Methodological Support of Game Modeling in the Educational Process

Sergey Sherbakov [a], Maria Lapina [b], Vitalii Lapin [c] and Joze Rugelj [d]

[a] *Rostov State Economic University, Rostov, 344002, Russia*
[b] *North-Caucasus Federal University, Stavropol, 355017, Russia*
[c] *Stavropol Regional Clinical Consulting and Diagnostic Center, 355000, Stavropol, Russia*
[d] *University of Ljubljana, Ljubljana, 1000, Slovenia*

Abstract
The paper examines the use of the tournament approach in the educational process on the example of the economic model "repeated prisoner's dilemma". Students study the model design using the UML and implement the solution to the "Tournament of Strategies" problem using an object-oriented language. The advantages of the proposed themes are "junction" of the economy and important for programming in Computer Sciences Economic Institutions; the development of a diverse set of skills (economics, game theory, design, UML, modeling, object-oriented design and programming, Java, Eclipse); introduction of a creative, competitive moment into the educational process, as well as an element of intrigue.

Keywords [1]
Educational and methodological support, economics, prisoner's dilemma, programming, integration

1. Introduction

1.1. Relevance of the task

IT courses at an university of economics include the preparation of a "two-headed" graduate. To master with equal ease the methods and tools of informatics and programming on the one hand, and methods of economics and management on the other [10].

The economic components are provided by such disciplines as economic theory, mathematical and simulation modeling, etc. However, there is a certain gap between these disciplines and the profile disciplines of the educational program. Knowledge of economic science is in most cases not used to solve specific programming problems. Therefore, it is often the case that these two areas do not overlap in training.

Game modeling makes it possible to bridge this gap to a large extent. Designing and programming of a game model allows on the one hand to increase competence in the design and implementation of software on the other hand to immerse oneself in the subject area, i.e. economics and management. And experiments with a game simulation model give students the opportunity to analyze in detail the work of the economic mechanisms in detail. The study of the methodological support of game modeling will be carried out using a similar problem as an example.

In order to achieve the goal of integration of education in economics and IT, it was decided to implement the "Strategy Tournament" task. Its importance lieas in the repetition of the famous

SLET-2020: International Scientific Conference on Innovative Approaches to the Application of Digital Technologies in Education, November 12-13, 2020, Stavropol, Russia
EMAIL: sergwood@mail.ru (Sergey Sherbakov); mlapina@ncfu.ru (Maria Lapina); vitlx@yandex.ru (Vitalii Lapin); joze.rugelj@pef.uni-lj.si (Joze Rugelj).
ORCID: 0000-0002-9124-0178 (Sergey Sherbakov); 0000-0001-8117-9142 (Maria Lapina); 0000-0002-0611-7002 (Vitalii Lapin); 0000-0003-1056-0055 (Joze Rugelj).

experiment of R. Axelrod to evaluate the effectiveness of different strategies in "Prisoner's Dilemma" [30].

As the task is quite simple from a technical point of view, it allows you to cover a wide range of areas of economics and programming, ranging from different market models to problems from cooperation in nature and society [1, 8] to UML-modeling, of object-oriented software and programming in the Java language [3, 7]. Another feature of the task is to increase the level of student participation, as the lesson includes a game with a previously unknown outcome [33, 35].

1.2. Lesson planning

For the study of the proposed topic 4, 6 or 8 hours of study time can be allocated, depending on the available options. Table 1 shows an indicative lesson plan with an approximate estimate of time for each element.

