22

Performance evaluation of protocols of multiagent information retrieval systems

Zofia Kruczkiewicz
Wrocław University of Technology, Wybrzeże Wyspiańskiego 27
Poland

1. Introduction

It is aimed at the limitation of the network traffic which is generated during the realization of protocol in distributed environment of multiagent systems. This paper presents the way of the decrease of the number and the way of the diminution of the size of messages sent between agents as the manner of the limitation of the network traffic. Multiagent system can be employed agents in searching information in distributed databases.
In literature, performance evaluation of the following multi-agent systems: *ZEUS* (Camacho et al., 2002), *JADE* (Camacho et al., 2002), *Skeleton Agent* (Camacho et al., 2002), *Aglets IBM* (Dikaiakos et al., 2001), (Yamamoto & Nakamura, 1999), *Concordia* (Dikaiakos et al., 2001), *Voyager* (Dikaiakos et al., 2001) is presented. When new MAS with performance requirements is designed then, according to software performance engineering (Smith, 1990), (Smith & Lloyd, 2002), (Wooldridge & Rao, 1999), (Babczyński et al., 2004), (Babczyński et al., 2005) performance requirements have to be considered at each phase of life cycle.
The paper presents the method of design communication protocols (FIPA00025, 2000), (FIPA00037, 2000), (Specification of JADE) which based on the limitation of the network traffic in multiagent systems as the use of dynamic protocols instead of static protocols embedded in system.
The method consists in assumption that agent communications and semantics based on Agent Communication Language (ACL), Interaction Protocols etc. (Odell et al., 2001), (Specification of FIPA), (Specyfication of JADE), (Pautret, 2005-2006). At a basic level, it is checked syntactic and semantic consistency of received and sent messages (for example, it is not possible to send an *agree* message with a content representing an action instead of a proposition). The behaviour of an agent is based on analyzing incoming messages and on drawing the conclusion from this analyze. Therefore, the agent semantics consists in setting up its initial beliefs, the rules for handling its beliefs, the domain-specific actions it is expected to handle, and in customizing its cooperative abilities. This way of designing agents of multiagent information retrieval systems will simplify adapting their protocols to the actual meaning of the exchanged messages. These protocols are named the dynamic protocols. This semantic adjustment of agent protocols during exchanging messages will improve the performance of retrieval tasks.

Each agent is built of components, which analyze the received or sent messages. If it is the incoming message, the agent chooses the action and communication act with regard to possibility of minimal loss of performance. It means that an agent sends messages to these agents who understand the content of messages and they are able to realize instructions of messages.
The method of the agent components consists in processing six lists of semantic and interpretation expressions:

- the lists of semantic expressions are FIPA-SL (FIPA00008, 2000), (FIPA00070, 2000) expressions that represent the sense of the message according to the FIPA-ACL semantics (FIPA00037, 2000)
- the belief knowledge, uncertainty knowledge and intention knowledge are lists of expressions that store the beliefs, uncertainties and intentions of the agent. A belief, uncertainty or intention can be a simple predicate, or a more complex formula;
- the lists of semantic actions store the description of the model components of communicative act. The reasons for which the act is selected are referred to as the rational effect (RE) and the conditions that have to be satisfied for the act to be planned are related to as the feasibility preconditions (FPs) (FIPA00037, 2000). They are qualifications for the act. Each action is associated to a semantic behaviour that performs the action;
- the lists of interpretation components perform semantic adjustment of the agent protocol.

Alike just the interpretation algorithm in (Pautret, 2005-2006), the presented interpretation algorithm of the content of semantic components is based on messages and on internal events the form of which are SL expressions. It is a loop, which applies all possible interpretation components to all existing semantic expressions and evaluates the performance of select dynamic protocols and stops when one of the following conditions holds:

- the semantic expressions list becomes empty,
- no interpretation components can be further applied to any semantic expression,
- the one of the semantic expressions is a false formula (e.g., in the case of inconsistency).

The chapter also presents the performance experiments enabling the comparison the dynamic protocols.
The paper is organized as follows. In Section 2, the project of multiagent information retrieval system adapting their protocols to the actual meaning of the exchanged messages is presented. In Section 3, performance metrics of protocols are defined and the methods of formation of the dynamic protocols are presented from performance point of view. The results of simulation experiments are described. Then there are conclusions.

2. System design

This section presents the project of retrieval multiagent system taking benefit from the semantic dimension of the FIPA-ACL language (FIPA00037, 2000), that applies in systems for presentation of interaction between agents. Language of agent includes communication acts so called a message and the grammar (FIPA00008, 2000). The project is done in MaSE technology (Deloach et al., 2001).

2.1 Concepts

It takes advantage of MaSE technology for identification of agents. During analysis, in AgentTool environment of MaSE technology (Deloach et al., 2001), model of goals is built (fig.1). Goals diagram expresses the mission fulfilled by the system. Then the use cases model is created, which presents interactions between roles using the sequence diagram (fig.2, fig.3). In next step of modeling multiagent system, the roles diagram is created, expressed the roles of system, executable tasks of each role and protocols of interactions between tasks (fig.4). The plan of each task is modeled by using the statecharts diagram.
The first step of formation of the design model is built the agent template diagram (fig.5). This diagram is created by the arbitrary choice of agents relate with individual roles they become tasks of agents - task role. Next, it is possible to generate the design model on base of the analysis model and the agent diagram, automatically. Diagram of agent architecture is built of components (fig.6) which represent agent tasks as classes on the design level. The conversation sends a message to another agent or receives a message from another agent (fig.7). The statecharts diagram of the task on analysis level is transformed to the statecharts diagram of the relative component on the design level (fig.8-14). The trigger as the communication act of the statecharts diagram of the task is transformed to the action as the conversation of the relative transition of the component statecharts diagram.

The goals diagram of the retrieval multiagent system

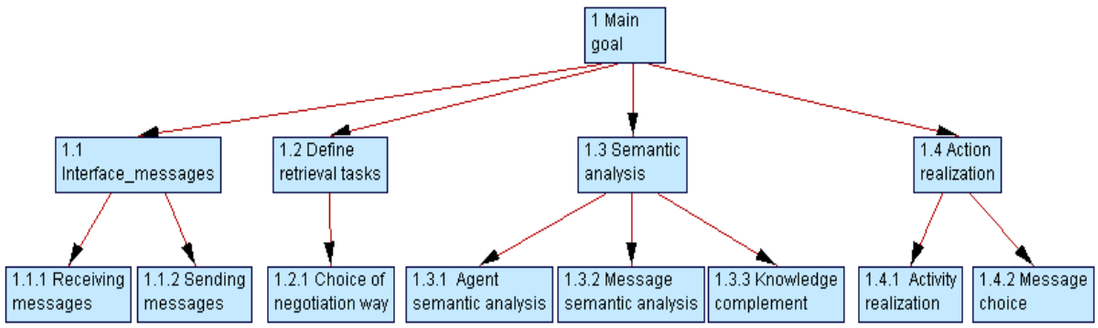

Fig. 1. The goals diagram

The goals diagram belongs to the analysis model. The *Main* goal of the retrieval multiagent system is adapting their protocols to the actual meaning of the exchanged messages because it is the way of the limitation of the network traffic which is generated during the realization of protocol in distributed environment of multiagent systems. The *Main* goal consists of a few goals of the first level. One of them is an *Interface_messages* which consists of a goal of the interface of the *Receiving messages* and a goal of the interface of *Sending messages*. The second goal of the first level is to define the retrieval tasks which include the goal of the choice of the negotiation way. The third goal of the first level is the *Semantic analysis* of communication acts at the point of view of agent properties. It consists of the three following goals. The *Message semantic analysis* goal aims at taking better benefit from semantic dimension of the FIPA-ACL language. The *Agent semantic analysis* goal takes into account the belief knowledge, uncertainty knowledge and intention knowledge in choice of the actions and the communication acts. The third *Knowledge complement* goal refers to gathering the knowledge of an agent properties and facts of actions and communicative

