Architecture of Reflective Artificial Intelligent Agents

Sergey Listopad
Kaliningrad Branch of the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences
Kaliningrad, Russia
ser-list-post@yandex.ru

Abstract—**Mechanisms of reflection in a team of specialists are special means of coordinating and synchronizing the activities of individual members. Thanks to them, team members are able to reason "for another", develop a solution, simulate its transmission to themselves and, in accordance with this solution, build their further reasoning. This allows significally reduce the duration and intensity of conflicts between team members. In this regard, relevant computer modeling of long-existing teams of specialists solving practical problems at a round table is impossible without modeling reflection. For this purpose, reflective-active systems of artificial heterogeneous intelligent agents are proposed. This paper considers the architectures of intelligent agents of various reflection ranks, as well as the functional structure of the system itself.**

I. Introduction

The traditional subject-object paradigm is irrelevant to the processes of managing complex organizational structures, such as logistics centers, energy distribution or medical organizations [1]. Such structures are not passive objects, but evolving networks of interacting subjects, which are characterized by activity, reflection, communicativeness, sociality, etc. When managing and automatically solving problems arising in these structures, it is necessary to consider them as self-developing reflective-active environments [1]. For this purpose, the concept of reflective-active system of artificial heterogeneous intelligent agents (RASAHIA) is proposed [2], within the multi-agent paradigm [3–5], as a computer model of a team managing complex organizational structures. RASAHIA's agents are autonomous software entities characterized by activity and reactivity, capable of reasoning, interacting and reflecting.

The foundations of mathematical modeling of reflective processes and control are laid in the works of V.A. Lefebvre, who considered reflection as the ability of the object-researcher to model other objects and itself, its actions and thoughts [6]. D.A. Novikov and A.G. Chkhartishvili proposed the concept of equilibrium in reflexive games [7]. In [8], an approach based on fuzzy logic was proposed for formalizing the reflection of a medical expert. In [9], the concept of constructing a virtual intelligent agent-assistant that reflectively models its user is considered. The research [10] is devoted to the development of self-learning mechanisms for autonomous intelligent robots when constructing a trajectory in an environment with obstacles unknown a priori.

In RASAHIA, the reflective abilities of agents serve to reduce the duration of negotiation processes during the development of a coordinated idea of the problem area, the goals of the system and the order of interaction, making it possible to attract new agents from the external environment and RASAHIA's self-organization in the strong sense [11]. The essential feature of RASAHIA is the presence of second-rank reflective agents, which suggests that the system may contain both agents without reflection and those capable of first-rank reflection [6]. Second-rank reflective agents try to distinguish reflection rank of other agents in the system based on an analysis of their behavior to provide own effective reflective control of both types of agents. This paper presents RASAHIA's functional structure and the architectures of its reflective agents, developed in accordance with the methodology [12] and the system's model [2].

II. Typical functional structure of reflective-active systems of artificial heterogeneous intelligent agents

The typical functional structure of RASAHIA (Fig. 1) describes the subsystems of agents and their functionality, information flows and relationships between them, as well as their interaction with the environment, which is considered as a multi-agent system of a higher level. At the same time, each of the presented agents can be a lower-level RASAHIA. The proposed structure acts as a basis for developing a system for a specific problem. During this process the number and composition of agents have to be specified, which are not predetermined in the typical structure.

The interface subsystem of RASAHIA is an intermediary between the system agents and its users, as well as the control object. The interface agent requests input data from the user and presents the result of the problem solving, as well as provide the user with the system configuration and visualize the processes in RASAHIA [13]. The interface agents receive information about control object's state and issue control actions through software interfaces.

The technological subsystem provide service functions to other agents of the system. Translator agent is included in the process of transmitting messages between a pair of agents if they do not support a single language and cannot communicate with each other directly. The intermediary agent ensures the "yellow pages" service, i.e. provides names of agents with

specific abilities. The basic ontology is a technological element of RASAHIA, ensuring that agents understand the semantics of each other's messages within basic communication to coordinate their own ontologies, built on its basis, goals and problem solving protocols.

