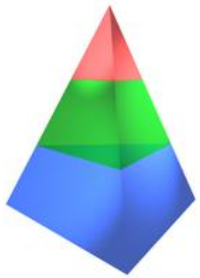
Van DEMO naar een Relationele database



Michael Fidom

DEMO platform

30 november 2009

Mijn achtergrond

- HTS Elektrotechniek (elektronica)
- TU Technische Informatica (2002)
  - Ontwerpen van informatiesystemen
  - Afgestudeerd op:
    DEMO toegepast bij reorganisatie van een afdeling bij Luchtverkeersleiding Nederland
- Manager functie bij LVNL
  - Ook: bedrijfsprocessen en informatiesysteem
- In vrije tijd DEMO toepassen
  - In MS Access database (op werk gebruikt)
- E-mail: michael.fidom@xs4all.nl

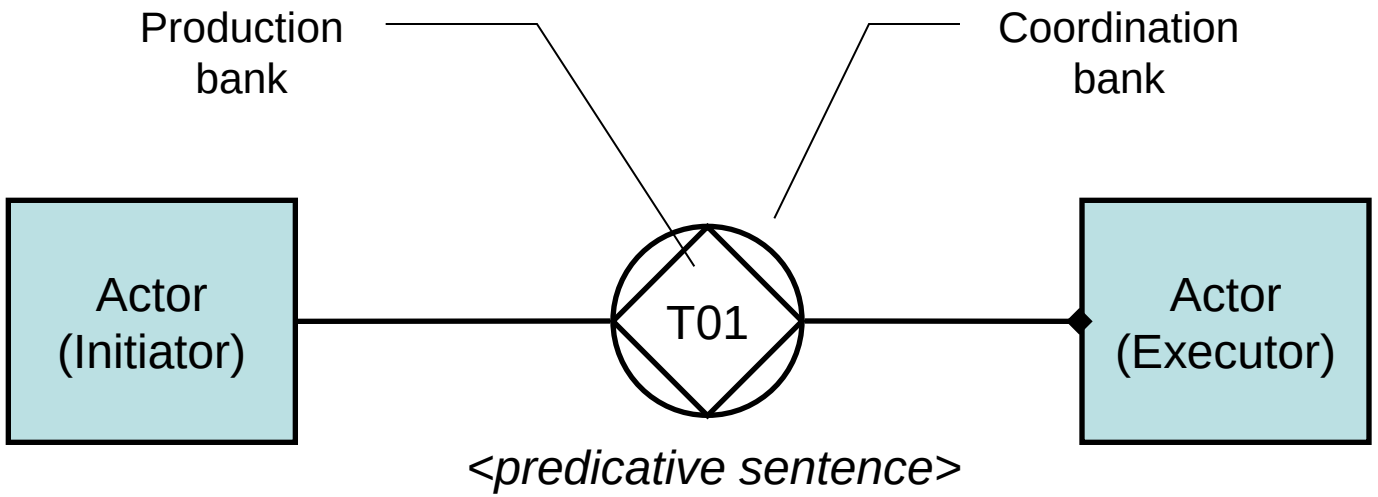
Inhoud

- Project “DEMO database”
- Wat is een transactieresultaat?
- Welke transactie (actor) is initiator?
- Implementatie van een production bank en een coordination bank
- Genereren van tabellen en query’s
- Cancellation en state machines
- Voorbeeld gebruikers interface (form)

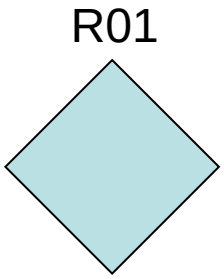
Project DEMO database

- Automatisch vertalen van transactie typen en feit typen naar tabellen in een MS Access relationele database;
- Registreren van alle C-acts (historie);
- Transactie status afleiden van C-acts;
- C-act “scripts” om intuïtieve transactie transities met stilzwijgende C-acts te ondersteunen, zoals:
  - PromiseDifferently (andere datum dan requested)
  - PromiseChange (wijzigen van promised datum)
  - RejectAndReDo (verbeteren productie)
- Conversie van bestaande database naar “demo database”;
- Gemakkelijke gebruikers interface:
  - geen last maar gemak van DEMO
  - met stilzwijgende C-acts

Elementair transactietype



Wat is een transactieresultaat?



<predicative sentence>

- Elke transactie is een exemplaar van een bepaald transactietype.
- Van een transactie wordt de predicatieve zin “waar” als het productieresultaat is geaccepteerd.
- Het transactieresultaat is afhankelijk van de Coördinatie feiten (C-fact accepted).
- Het transactieresultaat is een afgeleid feit.

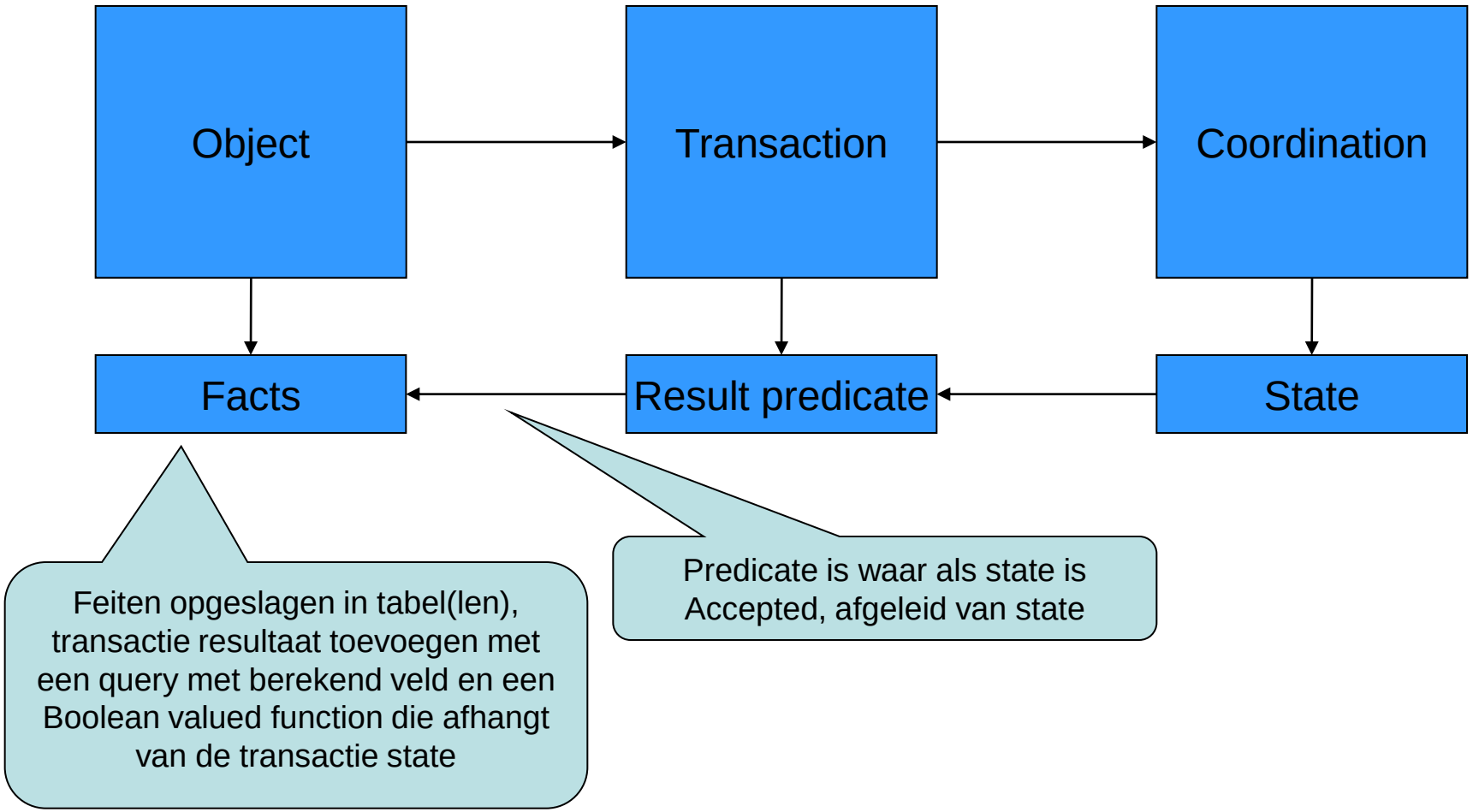
Resultaat en feiten

- Resultaat
  - Het lidmaatschap <L> is begonnen (is afgeleid van transactie status n.a.v. C-acts)
- Feit
  - Persoon <P> is het lid in <L>
  - De achternaam van <P> is <AN>
  - De voornaam van <P> is <VN>