acts. The fourth *Action realization* goal executes the communicative acts or internal event. This goal consists of two goals. The first *Activity realization* goal refers to the realization of activity of the task. The second *Message choice* goal deals with the choice of an appropriate message which would be sent to another agent. This is a choice at a point of view of the result of the agent action and the retrieval tasks.

The use cases and sequence diagrams of the retrieval multiagent system

The use cases and its sequence diagrams belong to the analysis model, too. The each use case presents the scenario of the multiagent system. The sequence diagram represents these interactions of system in a more formal way. The agents of the system fulfil the different roles. Each role accomplishes a few tasks.

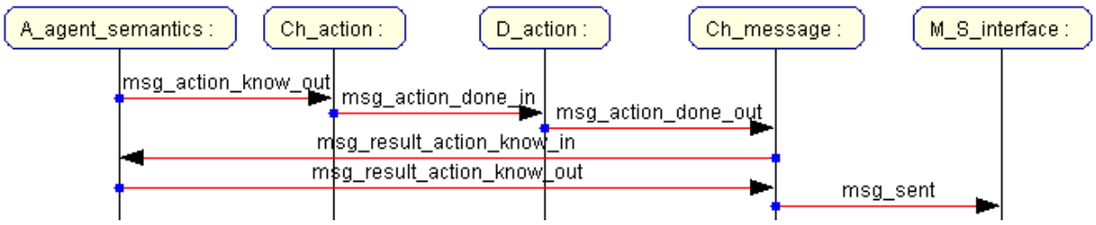

Fig. 2. Sequence diagram of sent process of the *send* use case

The sequence diagram of the *send* use case (fig.2) represents the scenario of fulfilling the process of sending communication acts, which is represented by the interactions between the roles of system over the protocols (fig.4). The *A_agent_semantics* role sends *msg_action_know_out* message by use the *Knowledge_complement* task of this role. There are two reasons of sending this message. The first one is the preparing the response of message received from another agent. The second reason refers to the initiation of the new action of agent.
The received message represents the knowledge of the agent need of activity of system for searching information. The content of the sent message includes these facts of the agent. This message is received by the *Action_choice* task of the *Ch_action* role. The *Action_choice* task choices the proper action which has to be done by the *Action_done* task of the *D_action* role. The *Action_choice* task choices the action by use the belief, uncertainty and intention knowledge of the agent. Each agent has itself knowledge.
Then the *Action_choice* task sends the *msg_action_done_in* message to the *Action_done* task of the *D_action* role. The received message gives information for executing the action or no principles for doing anything. Then the *Action_done* task of the *D_action* role sends the *msg_action_done_out* message to the *Message_choice* task of the *Ch_message* role. This task choices the appropriate message for the result of the *Action_done* task of the *D_action* role. The *Message_choice* task sends the *msg_result_action_know_in* message to the *Knowledge_complement* task of the *A_agent_semantics* role and receives the *msg_result_action_know_out* message with information supporting the choice of the proper message. At last the *Message_choice* task sends the chosen *msg_sent* message to the *Message_sending* task of the *M_S_interface* role.
The process of sending and receiving the *msg_in* result message to another agent is presented by the sequence diagram of the *receive* use case (fig.3) and role diagram (fig.4).

The *Message_sending* task of the *M_S_interface* role of the *ag1* agent sends the *msg_in* message to the *Message_receiving* task of the *M_R_interface* role of the *ag2* agent. The next interactions occur between tasks of roles of the *ag2* agent. The message as the communication act of FIPA ACL language (FIPA00037, 2000) is transformed to FIPA-SL expression (FIPA00008, 2000). The expression represents the semantics of the message.
Then the *msg_sl* message is sent to the *Message_sem_analysis* task of the *A_message_semantics* role. The reasons for which the act is selected are referred to as the rational effect (RE). The conditions that have to be satisfied for the act to be planned are related to as the feasibility preconditions (FPs). They are qualifications for the act (FIPA00037, 2000). Each action is associated to a semantic behaviour that performs the action. The *msg_know_in* message and the *msg_know_out* messages are sent between the *Message_sem_analysis* task of the *A_message_semantics* role and the *Knowledge_complement* task of the *A_agent_semantics* role.

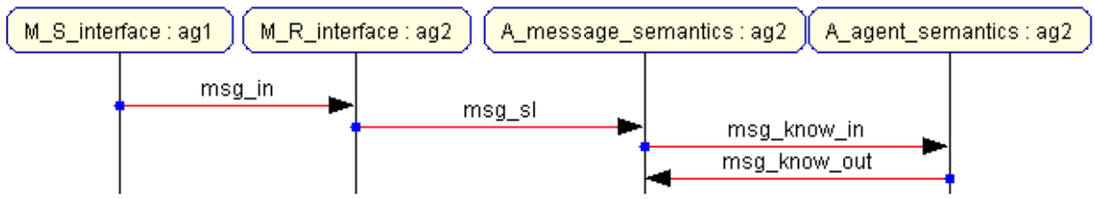

Fig. 3. Sequence diagram of receive process of the *receive* use case

The role diagram of the retrieval multiagent system

The role diagram is the next diagram of the analysis model. The role diagram expresses the roles of the system, executable tasks of each role and protocols of interactions between tasks (fig.4). Each agent of the system fulfils a few roles or the one role at least. These roles are defined at the sequence diagrams. Each role accomplishes different tasks.
The tasks exchange the messages over the protocols. For example, the *Message_sem_analysis* task sends the *msg_know_in* message to the *Knowledge_complement* task and receives the *msg_know_out* message from this task over *message_know* protocol. They are presented at the following diagrams: the sequence diagram of *receive* use case (fig.3), two statecharts diagrams – the *Knowledge_complement* task (fig.11) and the *Message_sem_analysis* task (fig.10), and the role diagram (fig.4).

