UDC 004.94 + 004.413.4 + 004.414.23

O.M. PAKHNINA

National Aerospace University named after M.Ye. Zhukovsky «Kharkiv Aviation Institute», Ukraine

THE INTERACTION OF INTELLECTUAL AGENTS IN THE PRODUCTION PROCESSES SIMULATION MODELING SYSTEM

The paper has considered the intellectual agents' interaction mechanisms in thread processes modeling. The major advantages of the approach have been demonstrated, related to the possibility to create elements of individual behavior, dynamically changing in the process of simulation, using adaptation and self-organization mechanisms. This paper deals with issues related to executors information interaction in the organizational management structure with allowance for auction mechanisms of works distribution and dynamic changes emerging during the simulation due to risk factors manifestations.

Key words: *agents, simulation modeling, agents' interaction, protocol, ontology, organizational structure, executors, risks.*

Introduction

The most effective analysis and prediction of dynamical characteristics of production systems are provided by the facilities based on simulation modeling methods. The use of intellectual information technologies implemented in the form of agent-based systems is of current interest for provision of flexible dynamic behavior mechanisms, autonomy and adaptation of individual components of a simulation model. The agent-based modeling [1] assumes that a model includes a number of agents – information (program) elements – interacting among themselves and with external environment, having their goals and tasks, internal state and behavioral rules. Therefore, since each individual agent executes its own task, and the common task is solved by a number of agents, there is a need to have methods of interaction and synchronization of different agents' actions. The agents exchange information with one another by means of negotiation. The agent interaction protocol determines a pattern (distributed algorithm), according to which negotiations are conducted. Considering the diversity of the agents' interaction mechanisms (cooperation, competition, compromise etc.), the agents must be provided with developed communication protocols.

The Analysis of Latest Researches And Publications

All researches in the field of multiagent systems point out such communication component of the intellectual agent interaction as message exchange. The principles of speech act theories form the basis of communications in the multiagent systems. The agent communication languages (ACLs) – standard message exchange protocols - are used for the agents' interaction.

At present, there exist two most widely used languages – FIPA ACL and KQML.

KQML was developed on the initiative of the *Advanced Research Projects Agency* (ARPA) and consists of two parts: *Knowledge Query and Manipulation Language* (KQML); *Knowledge Interchange Format* (KIF). The KQML language [2] consists of three levels: content, messages and communications. Knowledge representation in some language is on the content level. The message level adds extra attributes such as description of a language, in which the content is expressed, its ontology and the type of communication method used. The communication level adds information about the message sender and receiver, and indicates whether a message is synchronous or asynchronous. KQML determines the number of valid communication performatives, for example: ask-if, perform, tell, reply.

On the language level, KIF [3] is used, being a message content transfer format. Usually, KIF is not used as a language for internal knowledge representation but only as a language of knowledge/data exchange with other agents. The agent receives external knowledge expressed in the KIF language, translates it into an internal representation (lists, frames etc.), and performs all computations using these internal representations.

However, the agreement on the knowledge transfer language does not guarantee absolute knowledge exchange; therefore, this knowledge can be structured differently, defined by different terms, unknown to other agents. To preserve the declarative content of this knowledge, and thus to be able to form queries to other intellectual systems, exchange knowledge with them, the knowledge shall be represented in a formal form.

Therefore, a formal and declarative representation shall be formed, including a list (vocabulary) of references to terms in a desired subject area and logical expressions (rules), describing definitions of the said terms and their interaction. Such representation is called the subject area ontology, serves for simplification of programming the agents' behavior, and is used by them when interacting.

FIPA ACL [4] is developed by the FIPA Committee (*The Foundation for Intelligent, Physical Agents*) in charge of the issues of agents and agent platforms creation and interaction standards development. The general structure of this protocol is very similar to KQML: *performative, housekeeping, content-* an informative portion of a message.

The Research Objective

The article suggests an approach to the simulation modeling of production systems as processes of information interchange among different active functional components of a simulation model represented as agents. Employing this approach enables simulation of dynamic business-processes, whose scenarios are permanently varying with time under the influence of internal and external factors. In the suggested simulation modeling system, the agents are oriented at performing workflows comprising a business-process. Accordingly, the article deals with the features related to organization of effective interaction of intellectual agents, which can be associated with the executors cooperation during the common tasks execution, with the search for an optimum executor capable of performing the work on the most advantageous terms, for coordinations emerging when risk situations occur etc.

The Agent-Based Simulation Model

The architecture of the suggested simulation system is based on knowledge (has a knowledge base, and the decisions on the agents' actions are taken based on the inference engines) and is hierarchical, since there are the «meta-level» agents, coordinating distributed task execution by other agents.

Fig. 1 shows the system agents, considering their roles in interaction in the simulation process.

