Systèmes Multi-agents

Communication

Claude Moulin

Université de Technologie de Compiègne

IA04

Sommaire

Objectif

- Etudier les aspects de la technologie Agent relatifs à un groupe (société) d'agents, plutôt qu'à un agent individuel :
  - communication : Actes de langage (speech acts) ; KQML & KIF ; FIPA ACL.

Actes de langage

- Le traitement de la communication dans les systèmes multiagents est inspiré des théories des actes de langages.
- Ces théories sont avant tout pragmatiques :
  - Elles essaient d'expliquer comment le langage est utilisé par les humains pour atteindre leurs buts et leurs intentions.
- Toute parole est prononcée pour satisfaire un but ou une intention.
- Réciproquement une théorie qui étudie comment les paroles permettent d'atteindre un but est une théorie d'actes de langage.

Exemple

Searle (professeur de phylosophie) en 1969 identifia différents types d'actes de langage :

- Actes représentatifs :
  - Donner une information : "il pleut".
- Actes directifs :
  - Demander de faire quelque chose : "fais le café".
- Actes commissifs :
  - S'engager à faire quelque chose : "je promets de venir".
- Actes expressifs :
  - Exprimer un état mental : "merci"
- Actes déclaratifs :
  - Annoncer un événement : "il annonça le décès de ..."

Structure d'un acte de langage

- En général, on considère qu'un acte de langage a deux composants :
  - un verbe performatif : demander, informer, refuser, ...
  - un contenu propositionnel : la porte est ouverte

Exemple

- performatif = se renseigner
- contenu = la porte est ouverte
  - sens = Est-ce que la porte est ouverte ?

- performatif = informer
- contenu = la porte est ouverte
  - sens = la porte est ouverte

- performatif = demander
- contenu = la porte est ouverte
  - sens = s'il te plait ouvre la porte

Sommaire

KQML et KIF

- Le plus connu des ACL est KQML. Il comprend deux parties :
  - Le langage de connaissance et de requêtes : KQML ;
  - Le format d'échange de connaissance KIF.
- KQML définit les performatifs acceptables (demander, exécuter, informer, répondre)
- KIF exprime le contenu

Exemple KQML / KIF

```
A to B: (ask-if
          (> (size chip1) (size chip2)))
B to A: (reply true)
B to A: (tell (= (size chip1) 20))
B to A: (tell (= (size chip2) 18))
```

Sommaire

Sommaire

F.I.P.A

- IEEE foundation for intelligent physical agents
- `http://www.fipa.org/index.html`
- FIPA travaille sur la définition de standards pour la théorie des agents.
  - Architecture de plateformes, architecture d'agent (cycle de vie), mobilité, etc.
- La pièce centrale est la définition d'un ACL.
  - Structure d'un message
  - Performatifs d'un message
  - Structure du contenu du message

Message F.I.P.A

Norme FIPA SC00061G : FIPA ACL Message Structure Specification

- 13 attributs
- 1 performatif (22 performatifs FIPA)
- 11 attributs de gestion :
  - 3 attributs : participants à la communication (expéditeur, destinataire, redirection)
  - 3 attributs : description du contenu (langage, encodage, ontology)
  - 5 attributs : contrôle de la conversation (identificateur, ...)
- 1 contenu

Exemple de message F.I.P.A

```
(inform
   :sender agent1
   :receiver agent5
   :content (price good200 150)
   :language sl
   :ontology hpl-auction
)
```

INFORM et REQUEST

- Leur sens est défini en deux parties :
  - Pré-condition : ce qui doit être vrai pour que l'acte de langage réussisse.
  - Effet logique : ce que l'expéditeur du message espère provoquer.

INFORM

- Le contenu est une affirmation.
- Pré-condition : l'expéditeur considère que le contenu est vrai et ne pense pas que le destinataire a déjà été averti du fait.
- Effet logique : l'expéditeur s'attend à ce que le destinataire croit en la véracité du contenu.

REQUEST

- Le contenu est une action.
- Pré-condition : l'expéditeur pense que le destinataire est capable de réaliser l'action et ne l'a pas déjà fait.
- Effet logique : l'expéditeur s'attend à ce que l'action soit exécutée.

Sommaire

Paramètres - 1

Paramètre	Catégorie des paramètres
performative	Type of communicative acts : désigne le type de l'acte de communication.
sender	Participant in communication : désigne l'identité de l'émetteur du message (nom de l'agent de l'acte de communication.
receiver	Participant in communication désigne l'identité des destinataires prévus du message.
reply-to	Participant in communication : indique que la réponse au message doit être redirigée vers un autre agent à la place de l'agent émetteur.

