Contents

Document of conception

This document describes the conception of our chat System project. The realization of the project is accessible from https://github.com/paulEtse/ProjetJava.

I. Use case Diagram

This diagram describes the main use cases of our system.

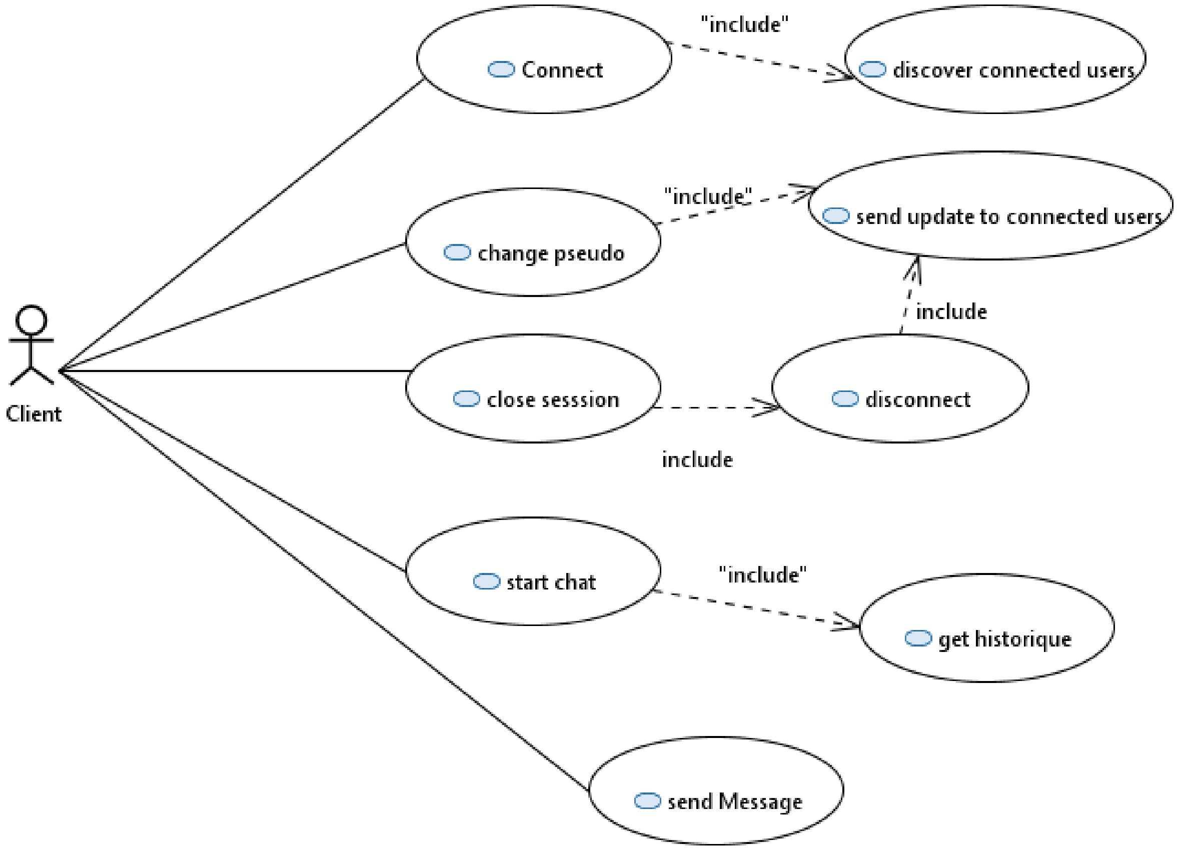

Figure 1 Use Case diagram

II. Class diagrams

The classes derived from our conception and the interaction between those classes are describe in the diagram below. It contains the class

- *Agent*: it represents the user interface between the user and our system. It handles all the actions of the user and call the corresponding method of the system.
- *ManageNewConnection*: as its name explain it, this class is used to manage new connection. It communication with other systems to handle new connection requests, new change of pseudo and maintain the current connectedUserList up to date.
- *ManageChat*: this class is used for the communication between two systems. It is used to send or receive messages.
- *User*: this class represent the user in our system.
- *Message*: it represents a message exchanged in the system

  MessageType: it indicates the type of the messages exchanged. A message can be a connection request, a connection response, a file, a text, …