Table 1
The sequence of mastering the material

№	Activity element	Minutes	Comments
1.	Economic model	20	conclusion about the advantage of refusing to cooperate with any choice of the opponent [15];
2.	Areas of use in economics and other areas	10	- advertising decision (advertise/not advertise [16]); - decision to cut prices; - cartel agreement and withdrawal from it (examples, OPEC [15]); - arms race; etc.
3.	Dynamic variant of the problem	5	- the importance of reputation [24]; - rejection of the strategy of betrayal [36]; - held tournaments [30];
4.	Dynamic task examples	5	examples from economics, business, politics, sociology, biology [39];
5.	Formulation of the problem of holding a strategy tournament	10	
6.	UML Modeling	20	- class diagrams, interactions; - "Strategy" (or "Player") as an abstract class [2];
7.	Designing strategies (discussing with students).	10	students offer their own strategies;
8.	Programming	40	if it is necessary to save classroom hours, students can program only players.
9.	Tournament run	30	
10.	The discussion of the results	10	

After the end of the lesson, students may be offered such directions for self-study, such as:
– formation of new strategies and analysis of their effectiveness;
– analysis of the results of the experiment depending on the number of players of different classes;
– an evolutionary version of the game [12], when the player's survival in each generation depends on the success of his game in the tournament;
– biological, economic, social examples of the implementation of the model "Prisoner's Dilemma" and "Repeating Prisoner's Dilemma" [21].

1.3. Evaluation tools for monitoring the effectiveness of the lesson

In order to control the assimilation of the teaching material, questions (closed form) were developed, divided into five categories: economic issues; elements of game theory; repetitive games; design and UML; object-oriented programming. Table 2 shows examples of developed questions. The test was implemented in LMS Moodle.

Table 2
Tests to control the mastery of the material (fragment)

Category	Question	Answer variant
economic issues	What type of market is the "Prisoner's Dilemma"	Monopoly Oligopoly Monopolistic competition Pure competition
elements of game theory	The Prisoner's Dilemma refers to	zero-sum games non-zero-sum games playing with nature
design and UML	The advantage of polymorphism	**the client class may not know anything about the derived classes. It works with their instances in the same way as with the parent class instance.** **It is possible to create new inheritance classes without modifying the client classes and the parent class** Inheritance is excluded It becomes possible to allocate on the stack instead of the heap (heap)
object oriented programming	What does this mean in a Java program?	**reference to the current object instance** constant recursive call
recurring games	Within the framework of the strategy tournament, "good" strategies (like an eye for an eye)	always lose the tournament "Angry" always lose individual matches to "evil" **benefit from cooperation with each other**

2. Development of teaching materials for the "tournament of strategies" task

2.1 Statement of the problem of the repetitive prisoner's dilemma

The well-known Prisoner's Dilemma is a popular model that illustrates various economic situations, for example, price competition in the oligopoly market. The “prisoner's dilemma” is also used in more complex cases, when two participants in economic relations make the decision to cooperate or refuse to cooperate.

The “prisoner's dilemma” can be formulated as follows: two accomplices were arrested for theft and placed in different cells. Each of them can confess to a crime or refuse to confess. The prisoner is not aware of the decision of his accomplice.

The punishment assigned to each of the prisoners, depending on the testimony, can be represented by the matrix shown in Table 3.

Table 3
Prisoner's Dilemma Matrix

		Prisoner A	
		Did not confess	Confessed
Prisoner B	Did not confess	A – 2 years B – 2 years	A – 1 year B – 10 years
	Confessed	A – 10 years B – 1 year	A – 5 years B – 5 years

Thus, the prisoner who has betrayed his comrade receives a bonus in the form of a small prison sentence. In case of loyalty among the prisoners, both receive rather soft conditions.

Depending on the condition of the task, it is beneficial for the prisoner to testify, thereby betraying his comrade. Suppose prisoner A assumes that prisoner B will break loyalty. In this case, it is also beneficial to violate loyalty by reducing his sentence from 10 to 5 years. If prisoner A assumes on the part of prisoner B the preservation of loyalty, then A is still beneficial to break loyalty by reducing his sentence from 2 years to 1.

A mutually beneficial solution can be achieved through mutual preservation of loyalty, but under the conditions described, such a possibility is difficult to reckon with.

