Genetic algorithm of optimizing the size, staff and number of professional teams of programmers

Anatoly Prihozhy, Arseni Zhdanouski
Belarusian National Technical University
Minsk, Republic of Belarus
prihozhy@yahoo.com

***Abstract*—This paper considers the problem of establishing and optimizing the teams of programmers taking into account their proficiency and the level of skills in programming technologies and tools. It proposes a method of formalizing and evaluating the qualification of individual programmers and entire teams of programmers, and proposes a genetic algorithm of optimizing the size, staff and number of the teams. Experimental results on a set of programmers graduated from universities of Belarus show the ability of the system to find the teams of programmers, which increase the overall qualification of the teams by 30%. The obtained results prove the practical importance of the model, genetic algorithm and software in the field of technologies and tools for the management of professional teams of programmers.**

***Keywords*—programmer; technology; tool; qualification; team of programmers; team size; team staff; optimization**

I. Introduction

Agile technology [1] of flexible software development provides requirements and finds solutions due to the joint efforts of development teams and customers. It supports adaptive planning, evolutionary development, continuous improvement, and rapid-flexible response to changes. Although many technological environments use Agile, it requires further development for distributed programming teams. Tools and processes are important, but it is more important to have competent people working together effectively. Improper distribution of work can eliminate the connection of leading experts. Assigning a job to a team that is hard to find an appropriate expert increases the total cost of recruiting and performing work, making it difficult to select the right people for the distributed team.

The success of a distributed team of programmers in the implementation of a large project strongly depends on the adequacy of the technologies and programming tools used, as well as on the ability of effectively decomposing the project into parts. These parts should be distributed among those teams of programmers who have the necessary knowledge and skills to perform them. Work [2] formulates a problem of optimal partitioning of the given set of programmers into teams. It is a multifactorial and poorly structured problem. In [2], the following factors were taken into account: the productivity of each programmer, the ability of pairs of programmers to increase or decrease productivity in the process of collaboration, the increase in interfaces' cost between programmers while increasing the number of programmers in one team. Evolutionary optimization methods [3] [4] [5] are capable of solving this kind of problems. Work [5] develops a genetic algorithm for solving this problem. However, it does not take into account the level of programmers' knowledge of the technologies and programming tools that are necessary for the project. Work [6] proposes a model of evaluating proficiency of programmers.

This article analyzes modern technologies and programming tools and evaluates the level of proficiency of each of the programmers as well as the level of proficiency of entire team with respect to these technologies and tools. The size and the staff of each of the teams is optimized.

This paper is organized as follows. Section II presents a method of evaluating the proficiency of programmers and whole teams of programmers in technologies and tools. Section III describes a genetic algorithm of optimizing the size, staff and number of teams. Section IV presents experimental results, and last section concludes the paper.

II. Proficiency of a programmer and a team of programmers in technologies and tools

A. *Rating of programming technologies and tools*

Analysis of the research results RedMonk company [7] has provided on the popularity of programming languages and tools, as well as the research results the IEEE Spectrum organization [8] has provided on rankings of programming languages allows the evaluation of rating of various programming technologies. The technology rating indicates its importance and breadth, as well as indicates the requirements and constraints at least one member of a team of programmers must meet.

B. *Proficiency of a programmer in technologies and tools*

We use a survey method to solve the problem of assessing the level of programmers' knowledge of technologies and tools. Each of the programmers interviewed is asked to fill out a questionnaire in which he/she indicates the level of proficiency in each of the technologies. The proficiency level is determined by a five-point scale: 0 - lack of knowledge of technology; 0.25 - the minimum knowledge; 0.5 - intermediate skills; 0.75 - extended technology proficiency experience; 1 - expert knowledge and experience. We consider a programmer as expert, if he/she possesses theoretical expert knowledge, has developed at least two large projects and has worked for at least two years with this technology. The programmer has advanced skills if he/she meets two criteria of the three ones, and has intermediate skills if meets only one of the criteria.