The discrete-event-driven simulation modeling system [5] forms a conceptual basis for creating the agent simulation system. Its major model element is a block of tasks or works, with an individual *TaskAgent* being assigned to each of them, the main task of the *TaskAgent* will be a successful task completion (in time,

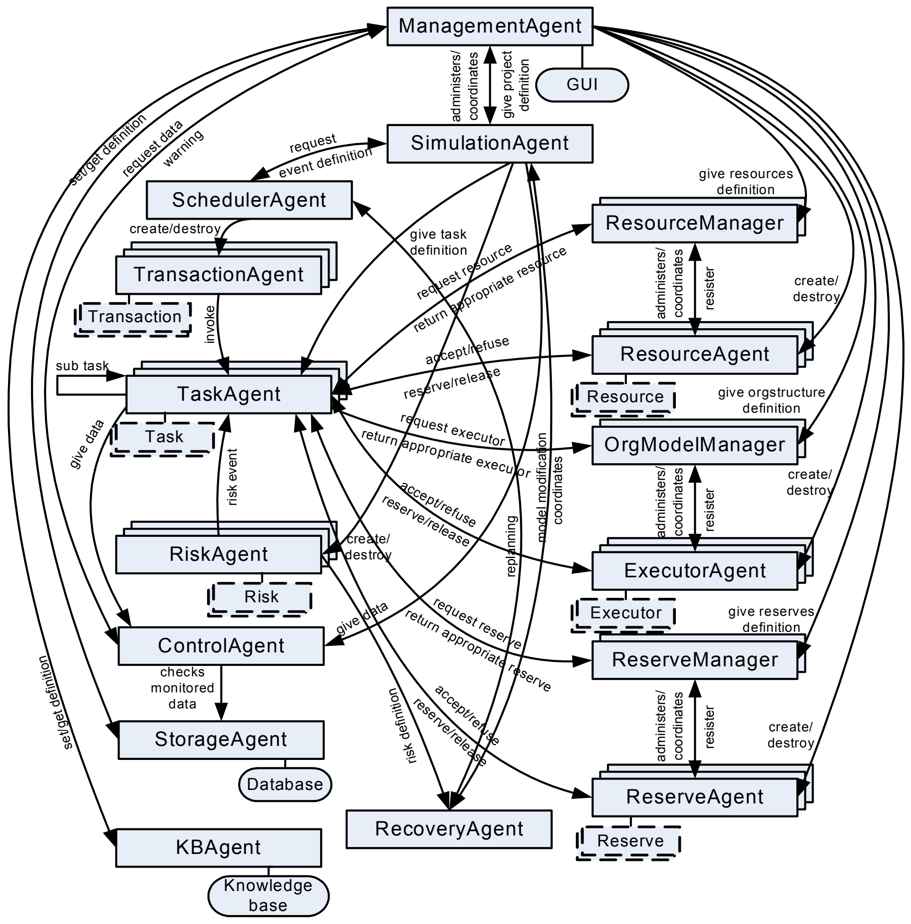

Fig. 1. The system agents

at the expense of the allocated funds, in full scope and at a required quality).

The *TaskAgent* class in the ontological knowledge base has the following attributes: the task type (fixed duration, fixed volume, extendible, check events); execution duration; work volume per task; labor consumption; direct costs per task; task execution period limitations, nesting tags, priorities, synchronization conditions, resources, schedule dates and terms of execution, risks, interruptability etc.

Dynamic processes in the discrete - event-driven simulation model exist in the form of interaction of a number of components [5]. When switching from the discrete - event-driven simulation model to the agent representation of the simulation model, one should proceed from distinguishing individually behaving elements [6].

With this approach used, all the simulated dynamic «entities» (products, parts, orders, queries, requests, documents etc.) are the *TransactionAgents*, and the next transaction generation in the system will correspond to a new agent creation.

In the discrete-event-driven simulation modeling system, the devices service transactions and are designed to limit the works performed based on the volume and configuration of the available process facilities and various resources.

Resources may be understood as people, equipment and materials required for the task execution [7]. The resources are individually behaving elements; therefore, they are represented by two types of agents: the *ExecutorAgents* (units of organizational management structure) and the *ResourceAgents* (all other resource types).

Separation of an individual *ExecutorAgent* class results from solving the task of modeling the information interaction among the organizational structure units in this system (individual employees, departments and other sections of the administrative machinery). Features of the *ExecutorAgents* interaction are primarily associated with the fact that the relations between them are supported due to the links existing in the organizational structure, conventionally subdivided into horizontal (matching links) and vertical (subordination links). Except for the subordination hierarchy, it is also necessary to take into account the structure of data link channels between the organizational structure units; the nature and dynamics of information exchange among them in the course of management at all levels. Moreover, behavior scenarios and protocols of an organizational structure unit interaction with other agents will depend on its role towards the task (initiator, decision-maker, coordinator, manager, direct executor, intermediary).