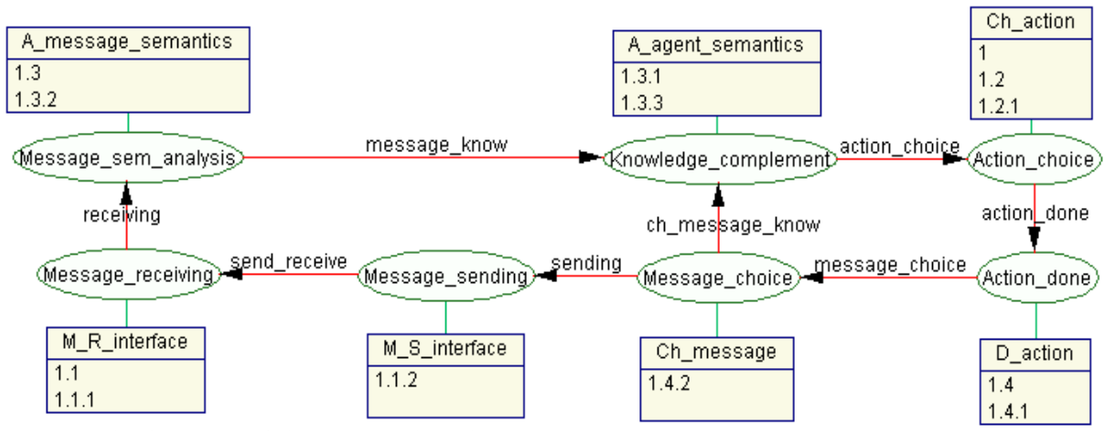

Fig. 4. Role diagram of the retrieval multiagent system

2.2. Design

On the figure 5 there is the *Agent Template Diagram.* It represents the roles fulfilled by each agent. The agent is the universal program, which realizes the same tasks by use itself belief, uncertainty and intention knowledge gathered during the life cycle. Therefore the each agent has the following features: flexibility in adaptation to environment changes, possibility of increasing the number of functions, mobile data and programs, etc. The *send_receive1* and *send_receive2* conversations are used to exchange the messages. The order of these messages realizes the dynamical protocols. The behaviour of the agents will simplify adapting their protocols to the actual meaning of the exchanged messages.

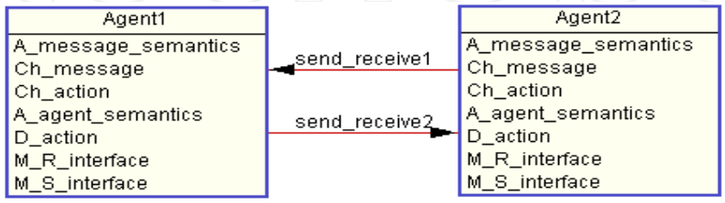

Fig. 5. Agent template diagram

The fig.6 shows the architecture diagram for each agent, which includes components transformed from the task of roles fulfilled by the each agent.

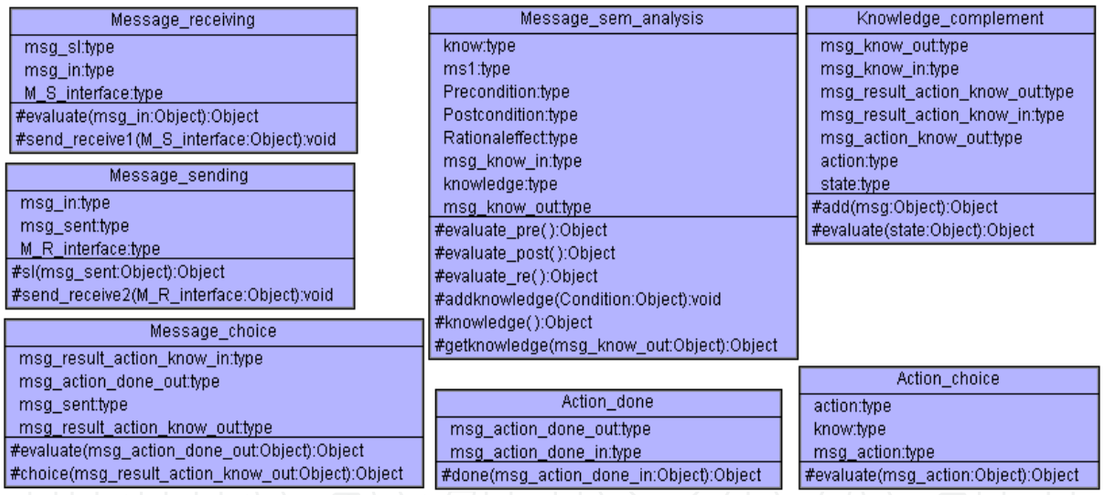

Fig. 6. Architecture diagram for each agent

On the figure 7, it is presented statecharts diagrams of the *send_receive1* and *send_receive2* conversations. Each conversation consists of two statecharts diagrams (Deloach et al., 2001), (Specification of UML): *send_receive1* or *send_receive2 Initiator* (the left statechars diagram of fig.7) and *send_receive1* or *send_receive2 Responder* (the right statechars diagram of fig.7).

Fig. 7. Statecharts diagrams of *send_receive1* or *send_receive2 Initiator* conversation (the left statecharts diagram) and *Responder* conversation (the right statecharts diagram).

The statecharts diagram of the *Message_receiving* component (fig.8) represents the actions executing during the receiving process.

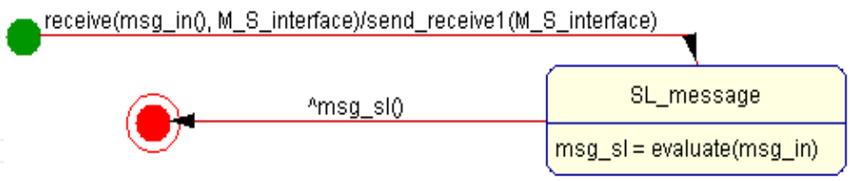

Fig. 8. Statecharts diagram of *Message_receiving* component

The statecharts diagram of the *Message_sending* component represents the actions executing during the sending process. The *send_receive1* and *send_receive2* conversations are used as the action of the transition of the statecharts diagram of the *Message_receiving* (fig.8) and *Message_sending* components (fig.9).
When the agent receives the *msg_in* message (fig.8), the *send_receive1 Responder* conversation is created. When the agent wants to send the message (fig.9), the *send_receive2* conversation *Initiator* is created.

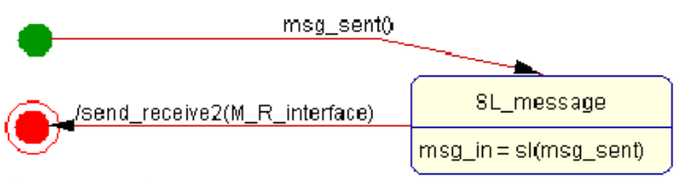

Fig. 9. Statecharts diagram of the *Message_sending* component

The statecharts diagram of the *Message_sem_analysis* component (fig.10) presents the process of interpreting of the received message.

