Challenges to UML 2 to describe FIPA Agent protocol

Øystein Haugen
SINTEF, Oslo, Norway
University of Oslo, Departement of Informatics, Oslo, Norway
oystein.haugen@sintef.no

Abstract. The FIPA Contract Net Interaction Protocol cannot be described properly with UML 2.1 Sequence Diagrams. The ability to multicast messages is one feature that is lacking. We show how small enhancements of UML will make it more suited to express agent protocols. We also explain why the enhancements proposed by the Agent UML are not quite satisfactory.

Introduction

The FIPA protocol for negotiations between agents [1] applies a variant of sequence diagrams to express the protocol together with natural language text. At the time when the FIPA standard was created, UML was still in its version 1. Since then the OMG has standardized as available technology UML 2 [2]. In this paper we investigate whether the FIPA protocol can be described by means of UML 2. We assume that the FIPA protocol represents interaction that is typical for the needs in the realm of agents. When we conclude that UML 2 cannot fully cope properly with the FIPA protocol, we therefore suggest a few enhancements that will make UML more capable of defining the FIPA protocol and thereby more suitable for the needs of the agent community. The research of this paper has been conducted within the ITEA2 project MoSIS [3].

The FIPA protocol shows how an initiator sends out a number of calls for proposal to a set of participants. Some of these participants will refuse the call, while others may come up with a proposal. The initiator will then consider the proposals and reject some, and accept some. For those that are accepted, there are three different final results sent back to the initiator. The FIPA protocol is depicted in Fig. 1.

The FIPA protocol is not a valid UML 2 diagram. It is described in a dialect of sequence diagrams using extensions to UML 1 known as “Agent UML” described in [4]. Some improvements to the notation was suggested by [5]. The key concept to express the multicasting is the multiplicity on the message ends to show how many message instances the diagram really describes. This approach functions well on an informal level, but is not sufficient if we want to describe what happens with every participant. The FIPA diagram says nothing about which participants are involved in which sequence of messages. E.g. there is nothing in the diagram that expresses that the reject-proposal messages are sent only to participants that earlier sent proposals back to the initiator. This is obvious to the human reader, but is nowhere defined. The problem in general is that multiplicity on messages is not sufficient to express the
subsets of senders or receivers. We show in this paper means to express subsets of agents through annotations on the lifelines and/or on the messages.

Fig. 1. FIPA Contract Net Protocol

In the sequel we shall look at how an experienced UML designer would attempt to express the FIPA protocol. Some of what we show here is similar to what was shown in [6] for how to use UML 2 concepts to express agents. We emphasize the precise interpretation of the diagrams and the fact that there are differences between approaches that all may serve the purpose of expressing agent interactions. Our aim is a very narrow one; we merely want to improve the use of sequence diagrams in the context of agents.

The single participant approach

One approach is to define a collaboration (Fig. 2) where there is one initiator and one participant. That participant has all the abilities of any participant, but of course for one single case, the participant is only one person.
Fig. 2. Agent Context for the single general participant

We apply the combined fragments of UML 2 sequence diagrams to define the different options that are open to the participant.

Fig. 3. FIPA protocol for the general participant

In Fig. 3 we apply the break-fragment which means that either the fragment contents happen or the rest of the diagram (actually the rest of the enclosing interaction fragment) happens. Thus we are able to express that for a participant that refuses the call for proposal there will be no continuation of the protocol. Our diagram here is very similar to that of [6], but we apply the more compact break-fragment rather than the nested alt-fragments.

Our notation is more compact than the original FIPA illustration, and more precise. Compared with the diagram shown in [6] the precision is obviously the same disregarding the use of multiplicities on messages that we shall return to shortly.

However, we still have the problem that the approach misses the point that the initiator sends calls to many potential participants and that he in fact must cope with many replies. The diagram shown in Fig. 3 shows a very limited view, the situation as seen from one participant rather than the initiator.
The approach of typical participants

Our next attempt is intended to express that there are different participants with different situations relative to the initiative described in the protocol. Rather than defining one participant that is fully general, we now describe one participant for each distinct situation. In our collaboration (Fig. 4) we define a set of participants with multiplicity.