To coordinate an access to a resource, a mechanism is needed that is implemented either centrally (dispatcher), or decentrally, through the agents interaction. The *ResourceManager* selects recourses for the task execution, monitors the resources' parameters, calculates storage costs, while observing a varying demand for the resources, it can procure (make additional orders for) materials and component parts in demand, or take decisions to reduce or enhance a warehouse space, thereby providing continuous adaptation and evolution. Each resource agent in the system is registered and assigned to one of the *ResourceManagers*, whose total number may be defined, for example, by the resource grouping according to certain features.

For the *ExecutorAgents, the OrgModelManager* plays the same part.

The *ReserveAgent* is created to provide risk-tolerant projects, when it is necessary to create an own additional stock of backup resources or reserves destined to counteract a possible risk complex. The *ReserveManager* coordinates the access to reserves.

The *RiskAgent* is destined for simulation of risk situations occurrence. At the stage of risk identification (probability of the risk of the classification list is indicated for the task), each task is assigned the risk probability, the risk degree, i.e. how significant the consequences of the adverse event occurrence will be, the risk criticality as a product of the risk probability and degree. Besides, the following parameters can be assigned to each risk: the task execution performance phase, at which the risk emerges, the type and amount of loss (for example, in the form of required extra costs or resources), insurance feature for this risk (the source is indicated (an own reserve fund or a third party), amount, manner of payment and rate of interest (for external insurance)). Creating a new risk agent in the system will correspond to occurrence of a risk event.

The *SimulationAgent* exercises the general supervision and management of the work package simulation, provides for support for cooperative solution of emerging conflicts, if this is within its competence, assists the agents in coordination of their actions, distributes privileges, suggests various common solutions, is capable of switching the agents negotiations (for example, from avoiding a penalty by meeting the project's scheduled dates to reserving the time for the equipment repair by their own means).

The *SchedulerAgent* is in charge of scheduling the system events related to the project tasks execution, synchronization, provides for transactions generation.

The *ControlAgent* collects all the data and sends them to the *StorageAgent*, identifies unfavorable situations defined for different aspects of the project model: work package (tasks with minor free reserves, long-term tasks with large resources, externally dependent tasks), resources (resources with a large volume of work, executors with unique skills, materials delivered by a single supplier), finance, reserves (levels of reserves, consumption rate) etc. Thus, identified are events, whose occurrence may prevent timely completing the project, or entail lack of resources or funds at a certain instance of its implementation. It informs the *ManagementAgent* about all the irregularities identified.

The *ManagementAgent* provides for functional capabilities to manage the entire project tasks flow, creation and deletion of new tasks during simulation (rescheduling), creation of new *ResourceAgents*, defining roles, initiates a user dialogue if the user's decisions are required to continue simulation. If the way to reduce the degree of risk due to an unfavorable situation occurrence becomes evident on receipt of a message about it from the *ControlAgent*, relevant changes are immediately introduced in the model, or the user dialogue is initiated.

The *RecoveryAgent* is charged with duties associated with coordination of actions performed by other agents (specifically, by the *SimulationAgent, the SchedulerAgent*) when solving risk situations emerging in the system (works rescheduling, resource recovery/repair, use of reserves etc.). Moreover, it can roll back to the previous simulation steps on the user's demand by referring to a historic database where all the data on the simulation progress are stored.

The *StorageAgent* manages storage of all the statistical data on simulation in the database.

The *KBAgent* (Knowledge Base Agent) manages interaction with the system knowledge base containing the ontology shared by all the system agents, and the rule bases for decision making in the course of simulation and at the results interpretation stage.

The General Provisions of the Agents Interaction

All the listed agents take an active part in negotiations (message exchange) on the project work package simulation, its "on-the-fly" rescheduling in case of the situation changes related to risk factors manifestations (a new resource emerging or any old resource failure, delay in performance of a certain type of work, resource unpreparedness etc.).

The protocol describes formal rules to be followed by the negotiation process participants (employing different strategies) so that their messages are understood by each party.

The basic Interaction Protocols (IP) within the FIPA standard framework are *Request IP* (a request for an action execution), *Request When IP* (a request for an action execution provided that a certain condition is fulfilled), *Query IP* (a query to perform an informaction associated with a response to a certain proposal or transfer of an object description), *Contract Net Protocol* (search for an agent to perform an action) [8].

In the most general case, the basic agent interactions can be represented as follows:

1. Upon its creation, the *TransactionAgent* calls the next *TaskAgent* in accordance with the project work package description stored in the ontological knowledge base.

2. The *TaskAgent* checks for its readiness and availability of all the necessary input conditions for the task execution (the task execution can be blocked according to the system calendar or for some other reasons).

3. If executors and resources have been assigned to this task, the *TaskAgent* communicates with the *ResourceAgents* and *ExecutorAgents* by sending a service request message, having an identifier of the copy of this task's ontological class for matching.

4. The *ResourceAgent* or *ExecutorAgent* responds with consent if it is free (is assigned immediately) or rejects the request in case it is busy executing another task, or due to insufficient quantity of the requested resource.

