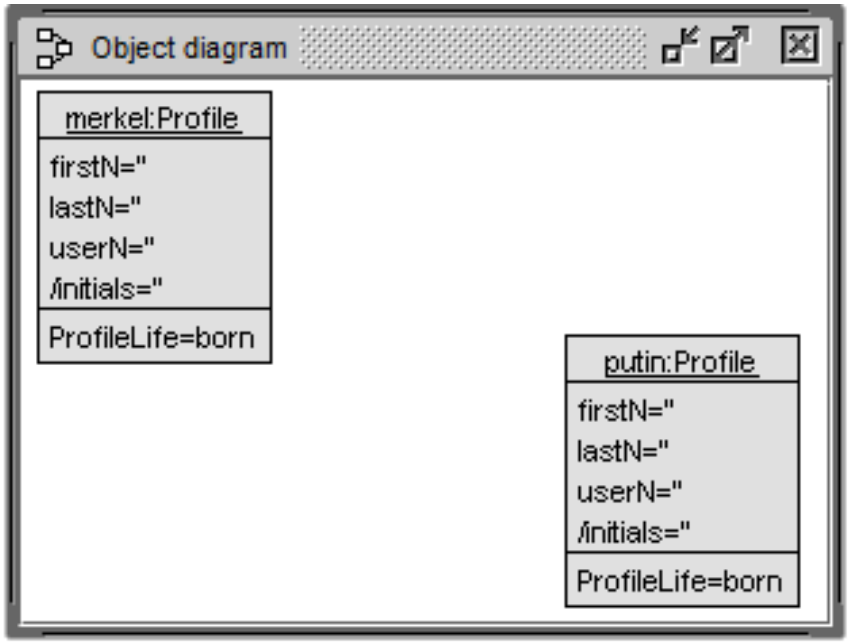
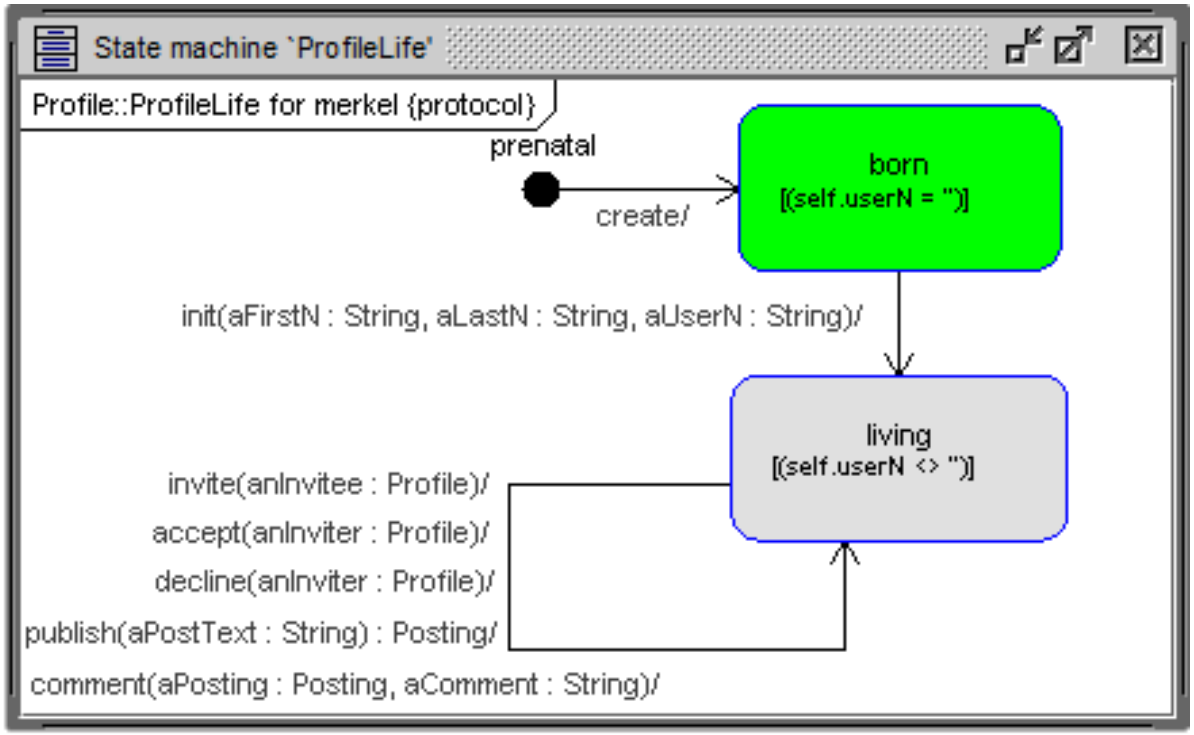
Animation and documentation of a simple scenario

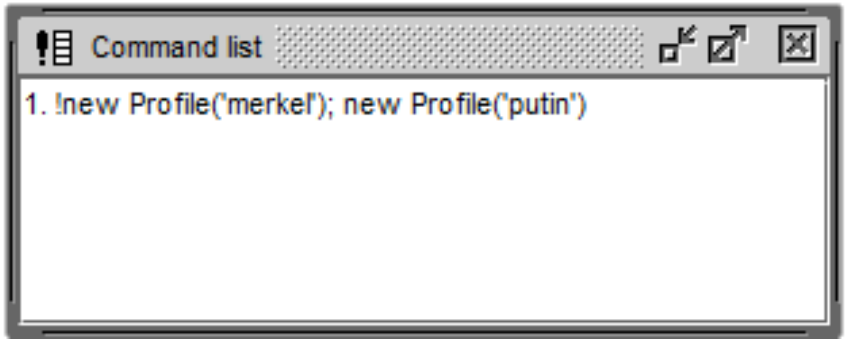
- Scenario: creation of two profiles and establishment of a friendship association link between them
- UML diagrams used: object, statechart, sequence and communication diagram together with USE command list
- Statechart diagrams together with object diagrams and USE command list
- Sequence diagrams together with USE command list
- Communication diagrams together with USE command list
- OCL queries on the USE shell

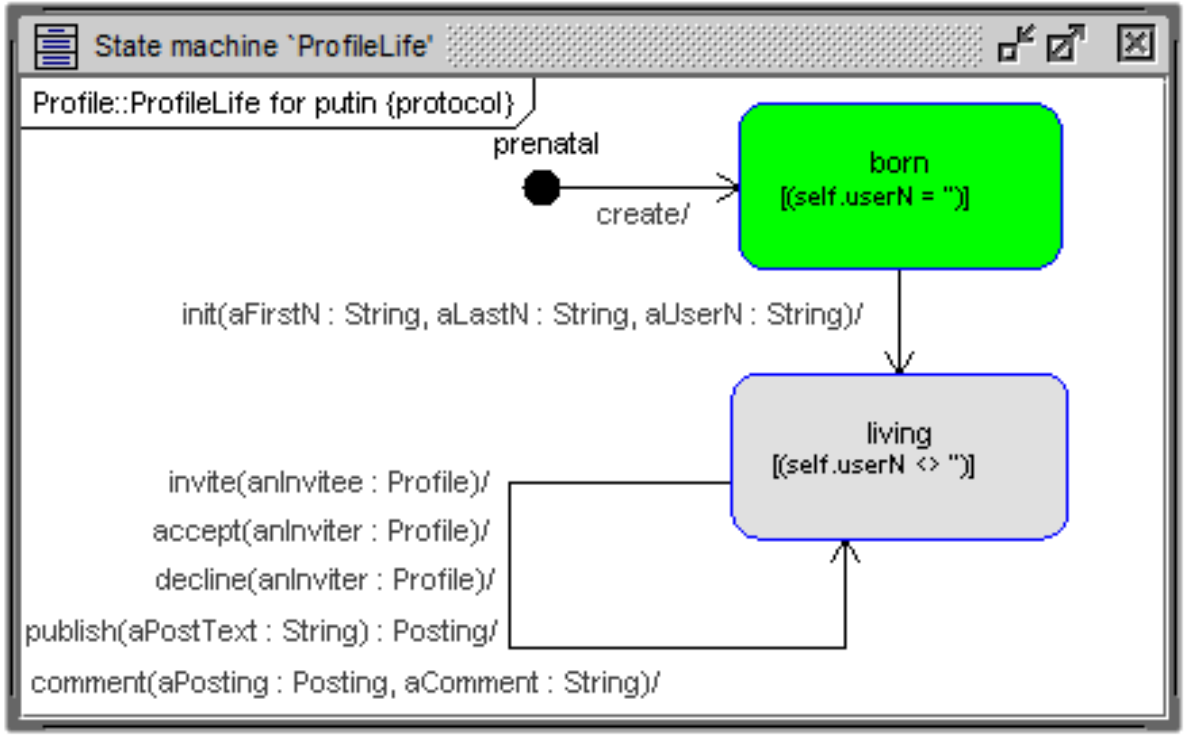
Animation and documentation of a simple scenario

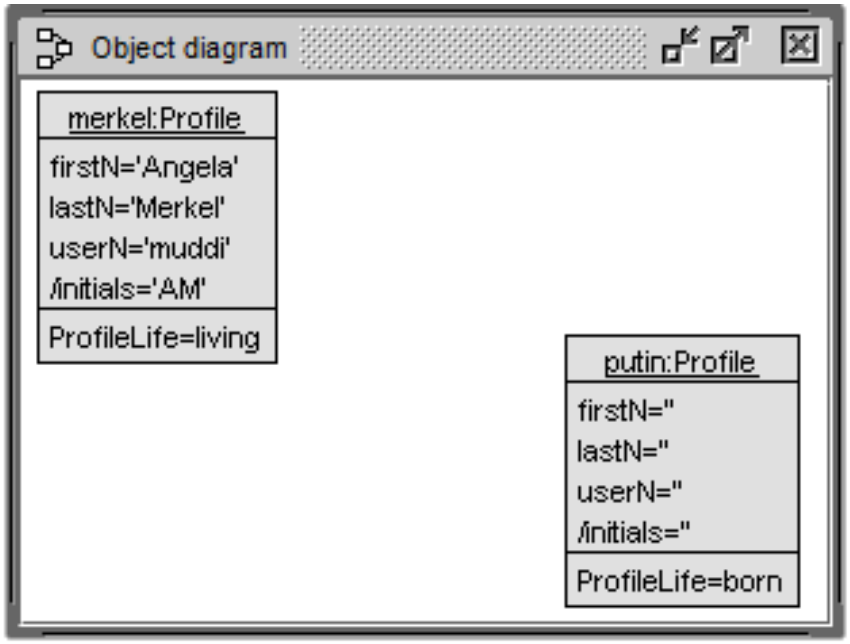
- Scenario: creation of two profiles and establishment of a friendship association link between them
- UML diagrams used: object, statechart, sequence and communication diagram together with USE command list
- Statechart diagrams together with object diagrams and USE command list
- Sequence diagrams together with USE command list
- Communication diagrams together with USE command list
- OCL queries on the USE shell