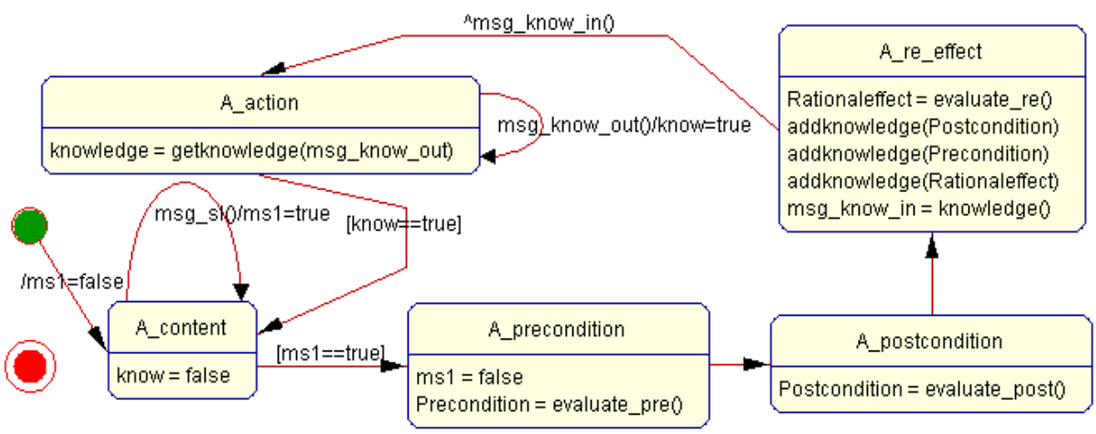

Fig. 10. Statecharts diagram of *Message_sem_analysis* component

The *Message_sem_analysis* component produces the data structure of the semantic representation handled by the mechanisms of gathering knowledge of the *Knowledge_complement* component and interpretation formulas of the following components: *Knowledge_complement*, *Action choice*, *Action_done*, *Message_choice*.

The data structure of the semantic representation defined by the following attributes represented by the SL formulas that represents the sense of the received message:
- a term that represents the agent of the action;
- the feasibility precondition FP, which represents a condition that must hold for an agent to be able to perform the action;
- the rational effect RE, which represents a state intended by the agent performing the action;
- the semantic behaviour, which implements its performance by the agent.

A communication act model of the request act is presented as follows (FIPA00037, 2000):
<i, request (j, a)>
FP: φ1
RE: φ2
where *i* is the agent of the act, *j* the recipient, *request* the name of the communication act, *a* stands for the semantic content or propositional content1, and φ1 and φ2 are propositions. This notational form is used for brevity, on the formal basis of ACL. The correspondence to the standard transport syntax (FIPA00070, 2000) adopted above is illustrated by a simple translation of the above example:
(request
:sender i
:receiver j
:content
a)
FP: FP (a) [i\j] ∧ Bi Agent (j, a) ∧ ¬ Bi Ij Done (a)
RE: Done (a)
where FP(a) [i\j] denotes the part of the FPs of a which are mental attitudes of i.

Example 1 (FIPA00037, 2000): Agent *i* requests *j* to reserve a ticket for *i*.
(request
:sender (agent-identifier :name i)
:receiver (set (agent-identifier :name j))
:content
"((action (agent-identifier :name j)
(reserve-ticket LHR MUC 27-sept-97)))"
:protocol fipa-request
:language fipa-sl
:reply-with order567)

The *Knowledge_complement* component (fig.11) stores facts believed by the agent and uncertain and intentional facts for this agent, according to the specific application domain. These facts are defined as SL semantic formulas. The *Knowledge_complement* component can only store and retrieve facts without actually interpreting their meaning. For instance, if an empty component is informed that (ac12345 100) there are 100 pounds in its ac12345 account, then agent *j* will correctly answer a Request-When message about (Done (reserve-ticket LHR MUC 27-sept-97, ac12345 100)) meaning about "Agent *i* tells agent *j* to notify it as soon as a reserve-ticket occurs and there are 100 pounds in its account just before that.".

However, it will not be able to answer a Request-Whenever message about (Done (reserve-ticket LHR MUC 27-sept-97, (> (price-ticket) 100))) meaning about "Agent *i* tells agent *j* to notify it whenever a reserve-ticket occurs and the price of tickets is lower than 100 before that.", because agent *j* cannot guess the semantic relationships between the ac12345 and the price-ticket predicates. This component includes the different operations towards the belief uncertainty and intention bases, which are needed by the interpretation process of other components.

These knowledge bases would have the following properties:

- storing the facts believed by the agent and uncertain and intentional facts for the agent (the *knowledge_message_out* state);
- providing a mechanism of listening changes in the contents of the knowledge bases by using formulas (the *knowledge_message_in* state);
- initializing the process of the internal action as the response of received message or as the initiation of the new task of the agent (the *knowledge_action* state).

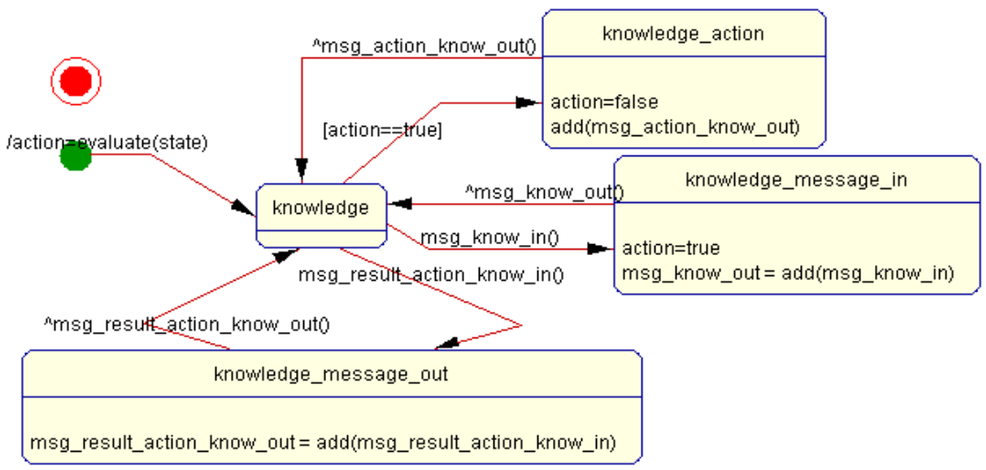

Fig. 11. Statecharts diagram of the *Knowledge_complement* component

The *Action_choice* component (fig.12) supports choices of the internal and the ontological actions.

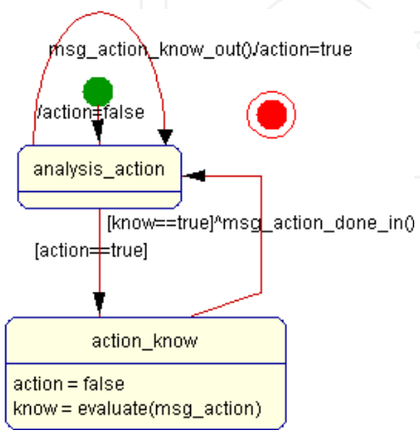

Fig. 12. Statecharts diagram of the *Action_choice* component

The ontological actions gather the actions related to the specific application. The internal actions gather the actions related to the response of the content of the received message or to a specific application. The action choice is based on the interpretation of the belief knowledge, uncertainty knowledge and intention knowledge and on the rules of FIPA- ACL language. The semantic interpretation formulas are related to the behaviours of the agent. They provide the interpretation mechanism of semantic SL formulas. Each semantic interpretation formula is applicable on a specific SL pattern. The use of semantic interpretation formulas is based on querying and updating the knowledged bases as the first result. The second result of the interpretation is based on adding to, or removing behaviours from the agent.