Paramètres - 2

Paramètre	Catégorie des paramètres
content	Content of message : désigne le contenu du message (l'objet de l'action). Le contenu doit être interprété par le destinataire.
language	Description of Content désigne le langage avec lequel le contenu est exprimé.
encoding	Description of Content désigne l'encodage du contenu.
ontology	Description of Content désigne l'ontologie dans laquelle sont définis les symboles utilisés dans le contenu.

Paramètres - 3

Paramètre	Catégorie des paramètres
protocol	Control of conversation désigne le protocole d'interaction que l'agent émetteur emploie pour ce message.
conversation-id	Control of conversation introduit un identifiant de conversation utilisé pour identifier les actes qui participent de la même conversation.
reply-with	Control of conversation introduit une expression qui sera utilisée par le destinataire pour identifier le message. est utile lorsque plusieurs conversations de même identifiant sont utilisées en parallèle

Paramètres - 4

Paramètre	Catégorie des paramètres
in-reply-to	Control of conversation désigne une expression qui référencie une précédente action dont le message est une réponse.
reply-by	Control of conversation désigne une date qui indique la limite avant laquelle l'émetteur souhaite recevoir une réponse.

Sommaire

Catégories de Performatif

- Norme FIPA SC00037J : FIPA Communicative Act Library Specification
- 22 performatifs.
- Catégories :
  - passage d'information
  - demande d'information
  - négociation
  - demande d'exécuter une action, exécution d'action
  - gestion d'erreur

Performatif - 1

Inform	passage d'information L'émetteur informe le destinataire qu'une proposition est vraie.
Confirm	passage d'information L'émetteur informe le destinataire qu'une proposition est vraie. Le destinataire est connu comme non certain de la proposition.
Disconfirm	passage d'information L'émetteur informe le destinataire qu'une proposition est fausse. Le destinataire est connu comme croyant qu'elle était vraie.
Request	demande d'action L'émetteur demande au destinataire d'exécuter une action.
Agree	exécution d'action L'émetteur accepte d'exécuter une action. typiquement en réponse à un acte : request.
Refuse	exécution d'action L'émetteur refuse d'exécuter une action et en explique la raison.

Performatif - 2

Failure	gestion d'erreur L'émetteur prévient un autre agent qu'une action a été tentée mais qu'elle a échoué.
Not Understood	gestion d'erreur L'émetteur informe le destinataire qu'il n'a pas compris un précédent message.
Query If	demande d'information L'émetteur demande à un autre agent si une proposition est vraie.
Query Ref	demande d'information L'émetteur demande à un autre agent l'objet référencé par une expression.
Inform Ref	passage d'information une macro action de l'émétteur pour informer le destinataire de l'objet qui correspond à une description (ex : un nom).

Performatif - 3

Call for Proposal	négociation L'émetteur fait appel à des propositions.
Propose	négociation L'émetteur propose d'exécuter une certaine action étant données certaines préconditions.
Accept Proposal	négociation L'émetteur accepte une proposition précédemment soumise (après un acte : propose).
Reject Proposal	négociation L'émetteur rejette une proposition d'exécuter une action.
Cancel	exécution d'action L'émetteur informe un autre agent que ce dernier n'a plus à exécuter une action.
Subscribe	demande d'information L'émetteur demande d'être averti lors du changement de la valeur d'un objet référencé.

Performatif - 4

Propagate	demande d'action L'émetteur demande que le destinataire identifie les agents concernés par le message et de le leur propager.
Proxy	demande d'action L'émetteur veut que le destinataire devienne un proxy sélectionne les agents cibles désignés par une description.
Inform If	passage d'information une macro action de l'émetteur pour informer le destinataire si une proposition est vraie ou fausse.
Request When	exécution d'action L'émetteur demande au destinataire d'exécuter une action dès qu'une certaine proposition devient vraie.
Request Whenever	exécution d'action L'émetteur demande au destinataire d'exécuter une action dès qu'une certaine proposition devient vraie et ensuite chaque fois que la proposition le sera de nouveau.

Performatif - 1

perfor-mative	passing info	requesting info	negot-iation	performing actions	error handling
accept-proposal			x		
agree				x	
cancel		x		x	
cfp			x		
confirm	x				

Performatif - 2

perfor-mative	passing info	requesting info	negot-iation	performing actions	error handling
disconfirm	x				
failure					x
inform	x				
inform-if	x				
inform-ref	x				

Performatif - 3

performative	passing info	requesting info	negot-iation	performing actions	error handling
not-understood					x
propagate				x	
proxy				x	
propose			x		
query-if		x			
query-ref		x			
refuse				x	

Performatif - 4

perfor-mative	passing info	requesting info	negot-iation	performing actions	error handling
reject-proposal			x		
request				x	
request-when				x	
request-whenever				x	
subscribe		x			

Bilan

- cancel correspond à 2 catégories ; les autres à une seule catégorie.
- passing info : 5
- requesting info : 4
- negotiation : 4
- performing action : 8
- error handling : 2

Sommaire

- Norme FIPA SC00008I : FIPA SL Content Language Specification.
  SL : Semantic Language.
- Norme FIPA XC00009B : FIPA CCL Content Language Specification
  CCL : Constraint Choice Language
- Langages FIPA : SL0, SL1, SL2
- Langages privés :
  - Listes Lisp.
  - JSON : JavaScript Object Notation.