The dynamic (repeating) version of the prisoner's dilemma (which corresponds more closely to economic realities), proposed by R. Axelrod, presupposes not a single decision-making about loyalty or betrayal, but a series of such decisions, continuing an unknown number of times. Both prisoners now remember the decisions of their comrade in the previous steps. Under such conditions, the benefit of betrayal in a particular step may be less than the benefit of cooperation in the following steps.

The question of the best behavioral strategy in a dynamic model is controversial.

The task is to organize a tournament of strategies (also proposed by R. Axelrod) to determine the best strategy. The following strategies participate in the tournament:

Random – random choice (the decision to betray / keep is chosen randomly, with equal probability).

Evil – renegade (betray at every step).

Talion – eye-for-eye (keep loyalty at the first step; repeat the previous move vis-a-vis at subsequent steps). This option was proposed by A. Rapport.

The game features three players implementing each strategy. The tournament is organized in a round robin system - each player plays one match with everyone else. The match consists of N steps. The players do not know the number N, while remaining the same for all matches of the tournament.

At each step, the distribution of points is set by the matrix in Table 4. The result of the tournament is determined by the total number of points scored by each player in all matches.

Table 4
Prisoner's Dilemma Matrix (digitized)

		Player A	
		Cooperate	Betray
Player B	Cooperate	A – 5 B – 5	A – 10 B – 0
	Betray	A – 0 B – 10	A – 1 B – 1

2.2. UML Modeling and Design

For programming a task, an object-oriented approach is most effective, allowing to separate the parts of the system that are responsible for decision making and for game play. At the same time, a flexible system can be created that allows new players with new strategies to be added.

For the visual representation of design decisions during their discussion, we use the UML language, which is now the de facto standard in the field of analysis, design and software development.

The language diagrams allow students to propose and discuss design solutions for the implementation of the tournament [4].

In this case, two diagrams are sufficient: a class diagram (Figure 1) describes the structure of the system, and a sequence diagram - the order of interaction of objects of different classes during the tournament. [11].