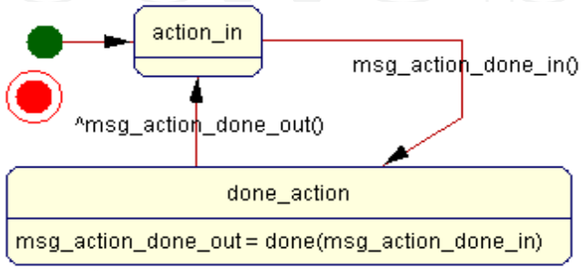

Fig. 13. Statecharts diagram of the *Action_done* component

The *Action_done* component (fig.13) executes the internal action choiced by the *Action_choice* component during the interpretation process. Then it delivers the results of this action to the *Message_choice* component. These results are defined in SL formulas.

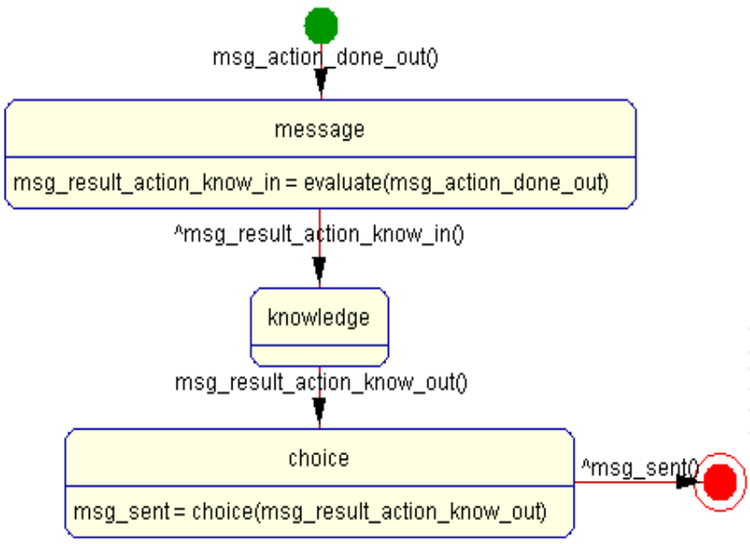

Fig. 14. Statecharts diagram of the *Message_choice* component

The *Message_choice* component (fig.14) choices the communicative act as the response for the received message or as the initialization of the interaction with other agents related to a specific application. The communicative actions gather all the FIPA Communicative Acts. The message choice is based on the semantic interpretation of the belief, uncertainty and intention knowledge databases and on the rules of FIPA- ACL language. The message

choice uses the semantic interpretation formulas that are based on querying and updating the knowledge bases as the first result. The second result of the interpretation is based on adding to, or removing behaviours from the agent.
The example 2 resumes the example 1: if the agent is able to do the action included in the content of the received message as the *request* communicative acts, it sends the *agree* message to the sender agent:
<i, agree (j, <i, request>, φ))> ≡ <i, inform (j, Ii Done (<i, act>, φ))>
FP: Bi α ∧ ¬ Bi (Bifj α ∨ Uifj α)
RE: Bj α
where: α = Ii Done(<i, act>, φ)
The example 3 continues the example 1: the agent *j* answers that it agrees to reserve a ticket for *i*, since there are sufficient funds in its account.

```
(agree
:sender (agent-identifier :name j)
:receiver (set (agent-identifier :name i))
:content
  "( (action (agent-identifier :name j)
     (reserve-ticket LHR MUC 27-sept-97))
     (sufficient- funds ac12345))"
:in-reply-to order567
:protocol fipa-request
:language fipa-sl)
```

The example 4 continues the example 2: if the agent is able to do the action defined in the content of the received message as the *request* communicative acts, it sends the *refuse* message to the sender agent: Agent *i* informs *j* that action act is not feasible, and further that, because of proposition *φ*, act has not been done and *i* has no intention to do act.
<i, refuse (j, <i, act>, φ)> ≡ <i, disconfirm (j, Feasible(<i, act>))>;
<i, inform (j, φ ∧ ¬ Done (<i, act>) ∧ ¬ Ii Done (<i, act>))>
FP: Bi ¬ Feasible (<i, act>) ∧ Bi (Bj Feasible (<i, act>) ∨
Uj Feasible (<i, act>)) ∧ Bi α ∧ ¬ Bi (Bifj α ∨ Uifj α)
RE: Bj ¬ Feasible (<i, act>) ∧ Bj α
where: α = φ ∧ ¬ Done (<i, act>) ∧ ¬ Ii Done (<i, act>)
Example 5 continues the example 3: Agent *j* answers that it refuses to reserve a ticket for *i*, since there are insufficient funds in its account.

```
(refuse
:sender (agent-identifier :name j)
:receiver (set (agent-identifier :name i))
:content
"((action (agent-identifier :name j)
  (reserve-ticket LHR MUC 27-sept-97))
  (insufficient- funds ac12345))"
:in-reply-to order567
:protocol fipa-request
:language fipa-sl)
```

The process of semantic interpretation enables to simplify adapting their protocols to the actual meaning of the exchanged messages.

3. Communication protocols

It is possible to create protocol for exchanging most appropriate messages between agents. These messages represent the role of the agent in community of agents taking part in conversations. The table 1 presents the properties of communicative acts.

Communitive act	Information passing	Requesting information	Negotiation	Action performing	Error handling
accept-proposal			+		
agree, cancel				+	
cfp			+		
confirm, disconfirm	+				
failure, not-understood					+
inform, inform-if (macro act), inform-ref (macro act)	+				
propose			+		
query-if ,query-ref		+			
refuse				+	
reject-proposal			+		
request, request-when, request-whenever				+	
subscribe		+			

Table 1. Communicative acts (Specification of FIPA, 2000)

3.1. Dependency between messages

This section presents the list of communicative acts with a concise description (tab.2). The *cancel, request, request-when, request-whenever* messages can be sent by the agent fulfilling superior role. This agent demands the execution action of the subordinate agent.

If there is the client-server relationship between agents, *subscribe, cfp, accept-proposal, query-if, query-ref* messages are sent by client agent concerned with making of cooperation with recipient agent. The recipient agent executes the task for sender agent. The client agent sends the *reject-proposal* message if it is not interested in services of the recipient agent. This is the response of the received *propose* message, that is sent by the recipient agent.

The *inform, inform-if(macro act), inform-ref (macro act)* messages are sent by agent that remains in neutral relationship with other agents. If sender assums, that recipient agent cannot believe to received information, then this receiver sends the *confirm* message to confirm acceptance of received information. If however, the sender assums that the recipient agent believes to received information, the receiver agent sends the *disconfirm* message, if it rejects this information. Subordinate or servicer agent can send the *agree* message as the answer to the *request* or the *subscribe* message from the superior or client agent. The *refuse* message can be sent by the subordinate or the client as the answer of the *request, subscribe* or *cfp* message from the superior or the client agent, if it declines execution of the action. The *failure*

message is sent by the subordinate agent, which cannot execute the action. In opposite case, if executable task will end progress, the subordinate agent sends the *inform* message as the result to the superior agent. Independently on the role, the each agent can send the *not-understood* message if it not understand the message that other agent has just sent to it.