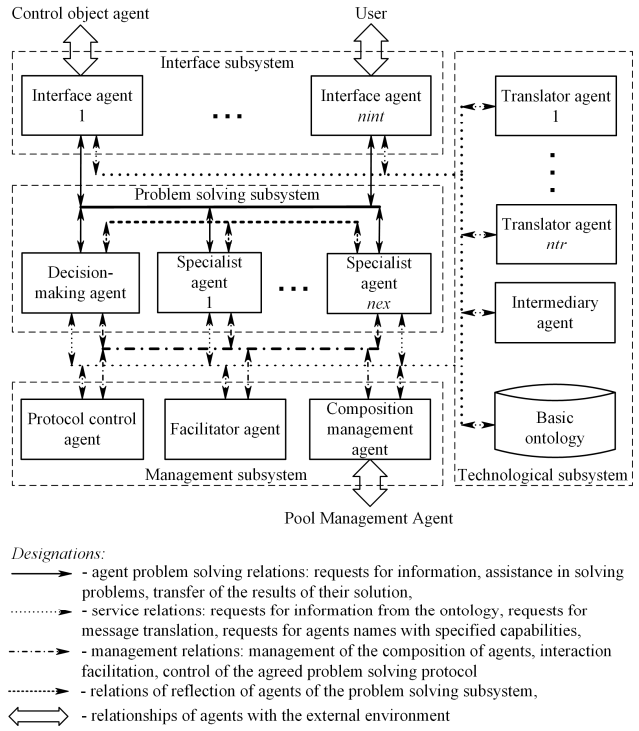

Fig. 1. Typical functional structure of reflective-active system of artificial heterogeneous intelligent agents

The management subsystem ensures the effective interaction of other agents and directs RASAHIA's self-organization. The protocol control agent monitors that the agents' actions comply with the problem solving protocol they have agreed upon. The facilitator agent ensures the effective joint work of the agents of the problem solving subsystem, in particular, assesses the current situation in it, initiates the actions of agents that stimulate or suppress conflicts arising between them [14]. The composition management agent attracts agents from the pool in the environment to the problem solving subsystem, simulating the methods of selecting specialists for real teams [15], and also excludes agents from RASAHIA, placing them in the pool.

The problem solving subsystem is designed for computer modeling of group work of specialists on solving a problem. This subsystem ensures the implementation of the principle of necessary diversity by modeling the reasoning of specialists in different fields and using different problem solving methods. The decision-making agent, having received the information necessary for solving the problem from the interface agent, decomposes the problem into sub-problems, distributes them between specialist agents, collects and evaluates their solutions, forms the result of the system's work. The specialist agent, modeling the reasoning of a real specialist, solves a sub-problem or the problem as a whole depending on the instructions of the decision-making agent. The agents of this subsystem can be created by different developers, which potentially leads to differences and contradictions in their ontologies and goals. However, due to the reflective modeling of each other's reasoning, the intensity of conflicts and the duration of negotiations are reduced.

III. ARCHITECTURES OF INTELLIGENT REFLECTIVE SPECIALIST AGENTS

The architectures of specialist agents of different ranks of reflection are developed based on their micro-level model [2]. The required set of actions of a specialist agent in accordance with the formal model [2] and the typical functional structure of RASAHIA (Fig. 1) can be described as follows:

$$ACT^{ag\ sp} = ACT^{ag}_{msg} \cup ACT^{ag}_{fnc} \cup ACT^{ag}_{mig} \cup ACT^{ag}_{ref} \cup \cup ACT^{ag}_{rec} \cup ACT^{ag}_{neg} \cup ACT^{ag}_{com}, \quad (1)$$

where ACT^{ag}_{msg} is a set of service actions on receiving and transmitting messages, interpreting and composing them, which are standard for all agents; ACT^{ag}_{fnc} is a set of actions on modeling the reasoning of specialists when solving a problem or its parts; ACT^{ag}_{mig} is a set of actions of an agent on moving to the pool in the environment and back; ACT^{ag}_{ref} is a set of actions in accordance with the reflective control model *refc* (7) from [2]; ACT^{ag}_{rec} is a set of actions in accordance with the method *agrec* for making recommendations of specialist agents from agent pool based on the experience of previous joint work; ACT^{ag}_{neg} is a set of actions (method) on coordinating among agents their domain models, goals and protocols [2].

The set ACT^{ag}_{ref} of actions in accordance with the model of reflective control (1) depends on the rank of reflection of the specialist agent. The set of actions of an agent with a zero rank of reflection, i.e. not possessing reflection, is empty

$$ACT^{ag}_{ref\,0} = \varnothing. \quad (2)$$

The set $ACT^{ag}_{ref\,1}$ of actions of agent with the first rank of reflection is described by the expression

$$ACT^{ag}_{ref\,1} = \{ACT^{ag}_{rmdl},\ ACT^{ag}_{rctr}\}, \quad (3)$$

where ACT^{ag}_{rmdl} are the actions of modeling other agents by reflective agents based on the options they propose for solving the problem; ACT^{ag}_{rctr} are the actions of developing negotiation tactics and strategies by reflective agents in accordance with their models of other agents.

