Asynchronous Queuing Pattern

Event-B Pattern Specifications

Asynchronous Queuing Specifications

Level 0

Context AQC0

```
CONTEXT
   AQC0
SETS
   Entity
   MessageType
CONSTANTS
   Participant
   ServiceA
   ServiceB
   ServiceChannel
   RequestMessage
   PushAB
   PushBA
   ResponseMessage
   Can_Send
AXIOMS
   axm2    :   partition(Entity, Participant)
   axm9    :   partition(Participant, {ServiceA},{ServiceB})
   axm1    :   ServiceChannel ∈ Entity ↔ Entity
   axm6    :   partition(MessageType, {RequestMessage}, {ResponseMessage})
   axm13    :   partition(ServiceChannel, {PushAB}, {PushBA})
   axm14    :   dom({PushAB})={ServiceA}
   axm15    :   ran({PushAB})={ServiceB}
   axm16    :   dom({PushBA})={ServiceB}
   axm17    :   ran({PushBA})={ServiceA}
   axm18    :   ServiceChannel ∩ id = ∅
   Can_Send_Relation    :   Can_Send ∈ Entity ↔ MessageType
   axm19    :   Can_Send = {ServiceA↦RequestMessage, ServiceB↦ResponseMessage}
END
```

Machine AQM0

```
MACHINE
   AQM0
SEES
   AQC0
VARIABLES
   Dispo
   Send
   Process
INVARIANTS
   Dispo_Function    :   Dispo ∈ Participant ⇸ BOOL
   Send_Relation     :   Send ∈ ServiceChannel ↔ MessageType
   Process_Function  :   Process ∈ Participant⇸MessageType
EVENTS
   INITIALISATION   ≙
      STATUS
    ordinary
   BEGIN
    act8    :   Send ≔ ∅
    act9    :   Process ≔ ∅
    act10   :   Dispo ≔ {ServiceA↦TRUE, ServiceB↦FALSE}
   END
```

```
Sending_Request   ≙
    STATUS
 ordinary
WHEN
 grd1    :   Send =∅
 grd2    :   ServiceA ∈ dom(Dispo) ∧ Dispo(ServiceA)=TRUE
THEN
 act1    :   Send ≔ Send ∪ {PushAB↦RequestMessage}
 act2    :   Dispo(ServiceA) ≔ FALSE
END

Processing_Request   ≙
    STATUS
 ordinary
WHEN
 grd1    :   RequestMessage ∈ ran(Send)
 grd2    :   RequestMessage ∉ ran(Process)
 grd3    :   ServiceB ∈ dom(Dispo)∧ Dispo(ServiceB)=TRUE
THEN
 act1    :   Process ≔Process ⩤ {ServiceB ↦ RequestMessage}
END
```

Machine AQM0

```
Sending_Response   ≙
    STATUS
 ordinary
WHEN
                ServiceB ∈ dom                          //
 grd2    :      (Dispo) ∧ Dispo(ServiceB)          (Send)
                =TRUE
 grd3    :      RequestMessage ∈ ran(Process)
 grd4    :      ResponseMessage ∉ ran(Send)
THEN
 act1    :      Send ≔ Send ∪ {PushBA↦ResponseMessage}
 act2    :    Dispo(ServiceB) ≔ FALSE
END

Processing_Response   ≙
    STATUS
 ordinary
WHEN
 grd1    :      ResponseMessage ∈ ran(Send)
 grd2    :    ServiceA ∈ dom(Dispo) ∧ Dispo(ServiceA) = TRUE
THEN
 act1    :    Dispo(ServiceA) ≔ FALSE
 act2    :    Send≔∅
 act3    :    Process ≔Process ⩤ {ServiceA ↦ ResponseMessage}
END
```

```
Activating_Participant   ≙
    STATUS
 ordinary
ANY
 P       //   Participant
WHERE
 grd1    :    P ∈ Participant
 grd2    :    P ∈ dom(Dispo)∧ Dispo(P)=FALSE
THEN
 act1    :    Dispo ≔ Dispo ⩤ {P↦TRUE}
END

END
```

Asynchronous Queuing Specifications

Level 1

Context AQC1

```
CONTEXT
   AQC1
EXTENDS
   AQC0
CONSTANTS
   Queue
   PushAQ
   PushQB
   PushBQ
   PushQA
AXIOMS
   participant_partition   :   partition(Participant,
                               { ServiceA } , { ServiceB }, { Queue})
   axm1   :   partition(ServiceChannel, {PushAQ},{PushQB},{PushBQ},{PushQA})
   axm14   :   {PushAB} = {PushAQ} ; {PushQB}
   axm6   :   ran({PushAQ})={Queue}
   axm8   :   dom({PushQB})={Queue}
   axm15   :   {PushBA} = {PushBQ} ; {PushQA}
   axm11   :   ran({PushBQ})={Queue}
   axm13   :   dom({PushQA})={Queue}
END
```

