Multi-Agent Modeling of Project Management Processes in Distributed Teams

Olga Cherednichenko[a], Oleksandr Matveiev[a], Olha Yanholenko[a], Rositsa Maneva[a]

[a] *National Technical University "Kharkiv Polytechnic Institute", Kyrpychova str., 2, Kharkiv, 61002, Ukraine*

Abstract

Changes in the business environment, the innovative nature of projects, lack of necessary skills of project team members lead to increased uncertainty and inability to plan with a given degree of accuracy. Such projects use adaptive project and program management methodologies. In the field of information technology, the use of multi-agent systems is of particular interest. In the context of the use of multi-agent systems for the design of intelligent systems for various purposes, the development and study of a model and software implementation of a prototype of an agent platform are relevant. The aim of this work is to develop and research an agent platform that can be implemented in the work of the distributed team in order to improve the assignment of tasks. The paper presents the formal agent architecture as a basis of multi-agent model. The task assignment is a case study to implement and test multi-agent model prototype. The agent platform is developed based on Kotlin programming language. A prototype of the agent platform based on the FIPA specification allows to increase the productivity, scalability and interoperability of multi-agent system.

Keywords 1

Project management, distributed team, task assignment, multi-agent platform, formal agent architecture, prototype

1. Introduction

The success of the company in the market depends on many factors: the range of services offered, market saturation, marketing policy, and so on. In order to maintain competitiveness, modern companies focused on continuous development are forced to constantly improve their activities, which requires the development of new technologies and methods of doing business, the introduction of more effective methods of management and organization. Changes in the business environment during the crisis, the innovative nature of projects, lack of necessary skills of project team members, the impact of the human factor lead to increased uncertainty and inability to plan with a given degree of accuracy. Such projects use adaptive project and program management methodologies.

Modern enterprises are facing an increasingly competitive market. In this context, the success of enterprises critically depends on the quality and efficiency of their business processes. The analysis of processes is aimed at building and improving models of functioning of interacting physical or information objects based on information processing that records such behavior [1]. The approach presented in this paper is aimed at building a software system for the formation of the organizational structure of work in real time and monitoring the effectiveness of project management processes in a dynamic business environment.

There are many pieces of research related to methods and methodologies of project management in the IT field [2, 3, 4]. For instance, in [2] the method of synthesis of the project management

Proceedings of the 2nd International Workshop IT Project Management (ITPM 2021), February 16-18, 2021, Slavsko, Lviv region, Ukraine
EMAIL: olga.cherednichenko@khpi.edu.ua (A. 1); matwei1970@gmail.com (A. 2); olha.yanholenko@khpi.edu.ua (A. 3); maneva.rositsa@gmail.com (A. 4)
ORCID: 0000-0002-9391-5220 (A. 1); 0000-0001-5907-3771 (A. 2); 0000-0001-7755-1255 (A. 3); 0000-0002-4361-1931 (A. 4)

methodology is considered. In [3] an adaptation method for the Agile project is introduced. The paper [4] investigates the issues of synthesis of the project team in an Agile environment. Machine learning techniques are proposed and team formation is considered [4]. Analysis of scientific publications shows that the issue of effective implementation of flexible management methodologies is becoming increasingly important. Thus, the authors [5] prove that differences in organizational types affect the success of projects in all aspects. The work [5] emphasizes that flexible methods work better than traditional methodologies, even in large, distributed projects. Building better communication and collaboration in the team is seen as the result of flexible practices that lead to improved relationships between team members and improved employee satisfaction. This becomes especially relevant with the development of virtual distributed project teams. In particular, the work [7, 8] is devoted to the study of the features of virtual commands.

Due to the fact that the structure of multi-agent systems is very close to the structures of the real world, so the scope of agent systems is very wide: robotics; logistics; research of social and biological systems; information search; the internet of things; software components for smart homes. Multi-agent models are actively used to improve the business processes of the organization. In [9] the ontology of business process modeling and multi-agent system of providing intellectual assistance to members of the software development team are proposed. Studies of the development of information systems to support management decisions have shown that agent technology can be effectively used. An agent platform is usually understood as a set of software interfaces that provide the ability to create, lifecycle, messaging, communication and access to information and knowledge bases of agents [10]. The agent platform is the most important part in the process of creating an agent system. Simulation systems are created to develop and plan the work of project teams [11]. The proposed in [11] system can be used as a virtual environment for selecting project management methods and tools. The system simulates management processes and project roles. Thus, studies of agents and multi-agent systems have been conducted for a long time, but the task of building an architecture of multi-agent system for management of project distributed teams is still not fully solved.

The aim of this work is to develop and research an agent platform that can be implemented in the work of the distributed team in order to improve the assignment of tasks.