The set $ACT^{ag}_{ref\,2}$ of actions of first-rank reflection agent is supplemented by actions ACT^{ag}_{rctrm} of reflective control of agents based on their models, as well as actions ACT^{ag}_{rrid} of identifying other agents' reflection rank

$$ACT^{ag}_{ref\,2} = \{ACT^{ag}_{rmdl},\ ACT^{ag}_{rctr},\ ACT^{ag}_{rctrm},\ ACT^{ag}_{rrid}\}. \quad (4)$$

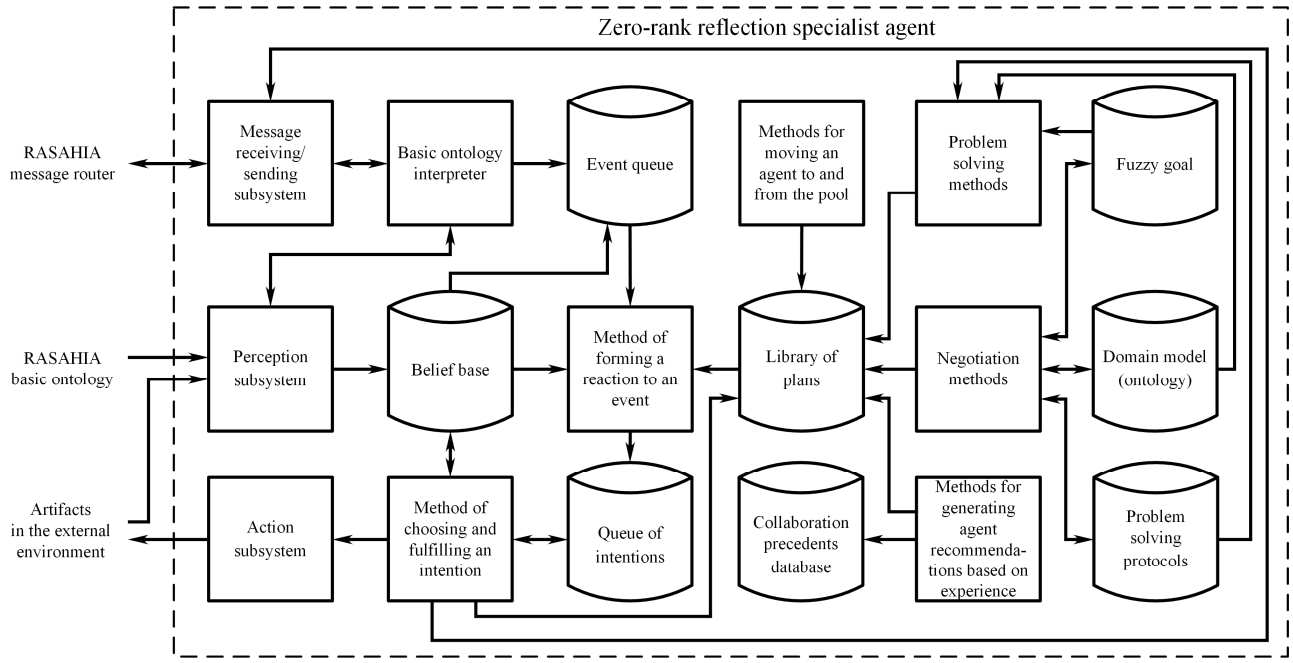

Fig. 2. Architecture of a zero-rank reflection specialist agent

Thus, based on (1) and (2), the architecture of a zero rank reflection specialist agent can be represented by Fig. 2.

The perception subsystem monitors the state of artifacts in the agent's external environment, which are divided into RASAHIA artifacts (the basic ontology) and artifacts of its environment. In terms of the JaCaMo platform [16], on which RASAHIA is implemented, and its component, the CArtAgO subsystem, an artifact is a functionally oriented computational abstraction providing services to agents [17] through a set of publicly available functions and observable properties [18]. If artefact's property, to which the agent is subscribed, changes, a corresponding notification is sent to it. Notifications are processed by the agent's perception subsystem, which, upon receiving one, forms a list of percepts and modifies the agent's belief base. If an agent needs to interpret a concept from other agents' messages, it can do so using the public functions of the artifact implementing the RASAHIA basic ontology.

The agent's belief base is a repository of the agent's ideas about its environment, i.e. about RASAHIA and its environment. The belief base is modified either as a result of observing changes in the environment using the perception subsystem, or as a result of the agent's reasoning and the execution of intentions by the appropriate method. Each time the belief base is adjusted, an event is generated, which can be external, caused by a change in the belief base by the perception subsystem, or internal, generated by the method of selecting and executing an intention (in this case, the intention that caused the event is also recorded). The belief base is implemented using standard tools of the JaCaMo platform and its component, the Jason subsystem [19].

The message receiving/sending subsystem ensures communication with other agents via the RASAHIA message router. The latter is a subsystem of the JaCaMo software platform, ensuring correct delivery of messages to their addressees. Upon receiving a message from another agent, the message receiving/sending subsystem places it in a queue. In each reasoning cycle, the agent selects the first message from the queue and processes it. Sending messages to other agents via the RASAHIA message router is performed upon request of the method of choosing and fulfilling an intention, while the basic ontology interpreter is used to form a semantically correct message. The message receiving/sending subsystem is implemented with standard tools of the Jason subsystem of the JaCaMo platform, which supports prioritization and filtering.

The basic ontology interpreter, receiving the message body from the message receiving/sending subsystem, performs its semantic analysis using the basic ontology, forms events containing the program objects generated as a result of the analysis, and places them in the queue. Upon receipt of a request from the message receiving/sending subsystem, the interpreter of the basic ontology generates a semantically correct message body in accordance with the intention that initiated the request.