Proposition

- Une proposition est utilisée dans un message de type INFORM et d'autres types d'actes de communication dérivés.
- On peut attacher une valeur de vérité à une proposition dans un contexte donné.
- C'est une formule bien formée (well-formed formula : wff) utilisant les règles décrites dans la norme.
- Ex : `(> (size chip1) (size chip2))`

Action

- Une action peut être exécutée.
- Une action peut être simple ou composite (séquence d'actions et actions alternatives - opérateur ou).
- Une action est utilisée dans un message de type REQUEST et d'autres types d'actes de communication dérivés.

Expression référence

- Une ”identifying reference expression (IRE)”, identifie un objet du domaine.
- Elle utilise l'operateur de Référence.
- Une action est utilisée dans un message de type INFORM-REF et d'autres types d'actes de communication dérivés.

Autres expressions

- Composition des cas de base.
  - une action-condition (Action + proposition) est utilisée dans l'acte PROPOSE.
  - action-condition-reason (Action + proposition + proposition) est utilisée dans l'acte REJECT-PROPOSAL

Sommaire

Profil SL0

- Représentation d'actions
- Détermination de résultats
- Complétion d'actions
- Propositions binaires simples

Example SL0

```
((action
 garage
 (register
  (vehicle
   :color red
   :max-speed 100
   :owner (person
           :name Luis
           :nationality portuguese)))))
```

Grammaire SL0

```
Content = "(" ContentExpression+ ")".
ContentExpression =
        ActionExpression | Proposition.
Proposition = Wff.
Wff = AtomicFormula
  | "(" ActionOp ActionExpression ")".
AtomicFormula = PropositionSymbol
  | "(" "result" Term Term ")"
  | "(" PredicateSymbol Term+ ")"
  | "true" | "false".
ActionOp = "done".
Term = Constant
     | Set | Sequence | FunctionalTerm
     | ActionExpression.
ActionExpression = "(" "action" Agent Term ")".
```

Grammaire SL0

```
FunctionalTerm = "(" FunctionSymbol Term* ")"
     | "(" FunctionSymbol Parameter* ")".
Parameter = ParameterName ParameterValue.
ParameterValue = Term.
Agent = Term.
FunctionSymbol = String.
PropositionSymbol = String.
PredicateSymbol = String.
Constant = NumericalConstant
     | String
     | DateTime.
Set = "(" "set" Term* ")".
Sequence = "(" "sequence" Term* ")".
NumericalConstant = Integer
     | Float.
```

Profil SL1

- Ajoute à SL0 des opérations booléennes pour former des expressions propositionnelles plus complexes.
- Opérateurs : NOT, AND, OR

```
Wff = AtomicFormula
   | "(" UnaryLogicalOp Wff ")"
   | "(" BinaryLogicalOp Wff Wff ")"
   | "(" ActionOp ActionExpression ")".
UnaryLogicalOp = "not".
BinaryLogicalOp = "and" | "or".
```

Profil SL2

- Autorise la logique du premier ordre et la logique modale.
- Opérateurs $\forall$, $\exists$
- Variables
- Logique modale des croyances incertaines :
  B (belief) ; U (uncertain belief)
  questions associées : Bif (croire ?) et Uif (ne pas croire ?)
  désirs : D (desire)
  intentions : PG (intention)

Sommaire

Introduction

- JSON est un format léger d'échange de données.
- JSON est un format texte indépendant de tout langage de programmation.
- JSON utilise des conventions similaires à celle des langages à objets.

Exemple

```
{
 "value" : 200,
 "comment" : {
              "w1" : "Value",
              "w2" : "Ok"
             },
 "list" : [
           10,
           {"language" : "fr"},
           20]
}
```

Format de données JSON

- Un objet :
  - Ensemble de couples <nom/valeur> non ordonnés, entre { et }.
  - Un couple <nom/valeur> est de la forme : "<nom>" : <valeur>.
  - Les couples <nom/valeur> sont séparés par , .
- Un tableau est :
  - une collection de valeurs ordonnées mises entre [et].
  - les valeurs sont séparées par , .
- Une valeur peut être :
  - une chaîne de caractères entre guillemets,
  - un nombre
  - true ou false ou null
  - un objet, un tableau