Communicative acts	Description
accept proposal	The action of accepting a previously submitted proposal to perform an action.
agree	The action of agreeing to perform some action, possibly in the future.
cancel	The action of one agent informing another agent that the first agent no longer has the intention that the second agent performs some action.
cfp - call for proposal	The action of calling for proposals to perform a given action.
confirm	The sender informs the receiver that a given proposition is true, where the receiver is known to be uncertain about the proposition.
disconfirm	The sender informs the receiver that a given proposition is false, where the receiver is known to believe, or believe it likely that, the proposition is true.
failure	The action of telling the another agent that an action was attempted but the attempt failed.
inform	The sender informs the receiver that a given proposition is true.
inform if (macro act)	A macro action for the agent of the action to inform the recipient whether or not a proposition is true.
inform ref (macro act)	A macro action for sender to inform the receiver the object which corresponds to a descriptor, for example, a name.
not understood	The sender of the act (for example, i) informs the receiver (for example, j) that it perceived that j performed some action, but that i did not understand what j just did. A particular common case is that i tells j that i did not understand the message that j has just sent to i.
propose	The action of submitting a proposal to perform a certain action, given certain preconditions.
query if	The action of asking another agent whether or not a given proposition is true.
query ref	The action of asking another agent for the object referred to by a referential expression.
refuse	The action of refusing to perform a given action, and explaining the reason for the refusal.
reject proposal	The action of rejecting a proposal to perform some action during a negotiation.
request	The sender requests the receiver to perform some action. One important class of uses of the request act is to request the receiver to perform another communicative act.
request when	The sender wants the receiver to perform some action when some given proposition becomes true.
request whenever	The sender wants the receiver to perform some action as soon as some proposition becomes true and thereafter each time the proposition becomes true again.
subscribe	The act of requesting a persistent intention to notify the sender of the value of a reference, and to notify again whenever the object identified by the reference changes.

Table 2. Characteristics of messages of ACL language (FIPA00037, 2000)

Such dependencies between messages allow designing effective protocols of communications, without redundancy in sent message. It is the first way of improve performance of multiagent systems. The limitation number of messages is the second way to correct performance of protocols.

3.2. Performance evaluation of interaction protocols

Applications working in distributed environment use the distributed data and the distributed clients. Clients send a lot of messages during realizing operations by using e-mail, web services, communicators programs etc. Therefore these applications generate the large network traffic lowering performance of the internet applications.

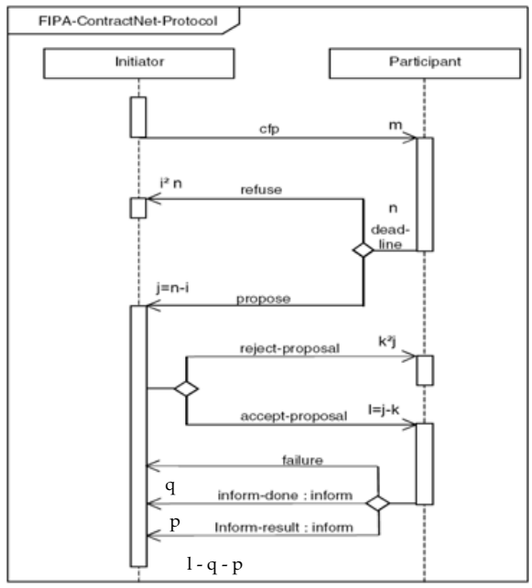

Fig. 15. The interaction diagram of *FIPA-Contract-Net* protocol (FIPA00029, 2002)

In section 2, it is introduced agents, which adapt their communicative acts (FIPA00037, 2000) to actual state of knowledge databases, main goal of application and character of interactions between agents. These interactions have the dynamic character of protocols based on negotiations, subscriptions, auctions. This sending proper sequence of messages is related to the state of realized task. As a proof that the dynamic protocols can limit the network traffic, it is presented two protocols as the products of adapting the communicative acts. These protocols are *FIPA-Contract-Net* (fig.15) (FIPA00029, 2002) and user defined protocol as the *Order-Request-NET* (fig.16) based on *FIPA-Request-Net* protocol (FIPA00026, 2002). The protocols are defined by use interaction diagram (FIPA00025, 2000), (Odell et al., 2001). If the choices of the sequences of messages as the protocol are proper, an agent can reduce the network traffic by sending messages only to these agents who are be able to collaborate over executing the task.

The *FIPA-Contract-Net* protocol (FIPA00029, 2002) is a small modification of the original protocol by added the reject and accepts communicative acts. The *initiator* agent fulfils the role of manager in this protocol. The *initiator* sends a *call for proposals* act (*cfp*) to m agents.

These messages are received by *n participants* of the protocol. They send the *proposal* message which specifies the task, as well as any conditions the *initiator* is placing upon the execution of the task or *refuse* message if they decline the proposal for the *initiator*. The *initiator* receives *i* the *refuse* messages and *j=n-i* the *proposal* acts. The *participant* finishes protocol if it sends the *refuse* message. Therefore *2i* messages generate the needlessly network traffic.
The manager evaluates *j* proposals according to certain function and accepts *j-k* (*n-i-k*) proposals. This estimate of proposal messages is expressed behind assistance of price, time etc. The *initiato*r sends *j-k accept-propose* messages to accepted *participant* and *k reject-proposal* messages to remaining *participants*. Therefore *3k* messages generate the needlessly network traffic.
At last the accepted *participants* send the result of the task by sending together *n-i-k-q inform-result* and *inform-done* messages. If *participant* cannot finish the task, it sends a *failure* message to the manager (*initiator*) - the *initiator* receives *q* failure messages. Therefore *4q* messages generate the needlessly network traffic. At last, it can be *2i+3k+4q* messages generating the needlessly network traffic.

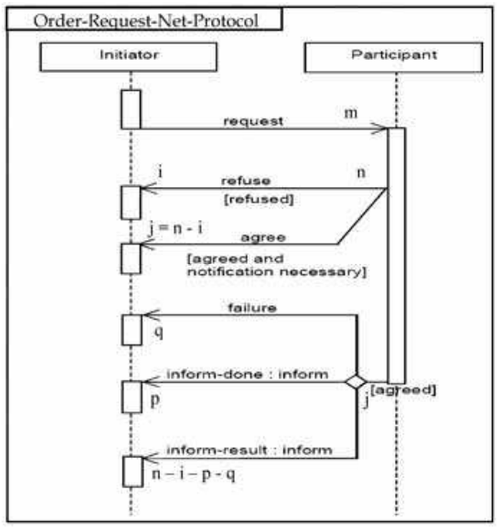

Fig. 16. The interaction diagram of user defined *Order-Request-Net* protocol (FIPA00026, 2002)

The *Order-Request-NET* protocol is based on the *FIPA-Request-Net* protocol (fig.16). It includes the informatiom about the multiplicity of participants. The *initiator* agent fulfils the role of the superior in this protocol. It sends *m request* messages with information about necessity of performing some action. These messages are received by *n participants* as subordinates. They send *agree* messages if they can perform the request action or *refuse* messages if they decline the request for the *initiator*. The *initiator* receives *i refuse* messages and *n-i agree* messages. Therefore *2i* messages generate the needlessly network traffic.
At last *participants* send the result of the task by sending together *n-i-q inform-result* and *inform-done* messages. If the *participant* cannot finish the task, it sends a *failure* message to the manager (*initiator*) - the *initiator* receives *q* failure messages. Therefore *3q* messages generate the needlessly network traffic. At last, it can be *2i+3q* messages generating the needlessly network traffic.