The event queue is a buffer containing an ordered set of change-intention pairs. A change can be a change in beliefs as a result of the perception of the environment (in this case, the second part of the pair, the intention, remains empty) by the agent itself or through other agents, or as a result of the execution of the corresponding intentions. The order of events can be configured by the agent developer.

The method of forming reaction to an event selects the first pending event from the queue, generates an intention relevant to it using the plan library, and places the latter in the queue of intentions. According to this method, all plans that have a triggering event that can be merged with the selected event in accordance with the merging mechanism adopted by the Jason subsystem of the JaCaMo platform are selected from the plan library. Of the found plans, those are selected whose contextual

part corresponds to the current beliefs of the agent [19]. If such plans are not found, the selected event is moved to the end of the queue for subsequent reprocessing. If more than one plan remains, the first one is selected for further processing in accordance with the order in which the plans are written in the source code of the agent, otherwise the only plan is selected. The selected plan becomes an intention and is placed in the queue of intentions.

The library of plans contains the agent's action algorithms for reacting to events occurring in a certain situation. The plan consists of a body and a header, which in turn contains the initiating event and context that determine the conditions for executing the plan. The initiating event of the plan describes a set of real events for which the plan should be used. If the initiating event of the plan corresponds to the real one, the plan is considered as a relevant one and, if the context is true, according to the agent's current beliefs, it becomes a candidate for execution. The plan body is a sequence of instructions (formulas) for processing of the event. The instructions in the plan body can be a call to Java functions, direct actions for changing objects of the environment or sending messages, as well as actions for generating beliefs or plans.

The block "Methods for moving an agent to and from the pool" contains functions in the Java language that can be called from the plan body when initializing an agent during its inclusion in the problem solving subsystem from the pool in the RASAHIA's environment or when excluding it from the system and moving it to the pool. When an agent is included in the problem solving subsystem, an empty agent is created that executes the agent initialization plan calling the corresponding method of the current block and passes the identifier of the agent being attracted to it. The initialization method requests the agent configuration from the agent pool in accordance with this identifier via the perception subsystem, after which it modifies all other agent blocks using it (the arrows displaying these interactions are omitted in Fig. 2 for clarity). Upon completion of initialization, the agent sends a message to the intermediary agent, registering its name, address, and capabilities. The agent is excluded from the problem-solving subsystem and moved to the pool by command from the composition management agent. Having received a message from it, the agent processes it in priority order, executing the plan for saving the configuration and shutting down the work, which calls the method for moving the agent to the pool. This method saves the state of all agent blocks at the current moment and, using the action subsystem, sends this information to the pool for saving the configuration and possible subsequent use. After that, actions are performed to shut down the agent: sending a message to the intermediary agent about the agent's termination, releasing possible connections and occupied resources, and destroying the agent.

The "Problem solving methods" block is a set of functions that provide modeling of the specialist solving the problem. These functions can use various methods of formal representation of systems [20], for example, analytical, stochastic, fuzzy, and are intended to solve a specific problem or part of it, so they are not considered in detail here.

The fuzzy goal, i.e. a fuzzy set specified on the set of states of the control object, is used as an optimality criterion when implementing problem solving methods [20].

The domain model (ontology) is a list of concepts, properties, and characteristics for describing this domain, as well as the laws of the processes occurring in it, implemented using the OWL ontology [21].

Problem solving protocols define schemes (distributed algorithms) for exchanging information and knowledge, as well as coordinating agents, the formal model of which is presented in [22].

The "Negotiation methods" block represents functions that implement models of coordination between agents of their domain models, goals and protocols to achieve their consistency when new agents from the pool are included in the system, since in the general case the domain models, goals and problem solving protocols of specialist agents created by different teams of developers do not coincide.

The block "Methods for generating agent recommendations based on experience" is a set of functions that provide modeling of referral recruitment in RASAHIA [23] case-based reasoning. The function of generating experience of joint work with agents is performed throughout the agent's activity when it requests help in solving parts of a problem from other agents. This function records as a precedent the concept of the problem for which the agent requests help, its characteristics, the identifier of the agent from whom this help is requested, the result obtained, the duration of obtaining the result, and other parameters defined by the agent developers. The recorded precedents are placed in the "Collaboration precedents database". Provision of information from this base is performed by a separate function at the request of the composition management agent.

The queue of intentions is an ordered list of plans accepted for execution and which have become intentions, or their parts. In the general case, the agent's queue of intentions contains more than one intention, each of which competes for the agent's "attention". In each reasoning cycle, the agent executes one instruction (formula) of the intention, after which it is moved to the end of the queue. As a result, pseudo-parallel execution of all plans accepted for execution is organized. The queue of intentions is implemented by standard means of the Jason subsystem of the JaCaMo platform.