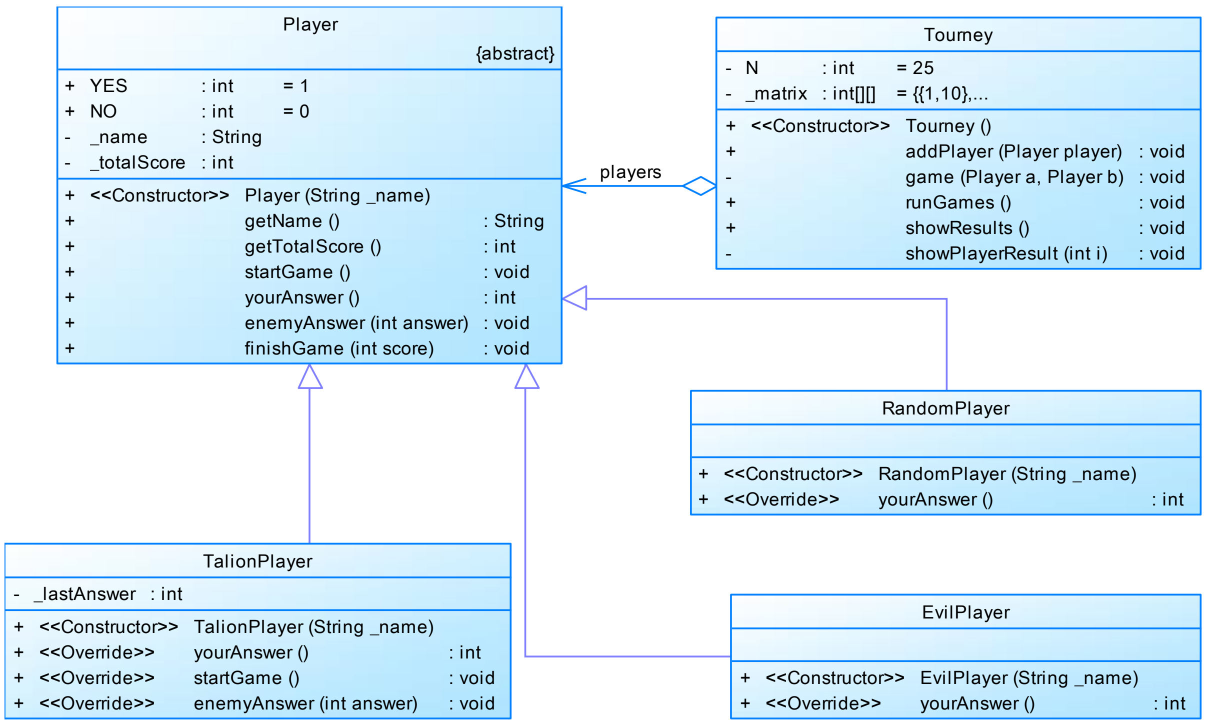
Player
{abstract}
+ YES : int = 1
+ NO : int = 0
- _name : String
- _totalScore : int
+ <<Constructor>> Player (String _name)
+ getName () : String
+ getTotalScore () : int
+ startGame () : void
+ yourAnswer () : int
+ enemyAnswer (int answer) : void
+ finishGame (int score) : void
players
Tourney
- N : int = 25
- _matrix : int[][] = {{1,10},...
+ <<Constructor>> Tourney ()
+ addPlayer (Player player) : void
- game (Player a, Player b) : void
+ runGames () : void
+ showResults () : void
- showPlayerResult (int i) : void
RandomPlayer
+ <<Constructor>> RandomPlayer (String _name)
+ <<Override>> yourAnswer () : int
TalionPlayer
- _lastAnswer : int
+ <<Constructor>> TalionPlayer (String _name)
+ <<Override>> yourAnswer () : int
+ <<Override>> startGame () : void
+ <<Override>> enemyAnswer (int answer) : void
EvilPlayer
+ <<Constructor>> EvilPlayer (String _name)
+ <<Override>> yourAnswer () : int

Figure 1: UML class diagram for the "Tournament of strategies" problem

The key role is played by the abstract class Player, which is responsible for the player (strategy). This class stores the player's name (_name), calculates and stores the player's total win since the start of the tournament (_totalScore). Class method yourAnswer () - the player returns the answer on the next turn. The answer is an integer YES = 1 (cooperate) or NO = 0 (betray).

The player classes RandomPlayer, EvilPlayer, TalionPlayer, as inheritors of the Player class, override the yourAnswer () method and other methods as appropriate. In this way, they each implement their own game strategy.

The Tourney class is fully responsible for running the tournament: creating the schedule, running the games, calculating the results. The Player class and its heirs know nothing about the tournament, and the Torney class depends only on the Player class, without knowing its specific heirs and without depending on their implementation of one strategy or another. When extending the task by adding a new player, it is sufficient to create a new class that inherits from Player [6].

The simple structure of the task and the very simple implementation allow to touch, if necessary, some aspects of object-oriented programming (inheritance, abstract classes, abstract methods, method overrides, composition, scopes, constructors, static class fields). Moreover, a simple example clearly demonstrates the usefulness of polymorphism [20].

2.3. Implementation in Java and Eclipse

To program the task, one can use, for example, the Java programming language [23] and one of the modern development tools, such as the free Eclipse environment [10].

Part of the program code can be created by students individually, part - together with the teacher as part of the work with a projector. In the process, the possibilities of the Eclipse environment can be explored, such as: automatic error detection and suggestions for automatic fixing; automatic creation of inherited classes with overriding methods; automatic refactoring [16]. The code of the Player class is shown in Listing 1

Listing 1: The program code of the abstract class Player

```
public abstract class Player {
	public static final int YES=1; //сотрудничать
	public static final int NO=0; //предать
	private String  _name;
	private int _totalScore;

	public Player(String _name) {
		super();
		this._name = _name;
		this._totalScore=0;
	}

	public String getName() {
		return _name;
	}
	public int getTotalScore() {
		return _totalScore;
	}
	public void startGame(){}
	public abstract int yourAnswer();
	public void enemyAnswer(int answer){}
	public void finishGame(int score){
		_totalScore+=score;
	}
}
```

Listing 2 shows the code for the rogue player class. The yourAnswer () method always returns a NO response. This is the only change that was required for the class, all the rest of the code is automatically generated [27].