Meaning of messages and their possibility of occurrence during interactions allow to value number of messages in extreme and average cases (Smith, 1990), (Smith & Lloyd, 2002). It assumes that each message has the same size or the same transmission *t* time by network. Average duration transmission of all messages is evaluated by using the probability *p* of occurance messages in both protocols.
The *cfp* messages of the *FIPA-Contract-Net* protocol (tab.3) reach participants with the probability *p1*. The *refuse* message is sent with the probability *p2* and the *propose* message with *1-p2* probability. The *aceept-proposal* message is sent with propability *p3* and the *reject-proposal* message with the probability *1-p3*. At last the *failure* message is sent with the probability *p4*, the *inform-done* message with the probability *p5* and the *inform-result* message with probability *1-p4-p5*. The average duration *t1* of *FIPA-Contract-Net* protocol is equal

$$t1=tm(1 + p1(3-2p2–p3+p2p3)) \quad (1)$$

however, maximum time is *4tm* and it is minimal equal with *tm*.
The *Loss1* avarage of duration of messages of *FIPA-Contract-Net* protocol which are the types of *failure, refuse* and *reject-proposal* messages etc. is equal

$$Loss1=tm\ p1(2p2+3(1-p2)p4) \quad (2)$$

estimation of number of messages	The number of messages								
	cfp	refuse	propose	reject-proposal	accept-proposal	failure	inform-done	inform-result	total
All messages	m	i	n - i	k	n – i - k	q	p	n–i–k– q-p	m + 3n – 2i - k
Average t1 t=1	m	m p1 p2	m p1 (1-p2)	m p1 (1-p2) p3	m p1 (1-p2) (1-p3)	m p1 (1-p2) (1-p3) p4	m p1 (1-p2) (1-p3) p5	m p1 (1-p2) (1-p3) (1–p4-p5)	m (1 + p1 (3-2p2-p3 +p2p3))
Maximum p1=1 (m=n) p2=0 (i=0) p3=0 (k=0) p4=0 (q=0)	m	0	m	0	m	0	mp5	m(1-p5)	4m
Minimum p1=0 (n=0)	m	0	0	0	0	0	0	0	m
Loss1, t=1		2m p1 p2		3m p1 (1-p2) p3		4m p1 (1-p2) (1-p3) p4			m p1 (2p2+(1-p2) (3p3+ 4p4(1-p3)))

Table 3. Performance parameters of *FIPA-Contract-Net* protocol

The *request* message of the *Order-Request-Net* protocol (tab.4) reaches participants with the probability *p1*. The *refuse* message is sent with the probability *p2* and the *agree* message with the probability *1-p2*. At last the *failure* message is sent with the probability *p4*, the *inform-done* message with the probability *p5* and the *inform-result* message with probability *1-p4-p5* however. Average duration of protocol is equal

$$t2=tm(1 + p1(2-p2)) \quad (3)$$

however, maximum time is *3tm* and it is minimal equal with *tm*.
The avarage of duration of messages *Loss1* of the *Order-Request-Net* protocol which are the types of *failure* and *refuse* messages etc. is equal

$$Loss2= m\ p1(2p2+3(1-p2)p4) \quad (4)$$

The metric of lost messages *Lost1*, which excludes some agents from executing the task realizing over the *FIPA-Contract-Net* protocol is equal

$$Lost1=Loss1/t1 \quad (5)$$

The metric of lost messages *Lost2*, which excludes some agents from executing the task realizing over the *Order-Request-Net* protocol is equal

$$Lost2=Loss2/t2 \quad (6)$$

Measure of messages	The number of messages						
	request	refuse	agree	failure	inform-done	inform-result	total
All messages	m	i	n - i	q	p	n – i – q - p	m + 2n – i
Average t2 t=1	m	m p1 p2	m p1 (1-p2)	m p1 (1-p2) p4	m p1 (1-p2) p5	m p1 (1-p2) (1–p4-p5)	m (1 + p1(2- p2))
Maximum p1=1 (m=n) p2=0 (i=0) p4=0 (q=0)	m	0	m	0	mp5	m(1-p5)	3m
Minimum p1=0 (n=0)	m	0	0	0	0	0	m
Loss2, t=1		2m p1 p2		3m p1 (1-p2) p4			m p1 (2p2+3(1-p2)p4)

Table 4. Performance parameters of *Order-Request-Net* protocol

The table 5 presents the results of simulation experiments for estimation of the duration of both protocols.

Protocol	Evalution of transmission time	Duration			
		m=10	m=100	m=1000	m=10000
FIPA-Contract-Net	value of expression (1) (t=1)	t1=18.76	t1=187.67	t1=1876.60	t1=18761.15
	Minimum: m	10	100	1000	10000
	Maximum: 4m	40	400	4000	40000
	Loss1, value of expression (3)	12.26	112.59	1127.43	11264.69
	Lost1, value of expression (5)	0.6536	0.5999	0.6008	0.6004
Order-Request-Net	value of expression (2) (t=1)	t2=17.50	t2=175.14	t2=1751.38	t2=17510.79
	Minimum: m	10	100	1000	10000
	Maksimum: 3m	30	300	3000	30000
	Loss2, value of expression (4)	9.53	87.45	877.54	8756.15
	Lost2= value of expression (6)	0.5444	0.4993	0.5011	0.5000

Table 5. Results of performance experiments of *FIPA-Contract-Net* protocol and user defined protocol as the *Order-Request-NET* protocol

The number of messages sending over the protocols is calculated by using the generator of uniform distribution values. The experiment relied on 100000 repetition of measurements for number of 10, 100, 1000, 10000 agents. The time of transmission (*t*) of the one message is equal with the 1 time unit. The table 5 includes values of measurements of average values and values of maximum and minimum. Results of performance experiments in the table 5 and on the figure 17 are presented the influence of number of agents and their behaviour on performance of the multiagent system. For example, if the interpreting mechanism detects the relationship superior-subordinate, the *Order-Request-Net* protocol is better than *FIPA-Contract-Net* because of smaller generated network traffic. If the interpreting mechanism enables the selection of proper agents which are capable of executing actions of the task, the network traffic will be smaller because the number of the *refuse, failure and reject* messages can be reduced.