The method of choosing and fulfilling an intention uses the standard function of selecting an intention of the Jason subsystem, which provides prioritization. The method selects the first intention in the queue, removes it from the queue, and executes one of its instructions. The executed instruction is removed from the intention. If there are no instructions left, the intention is considered executed and removed from the queue, otherwise the adjusted intention is inserted at its end.

The action subsystem is designed to change the state of artifacts in the agent's environment using their publicly available functions.

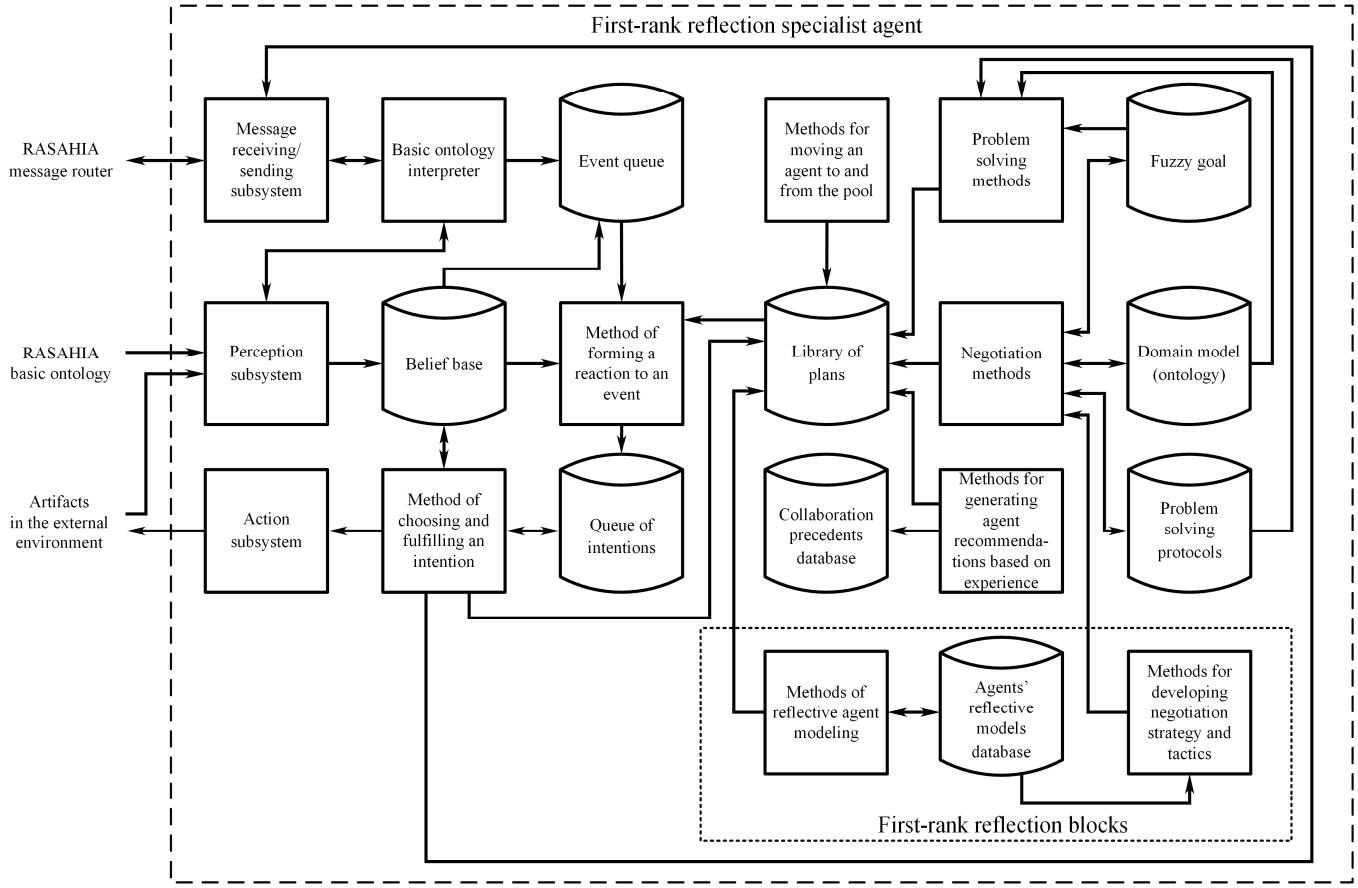

Fig. 3. Architecture of a first-rank reflection specialist agent

In accordance with (1) and (3), the architecture of the first-rank reflection specialist agent can be represented by the Fig. 3. The main difference from the Fig. 2 is the presence of the blocks “Methods of reflective agent modeling”, “Agents’ reflective models database” and “Methods for developing negotiation strategy and tactics”.