2. Problem Statement

E-business has become a wide-spread and quite popular form of business architecture nowadays. It is different from traditional business architectures in many ways. Although any e-business has some common components like marketing, engineering, financial, administrative, CRM (customer relationship management) divisions, its operation division is actually represented by a distributed set of orders and contractors. Customers and executives meet each other on the online platform (fig. 1) and do not have to interact directly like in traditional business organizations. Online e-business platform supports continuous communication between both parties and provides all business logic functionality. While customers can post their orders (jobs) in real time mode, executives can post their contractors (services) to satisfy customers' needs. In such a distributed business architecture, the number of orders and contractors is not stable and can vary within time. This makes a distributed structure of e-business platform complex enough for analysis and management process.

At the same time, e-business exists in the external macro-environment. The online character of business operations is regulated and controlled to some extent by international and country-specific rules and policies. Social factors affect customers' needs and supply side as well. Moreover, of course, technological innovations and trends define the speed and new directions of development and growth of e-business in all areas.

E-business often deals with software projects that in terms of project management combine groups of people who have to perform some jobs like for example, requirements specification, software design, construction and testing. Primarily, project management is explained by the complexity and comprehensive structure of the projects. A complicated project consisting of several tasks cannot be implemented by a single person and even if it could be realized, it would take too much time for completion. So the natural solution of this problem is to organize the team of people, who can perform the necessary volume of work in a shorter time.

All basic processes of software engineering require the cohesive teamwork. Modern enterprises that produce the software work usually according to a software development methodology. This can be either a waterfall-based, or Agile methodology [4]. Waterfall-based methodologies consider "fundamental process activities of specification, development, validation, and evolution and represents them as separate process and phases such as requirements specification, software design, implementation, testing". The examples of such methodology realization include Rational Unified Process (RUP), Microsoft Solutions Framework (MSF), which are "huge" and documentation-intensive methodologies. As an opposite of waterfall methodology we can consider agile methodologies. For example, they include eXtreme programming and SCRUM. The Agile manifest states that it is necessary for business people and developers to work together daily throughout the project. It pays more attention to trust individuals and interactions over processes and tools. The role of self-organizing teams is essential in agile methodology.

Agile methodologies of software project management always deal with managing group dynamics in globally distributed teams where the communication channels play an essential role in group member's interaction. The theory of the organization and the concept of the process approach to the management of the distributed teams indicate that the main recipient of the process is the user, and his view of the business process is the most important. Therefore, the use of software agents in e-business distributed teams can support and improve these processes.

A multi-agent system is a system formed by several interacting agents. To build agent-oriented systems, many methodologies and appropriate tools designed to automate their software implementation have been proposed [12]. The basic tool for developing intelligent multi-agent systems that allows to create, destroy, interpret, run and move agents is the agent platform. The main functions of agent platforms are: organization of agent interaction; transmission of messages within the platform and between different platforms; ontology support; management of agents, their life cycles; search for agents and data about them within the system; security agents.