Machine AQM1

```
MACHINE
  AQM1
REFINES
  AQM0
SEES
  AQC1
VARIABLES
  Dispo
  Send
  Process
  Stores
  Transmit
INVARIANTS
  inv1   :   Stores ∈ Participant ↔ MessageType
  inv2   :   dom(Stores)={Queue} ∨ Stores=∅
  inv3   :   Transmit ∈ Participant ⇸ MessageType
  inv4   :   dom(Transmit)={Queue} ∨ Transmit=∅
EVENTS
  INITIALISATION   ≙
   extended
     STATUS
   ordinary
  BEGIN
   act8   :   Send ≔ ∅
   act9   :   Process ≔ ∅
   act10   :   Dispo ≔ {ServiceA↦TRUE, ServiceB↦FALSE}
   act12   :   Stores ≔ ∅
   act11   :   Transmit ≔ ∅
  END
```

```
Sending_Request   ≙
 extended
    STATUS
 ordinary
REFINES
 Sending_Request
WHEN
 grd1   :   Send =∅
 grd2   :   ServiceA ∈ dom(Dispo) ∧ Dispo(ServiceA)=TRUE
THEN
 act1   :   Send ≔ Send ∪ {PushAB↦RequestMessage}
 act2   :   Dispo(ServiceA) ≔ FALSE
END

Storing_Request   ≙
    STATUS
 ordinary
WHEN
 grd1   :   RequestMessage ∈ ran(Send)
 grd2   :   RequestMessage ∉ ran(Process)
 grd3   :   ServiceB ∈ dom(Dispo) ∧ Dispo(ServiceB)=FALSE
 grd4   :   Stores = ∅
THEN
 act1   :   Stores ≔ Stores ∪ {Queue↦RequestMessage}
END
```

Machine AQM1

```
Transmissing_Request   ≙
    STATUS
 ordinary
WHEN
 grd1    :   RequestMessage ∈ ran(Stores)
THEN
 act1    :   Transmit ≔ Transmit ⩤ {Queue↦RequestMessage}
END

Processing_Request   ≙
 extended
    STATUS
 ordinary
REFINES
 Processing_Request
WHEN
 grd1    :   RequestMessage ∈ ran(Send)
 grd2    :   RequestMessage ∉ ran(Process)
 grd3    :   ServiceB ∈ dom(Dispo)∧ Dispo(ServiceB)=TRUE
 grd4    :   RequestMessage ∈ ran(Transmit)
THEN
 act1    :   Process ≔Process ⩤ {ServiceB ↦ RequestMessage}
 act2    :   Stores ≔ ∅
END
```

```
Sending_Response   ≙
 extended
    STATUS
 ordinary
REFINES
 Sending_Response
WHEN
                ServiceB ∈ dom                    //   RequestMess
 grd2    :      (Dispo) ∧ Dispo(ServiceB)      (Send)
                =TRUE
 grd3    :      RequestMessage ∈ ran(Process)
 grd4    :      ResponseMessage ∉ ran(Send)
THEN
 act1    :      Send ≔ Send ∪ {PushBA↦ResponseMessage}
 act2    :      Dispo(ServiceB) ≔ FALSE
END

Storing_Response   ≙
    STATUS
 ordinary
WHEN
 grd1    :   ResponseMessage ∈ ran(Send)
 grd2    :   ResponseMessage ∉ ran(Process)
 grd3    :   ServiceA ∈ dom(Dispo) ∧ Dispo(ServiceA)=FALSE
 grd4    :   Stores = ∅
THEN
 act1    :   Stores ≔ Stores ∪ {Queue↦ResponseMessage}
END

Processing_Response   ≙
 extended
```

Machine AQM1

```
    STATUS
  ordinary
REFINES
  Processing_Response
WHEN
  grd1   :      ResponseMessage ∈ ran(Send)
  grd2   :    ServiceA ∈ dom(Dispo) ∧ Dispo(ServiceA) = TRUE
THEN
  act1   :    Dispo(ServiceA) ≔ FALSE
  act2   :    Send≔∅
  act3   :    Process ≔Process ⩤ {ServiceA ↦ ResponseMessage}
END

Activating_Participant   ≙
  extended
    STATUS
  ordinary
REFINES
  Activating_Participant
ANY
  P       //   Participant
WHERE
  grd1   :    P ∈ Participant
  grd2   :    P ∈ dom(Dispo)∧ Dispo(P)=FALSE
THEN
  act1   :    Dispo ≔ Dispo ⩤ {P↦TRUE}
END

END
```