5. In case of rejection, the *TaskAgent* makes a request for an attempt to parallelize the resources simultaneous participation in several tasks.

6. If it goes wrong, the conflict is escalated and the *ResourceManager* enters into negotiations. It attempts to select a similar free resource or to swap the resources already engaged in the task (for other resources with similar properties).

7. If this attempt is not a success, the *ResourceManager* enters into negotiations with the *ReserveAgent*.

8. If there no required resources in the reserve fund, the *Resource Manager* tries to hire/buy new resources of the same type, which will demand for extra time and financial expenses.

9. In this case, the *ResourceManager* (its purpose is to satisfy demand in the resource and to minimize the task delays) enters into negotiations with the *SimulationAgent* to assess the conflict resolution options: waiting until the resource is free, give an extra order for the resource, use the priorities and delay execution of a lower priority task. The general goals of the *SimulationAgent* are to minimize the number of resources, the frequency of resource switching between the tasks, and the number of resource conflicts.

10. If the executors and resources are not explicitly assigned to this task and only execution requirements are stated, the *TaskAgent* makes a request to the *ResourceManager* to select a resource appropriate for the task execution. Different criteria may be set as the search scenario criteria: a minimum cost; a required value of a conformity or competence factor; a minimum utilization factor; a highest or lowest priority; a shortest queue etc.

11. The *ResourceManager* searches for conformance with the stated requirements for the task execution according to each *ResourceAgent* or *ExecutorAgent* ontology. If several variants are available, the communication is established with the agent for which the largest number of requirements coincided, and which may be engaged in this task execution earlier than others agents.

12. When all resources are represented, the *TaskAgent* switches to the servicing status.

13. On completion of the main activity phase, the *TaskAgent* releases the engaged resources, and the *TransactionAgent* performs exit operations (if needed, activated are the rules for management of transactions transfer to the next model *TaskAgents*, decisions are taken on subsequent simulation direction).

14. On completion of interaction with the last of the work package model tasks, the *TransactionAgent* is deleted.

According to the suggested algorithm, the agents' statuses are reviewed and modified at each instance of an event occurrence.

Fig. 2 shows an example of the agent interaction in case of need to request resources for a task execution from the reserve fund.

Specific Features of the ExecutorAgents Interaction

The analysis of the executors information interaction in the project organizational management structure is of the main interest from the point of view of the agents individual behavior simulation. One of the key features of the *ExecutorAgents* is their integration into hierarchies (groups) according to organizational division in the management structure (divisions, departments etc.), resulting in creation of «horizontal» and «vertical» links between the agents.

As already stated, the organizational structure unit may have the following roles with respect to a task: an initiator (generation of task solving transactions), a decision-maker (gives an order for decision, delegates powers in the task solution management and coordination, task execution quality control), a manager (task solution management and control, delegation of powers in the task solution management and coordination), a coordinator (coordination of executors activities), direct executor (task execution, no management functions), intermediary (information transfer).

In [9], specified were general operations performed by the organizational structure units regardless of the specific management task being solved: preparation for execution, preparation of control action, information exchange, active management, passive management, direct execution, response formation, result processing.

In simulation of the executors' interaction, in most cases the agents negotiate under the *Request* protocol. The only distinction consists in availability of a special procedure for accumulation of statistical data on the time spent by the agents for execution of all the above operations. This is required for subsequent evaluation of effectiveness of their interaction within the framework of the organizational management structure.

In the course of exercising management, situations of parallel task execution occur, which entail complicated nested executors' interaction protocols. Situations may occur when the management roles are combined (usually, related roles are combined according to the order they were listed in), as well as situations requiring the task execution coordination. Coordination occurs only in case when solution of a certain subtask may depend on or influence results of solution of other tasks beyond the scope of the current task. Besides, in the general case, entering into negotiations, the agents may both compete and cooperate to achieve the set goal.

In case of a need for coordination of several executors' actions or organization of search for the most appropriate executors for a certain task, the *Contract Net Protocol* – based protocols are used.

According to the *Contract Net Protocol,* the initiator prepares a *call-for-proposals* to other agents, wherein it defines the task and certain conditions related