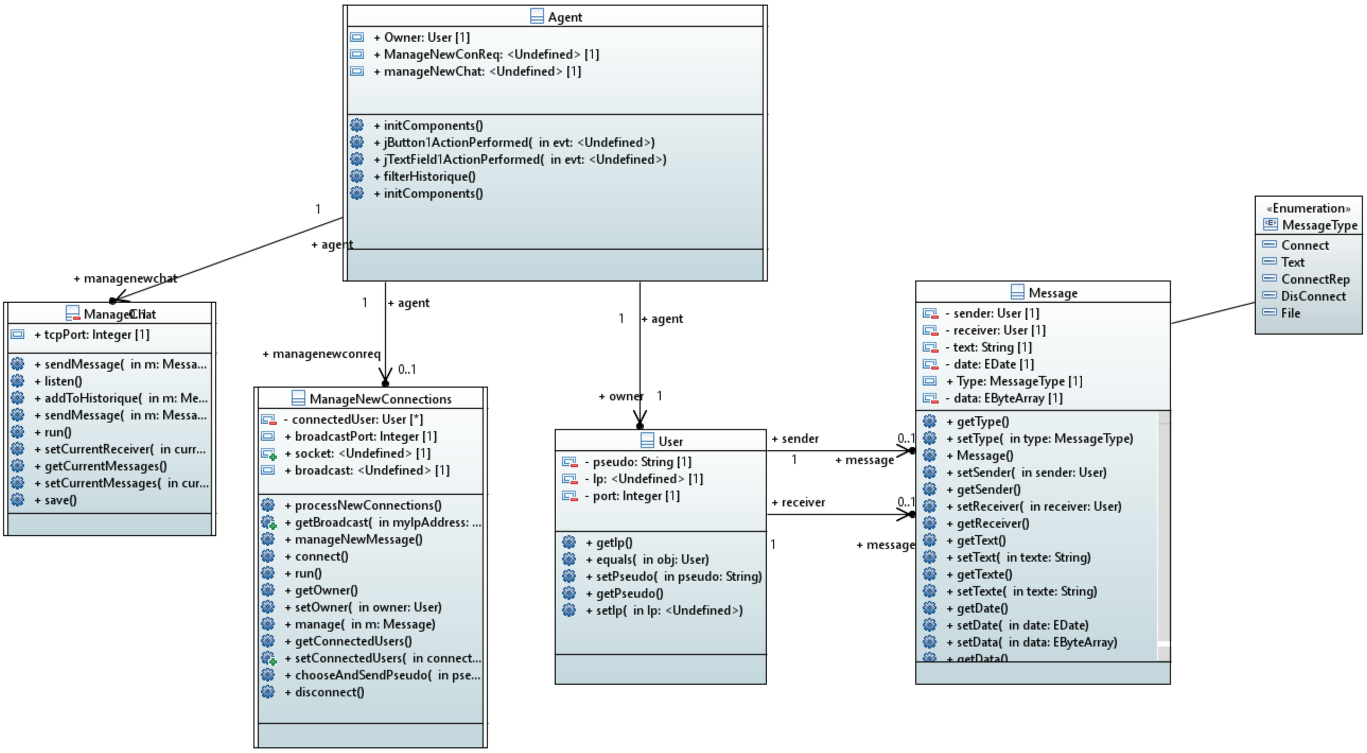
Agent
+ Owner: User [1]
+ ManageNewConReq: <Undefined> [1]
+ manageNewChat: <Undefined> [1]
+ initComponents()
+ jButton1ActionPerformed(in evt: <Undefined>)
+ jTextField1ActionPerformed(in evt: <Undefined>)
+ filterHistorique()
+ initComponents()
1
+ agent
+ managenewchat
ManageChat
+ tcpPort: Integer [1]
+ sendMessage(in m: Messa...
+ listen()
+ addToHistorique(in m: Me...
+ sendMessage(in m: Messa...
+ run()
+ setCurrentReceiver(in curr...
+ getCurrentMessages()
+ setCurrentMessages(in cur...
+ save()
1
+ agent
+ managenewconreq
0..1
ManageNewConnections
- connectedUser: User [*]
+ broadcastPort: Integer [1]
+ socket: <Undefined> [1]
+ broadcast: <Undefined> [1]
+ processNewConnections()
+ getBroadcast(in myIpAddress: ...
+ manageNewMessage()
+ connect()
+ run()
+ getOwner()
+ setOwner(in owner: User)
+ manage(in m: Message)
+ getConnectedUsers()
+ setConnectedUsers(in connect...
+ chooseAndSendPseudo(in pse...
+ disconnect()
1
+ agent
+ owner
1
User
- pseudo: String [1]
- Ip: <Undefined> [1]
- port: Integer [1]
+ getIp()
+ equals(in obj: User)
+ setPseudo(in pseudo: String)
+ getPseudo()
+ setIp(in Ip: <Undefined>)
+ sender
1
+ message
0..1
+ receiver
1
+ message
0..1
Message
- sender: User [1]
- receiver: User [1]
- text: String [1]
- date: EDate [1]
+ Type: MessageType [1]
- data: EByteArray [1]
+ getType()
+ setType(in type: MessageType)
+ Message()
+ setSender(in sender: User)
+ getSender()
+ setReceiver(in receiver: User)
+ getReceiver()
+ getText()
+ setText(in texte: String)
+ getTexte()
+ setTexte(in texte: String)
+ getDate()
+ setDate(in date: EDate)
+ setData(in data: EByteArray)
+ getData()
«Enumeration»
MessageType
Connect
Text
ConnectRep
DisConnect
File