Listing 2: EvilPlayer class program code

```
public class EvilPlayer extends Player {

	public EvilPlayer(String _name) {
		super(_name);
	}

	@Override
	public int yourAnswer() {
		return NO;
	}
}
```

Somewhat more complex is the code for a player who implements the eye-for-eye strategy (Listing 3).

Listing 3: TalionPlayer class code

```
public class TalionPlayer extends Player {
      private int _lastAnswer;

      public TalionPlayer(String _name) {
            super(_name);
      }

      @Override
      public int yourAnswer() {
            return _lastAnswer;
      }

      @Override
      public void startGame() {
            _lastAnswer=YES;
            super.startGame();
      }

      @Override
      public void enemyAnswer(int answer) {
            _lastAnswer=answer;
      }
}
```

The private field _lastAnswer stores the last response of the adversary. It is written there by the overridden method enemyAnswer (). In the next step, the method yourAnswer () simply repeats the opponent's previous move.

2.4 Tournament holding

Running the model yields the expected results-an overall win for the tit-for-tat strategy, despite this player's losses in individual matches.

The matrix of results by match and the graph of results and are shown in Table 5 and Figure 2, respectively.

Table 5
Tournament table

	Random 1	Evil 1	Talion 1	Random 2	Evil 2	Talion 2	Random 3	Evil 3	Talion 3
Random 1	**x**	13:133	90:80	99:119	16:106	106:96	101:101	12:142	122:112
Evil 1		**x**	34:24	106:16	25:25	34:24	115:15	25:25	34:24
Talion 1			**x**	97:97	24:34	125:125	85:85	24:34	125:125
Random 2				**x**	10:160	97:97	104:74	16:106	85:85
Evil 2					**x**	34:24	160:10	25:25	34:24
Talion 2						**x**	97:97	24:34	125:125
Random 3							**x**	13:133	110:100
Evil 3								**x**	34:24
Talion 3									**x**

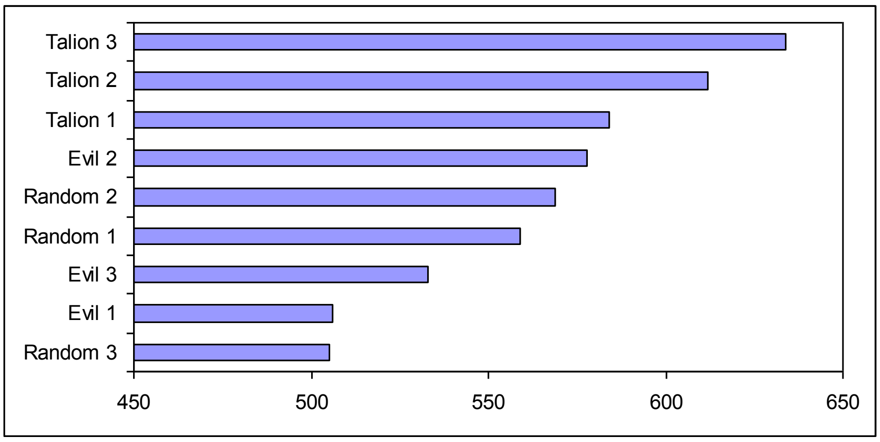

Figure 2: Example of tournament results

Further, the teacher may discuss the results obtained. Explain the reasons for the advantages of cooperation under certain conditions. Think about further work on the model.