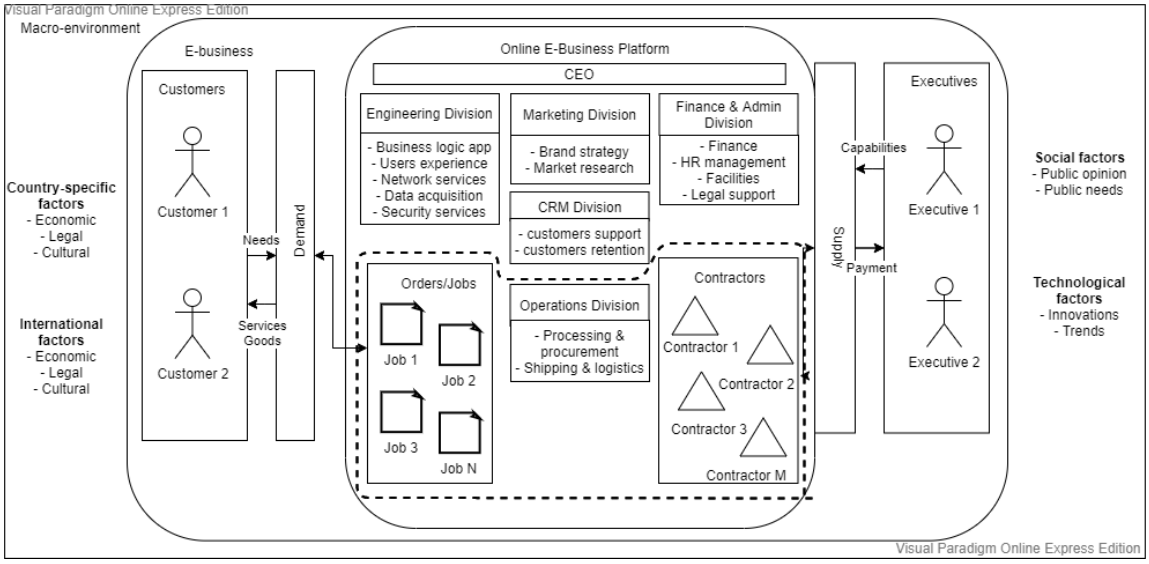

Figure 1: E-business overview

Today, the methodological basis for creating an agent platform is determined by the standards of MASIF (Mobile Agent System Interoperability Facility) [13] and FIPA (Foundation of Physical Intelligent Agents) [14]. The purpose of the MASIF standard is to achieve a certain degree of compatibility between the platforms of mobile agents from different manufacturers. The main specifications of FIPA are the Agent Management (AM) standard and the standardized Agent Communication Language (ACL). FIPA standards provide basic definitions of agent communication concepts. The structure of the agent platform is not included in the FIPA standards and may be based on different architectural approaches. An agent platform is a physical infrastructure in which agents

can be deployed. The concept of an agent platform allows to use different approaches to its organization of software architecture.

Therefore, the main task of this work is to develop and research an agent platform that can be implemented in project management as a tool for modeling and investigating real-time business processes.

3. Fundamentals of Agent-Based Simulation Modeling

Talking about Agile methodology of project management, the project is considered as a finalized result of every development cycle. Implementation of each task by distributed team members requires their pre-assignment. In this work it is suggested to perform tasks definition and assignment among group members in real time mode based on agent technologies.

From the point of view of software engineering, an agent is an autonomous software entity that has a long existence and adapts its behavior to changes in the environment, as well as the ability to interact with other agents. In artificial intelligence systems, an agent is understood as an entity that is able to perceive information through sensors and influence the environment through actuators. The main features of agents are their autonomy, interactivity and ability to learn. Agents can decide for themselves how to behave in order to achieve a goal. Agents can also interact with each other in a similar way to social interaction. The ability to learn is associated with an intelligent agent from artificial intelligence systems. The agent approach to software design combines the features of object-oriented and component methodologies, allowing to implement certain properties of the agent. Agent methodology allows designing multi-agent systems, consisting of agents who cooperate with each other to achieve individual and common goals.

The main objective of simulation is to investigate the real-life processes. The simulation method allows to create a model based on the execution time of the operation and with the means of analyzing the dynamics of business processes. The general algorithm for modeling the system can be divided into four points:

1. Build a model of the process that has to be analyzed.
2. Run a simulation of the process models.
3. Analyze the results.
4. If necessary, repeat the previous steps for other scenarios.

Once the model is designed, you can run the simulation. Estimates of indicators are formed by means of repeated changes.

There are several directions in the modeling and simulation theory. Each area is characterized by its own tools and simulation systems. Here are some examples:

- modeling of dynamic systems (implemented in MATLAB Simulink, VimSim systems, etc.);
- discrete-event modeling (Arena, Aris, AnyLogic, AutoMod, etc.);
- system dynamics (PowerSim, iThink. AnyLogic, etc.);
- agent modeling (Swarm, AnyLogic, Repast, etc.).

System dynamics and modeling of dynamic systems involve time-continuous processes. Discrete-Event and Agent Modeling are discrete. The most popular approaches in solving the problem of business process analysis are considered to be discrete-event modeling and agent modeling.

In discrete-event systems, modeling is based on the description of processes. The advantages of this model include statistical data processing at the input and output, optimization, relative ease of implementation, interface. However, this model does not allow modeling at a high level. In agent modeling, the dynamics and functioning are determined as a result of individual activity of model participants. The advantages of the system include any level of detail, optimization, interface, realism in the description of the system. The disadvantages are the complexity of describing the logic of the agent's behavior in terms of the model, high power costs.

Thus, with the help of simulation it is possible to perform any process in the accelerated time mode in the conditions corresponding to reality, taking into account delays and breaks. Analyzing the results obtained, it is possible to identify resources in need of unloading (resources to which queues are often lined up), low-load resources, identify shortages or overworking.

Multi-agent system can be used to solve many problems that cannot be solved with a single agent or a monolithic system. A multi-agent system (MAS) is a system formed by several interacting agents. In MAS, all elements must interact with each other. A system can be considered as a MAS, not just a community of agents, only if all agents act to achieve one goal. However, the lack of a common goal for agents does not deny the possibility of their group behavior. It is important to note that agents that are part of MAS can be reused or developed to solve more versatile agent tasks. In such cases, the agents may have their own goals, which do not completely coincide with the goals of the system, but are compatible with them. Such agents can be useful to each other in solving their tasks. Therefore, for agents included in the MAS, the most important property is communication.