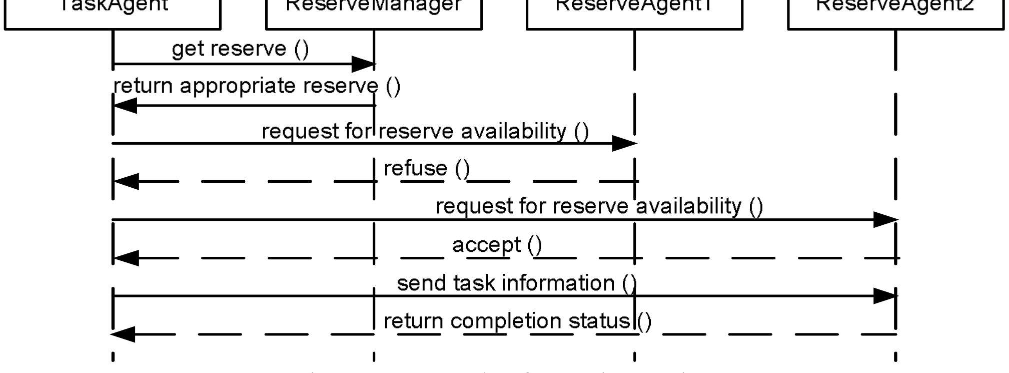
TaskAgent
ReserveManager
ReserveAgent1
ReserveAgent2
get reserve ()
return appropriate reserve ()
request for reserve availability ()
refuse ()
request for reserve availability ()
accept ()
send task information ()
return completion status ()

Fig. 2. An example of agent interaction

to execution thereof. The agents receiving this call are considered as potential executors, and analyze their workload, capabilities and conditions of the task execution. Some of the agents may reply to the initiator’s call with a *propose* to execute the task, while setting out their terms, including time, cost etc. On expiry of the proposals collection deadline (determined by the *reply-by* parameter), the initiator evaluates them and selects one or several agents to execute the task. Should it fail to select an agent for the task execution, the initiator may revise the conditions and make a new *call-for-proposal* to the *ExecutorAgents*. The selected agents are sent an *accept-proposal* message, the rest of the agents are sent a *reject-proposal* message. The initiator’s *accept-proposal* message receipt by the agent assigns the latter for this task execution. As soon as the executor completes the task execution, it sends an *inform-done* message on completion (only the fact of completion), or in its more informative form – *inform-result* (including results of the task solution). Should the executor fail to execute the task, it sends a *failure* message.

Therefore, only two agent roles are considered within the framework of the standard *Contract Net Protocol* – initiator and executor. In the general case, the agents interaction in the task solution management can be represented as follows. The initiator agent makes a request to the agent taking a decision on the task execution, which, in its turn, defines the *ManagementAgent*, to which the powers and responsibility in the set task execution will be delegated. The *ManagementAgent*, if needed, defines the agent, to which the powers and responsibility in coordination of the task execution by the *ExecutorAgents* are delegated. If the units for interaction have been defined in advance, the Request Protocol is used, if the unit to perform some role or other in the task execution management shall be searched for, the *Contract Net Protocol* is used.

An example of the *ExecutorAgents* interaction in the task solution management is given in Fig. 3.

To assess effectiveness of information interaction and manage the agents behavior strategies, it is reasonable to keep accounting of properties or criteria, which is especially important for the *ExecutorAgents*. The following can be mentioned among them: caution, independence, activity, management, resource, cooperativity, conflict ability. Thus, the agents’ behavior strategy will be defined depending on the degree of manifestation of the above properties, assessed according to the scales:

- «caution – risk» (a «cautious» agent bargains and, finally, finds the best proposal, and a «risky» agent immediately reserves the first proposal, even if it does not comply with its requirements);

- «activity– passivity» (determined by the rate of proposals advancing, acting as a task initiator);

- «independence-dependability» (determined by the ratio between the received and prepared control actions)

- management «centralization – decentralization» (indicates the agent’s inclination to a directive or collective management style);

- resource «centralization – decentralization» (characterizes the agent’s readiness to render resources for common use, or, vice versa, borrow someone else's resources);

- «cooperativity – competitiveness» (characterizes the agent from the point of view of integration in case of resource shortage);

- «consent – conflict ability» (if this estimate is high, the agent is inclined to frequently cancel agreements, if low, the agent is inclined to long-term cooperation).

Besides, the use of the multiagent approach enables solving such problems as: horizontal compression (several works are integrated into one work performed by one unit); vertical compression (in the points of task execution, where the executor shall apply to the management hierarchy, it takes independent decisions);