C. Programmer qualification

The qualification of programmer $p \in P$ with regard to the level of knowledge / possession of technologies of set T in relation to the maximum level of knowledge / possession can be evaluated with Equation (1).

$$Qualif(p) = \frac{\sum_{t \in T} rank(t) \times factor(p,t)}{MaxQualif} \tag{1}$$

where

$$factor(p,t) = \begin{cases} level(p,t) & \text{if } level(p,t) \leq RL_t^p \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and

$$MaxQualif = \sum_{t \in T} rank(t) \tag{3}$$

The value of *rank*(*t*) belonging to range [0,1] is the rank of the technology. The value of *level*(*p,t*) of the range [0,1] is determined by the results of the survey of the programmers. The threshold value RL_t^p of *level*(*p,t*) reduces the chances of admission to work on the project for programmers with an insufficiently high level of qualification. This value is selected from the range [0, 1]. If $RL_t^p \cong 0$ then start-up programmers or programmers who possess of only small part of the technologies may participate in the project. If $RL_t^p \cong 1$ then only experts may participate in the project. As a result, the value of *Qualif*(*p*) is in the range [0,1]. According to Equation (1), programmers who possess of preferably high-rated technologies are more highly qualified than programmers who preferably possess of low-rated technologies.

D. Qualification of a team of programmers

Let the whole set P of programmers be divided into k teams, producing set $G = \{g_1, ..., g_k\}$. Programmers of team g constitute set P_g. The average qualification of team g, including N_g programmers, is defined as an average value *Qualif*(*p*) of qualifications over all programmers $p \in P_g$:

$$Qualif^{avg}(g) = \frac{\sum_{p \in P_g} Qualif(p)}{N_g} \tag{4}$$

In addition to the average qualification $Qualif^{avg}(g)$, the most important parameter that characterizes team g is the best representative qualification $Qualif^{best}(g)$, which is determined with Equations (5) and (6):

$$Qualif^{best}(g) = \begin{cases} 0, \text{if } \exists t, oblgt(t), mxl(g,t) < RL_t^g \\ \dfrac{\sum_{t \in T} rank(t) \times mxl(g,t)}{MaxQualif}, \text{otherwise} \end{cases} \tag{5}$$

$$mxl(g,t) = \max_{p \in P_g} level(p,t) \tag{6}$$

Equation (5) uses the values as follows:

- $oblgt(t)$ is a predicate that takes value $true$ if technology t is mandatory for a team (see Table 1);
- $mxlevel(g,t)$ is the level of the best representative of team g in technology t;
- $RLgt$ is a threshold value of the level of the best representative of team g in technology t.

According to Equation (5), qualification $Qualif^{best}(g)$ over the best representatives is equal to zero if there is at least one mandatory technology t for the team, for which $oblgt(t)$ is true, and the level $mxl(g,t)$ of qualification over the best representatives of team g is less than the threshold value of RL_t^g. The explanation is this team is not capable of carrying out projects without highly qualified specialists in key technologies.

The weighted qualification $Qualif^w(g)$ of team g is estimated with Equation (7) as the sum of qualification $Qualif^{best}(g)$ over the best representatives with weight λ, and the average qualification $Qualifavg(g)$ with weight $1 - \lambda$.

$$Qualif^w(g) = \lambda \times Qualif^{best}(g) + (1-\lambda) \times Qualif^{avg}(g) \tag{7}$$

The weighted qualification at $0 \leq \lambda \leq 1$ can take any value in the range [0,1]. The larger the value of λ, the larger the weight of the qualification over the best representatives, and vice verso, the lower the value of λ, the larger the weight of the average qualification of the programmers in the team. The average qualification reflects the skills of programmers. The qualification over the best representatives show opportunities for the growth of skills of the team members who are guided and trained by technology experts, who are members of the team and are exemplary.

It is obvious, the teams with low qualifications cannot be recognized as workable, and the participation of such teams in the project is unreasonable or at least controversial. We can formalize the exclusion of the appearance of such teams with the concept of weighted threshold qualification:

$$Qualif(g) = \begin{cases} Qualif^w(g) & \text{if} Qualif^w(g) \geq RQ^g \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

It is reasonable to recommend to choose the threshold value RQ^g of qualification from the range from 0.5 to 1.0, depending on the requirements of the project and the technologies that are used for its development, as well as depending on the expected quality of future design results. Bellow, the qualification of a team of programmers will be understood as the threshold weighted qualification.