For successful operation, the agent must have receptors, effectors, processor and memory. Receptors are special devices that receive information from the environment. Effectors are executive bodies that act on the environment. Memory is the ability of an agent to store information about its state and the state of the environment. Receptors form a system of perception of the agent, providing the reception and primary processing of information coming from the environment and then sent to memory. The system of perception can control actions by comparing the differences between the existing and expected state. The agent's memory should contain data on typical responses to information signals from receptors, as well as information on the state of effectors and resources. In addition, the memory must store programs for processing input information into control signals that are applied to the effectors and the results of reactions to a situation.

To build agent-oriented systems, many methodologies and appropriate tools designed to automate their software implementation have been proposed [10, 15-21]. Therefore, the choice of the best methodology and tools taking into account the specifics of the object of automation is an important task. The basic tool for developing intelligent multi-agent systems that allow to create, destroy, interpret, run and move agents is the agent platform.

The most well-known are the following agent platforms [15, 16]:

1. JADE (Java Agent DEvelopment). The main characteristics of the JADE agent platform include:
    - compliance with the latest FIPA 2.0 specifications;
    - openness of the LGPL (Lesser General Public License) code;
    - support of multiplatform technologies;
    - support of wireless mobile devices up to JME (Java Platform, Micro Edition) MIDP (Mobile information device profile) 1.0 (cell phones);
    - availability of own and third-party libraries for extending agent interaction protocols (RDF, XML);
    - agent management;
    - management of the platform itself;
    - statistics.

On the basis of this agent platform it is possible to implement an autonomous agent which will be able to fully control its management influences within the functions assigned to it, independently manage its life cycle, will be able to perform several parallel tasks. Because JADE supports intra-platform mobility and agent cloning, in practice the platform can be distributed between multiple nodes. Each node may have several agents, each of which will be responsible for performing its task and, at the same time, will be able to move from one node to another if necessary. The disadvantages of this agent platform include the openness of the code, which is unacceptable when developing special-purpose communications.

2. Coguaar (Cognitive Agent Architecture) - the most powerful project, supported by DARPA (Defense Advanced Research Projects Agency). Considering the Cougaar agent platform, it can be noted that it has an open source architecture. Agents are autonomous formations of software that interact with other agents and with the external environment, which can ensure the reliable operation of the node. Software agents are implemented on the basis of a programming methodology that allows decomposition of complex tasks into simpler ones. Agents will be able to manage the MR node, change its state according to changes in the environment.

3. Aglobe does not have enough support of the FIPA (an IEEE standard organization that promotes agent-based technology and their interaction with other technologies), has many customizable features.
4. Jack – one of the few platforms that use the model of agent logic, based on the principles of beliefs-desires-intentions (BDI) and built-in formal-logical planning of the agent.

Research on agents and multi-agent systems has been conducted for a long time, but the task of building a universal architecture of multi-agent systems is still not fully solved. The reason may be the narrow focus of most research in the subject area. However, to determine the generalized architecture of the multi-agent system in a particular case is quite possible.

Therefore, agent-based simulation appears to be a powerful tool for modeling of project management processes. Agents and their communication mechanisms provide a framework for modeling of interactions in distributed teams and tasks assignment within a team. To build an appropriate simulation model it is necessary to have a formal representation of an agent which is one of the main results of the given work.

4. Results

The main idea of this research is to model project management processes in distributed teams via interaction of intelligent agents who can solve the problem of jobs assignment among executors. In this paper, the agent will be understood as a computer system placed in the external environment, able to interact with it, performing autonomous rational actions to achieve a certain goal [14]. The main difference between agents and systems in general is activity, i.e. the ability to perform any actions independently. In addition, the agent is usually considered not as a set of parts, but as a single entity, while, for example, when studying the properties of systems, the first approach is the main one. Another characteristic is that the agent can be embodied not as a material object, but as a stand-alone program. However, this program, without affecting the material world (remaining within the computer or computer systems), can perform useful actions.

Agents are subject to the following basic characteristics [15]:

1. Autonomy. The agent is able to function without direct human intervention, independently monitoring the state of the environment and its own parameters.
2. Reactivity. The agent is able to perceive the external environment and respond adequately to it.
3. Pro-activity. The agent has purposeful behavior and can take the initiative.

In addition, to perform certain tasks, agents must meet additional classification requirements [15]:

1. Ability to communicate. The agent must be able to communicate and interact with other agents or people.
2. Modeling the situation. The ability of the agent to model the development of the situation, to predict the course of its development.
3. Mobility. The agent must be able to change his position in the environment.
4. Intelligence. The agent must be able to draw a logical conclusion to decide on their next steps.
5. Binding to the environment. The agent must exist in a specific environment (real or virtual).

To present the possible actions of the agent and its interaction with the external environment, it is necessary to have a tool that allows to formally describe the behavior of the agent. The behavior of the agent is the basis of its formal architecture (Fig. 2). Formal agent architecture is a tool that allows to design agent behavior using clear formal methods. The formal architecture of the agent is defined through the description of the environment in which the agent operates, the agent's perception of this environment and its actions.

