Communication Model for Cooperative Robotics Simulator

Formal Requirement Specification

Version 2.0

---

1. Introduction

The purpose of this document is to provide formal requirement specification of "Communication Model for Cooperative Robotics Simulator". This specification uses UML/OCL methodology. The constraint and variant in the specification are based on the critical requirement as stated in the Software Requirement Specification Version 1.1 and Class Diagram presented in Architecture Design Version 1.0. Furthermore, we use the UML- based Specification Environment (USE) tool to check the type and syntax to ensure correctness of the specification.

2. Scope

In the specification, the variant, pre and post condition of interest properties are defined to ensure that these properties will hold in the system model. These properties are:

- Robot has unique name.
- Only robot with broadcast ability can send and receive broadcast message.
- Only robot with point-to-point ability can send and receive point-to-point message.
- Only robot with active send link can send message.
- Only robot with active receive link can receive message.
- Messages are kept into the right queue.
- Messages are kept in priority queue ordered by received time.
- If message's received time is defined, then received time is equal or greater than sent time.
- If all links are shutdown, robot cannot send or receive message.
- Robots cannot receive their own sending message.

3. References

- Software Requirement Specification, Version 1.1, Kansas State University, 2003,
- Architecture Design, Version 1.0, Kansas State University, 2003,
- USE manual, University of Bremen (http:// www.db.informatik.uni-bremen.de/project/USE
- Warmer, Jos B., Kleppe, Anneke G., *The Object Constraint Language Precise Modeling with UML*, Addison-Wesley, 1998.

4. Formal Specification Description

This section explains the Communication Model component specification based on Class diagram presented in Architecture Design Version 1.0

4.1 Classes

4.1.1 CommunicationsSystem class

There is one attribute in this class, which is isLinkEnabled, used to keep status of the system link. RegisterRobot is the operation used for registering a robot to the communication model and set type of communication that will be allowed for this robot. This operation has two parameters, n-robot name and c-communication type. There are three possible values of communication type, 1 – broadcast communication, 2 – point-to-point communication and 3 – both.

```
class CommunicationsSystem
attributes
  delay : Integer
  range : Integer
  deliveryProb : Integer
  isLinkEnabled : Boolean
operations
  registerRobot(n:String,c:Integer)
  getMessage(n:String,timeStep:Integer):Set(Message)
  sendMessage(msg:Message,timeStep:Integer)
  processBroadcast()
  processP2P()
  distributeMessage()
  getRobotCommRecord():RobotCommRecord
  setDelay(delay:Integer)
  setRange(range:Integer)
  setDeliveryProb(prob:Integer)
  startupAllLink()
  shutdownAllLink()
  setRobotDelay(sender:String,receiver:String,delay:Integer)
  setRobotRange(name:String,range:Integer)
  setRobotDeliveryProb(sender:String,receiver:String,prob:Integer)
  startupSendLink(name:String)
  startupReceiveLink(name:String)
  shutdownSendLink(name:String)
  shutdownReceiveLink(name:String)
  isLinkEnabled():Boolean
end
```

4.1.2 RobotCommRecord class

The attributes of this class are name, range, isSendLinkEnabled, isReceivedLinkEnabled, isBroadcastEnabled and isP2PEnabled. The attribute name is used for storing name of robot. Range is used for keeping the maximum sending range. The attribute isSendLinkEnabled is used for storing the status of the outgoing link of the robot. The attribute isReceiveLinkEnabled is used for storing the status of the incoming link of the robot. The attribute isBroadcastEnabled is used for storing if broadcast communication is allowed for this robot. Finally, the attribute isP2PEnabled is used for storing if point-to-point communication is allowed for this robot.

```
class RobotCommRecord
attributes
  name : String
  range : Integer
  isSendLinkEnabled : Boolean
  isReceiveLinkEnabled : Boolean
  isBroadcastEnabled : Boolean
  isP2PEnabled : Boolean
operations
  getMessage(timeStep:Integer):Set(Message)
  startupSendLink()
  startupReceiveLink()
  shutdownSendLink()
  shutdownReceiveLink()
  enableBroadcast()
  enableP2P()
  disableBroadcast()
  disableP2P()
  isSendLinkEnabled():Boolean
  isReceiveLinkEnabled():Boolean
  isBroadcastEnabled():Boolean
  isP2PEnabled():Boolean
  isSendBroadcastEnabled():Boolean
  isReceiveBroadcastEnabled():Boolean
  isSendP2PEnabled():Boolean
  isReceiveP2PEnabled():Boolean
  getRobotParameter():RobotParameter
  getDelay(name:String):Integer
  getRange():Integer
  getDeliveryProb(name:String):Integer
  addMsgToQueue(msg:Message):Boolean
  setCommType(commType:Integer)
end
```

4.1.3 RobotParameter class

This class is used to keep the parameter for each pair of robot such as delay time, delivery probability.

```
class RobotParameter
attributes
  receiverName : String
  delayTime : Integer
  deliverProb : Integer
operations
  getReceiveName():String
  getDelay():Integer
  getDeliveryProb():Integer
  setReceiveName(name:String)
  setDelay(delay:Integer)
  setDeliveryProb(prob:Integer)
end
```

4.1.4 PriorityQueue class

This class is used to keep outgoing messages for each robot.

```
class PriorityQueue
attributes
operations
  add(msg:Message):Boolean
  get(index:Integer):Message
  remove(index:Integer):Message
  isEmpty():Boolean
end
```

4.1.5 Message class

This class is the format of the message that will be used to communicate with other robots. The attributes are sender, receiver, receivedTime, sentTime and content. The receivedTime is the time, which the message will be sent out of the system. It is also the time in which the receiver will get this message. The sentTime is the time, which the sender sent the message. It is also the time that the communication system gets this message.

```
class Message
attributes
  sender : String
  receiver : String
  content : MessageContent
  receivedTime : Integer
  sentTime : Integer
operations
  setSender(name:String)
  setReceiver(name:String)
  setContent(content:MessageContent)
  setReceivedTime(time:Integer)
  setSentTime(time:Integer)
  getSender():String
  getReceiver():String
  getContent():MessageContent
  getReceivedTime():Integer
  getSentTime():Integer
  isBroadcastMessage():Boolean
  isP2PMessage():Boolean
end
```

4.2 Associations

```
association robot between
  CommunicationsSystem[1] role belongTo
  RobotCommRecord[*] role robots
end


association parameter between
  RobotCommRecord[1] role parameterOwner
  RobotParameter[*] role hasParameters
end


association queue between
```

```
  RobotCommRecord[1] role queueOwner
  PriorityQueue[1] role hasQueue
end


association message between
  PriorityQueue[*] role inQueue
  Message[*] role hasMessages ordered
end
```

4.3 Invariants

4.3.1 Robot has unique name.

```
context RobotCommRecord
  inv UniqueName:
      RobotCommRecord.allInstances->forAll(p1,p2| p1 <> p2
      implies p1.name <> p2.name)
```

4.3.2 Only robot with broadcast ability can send and receive broadcast message.

```
context RobotCommRecord
  inv BroadcastAbility1:
      RobotCommRecord.allInstances.hasQueue.hasMessages
        ->select(receiver='broadcast' and sender=self.name)->notEmpty
        implies self.isBroadcastEnabled = true


context r:RobotCommRecord
  inv BroadcastAbility2:
      r.hasQueue.hasMessages
        ->select(receiver='broadcast')->notEmpty
        implies r.isBroadcastEnabled = true
```

4.3.3 Only robot with point-to-point ability can send and receive point-to-point message.

```
context RobotCommRecord
  inv P2PAbility1:
      RobotCommRecord.allInstances.hasQueue.hasMessages
        ->select(receiver <> 'broadcast' and sender=self.name)
        ->notEmpty implies self.isP2PEnabled = true

context r:RobotCommRecord
  inv P2PAbility2:
      r.hasQueue.hasMessages
        ->select(receiver <> 'broadcast')->notEmpty
        implies r.isP2PEnabled = true
```

4.3.4 Only robot with active send link can send message.

```
context RobotCommRecord
  inv sendAbility:
      RobotCommRecord.allInstances.hasQueue.hasMessages
```

```
        ->select(sender=self.name)->notEmpty
        implies self.isSendLinkEnabled = true
```

4.3.5 Only robot with active receive link can receive message.

```
context r:RobotCommRecord
  inv receiveAbility:
      r.hasQueue.hasMessages->notEmpty implies
      r.isReceiveLinkEnabled = true
```

4.3.6 Messages are kept into the right queue.

```
context RobotCommRecord
   inv rightQueue:
       hasQueue.hasMessages->forAll((receiver=self.name) or
(receiver='broadcast'))
```

4.3.7 Messages are kept in priority queue ordered by received time.

```
context p:PriorityQueue
  inv priorityQueue:
      Sequence{1..(p.hasMessages->size-1)}
      ->forAll(i | p.hasMessages->at(i).receivedTime
      <= p.hasMessages->at(i+1).receivedTime)
```

4.3.8 If message's received time is defined, then received time is equal or greater than sent time.

```
context m:Message
  inv rightTime:
      m.receivedTime.isDefined implies m.receivedTime >= m.sentTime
```

4.3.9 If all links are shutdown, robot cannot send or receive message.

```
context c:CommunicationsSystem
  inv allLinkShutdown:
      RobotCommRecord.allInstances.hasQueue.hasMessages->notEmpty
       implies c.isLinkEnabled = true
```

4.3.10 Robots cannot receive their own sending message.

```
context r:RobotCommRecord
  inv sendToYourself:
      r.hasQueue.hasMessages->forAll(sender <> r.name)
```