![Fig. 4. Agent context with set of participants](image)

We apply the standard notation in the lifeline header to select one object from a set by a selector. The selector in this case is only a symbolic name serving as an index indicating what situation that given participant object is representing.

![Fig. 5. FIPA protocol for typical participants](image)

In Fig. 5 we have one lifeline for each of the typical situations a participant may be in. In this way we visualize more directly that there are several participants that the
initiator must relate to. We are not able, however, to describe the multiplicities of each of the subgroups because each of the lifelines represents only one participant.

This approach fails to give the impression that the initiator has the same original approach to all participants in the first place. It also shows a situation where the initiator handles the different typical participants in strict sequence. This is by no means illegal as this diagram is not necessarily intended to define all possible traces of the protocol. Sequence diagrams show possible runs, but seldom all possible runs. Still we may want to express that there is nothing that prevents the initiator from handling the return from an accepting participant before the return from a refusing participant. We can achieve that by introducing such standard constructs as coregions which express that events may come in any order.

Introducing configurations with subsets

The problem with the approach of typical participants was that the sets from which these typical lifelines were selected, were not properly described. Their multiplicity and internal relationships were not defined.

We remedy this by introducing configurations with subsets. In fact this is available in UML 2 already, but seldom used in modeling with composite structures, while the UML 2 metamodel has very many subset declarations on association ends in class diagrams. Subsets are constraints on a class property indicating that the defined property is a subset of some other property defined in a superclass of the encloser. An example (Fig. 6) will make this clearer.

In AgentContext we define one total set of participants \( p \) with multiplicity \( m \). AgentContextSpecial is a specialization of the general AgentContext where we have defined three subsets of \( p \) and these subsets are given new names refused, rejected and accepted. What this does is to keep the information that all objects of these sets are still contained in the original \( p \), but that they may have added capabilities or situations. In fact it would be natural in our FIPA context to have three layers of specialization. The first distinction goes between those that refuse and those that propose. The second distinction defines subsets within those that propose, namely between the rejected and the accepted. We flattened the two lower subclasses for illustration simplicity.
The behavior definition given in Fig. 7 is structurally equivalent to that of Fig. 5, but the names of the lifelines reflect the subsets. Within each of the subsets we apply the single lifeline approach as every object within the subset should exhibit the same behavior.
Notice that we have not applied any constructs that are not in UML 2 already, but we have in this latest approach applied a description technique that is not very common. The technique has been presented with additions by Haugen and Møller-Pedersen in [7].

Introducing subset notation on messages

We have found that the subset construct that can be effectively applied to composite structures as shown in Fig. 6 are useful also as identifiers in sequence diagrams as shown in Fig. 7. But the latter still fails to capture properly the distinction between showing one typical instance of the set (as depicted first in Fig. 5) and expressing that the interaction actually holds for every instance of the set.

Applying multiplicities to the messages as suggested by the FIPA protocol standard (Fig. 1) does not quite express this precisely as this is not a matter of numbers only, but rather a matter of subsets. That is why we suggest introducing a subset notation for messages that very much corresponds to the FIPA protocol standard.
The notation is simple. Attached to one (or both) ends of a message there is a constraint that has the keyword `all` followed by a part name. That part must be a subset of the part represented by the lifeline on the message end with the `all` constraint. In Fig. 8 we have that `refused`, `rejected` and `accepted` are all subsets of $p$.

Notice also that we have reached the same kind of compactness in our description that we had with our first approach with the general participant in Fig. 3.

Informally the meaning of this notation is the same as described by the FIPA protocol. Take e.g. the `refuse` message from $p$ lifeline to `initiator` where the sending has attached the constraint `{all refused}`. This obviously is intended to mean that there is one message from every member of the `refused` subset to the `initiator`.

Likewise the `reject-proposal` message with `{all rejected}` constraint on its receiving side intends to describe one message sent from `initiator` to every member of the `rejected` subset of $p$.