Figure 2 Class Diagram

III. Sequence diagrams

In the following paragraphs, we’ll describe our sequence diagrams.

1. Scenario “ Connect ”

This diagram describes the connection scenario. When a user runs the app, a connection request is sent to all connected users. The answer to this request by providing their pseudo. Then the new user can set his pseudo and Agent must check if he can use it or not.

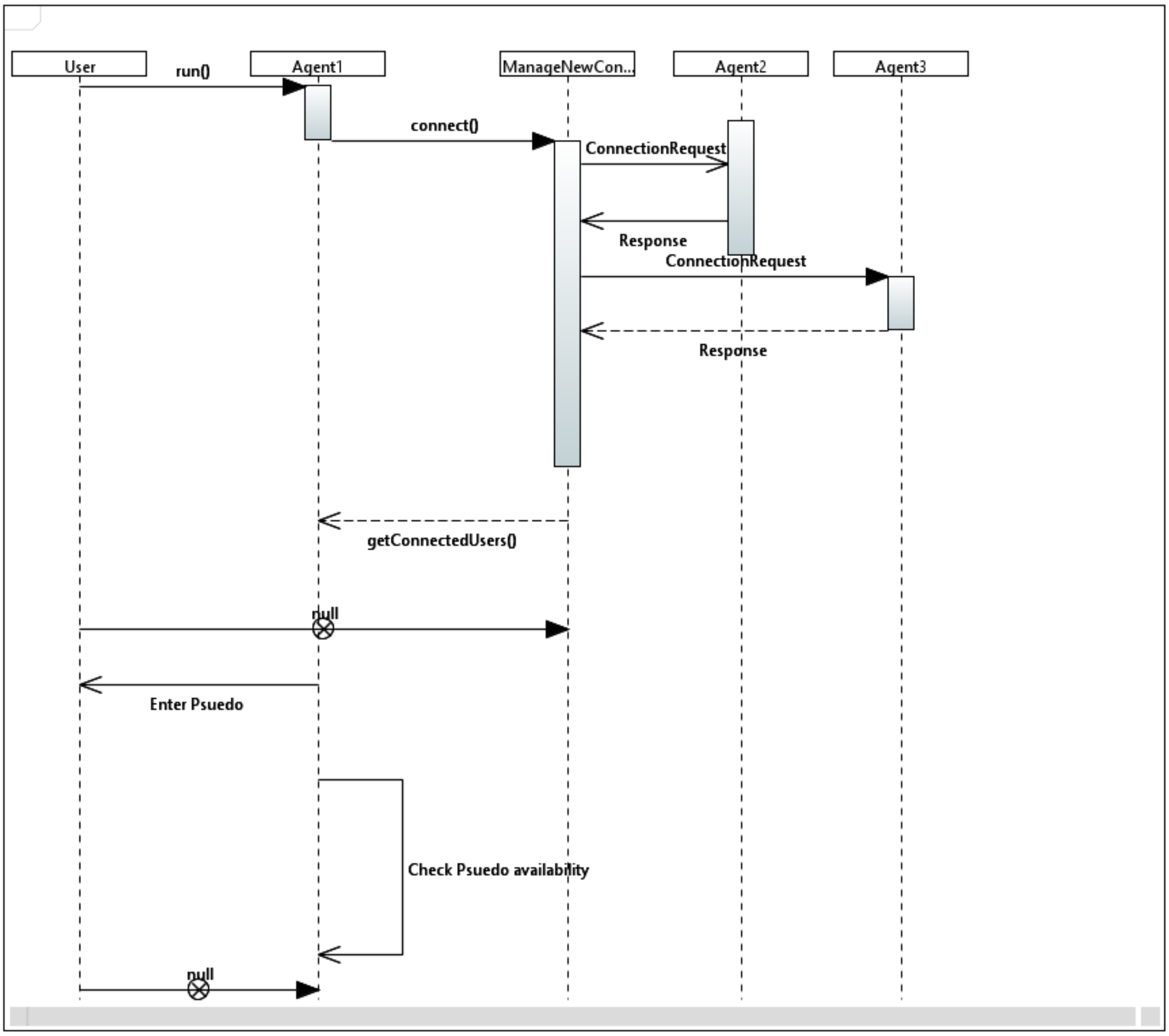

Figure 3 Connection Scenario

2. Scenario “send message”

In other to discuss with a specific user, a user has to select the name of the host user in