Let’s denote the external environment of the agent by the set of states S. Possible actions of the agent are described by the set of actions A. In the abstract, the agent can be represented as a function

$$g_S : S \to A \qquad (1)$$

that is, the choice of a particular action from a set of possible actions is carried out by the agent on the basis of the current state of the environment $s_i \in S$. The actions of the agent may affect the environment, but not control it completely. It is convenient to use the model of perception of the external environment to represent the agent. To do this, let’s introduce a set of possible perceptions of

P and a function $f : S \to P$ which describes how certain environmental conditions are perceived by the agent. Then the agent is represented by a function

$$g_P : P \to A \quad (2)$$

that is, the action of the agent is determined in the general case by the current perception of the state of the environment $p_j \in P$.

The model of the agent with perception is equivalent to the basic one. However, it allows you to introduce the following additional property of the agent: different states of the environment can be perceived in the same way and vice versa - one state can be perceived differently by the agent.

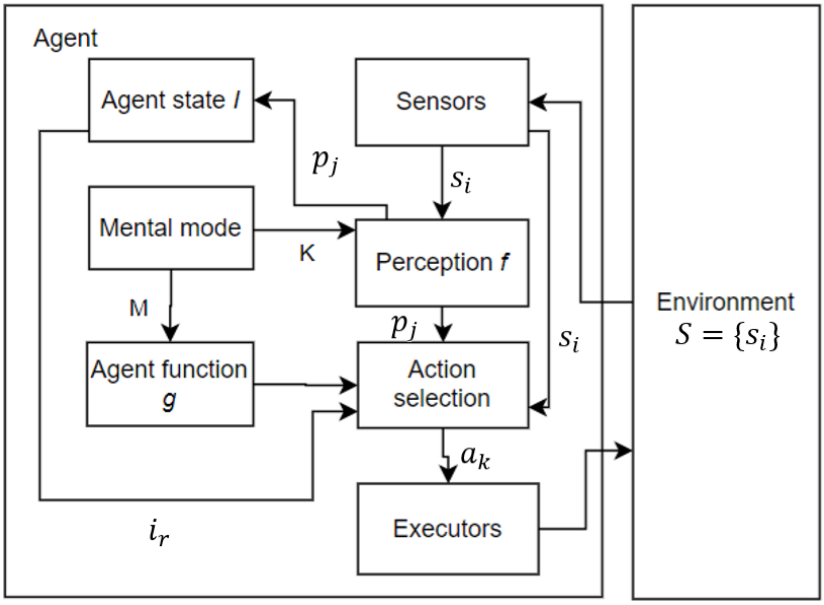

Figure 2: Formal agent architecture

Another option for solving the problem of including previous actions when selecting the current action is to introduce the concept of the state of the agent. It is believed that the agent has certain internal data structures, which it modifies depending on the perception of the current state of the environment, and on the basis of the obtained results chooses the action. To formalize this process, let's introduce a set I of internal states of the agent and the function of updating the internal state, which is responsible for updating the internal state in accordance with the current perception of the environment:

$$h : I \times P \to I \quad (3)$$

Then the agent is described by a function

$$g_I : I \to A \quad (4)$$

that is, the action is selected based on the current state of the agent. To correctly describe the behavior of the agent with the state, it is necessary to determine the initial state.

This agent architecture has one significant drawback, namely that the agent specified in this way does not receive information about his actions, which limits his ability to gain experience and analyze the potential consequences of his actions. One possible way to overcome this shortcoming is to present information about the agent's actions as part of the information about the external environment, but this approach is not clear and intuitive. A more correct solution to this problem is to include information about the actions performed explicitly in the input data of the action selection function:

$$g_A : (P \times A)^* \to A \quad (5)$$

In this form, the agent clearly receives information about the actions already taken and when choosing an action it uses the perception of environmental conditions.

For an agent with a status, information about previous actions is taken into account in the status update function:

$$h: I \times P \times A \to I \quad (6)$$

The parameter of the status update function is not the sequence of all actions of the agent, but only the last performed action.

When using approaches based on formal logic, the function of perception and choice of action of the agent is described as a set of statements, or rules, of this logic. Doing so, the agent maintains a knowledge base that contains a set of statements of formal logic that describe the causal relationships between the state of the environment and the perceptions of the agent, as well as between perceptions of the environment and the actions of the agent. Such an agent is called a logical agent.

An agent who chooses an action based on the current perception, ignoring the entire history of previous perceptions, is a simple reflex agent. This type of agent is quite simple. In many cases, knowledge of two types may be required for an agent to function successfully. On the one hand, it is information about how the environment changes independently of the agent. On the other hand, it is the knowledge of how the agent's own actions affect the environment. An agent that uses such knowledge about the existence of the external environment is a reflex agent based on the model.