Team g is called redundant by qualification if there is a programmer $p \in g$ such that $Qualif(g \backslash p) \geq Qualif(g)$.

In other words, the qualification of team g after removing programmer p from it should not be lower than the qualification before removing of the programmer. This can happen when at least one of two following conditions are met:

- programmer p is not the only best representative of team g on any technology $t \in T$;
- qualification $Qualif(p)$ of programmer p is lower than the average qualification $Qualif^{avg}(g)$ over team g.

If set G divides the set of programmers into teams, then we can estimate the overall qualification of the teams as a sum of the teams' threshold weighted qualifications:

$$Qualification(G) = \sum_{g \in G} Qualif(g) \tag{9}$$

The overall qualification $Qualification(G)$ takes values in the range from $k \times RQ^g$ to k, where k is the number of teams.

III. Optimizing the size, staff and number of teams

A. Problem formulation

Let Ω be a set of feasible solutions, which is a set of various partitions of set P of programmers into various set G of teams. The main parameter of solution G is the overall qualification $Qualification(G)$ of all teams. Therefore, we formulate the objective function as:

$$\max_{G \in \Omega} Qualification(G) \tag{10}$$

We formulate the set of feasible solutions over a system of constraints and describe them in the form of requirements for programmers and teams of programmers to be proficient in technologies and programming tools.

$Requirement1$. It refers to the minimum threshold qualification level RL_t^p of skills the programmer p of set P has regarding technology t. The minimum qualification level requirement for all technologies is represented with vector $RL^p = RL_1^p, ..., RL_m^p$. Equation (11) determines the actual level of L_t^p.

$$L_t^p = rank(t) \times level(p, t) \tag{11}$$

$Requirement2$. It refers to the minimum threshold qualification level RL_t^g the best representative of team $g \in G$ of programmers has regarding technology t. For all technologies, the minimum threshold level is represented with vector $RL^g = \{RL_1^g, ..., RL_m^g\}$. We determine the actual qualification level L_t^g of the best representative of team g for technology t with equation as follows:

$$L_t^g = rank(t) \times \max_{p \in P_g} level(p, t) \tag{12}$$

$Requirement3$. It refers to the minimum-threshold weighted qualification RQ^g of each team $g \in G$ of programmers. The demanded minimum qualification is the same for all teams. The actual weighted qualification $Qualif^w(g)$ of a team is estimated by Equation (7).

Set G typically includes a special team of unemployed programmers, who are included in a reserve team. The number N_empl of programmers included in a working team can be computed with Equation (13).

$$N_{empl} = \sum_{g \in G} N_g \tag{13}$$

The number N_res of programmers who are unemployed and included in the reserve team can be computed with Equation (14).

$$N_{res} = |P| - N_{empl} \tag{14}$$

The average weighted threshold qualification $Qualif^avg$ over all teams can be evaluated with Equation (15).

$$Qualif^{avg} = \frac{\sum_{g \in G} Qualif(g)}{|G|} \tag{15}$$

B. Genetic algorithm for solving the optimization problem

The genetic algorithm (GA) implements a random process of evolution of a population of chromosomes in order to find the best partitioning of the set of programmers into developers teams. We build the chromosome as a vector of genes that correspond to the programmers. The gene value is the team number that includes the programmer. The fitness function is the overall qualification $Qualification(G)$ of the programmers of teams G.

The genetic operation of crossing two chromosomes recombines their gen-parts and moves programmers from one team to other team in two resulting offsprings. This crossover operation can yield an offspring that does not refer to a team of programmers, thus reducing the number of teams. Such a situation may require re-enumerating the teams and introducing facilities, which can extend the set of teams.

The genetic mutation operation randomly chooses one or more programmers and transfers them to other teams. The selection operation selects parents according to the rule of roulette wheel in order to perform crossing and mutation operations and to select chromosomes for producing the next generation of chromosomes and for updating the population. The chromosome with the highest value of the fitness function is a solution of the optimization problem.