Methods of reflective agent modeling are intended to form relevant models of other agents and save them in the “Agents’ reflective models database”. These methods are intended to solve the identification problem, i.e. finding the optimal model of the agent under study as a result of observing its behavior and response to external disturbances, for example, incoming messages, changes in the environment or in RASAHIA itself [24]. The implementation of methods of reflective agent modeling varies depending on the artificial intelligence technology used and is determined by the developers of a specific agent. In the most general form, the methods can be described as follows: for each simulated agent, a blank model is formed using its own model as a basis; upon receiving information about the simulated agent during negotiations, adjust it; if as a result of the adjustment the model has become irrelevant to the simulated agent, adjust other parts of the model to compensate for the decrease in the quality of the model; upon receiving the next message, it is necessary to re-evaluate the relevance of the model to the agent, and, if necessary, adjust the model; if a model is assessed as irrelevant to the agent, and the modeling agent does not have enough information to build a relevant model, it is necessary to exclude such a model from further considerations during negotiations, but continue to adjust it as new information about the agent arrives.

The block “Methods for developing negotiation strategy and tactics” is designed to plan behavior when negotiating with other agents. Negotiation strategy is a predetermined approach or general agent’s plan of action to achieve own goal as a result of negotiations [25]. Negotiation tactics are a detailed method used by an agent to gain an advantage, including the use of manipulative techniques and reflective control. To form a negotiation strategy for a reflective agent, a method based on Mamdani's fuzzy inference is proposed, implementing D. Pruitt’s “double concern” model [26]. As part of the implementation of the negotiation strategy, the agent can use one or more tactics implemented by it in accordance with the chosen strategy, the current situation and the model of the agent-opponent. To select negotiation tactics, a case-based reasoning [27] approach can be used. The method of developing behavioral tactics by reflective agents includes two functions (actions): choosing tactics and their settings, and saving the results of applying the tactics.

In accordance with expressions (1) and (4), the architecture of the second-rank reflection specialist agent can be represented by the Fig. 4. The main difference from the Fig. 3 is the presence of the blocks “Reflective control methods” and “Method for identifying the rank of agent reflection”. In addition, the agent’s reflective models database contains two