The suggested formal architecture of an agent can be used for modeling of interaction processes inside the distributed team. The ability of an agent to percept the environment and act according to its mental model allows, for example, to simulate tasks assignment within a team and find solutions of project management problems.

5. Case Study

As a case-study example, let’s consider the following statement of the problem. Two types of users of the information system of the e-business organization are given: the customer and the executor. Each customer with its order forms a set of requirements for the task to be performed. Each executor is characterized by a set of restrictions on the ability to perform on each type of tasks of the organization. Requirements for tasks and limitations of their solutions are set by the system of predicate expressions and are stored by the customer and the executor agents, respectively. It is necessary to model a multi-agent system of communication between customers and contractors so that when adding a contractor to the system, each customer can have the opportunity to "negotiate" with a new contractor. Accordingly, when adding a customer, each executor has the opportunity to "agree" on a new task. We determine that each customer and each contractor must form their own list of contractors and customers who meet or not their requirements and restrictions.

Applying the above problem statement to the process of tasks assignment in distributed teams, customers are the source of tasks in the given software project and executors are team members. Agents of team members can be added to the system or deleted from it in real time mode, thus, modeling the processes of software development when a team can be reinforced by additional members or when some members may leave a project. In the same way, agents of tasks are continuously added or removed with the appearance of new requirements or implementation of the old ones.

To model such agent-based system, agent.kt was used to provide the functionality of agent platform system according to FIPA standard. The component diagram on figure 3. describes the components of designed agent platform.

Component “OrdersFacilirator” contains set of “OrderAgents” registered in it. Each order after consuming a message from executor checks requirements fulfillment to provide “handshake” or “cancellation” between itself and executor. Componet “ExecutorsFacilirator” contains a set of “ExecutorAgent” registered in it. Executor`s agent behavior is similar to “OrderAgent”, but instead of requirements it contains set of restriction. So if restrictions meet order agent’s restrictions, executor agent sends the “handshake” request to correspond order-agent and “cancellation” request otherwise. Each agent contains a list of appropriate and inappropriate agent identifiers to refuse the request from already requested agents. Class diagram of agent behavior is described in figure 4.