4.4 Operations

4.4.1 Register Robot

```
context CommunicationsSystem::registerRobot(n:String,c:Integer)
  pre precond_1:    n.isDefined
  pre precond_2:    c.isDefined
  pre precond_3:    (c=1 or c=2 or c=3)
  pre precond_4:    robots->select(name=n)->isEmpty
  post postcond_1:  robots->exists(r | r.oclIsNew and r.name = n)
  post postcond_2:  robots=robots@pre->union(robots->select(name=n))
  post postcond_3:  robots->select(name=n)->size = 1
  post postcond_4:  (c=1)implies
                    robots->select(name=n and
                    isBroadcastEnabled=true and
                    isP2PEnabled=false)->notEmpty
  post postcond_5:  (c=2) implies
                    robots->select(name=n and
                    isP2PEnabled=true and
                    isBroadcastEnabled=false)->notEmpty
  post postcond_6:  (c=3) implies
                    robots->select(name=n and
                    isBroadcastEnabled=true and
                    isP2PEnabled=true)->notEmpty
```

This part of specification defines pre and post condition of register robot operation. It takes two arguments, n – robot name and c – communication type. The precond_1 states that robot name is defined. The precond_2 states that communication type is defined. The precond_3 states that communication type can be 1, 2 or 3 only. The last pre condition is there is no RobotCommRecord named n in the system before registering this robot. There are six post conditions. The first post condition is RobotCommRecord is new created and named “n”. The second post condition says that the new set of RobotCommRecord is the previous set plus the new RobotCommRecord, which is just created and named “n”. The third post condition states that there is only one RobotCommRecord named “n”. The last three post conditions depend on type of communication. If communication type is 1 then RobotCommRecord, which is named “n”, has isBroadcastEnabled attribute sets to true while isP2PEnabled attribute sets to false. If communication type is 2 then RobotCommRecord, which is named “n”, has isBroadcastEnabled attribute sets to false while isP2PEnabled attribute sets to true. Finally, setting robot to have both broadcast and point-to-point capability, which is if communication type is 3 then RobotCommRecord, which is named “n”, has both isBroadcastEnabled attribute and isP2PEnabled attribute set to true

4.4.2 Send Message operation

```
context CommunicationsSystem::sendMessage(msg:Message,timeStep:Integer)
  pre precond_1:    timeStep > 0 and timeStep.isDefined
  pre precond_2:    msg.isDefined
  pre precond_3:    msg.sender.isDefined
  pre precond_4:    msg.receiver.isDefined
  pre precond_5:    isLinkEnabled = true
```

```
pre precond_6:    robots->select(name=msg.sender)
                   ->forAll(isSendLinkEnabled = true)

pre precond_7:    msg.receiver = 'broadcast'
                   implies robots->select(name=msg.sender)
                   ->forAll(isBroadcastEnabled = true)

pre precond_8:    msg.receiver <> 'broadcast'
                     implies robots->select(name=msg.sender)->
                     forAll(isP2PEnabled = true)

post postcond_1:  msg.sentTime = timeStep

post postcond 2:  msg.receiver <> 'broadcast' implies
                     robots->select(name=msg.sender)->
                     forAll(hasParameters->
                     forAll(receiverName=msg.receiver implies
                     msg.receivedTime = parameterOwner.belongTo.delay
                     + timeStep + delayTime))

post postcond 3:  msg.receiver = 'broadcast' implies
                     robots->select(name=msg.sender)->
                     forAll(hasParameters->
                     forAll(msg.receivedTime =
                     parameterOwner.belongTo.delay
                     + timeStep + delayTime))

post postcond_4:  msg.receiver = 'broadcast' implies
                     robots->select(name <> msg.sender
                     and isReceiveLinkEnabled = true
                     and isBroadcastEnabled = true)
                     ->forAll(r| r.hasQueue.hasMessages->asSet
                     = r.hasQueue.hasMessages@pre
                     ->including(msg)->asSet)

post postcond_5:  msg.receiver <> 'broadcast' implies
                     robots->select(name = msg.receiver
                     and isReceiveLinkEnabled = true
                     and isP2PEnabled = true)
                     ->forAll(r| r.hasQueue.hasMessages->asSet
                     = r.hasQueue.hasMessages@pre
                     ->including(msg)->asSet)
```

This part of specification defines pre and post condition for sendMessage operation. The pre conditions are time step is greater than 0, time step is defined, message is defined, the system link is active, sender's send link is active. Furthermore, if the message is broadcast then sender must have broadcast capability; otherwise sender must have point-to-point capability. The post conditions are message's sentTime and receivedTime are set; the message will be included in the receiver's priority queue if the receiver is qualified. The process of qualifying receiver is check if receiver's receive link is active and if the message is broadcast then the receiver must have broadcast capability; otherwise the receiver must have point-to-point capability

4.4.3 Get Message operation

context CommmunicationsSystem::getMessage(n:String,timeStep:Integer)
:Set(Message)

```
  pre precond_1:    n.isDefined
  pre precond_2:    robots.exists(r| r.name=n)
  pre precond_3:    timeStep.isDefined
  pre precond_4:    timeStep > 0
  post postcond_1:  robots->select(name = n)
                     ->forAll(r | r.hasQueue.hasMessages->asSet
                     = r.hasQueue.hasMessages@pre->asSet
                     - r.hasQueue.hasMessages@pre
                     ->select(receivedTime = timeStep)->asSet)

  post postcond_2:  robots->select(name=n)
                     ->forAll(r| result = r.hasQueue.hasMessages@pre
                     ->select(receivedTime = timeStep)->asSet)
```

This part is the specification for getMessage operation. The post conditions are n (robor name) is defined, there exists robot named n, time step is defined and greater than 0. The post conditions are priority queue of robot n excludes messages which receivedTime is equal to time step and the operation will return the result which is a set of messages which receivedTime is equal to time step.

4.5 USE Test Script

All test scripts are used counter example to test the correctness of the specification.

4.5.1 Robots Robot has unique name.

Scenario: The test script has two robots named "A"

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true

!create robotA:RobotCommRecord
!create robotB:RobotCommRecord
!set robotA.name = 'A'
!set robotB.name = 'A'

!insert (comm1,robotA) into robot
!insert (comm1,robotB) into robot

!create queue1:PriorityQueue
!create queue2:PriorityQueue
!insert (robotA,queue1) into queue
!insert (robotB,queue2) into queue
```

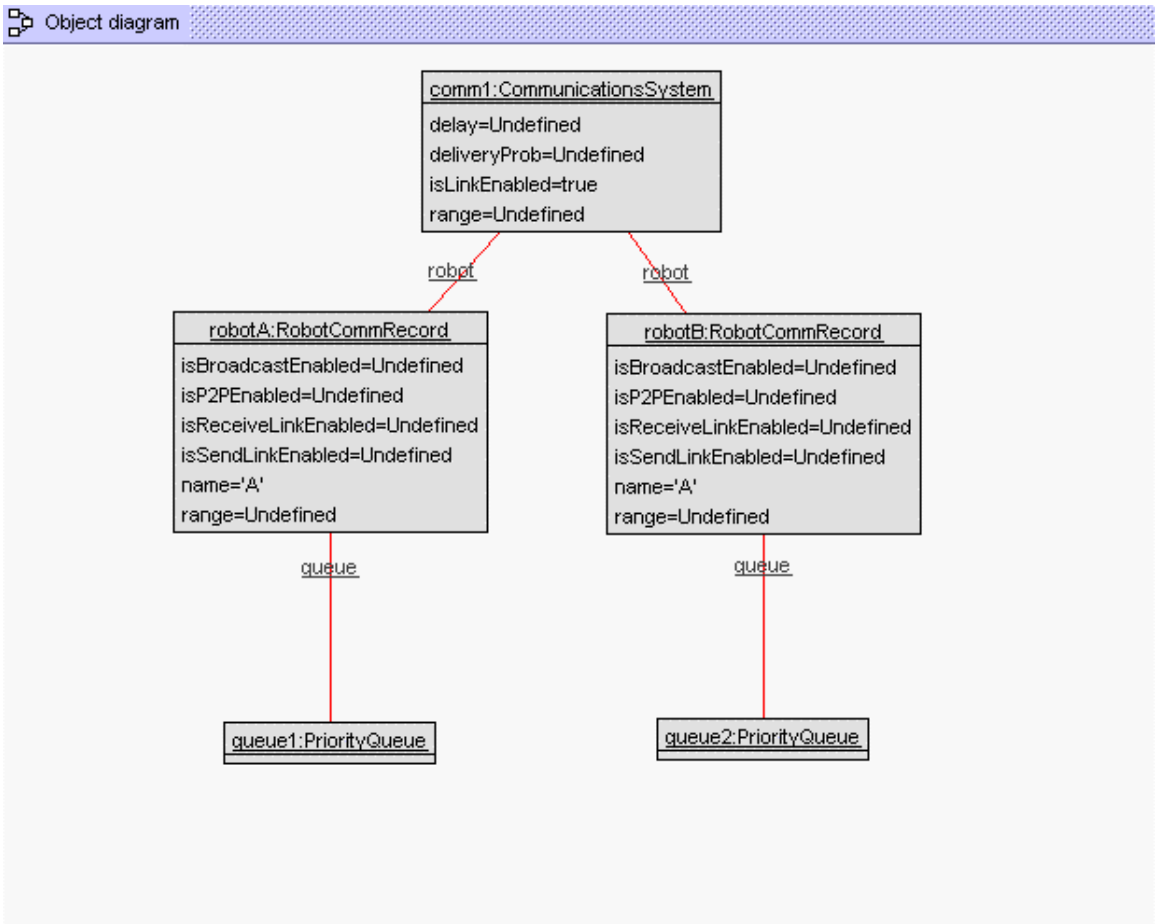

Figure1. USE Object Diagram – UniqueName Constraint

4.5.2 Only robot with broadcast ability can send and receive broadcast message.

Scenario 1: RobotB and RobotC received a broadcast message from RobotA which broadcast is disabled.

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true

!create robotA:RobotCommRecord
!create robotB:RobotCommRecord
!create robotC:RobotCommRecord

!set robotA.name = 'A'
!set robotB.name = 'B'
!set robotC.name = 'C'
!set robotA.isBroadcastEnabled = false
!set robotA.isSendLinkEnabled = true
!set robotA.isReceiveLinkEnabled = true

!set robotB.isBroadcastEnabled = true
!set robotB.isSendLinkEnabled = true
!set robotB.isReceiveLinkEnabled = true
```

```
!set robotC.isBroadcastEnabled = true
!set robotC.isSendLinkEnabled = true
!set robotC.isReceiveLinkEnabled = true

!insert (comm1,robotA) into robot
!insert (comm1,robotB) into robot
!insert (comm1,robotC) into robot

!create queueA:PriorityQueue
!create queueB:PriorityQueue
!create queueC:PriorityQueue

!insert (robotA,queueA) into queue
!insert (robotB,queueB) into queue
!insert (robotC,queueC) into queue

!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'broadcast'

!insert (queueB,msg1) into message
!insert (queueC,msg1) into message
```