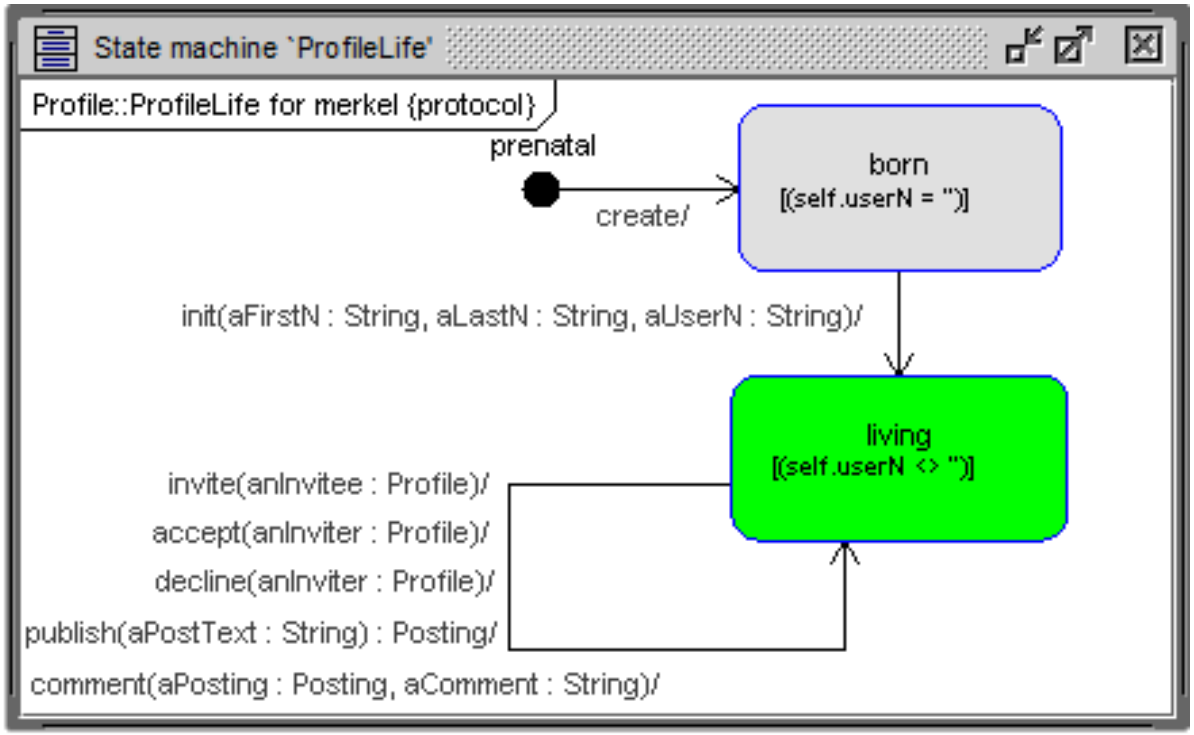

Object diagram
merkel:Profile
firstN=''
lastN=''
userN=''
/initials=''
ProfileLife=born
putin:Profile
firstN=''
lastN=''
userN=''
/initials=''
ProfileLife=born

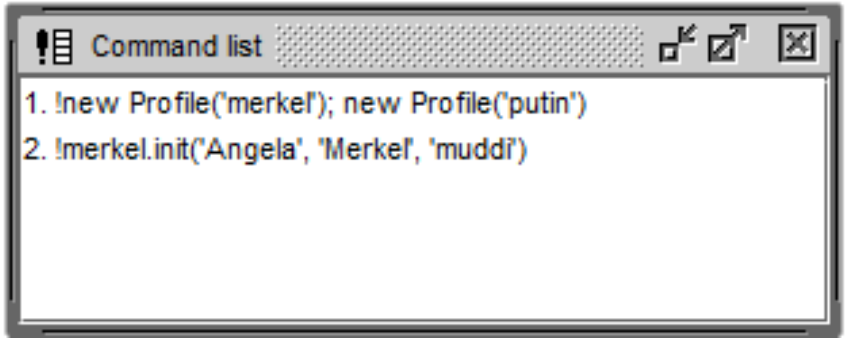

State machine `ProfileLife'
Profile::ProfileLife for merkel {protocol}
prenatal
create/
born
[(self.userN = '')]
init(aFirstN : String, aLastN : String, aUserN : String)/
living
[(self.userN <> '')]
invite(anInvitee : Profile)/
accept(anInviter : Profile)/
decline(anInviter : Profile)/
publish(aPostText : String) : Posting/
comment(aPosting : Posting, aComment : String)/

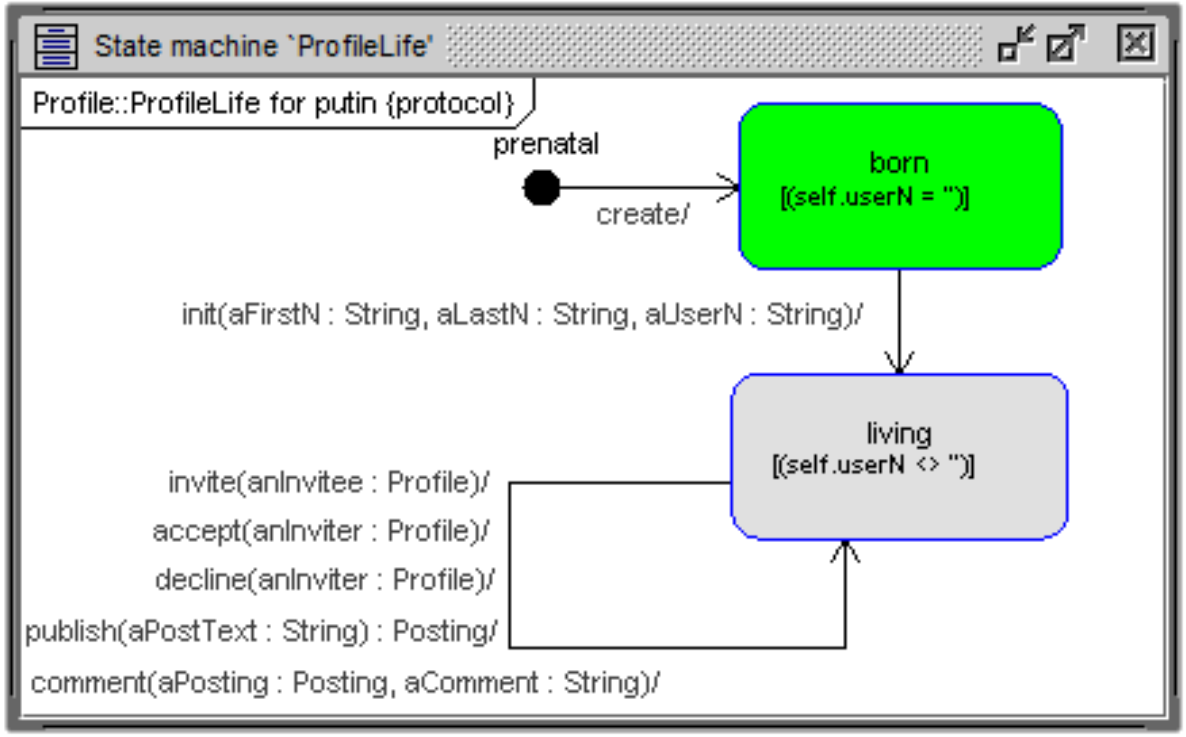

Command list
1. !new Profile('merkel'); new Profile('putin')

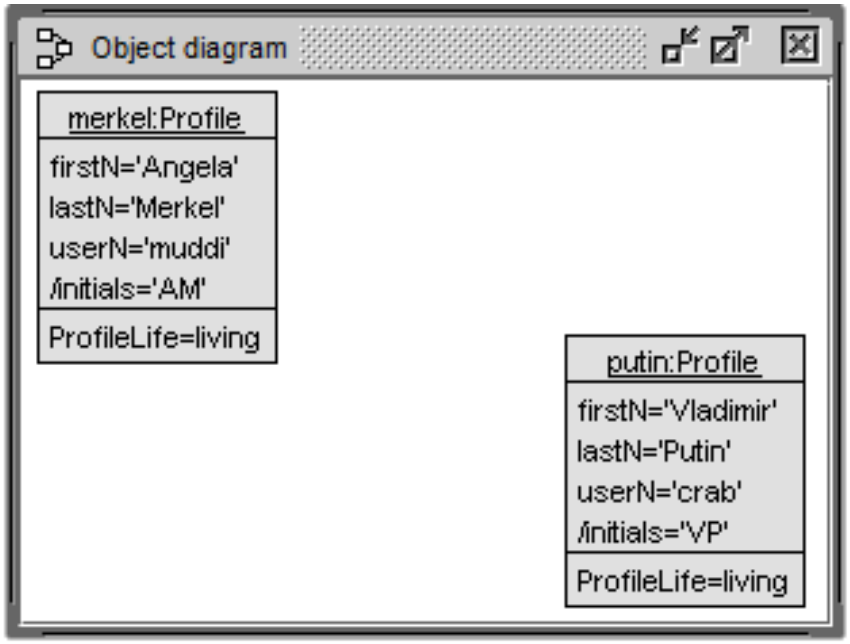

State machine `ProfileLife'
Profile::ProfileLife for putin {protocol}
prenatal
create/
born
[(self.userN = '')]
init(aFirstN : String, aLastN : String, aUserN : String)/
living
[(self.userN <> '')]
invite(anInvitee : Profile)/
accept(anInviter : Profile)/
decline(anInviter : Profile)/
publish(aPostText : String) : Posting/
comment(aPosting : Posting, aComment : String)/

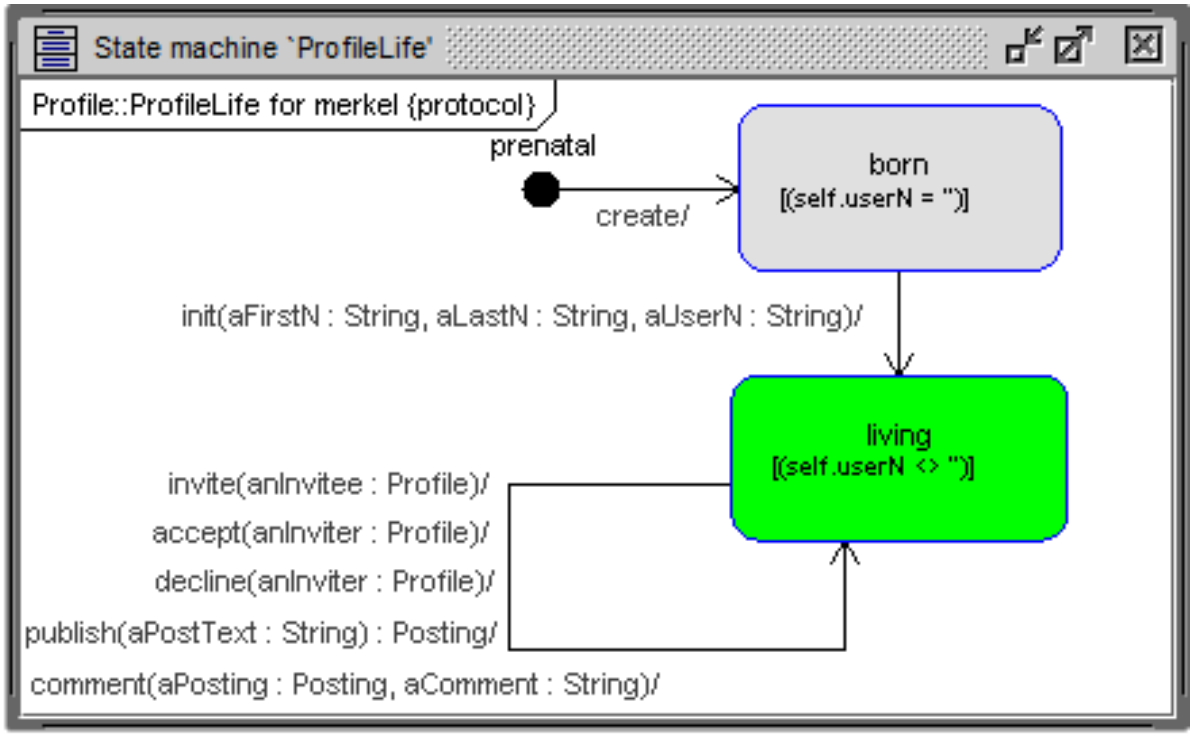

Object diagram
merkel:Profile
firstN='Angela'
lastN='Merkel'
userN='muddi'
/initials='AM'
ProfileLife=living
putin:Profile
firstN=''
lastN=''
userN=''
/initials=''
ProfileLife=born

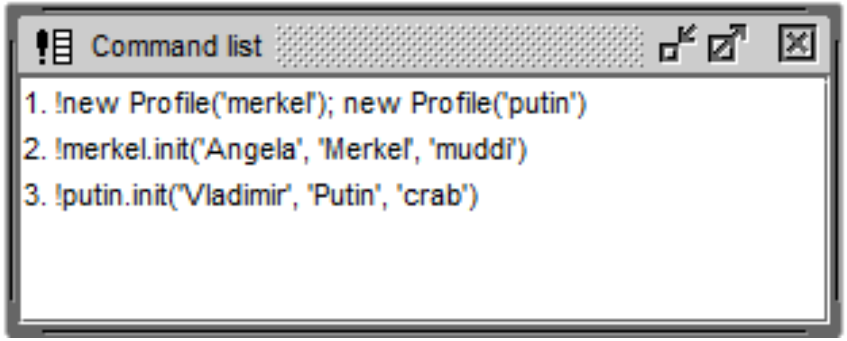

State machine `ProfileLife'
Profile::ProfileLife for merkel {protocol}
prenatal
create/
born
[(self.userN = '')]
init(aFirstN : String, aLastN : String, aUserN : String)/
living
[(self.userN <> '')]
invite(anInvitee : Profile)/
accept(anInviter : Profile)/
decline(anInviter : Profile)/
publish(aPostText : String) : Posting/
comment(aPosting : Posting, aComment : String)/