Of interest is the expansion of the number of strategies [14], e.g. a strategy that does not forgive a partner even after a betrayal; a strategy that punishes a partner for betrayal over two rounds; strategy of permanent cooperation, etc. Another area is a different relationship of players with certain strategies.

Finally, it is possible to implement the evolutionary principle - the tournament is repeated several times (generations), each strategy is considered an "individual" and the number of these strategies in each next generation depends on the results of the previous step. [12].

3. Evaluation of the effectiveness of using the "Tournament of Strategies" task

To assess the effectiveness of mastery of the material as a result of the lesson, students were tested on a pre-developed set of tests.

Table 6 shows the results of the tests for one group of students. The tests were administered during the next lesson one week after the completion of the "Strategy Tournament".

Table 6
Tests results

student	economic issues	elements of game theory	Recurring games	UML	OOP	Total
1	75%	100%	100%	75%	100%	88%
2	50%	67%	67%	100%	100%	76%
3	100%	100%	100%	75%	100%	94%
4	50%	100%	67%	100%	100%	82%
5	25%	33%	0%	75%	67%	41%
6	100%	100%	67%	100%	100%	94%
7	75%	100%	67%	75%	67%	76%
8	100%	100%	100%	100%	67%	94%
9	100%	100%	100%	100%	100%	100%
10	100%	100%	100%	100%	100%	100%
11	50%	100%	100%	100%	100%	88%
12	100%	67%	67%	50%	100%	76%

student	economic issues	elements of game theory	Recurring games	UML	OOP	Total
13	100%	67%	100%	75%	100%	88%
14	75%	67%	67%	75%	100%	76%
15	100%	100%	100%	100%	100%	100%
16	100%	100%	33%	100%	100%	88%
17	75%	100%	100%	100%	100%	94%
18	50%	100%	67%	100%	67%	76%
total	**79%**	**89%**	**78%**	**89%**	**93%**	**85%**

The results of the control group (which examined similar sections using traditional methods) are shown in Table 7.

To assess the significance of the differences in the two groups, we use the statistical test Mann-Whitney. This is necessary because they are independent samples and the sample size requires the use of non-parametric methods.

Table 7
Test results (control group)

student	Economic questions	elements of game theory	Recurring games	UML	OOP	Total
1	50%	33%	33%	75%	67%	53%
2	75%	67%	33%	75%	100%	71%
3	50%	67%	67%	50%	67%	59%
4	100%	100%	100%	100%	100%	100%
5	25%	33%	33%	100%	100%	59%
6	0%	33%	0%	75%	100%	41%
7	25%	33%	33%	50%	67%	41%
8	25%	33%	0%	50%	100%	41%
9	50%	0%	0%	75%	100%	47%
10	50%	67%	67%	75%	100%	71%
11	75%	67%	33%	50%	67%	59%
12	75%	100%	0%	0%	100%	53%
13	50%	67%	67%	25%	67%	53%
14	75%	67%	0%	50%	67%	53%
15	50%	67%	0%	25%	67%	41%
Total	**52%**	**56%**	**31%**	**58%**	**84%**	**56%**

The assessment is carried out according to the formula:

$$u_{emp} = n_1 n_2 + \frac{n_x(n_x+1)}{2} - T_x$$

where Tx is the largest sum of ranks, nx is the largest of the sample sizes n1 and n2.

Calculation results:

$$u_{emp} = 18 \cdot 18 + \frac{18(18+1)}{2} - 463 = 32$$

The hypothesis H0 about the insignificance of the differences between the samples is accepted if Ucr < Uemp. Otherwise, H0 is rejected and the difference is identified as significant, where Ukp is the critical point, which is found using the Mann-Whitney table.

From the table we find Ukp (0.05) = 99. Since Ukp > Uemp, we reject the null hypothesis.

Figure 6 shows an illustration of the results obtained.

As can be seen from the tables and the figure, as a result of introducing an innovative teaching method, there is a significant increase in the uptake of material on "economic" topics, but also some increase in topics specific to computer science and programming [37].