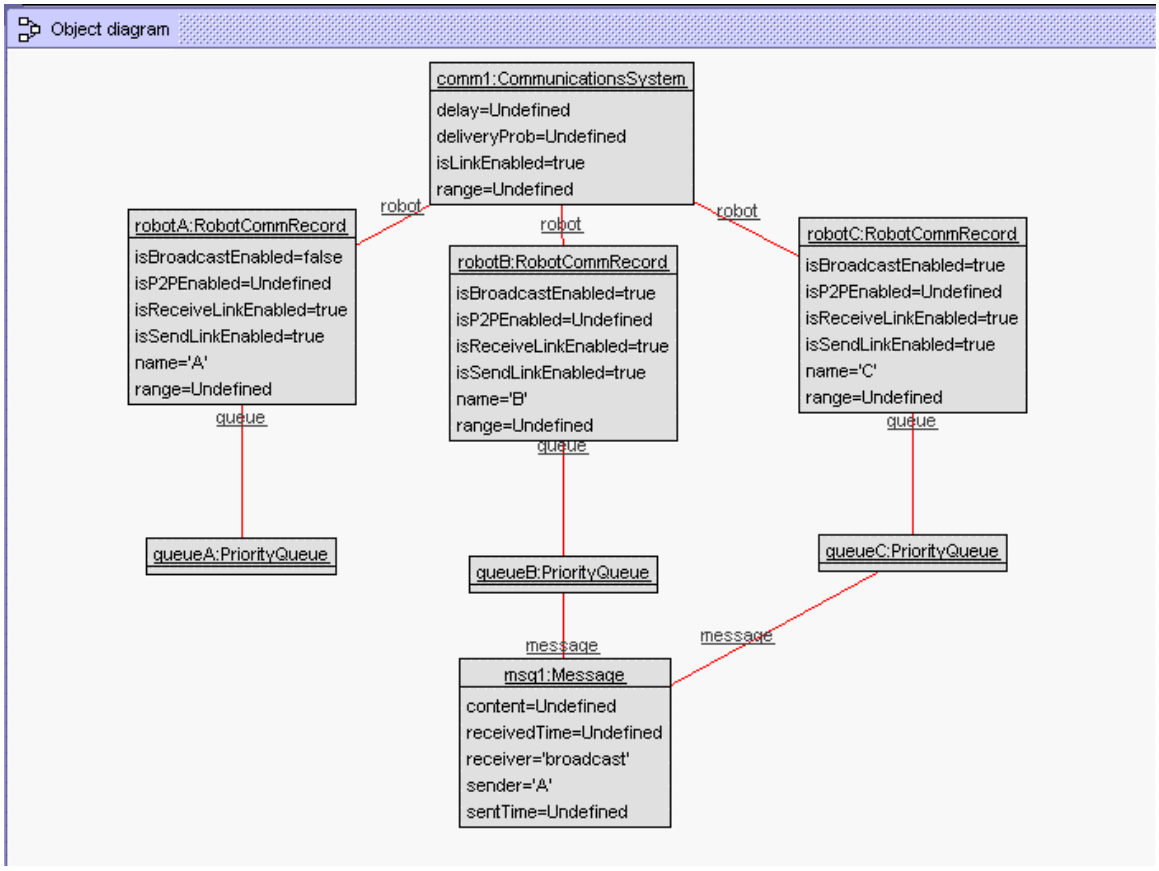

Figure2. USE Object Diagram – BroadcastAbility1 Constraint

Scenario2: RobotB with disabled broadcast ability received a broadcast message from RobotA.

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true

!create robotA:RobotCommRecord
!create robotB:RobotCommRecord
!create robotC:RobotCommRecord

!set robotA.name = 'A'
!set robotB.name = 'B'
!set robotC.name = 'C'
!set robotA.isBroadcastEnabled = true
!set robotA.isSendLinkEnabled = true
!set robotA.isReceiveLinkEnabled = true

!set robotB.isBroadcastEnabled = false
!set robotB.isSendLinkEnabled = true
!set robotB.isReceiveLinkEnabled = true

!set robotC.isBroadcastEnabled = true
!set robotC.isSendLinkEnabled = true
!set robotC.isReceiveLinkEnabled = true

!insert (comm1,robotA) into robot
!insert (comm1,robotB) into robot
!insert (comm1,robotC) into robot

!create queueA:PriorityQueue
!create queueB:PriorityQueue
!create queueC:PriorityQueue

!insert (robotA,queueA) into queue
!insert (robotB,queueB) into queue
!insert (robotC,queueC) into queue

!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'broadcast'

!insert (queueB,msg1) into message
!insert (queueC,msg1) into message
```

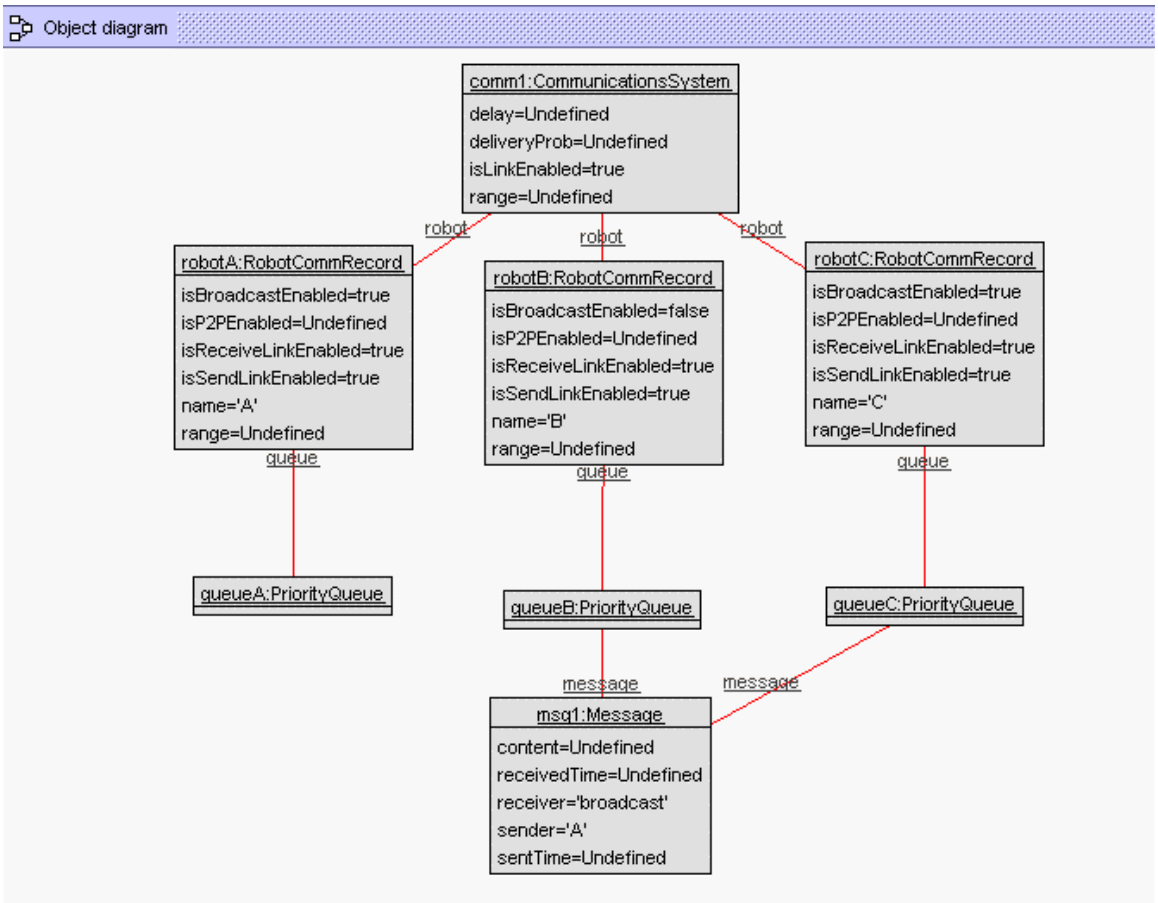

Figure3. USE Object Diagram - BroadcastAbility2 Constraint

4.5.3 Only robot with point-to-point ability can send and receive point-to-point message.

Scenario 1: RobotB received a message from RobotA which point-to-point is disabled.

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true

!create robotA:RobotCommRecord
!create robotB:RobotCommRecord

!set robotA.name = 'A'
!set robotB.name = 'B'
!set robotA.isBroadcastEnabled = true
!set robotA.isP2PEnabled = false
!set robotA.isSendLinkEnabled = true
!set robotA.isReceiveLinkEnabled = true

!set robotB.isBroadcastEnabled = false
!set robotB.isP2PEnabled = true
!set robotB.isSendLinkEnabled = true
!set robotB.isReceiveLinkEnabled = true

!insert (comm1,robotA) into robot
!insert (comm1,robotB) into robot

!create queueA:PriorityQueue
!create queueB:PriorityQueue

!insert (robotA,queueA) into queue
!insert (robotB,queueB) into queue

!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'B'

!insert (queueB,msg1) into message
```