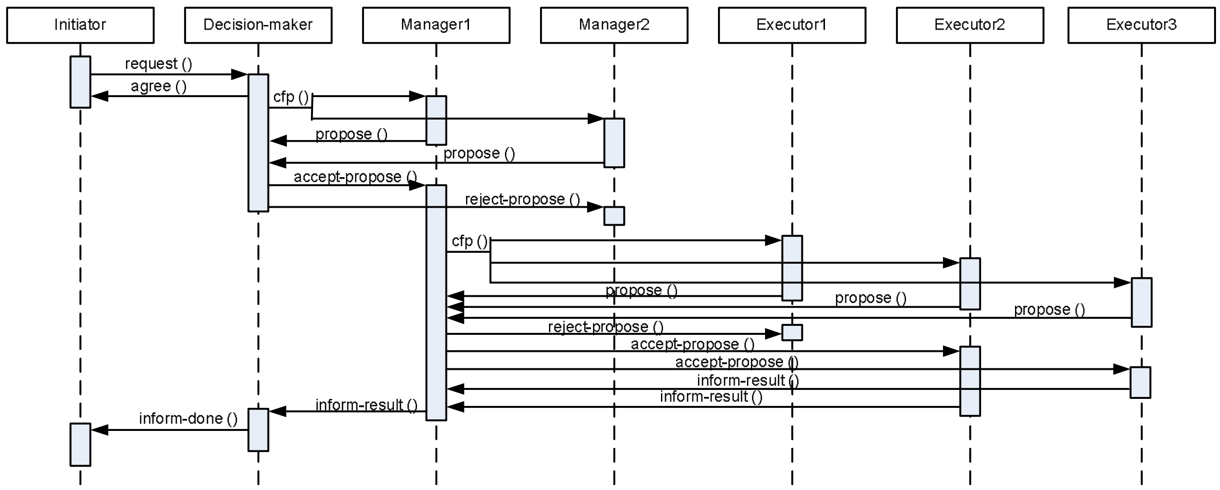

Fig. 3. The *ExecutorAgents* interaction in the task solution management

selection/change of executors' roles in the organizational structure; simulation releases from linear works ordering, allowing executing tasks in their natural order (i.e. to execute tasks in parallel where it is possible), which shortens the information query paths, and finally accelerates a process execution, redistribution of managerial and executive loads, reduces checks and coordinations in the course of management processes execution.

The Agent Interaction with Allowance for Risks

Paper [5] deals with internal mechanisms of the dynamic parametric and structural change of the simulation model resulting from taking into account the resource, organizational, and financial risk factors, consequences of their manifestations and counteraction measures. We shall consider specific features related to the agents interaction in the model for some of them.

For example, the resource recovery simulation mechanisms may include the work transfer to a free resource, repair with own means, transfer for repair, work execution transfer to another enterprise. Each of the mentioned recovery options mostly assumes one extra work or a whole extra work package, including not only the basic recovery operation, but auxiliary works as well (dismantling, transportation, diagnostics, purchase, installation of a new resource, start-up and commissioning etc.).

For instance, to avoid the risk of the works breakdown due to late delivery of materials, we shall add the task «Make an advance order for materials» to the work plan. Let us consider the nature of negotiations between the *TaskAgents* when a new work appears. Let the executor initially have three tasks: Task1, Task2 and Task3. Now let us suppose that a new Task4 emerges, having an early completion date, causing a conflict with Task2. The *TaskAgent4* initiates the *TaskAgent2* and asks if it can give its position in the executor's schedule. The *TaskAgent2* checks for its early completion date and finds out that it has some reserve, and, potentially, it could have been replaced, but a conflict with Task3 emerges. The similar negotiations start between Task2 and Task3, if from ontology these tasks are known to be independent, and Task2 result is not input to Task3. As is seen from Fig. 4, Task3 has no time reserve, and in this case it cannot be postponed to a later period, but it can be advanced to an earlier period, where there is a suitable time window. Finally, the system finds an option solving the conflict between Task4 and Task2, and Task4 calls the *ManagementAgent* to approve modifications in the executor's schedule.

Therefore, the schedule structure, formed as of some instant of time, was partially (locally) destroyed with a new task emergence, whereafter a new schedule structure was recovered, reflecting a new balance of interests between all the tasks. In this case, adding this task did not influence the project duration. At that, this structure was destroyed and recovered by the tasks themselves in the course of negotiations and decision making based on knowledge about themselves, including descriptions of possible work consequences, their priorities, duration etc.

Besides, the agents can take care of preservation or development of their resource type. For example, the incompetence risk is simulated, which influences the increase in duration of the project works performance, creates extra work or a nested model corresponding to retraining of specialists and their professional improvement (search for and hiring new specialists). Let the customer demand for utilization of a new technology familiar to none of available specialists (checked for conformance according to each *ExecutorAgent's* ontology). Consequently, the programming works may take up to 25% more time (productivity factor = 1.25, and, as a consequence, a penalty for noncompliance with the scheduled project dates, probable value of coding labor consumption: 1.25x20 = 25 man-days, probable value of coding duration: 1.25x4 = 5 months). If the programmers are sent for technology training, they become inaccessible for the training period, the project costs increase by 4x$1000 (cost of one specialist training) = $4000. However, the expected value of the productivity factor will drop to 1.1. Another behavior scenario may consist in the following option: to appoint one of the programmers as a group leader, and allocate 1 day per week for him for the technology study (on this day, the executor becomes inaccessible for execution of other project works) and elaboration of company standards to be used by other specialists. Let overall costs for the group leader work and training make 5 working

