Variations for USE model definition and USE shell commands

- Textual USE model definition [repeated]
- Variation: `implementation` Profile::publish(...)::Posting
- Variation: `postcondition` Profile::comment(...)
- Variation: `invariant` Profile::uniqueUserName
- Variation: `attribute` Friendship::status
- Variation: `composition` PosterPosting - multiplicity
- Variation: `association` Interest - role names
- Variation: `invariant` outside class - `inside` class
- Variation: `shell` variables - `shell` queries
model SocialNetwork

class Profile
attributes
  firstName:String init: ''
  lastName:String init: ''
  userN:String init: ''
  initials:String derived:
    firstName.substring(1,1).concat(lastName.substring(1,1))
operations
  init(aFirstName:String, aLastName:String, aUserN:String)
    begin
    self.firstName:=aFirstName; self.lastName:=aLastName; self.userN:=aUserN end
  pre aUserNNonEmpty: aUserN<>''
  post userNAssigned: aUserN=userN
  invite(anInvitee:Profile)
    begin
    new Friendship between (self,anInvitee) end
  pre notAlreadyTried: invitee->union(inviter)->excludes(anInvitee)
  post madeFS: friendship[inviter]->
    select(oclInState(pending)).invitee->includes(anInvitee)
  accept(anInviter:Profile)
    begin
    self.friendship(anInviter).acceptF() end
  pre pendingFS: friendship[inviter]->
    select(oclInState(pending)).inviter->includes(anInviter)
  post acceptedFS: friendship[inviter]->
    select(oclInState(accepted)).inviter->includes(anInviter)
Textual model definition in USE (complete model part B)

decline(anInviter:Profile)
   begin self.friendship(anInviter).declineF() end
   pre  pendingFS: friendship[invitee]->
       select(oclInState(pending)).inviter->includes(anInviter)
   post declinedFS: friendship[invitee]->
       select(oclInState(declined)).inviter->includes(anInviter)

publish(aPostText:String):Posting
   begin declare p:Posting;
       p:=new Posting(); p.posting:=aPostText;
       insert(self,p) into PosterPosting; result:=p
   end
   pre  nonEmpty: aPostText<>''
   post newPosting: Posting.allInstances->exists(p |
       p.posting=aPostText and result=p)

coment(aPosting:Posting,aComment:String)
   begin declare c:Commenting;
       c:=new Commenting between (self,aPosting); c.comment:=aComment
   end
   pre  aPostingNonNullACommentNonEmpty: aPosting<>null and aComment<>''
   post commentingExists: Commenting.allInstances->exists(c |
       c.comment=aComment and aPosting.commenting->includes(c) and
       self.commenting->includes(c))
Textual model definition in USE (complete model part C)

friends():Set(Profile)=
  friendship[inviter]->select(oclInState(accepted)).invitee->union(
    friendship[invitee]->select(oclInState(accepted)).inviter)->asSet()
friendship(anInviter:Profile):Friendship=
  self.friendship[invitee]->any(fs|fs.inviter=anInviter)

constraints
inv asymmetricFriendship: invitee->intersection(inviter)->isEmpty()
inv uniqueUserName: Profile.allInstances->isUnique(userN)

statemachines
psm ProfileLife
states
  prenatal:initial
  born   [userN='']
  living [userN<>'']
transitions
  prenatal -> born   { create }
  born     -> living { init() }
  living   -> living { invite() }
  living   -> living { accept() }
  living   -> living { decline() }
  living   -> living { publish() }
  living   -> living { comment() }
end
Textual model definition in USE (complete model part D)

associationclass Friendship between
  Profile [*] role inviter
  Profile [*] role invitee
attributes
  status:String init:'pending'
operations
  acceptF()
    begin self.status:='accepted' end
  declineF()
    begin self.status:='declined' end
statemachines
  psm FriendshipLife
  states
    prenatal:initial
    pending
    accepted:final
    declined:final
  transitions
    prenatal -> pending { create }
    pending  -> accepted { acceptF() }
    pending  -> declined { declineF() }
end
end
Textual model definition in USE (complete model part E)

composition PosterPosting between
  Profile [1] role poster
  Posting [*] role posting
end

class Posting
attributes
  posting: String
end

associationclass Commenting between
  Profile [*] role commenter
  Posting [*] role commented
attributes
  comment: String
end

constraints

context Commenting inv commentOnlyByFriends:
  commented.poster.friends() -> includes(commenter)
Textual model definition in USE (complete model part F)

class Subject
attributes
  subject: String
constraints
  inv noDuplicates:
    Subject.allInstances->size=Subject.allInstances.subject->asSet->size
end

association Interest between
  Profile [*]
  Subject [*]
end
publish(aPostText:String):Posting
begin
declare p:Posting;
p:=new Posting();
p.posting:=aPostText;
insert(self,p) into PosterPosting;
result:=p
end

publish(aPostText:String):Posting
begin
declare p:Posting;
p:=Posting.allInstances->any(p|p.posting=aPostText);
if p=null then
  p:=new Posting();
  p.posting:=aPostText;
  insert(self,p) into PosterPosting;
end;
result:=p;
end

pre  nonEmpty: aPostText<>''
post newPosting: Posting.allInstances->exists(p |
    p.posting=aPostText and result=p)
Textual model definition in USE - Variations B

class Profile
...

comment(aPosting:Posting,aComment:String)
begin
  declare c:Commenting;
  c:=new Commenting between (self,aPosting);
  c.comment:=aComment
end

pre  aPostingNonNullACommentNonEmpty: aPosting<>null and aComment<>''

post commentingExists: Commenting.allInstances->exists(c |
  c.comment=aComment and
  aPosting.commenting->includes(c) and
  self.commenting->includes(c))

post commentingExists: Commenting.allInstances->exists(c |
  c.comment=aComment and
  c.commented=aPosting and
  c.commenter=self)
class Profile
...

Inv uniqueUserName:
Profile.allInstances->isUnique(userN)

Inv uniqueUserName:
Profile.allInstances->select(p | p.userN=self.userN)->size=1

Inv uniqueUserName:
Profile.allInstances->one(p | p.userN=self.userN)

Inv uniqueUserName:
Profile.allInstances->reject(p | p.userN<>self.userN)->size=1
-- COL->select(e | p[e]) = COL->reject(e | not p[e])

Inv uniqueUserName:
Profile.allInstances->forAll(p1,p2 | p1<>p2 implies p1.userN<>p2.userN)

Inv uniqueUserName:
Profile.allInstances->forAll(p1,p2 | p1.userN=p2.userN implies p1=p2)
-- (A implies B) <=> (not B implies not A)
...

class Friendship
attributes
    status: String init: 'pending'
operations
    acceptF() begin self.status := 'accepted' end
    declineF() begin self.status := 'declined' end

class Friendship
attributes
    status: String derived:
        if oclInState(pending) then 'pending' else
        if oclInState(accepted) then 'accepted' else
        if oclInState(declined) then 'declined' else null endif endif endif
operations
    acceptF() begin end
    declineF() begin end
-- operation call changes statechart status
composition PosterPosting between
  Profile [1] role poster
  Posting [*] role posting
end

composition PosterPosting between
  Profile [0..1] role poster
  Posting [*] role posting
end

-----------------------------

association Interest between
  Profile [*]
  Subject [*]
  -- implicit role names: class name with starting lower case letter
end

association Interest between
  Profile [*] role profile
  Subject [*] role subject
  -- explicit role names
end
associationclass Commenting between
  Profile [*] role commenter
  Posting [*] role commented
attributes
  comment:String
end

constraints
context Commenting inv commentOnlyByFriends:
  commented.poster.friends() \rightarrow \text{includes}(\text{commenter})

associationclass Commenting between
  Profile [*] role commenter
  Posting [*] role commented
attributes
  comment:String
constraints
  inv commentOnlyByFriends:
    commented.poster.friends() \rightarrow \text{includes}(\text{commenter})
end
USE shell with SOIL statements - Variations G

!create merkel,putin,trump:Profile
!merkel.init('Angela','Merkel','muddi')
!putin.init('Vladimir','Putin','crab')
!trump.init('Donald','Trump','theDonald')
!putin.invite(merkel)
!trump.invite(putin)
!putin.decline(trump)
!merkel.accept(putin)

!p:=merkel.publish('BMW, we have a problem')
!create may:Profile
!may.init('Theresa','May','motherTheresa')
!putin.comment(p,'May the Donald be with you')
!may.invite(merkel)

!merkel.publish('BMW, we have a problem')
!create may:Profile
!may.init('Theresa','May','motherTheresa')
!putin.comment(Posting.allInstances->any(p|p.poster=merkel),
                  'May the Donald be with you')
!may.invite(merkel)

Shell command on several lines
use> \
> !putin.comment(Posting.allInstances->any(p|p.poster=merkel),
>                   'May the Donald be with you')
> .
Thanks for your attention!