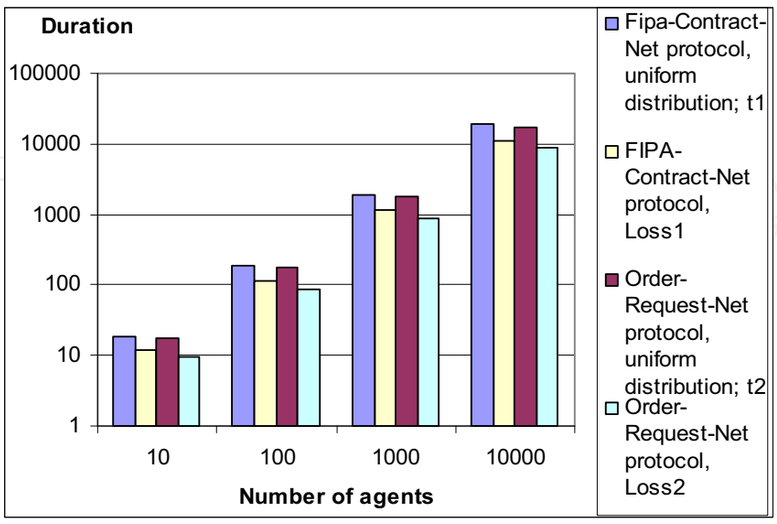

Fig. 17. The chart of results of performance experiments of *FIPA-Contract-Net* protocol and user defined protocol as the *Order-Request-NET* protocol

4. Conclusion

In the paper it is presented the design of multiagent system, which includes agents with special abilities. They are able to interpret the communicative acts and they can adapt to actual state of executing task. This is the way to choice the proper sequence of messages for example the *Order-Reqest-Net* protocol instead of the *FIPA-Contract-Net protocol*.
The other ability of an agent is reduction of the *refuse, failure, and reject* etc. messages by interpreting belief, uncertainty and intention knowledge databases. In the paper agent reduces the messages with propabilites generated from uniform distribution. In the future, it can complete the mechanism of interpreting messages with learning process of behaviours of agents collaborating over the tasks. These are reasons of reduction the network traffic and improving the performance of multiagent systems.
Results of performance experiments authenticate the influence of number of exchanging messages between agents and agent's abilities on performance of the multiagent system. Therefore, if the interpreting mechanism can detect properly the relationship between agent and their abilities, the generated network traffic can become smaller. The first reason is the possibility of choice of the protocol with smaller duration, for example the *Order-Reqest-Net* protocol instead of the *FIPA-Contract-Net protocol*. The second reason is the possibility of reduction the *refuse, failure, and reject* etc. messages by the selection of proper agents which are capable of executing action of the task.

5. References

Babczyński, T.; Kruczkiewicz, Z.; Magott, J. (2004). Performance Analysis of Multiagent Industrial System, *Proceedings of 8th International Workshop, CIA 2004, Cooperative Information Agents VIII*, LNAI 3191, pp. 242-256, ISBN-13 978-3-540-29046-9, Erfurth, Germany, September, 2005, Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin Heidelberg New York

Babczyński, T.; Kruczkiewicz, Z.; Magott, J. (2005). Performance Comparison of Multi-agent Systems, *Proceedings of 4th Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005, Multi-Agent Systems and Applications IV,* LNAI 3690, pp. 612-615, ISBN 3-540-23170-6, Budapest, Hungary, September, 2005, Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin Heidelberg New York

Bennett, A. J.; Field, A. J.; Woodside, C. M. (2004). Experimental evaluation of the UML Profile for Schedulability, Performance, and Time, In: *The Unified Modelling Language,* Baar, T. et al., eds, Lecture Notes in Computer Science, pp. 143-157, Vol. 3273, Springer, ISBN 978-3-540-23307-7, Berlin Heidelberg

Camacho, D.; Aler, R.; Castro, C.; Molina, J. M. (2002). Performance evaluation of ZEUS, JADE, and SkeletonAgent frameworks, *Proceedings of the 2002 IEEE Systems, Man, and Cybernetics Conference*, on page(s): 6 pp., ISBN: 0-7803-7437-1, October 2002

Deloach, S.A.; Wood, M.F.; Sparkman, C.H. (2001), Multiagents systems engineering, *International Journal of Software Engineering and Knownledge Engineering*, Vol 11, No.3 June 2001, pp. 231-258, World Scientific Publishing Company

Dikaiakos, M.; Kyriakou, M.; Samaras, G. (2001). Performance evaluation of mobile-agent middleware: A hierachical approach, *Proceedings of the 5th IEEE International Conference on Mobile Agents*, Picco, J.P. (ed.), Lecture Notes in Computer Science series, pp. 244-259, ISBN 978-3-540-42952-4, Springer, December 2001, Berlin Heidelberg

FIPA00008 (2000), FIPA SL Content Language Specification. Foundation for Intelligent Physical Agents, http://www.fipa.org/specs/fipa00008/

FIPA00025 (2000), FIPA Interaction Protocol Library Specification. Foundation for Intelligent Physical Agents, http://www.fipa.org/specs/fipa00025/

FIPA00026 (2002), FIPA Request Interaction Protocol Specification. Foundation for Intelligent Physical Agents

FIPA00029 (2002), FIPA Contract Net Interaction Protocol Specification. Foundation for Intelligent Physical Agents

FIPA00037 (2000), FIPA Communicative Act Library Specification. Foundation for Intelligent Physical Agents, http://www.fipa.org/specs/fipa00037/

FIPA00070 (2000), FIPA ACL Message Representation in String. Foundation for Intelligent Physical Agents, http://www.fipa.org/specs/fipa00070/

http://jade.cselt.it/doc/tutorials/SemanticsProgrammerGuide.pdf

Odell, J.; Parunak, V.D.H.; Bauer, B. (2001). Representing Agent Interaction Protocols in UML, In: *Agent-Oriented Software Engineering*, Ciancarini, P. and Wooldridge, M., Eds., pp. 121-140, Springer - Verlag, ISBN 3-540-41594-7, New York

Pautret V. (2005-2006). Jade Semantics Add-on Programmer's guide, *Research & Development*, Version: 1.0, France Telecom,

Smith, C.U. (1990). *Performance Engineering of Software Systems*, Addison – Wesley, ISBN 0-201-53769-9, United States of America

Smith, C.U.; Lloyd G.W. (2002). *Performance Solutions, A Practical Guide to Creating Responsive, Scalable Software*, Addison - Wesley, ISBN 0-201-72229-1, Canada

Specification of FIPA, http://www.fipa.org/specs/

Specification of JADE, http://sharon.cselt.it/projects/jade/doc/

Specification of UML, http://www.uml.org/

Wooldridge, M.; Rao, A. (Eds.) (1999). *Foundations of Rational Agency,* Kluwer Academic Publishers, ISBN 0-7923-5601-2, The Netherlands

Yamamoto, G.; Nakamura, Y. (1999). Architecture and performance evaluation of a massive multi-agent system, *Proceedings of the third annual conference on Autonomous Agents*, pp. 319-325, ISBN 1-58113-066-X, Seattle, 1999, ACM, New York, NY, USA

Engineering the Computer Science and IT

Edited by Safeeullah Soomro

ISBN 978-953-307-012-4

Hard cover, 506 pages

Publisher InTech

Published online 01, October, 2009

Published in print edition October, 2009

It has been many decades, since Computer Science has been able to achieve tremendous recognition and has been applied in various fields, mainly computer programming and software engineering. Many efforts have been taken to improve knowledge of researchers, educationists and others in the field of computer science and engineering. This book provides a further insight in this direction. It provides innovative ideas in the field of computer science and engineering with a view to face new challenges of the current and future centuries. This book comprises of 25 chapters focusing on the basic and applied research in the field of computer science and information technology. It increases knowledge in the topics such as web programming, logic programming, software debugging, real-time systems, statistical modeling, networking, program analysis, mathematical models and natural language processing.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen