Suggested Solution

Part 1: Modeling (35%)

Protocol interpretation and assumptions

The sequence diagram leaves many questions unanswered and therefore also room for interpretation. We will begin by stating the interpretation we will use for the rest of this discussion.

While the "independent agents" of the FIPA protocol allow every agent to initiate a meeting, that does not mean it doesn't distinguish between roles. The protocol makes a clear distinction between a meeting initiator and a (potential) meeting participant. From the initiator's point of view, we assume the following protocol based on the sequence diagram Negotiation:

1. The initiator makes a call-for-proposal;
2. Waits for proposals and/or refusals to come in;
3. Accepts one proposal or rejects all proposals (and the meeting);
4. If a proposal was accepted, attends the meeting and (possibly optionally) submits an evaluation.

From the participant's point of view, the protocol looks like this:

1. The (potential) participant receives a call-for-proposal;
2. Evaluates the call-for-proposal and responds with either a refusal or a proposal;
3. Waits for the initiator's judgement;
4. If the initiator accepts one of the proposals (and this participant also accepts, in case it was someone else's proposal that was accepted) attends the meeting and (possibly optionally) submits an evaluation.

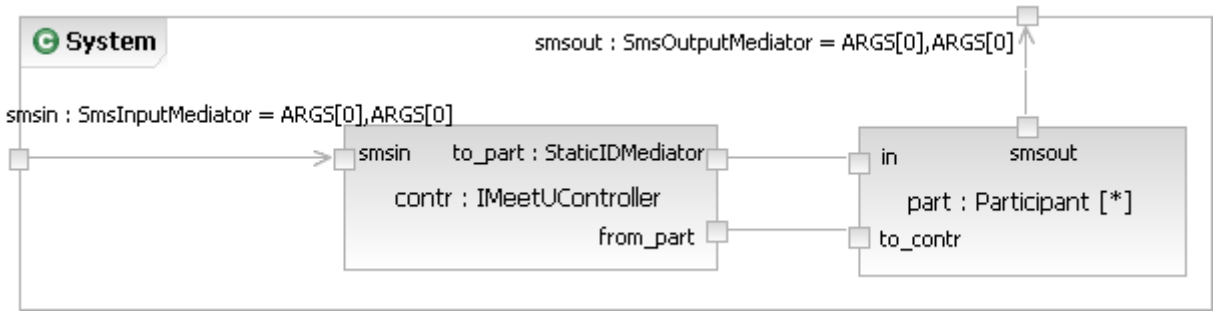
Because the protocol is experienced quite differently by the two roles, we will use the terms "Initiator" and "Participant" in our discussions to denote each kind of user when the difference is essential. When we need to refer to all users, regardless of kind, we will use the term "User".

For the same reason it is not necessarily natural to treat all participants equally (as instances of the same type). Nevertheless, despite the protocolary distinction, the requirements specifically demand that all participants be represented structurally by the same part type. This requirement limits our architectural choices but there are still a number of various designs available to us: 1) Define an abstract supertype, which is defined as the part type. Then two concrete subtypes are defined to correspond with the two roles. 2) Use one part type, but let the type contain two separate state machines, one for each role. Which state machine to use is determined once and for all at instantiation time. 3) Use one part type and one state machine which is logically divided interally to represent the behavior of the two roles.

The first two solutions are not readily implementable with the tools the students have used in the course. Therefore the last option is probably the most likely solution and the one we will pursue here.

1 a) Composite structure

The following diagram presents one possible composite structure of the system.



The **router (IMeetUController)** is responsible for

- Receiving SMS messages from the users, parsing the input messages and sending appropriate signals to the relevant participants.
- Route internal signals.