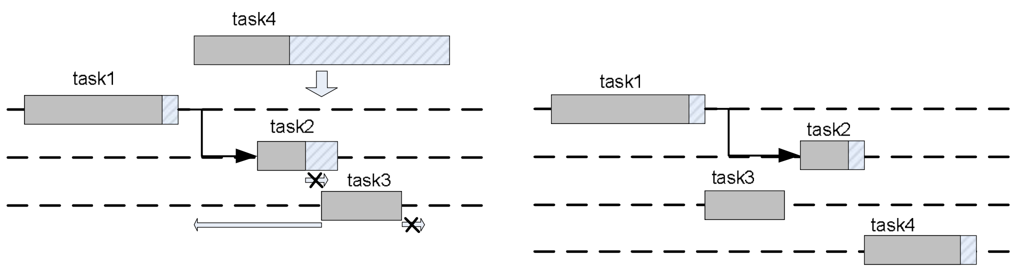

Fig. 4. An example of conflict resolution with tasks rescheduling

days. At that, the productivity factor will drop to 1. In this case, negotiations between the agents start, related to selection of the most efficient way of the situation resolution. Moreover, the *ResourceAgent (ExecutorAgent*), finding out that it failed to obtain the project work once again due to the lack of required skills, will be able to find a spare «window» in its schedule and to plan the work to master this knowledge.

It becomes possible to delete nearly any agent in the system (for example, a resource, executor or an entire division agent), whereupon the project is not completely disrupted, since the system tries to find alternate forms of its implementation, gradually «closing the wound» and demonstrating its stability.

Using the Agent Catalogues

For the *ManagementAgents* in charge of coordinating the other agents work within the framework of multiagent platforms (according to FIPA standards), the agent catalogues are created (Directory Facilitator, DF) – the "yellow pages" service, allowing the agents to search for other agents according to the type of the service rendered. We shall deal with peculiar features of the agent interaction with the use of the Directory Facilitator (Fig. 5).

Fig. 5 shows the process of negotiations between two agents. The TaskAgent1 is willing to initiate execution of "Task1", to which an appropriate executor is assigned. The *ExecutorAgent* is registered with the Directory Facilitator as an agent capable of receiving *achieve* messages, containing this task as a content, which means that this agent is capable of executing Task1. As an example, we shall consider the case when only one such agent is registered with the Directory Facilitator, though, in the general case, there may be several agents having placed a similar advertisement. First, the TaskAgent1 communicates with the Directory Facilitator and makes a request about an *ExecutorAgent* capable of executing the task. Then, the Directory Facilitator recommends the ExecutorAgent1 to the TaskAgent1 for the task concerned. Afterwards, an auction starts with the TaskAgent1's sending a respective message to the ExecutorAgent1. The ExecutorAgent1 responds to this message with a *propose* message, and if the TaskAgent1 had not received any better proposal, it accepts the ExecutorAgent1's proposal and confirms it by sending an *accept* message along with the execution terms in its informative portion. After that, the ExecutorAgent1 becomes engaged in the "Task1" execution.

Conclusion

The paper deals with issues associated with utilization of the multiagent approach in simulation modeling of production systems. The emphasis was laid on mechanisms of the agents' information interaction in the suggested system. The major advantages of the approach have been demonstrated, related to the possibility to create elements of individual behavior, dynamically changing in the process of simulation, using adaptation and self-organization mechanisms. This allows effective solving issues related to simulation and analysis of the executor agents in the organizational management structure, works distribution among the executor agents, modifications emerging during the simulation due to risk factors manifestations.

References

1. *Vittikh, V.A. Multiagent models of interaction in decision-making processes [Text] / V.A. Vittikh, G.A. Rzhevsky, P.O. Skobelev // Proc. 4th Int. Conf. in Management and Modeling of Complex Systems. – Samara: SNTsRAN, 2002. – P. 116-126.*

2. *Finin, T. KQML as an agent communication Language. In Software Agents [Text] / T. Finin, Y. Labrou // MIT Press. – Cambridge (England): Mass, – 1997. – P. 291-316.*

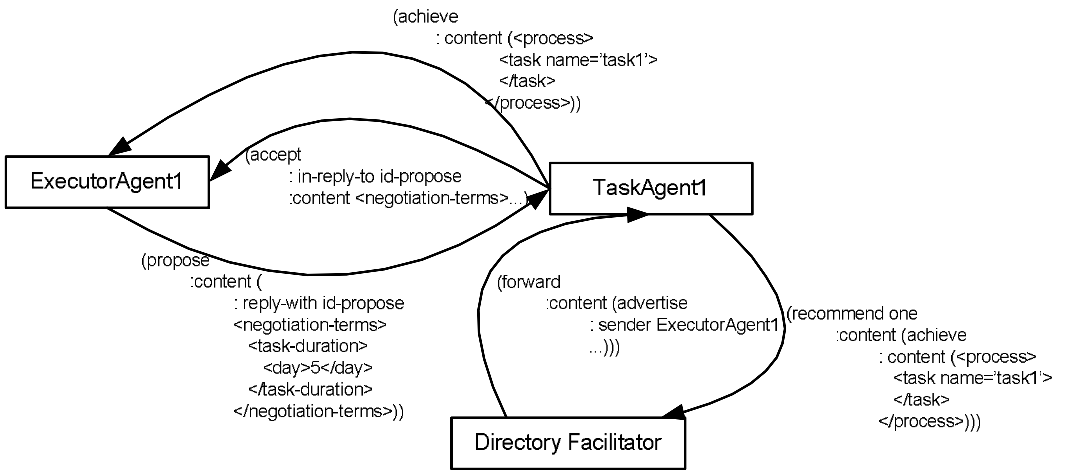

Fig. 5. An example of the agent interaction with the use of the Directory Facilitator