This suggests that I program to understand learning is quite effective [13].

At the same time, it is necessary to pay attention to the cost of introducing the described approach into the educational process, which is associated with the need for deep study of the material for the implementation of the game "balance". At the same time, the pedagogical and methodological documentation of the university in the traditional version does not imply a reflection of this level of methodological materials, which means that the corresponding costs remain hidden. The study of the economic side of the pedagogical and methodological involvement of game methods and project-based learning must be further developed.

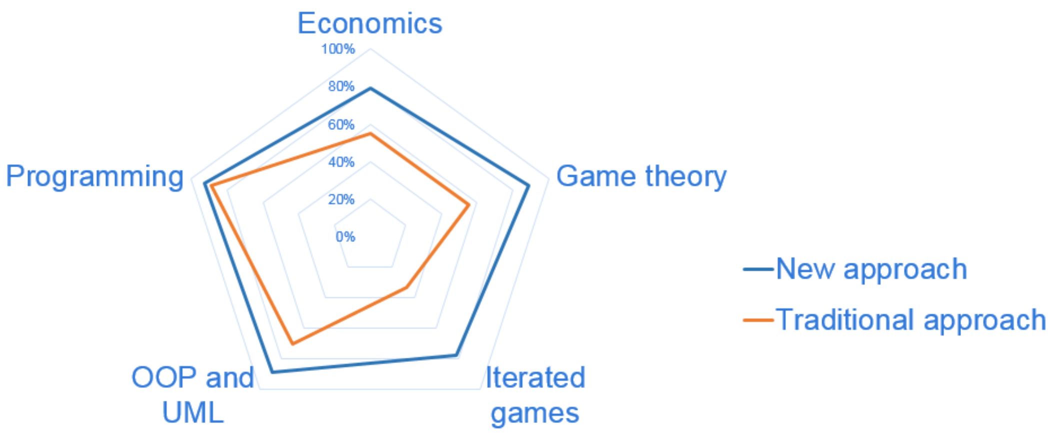

Figure 6: The effectiveness of training by the tournament method

4. Conclusion

In this paper, the methodological support for the construction of a game simulation model was analyzed using the example of the "Tournament of strategies" problem.
It was experimentally shown that the use of such models can improve both programming skills and understanding of the subject area (economics and management).

5. Acknowledgements

The research was carried out with the financial support of the Russian Foundation for Basic Research (RFBR) within the framework of the scientific project 19-013-00690 "Economics of educational and methodological activities in higher education".

6. References

[1] Weissfeld M. Object-oriented thinking / M. Weisfeld. - M.: Peter, 2014. - 998 p.
[2] Vasiliev, A. Java. Object Oriented Programming: A Tutorial. - SPb.: Peter, 2013. - 400 p.
[3] Varian Hal R. Microeconomics. Intermediate level. Modern approach. - Textbook for universities. - M.: UNITI, 1997. - 767 p.
[4] Golovanov, NF Pedagogy: textbook and workshop for academic bachelor's degree / NF Golovanov. - 2nd ed., Rev. and add. - Moscow: Yurayt Publishing House, 2019. - 377 p.
[5] Gromkova, M.T. Higher School Pedagogy: Textbook. - M.: Unity, 2017 .-- 80 p.