The **Participant** parts are responsible for

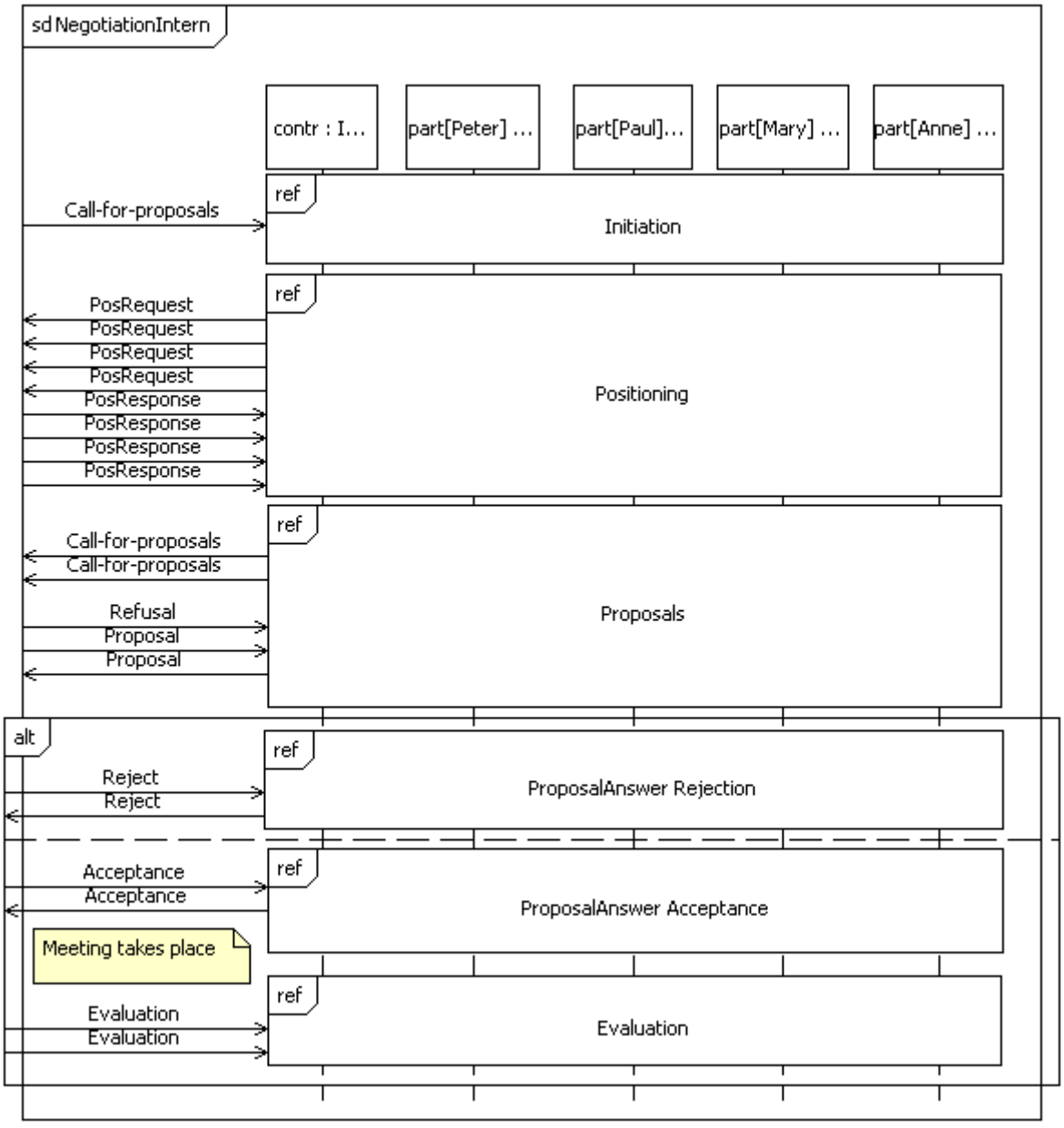
- Implementing the meeting setup protocol for each kind of user.

There is one instance of this part for every user in the context of one meeting. Indeed, if the system supported multiple simultaneous meeting negotiations, there could be multiple instances per physical user.

1 b) Decomposition

For clarity, we will divide the entire negotiation process into phases. In phase one, “Initiation”, the system receives a call-for-proposal and prepares for the negotation. The first task of the negotiation is to perform positioning of all the potential meeting participants and make an initial decision on who are eligible for participation (i.e. are close to the initiator). In the third phase, “Proposals”, the potential partitipators receive and evaluate the call-for-proposal and decides whether the respond with a proposal or to refuse the meeting. Next, the initiator evaluates the proposals (if any) and decides whether to accept one proposal or cancel the meeting (rejecting all proposals). If a proposal is accepted, the meeting takes place and the system awaits a final evaluation.

The diagram NegotiationIntern illustrates how we divide the negotiation into multiple phases. The messages exchanged in this diagram must correspond exactly with the messages in the diagram Negotiation.