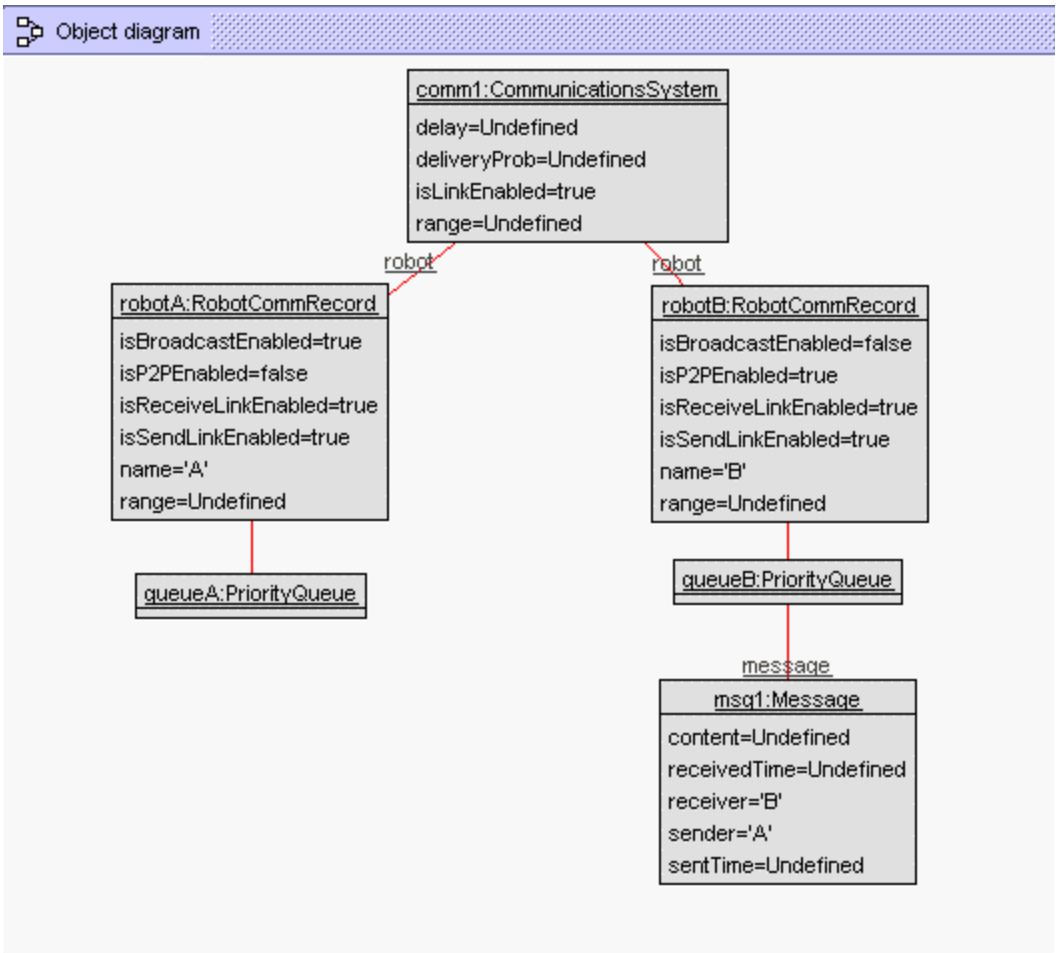

Figure4. USE Object Diagram - P2PAbility1 Constraint

Scenario 2: RobotB with disabled point-to-point ability received a message from RobotA.

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true

!create robotA:RobotCommRecord
!create robotB:RobotCommRecord

!set robotA.name = 'A'
!set robotB.name = 'B'
!set robotA.isBroadcastEnabled = true
!set robotA.isP2PEnabled = true
!set robotA.isSendLinkEnabled = true
!set robotA.isReceiveLinkEnabled = true

!set robotB.isBroadcastEnabled = false
!set robotB.isP2PEnabled = false
!set robotB.isSendLinkEnabled = true
!set robotB.isReceiveLinkEnabled = true
```

```
!insert (comm1,robotA) into robot
!insert (comm1,robotB) into robot

!create queueA:PriorityQueue
!create queueB:PriorityQueue

!insert (robotA,queueA) into queue
!insert (robotB,queueB) into queue

!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'B'

!insert (queueB,msg1) into message
```

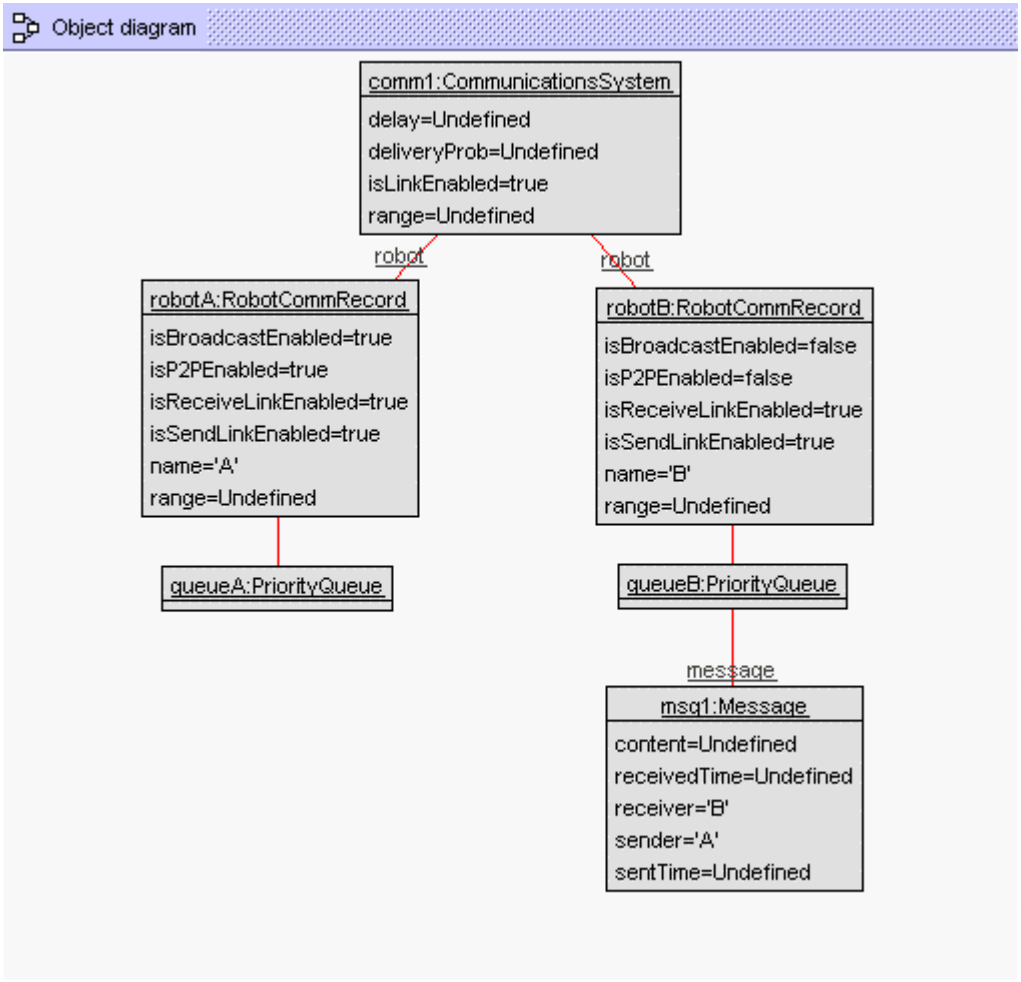

Figure5. USE Object Diagram - P2Pability2 Constraint

4.5.4 Only robot with active send link can send message.

Scenario: RobotB received a message from RobotA which outgoing link is disabled

```
!create comm1:CommunicationsSystem
```

```
!set comm1.isLinkEnabled = true

!create robotA:RobotCommRecord
!create robotB:RobotCommRecord

!set robotA.name = 'A'
!set robotB.name = 'B'
!set robotA.isBroadcastEnabled = true
!set robotA.isP2PEnabled = true
!set robotA.isSendLinkEnabled = false
!set robotA.isReceiveLinkEnabled = true

!set robotB.isBroadcastEnabled = false
!set robotB.isP2PEnabled = true
!set robotB.isSendLinkEnabled = true
!set robotB.isReceiveLinkEnabled = true
!insert (comm1,robotA) into robot
!insert (comm1,robotB) into robot

!create queueA:PriorityQueue
!create queueB:PriorityQueue
!insert (robotA,queueA) into queue
!insert (robotB,queueB) into queue

!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'B'