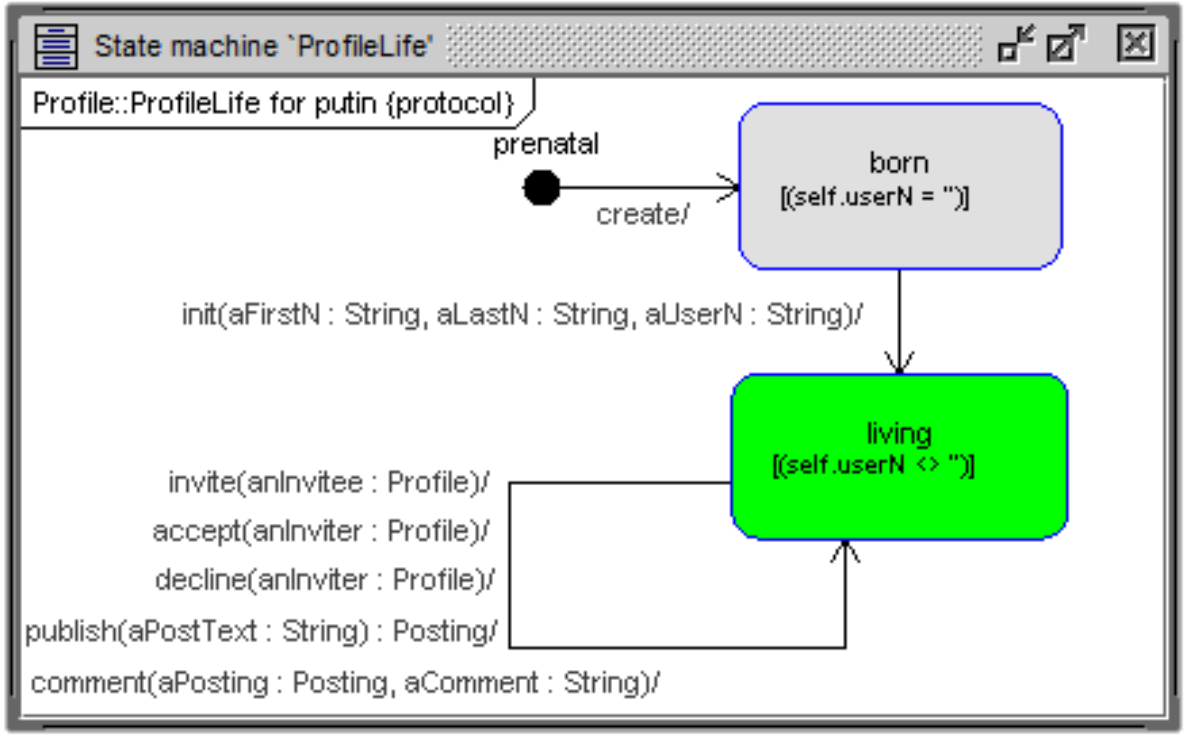

Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')

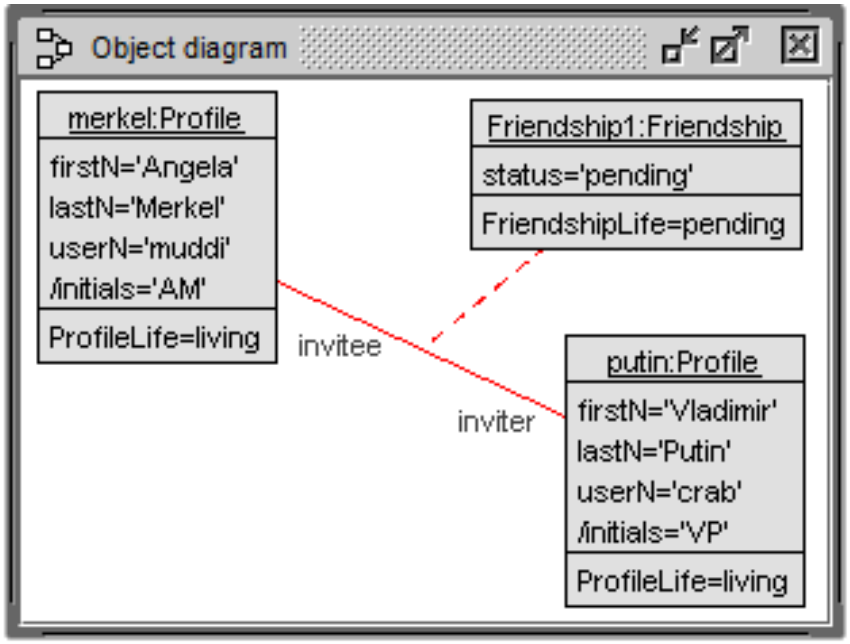

State machine `ProfileLife'
Profile::ProfileLife for putin {protocol}
prenatal
create/
born
[(self.userN = '')]
init(aFirstN : String, aLastN : String, aUserN : String)/
living
[(self.userN <> '')]
invite(anInvitee : Profile)/
accept(anInviter : Profile)/
decline(anInviter : Profile)/
publish(aPostText : String) : Posting/
comment(aPosting : Posting, aComment : String)/

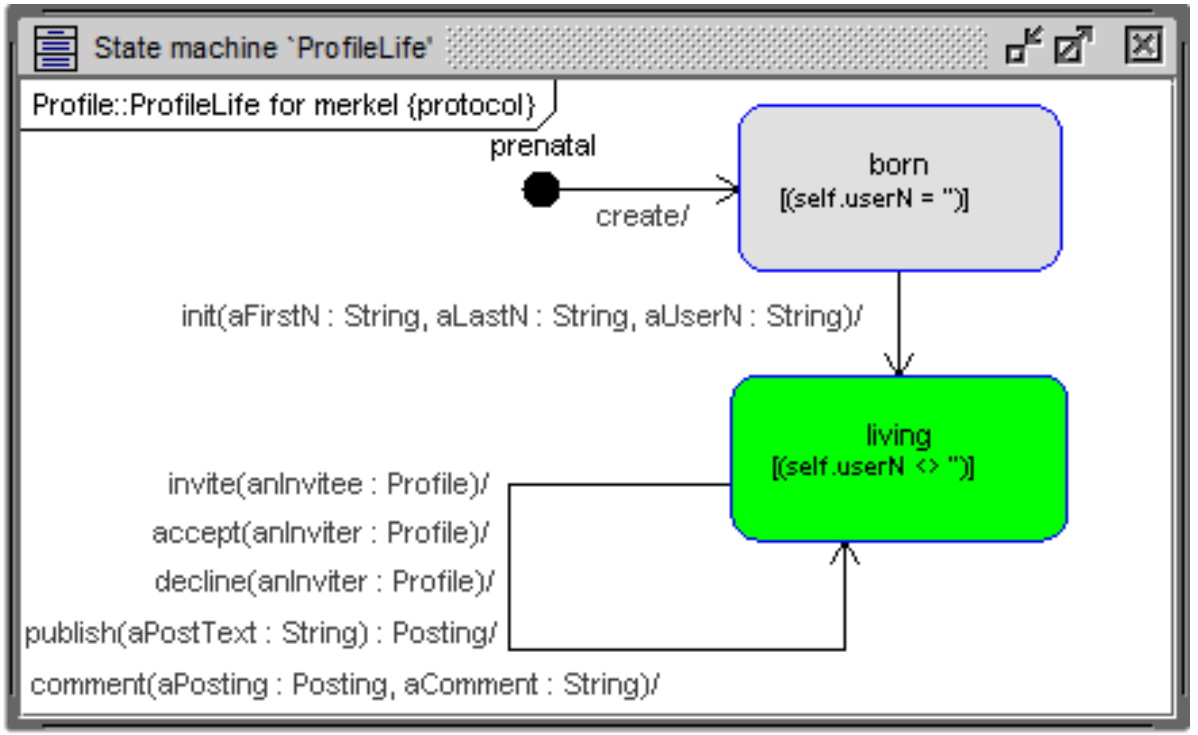

Object diagram
merkel:Profile
firstN='Angela'
lastN='Merkel'
userN='muddi'
/initials='AM'
ProfileLife=living
putin:Profile
firstN='Vladimir'
lastN='Putin'
userN='crab'
/initials='VP'
ProfileLife=living

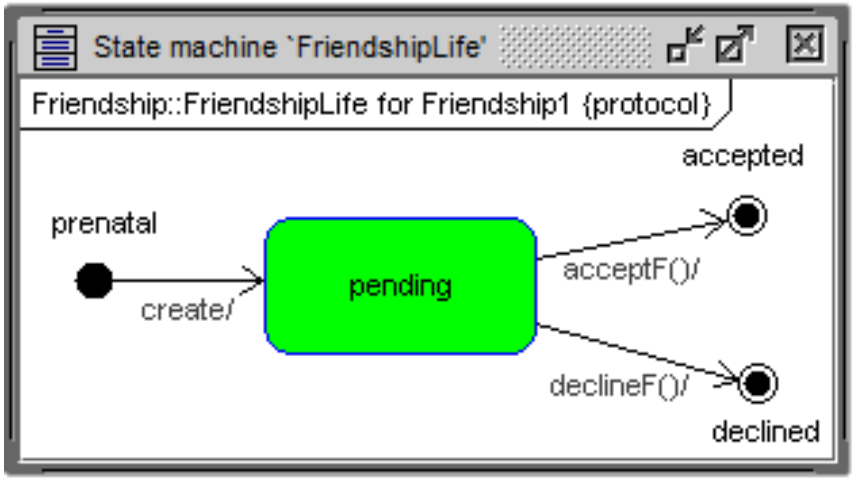

State machine `ProfileLife'
Profile::ProfileLife for merkel {protocol}
prenatal
create/
born
[(self.userN = '')]
init(aFirstN : String, aLastN : String, aUserN : String)/
living
[(self.userN <> '')]
invite(anInvitee : Profile)/
accept(anInviter : Profile)/
decline(anInviter : Profile)/
publish(aPostText : String) : Posting/
comment(aPosting : Posting, aComment : String)/

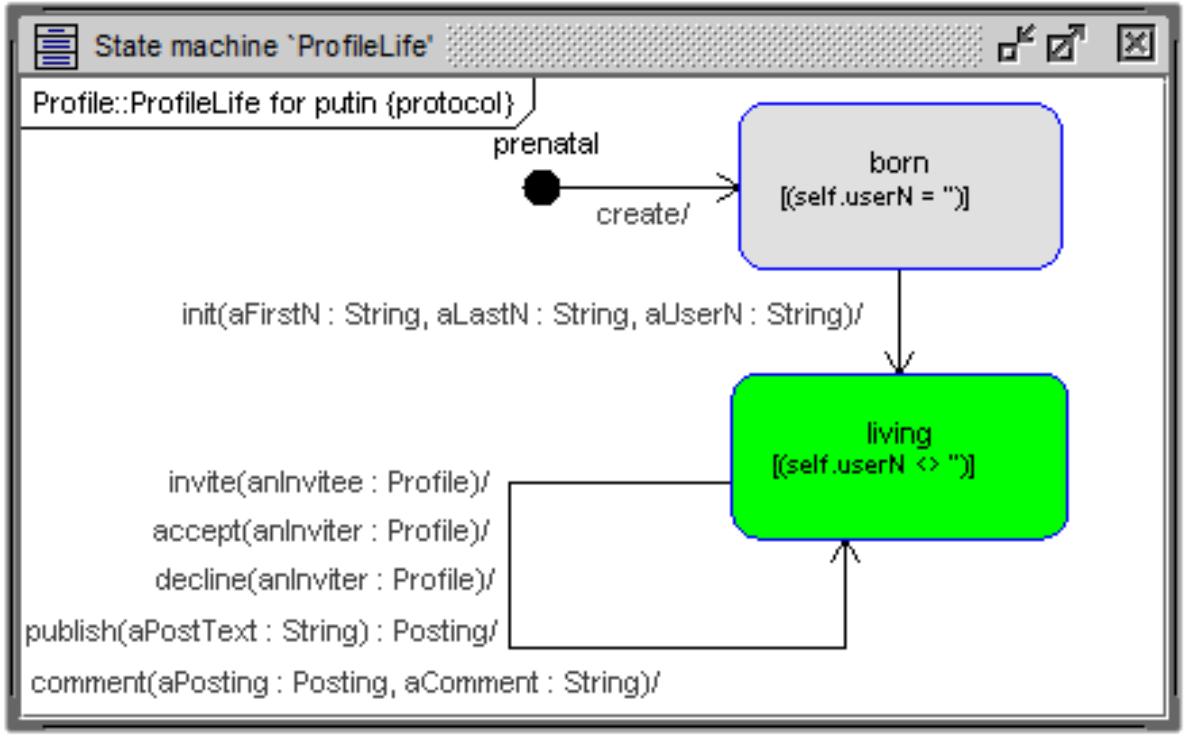

Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')
3. !putin.init('Vladimir', 'Putin', 'crab')

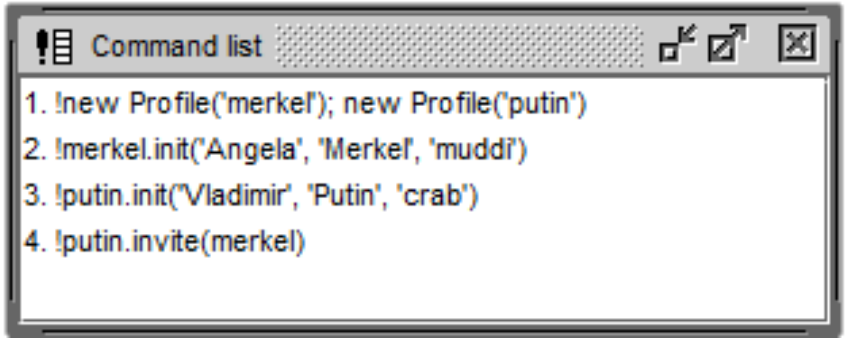

State machine `ProfileLife'
Profile::ProfileLife for putin {protocol}
prenatal
create/
born
[(self.userN = '')]
init(aFirstN : String, aLastN : String, aUserN : String)/
living
[(self.userN <> '')]
invite(anInvitee : Profile)/
accept(anInviter : Profile)/
decline(anInviter : Profile)/
publish(aPostText : String) : Posting/
comment(aPosting : Posting, aComment : String)/

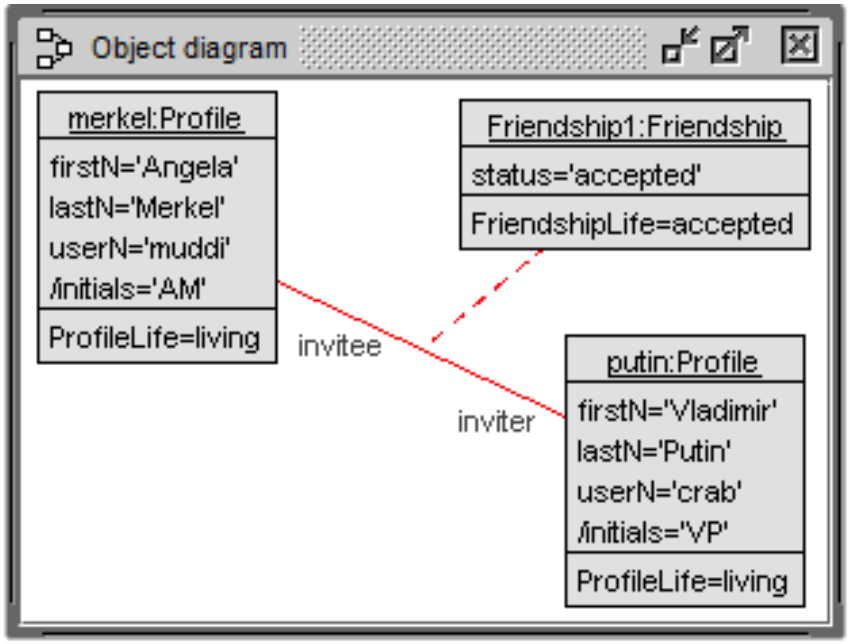

Object diagram
merkel:Profile
firstN='Angela'
lastN='Merkel'
userN='muddi'
/initials='AM'
ProfileLife=living
Friendship1:Friendship
status='pending'
FriendshipLife=pending
invitee
inviter
putin:Profile
firstN='Vladimir'
lastN='Putin'
userN='crab'
/initials='VP'
ProfileLife=living

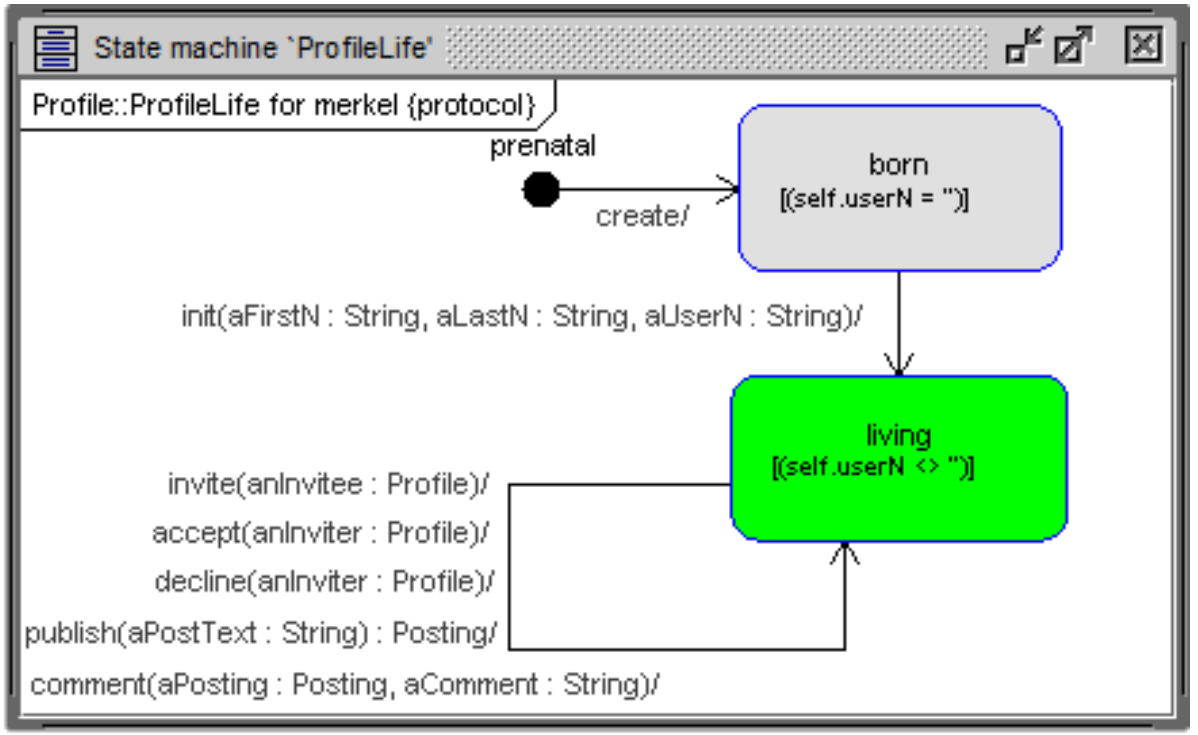

State machine `ProfileLife'
Profile::ProfileLife for merkel {protocol}
prenatal
create/
born
[(self.userN = '')]
init(aFirstN : String, aLastN : String, aUserN : String)/
living
[(self.userN <> '')]
invite(anInvitee : Profile)/
accept(anInviter : Profile)/
decline(anInviter : Profile)/
publish(aPostText : String) : Posting/
comment(aPosting : Posting, aComment : String)/

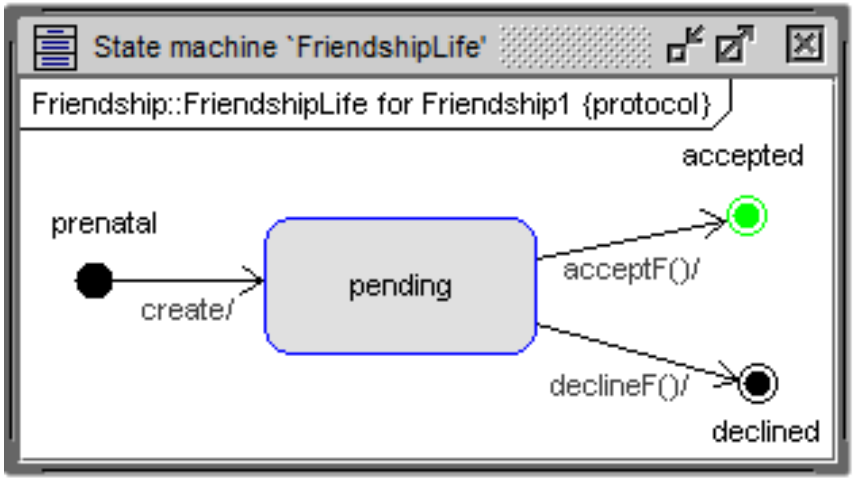

State machine `FriendshipLife'
Friendship::FriendshipLife for Friendship1 {protocol}
prenatal
create/
pending
acceptF()/
declineF()/
accepted
declined

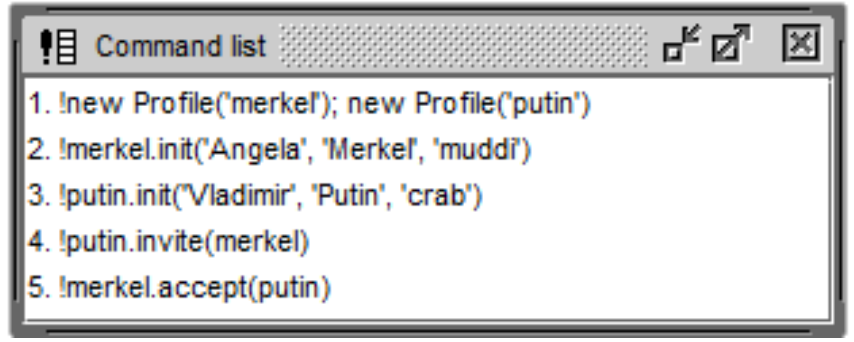

State machine `ProfileLife'
Profile::ProfileLife for putin {protocol}
prenatal
create/
born
[(self.userN = '')]
init(aFirstN : String, aLastN : String, aUserN : String)/
living
[(self.userN <> '')]
invite(anInvitee : Profile)/
accept(anInviter : Profile)/
decline(anInviter : Profile)/
publish(aPostText : String) : Posting/
comment(aPosting : Posting, aComment : String)/

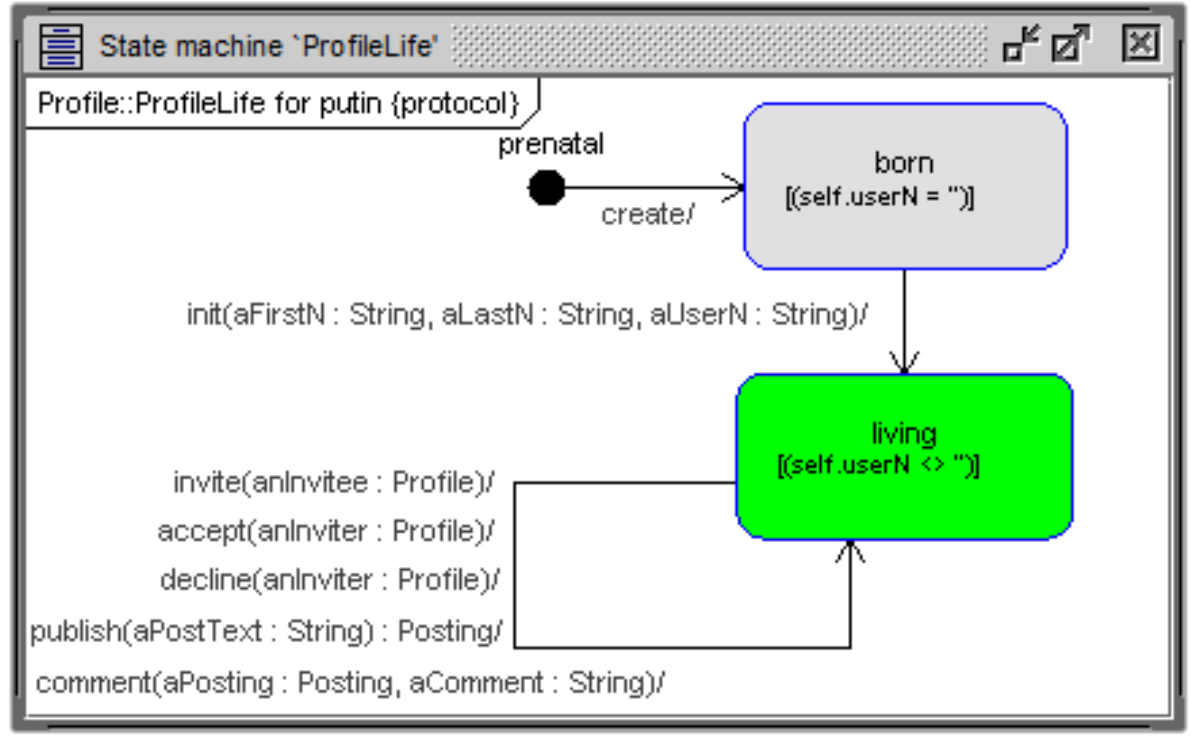

Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')
3. !putin.init('Vladimir', 'Putin', 'crab')
4. !putin.invite(merkel)


Object diagram
merkel:Profile
firstN='Angela'
lastN='Merkel'
userN='muddi'
/initials='AM'
ProfileLife=living
Friendship1:Friendship
status='accepted'
FriendshipLife=accepted
invitee
inviter
putin:Profile
firstN='Vladimir'
lastN='Putin'
userN='crab'
/initials='VP'
ProfileLife=living


State machine `ProfileLife'
Profile::ProfileLife for merkel {protocol}
prenatal
create/
born
[(self.userN = '')]
init(aFirstN : String, aLastN : String, aUserN : String)/
living
[(self.userN <> '')]
invite(anInvitee : Profile)/
accept(anInviter : Profile)/
decline(anInviter : Profile)/
publish(aPostText : String) : Posting/
comment(aPosting : Posting, aComment : String)/

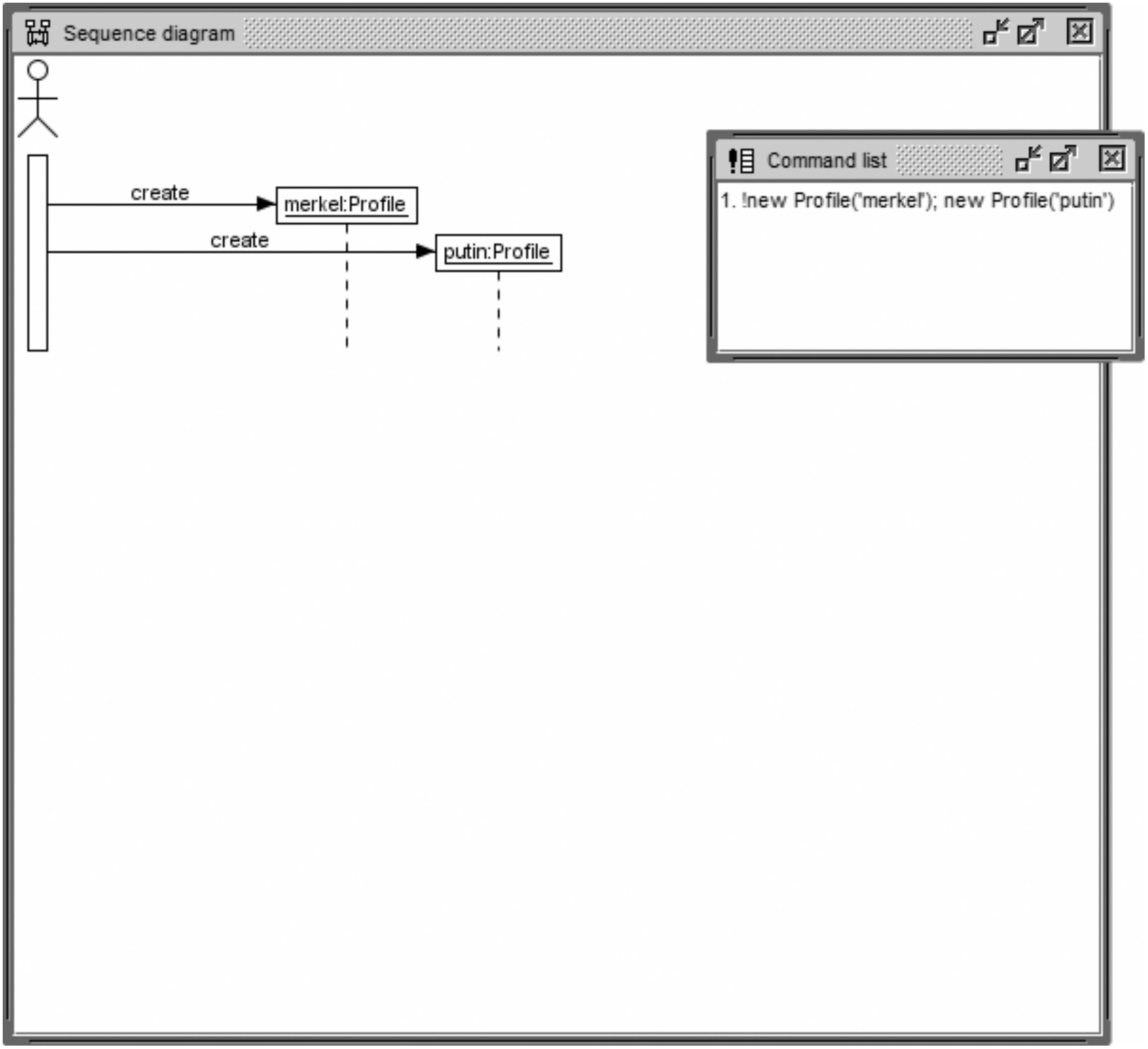

State machine `FriendshipLife'
Friendship::FriendshipLife for Friendship1 {protocol}
accepted
prenatal
create/
pending
acceptF()/
declineF()/
declined

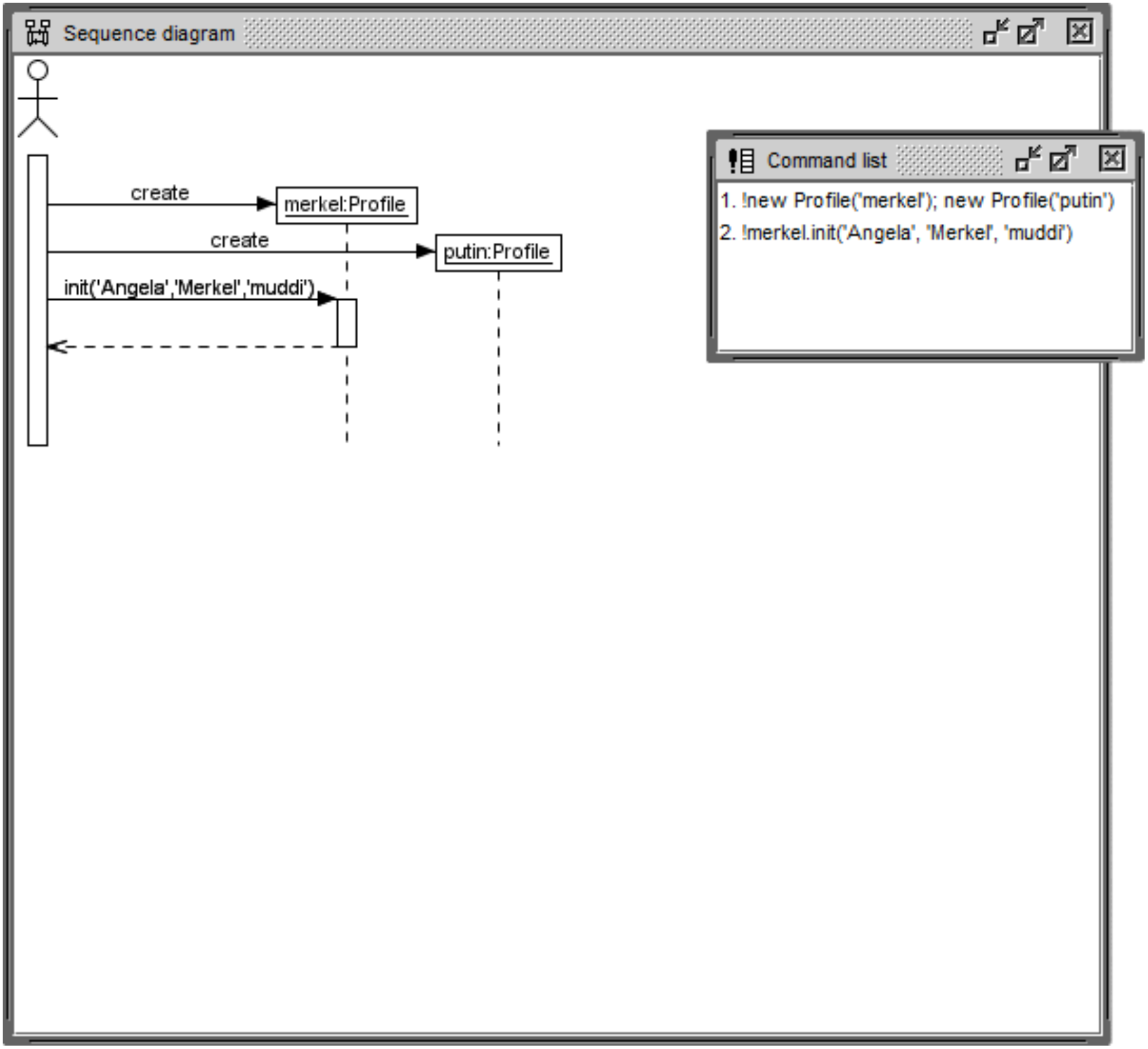

Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')
3. !putin.init('Vladimir', 'Putin', 'crab')
4. !putin.invite(merkel)
5. !merkel.accept(putin)

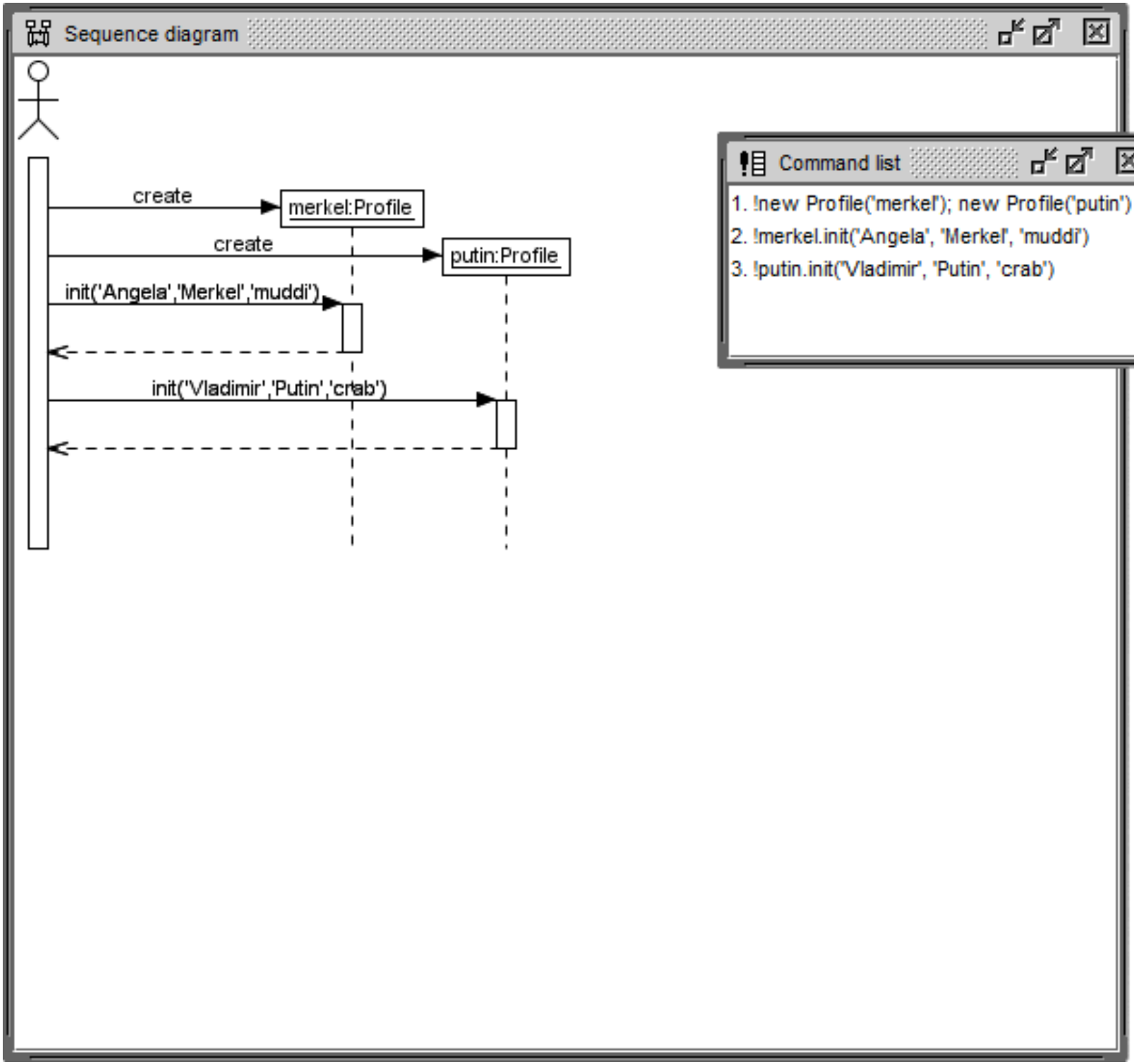

State machine `ProfileLife'
Profile::ProfileLife for putin {protocol}
prenatal
create/
born
[(self.userN = '')]
init(aFirstN : String, aLastN : String, aUserN : String)/
living
[(self.userN <> '')]
invite(anInvitee : Profile)/
accept(anInviter : Profile)/
decline(anInviter : Profile)/
publish(aPostText : String) : Posting/
comment(aPosting : Posting, aComment : String)/

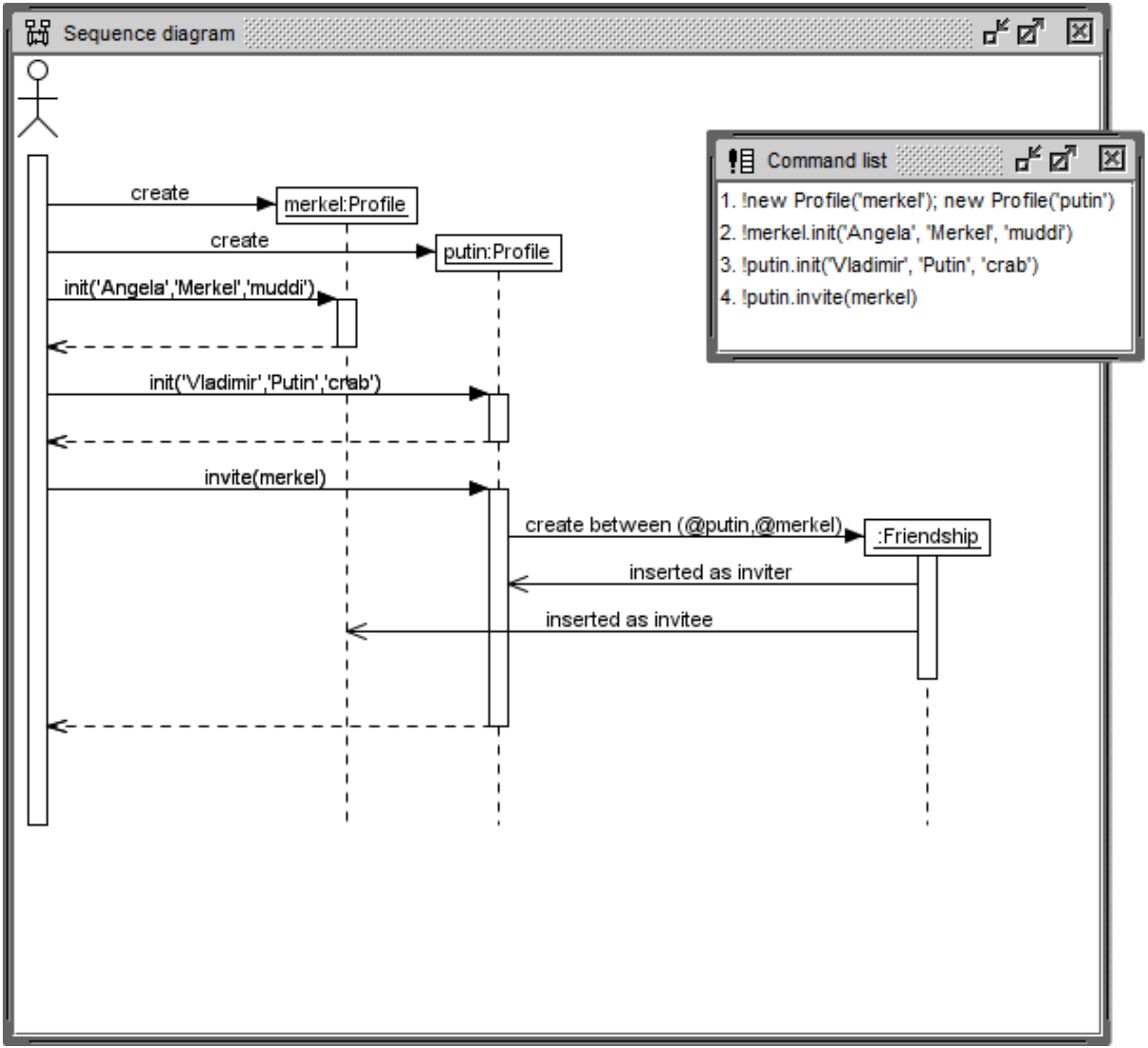
Animation and documentation of a simple scenario

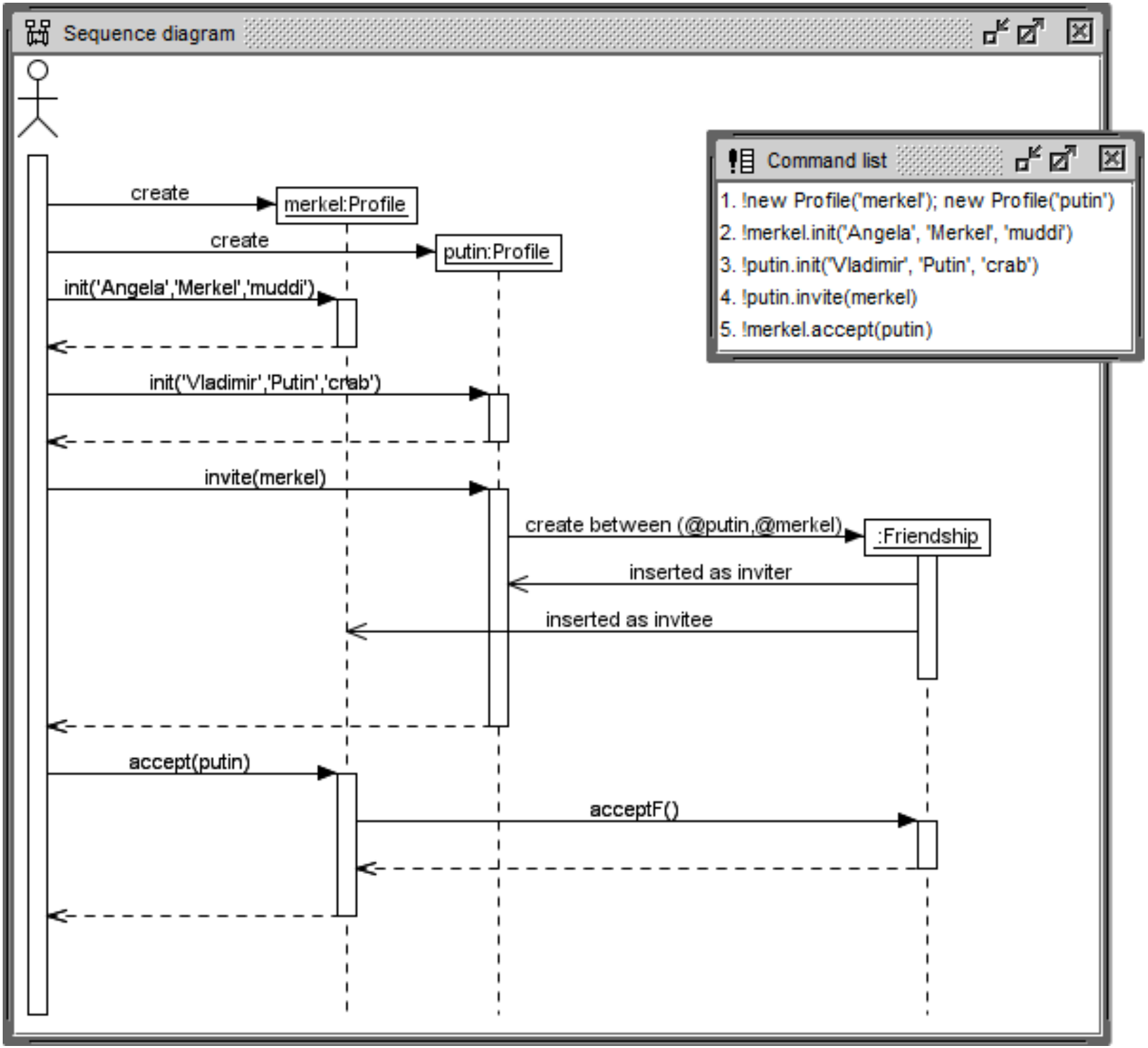
- Scenario: creation of two profiles and establishment of a friendship association link between them
- UML diagrams used: object, statechart, sequence and communication diagram together with USE command list
- Statechart diagrams together with object diagrams and USE command list
- Sequence diagrams together with USE command list
- Communication diagrams together with USE command list
- OCL queries on the USE shell


Sequence diagram
create
merkel:Profile
create
putin:Profile
Command list
1. !new Profile('merkel'); new Profile('putin')


Sequence diagram
create
merkel:Profile
create
putin:Profile
init('Angela','Merkel','muddi')
Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')

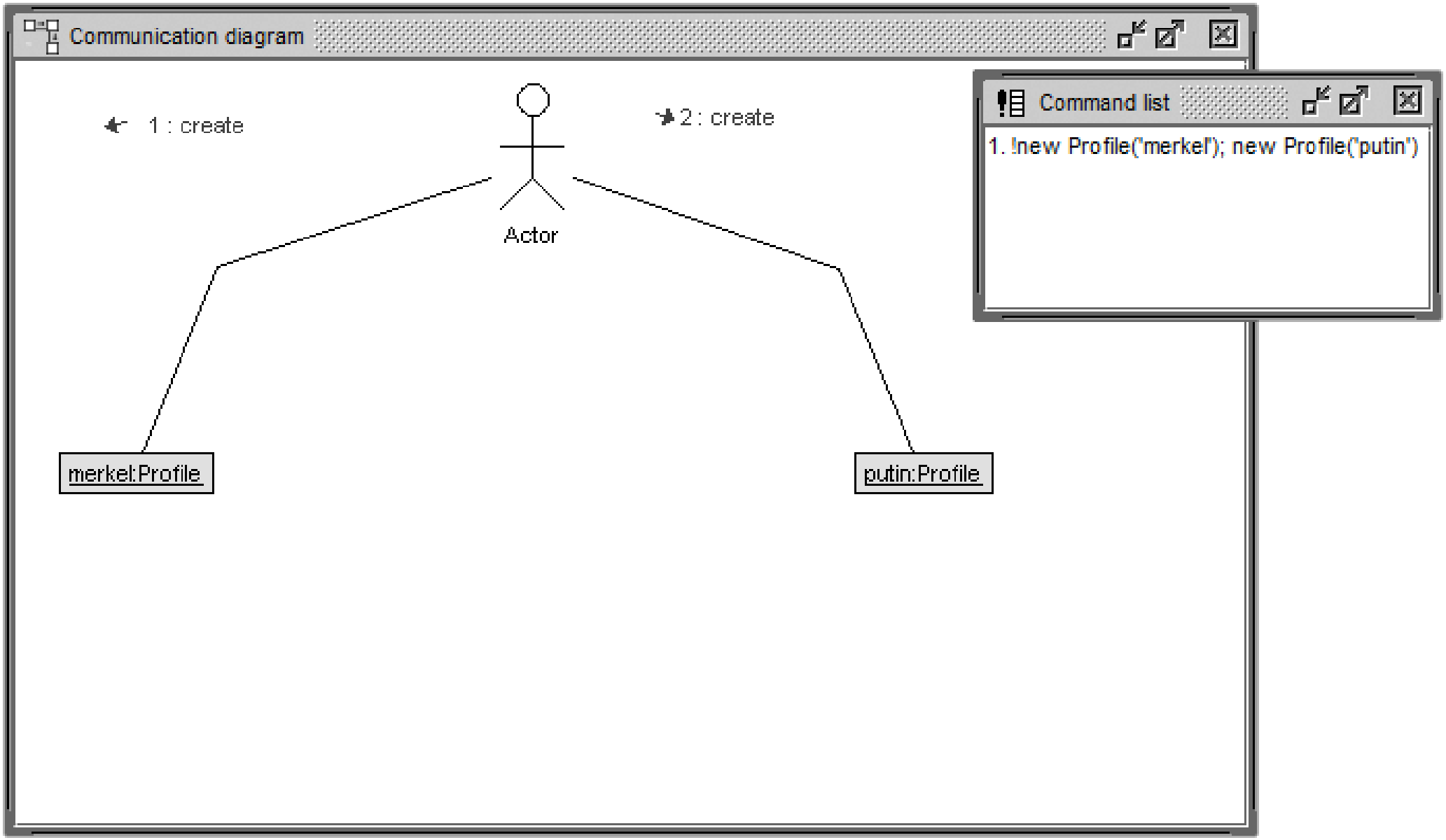

Sequence diagram
create
merkel:Profile
create
putin:Profile
init('Angela','Merkel','muddi')
init('Vladimir','Putin','crab')
Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')
3. !putin.init('Vladimir', 'Putin', 'crab')

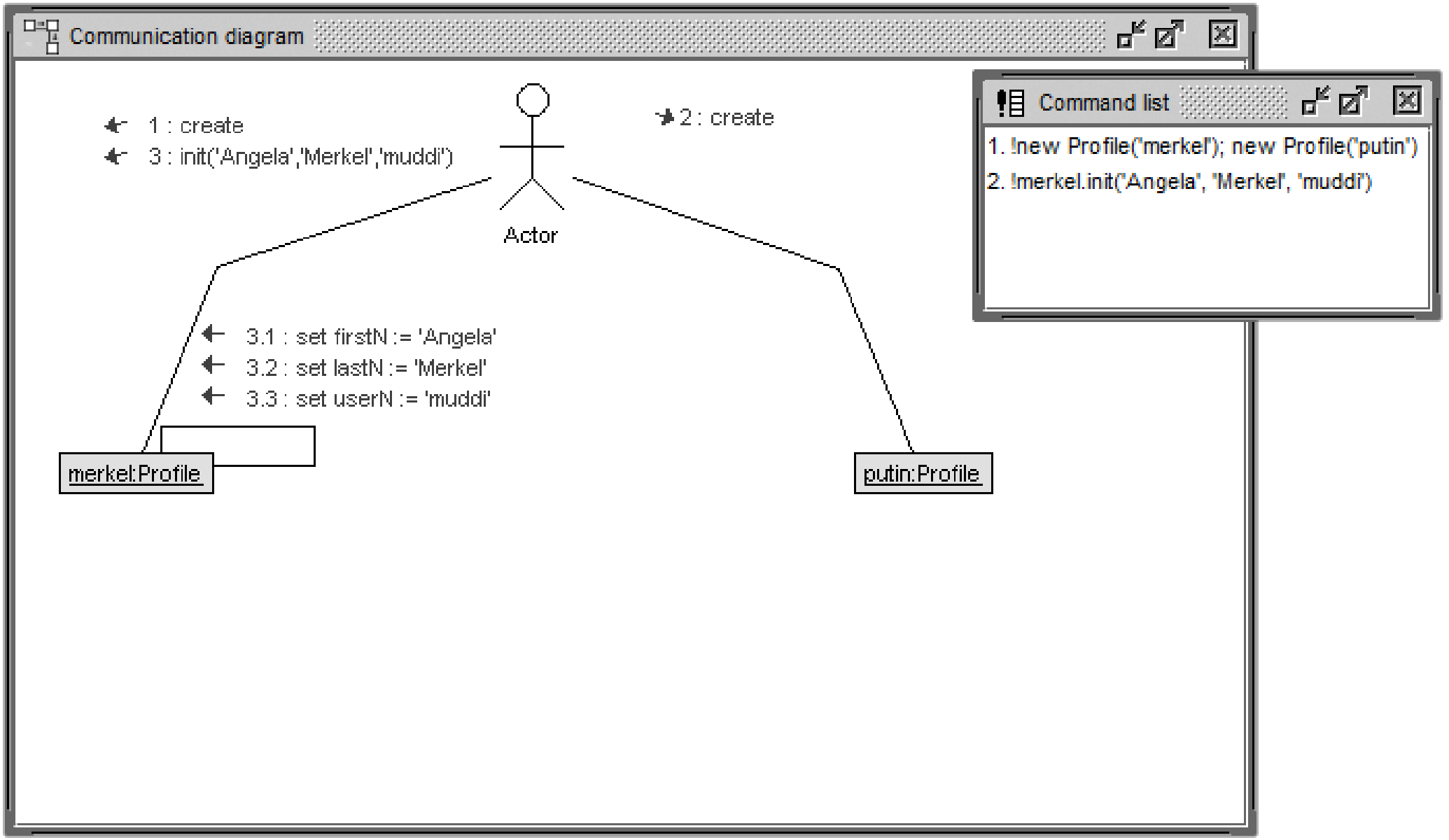

Sequence diagram
create
merkel:Profile
create
putin:Profile
init('Angela','Merkel','muddi')
init('Vladimir','Putin','crab')
invite(merkel)
create between (@putin,@merkel)
:Friendship
inserted as inviter
inserted as invitee
Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')
3. !putin.init('Vladimir', 'Putin', 'crab')
4. !putin.invite(merkel)

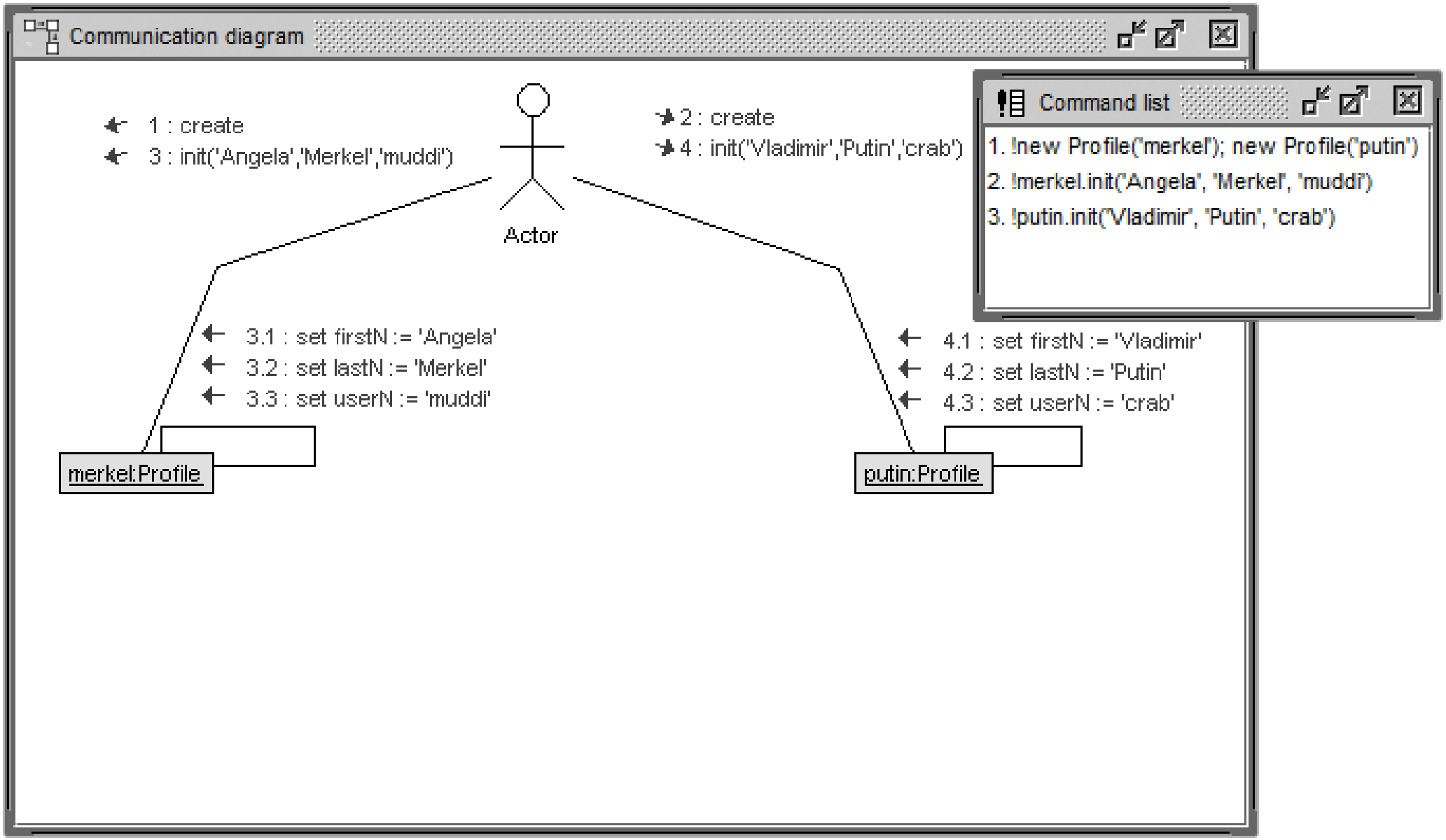

Sequence diagram
Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')
3. !putin.init('Vladimir', 'Putin', 'crab')
4. !putin.invite(merkel)
5. !merkel.accept(putin)
create
merkel:Profile
create
putin:Profile
init('Angela','Merkel','muddi')
init('Vladimir','Putin','crab')
invite(merkel)
create between (@putin,@merkel)
:Friendship
inserted as inviter
inserted as invitee
accept(putin)
acceptF()

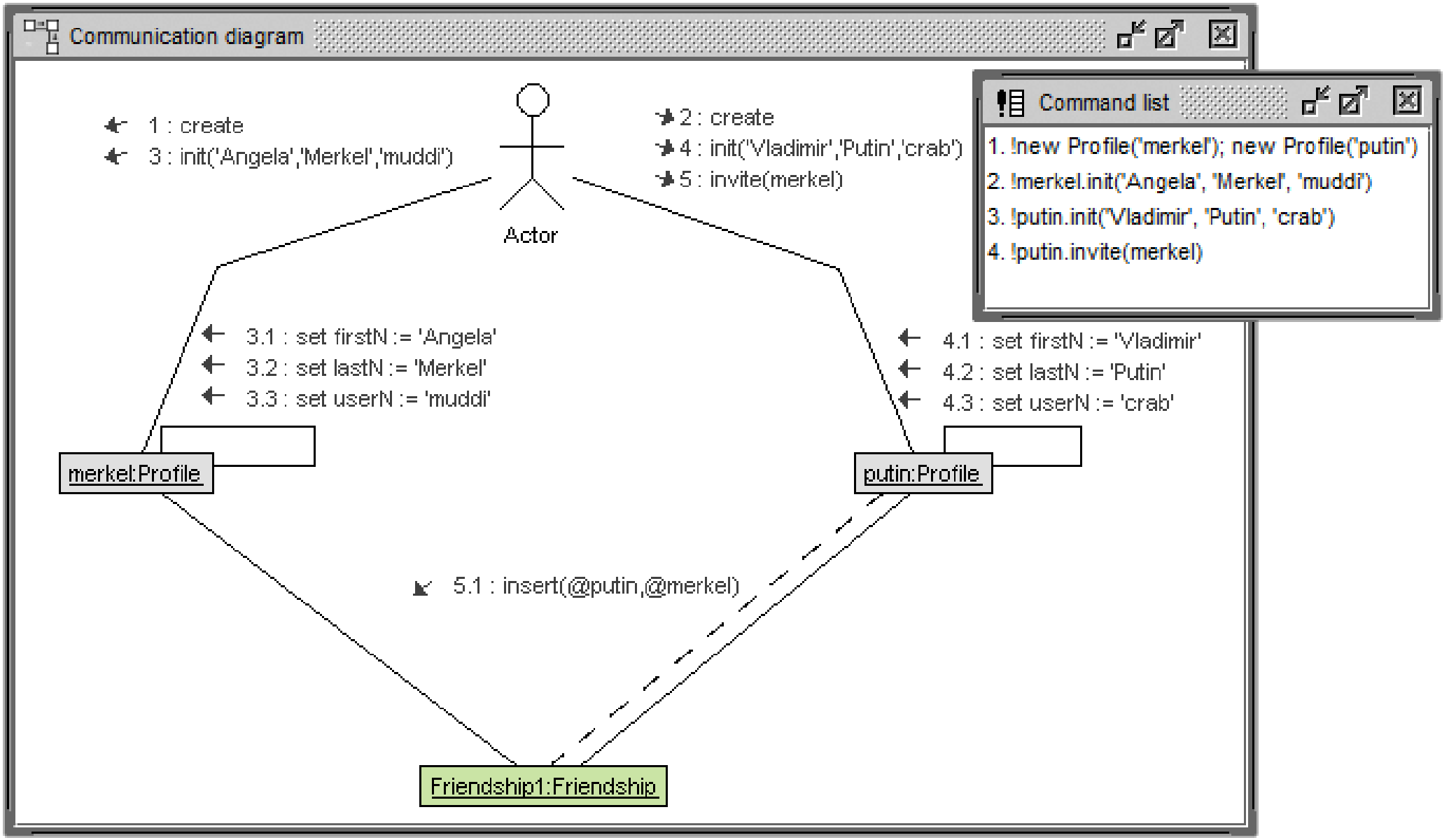
Animation and documentation of a simple scenario

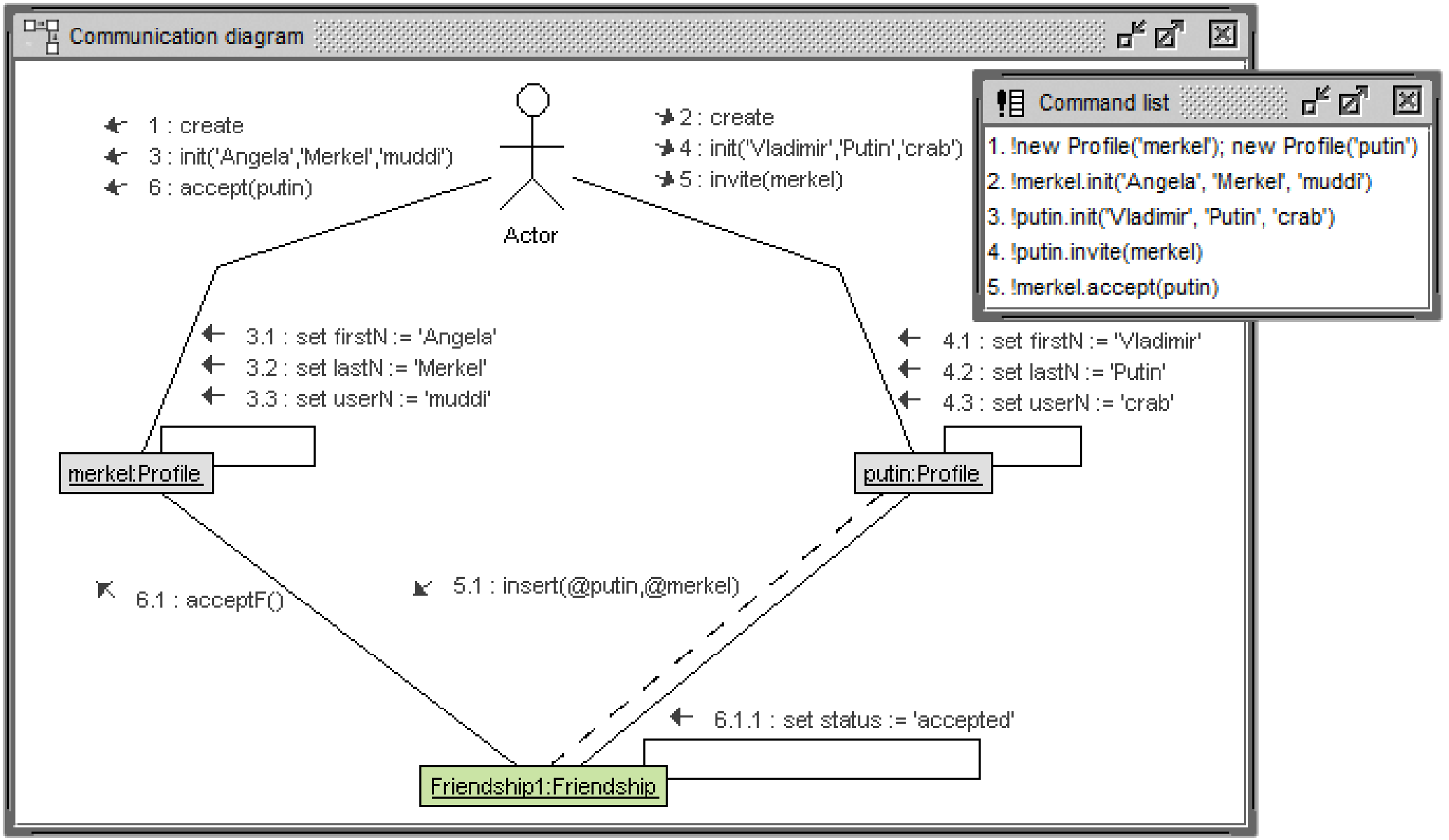
- Scenario: creation of two profiles and establishment of a friendship association link between them
- UML diagrams used: object, statechart, sequence and communication diagram together with USE command list
- Statechart diagrams together with object diagrams and USE command list
- Sequence diagrams together with USE command list
- Communication diagrams together with USE command list
- OCL queries on the USE shell


Communication diagram
1 : create
2 : create
Actor
merkel:Profile
putin:Profile
Command list
1. !new Profile('merkel'); new Profile('putin')


Communication diagram
1 : create
3 : init('Angela','Merkel','muddi')
2 : create
Actor
3.1 : set firstN := 'Angela'
3.2 : set lastN := 'Merkel'
3.3 : set userN := 'muddi'
merkel:Profile
putin:Profile
Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')


Communication diagram
1 : create
3 : init('Angela','Merkel','muddi')
2 : create
4 : init('Vladimir','Putin','crab')
Actor
3.1 : set firstN := 'Angela'
3.2 : set lastN := 'Merkel'
3.3 : set userN := 'muddi'
4.1 : set firstN := 'Vladimir'
4.2 : set lastN := 'Putin'
4.3 : set userN := 'crab'
merkel:Profile
putin:Profile
Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')
3. !putin.init('Vladimir', 'Putin', 'crab')


Communication diagram
1 : create
3 : init('Angela','Merkel','muddi')
2 : create
4 : init('Vladimir','Putin','crab')
5 : invite(merkel)
Actor
3.1 : set firstN := 'Angela'
3.2 : set lastN := 'Merkel'
3.3 : set userN := 'muddi'
4.1 : set firstN := 'Vladimir'
4.2 : set lastN := 'Putin'
4.3 : set userN := 'crab'
merkel:Profile
putin:Profile
5.1 : insert(@putin,@merkel)
Friendship1:Friendship
Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')
3. !putin.init('Vladimir', 'Putin', 'crab')
4. !putin.invite(merkel)


Communication diagram
1 : create
3 : init('Angela','Merkel','muddi')
6 : accept(putin)
2 : create
4 : init('Vladimir','Putin','crab')
5 : invite(merkel)
Actor
Command list
1. !new Profile('merkel'); new Profile('putin')
2. !merkel.init('Angela', 'Merkel', 'muddi')
3. !putin.init('Vladimir', 'Putin', 'crab')
4. !putin.invite(merkel)
5. !merkel.accept(putin)
3.1 : set firstN := 'Angela'
3.2 : set lastN := 'Merkel'
3.3 : set userN := 'muddi'
4.1 : set firstN := 'Vladimir'
4.2 : set lastN := 'Putin'
4.3 : set userN := 'crab'
merkel:Profile
putin:Profile
6.1 : acceptF()
5.1 : insert(@putin,@merkel)
6.1.1 : set status := 'accepted'
Friendship1:Friendship

Animation and documentation of a simple scenario

- Scenario: creation of two profiles and establishment of a friendship association link between them
- UML diagrams used: object, statechart, sequence and communication diagram together with USE command list
- Statechart diagrams together with object diagrams and USE command list
- Sequence diagrams together with USE command list
- Communication diagrams together with USE command list
- OCL queries on the USE shell

Scenario documentation with OCL queries


```
! create merkel,putin:Profile
? Profile.allInstances->collect(p|Tuple{PF:p,UN:p.userN})
  Bag{Tuple{PF=merkel,UN=''},Tuple{PF=putin,UN=''}}
? Friendship.allInstances->collect(f|Tuple{FS:f,ST:f.status})
  Bag{}
```

Scenario documentation with OCL queries


```
! create merkel,putin:Profile
? Profile.allInstances->collect(p|Tuple{PF:p,UN:p.userN})
  Bag{Tuple{PF=merkel,UN=''},Tuple{PF=putin,UN=''}}
? Friendship.allInstances->collect(f|Tuple{FS:f,ST:f.status})
  Bag{}

! merkel.init('Angela','Merkel','muddi')
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN=''}}
  Bag{}
```

Scenario documentation with OCL queries


```
! create merkel,putin:Profile
? Profile.allInstances->collect(p|Tuple{PF:p,UN:p.userN})
  Bag{Tuple{PF=merkel,UN=''},Tuple{PF=putin,UN=''}}
? Friendship.allInstances->collect(f|Tuple{FS:f,ST:f.status})
  Bag{}

! merkel.init('Angela','Merkel','muddi')
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN=''}}
  Bag{}

! putin.init('Vladimir','Putin','crab')
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN='crab'}}
  Bag{}
```

Scenario documentation with OCL queries


```
! create merkel,putin:Profile
? Profile.allInstances->collect(p|Tuple{PF:p,UN:p.userN})
  Bag{Tuple{PF=merkel,UN=''},Tuple{PF=putin,UN=''}}
? Friendship.allInstances->collect(f|Tuple{FS:f,ST:f.status})
  Bag{}

! merkel.init('Angela','Merkel','muddi')
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN=''}}
  Bag{}

! putin.init('Vladimir','Putin','crab')
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN='crab'}}
  Bag{}

! putin.invite(merkel)
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN='crab'}}
  Bag{Tuple{FS=Friendship1,ST='pending'}}
```

Scenario documentation with OCL queries


```
! create merkel,putin:Profile
? Profile.allInstances->collect(p|Tuple{PF:p,UN:p.userN})
  Bag{Tuple{PF=merkel,UN=''},Tuple{PF=putin,UN=''}}
? Friendship.allInstances->collect(f|Tuple{FS:f,ST:f.status})
  Bag{}

! merkel.init('Angela','Merkel','muddi')
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN=''}}
  Bag{}

! putin.init('Vladimir','Putin','crab')
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN='crab'}}
  Bag{}

! putin.invite(merkel)
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN='crab'}}
  Bag{Tuple{FS=Friendship1,ST='pending'}}

! merkel.accept(putin)
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN='crab'}}
  Bag{Tuple{FS=Friendship1,ST='accepted'}}
```

Scenario documentation with OCL queries


```
! create merkel,putin:Profile
? Profile.allInstances->collect(p|Tuple{PF:p,UN:p.userN})
  Bag{Tuple{PF=merkel,UN=''},Tuple{PF=putin,UN=''}}
? Friendship.allInstances->collect(f|Tuple{FS:f,ST:f.status})
  Bag{}

! merkel.init('Angela','Merkel','muddi')
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN=''}}
  Bag{}

! putin.init('Vladimir','Putin','crab')
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN='crab'}}
  Bag{}

! putin.invite(merkel)
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN='crab'}}
  Bag{Tuple{FS=Friendship1,ST='pending'}}

! merkel.accept(putin)
  Bag{Tuple{PF=merkel,UN='muddi'},Tuple{PF=putin,UN='crab'}}
  Bag{Tuple{FS=Friendship1,ST='accepted'}}

? Friendship.allInstances->collect(f|
   Tuple{TER:f.inviter.userN,TEE:f.invitee.userN,ST:f.status})
  Bag{Tuple{TER='crab',TEE='muddi',ST='accepted'}}
```

Animation and documentation of a simple scenario

- Scenario: creation of two profiles and establishment of a friendship association link between them
- UML diagrams used: object, statechart, sequence and communication diagram together with USE command list
- Statechart diagrams together with object diagrams and USE command list
- Sequence diagrams together with USE command list
- Communication diagrams together with USE command list
- OCL queries on the USE shell

Thanks for your attention!