But, we need one construct more. It is not sufficient only to be able to define multicasting of messages to or from a given subset. The problem is not to define how to send a bunch of messages out, the problem lies in defining how to deal with the various responses that will return from that multicast. In Fig. 8 we see how the `propose` message or the `refuse` message are responses to the initial `cfp`. We find that defining the `refused` and `rejected and accepted` subsets is a fruitful way to define the different alternatives. We could continue to subset the participants into smaller and smaller subsets, but in the end we choose a different variant. We define a combined...
alt-fragment with an iterator-clause. The “alt \{all accepted\}” fragment iterates over all participants of the accepted subset meaning that every accepted participant has that choice between sending back failure, inform-done or inform-result.

The semantics of the multicasting and the iterator-clause

We define the semantics of our proposed constructs as shorthands. We define a transformation procedure that transforms a diagram with multicasts and iterators into a standard UML 2 sequence diagram. The transformation should not really be carried out because it assumes creating lifelines for all objects of all parts in the composite structure. It is, however, a simple way to define the semantics since the transformation may in principle be carried out and the semantics of the imaginary standard diagram are defined elsewhere.

Fig. 9. Subsetcompact – multicasting and iterator

The sequence diagram in Fig. 9 shows a compact version of the protocol isolating only the multicast and the iterator constructs.

Assume that the accepted subset of $p$ only contains two participants, the expansion of the constructs results in the sequence diagram in Fig. 10. The procedure for transforming multicasting has three steps:

1. Create one lifeline for each object in the subset mentioned in the all-constraint.
2. Replicate the message for every created lifeline.
3. Contain the multiple messages sends (or correspondingly receives) in a coregion indicating that the sending / receiving may come in any order.

In the situation where the all-constraint is used on both ends of a message, it is necessary to replicate the message for every pair of lifelines from the two lifeline sets involved (on either side of the message). On every lifeline there will be a coregion.
The iterator has a similar definition and in fact it can be seen as the more general of the two procedures since the multicasting can be described as if the message was contained in a special combined fragment with iterator.

1. Create one lifeline for each object in the subset mentioned in the iterator clause.
2. Replicate the combined fragment for every created lifeline.
3. Contain each created combined fragment in an operand of an enclosing par-combined fragment.

The par-fragment is defined in UML2 as a parallel merge operator that will merge in all possible ways the sequences of the operands.

That we can define our new constructs as shorthands that may be transformed into standard sequence diagrams means that we have not introduced anything that obstructs the good semantic properties of sequence diagrams such as compositionality. By compositionality we mean that the sequence diagrams can be refined piecewise and we can be certain that the result when putting these pieces back together is a refinement of the original. Using the STAIRS approach [8-10] we formalize this by a trace semantics where we can show that sequence diagrams are
monotonic with respect to refinement for most of the standard operators such as alt, seq, loop, par [11].

Our definition of multicast is not the first attempt to define multicasting in the area of sequence diagrams. Helouet made a definition in [12] which was combined with the ITU version Z.120 where he defines multicast groups that in some way resemble our subsets. Another definition was given by Gherbi and Khendek [13] using OCL for defining the transformations. Event Studio (http://eventhelix.com/) claim to have included a multicast feature in their sequence diagram tool.

Conclusions

We have shown how UML 2 can be used to express the FIPA Contract Net Interaction Protocol. We have gone through a series of approaches with varying capabilities with regards to precisions, completeness and compactness. Finally we suggested two improvements to the sequence diagram notation to define multicast messages and combined-fragment iterators over subsets.

The clue to these new constructs is to apply the already existing UML construct of subsets of properties. The new constructs are defined as shorthands that may be transformed into standard sequence diagrams. This preserves the positive characteristics of sequence diagrams such as compositionality. Furthermore we believe that it preserves the intents of the FIPA protocol designers and that it is more accurate than the earlier proposals to use multiplicities.

References

2. OMG, Unified Modeling Language 2.1. 2006, OMG.