!insert (queueB,msg1) into message
```

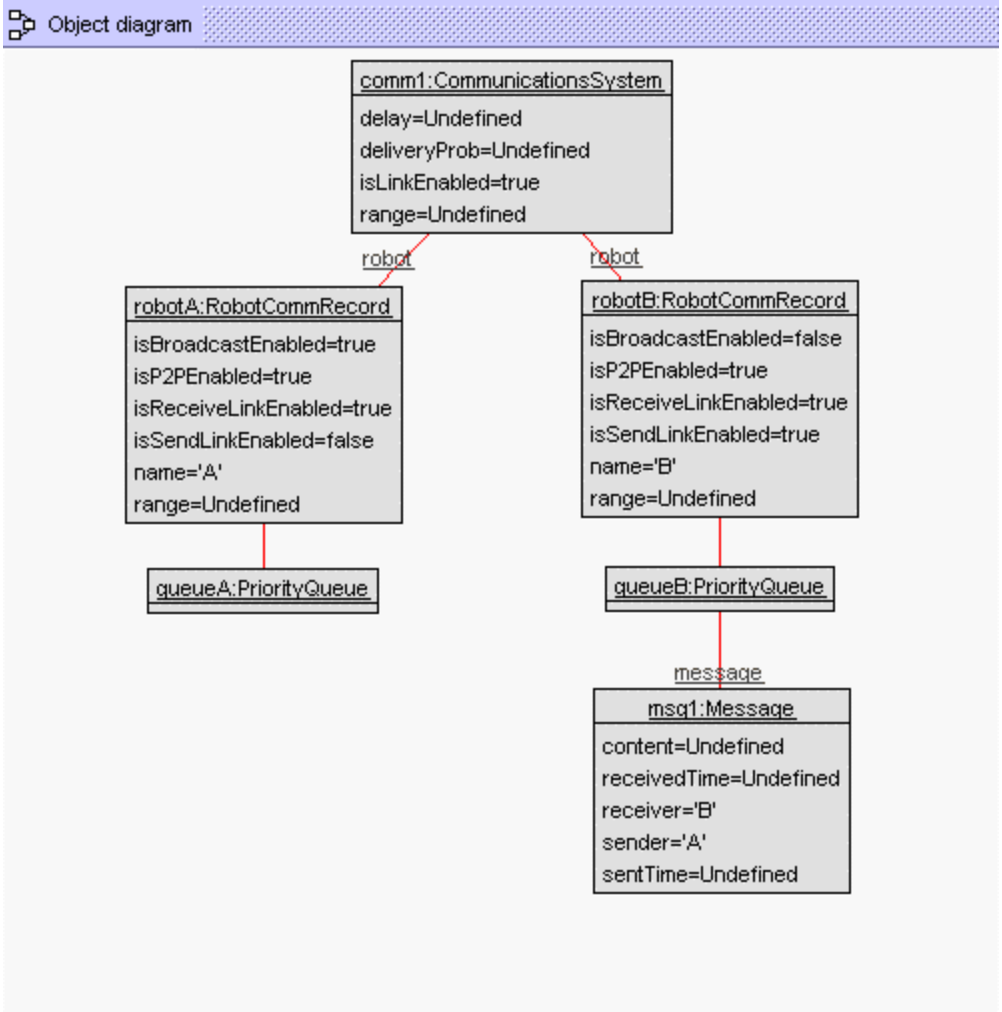

Figure6. USE Object Diagram - SendAbility Constraint

4.5.5 Only robot with active receive link can receive message.

Scenario: RobotB with disabled incoming link received a message from RobotA,

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true

!create robotA:RobotCommRecord
!create robotB:RobotCommRecord

!set robotA.name = 'A'
!set robotB.name = 'B'
!set robotA.isBroadcastEnabled = true
!set robotA.isP2PEnabled = true
!set robotA.isSendLinkEnabled = true
!set robotA.isReceiveLinkEnabled = true

!set robotB.isBroadcastEnabled = true
!set robotB.isP2PEnabled = true
```

```
!set robotB.isSendLinkEnabled = true
!set robotB.isReceiveLinkEnabled = false

!insert (comm1,robotA) into robot
!insert (comm1,robotB) into robot

!create queueA:PriorityQueue
!create queueB:PriorityQueue

!insert (robotA,queueA) into queue
!insert (robotB,queueB) into queue

!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'B'

!insert (queueB,msg1) into message
```

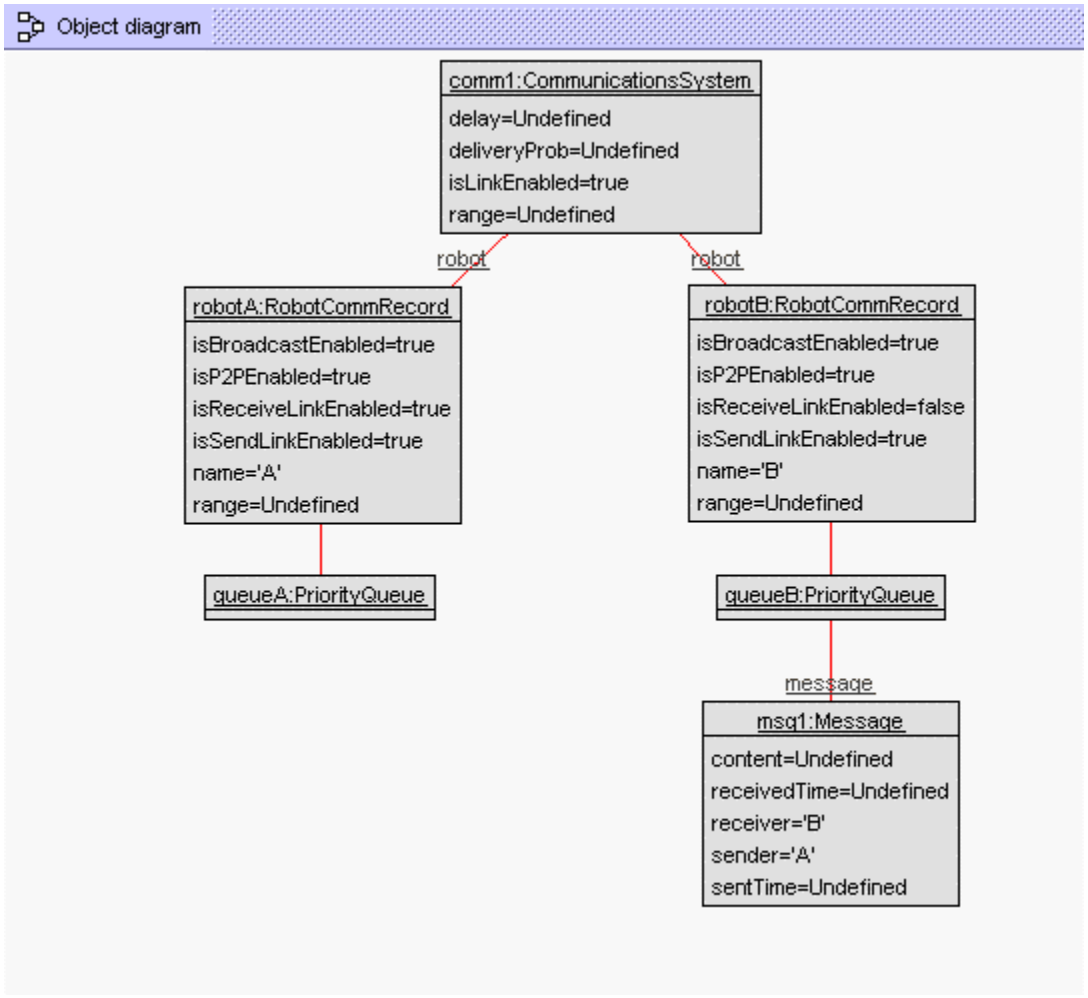

Figure7. USE Object Diagram – ReceiveAbility Constraint

4.5.6 Messages are kept into the right queue.

Scenario: RobotC's priorityQueue has a message which belongs to A.

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true

!create robotA:RobotCommRecord
!create robotB:RobotCommRecord
!create robotC:RobotCommRecord

!set robotA.name = 'A'
!set robotB.name = 'B'
!set robotC.name = 'C'
!set robotA.isBroadcastEnabled = true
!set robotA.isP2PEnabled = true
!set robotA.isSendLinkEnabled = true
!set robotA.isReceiveLinkEnabled = true

!set robotB.isBroadcastEnabled = true
!set robotB.isP2PEnabled = true
!set robotB.isSendLinkEnabled = true
!set robotB.isReceiveLinkEnabled = true

!set robotC.isBroadcastEnabled = true
!set robotC.isP2PEnabled = true
!set robotC.isSendLinkEnabled = true
!set robotC.isReceiveLinkEnabled = true

!insert (comm1,robotA) into robot
!insert (comm1,robotB) into robot
!insert (comm1,robotC) into robot

!create queueA:PriorityQueue
!create queueB:PriorityQueue
!create queueC:PriorityQueue

!insert (robotA,queueA) into queue
!insert (robotB,queueB) into queue
!insert (robotC,queueC) into queue

!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'B'

!insert (queueC,msg1) into message
```

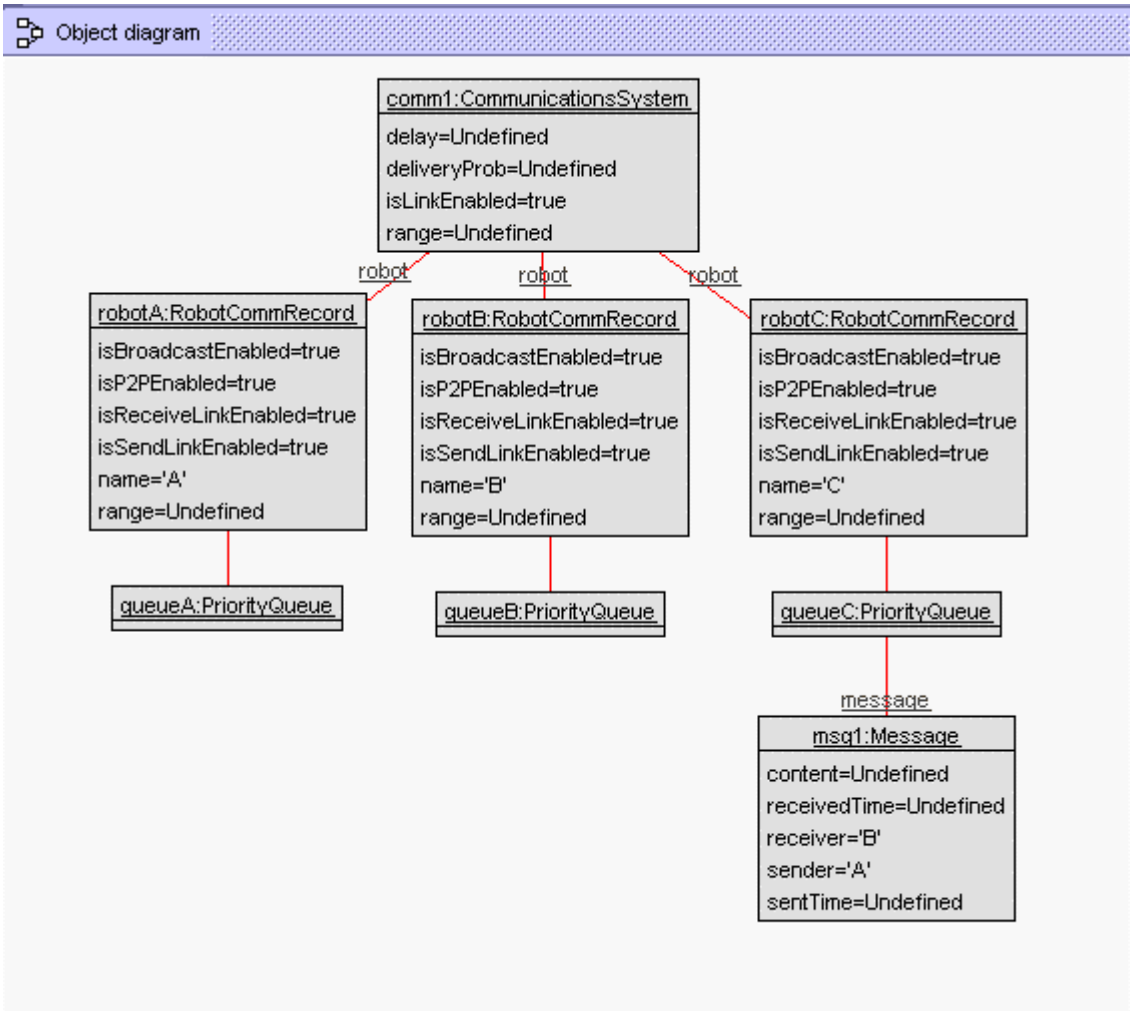

Figure8. USE Object Diagram - RightQueue Constraint

4.5.7 Messages are kept in priority queue ordered by received time.

Scenario: QueueB has msg2, which received time is 2,located before msg1, which received time is 1

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true

!create robotA:RobotCommRecord
!create robotB:RobotCommRecord
!create robotC:RobotCommRecord

!set robotA.name = 'A'
!set robotB.name = 'B'
!set robotC.name = 'C'
!set robotA.isBroadcastEnabled = true
!set robotA.isP2PEnabled = true
!set robotA.isSendLinkEnabled = true
!set robotA.isReceiveLinkEnabled = true
```

```
!set robotB.isBroadcastEnabled = true
!set robotB.isP2PEnabled = true
!set robotB.isSendLinkEnabled = true
!set robotB.isReceiveLinkEnabled = true

!set robotC.isBroadcastEnabled = true
!set robotC.isP2PEnabled = true
!set robotC.isSendLinkEnabled = true
!set robotC.isReceiveLinkEnabled = true

!insert (comm1,robotA) into robot
!insert (comm1,robotB) into robot
!insert (comm1,robotC) into robot

!create queueA:PriorityQueue
!create queueB:PriorityQueue
!create queueC:PriorityQueue

!insert (robotA,queueA) into queue
!insert (robotB,queueB) into queue
!insert (robotC,queueC) into queue

!create msg1:Message
!create msg2:Message

!set msg1.sender = 'A'
!set msg1.receiver = 'B'
!set msg1.sentTime = 1
!set msg1.receivedTime = 1
!set msg2.sender = 'C'
!set msg2.receiver = 'B'
!set msg2.sentTime = 2
!set msg2.receivedTime = 2

!insert (queueB,msg2) into message
!insert (queueB,msg1) into message
```