Gevolgen voor implementatie

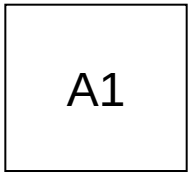
- Voor een transactieresultaat is geen veld in een tabel (van een relationele database) nodig.
- De waarheid van een predicatieve zin kan met een SQL query (en functie) worden verkregen uit een tabel met geregistreerde C-facts.

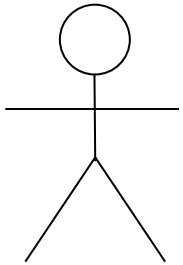
De relatie van resultaat en feiten



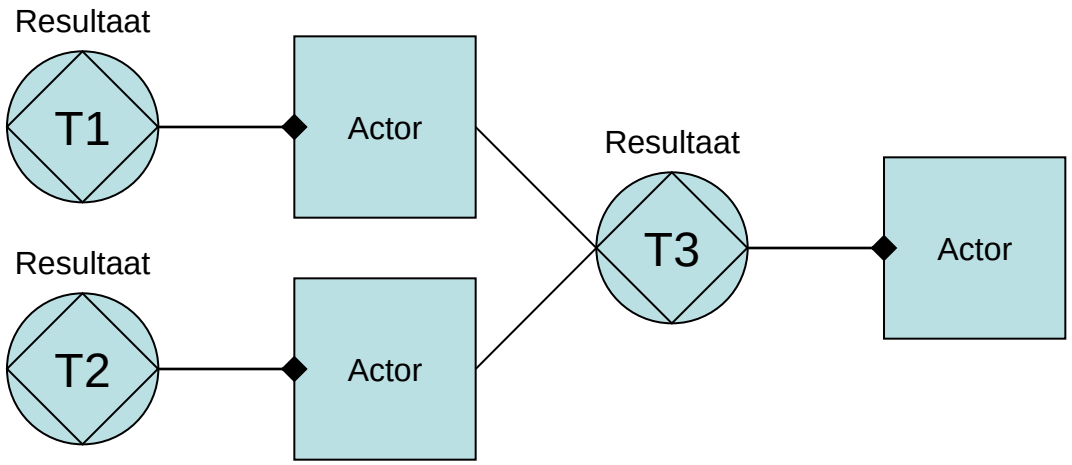
Actor

- Een transaction type wordt uitgevoerd door één actor type.
- Een transaction (instance) wordt uitgevoerd door één actor.
- Een transaction type kan worden geïnitieerd door één of meer actor typen.
- Een transaction (instance) kan worden geïnitieerd door één actor.



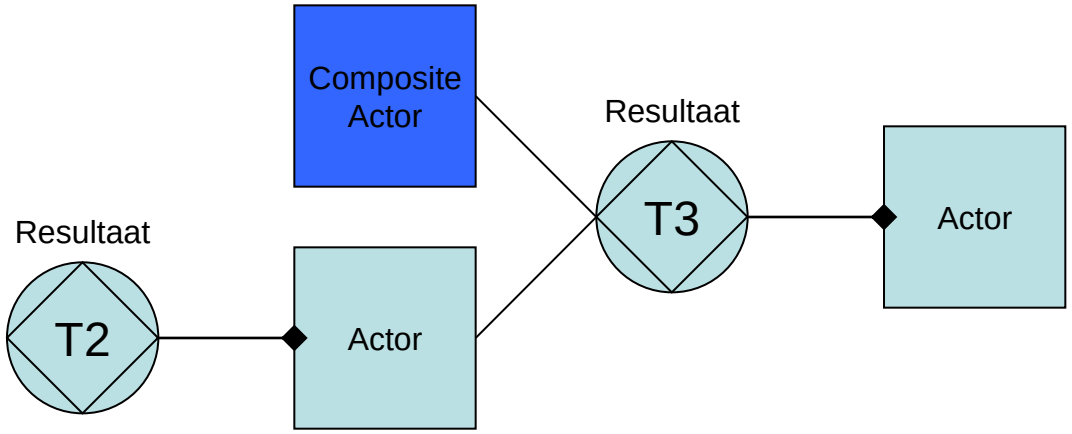


Welke actor is initiator?



- Dit kan niet worden afgeleid van het Actor Transactie Diagram omdat dit Transactie Typen betreft.
- In de Production Bank van T3 kan worden geregistreerd welke Actor de initiator is van een transactie (instance of transaction type).

Ook bij composite actor



- In de Production Bank van T3 kan worden geregistreerd welke Actor de initiator is van een transactie (instance of transaction type).

Universele structuur tabellen

- Production Bank (transaction instances)
  - Verwijzing naar Initiating actor
  - Bevat data Executing actor
  - Ook gebruikt voor composite actor
  - Bevat roles voor verwijzing naar objects
- Coordination Bank (C-acts)

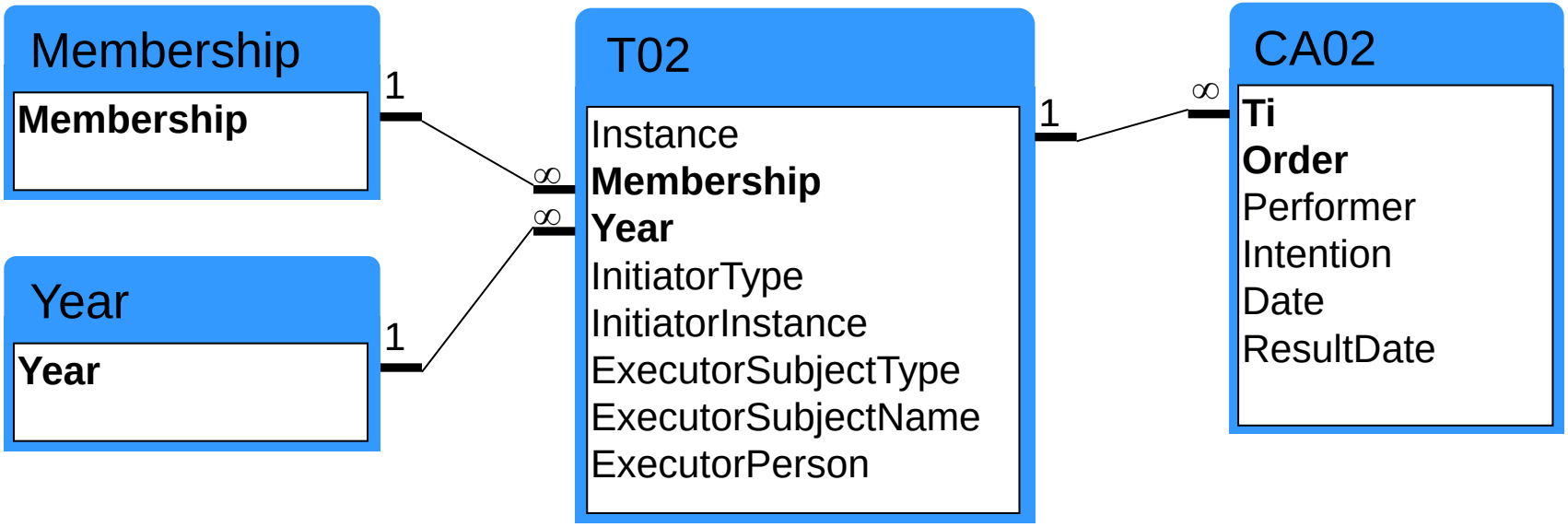
Interpreteren van een result type

- Vertalen van “result type sentence”
  - Lexical analysis (tokens, role = token)
  - Syntax analysis (parser)
  - Tabellen genereren
  - Query genereren
- Voorbeelden:
  - <Membership> has been started
  - The membership fee for <Membership> regarding year <Year> has been paid