[6] Graham Ian. Object oriented methods. Principles and Practice = Object-Oriented Methods: Principles & Practice. - 3rd ed. - M.: "Williams", 2004. p. 880.
[7] Dawkins R. Selfish gene. - Moscow: AST: CORPUS, 2013. 512 p.
[8] Kuboniva M., Tabata M., Tabata S., Hasebe Yu. Mathematical economics on a personal computer. Per. with jap. / Edited by M. Kuboniva. - M.: Finance and statistics, 1991. - 301 p.
[9] Lychkina, N.N. Simulation modeling of economic processes: Textbook. - M.: INFRA-M, 2012. - 254 p.
[10] McConnell Campell R., Bru Stanley L., Flynn Sean M. Economics: principles, problems and politics: a textbook. - 19th ed. - M.: INFRA-M, 2013. - 1027 p.
[11] Mankiw N. Gregory Principles of Economics. - SPb.: Peter, 2007. - 624 p.
[12] Nartova Anna PowerDesigner 15. Data Modeling. - M.: Lori, 2012. - 418 p.
[13] General foundations of pedagogy: A textbook for students of pedagogical universities / IA Solovtsova, NM Borytko; Ed. N.M. Borytko. - Volgograd: Publishing house of VGIPK RO, 2006. - 60 p.
[14] Pedagogy: textbook / ed. P.I. Perky. - 5th ed., Add. and revised - Moscow: Pedagogical Society of Russia, 2008. - 580 p.
[15] Rambeau J., Blah M. UML 2.0. Object Oriented Modeling and Development. - SPb.: Peter, 2007. - 544 p.
[16] Ridley Matt The Origin of Altruism and Virtue. From instincts to cooperation. - M.: Eksmo, 2013. - 701 p.
[17] Sybase Power Designer modeling system [Electronic resource]. - dir. available: www.sybase.ru/
[18] Sierra K. Learning Java. Head First series. - M.: Eksmo, 2017. - 728 p.
[19] Topazh A.G. Agent models of evolutionary games // The Ninth All-Russian Scientific and Practical Conference on Simulation Modeling and Its Application in Science and Industry “Simulation Modeling. Theory and Practice "(IMMOD-2019). Conference proceedings, October 16-18, 2019, Yekaterinburg: Ural. state ped. un-t., 2019.- 678 p. - ISBN 978-5-91450-172-0. S. 227-234.
[20] Fowler M. UML. The basics. A quick guide to the standard object modeling language. - SPb.: Symbol-plus, 2011. - 192 p.
[21] Shannon R. Systems Simulation - Art and Science. - M.: Mir, 1978. - 420 p.
[22] Schildt Herbert Java. The Complete Guide, 8th Edition. –M.: Vilyams, 2012. - 1014 p.
[23] Eckel, Bruce The Java Philosophy / Bruce Eckel. - M.: Peter, 2016. - 809 p.
[24] Araujo R.F., Durelli V.H.S., Teixeira R.M. Getting Started with Eclipse Juno: A fast paced tutorial to get you up and running with Eclipse Ju-no IDE. - Packt Publishing, 2013. - 256 p.
[25] Axelrod, Robert Evolution of Cooperation: Revised Edition. -New York: Basic Books, 2009. - 265 p.
[26] Heineke, J. & Meile, L. Classroom service games. Presentation at the Decision Sciences Institute Annual Meeting, Nov. 18.
[27] http://simulation.su/ – Society for Simulation
[28] Klassen Kenneth J., Willoughby Keith A. In-Class Simulation Games: Assessing Student Learning // Journal of Information Technology Education, 2003, Volume 2, p. 1-13.
[29] Markov A. V., Markov M. A. Runaway brain-culture coevolution as a reason for larger brains: Exploring the «cultural drive» hypothesis by computer modeling // Ecology and Evolution. 2020.
[30] Shellman, S. M., Turan, K. Do Simulations Enhance Student Learning? An Empirical Evaluation of an IR Simulation // Journal of Political Science Education, 2006, 2, p. 19-32.
[31] Sigmund, Karl & Fehr, Ernst & Nowak, Martin. (2002). The Economics of Fair Play. Scientific American. 286. 82-7.
[32] Silvia Chris The impact of simulations on higher-level learning // Journal of Public Affairs Education, 2012, 18(2), p. 397-422.
[33] stepik.org
[34] Super Cooperators: Altruism, Evolution, and Why We Need Each Other to Succeed. Martin A. Nowak, with Roger Highfield. Free Press, 2012.
[35] www.eclipse.org