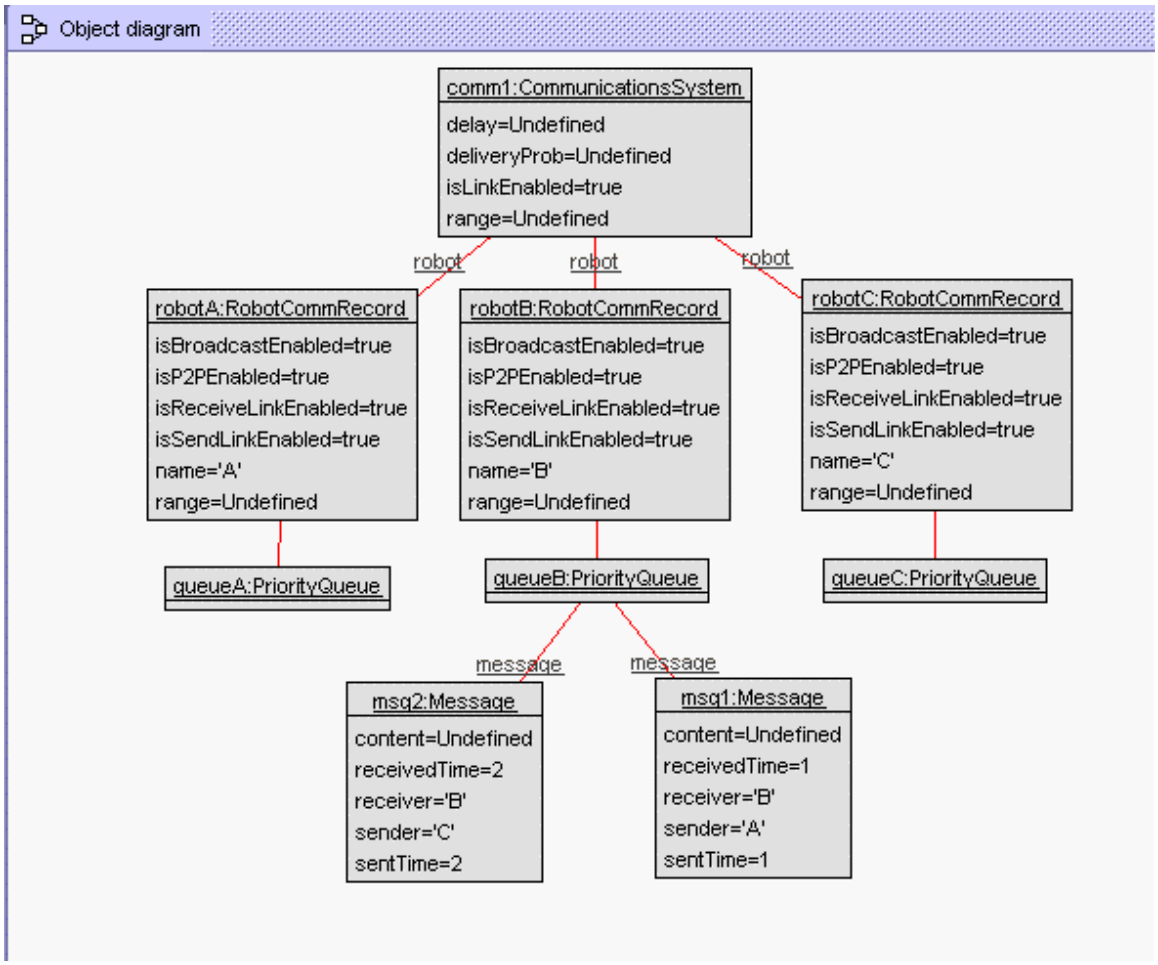

Figure9. USE Object Diagram – PriotityQueue Constraint

4.5.8 If message's received time is defined, then received time is equal or greater than sent time.

Scenario: msg1 has received time equals to 1 which is less then sent time which is 2

```
!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'broadcast'
!set msg1.sentTime = 2
!set msg1.receivedTime = 1
```

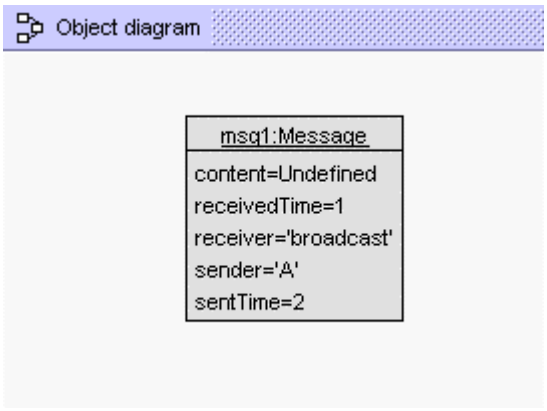

Figure10. USE Object Diagram – RightTime Constraint

4.5.9 If all links are shutdown, robot cannot send or receive message.

Scenario: All links in the system is shutdown, but RobotB received a message.

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = false

!create robotA:RobotCommRecord
!create robotB:RobotCommRecord

!set robotA.name = 'A'
!set robotB.name = 'B'
!set robotA.isBroadcastEnabled = true
!set robotA.isP2PEnabled = true
!set robotA.isSendLinkEnabled = true
!set robotA.isReceiveLinkEnabled = true

!set robotB.isBroadcastEnabled = true
!set robotB.isP2PEnabled = true
!set robotB.isSendLinkEnabled = true
!set robotB.isReceiveLinkEnabled = true

!insert (comm1,robotA) into robot
!insert (comm1,robotB) into robot

!create queueA:PriorityQueue
!create queueB:PriorityQueue

!insert (robotA,queueA) into queue
!insert (robotB,queueB) into queue

!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'B'

!insert (queueB,msg1) into message
```

Figure11. USE Object Diagram – AllLinkShutdown Constraint

4.5.10 Robots cannot receive their own sending message.
Scenario: RobotA received a message, which is sent by him.

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true

!create robotA:RobotCommRecord
!create robotB:RobotCommRecord

!set robotA.name = 'A'
!set robotB.name = 'B'
!set robotA.isBroadcastEnabled = true
!set robotA.isP2PEnabled = true
!set robotA.isSendLinkEnabled = true
!set robotA.isReceiveLinkEnabled = true

!set robotB.isBroadcastEnabled = true
!set robotB.isP2PEnabled = true
!set robotB.isSendLinkEnabled = true
```

```
!set robotB.isReceiveLinkEnabled = true

!insert (comm1,robotA) into robot
!insert (comm1,robotB) into robot

!create queueA:PriorityQueue
!create queueB:PriorityQueue

!insert (robotA,queueA) into queue
!insert (robotB,queueB) into queue

!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'broadcast'

!insert (queueA,msg1) into message
```