3. Genesereth, M.R. Knowledge Interchange Format, Version 3.0. Reference Manual. Technical Report Logic-92-1 [Text] / M.R. Genesereth, R.E. Fikes // Computer Science Department, Stanford University. – California (USA). – 1992. – P. 7-13.

4. FIPA ACL Message Structure Specification. Foundation for Intelligent Physical Agents. [Electronic resource] – 2002. – Режим доступу: http://www.fipa.org/specs/fipa00061/SC00061G.html. – 11.06.2013.

5. Fedorovich, O.Ye. Simulation model for analyses in project management processes due to risk factors [Text] / O.Ye. Fedorovich, O.V. Prokhorov, O.M. Zhigulina // Aerospace Engineering and Technology: Research and Engineering Journal. – National Aerospace University Kharkiv Aviation Institute. – Kharkiv (Ukraine). – 2007. – Issue 1(37). – P. 75-84.

6. Borshchev, A. From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools [Text] / A. Borshchev, A. Filippov // The 22nd International Conference of the System Dynamics Society. – Oxford (England). – 2004. – P. 238-247.

7. Yemelianov, V.V. Multiagent model of decentralised management of a production resources flow [Text] / V.V. Yemelianov // Proc. Int. Conf. Intellectual Management: New Intellectual Technologies in Management Problems ICIT'99. – M.: Fizmatlit, 1999. – P. 121-126.

8. Smith, R. The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver [Text] / R. Smith // IEEE Transactions on Computers. – 1980. – Vol. C-29, No. 12. – P. 1104–1113.

9.

10. Gorlov, D.A. Building protocols for controlling functional tasks in complex socio-engineering systems [Text] / D.A. Gorlov // Aerospace Engineering and Technology: Research and Engineering Journal. – National Aerospace University Kharkiv Aviation Institute. – Kharkiv (Ukraine). – 2003. – Issue 8(43). – P. 80-85.

11. Ermolayev, V.A. Cooperation Layers in Agent-Enabled Business Process Management [Text] / V.A. Ermolayev, S.L. Plaksin // Problems of Programming. – Pennsylvania (USA). – 2002. – V.1-2. – P. 354-368.

Поступила в редакцию 1.06.2013, рассмотрена на редколлегии 12.06.2013

Рецензент: д-р техн. наук, профессор, зав. кафедрой «Автоматизированные системы управления» М.Д. Годлевский, Национальный технический университет «Харьковский политехнический институт», Харьков.

ВЗАЄМОДІЯ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ В СИСТЕМІ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ ВИРОБНИЧИХ ПРОЦЕСІВ

О.М. Пахніна

Розглянуто механізми взаємодії інтелектуальних агентів при моделюванні потокових процесів. Наведено основні переваги підходу, пов'язані з можливістю створення елементів індивідуальної, що динамічно змінюється в процесі моделювання, поведінки, за рахунок механізмів адаптації та самоорганізації. Розглянуто питання інформаційної взаємодії виконавців в організаційній структурі управління з урахуванням аукціонних механізмів розподілення робіт та динамічних змін, що виникають в процесі моделювання внаслідок виявлення факторів ризику.

Ключові слова: агенти, імітаційне моделювання, взаємодія агентів, протокол, онтологія, організаційна структура, виконавці, ризики.

ВЗАИМОДЕЙСТВИЕ ИНТЕЛЛЕКТУАЛЬНЫХ АГЕНТОВ В СИСТЕМЕ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ПРОИЗВОДСТВЕННЫХ ПРОЦЕССОВ

Е.М. Пахнина

Рассмотрены механизмы взаимодействия интеллектуальных агентов при моделировании потоковых процессов. Показаны основные преимущества подхода, связанные с возможностью создания элементов индивидуального, динамически меняющегося в процессе моделирования, поведения, за счет механизмов адаптации и самоорганизации. Рассмотрены вопросы информационного взаимодействия исполнителей в организационной структуре управления с учетом аукционных механизмов распределения работ и динамических изменений, возникающих в процессе моделирования вследствие проявления факторов риска.

Ключевые слова: агенты, имитационное моделирование, взаимодействие агентов, протокол, онтология, организационная структура, исполнители, риски.

Пахнина Елена Михайловна – инженер каф. Информационных управляющих систем, Национальный аэрокосмический университет им. Н.Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: elena.pakhnina@khai.edu.