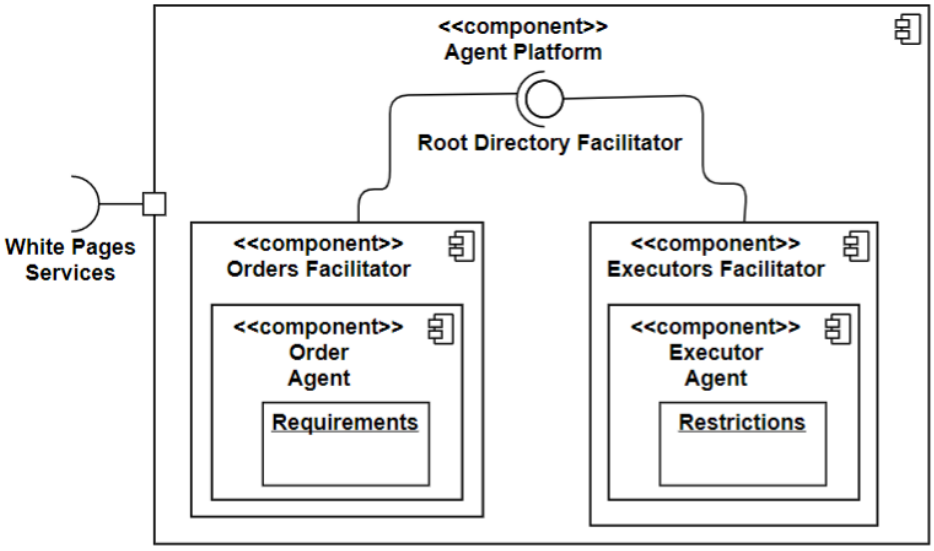

Figure 3: Component diagram

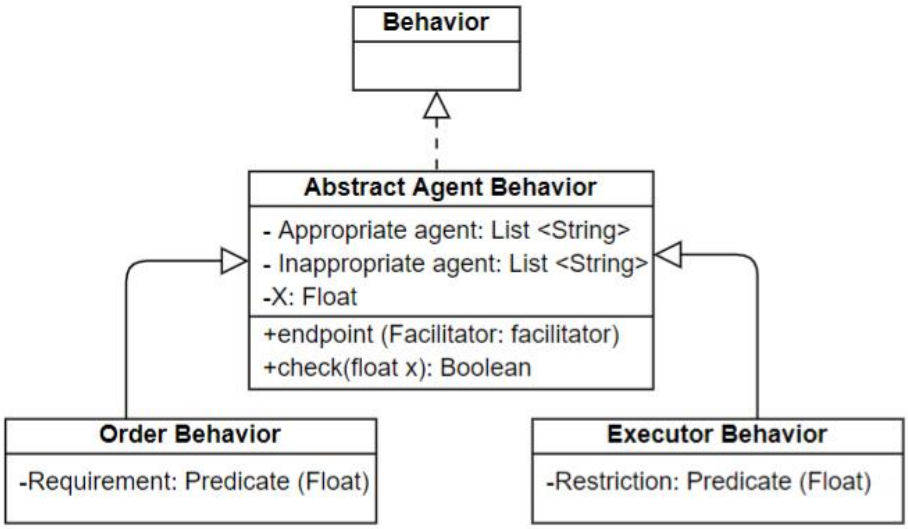

Figure 4: Class diagram

The main purpose of current research is to check the performance parameters of the developed system. That`s why we use performance testing (MEMORY, OPERATION, CPU analysis) to check the usability and reliability of developed agent platform. Environment: MacBook Pro (15-inch, 2018), processor: 2.2 GHz Intel Core i7, memory: 16 GB 2400 MHz DDR4, graphics: Radeon Pro 555X 4 GBIntel UHD Graphics 630 1536 MB.

In the first experiment, both catalog catalysts contain 4 agents. In the second experiment, both catalog catalysts contain thousand agents. In the third experiment, both catalog catalysts contain one million agents. The results of the three experiments are shown in table 1.

Table 1
Experiments data

Experiment	Agents quantity	Thread quantity	Memory usage	CPU usage
1	8	27	268MB	0.03-1%
2	1000	56	1.2GB	5-10%
3	1000000	107	4.0GB	32-67%s

The agent.kt platform uses coroutines to describe agents and directory facilitators – reliability of this component is higher than JADE or other analogues, because it uses nonblocking structured concurrency and does not use any reflection methods to synchronize message I/O functions and memory allocation. Even on two million agent the CPU usage was near the 34 percentage and max CPU usage was 65. Main disadvantage of this approach is that memory used by 2000000 million of coroutine is near 3.5 Gb. If number of agents increased, system would require more than allowed size of heap which can produce OutOfMemory exception. So the number of agents in agent platform is major parameter which has its influence on both agent-platform reliability and performance.

Given the technical limitations of the testing environment, it can be noted that the developed prototype meets the requirements and the test results can be considered satisfactory. Thus, it can be concluded that the given prototype with its performance characteristics can be used for modeling of interaction of agents that represent team members and tasks while solving project management problems.

6. Discussion and Conclusion

The given research represents a general architecture of e-business that is characterized by a distributed structure of customers and their orders and executives who can process orders. Both orders and contractors perform on the online platform and communicate via virtual channels only. The basic idea of this work is to resolve the problem of orders processing by contractors in real-time mode by means of assigning intelligent agents to each order and each contractor. This kind of problem is considered on the example of project management problem of tasks assignment in the distributed teams. Thus, a multi-agent system of order agents and executor agents has to solve the assignment problem. This allows to build pairs of orders and executives who will complete a particular job and satisfy a customer's demand. Agent communication protocols support a process of agents interaction and message exchange which is the basis of order-executor pairs formation.

The analysis of the existing approaches to designing multi-agent software systems and software components of the agent platform is carried out. A prototype of an agent platform based on the use of formal methods for designing multithreaded systems using the Kotlin programming language has been developed. According to the test results, the developed prototype has advantages over existing solutions in terms of CPU load and memory usage. It is concluded that the number of agents is the main parameter that affects the reliability and efficiency of the developed agent platform.

Approbation of the agent platform is carried out on the example of solving the problem of division of labor. The results of simulation experiments showed that the effect of the multi-agent approach is achieved through the implementation of communications between agents to find a pair "task - performer". Based on the built simulation model, an experiment was performed, during which the load on resources was calculated with different number of agents simultaneously acting in the system. Thus, the applicability of the agent platform for solving project management problems in a distributed development team was demonstrated. A future work in this research will be focused on modeling of tasks assignment among team members based on the agent architecture and conducting corresponding experiments.

7. References

[1] W. van der Aalst, Process Mining: Data Science in ActionAalst, Springer-Verlag, Berlin, 2016. doi: 10.1007/978-3-662-49851-4_1.

[2] I. Kononenko, S. Lutsenko, Application of the Project Management Methodology Formation's Method, Organizacija Journal of Management, Informatics and Human Resources 4 (2019) 286–308. doi: 10.2478/orga-2019-0018.

[3] A. Rasnacis, S. Berzisa, Method for Adaptation and Implementation of Agile Project Management Methodology, Procedia Computer Science 104 (2017) 43–50. doi: /10.1016/j.procs.2017.01.055.