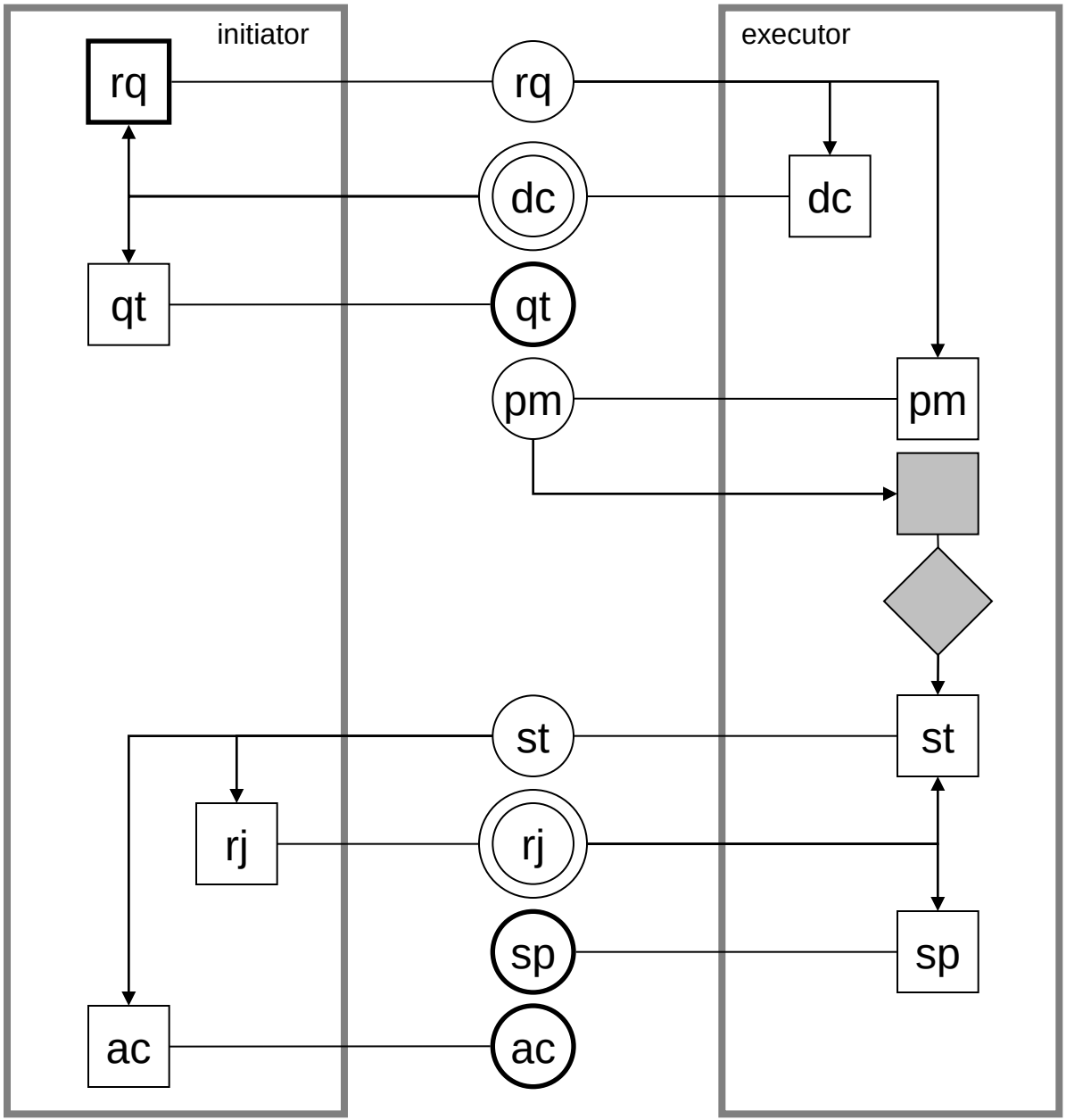
Van result formulation naar tabellen

Library example R02:

The membership fee for <Membership> regarding year <Year> has been paid



Standard pattern transaction



rq: request

pm: promise

st: state

ac: accept

dc: decline

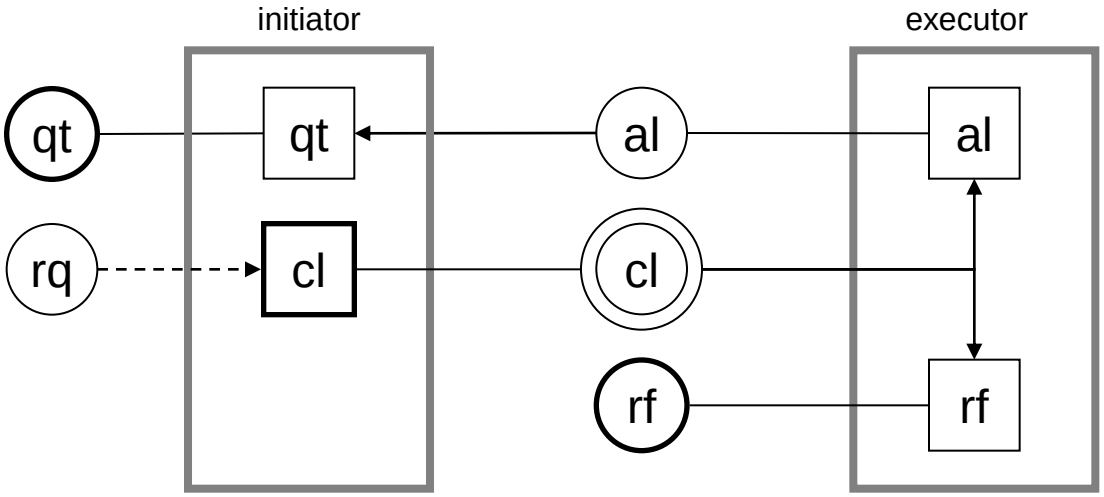
qt: quit

rj: reject

sp: stop

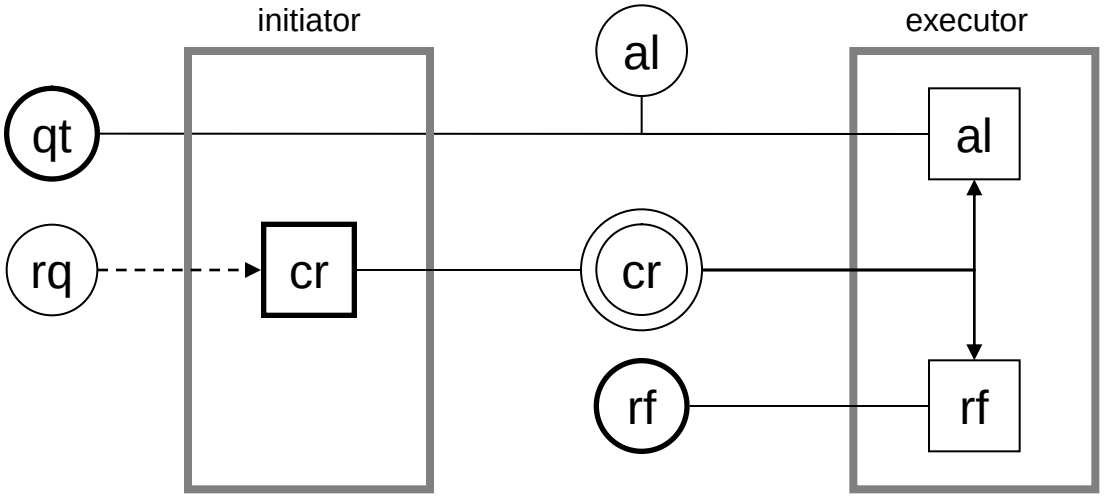
Boek fig. 10.2

Request cancellation pattern



rq: request
cl: cancel
al: allow
rf: refuse
qt: quit

Boek fig. 10.3



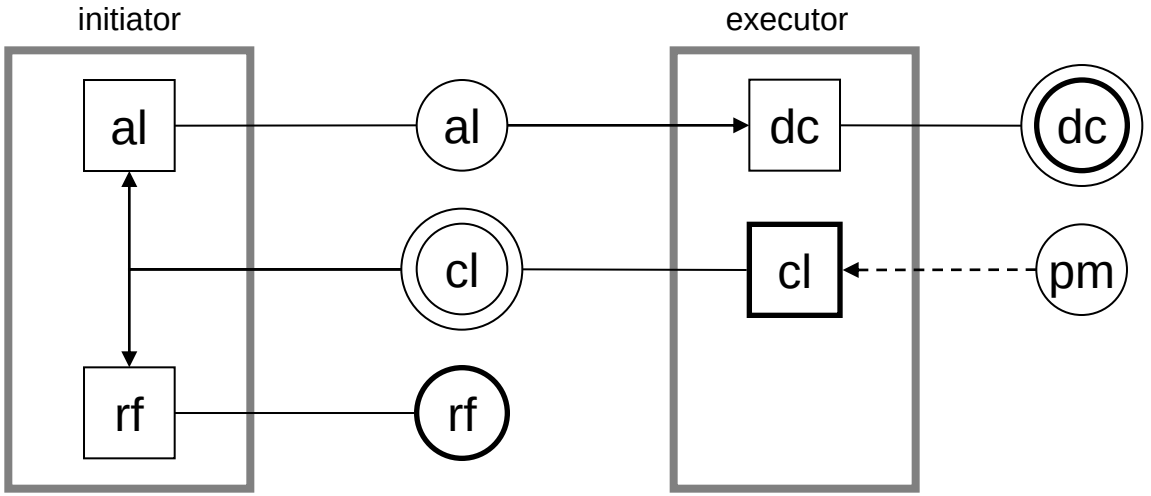
rq: request
cr: cancel request
al: allow
rf: refuse
qt: quit

Voorstel

Promise cancellation pattern

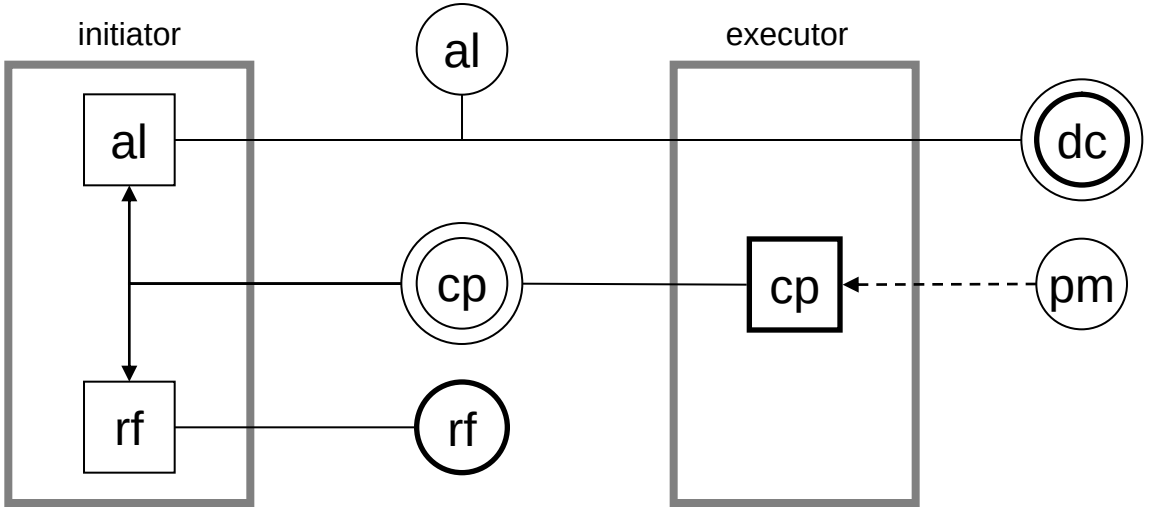
pm: promise
cl: cancel
al: allow
rf: refuse
dc: decline

Boek fig. 10.4



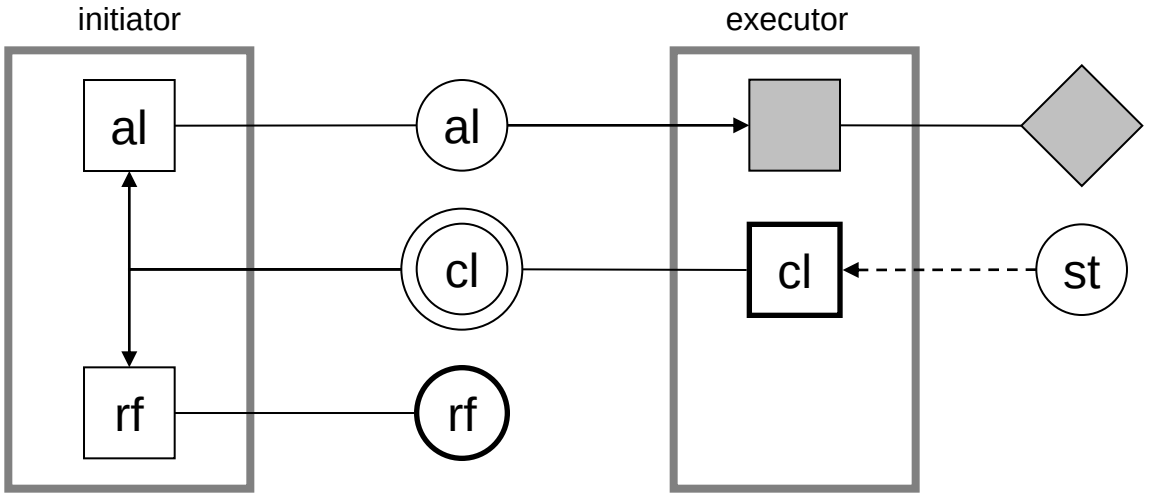
pm: promise
cp: cancel promise
al: allow
rf: refuse
dc: decline