IV. Experimental results

A. Rating of programming technologies and tools

Table I describes a set T of 16 basic technologies and tools, and reports the rating of each of them [7, 8]. By appointment, the whole set of technologies is divided into 6 subsets. Version control and project management systems include Git, Tortoise SVN, VJR and TFS with rating 0.3 each. The development environments include Visual Studio and Eclipse, the rating of both is 0.6. Oracle SQL (rating 0.5) and Microsoft SQL Server (rating 0.6) represent database management systems. Programming languages are Java, C#, Visual Basic, C++, Java Script and XSL with rating of 1.0, 0.9, 0.7, 0.9, 0.8 and 0.6 respectively. Windows and Linux represent operating systems with rating of 0.6 and 0.5 respectively. The high rating of a technology shows the groundlessness of the creation of teams of programmers, in which there is no any expert on this technology.

Table I
KEY TECHNOLOGIES AND PROGRAMMING TOOLS

No	Name	Code	Rating	Oblgt
1	Git	VGT	0.3	no
2	Tortoise SVN	VTS	0.3	no
3	TFS	VTF	0.3	no
4	Jira	VJR	0.3	no
5	Visual Studio	DVS	0.6	yes
6	Eclipse	DEC	0.6	yes
7	Oracle SQL	OBM	0.5	no
8	Microsoft SQL Server	DBM	0.6	no
9	Java	LJ	1.0	yes
10	C#	LC#	0.9	yes
11	Visual Basic	LVB	0.7	no
12	C++	LCP	0.9	yes
13	Java script	LJS	0.8	yes
14	XSL	LXS	0.6	no
15	Windows	OSW	0.6	yes
16	Linux	OSL	0.5	yes

B. *Proficiency of programmers in technologies and tools*

Fig.1 shows results of a survey of 24 programmers (set P) with higher education, who graduated from universities of Belarus and work at programming companies. The rows correspond to the programmers, and the columns correspond to the technologies of Table I. At the intersection of row p and column t, the rectangle's height indicates one of the five levels 0, 0.25, 0.5, 0.75 and 1 of possession by programmer p of technology t. The absence of a rectangle means zero level. Rows with a large total area of rectangles indicate highly qualified programmers. Columns with a large area of rectangles indicate highly demanded and widely used technologies.

Figure 2 shows the level of importance of each of the 16 technologies and programming tools for 24 programmers in relation to the highest possible level without taking into account the technology rating. The Windows operating system (code OSW) has the highest level of 0.72. In second place is the version control system / project management system TFS (code VTF) with the level of 0.63.

Figure 3 shows the level of possession of technologies and programming tools by 24 programmers, taking into account the technology rating (Table 1). The Java programming language (code LJ) has the highest level of 0.98. The Windows operating system (code OSW) with the level of 0.73 has moved to the second place.