[4] I. Budacu, E. Mihai, Using profiling to assemble an agile collaborative software development team made up of freelancers, Procedia Computer Science 162 (2019) 562–570. doi: 10.1016/j.procs.2019.12.024.
[5] D. Pimchangthong, V. Boonjing, Effects of Risk Management Practice on the Success of IT Project, in: Proceedings of the 7th International Conference on Engineering, Project, and Production Management Procedia Engineering 182, 2017, pp. 579- 586. doi: 10.1016/j.proeng.2017.03.158.
[6] D. Ciric, B. Lalic, D. Gracanina, N. Tasica, M. Delica, N. Medica, Agile vs. Traditional Approach in Project Management: Strategies, Challenges and Reasons to Introduce Agile, Proceedings of the 25th International Conference on Production Research Manufacturing Innovation, volume 39 of Cyber Physical Manufacturing, Chicago, Illinois, 2019, pp. 1407–1414. doi: 10.1016/j.promfg.2020.01.314.
[7] N. Ale Ebrahim, S. Ahmed, Z. Taha, Australian Virtual Teams: a Literature Review, J. of Basic and Applied Sciences 3(3) (2009) 2653–2669. doi: 10.6084/M9.FIGSHARE.103369.
[8] P. Weimann, H. E. S. Christian, M. Pollock, Changing the Communication Culture of Distributed Teams in a World Where Communication is Neither Perfect nor Complete, The Electronic Journal Information Systems Evaluation 13 (2) (2010): 187–196.
[9] Y. Lin, N. Gaud, V. Hilaire, P. Descamps, Multi-Agent System for intelligent Scrum project management, Integrated Computer-Aided Engineering 22 (2015) 281–296. doi: 10.3233/ICA-150491.
[10] G. K. Theodoropoulos, P. Lemarinier, Gr. M.P. O'Hare, Agent Based Modelling and Simulation tools: A review of the state-of-art software, Computer Science Review —S.Computer Science Review 24 (2017), 13–33. doi:10.1016/J.COSREV.2017.03.001.
[11] C. Orłowski, I. Bach-Dąbrowska, P. Kapłański, W.Wysocki, Hybrid Fuzzy-ontological Project Framework of a Team Work Simulation System, Procedia Computer Science 35 (2014) 1175–1184. doi: 10.1016/j.procs.2014.08.214.
[12] A. Dorri, R. Jurdak, S. Kanhere, Multi-Agent Systems: A survey, IEEE Access 4 (2018) 1–21. doi: 10.1109/ACCESS.2018.2831228.
[13] D. Milojicic, M. Breugst, I. Busse, et al, MASIF: The OMG mobile agent system interoperability facility, Personal Technologies 2 (1998) 117–129. doi: 10.1007/BF01324942.
[14] S. Poslad, P. Charlton, Standardizing agent interoperability: The FIPA approach. in: ECCAI Advanced Course on Artificial Intelligence. Springer, Berlin, 2001, pp. 98-117. doi: 10.1007/3-540-47745-4_5.
[15] G. Nguyen, T.T. Dang, L. Hluchy, M. Laclavik, Z. Balogh, I. Budinska, Agent platform evaluation and comparison, Institute of Informatics, Slovak Academy of Science, 2001, doi: 10.13140/RG.2.1.4648.9042.
[16] V. Pasichnyk, N. Kunanets, N. Veretennikova, A. Rzheuskyi, M. Nazaruk, Simulation of the Social Communication System in Projects of Smart Cities, in: Proceedings of the 14th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2019, 2019, pp. 94–98.
[17] R. Kaminskyi, N. Kunanets, V. Pasichnyk, A. Rzheuskyi, A. Khudyi, Recovery gaps in experimental data. CEUR Workshop Proceedings 2136 (2018) 108–118.
[18] V. Tomashevskyi, A. Yatsyshyn, V. Pasichnyk, N. Kunanets, A. Rzheuskyi, Data Warhouses of Hybrid Type: Features of Construction. Advances in Intelligent Systems and Computing book series 938 (2019) 325–334.
[19] M. Odrekhivskyy, V. Pasichnyk, A. Rzheuskyi, V. Andrunyk, M. Nazaruk, O. Kunanets, D. Tabachyshyn, Problems of the intelligent virtual learning environment development. CEUR Workshop Proceedings 2386 (2019) 359–369.
[20] H. Lypak, V. Lytvyn, O. Lozynska, R. Vovnyanka, Y. Bolyubash, A. Rzheuskyi, D. Dosyn, Formation of Efficient Pipeline Operation Procedures Based on Ontological Approach. Advances in Intelligent Systems and Computing 871 (2019) 571–581.
[21] R. Kaminskyi, N. Kunanets, A. Rzheuskyi, A. Khudyi, Methods of statistical research for information managers, in: Proceedings of the 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2018, 2018, pp. 127–131.