Voorstel



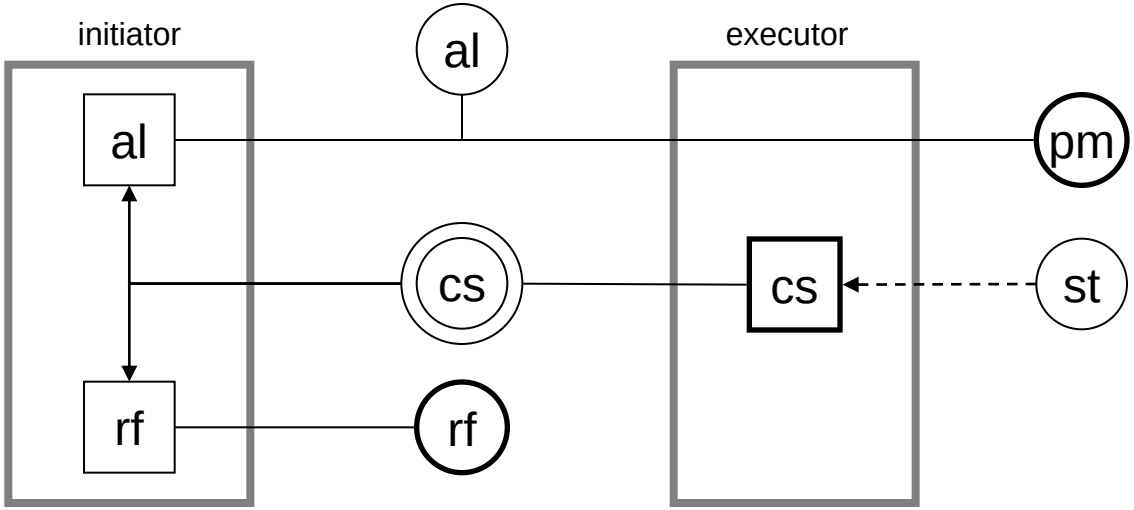
State cancellation pattern

st: state
cl: cancel
al: allow
rf: refuse



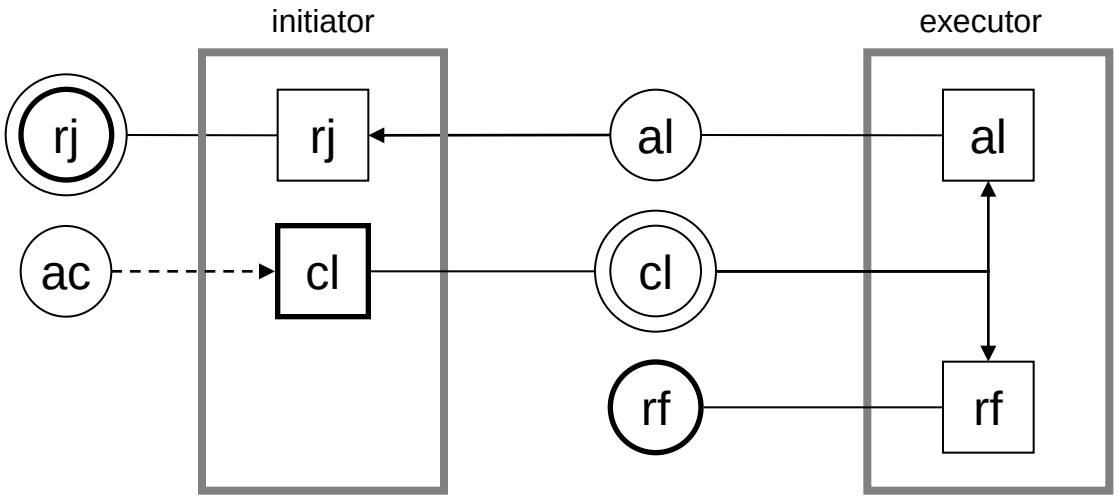
Boek fig. 10.5

st: state
cs: cancel state
al: allow
rf: refuse
pm: promise



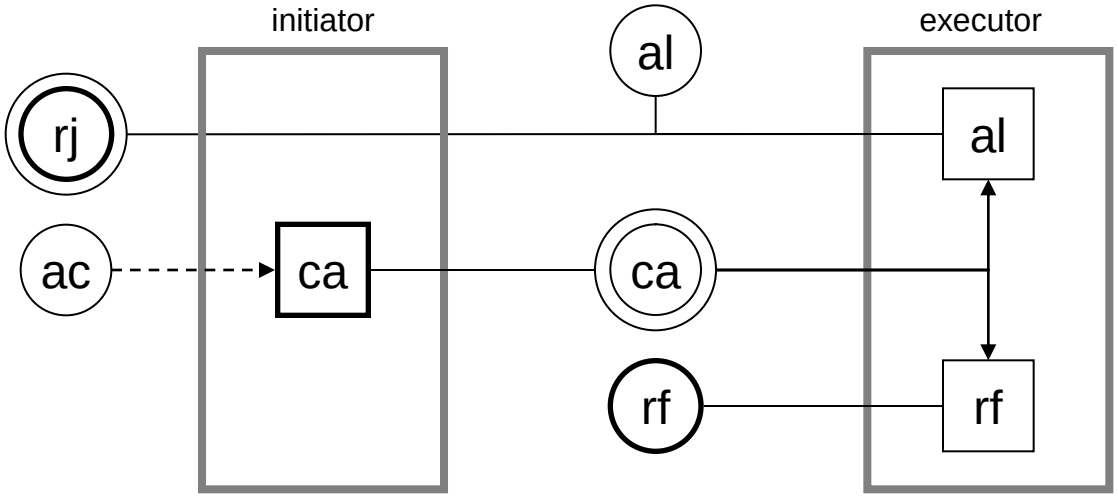
Voorstel

Accept cancellation pattern



ac: accept
cl: cancel
al: allow
rf: refuse
rj: reject

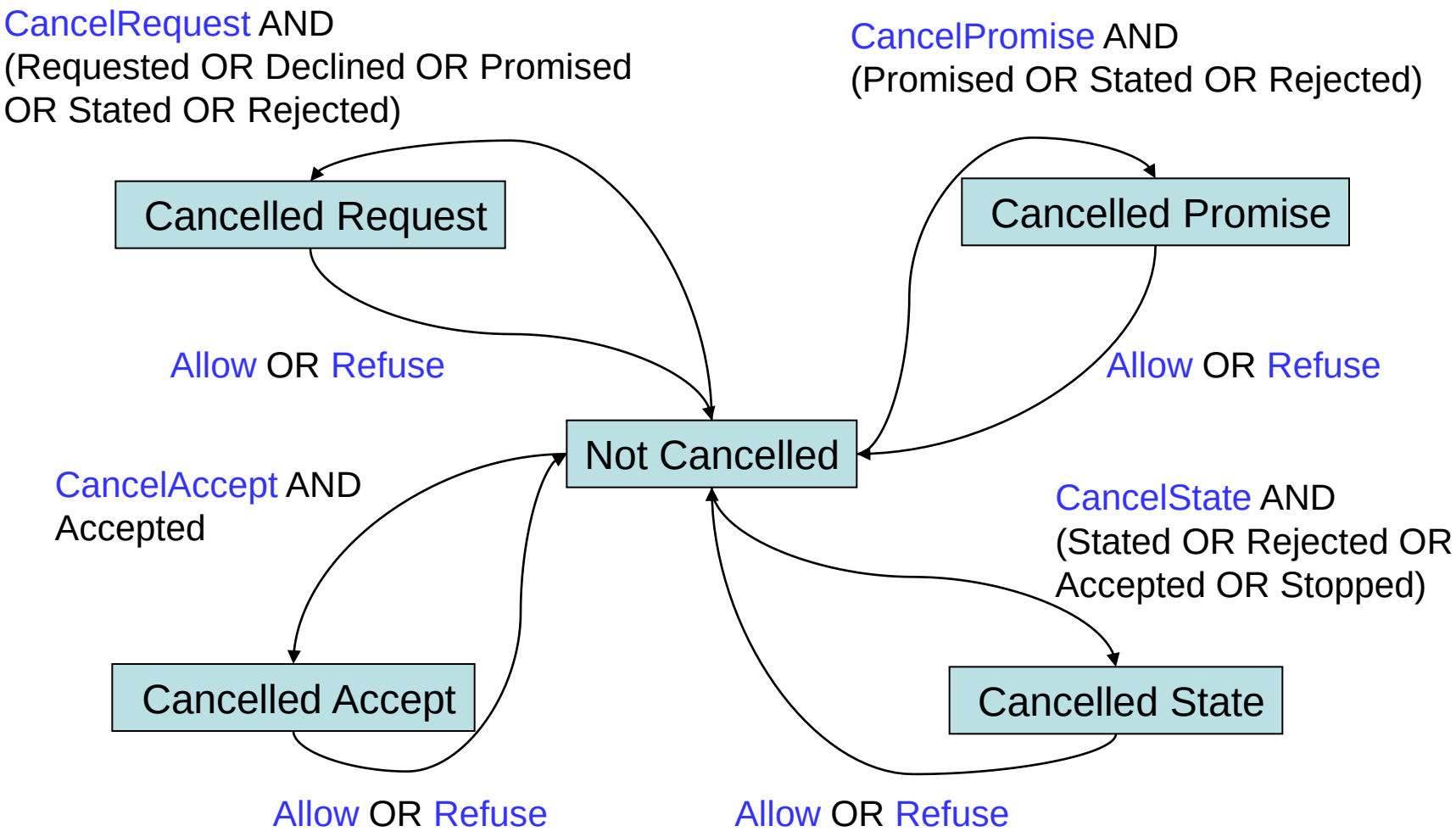
Boek fig. 10.6



ac: accept
ca: cancel accept
al: allow
rf: refuse
rj: reject

Voorstel

Finite state machine cancellations

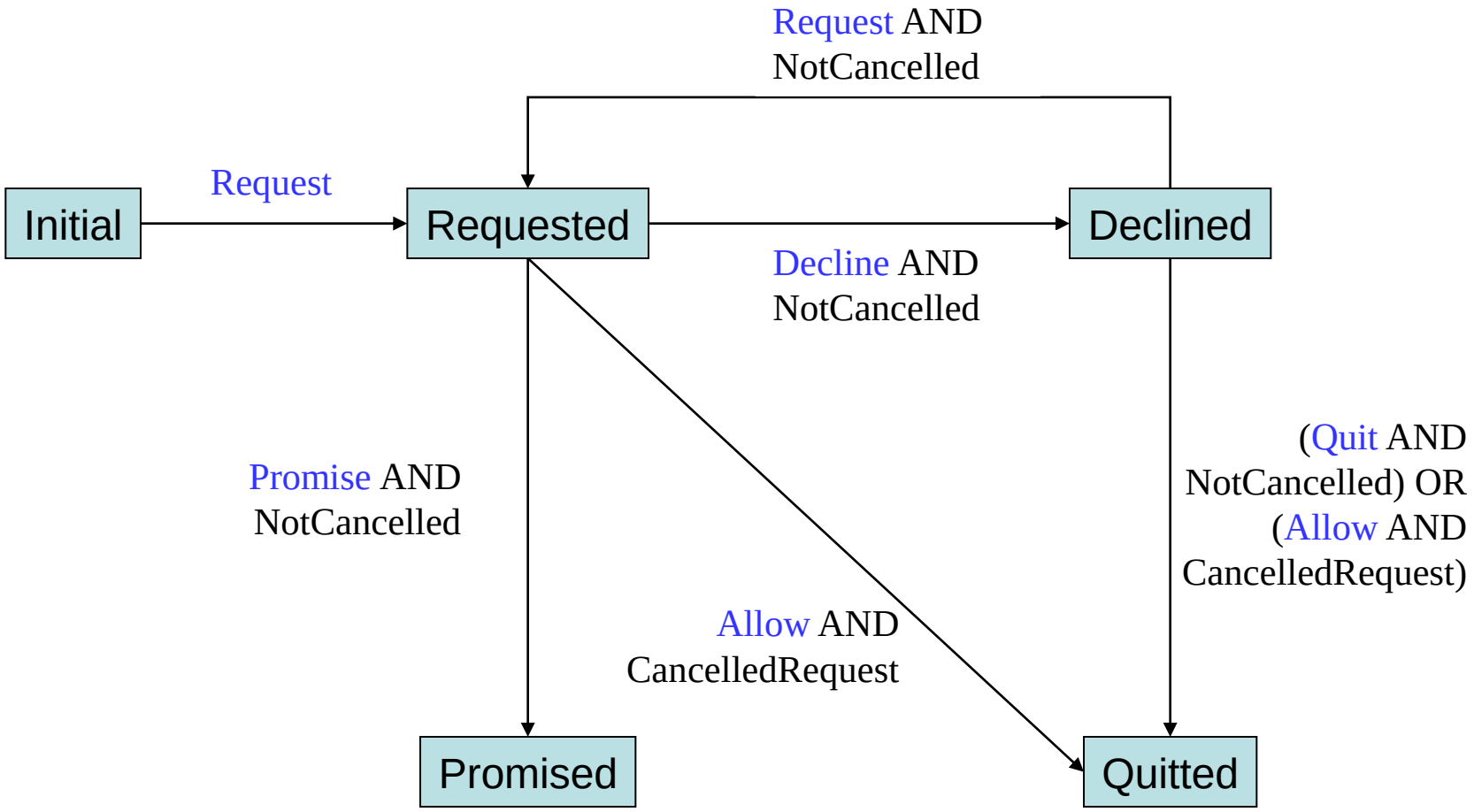


C-acts en Statemachine states

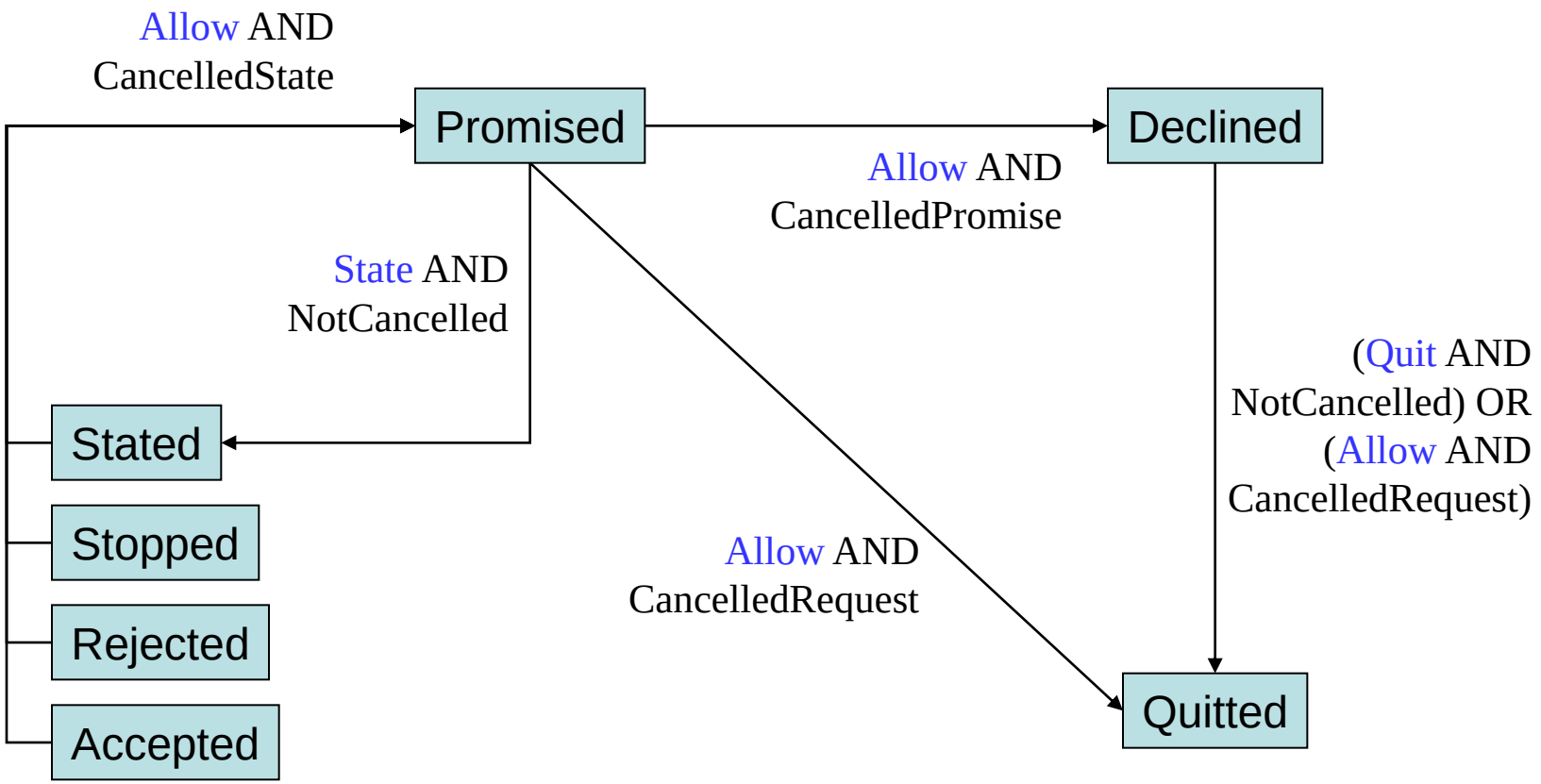
```
Public Enum CactEnum
    caRequest = 1
    caPromise = 2
    caState = 3
    caAccept = 4
    caDecline = 5
    caReject = 6
    caQuit = 7
    caStop = 8
    caCancelRequest = 11
    caCancelPromise = 12
    caCancelState = 13
    caCancelAccept = 14
    caAllow = 15
    caRefuse = 16
End Enum
```

```
Public Enum StateEnum
    stInitial = 0
    stRequested = 1
    stPromised = 2
    stStated = 3
    stAccepted = 4
    stDeclined = 5
    stRejected = 6
    stQuitted = 7
    stStopped = 8
    stNotCancelled = 10
    stCancelledRequest = 11
    stCancelledPromise = 12
    stCancelledState = 13
    stCancelledAccept = 14
End Enum
```