Diagramme - 1

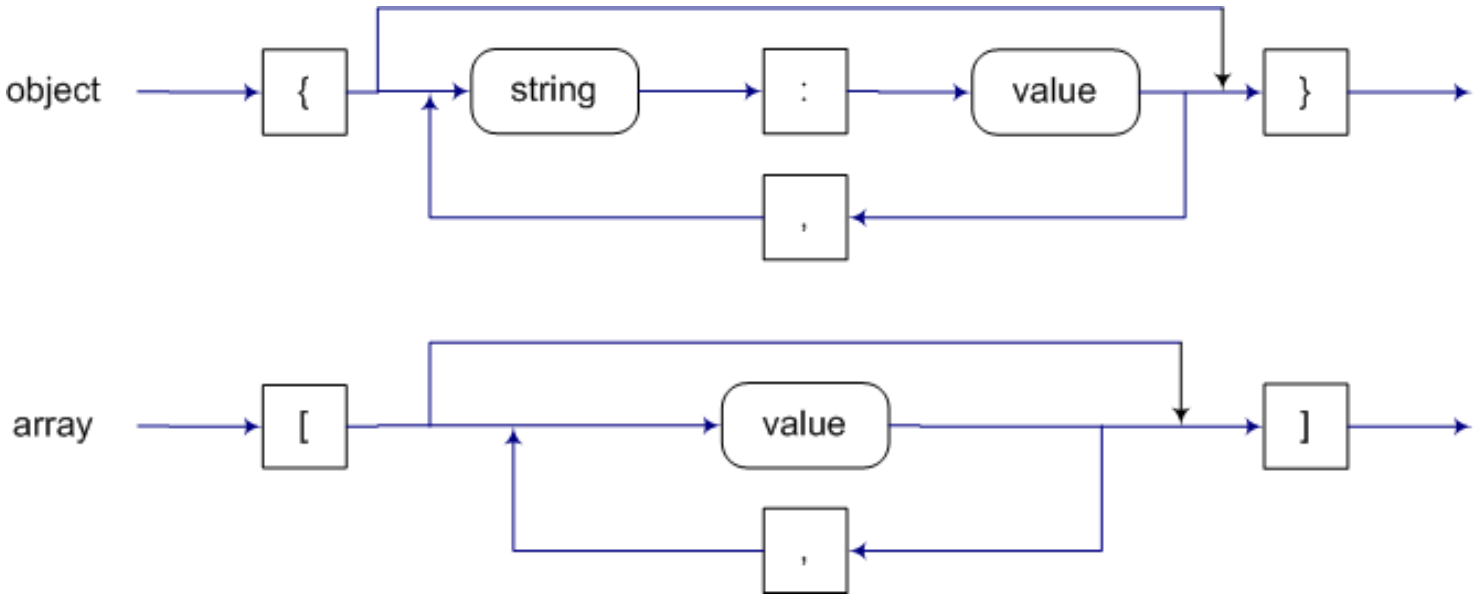
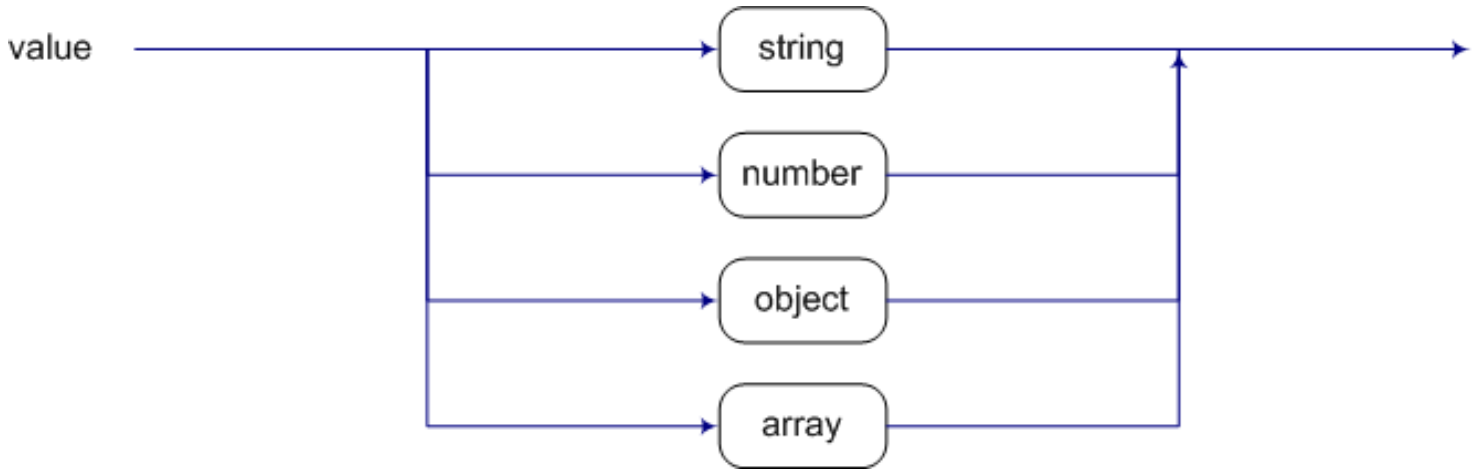