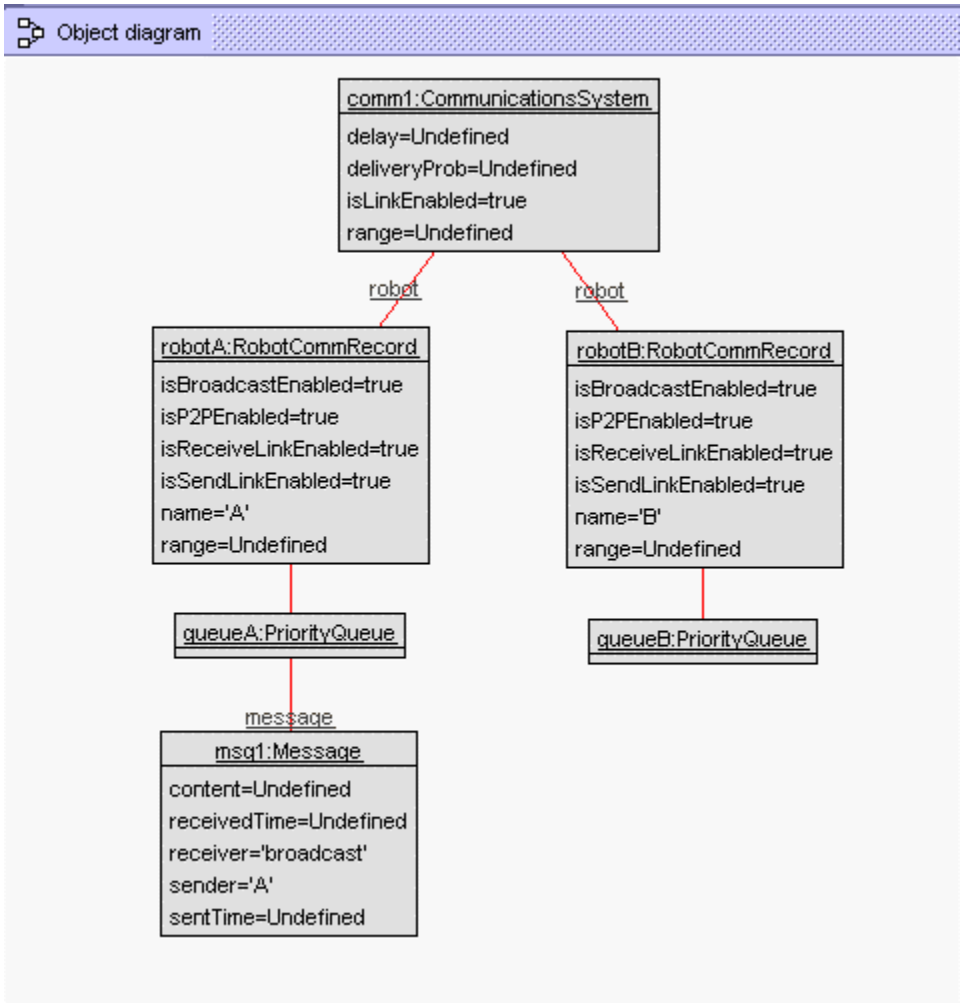

Figure12. USE Object Diagram –SendToYourSelf Constraint

4.5.11 registerRobot Operation

Scenario: The script adds new Robot named A which already exists in the system. This script violates post condition 1,2,3 and 5.

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true
!create RobotA:RobotCommRecord
!set RobotA.name ='A'
!insert (comm1,RobotA) into robot
!create queueA:PriorityQueue
!insert (RobotA,queueA) into queue

!openter comm1 registerRobot('A',2)

!create RobotB:RobotCommRecord
!set RobotB.name = 'A'
!set RobotB.isP2PEnabled = true
!set RobotB.isBroadcastEnabled = false
!insert (comm1,RobotB) into robot
!create queueB:PriorityQueue
!insert (RobotB,queueB) into queue

!opexit
```

4.5.12 sendMessage operation

Scenario: There is no message added to RobotB's priorityQueue after message is sent. This script violates post condition 5.

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true
!set comm1.delay = 2
!create RobotA:RobotCommRecord
!set RobotA.name ='A'
!set RobotA.isP2PEnabled = true
!set RobotA.isBroadcastEnabled = true
!set RobotA.isSendLinkEnabled = true
!set RobotA.isReceiveLinkEnabled = true
!insert (comm1,RobotA) into robot
!create queueA:PriorityQueue
!insert (RobotA,queueA) into queue
!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'B'

!create Param1:RobotParameter
!set Param1.receiverName = 'B'
!set Param1.delayTime = 1
!insert (RobotA,Param1) into parameter

!create RobotB:RobotCommRecord
!set RobotB.name = 'B'
!set RobotB.isP2PEnabled = true
!set RobotB.isBroadcastEnabled = false
!set RobotB.isSendLinkEnabled = true
!set RobotB.isReceiveLinkEnabled = true
!insert (comm1,RobotB) into robot
!create queueB:PriorityQueue
!insert (RobotB,queueB) into queue

!create Param2:RobotParameter
!set Param2.receiverName = 'A'
!set Param2.delayTime = 1
!insert (RobotB,Param2) into parameter

!openter comm1 sendMessage(msg1,1)
!set msg1.sentTime = 1
!set msg1.receivedTime = 4

!opexit
```

4.5.13 getMessage Operation

Scenario1: There is no message removed from RobotB's priorityQueue after calling getMessage operation. (It is supposed to remove the messages which received time is equal to current time step. For this case, message msg1 and msg2 should be removed and the result should return msg1 and msg2). This script violates post condition 1.

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true
!create RobotA:RobotCommRecord
!set RobotA.name ='A'
!set RobotA.isP2PEnabled = true
!set RobotA.isBroadcastEnabled = true
!set RobotA.isSendLinkEnabled = true
!set RobotA.isReceiveLinkEnabled = true
!insert (comm1,RobotA) into robot
!create queueA:PriorityQueue
!insert (RobotA,queueA) into queue
!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'B'

!create RobotB:RobotCommRecord
!set RobotB.name = 'B'
!set RobotB.isP2PEnabled = true
!set RobotB.isBroadcastEnabled = true
!set RobotB.isSendLinkEnabled = true
!set RobotB.isReceiveLinkEnabled = true
!insert (comm1,RobotB) into robot
!create queueB:PriorityQueue
!insert (RobotB,queueB) into queue
!set msg1.sentTime = 1
!set msg1.receivedTime = 1

!create msg2:Message
!set msg2.sender = 'A'
!set msg2.receiver = 'B'
!set msg2.sentTime = 1
!set msg2.receivedTime = 1
!insert (queueB,msg2) into message
!insert (queueB,msg1) into message

!openter comm1 getMessage('B',1)
!opexit Set{msg1,msg2}
```

Scenario2: There is only one message which received time is 1 in RobotB's queue, but the result returns two messages, msg1 and msg2. This script violates post condition 2.

```
!create comm1:CommunicationsSystem
!set comm1.isLinkEnabled = true
!create RobotA:RobotCommRecord
!set RobotA.name ='A'
!set RobotA.isP2PEnabled = true
!set RobotA.isBroadcastEnabled = true
!set RobotA.isSendLinkEnabled = true
!set RobotA.isReceiveLinkEnabled = true
!insert (comm1,RobotA) into robot
!create queueA:PriorityQueue
!insert (RobotA,queueA) into queue
!create msg1:Message
!set msg1.sender = 'A'
!set msg1.receiver = 'B'

!create RobotB:RobotCommRecord
!set RobotB.name = 'B'
!set RobotB.isP2PEnabled = true
!set RobotB.isBroadcastEnabled = true
!set RobotB.isSendLinkEnabled = true
!set RobotB.isReceiveLinkEnabled = true
!insert (comm1,RobotB) into robot
!create queueB:PriorityQueue
!insert (RobotB,queueB) into queue

!set msg1.sentTime = 2
!set msg1.receivedTime = 2

!create msg2:Message
!set msg2.sender = 'A'
!set msg2.receiver = 'B'
!set msg2.sentTime = 1
!set msg2.receivedTime = 1
!insert (queueB,msg2) into message
!insert (queueB,msg1) into message

!openter comm1 getMessage('B',1)
!delete (queueB,msg2) from message
!opexit Set{msg1,msg2}
```

Appendix A

```
--
-- Description: Formal Requirement Specification based on
-- Communications System for Cooperative Robotics Simulator
-- architecture design using UML/OCL methodology.
--
-- Author: Acharaporn Pattaravanichanon
-- File name: communicationSystem.use
-- Course: CIS895 MSE project
-- Computing and Information Systems Department
-- Kansas State University, Spring 2003
-- Date: March 12, 2004
--
--
--

model Communication

--
-- C L A S S E S
--

class CommunicationsSystem
attributes
  delay : Integer
  range : Integer
  deliveryProb : Integer
  isLinkEnabled : Boolean
operations
  registerRobot(n:String,c:Integer)
  getMessage(n:String,timeStep:Integer):Set(Message)
  sendMessage(msg:Message,timeStep:Integer)
  processBroadcast()
  processP2P()
  distributeMessage()
  getRobotCommRecord():RobotCommRecord
  setDelay(delay:Integer)
  setRange(range:Integer)
  setDeliveryProb(prob:Integer)
  startupAllLink()
  shutdownAllLink()
  setRobotDelay(sender:String,receiver:String,delay:Integer)
  setRobotRange(name:String,range:Integer)
  setRobotDeliveryProb(sender:String,receiver:String,prob:Integer)
  startupSendLink(name:String)
  startupReceiveLink(name:String)
  shutdownSendLink(name:String)
  shutdownReceiveLink(name:String)
  isLinkEnabled():Boolean
end

class RobotCommRecord
attributes
  name : String
  range : Integer
  isSendLinkEnabled : Boolean
```

```
  isReceiveLinkEnabled : Boolean
  isBroadcastEnabled : Boolean
  isP2PEnabled : Boolean
operations
  getMessage(timeStep:Integer):Set(Message)
  startupSendLink()
  startupReceiveLink()
  shutdownSendLink()
  shutdownReceiveLink()
  enableBroadcast()
  enableP2P()
  disableBroadcast()
  disableP2P()
  isSendLinkEnabled():Boolean
  isReceiveLinkEnabled():Boolean
  isBroadcastEnabled():Boolean
  isP2PEnabled():Boolean
  isSendBroadcastEnabled():Boolean
  isReceiveBroadcastEnabled():Boolean
  isSendP2PEnabled():Boolean
  isReceiveP2PEnabled():Boolean
  getRobotParameter():RobotParameter
  getDelay(name:String):Integer
  getRange():Integer
  getDeliveryProb(name:String):Integer
  addMsgToQueue(msg:Message):Boolean
  setCommType(commType:Integer)
end

class RobotParameter
attributes
  receiverName : String
  delayTime : Integer
  deliverProb : Integer
operations
  getReceiveName():String
  getDelay():Integer
  getDeliveryProb():Integer
  setReceiveName(name:String)
  setDelay(delay:Integer)
  setDeliveryProb(prob:Integer)
end

class PriorityQueue
attributes
operations
  add(msg:Message):Boolean
  get(index:Integer):Message
  remove(index:Integer):Message
  isEmpty():Boolean
end

class Message
attributes
  sender : String
  receiver : String
  content : OclAny
```