connectedUserLIst. After that the system will display the historic of the exchange between those users. Finally, the user can send his messages.

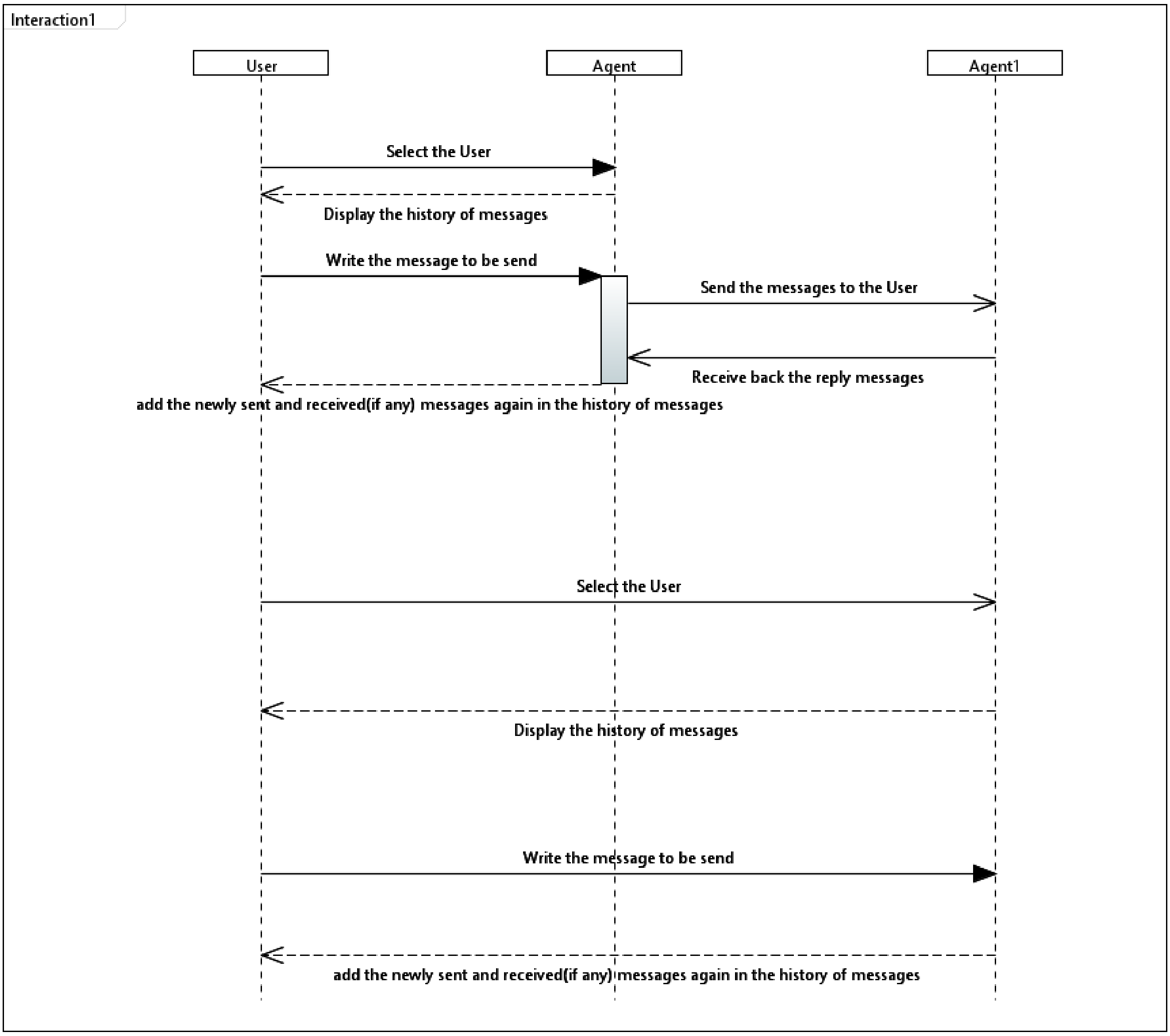
Interaction1
User
Agent
Agent1
Select the User
Display the history of messages
Write the message to be send
Send the messages to the User
Receive back the reply messages
add the newly sent and received(if any) messages again in the history of messages
Select the User
Display the history of messages
Write the message to be send
add the newly sent and received(if any) messages again in the history of messages