Diagramme - 2



Parsing - 1 (org.codehaus.jackson)

```
//{"value" : 200,"comment":{"w1" : "Value","w2":"Ok"},
// "list":[10,{"language":"fr"},20]}
String message = ... // structure json
ObjectMapper mapper = new ObjectMapper();
try {
 JsonNode jrootNode =
            mapper.readValue(message,JsonNode.class);
 JsonNode jlistNode = jrootNode.path("list");
 JsonNode jnameNode = jrootNode.path("comment")
                        .path("w2");
 String l = jrootNode.path("list").path(1)
            .path("language").getTextValue();
}
catch(Exception ex) { ... }
```

Parsing - 2 (org.codehaus.jackson)

```
ObjectMapper mapper = new ObjectMapper();
String message = ... // structure json
String answer;
try {
 Map<String,Object> map =
          mapper.readValue(message,Map.class);
 // instructions
 map.put("result", 200);
 StringWriter sw = new StringWriter();
 mapper.writeValue(sw, map);
 answer = sw.toString()
}
catch(Exception ex) {...}
```

Bilan JSON

- Le binding Json d'objets est simple à réaliser.
- Plusieurs binding sont disponibles :
  - objet, arbres JSON, Maps.
- La grammaire de JSON est beaucoup plus simple que celle de SL0.
- Le langage est moins expressif.
- Dans les deux cas les termes utilisés doivent faire partie d'un vocabulaire partagé.

Sommaire

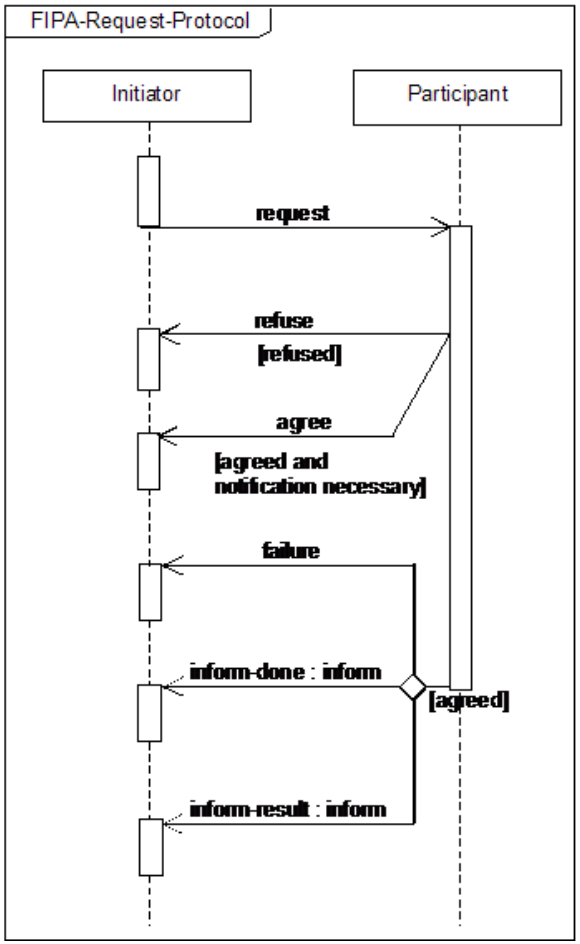
Protocoles

- Les protocles d'interaction sont des patrons de conversation entre agents.
- Principaux Protocoles :
  - Request : Norme FIPA SC00026H - FIPA Request Interaction Protocol Specification
  - Query : Norme FIPA SC00027H - FIPA Query Interaction Protocol Specification
  - Propose : Norme FIPA SC00036H - FIPA Propose Interaction Protocol Specification
  - Subscribe : Norme FIPA SC00035H - FIPA Subscribe Interaction Protocol Specification
  - Contract Net : Norme FIPA SC00029H - FIPA Contract Net Interaction Protocol Specification