```
  receivedTime : Integer
  sentTime : Integer
operations
  setSender(name:String)
  setReceiver(name:String)
  setContent(content:OclAny)
  setReceivedTime(time:Integer)
  setSentTime(time:Integer)
  getSender():String
  getReceiver():String
  getContent():OclAny
  getReceivedTime():Integer
  getSentTime():Integer
  isBroadcastMessage():Boolean
  isP2PMessage():Boolean
end

--
-- A S S O C I A T I O N S
--

-- robot: a communication system consists of many robots

association robot between
  CommunicationsSystem[1] role belongTo
  RobotCommRecord[*] role robots
end

-- parameter : a RobotCommRecord has a list of robot parameter

association parameter between
  RobotCommRecord[1] role parameterOwner
  RobotParameter[*] role hasParameters
end

-- queue : a RobotCommRecord has a priorityQueue

association queue between
  RobotCommRecord[1] role queueOwner
  PriorityQueue[1] role hasQueue
end

-- message : A queue consists of some messages

association message between
  PriorityQueue[*] role inQueue
  Message[*] role hasMessages ordered
end

--
-- C O N S T R A I N T S
--

constraints

--
-- UniqueName
```

```
-- Robot has unique name
--

context RobotCommRecord
  inv UniqueName:
      RobotCommRecord.allInstances->forAll(p1,p2| p1 <> p2
      implies p1.name <> p2.name)

--
-- BroadcastAbility:
-- Robot has broadcast ability can send and receive broadcast messages.
--

context RobotCommRecord
  inv BroadcastAbility1:
      RobotCommRecord.allInstances.hasQueue.hasMessages
        ->select(receiver='broadcast' and sender=self.name)->notEmpty
        implies self.isBroadcastEnabled = true


context r:RobotCommRecord
  inv BroadcastAbility2:
      r.hasQueue.hasMessages
        ->select(receiver='broadcast')->notEmpty
        implies r.isBroadcastEnabled = true


--
-- P2PAbility
-- Robot has point-to-point ability can send and receive point-to-point
-- messages.
--

context RobotCommRecord
  inv P2PAbility1:
      RobotCommRecord.allInstances.hasQueue.hasMessages
        ->select(receiver <> 'broadcast'
        and sender=self.name)->notEmpty
        implies self.isP2PEnabled = true

context r:RobotCommRecord
  inv P2PAbility2:
      r.hasQueue.hasMessages
        ->select(receiver <> 'broadcast')->notEmpty
        implies r.isP2PEnabled = true

--
-- sendAbility
-- Robot with active send link can send messages to other robots
--

context RobotCommRecord
  inv sendAbility:
      RobotCommRecord.allInstances.hasQueue.hasMessages
        ->select(sender=self.name)->notEmpty
        implies self.isSendLinkEnabled = true
```

```
--
-- receiveAbility
-- Robot with active receive link can receive messages from other
robots
--

context r:RobotCommRecord
  inv receiveAbility:
      r.hasQueue.hasMessages->notEmpty
      implies r.isReceiveLinkEnabled = true

--
-- rightQueue
-- Messages are distributed to the right robot queue
--

context RobotCommRecord
   inv rightQueue:
       hasQueue.hasMessages->
       forAll((receiver=self.name) or (receiver='broadcast'))

--
-- priorityQueue
-- The messages in priorityQueue are ordered by received time
--
--

context p:PriorityQueue
  inv priorityQueue:
      Sequence{1..(p.hasMessages->size-1)}->
      forAll(i | p.hasMessages->at(i).receivedTime
      <= p.hasMessages->at(i+1).receivedTime)
--
-- rightTime
-- If received time is defined, then received time is equal or greater
-- than sent time.
--

context m:Message
  inv rightTime:
      m.receivedTime.isDefined implies m.receivedTime >= m.sentTime

--
-- allLinkShutdown
-- Robots cannot send or receive any messages if all the links are --
-- shutdown
--

context c:CommunicationsSystem
  inv allLinkShutdown:
      RobotCommRecord.allInstances.hasQueue.hasMessages->notEmpty
       implies c.isLinkEnabled = true

--
-- sendToYourself
-- Robots cannot get the messages which are sent my themselves.
--
```

```
context r:RobotCommRecord
  inv sendToYourself:
      r.hasQueue.hasMessages->forAll(sender <> r.name)

--
-- O P E R A T I O N S
--
-- registerRobot pre/post-conditions
--  .. pre-conditions
--       1. robot name (n) is defined
--       2. communication type (c) is defined
--       3. c must be only 1,2 or 3
--       4. there's no exist robot named "n" in the RobotCommRecord set
--  .. post-conditions
--       1. new Robot named "n" is created
--       2. new set of RobotCommRecord is the previous set plus new
--          robot named "n"
--       3. there is only one record of RobotCommRecord which is named
--          "n"
--       4. if c=1 then Robot has broadcast capability
--       5. if c=2 then Robot has Point-to-point capability
--       6. if c=3 then Robot has both broadcast and point-to-point
--          capability

context CommunicationsSystem::registerRobot(n:String,c:Integer)
  pre precond_1:    n.isDefined
  pre precond_2:    c.isDefined
  pre precond_3:    (c=1 or c=2 or c=3)
  pre precond_4:    robots->select(name=n)->isEmpty
  post postcond_1:  robots->exists(r | r.oclIsNew and r.name = n)
  post postcond_2:  robots=robots@pre->union(robots->select(name=n))
  post postcond_3:  robots->select(name=n)->size = 1
  post postcond_4:  (c=1) implies robots->select(name=n and
                    isBroadcastEnabled=true
                    and isP2PEnabled=false)->notEmpty
  post postcond_5:  (c=2) implies robots->select(name=n and
                    isP2PEnabled=true and
                    isBroadcastEnabled=false)->notEmpty
  post postcond_6:  (c=3) implies robots->select(name=n and
                    isBroadcastEnabled=true and
                    isP2PEnabled=true)->notEmpty

--
-- sendMessage pre/post-conditions
--  .. pre-conditions
--       1. timeStep is greater than zero
--       2. message is defined
--       3. sender is defined
--       4. receiver is defined
--       5. System link is enabled
--       6. sender's send link is active
--       7. if it is broadcast message, sender has broadcast capability
--       8. if it is point-to-point message, sender has point-to-point
--          capability
--
--  .. post-conditions
```

```
--       1. sentTime which is timeStep is added to the Message
--       2,3. receivedTime is added based on system delay and robot ---
--          delay
--       The msg is added to the receiver's queue
--       4. if it is broadcast message then
--            message is added to all robot's queue except sender's --
--            queue
--       5. if it is Point-to-point message then
--            the message is added to the specified robot's queue

context CommunicationsSystem::sendMessage(msg:Message,timeStep:Integer)
  pre precond_1:    timeStep > 0 and timeStep.isDefined
  pre precond_2:    msg.isDefined
  pre precond_3:    msg.sender.isDefined
  pre precond_4:    msg.receiver.isDefined
  pre precond_5:    isLinkEnabled = true
  pre precond_6:    robots->select(name=msg.sender)
                     ->forAll(isSendLinkEnabled = true)
  pre precond_7:    msg.receiver = 'broadcast'
                      implies robots->select(name=msg.sender)->
                      forAll(isBroadcastEnabled = true)
  pre precond_8:    msg.receiver <> 'broadcast'
                      implies robots->select(name=msg.sender)->
                      forAll(isP2PEnabled = true)

  post postcond_1:  msg.sentTime = timeStep
  post postcond_2:  msg.receiver <> 'broadcast' implies
                      robots->select(name=msg.sender)->
                      forAll(hasParameters->
                      forAll(receiverName=msg.receiver implies
                      msg.receivedTime = parameterOwner.belongTo.delay
                      + timeStep + delayTime))

  post postcond_3:  msg.receiver = 'broadcast' implies
                      robots->select(name=msg.sender)->
                      forAll(hasParameters->
                      forAll(msg.receivedTime =
                      parameterOwner.belongTo.delay
                      + timeStep + delayTime))

  post postcond_4:  msg.receiver = 'broadcast' implies
                      robots->select(name <> msg.sender
                      and isReceiveLinkEnabled = true
                      and isBroadcastEnabled = true)
                      ->forAll(r| r.hasQueue.hasMessages->asSet
                      = r.hasQueue.hasMessages@pre
                      ->including(msg)->asSet)

  post postcond_5:  msg.receiver <> 'broadcast' implies
                      robots->select(name = msg.receiver
                      and isReceiveLinkEnabled = true
                      and isP2PEnabled = true)
                      ->forAll(r| r.hasQueue.hasMessages->asSet
                      = r.hasQueue.hasMessages@pre
                      ->including(msg)->asSet)
```

```
--
-- getMessage pre/post-conditions
--  .. pre-conditions
--       1. n (robot name) is defined
--       2. robot named n exists in the system
--       3. timestep is defined
--       4. timestep is greater than zero
--
--  .. post-conditions
--       1. The messages in priority queue of robot n
--          excludes message which has receivedTime = timestep
--       2. Return result which is a set of messages
--          which receivedTime = timestep
--

context CommunicationsSystem::getMessage(n:String,timeStep:Integer)
:Set(Message)
  pre precond_1:    n.isDefined
  pre precond_2:    robots.exists(r| r.name=n)
  pre precond_3:    timeStep.isDefined
  pre precond_4:    timeStep > 0
  post postcond_1:  robots->select(name = n)
                     ->forAll(r | r.hasQueue.hasMessages->asSet
                     = r.hasQueue.hasMessages@pre->asSet
                     - r.hasQueue.hasMessages@pre
                     ->select(receivedTime = timeStep)->asSet)

  post postcond_2:  robots->select(name=n)
                     ->forAll(r| result = r.hasQueue.hasMessages@pre
                     ->select(receivedTime = timeStep)->asSet)
```