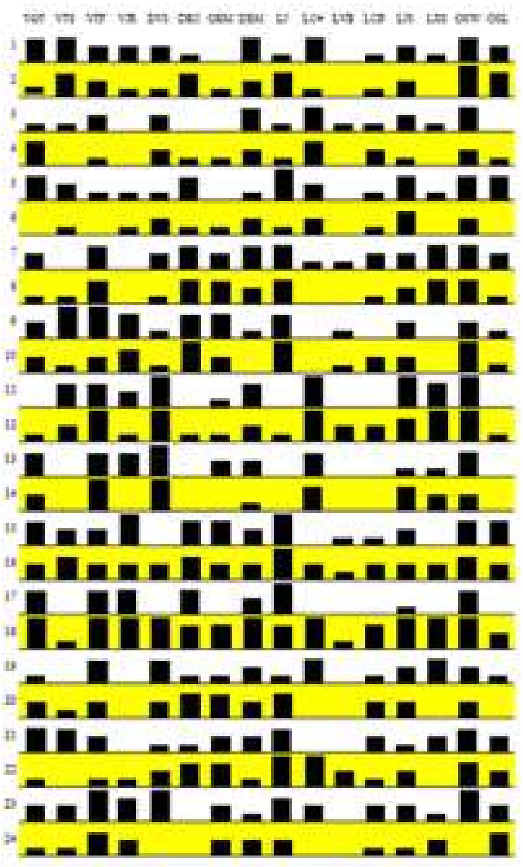

Figure 1. Proficiency of 24 programmers in 16 technologies

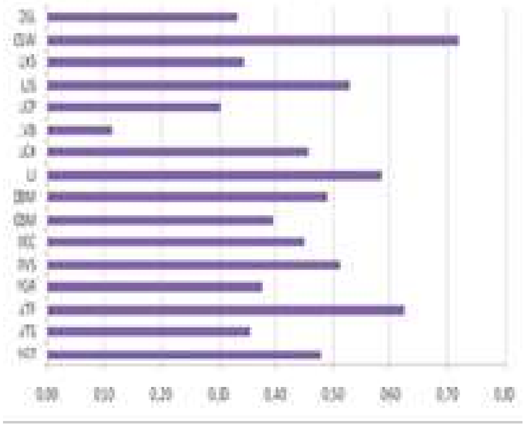

Figure 2. The level of possession of 16 technologies by 24 programmers without taking into account the rating of technologies.

Fig. 4 shows the qualification of each of the 24 programmers in terms of possession of all 16 programming technologies without taking into account the technology rating (in this case the rank (t) value of 1 for all t and the MaxQualif value of 16 is taken). Figure 5 shows the qualification of programmers on all programming technologies, taking into account the rating of technologies (in this case, the value of rank (t) is taken from Table 1 and the MaxQualif value is 9.5). Note that the consideration of technology rating (Fig. 5) led to the fact that programmer 8 became more qualified than programmer 9 (in Fig.4, programmer 9 is more qualified than 8). The same concerns programmers 22 and 23.

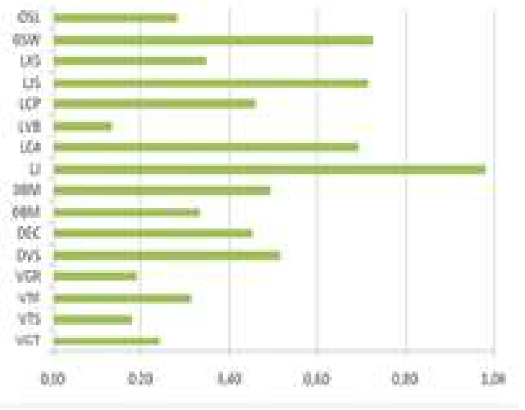

Figure 3. The level of possesion of 16 technologies by 24 programmers taking into account the rating of technologies.

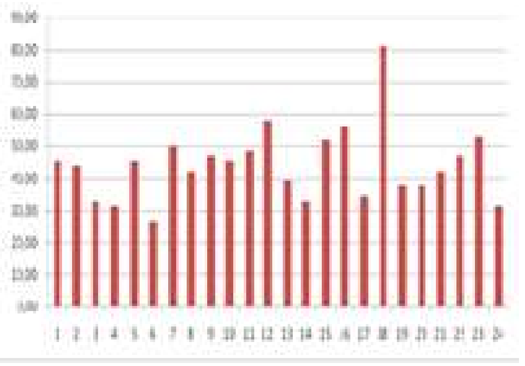

Figure 4. Qualifications of 24 programmers in all technologies without taking into account the technology rating.

C. Teams optimization constraints

We have developed a computer program that implements the proposed genetic algorithm, and have used this program for carrying out computational experiments on optimizing the distribution of programmers on a set of teams. The requirements and constraints are as follows:

- the one-programmer qualification on each of the 16 technologies that are listed in table I must not be lower than RL^p = 0.1, 0, 0.1, 0, 0.3, 0.25, 0, 0.15, 0.6, 0, 0, 0.25, 0, 0.2, 0.4, 0.2;
- the one-team-of-programmers qualification on each of the 16 technologies must not be lower than RL^g = 0.2, 0, 0.2, 0, 0.4, 0.3, 0, 0.3, 0.75, 0.4, 0, 0.5, 0, 0.3, 0.5.0.25;
- the threshold weighted qualification of one team of programmers on all technologies must not be lower than RQ^g taking value from the range 0.40 ... 0.75;
- the weight λ of one team qualification over the best representatives is equal to 0.7, and the weight $1 - \lambda$ of one team average qualification is equal to 0.3.