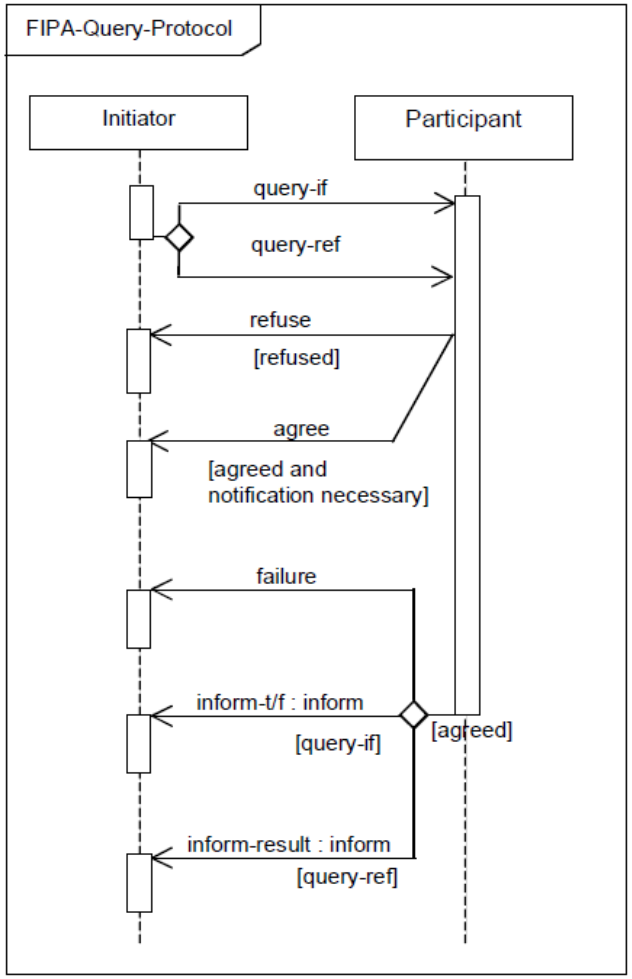
Request - 1

- Request Interaction Protocol : il permet à un agent de demander à un autre agent d'exécuter une action.
- L'agent sollicité peut accepter ou refuser la demande.
- Si l'action demandée est rapide le Participant doit communiquer soit :
  - un failure s'il a échoué dans sa tentative de réponse
  - un inform-done s'il a réussi mais veut seulement indiquer qu'il a la réponse.
  - un inform-result pour indiquer que l'action est réalisée et donner le résultat.

Request - 2


FIPA-Request-Protocol
Initiator
Participant
request
refuse
[refused]
agree
[agreed and
notification necessary]
failure
inform-done : inform
[agreed]
inform-result : inform

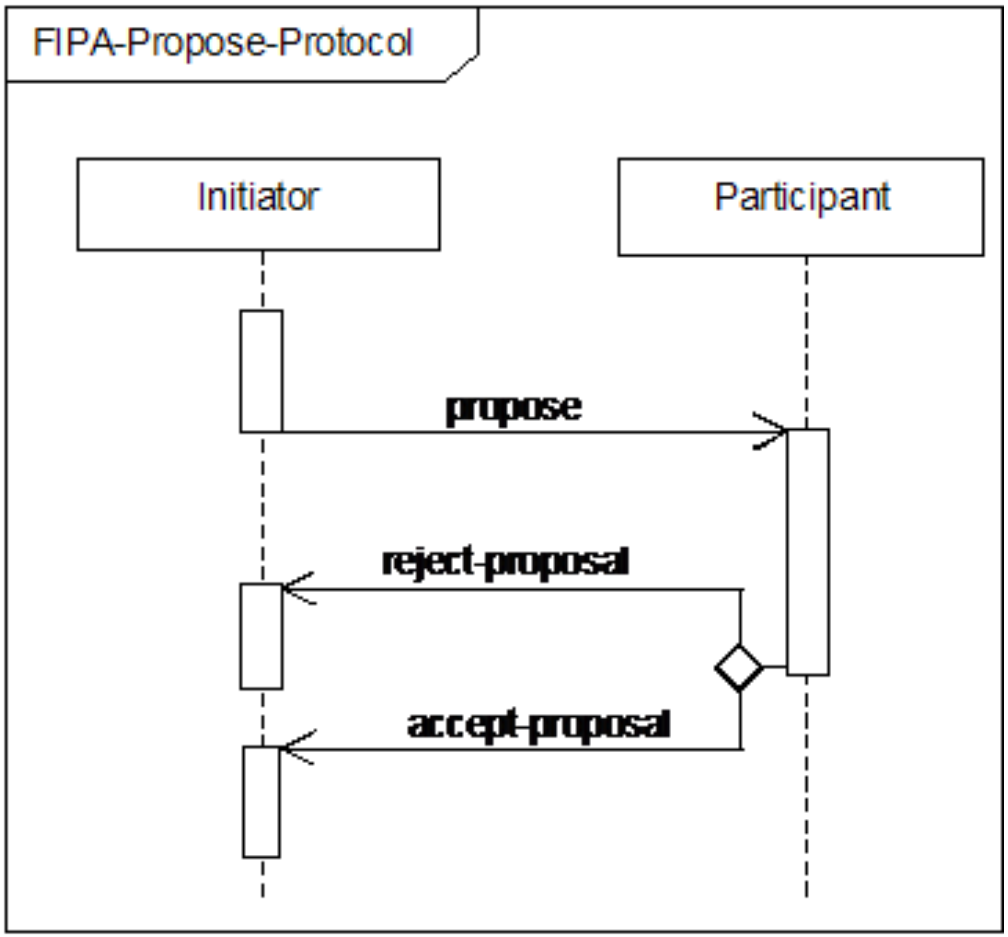
Query


FIPA-Query-Protocol
Initiator
Participant
query-if
query-ref
refuse
[refused]
agree
[agreed and
notification necessary]
failure
inform-t/f : inform
[query-if]
[agreed]
inform-result : inform
[query-ref]

Propose - 1

- Propose Interaction Protocol : il permet à un agent de proposer à un autre agent de faire l'action décrite lorsqu'il en recevra son acceptation.