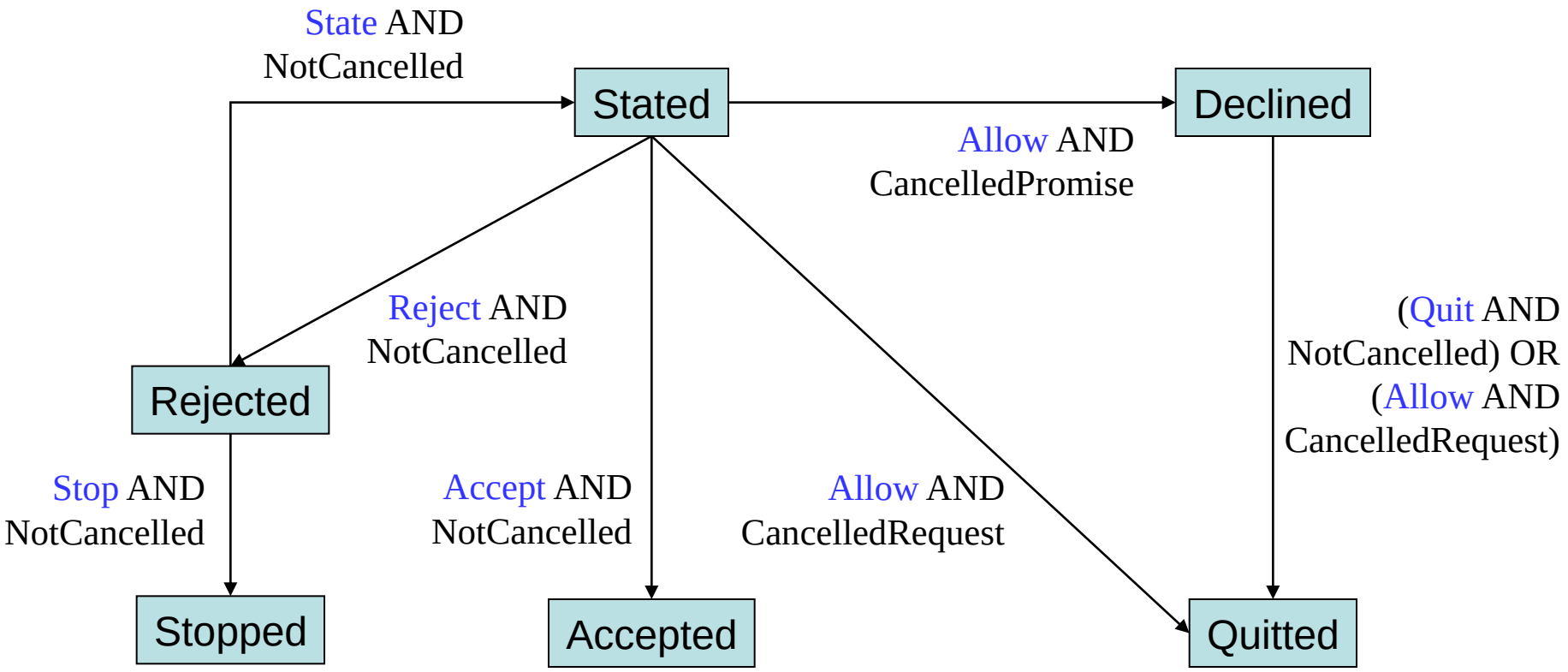
FSM transitions: Requested



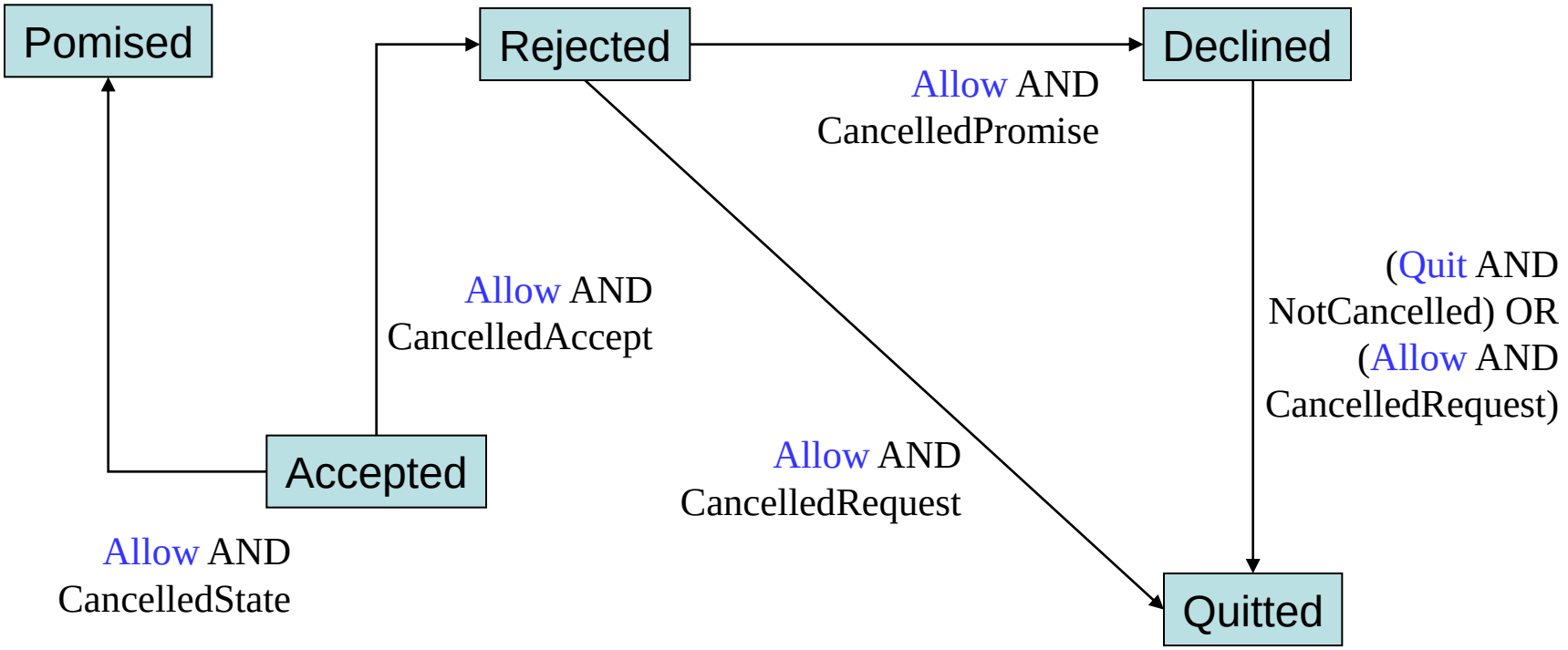
FSM transitions: Promised



FSM transitions: Stated



FSM transitions: Accepted



Request C-act programmacode

```
Public Sub Request(PerformerName, CoordinationDate, ProductionDate)

    Select Case TransactionState
        Case stInitial, stDeclined
            AddCoordination caRequest, PerformerName, CoordinationDate,
                                        ProductionDate
        Case Else
            'Error: Initial or declined state is required to request
    End Select
End Sub
```

Transactie resultaat uit C-acts

```
Public Property Get Actual() As Boolean

    UpdateStateMachines
    If TransactionMachine = stAccepted Then
        Actual = True
    Else
        Actual = False
    End If
End Property
```

Fragment van programmacode Function TransactionNextState

```
Private Function TransactionNextState(CurrentState As StateEnum, Cact As
                                      CactEnum, CancellationState) As StateEnum

    TransactionNextState = CurrentState
    Select Case CurrentState
        Case stInitial
            Select Case Cact
                Case caRequest
                    TransactionNextState = stRequested
            End Select
              .
              .
```

Fragment van programmacode
Function TransactionNextState (2)

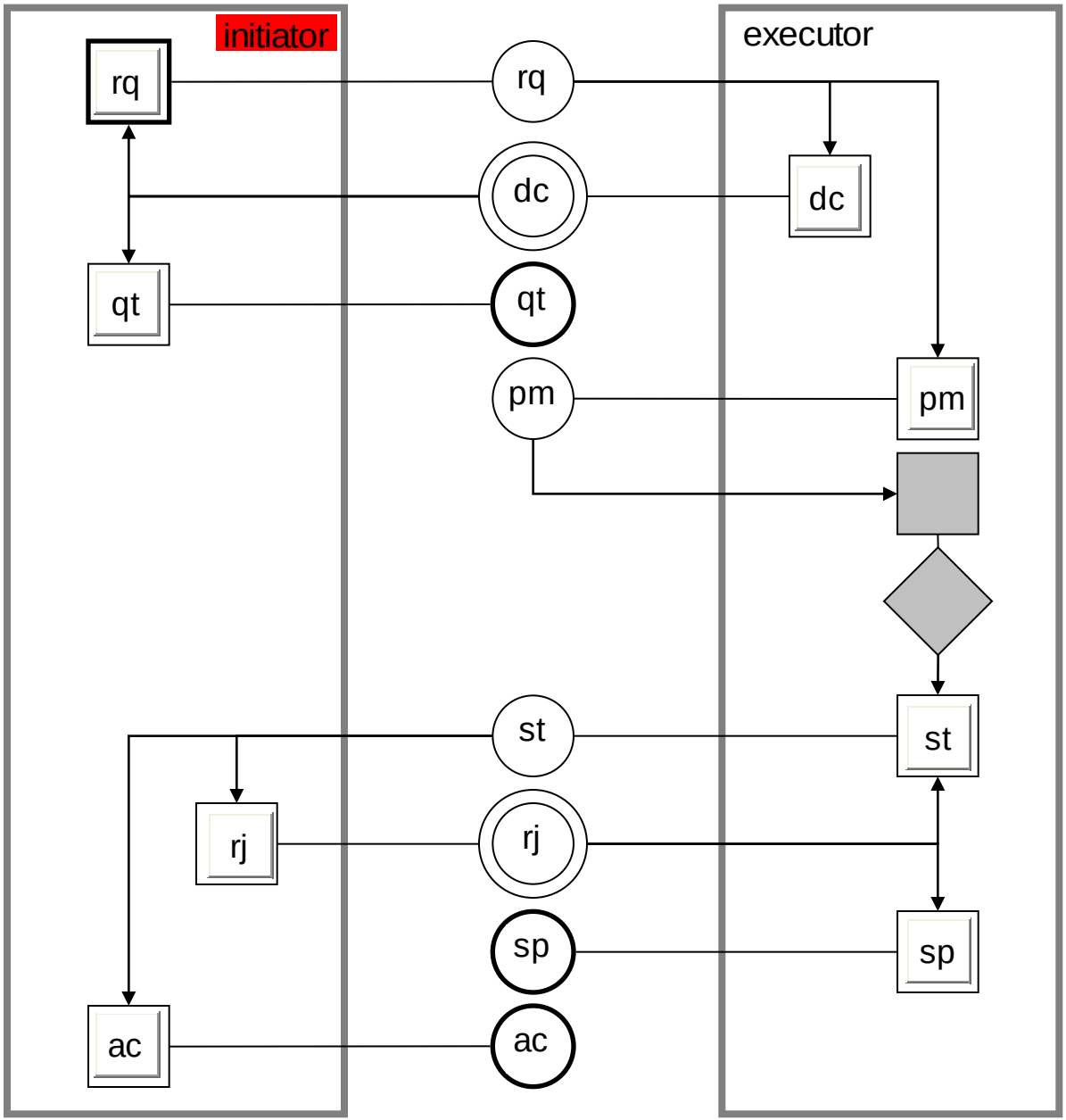
```
Case stRequested
        Select Case Cact
            Case caPromise
                If CancellationState = stNotCancelled Then
                    TransactionNextState = stPromised
                End If
            Case caDecline
                If CancellationState = stNotCancelled Then
                    TransactionNextState = stDeclined
                End If
            Case caAllow
                If CancellationState = stCancelledRequest Then
                    TransactionNextState = stQuitted
                End If
        End Select
```

Fragment van programmacode Function CancellationNextState

```
Select Case CurrentState
    Case stNotCancelled
        Select Case Cact
            Case caCancelRequest
                Select Case TransactionState
                    Case stRequested, stDeclined, stPromised, stStated, stRejected
                        CancellationNextState = stCancelledRequest
                End Select
            Case caCancelPromise
                Select Case TransactionState
                    Case stPromised, stStated, stRejected
                        CancellationNextState = stCancelledPromise
                End Select
```

Standard pattern transaction & state machine

init



rq: request

pm: promise

st: state

ac: accept

dc: decline

qt: quit

rj: reject

sp: stop

Cancel Request

Cancel Promise

Cancel State

Cancel Accept

Allow

Refuse

Boek fig. 10.2

Voorbeeld gebruikersformulier