ptr to the initialized RateConstant as the only arguement
2. MarkovModel searches map_of_rates for a RateConstant with the same key value as the RateConstant to be added; none exist
3. The RateConstant is added to map_of_rates with its name as the key

*Extensions:*

2a. A RateConstant with the same name exists in map_of_rates:

   1. System throws a runtime exception notifying the user that the RateConstant already exists
   2. map_of_rates is unmodified

| **Use Case 1.2** | **Add State** |
|---|---|
| *Primary Actor:* | End-User |
| *Preconditions:* | • An instance of MarkovModel is initialized<br>• A State is initialized with valid parameters and stored in a smart_ptr |
| *Postconditions:* | • The State is added to MarkovModel's map_of_states with its name as the key<br>• MarkovModel is set to unvalidated |

*Main Success Scenario:*

1. The user calls MarkovModel's method addState() with a smart_ptr to the initialized State as the only arguement
2. MarkovModel searches map_of_states for a State with the same key value as the State to be added; none exist
3. The State is added to map_of_states with its name as the key

*Extensions:*

2a. A State with the same name exists in map_of_states:

1. System throws a runtime exception notifying the user that the State already exists
2. map_of_states is unmodified

| **Use Case 1.3** | **Add Connection** |
|---|---|
| *Primary Actor:* | End-User |
| *Preconditions:* | • An instance of MarkovModel is initialized<br>• A Connection is initialized with valid parameters and stored in a smart_ptr |
| *Postconditions:* | • The Connection is added to MarkovModel's vector of connections<br>• MarkovModel is set to unvalidated |
| *Main Success Scenario:*<br>1. The user calls MarkovModel's method addConnection() with a smart_ptr to the initialized Connection as the only arguement<br>2. MarkovModel searches the vector of connections for a Connection between the same States as the Connection to be added; none exist<br>3. The Connection is added to the vector of connections | |
| *Extensions:*<br>2a. A Connection between the same States exists in the vector of connections<br>1. System throws a runtime exception notifying the user that the two States in the Connection have already been connected<br>2. The vector of connections is unmodified | |

| **Use Case 1.4** | **Set initial State** |
|---|---|
| *Primary Actor:* | End-User |
| *Preconditions:* | • An instance of MarkovModel is initialized |
| *Postconditions:* | • MarkovModel's initial State is set to the specified value<br>• MarkovModel is set to unvalidated |
| *Main Success Scenario:*<br>1. The user calls MarkovModel's method setInitialState() with the name of the State as the only argument<br>2. MarkovModel checks if the initial State already been set; it has not<br>3. The initial State is set to the specified value | |
| *Extensions:*<br>2a. The initial State has already been set<br>1. System throws a runtime exception notifying the user that the initial State has already been set<br>2. The initial State is unmodified | |

| **Use Case 1.5** | **Validate MarkovModel** |
|---|---|
| *Primary Actor:* | End-User |
| *Preconditions:* | • An instance of MarkovModel is initialized |
| *Postconditions:* | • The user is notified that the MarkovModel is correct<br>• MarkovModel's validation flag is set to true<br>• The states in the map_of_states are assigned unique indices<br>• The State which was defined as initial State has its inital_state flag set |

*Main Success Scenario:*

1. The user calls MarkovModel's method validate() with a smart_ptr to a valid StateOfTheWorld instance as the only arguement
2. The system confirms that at least one Connection has been defined
3. The system confirms that every State referenced in the vector of connections has a corresponding entry in the map_of_states
4. The system confirms that every RateConstant referenced in the vector of connections has a corresponding entry in the map_of_rates
5. The system confirms that the initial State has been defined
6. For every LigandGatedRateConstant in the map_of_rates, the system confirms that the Concentration has been declared
7. The system confirms that no model errors have been found in during validation process
8. The system sets the validation flag to true
9. The system sets the state indices in every State in the map_of_states
10. The system sets the initial_state flag on the initial State
11. The system returns the validation results to the user with a success result, no errors, and no warnings

*Extensions:*

2a. No connections have been defined

  1. The system sets the validation error flag
  2. The system saves a no Connections error message in the vector of error messages
  3. The system resumes at step 3

3a. A State referenced in the vector of connections does not exist in the map_of_states

  1. The system sets the validation error flag
  2. The system saves a State not defined error message in the vector of error messages
  3. The system resumes at step 4

4a. A RateConstant referenced in the vector of connections does not exist in the map_of_rates

  1. The system sets the validation error flag
  2. The system saves a RateConstant not defined error message in the vector of error messages
  3. The system resumes at step 5

5a. The initial State has not beed defined

  1. The system sets the validation error flag
  2. The system saves an initial State not defined error message in the vector of error messages
  3. The system resumes at step 6

6a. The pointer to the StateOfTheWorld is NULL

  1. The system sets the validation error flag
  2. The system saves a StateOfTheWorldIsNull error message in the vector of error messages
  3. The system returns the validation results to the user

6b. A Concentration referenced by a LigandGatedRateConstant has not beed defined

  1. The system sets the validation error flag
  2. The system saves a Concentration not defined error message in the vector of error messages
  3. The system returns the validation results to the user

7a. The system has detected errors during the validation process

  1. The system returns the validation results to the user