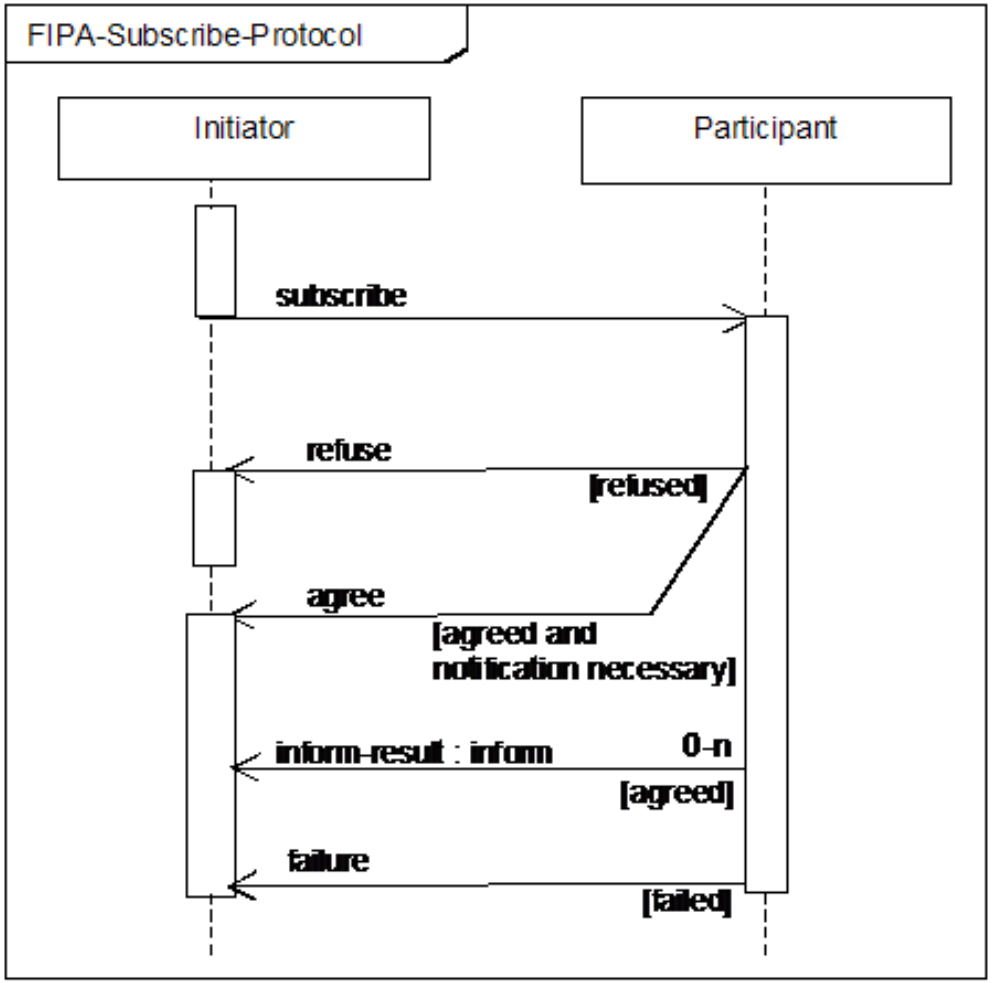
Propose - 2


FIPA-Propose-Protocol
Initiator
Participant
propose
reject-proposal
accept-proposal

Subscribe - 1

- Subscribe Interaction Protocol : il permet à un agent de souscrire auprès d'un autre agent d'être informé du changement de l'état d'un objet référencé.
- Les conditions peuvent indiquer un accord explicite ou optionnel.
- Le Participant pépond en envoyant dès que l'objet souscrit change des messages inform-result dont le contenu réfère à l'objet souscrit.