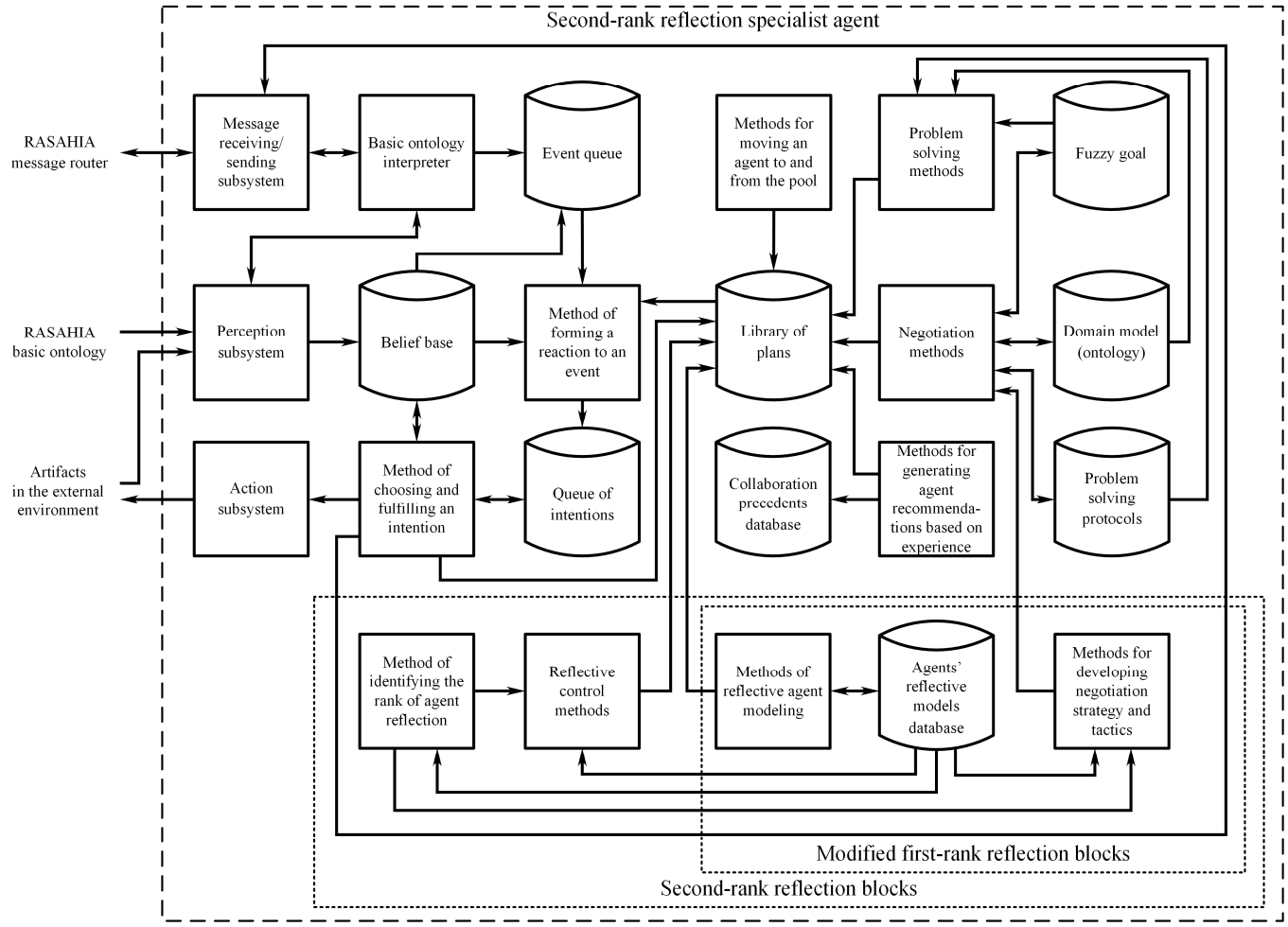

Fig. 4. Architecture of a second-rank reflection specialist agent

models of zero and first rank for each agent. The method for identifying the rank of agent reflection is designed to select a relevant model of an agent of the corresponding rank for subsequent reasoning based on its behavior during negotiations. The block "Reflective control methods" is designed to select negotiation tactics that mislead the opposing agent, which is beneficial to the second-rank reflection specialist agent for achieving its goals. In this case, the second-rank specialist agent of reflection assumes that the opposing agent models its behavior and selects its negotiation strategies and tactics in accordance with this model, i.e. is an agent of the first rank of reflection. In general, reflective control methods are considered in [28].

IV. Conclusion

The paper presents the functional structure of the reflective-active system of artificial heterogeneous intelligent agents and the architectures of its agents of the zero, first and second ranks of reflection, implementing the basic principles of constructing such systems in accordance with the methodology [12]. In particular, due to the dynamic composition and diversity of the agents of the problem solving subsystem, the heterogeneity and variability of the problem is taken into account. The proposed architectures of the agents implements such properties as autonomy, activity, reactivity, communicativeness, reflection, the ability to model the domain and goal setting. The reflective control mechanisms used by the agents ensure homeostasis of the system due to the coordination of their own goals, ontologies and protocols. Due to the open nature of the system and the reflective control mechanisms, self-organization of agents in the strong sense arises in the system [11], without centralized control of this process by one of them.

Acknowledgment

The study was supported by the Russian Science Foundation grant No. 23-21-00218, https://rscf.ru/project/23-21-00218/

References

[1] D. Dubrovsky, V. Lepskiy, and A. Raikov, "General Artificial Intelligence in Self-developing Reflective-Active Environments," in *World Organization of Systems and Cybernetics 18. Congress-WOSC2021. WOSC 2021. Lecture Notes in Networks and Systems,* vol 495, 2022, pp. 3–13.

[2] S.V. Listopad, "Modelirovanie refleksivnyh processov v kollektivah specialistov, reshajushhih problemy za kruglym stolom" ["Modeling reflective processes in teams of specialists solving problems at a round table"], *in Modelirovanie neravnovesnyh, adaptivnyh i upravljaemyh sistem: Materialy XXVI Vserossijskogo seminara [Modeling nonequilibrium, adaptive and controlled systems: Proceedings of the XXVI All-Russian seminar]*, 2023, pp. 57–66. (in Russian)

[3] V.B. Tarasov, *Ot mnogoagentnykh sistem k intellektual'nym organizatsiyam: filosofiya, psikhologiya, informatika [From multi-agent systems to intelligent organizations: philosophy, psychology, computer science]*. Moscow: Editorial URSS, 2002. (in Russian)

[4] V.I. Gorodetskiy, O.V. Karsaev, V.V. Samoylov, and S.V. Serebryakov, "Instrumental'nye sredstva dlya otkrytykh setey agentov" ["Tools for open agent networks"], *Izvestiya RAN. Teoriya i sistemy upravleniya [News of the Russian Academy of Sciences. Theory and control systems]*, vol. 3, 2008, pp. 106–124. (in Russian)

[5] M. Wooldridge, *An Introduction to Multiagent Systems*. New York: Wiley, 2009.

[6] V.A. Lefebvre, *Conflicting Structures.* New York: Leaf & Oaks Publishers, 2015.

[7] D.A. Novikov, and A.G. Chkhartishvili, *Reflexion and Control: Mathematical Models*. London: CRC Press, 2014.

[8] B. Kobrinskii, "Expert reflection in the process of diagnosis of diseases at the extraction of knowledge", *in IV International Research Conference "Information Technologies in Science, Management, Social Sphere and Medicine" (ITSMSSM 2017), December 5-8, 2017, Proceedings. Advances in Computer Science Research*, vol.72, 2017, pp. 321–323.