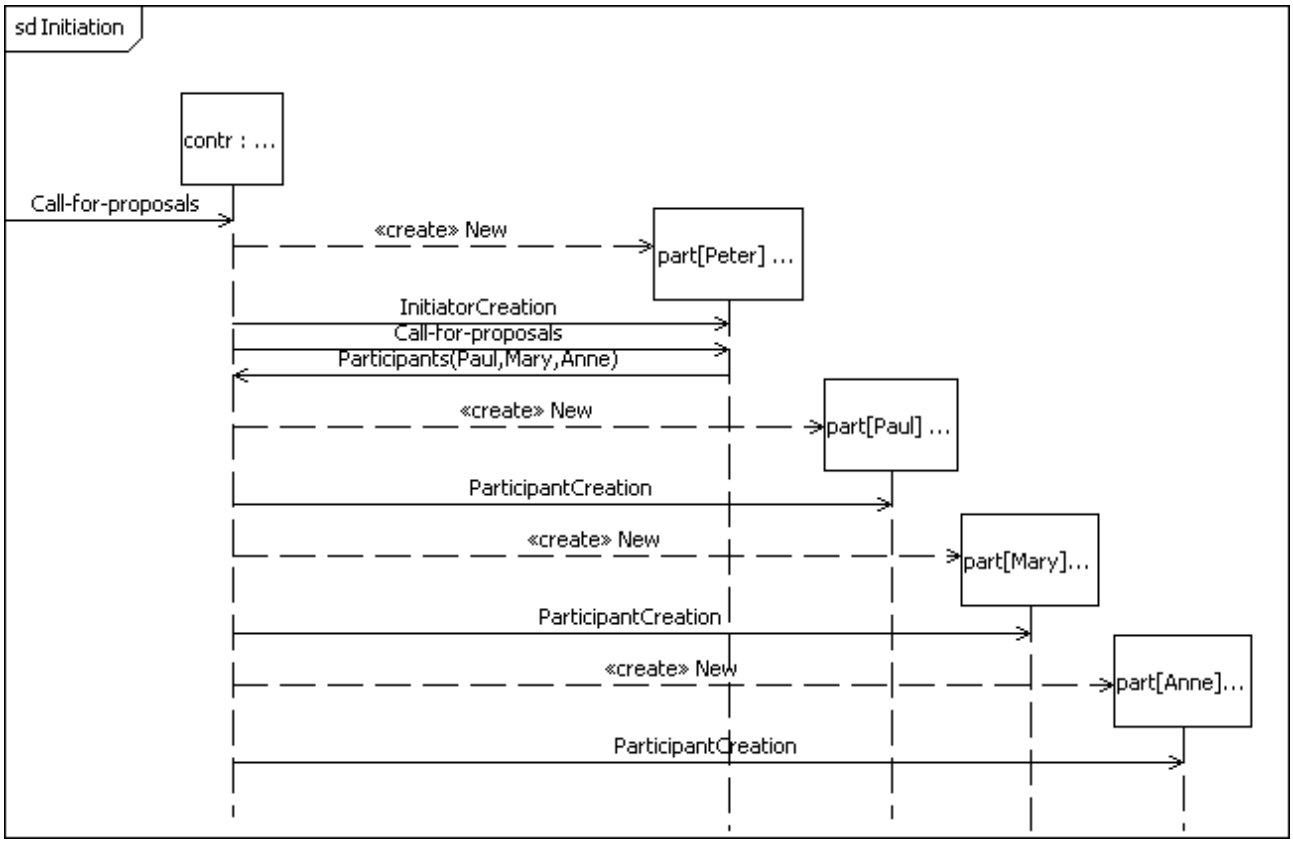


Initiation

The diagram Initiation below illustrates the internal communication during the first phase of the negotiation. The Call-for-proposals is received by the system and a new participant is created to handle the call. Since Peter sent the call, the new part represents Peter and he is assigned the Initiator role for the duration of the negotiation. The Call-for-proposal is then forwarded to this new part.

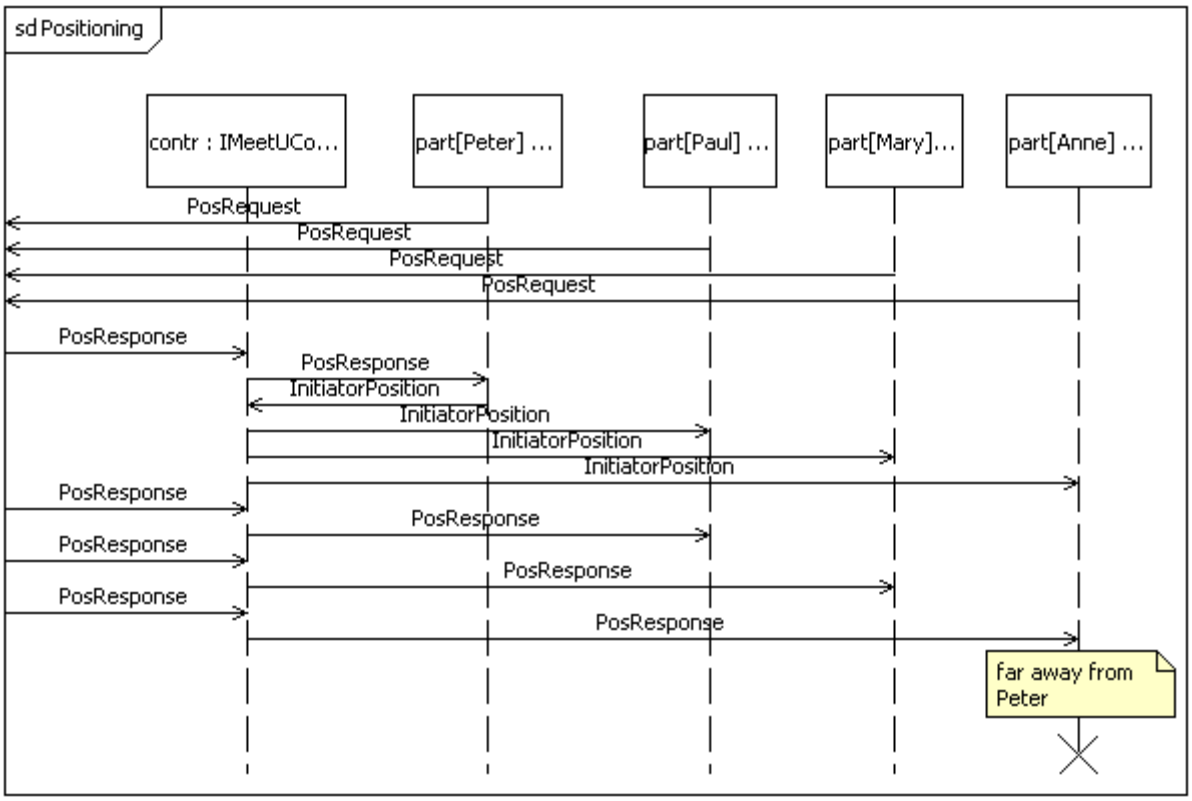
The first thing it does it to find out who are the potential participants to this meeting. Since we are allowed to assume a small system, we let the part discover internally who the other participants may be. For example, the part may contain a static list of participants. For larger systems, it would make sense to add an dedicated Archive part to the system and let that maintain the user database. For now, however, the part discovers the other participants internally and sends the information back to the controller.

The controller in turn creates additional Participant parts and assign them Participant roles.



Positioning

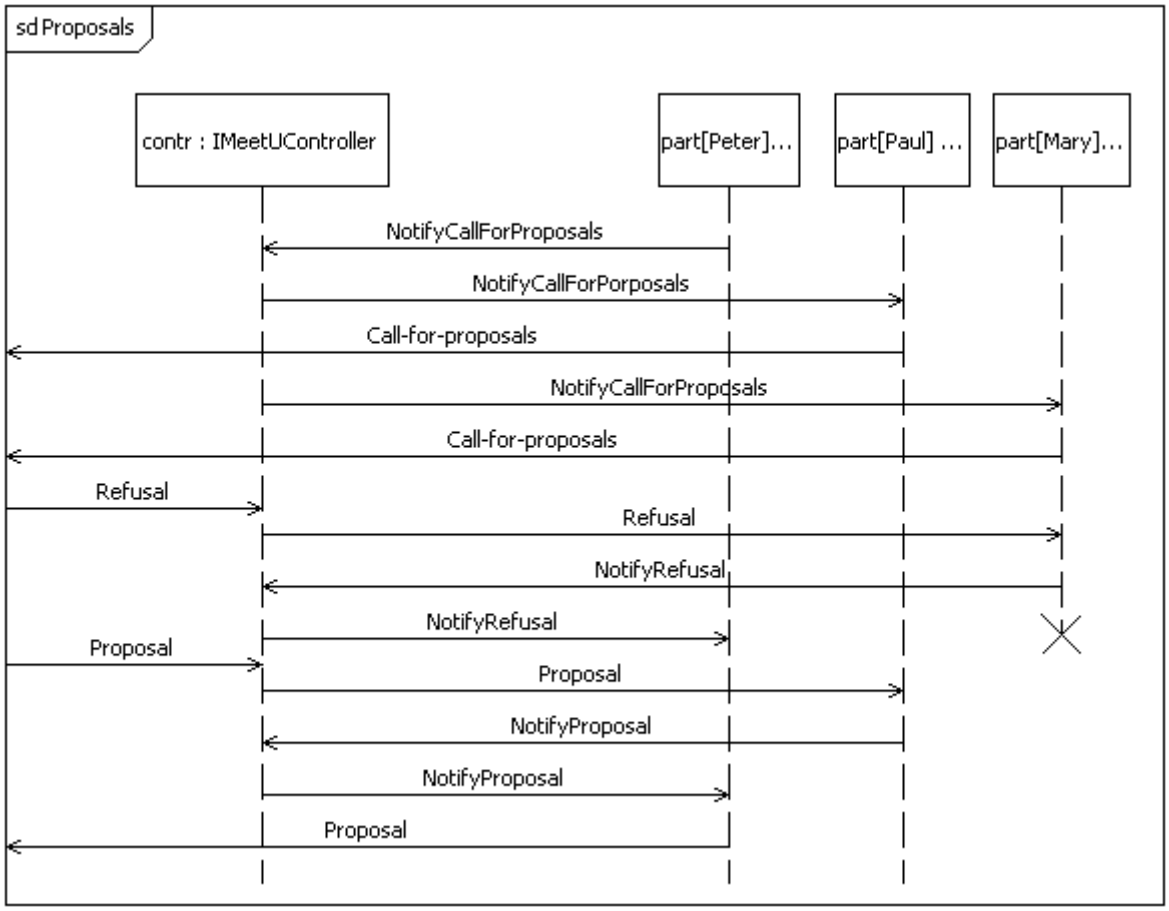
The diagram Poisitioning shows how the system performs positioning to determine the distance between the Initiator and the other participants. Each part sends a PosRequest after they have been assigned a role, as shown in the diagram Initiation above .



The two roles respond somewhat differently to the positioning response. The initiator broadcasts his position to all the other participants. The other participants receive the position of the Initiator as well as their own position and can then calculate the distance. If the distance turns out to be too large, as is the case for Anne, the part can immediately leave the negotiation and terminate. The remaining parts continue to the next phase, Proposals.

Proposals

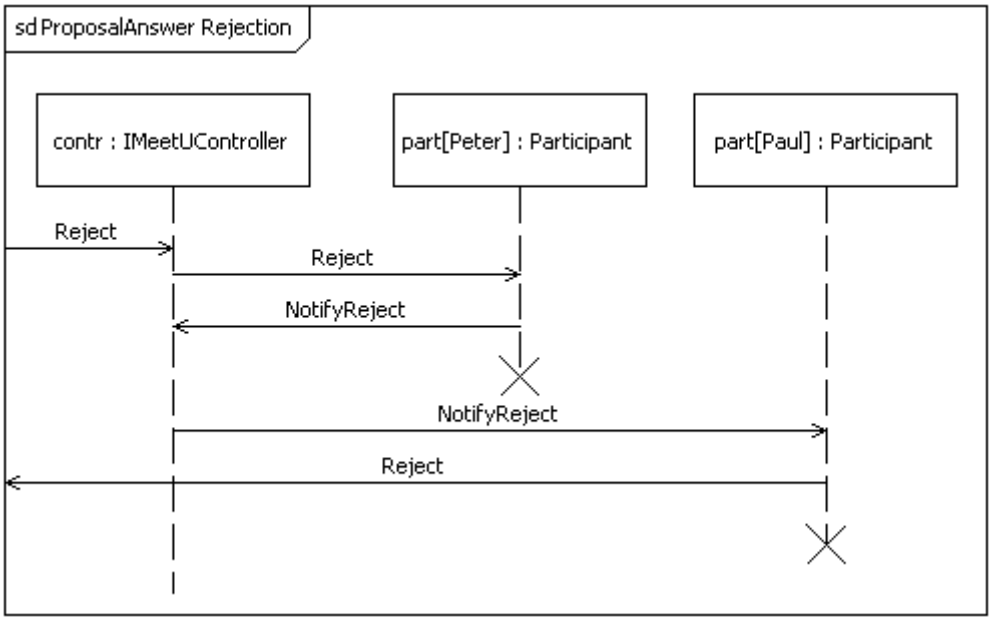
The diagram Proposals below show how the Call-for-proposal is forwarded to the potential participants and how they respond. There are two possible explicit responses to a Call-for-proposals: either the participant responds with a Proposal or he/she refuses the meeting.



When the refusal arrives at the part representing the participant, it notifies the Initiator about the refusal as well. Similarly when a proposal arrives. Proposals, however, are forwarded to the user. As we will see, both NotifyRefusal and NotifyProposal signals are used internally by the initiator participant for book-keeping purposes.

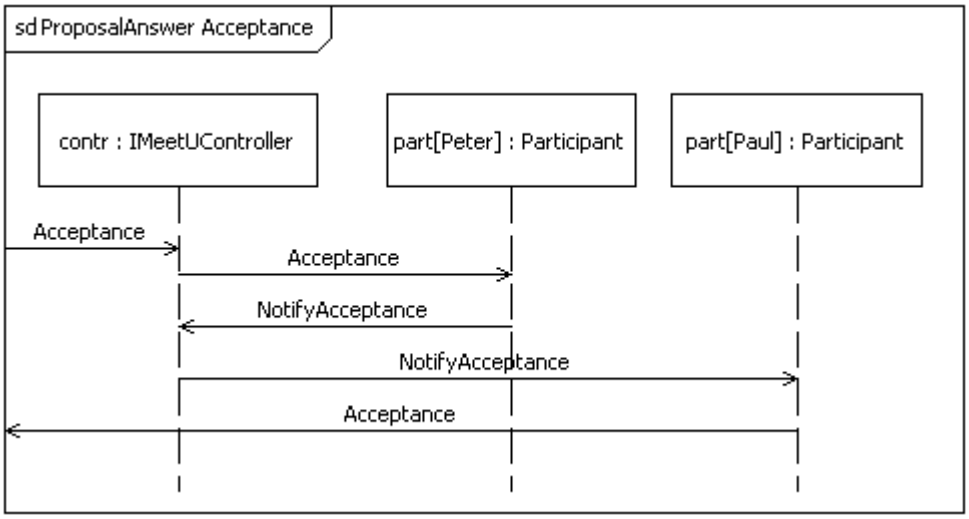
ProposalAnswer: Reject

Once the proposals have been received, Peter may choose to reject all of them, effectively canceling the meeting and ending the negotiation. The rejection is forwarded to all the participants who still take part in the negotiation. At the end, all participant parts have been terminated.



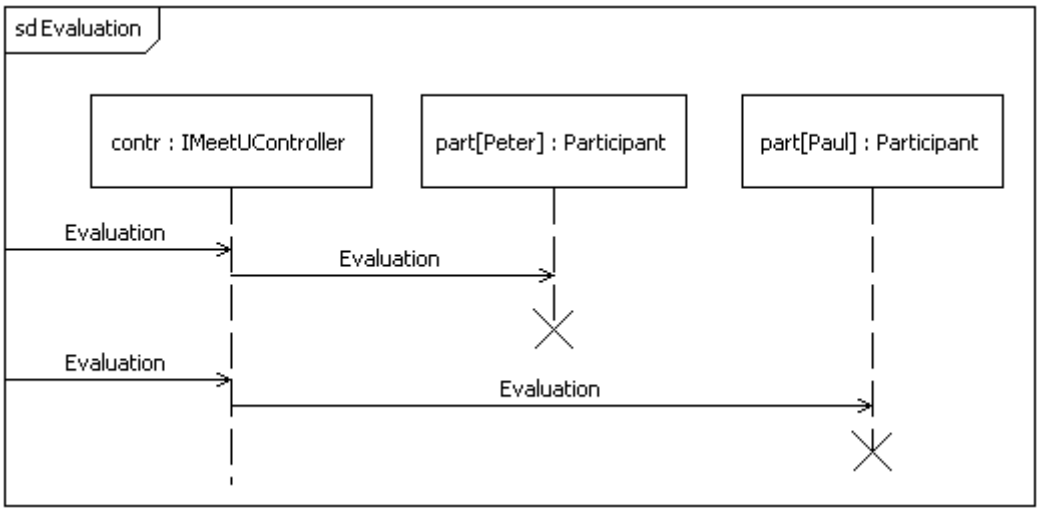
ProposalAnswer: Acceptance

Normally the initiator will accept one proposal. When the acceptance is arrives at the system, the part representing the Initiator (Peter) is first informed about the decision. This part then notifies all the other remaining participants in the negotiation. In the diagram Negotiation, only Peter and Paul remain at this time, and the Participant parts representing each of them receive the signal. The Initiator part (representing Peter) need the signal to know it should begin to wait for the evaluation. The other participant parts (representing Paul) need the signal in order to forward the message to the respective users (and to begin waiting for the meeting evaluation).



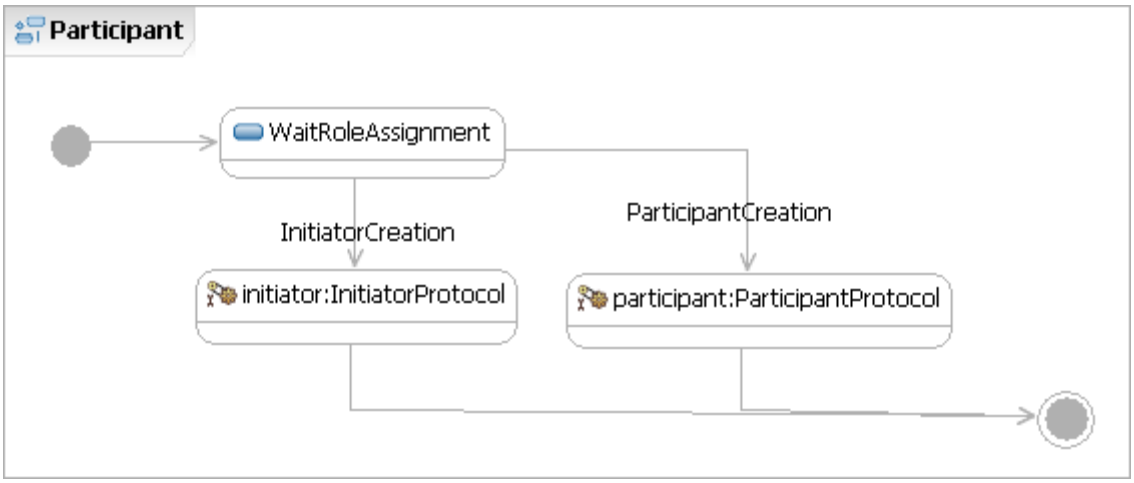
Evaluation

As the final meeting evaluations arrive at the system, they are routed to the corresponding part, which subsequently terminates.



1 c) State machine

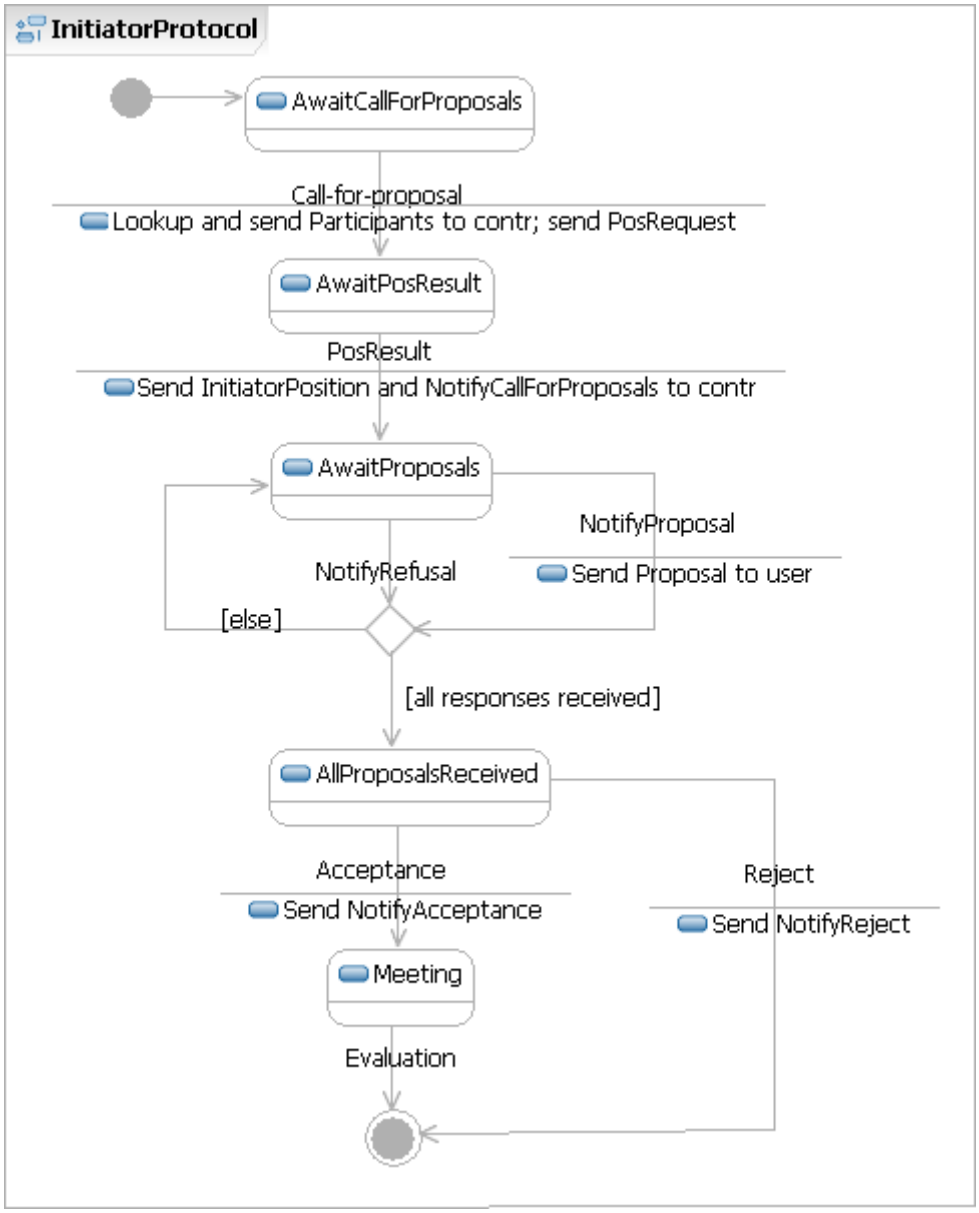
We propose the following top-level state machine for initiators and participants. The initial effect after creation is to await a role assignment. The Initiation sequence diagram illustrates how each part receives a role that is either Initiator or Participant. Based on the assigned user role, the state machine enters one of two submachines state.



The following sections discuss additional details of the machine.

SM InitiatorProtocol

The protocol specific for the Initiator role in a meeting negotiation is implemented by the substate machine InitiatorProtocol.



When this state machine begins, it expects the original Call-for-proposal. In response, it will look up all the other potential participants and submit a Participants signal to the controller. Next it sends a PosRequest in order to position the initiator user.

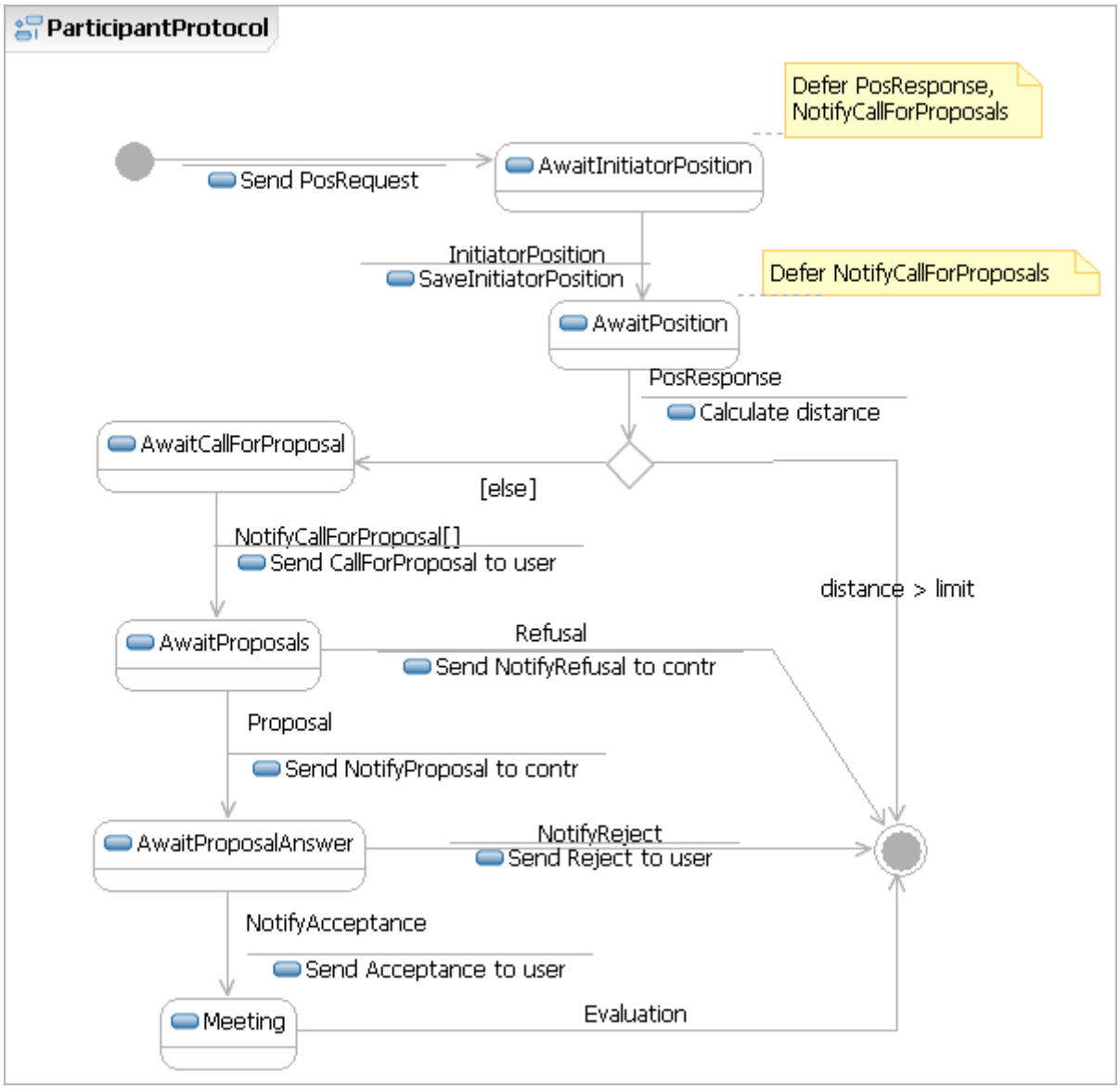
When the position result arrives, it is bounced back to the controller as an InitiatorPosition signal. The controller is then responsible to route this signal to all the other participants. Immediately afterwards, it also sends the NotifyCallForProposals signal that is also broadcasted to all the other participants. These two signals could be combined into one.

Now the state machine enters a state in which it awaits proposals from the other participants. The machine must maintain an internal counter that will tell us when all responses have been received.

If a proposal is accepted, the machine enters the Meeting state in which it awaits the Evaluation.

SM ParticipantProtocol

The protocol specific for the Participant role in a meeting negotiation is implemented by the substate machine ParticipantProtocol.



When this state machine begins, the first thing we need to do is to determine the user's position by sending a PosRequest. Next, we await three pieces of information: the Initiator's position, this user's position and the call-for-proposal details. By using the defer mechanism, we can ensure that these signals are handled in a sensible order. Only if the user is sufficiently close to the initator, will the negotiation continue and the call-for-proposal is forwarded to the user.

The user may response either with a Proposal or with a Refuse message. In either case, an appropriate Notify signal is sent to the controller (and from there to the initiator). The initiator will in turn answer the proposals with either Acceptance or a Reject message. In either case, an appropriate message is sent to the user.