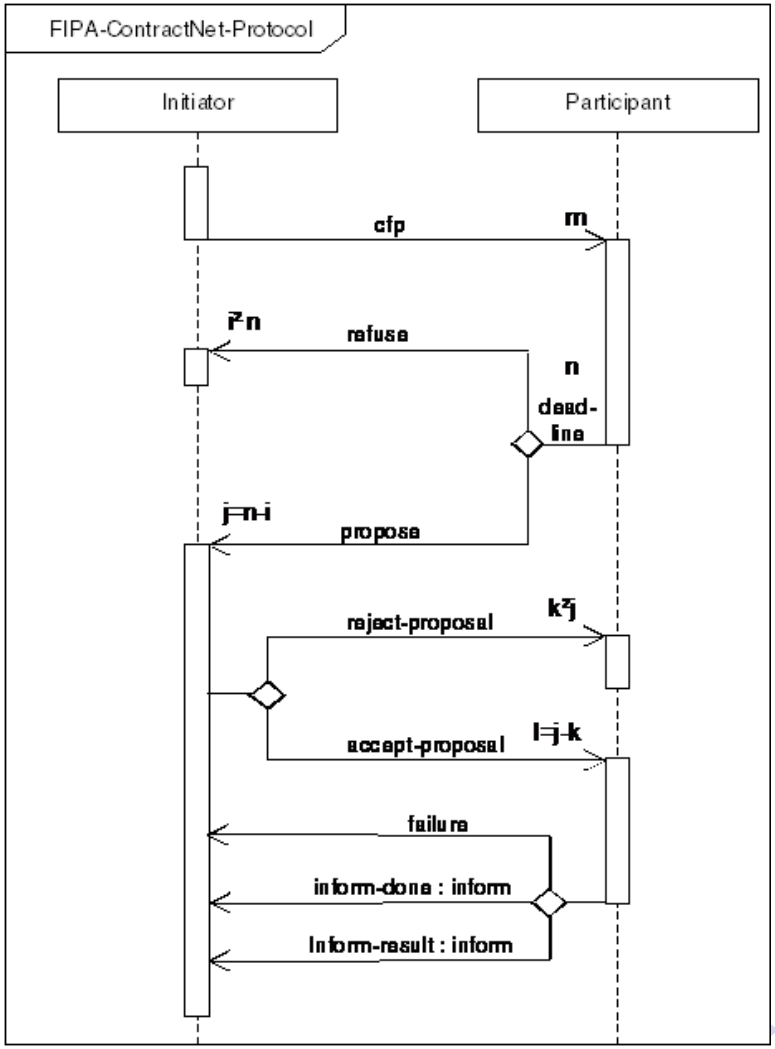
Subscribe - 2


FIPA-Subscribe-Protocol
Initiator
Participant
subscribe
refuse
[refused]
agree
[agreed and
notification necessary]
inform-result : inform
0-n
[agreed]
failure
[failed]

Contract Net - 1

- Contract Net Interaction Protocol : un agent (l'Initiateur) prend le role de manager qui désire qu'une tâche soit exécutée par un ou plusieurs autres agents (les Participants).
- L'initiateur désire également optimiser une fonction qui caractérise la tâche.
  - ex : un prix, une date au plus tôt.
- Pour une tâche donnée un nombre quelconque d'agents Participants peuvent répondre avec une proposition d'autres peuvent refuser.
- La négociation continue ensuite avec les Participants qui ont fait une proposition.

Contract Net - 2



Sommaire

Initiator

- JADE fournit la classe AchieveREInitiator qui simplifie l'écriture du behaviour jouant le rôle de l'agent initiateur dans un acte de type REQUEST.
- Elle prend en constructeur le message à envoyer.
- Elle fournit les méthodes gérant les réponses des destinataires du message.
- Cette classe peut facilement être étendue en surchargeant une ou plusieurs de ces méthodes.

Message

```
public class FactREAgent extends Agent {
 private int max;
 int level = 2;
 double result = 1;
 ...
 private ACLMessage multMessage(int l,double result) {
  ACLMessage request =
    new ACLMessage(ACLMessage.REQUEST);
  request.setProtocol(
   FIPANames.InteractionProtocol.FIPA_REQUEST);
  request.addReceiver(<un agent>);
  request.setContent(l+"x"+result);
  return request;
 }
}
```

Agent setup

```
public class FactREAgent extends Agent {
 private int max;
 int level = 2;
 double result = 1;
 protected void setup() {
  max = (Integer)getArguments()[0];
  addBehaviour(
    new FactREInitiator(
       this,multtMessage(level,result)));
 }
}
```

Behaviour

```
private class FactAchieveREInitiator
                    extends AchieveREInitiator {
  public FactREInitiator(Agent a, ACLMessage msg) {
   super(a, msg); }
  @Override
protected void handleInform(ACLMessage inform) {
   result = Double.parseDouble(inform.getContent());
   level++;
   if (level <= max)
    addBehaviour(new FactREInitiator(myAgent,
                        multMessage(level,result)));
   else {
    System.out.println("fact(" + max + ") = " + result);
   }
  }
}
```