[9] I.V. Smirnov, A.I. Panov, A.A. Skrynnik, and E.V. Chistova, "Personal'nyj kognitivnyj assistent: koncepcija i principy raboty" ["Personal cognitive assistant: concept and operating principles"], *Informatika i ee primenenija [Informatics and Applications]*, vol. 13(3), 2019, pp. 105–113. (in Russian)

[10] V.B.Melehin, V.M. Hachumov, and M.V. Hachumov, "Samoobuchenie avtonomnyh intellektual'nyh robotov v processe poiskovo-issledovatel'skoj dejatel'nosti" ["Self-learning of autonomous intelligent robots in the process of search and research activities"], *Informatika i ee primenenija [Informatics and Applications]*, vol. 17(2), 2023, pp. 78–83. (in Russian)

[11] G.D.M. Serugendo, M.-P. Gleizes, and A. Karageorgos "Self-organization in multiagent systems", *The Knowledge engineering review*, vol. 20(2), 2005, pp. 165–189.

[12] S.V. Listopad, "Zhiznennyy tsikl metodologii postroeniya refleksivno-aktivnykh sistem iskusstvennykh geterogennykh intellektual'nykh agentov" ["Life cycle of the methodology for constructing reflective-active systems of artificial heterogeneous intelligent agents"], *Informatika i ee primeneniya [Informatics and Applications]*, vol. 18(1), 2024, pp. 84–91. (in Russian)

[13] S.B. Rumovskaya, and I.A. Kirikov, "Metod vizual'nogo predstavleniya konfliktov v gibridnykh intellektual'nykh mnogoagentnykh sistemakh" ["Method of visual representation of conflicts in hybrid intelligent multi-agent systems"], *Informatika i ee primeneniya [Informatics and Applications]*, vol. 14(4), 2020, pp. 77–82. (in Russian)

[14] S.V. Listopad, and I.A. Kirikov, "Metod na osnove nechetkikh pravil dlya upravleniya konfliktami agentov v gibridnykh intellektual'nykh mnogoagentnykh sistemakh" ["A method based on fuzzy rules for managing agent conflicts in hybrid intelligent multi-agent systems"], *Informatika i ee primeneniya [Informatics and Applications]*, vol. 17(1), 2023, pp. 66–72. (in Russian)

[15] S.B. Rumovskaya, "Podkhody k podboru spetsialistov pri organizatsii kollektivnogo resheniya problem" ["Approaches to the selection of specialists in organizing collective problem solving"], *Informatika i ee primeneniya [Informatics and Applications]*, vol. 17(2), 2023, pp. 96–103. (in Russian)

[16] O. Boissier, R.H. Bordini, J. Hubnerand, and A. Ricci, *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*. Cambridge: The MIT Press, 2020.

[17] A. Freitas, A.R. Panisson, L.W. Hilgert, F. Meneguzzi, R. Vieira, and R.H. Bordini, "Integrating Ontologies with Multi-Agent Systems through CArtAgO Artifacts," *in 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2015, pp. 143–150.

[18] A. Ricci, M. Piunti, and M. Viroli, "Environment programming in multi-agent systems: An artifact-based perspective," *Autonomous Agents and Multi-Agent Systems*, vol. 23(2), 2011, pp. 158–192.

[19] R.H. Bordini, J.F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*. Chichester: Wiley-Interscience, 2007.

[20] A.V. Kolesnikov, I.A. Kirikov, and S.V. Listopad. *Gibridnye intellektual'nye sistemy s samoorganizatsiey: koordinatsiya, soglasovannost', spor [Hybrid intelligent systems with self-organization: coordination, consistency, dispute]*. Moscow: IPI RAN, 2014. (in Russian)

[21] Web Ontology Language (OWL), Web: https://www.w3.org/OWL/

[22] I.A. Kirikov, and S.V. Listopad, "Cohesive Interaction Protocol Development in Hybrid Intelligent Multi-agent Systems," *in Proceedings - 2021 3rd International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA-2021)*, 2021, pp. 553-558.

[23] A. Mishra, Referral recruitment – the most effective way of recruitment and how you can improve it, Web: https://www.hackerearth.com/blog/talent-assessment/referral-recruitment-effective-way-recruitment-can-improve/

[24] V.E. Pjateckij, V.S. Litvjak, and I.Z. Litvin, *Metody prinjatija optimal'nyh upravlencheskih reshenij: modelirovanie prinjatija reshenij [Methods of making optimal management decisions: decision-making modeling]*. Moscow: MISiS, 2014. (in Russian)

[25] Negotiation Strategy, Web: https://www.negotiations.com/definition/negotiation-strategy/

[26] P. Carnevale, and D. Pruitt, "Negotiation and Mediation", *Annual Review of Psychology*, vol. 43, 2003, pp. 531–582.

[27] R.L. de Mantaras, "Case-Based Reasoning", *in: Machine Learning and Its Applications. ACAI 1999. Lecture Notes in Computer Science*, vol 2049, 2001, pp. 127–145.

[28] V. Lefebvre, *Lectures on the Reflexive Games Theory*. New York: Leaf & Oaks Publishers, 2010.