Figure 4 Exchange messages with a user

3. Scenario “Change pseudo”

In other to change one’s pseudo, a user will set put his new pseudo and the System must check it availability. If the pseudo is available, the user’s pseudo will be updated and the system must send the update to the connected users.

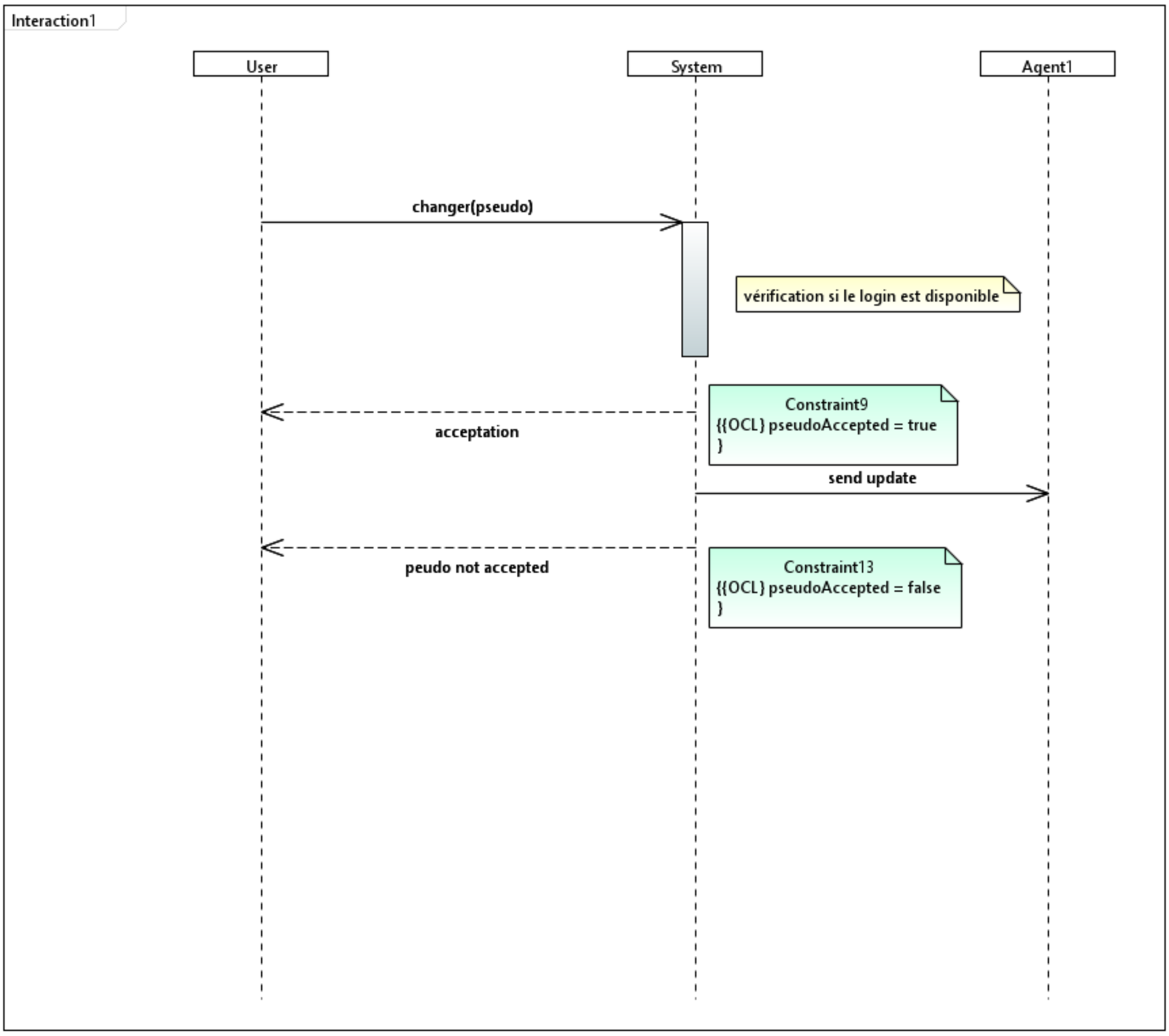

Figure 5 Scenario Change pseudo

4. Close section

When a user closes his section, the system will inform the connected users so they can update their connectedUserList.

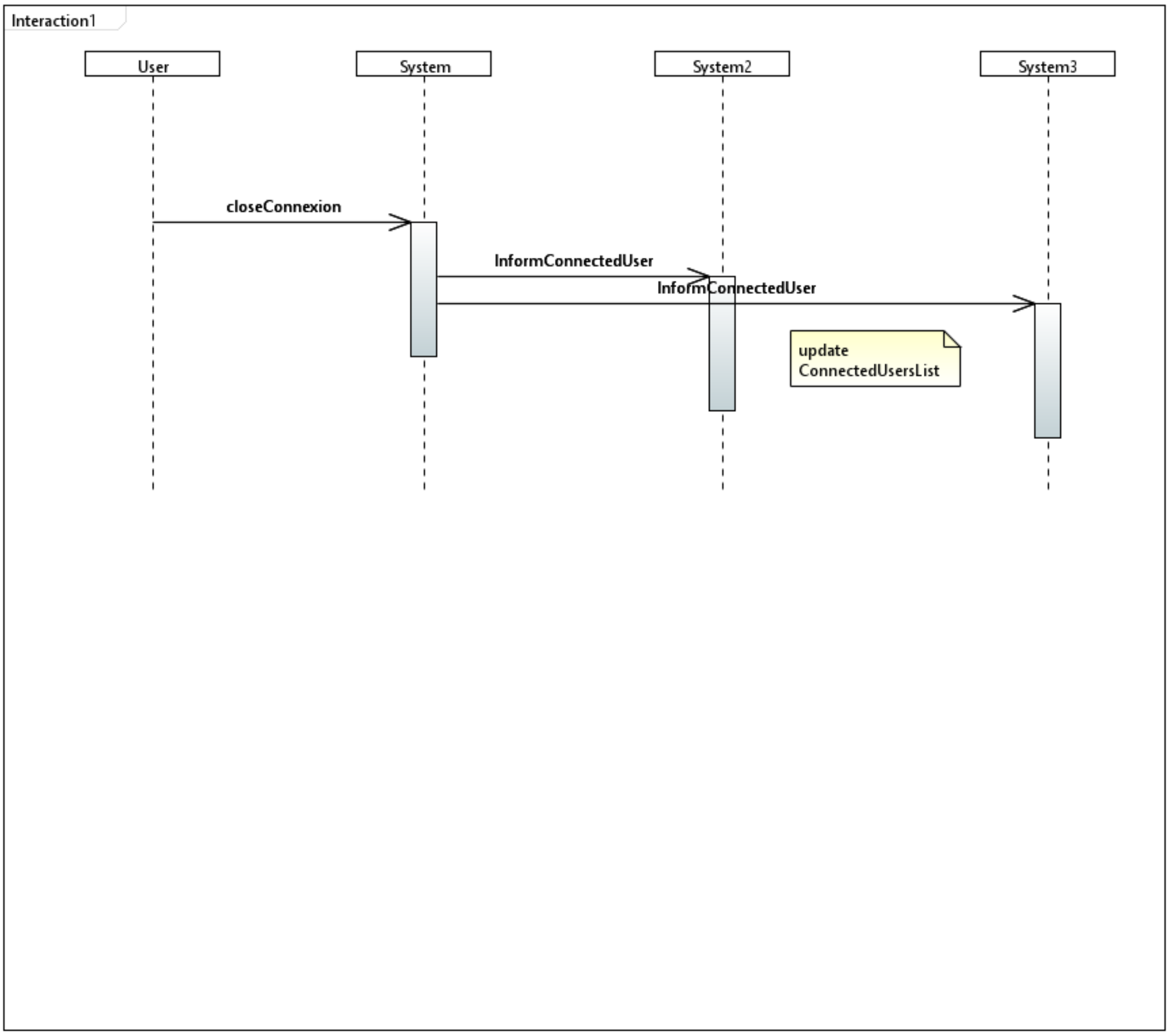

Figure 6 Scenario Close section

IV. Composite structure

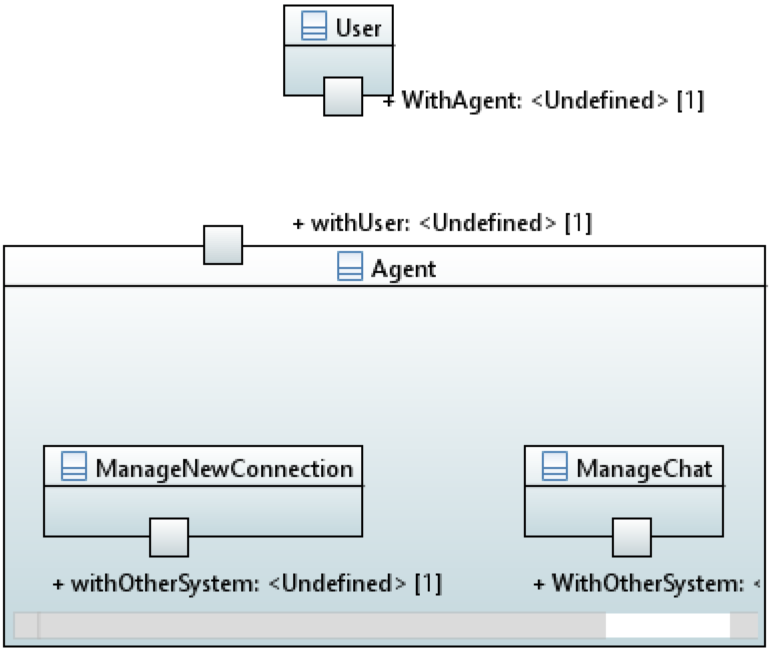
User
+ WithAgent: <Undefined> [1]
+ withUser: <Undefined> [1]
Agent
ManageNewConnection
ManageChat
+ withOtherSystem: <Undefined> [1]
+ WithOtherSystem: <

Figure 7 Composite Structure Diagrams

V. State machine

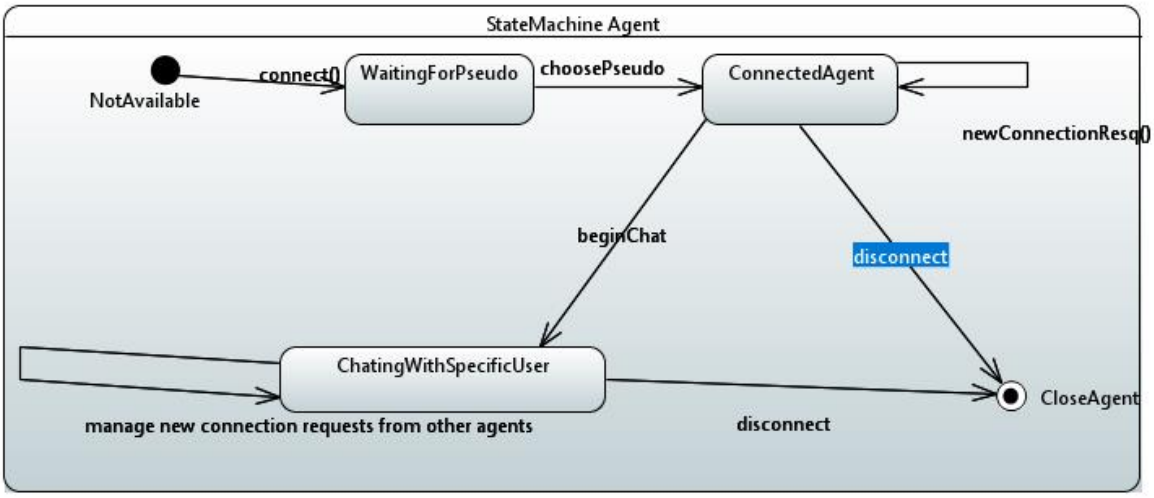

Figure 8 Statechart Diagrams