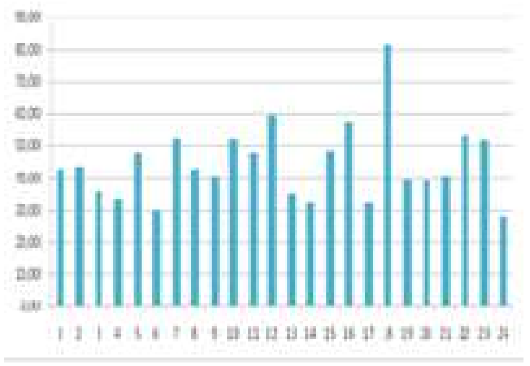

Figure 5. Qualifications of 24 programmers in all technologies taking into account the technology rating.

D. Experimental results

Table II reports experimental results that are obtained with the genetic algorithm while optimizing the size, staff and number of teams composed of 24 programmers (Fig.1), using the 16 key programming technologies and tools (Table I). Previous Section describes the requirements and constraints on teams establishing, which are associated with a project the teams will work on. Varying the value of RQ^g in the range from 0.40 to 0.75 with step 0.05, GA has given 8 solutions listed in Table II. The increase in the proficiency level RQ^g of one team decreases monotonically the number $|G|$ of teams from 9 down to 2. Moreover, it decreases the number N_empl of programmers involved in these teams. The qualification $Qualif^avg$ of one team grows from 0.602 to 0.785, although the overall $Qualification(G)$ of all teams falls rapidly from 5.42 down to 1.57 due to many programmers appear not involved in the project and are included in the reserve team. The staff of all teams is given in Table III.

Figure 6 shows the dynamics the genetic algorithm yields in the process of evolution of the chromosome population. The total weighted threshold qualification for all teams has grown (Fig. 6) from 3.5 to 5.04 over 25 generations. Growth is about 30%. During this time, the number of teams that meet all qualification requirements has increased from 5 to 8, while the number of programmers included in these teams has increased from 15 to 23 . The number of teams that do not meet

Table II
EXPERIMENTAL RESULTS ON OPTIMIZATION OF PROGRAMMING TEAMS WITH GENETIC ALGORITHM

Run	RQ^g	$\|G\|$	N_empl	$Qualif^avg$	$Qualification(G)$
1	0.40	9	22	0.602	5.42
2	0.45	8	23	0.631	5.05
3	0.50	8	23	0.630	5.04
4	0.55	8	21	0.626	5.01
5	0.60	6	22	0.683	4.10
6	0.65	5	16	0.696	3.48
7	0.70	3	13	0.757	2.27
8	0.75	2	6	0.785	1.57

the requirements has ranged from 1 to 2, and the number of programmers who are included in the reserve team has ranged from 9 to 1. As a result, the evolutionary process has been accompanied by a steady increase in the total threshold weighted qualification of the set of teams represented by the best chromosome.

Table III
STAFF OF WORKING AND RESERVE TEAMS

Run	**Working teams**	**Reserve team**
1	g1={6,7,15,17,24}, g2={11,20,23}, g3={1,3,5,14}, g4={8,19,21}, g5={18}, g6={16}, g7={22}, g8={2,12}, g9={10,13}	{4,9}
2	g1={8,13,15,21,24}, g2={2,5,6,9,14}, g3={3,4,7}, g4={12,22}, g5={10,11}, g6={16,19}, g7={1,17,23}, g8={18}	{20}
3	g1={4,11,17,20}, g2={3,5,14,19}, g3={7,9,13,22}, g4={1,12,15,24}, g5={2,6,21}, g6={18}, g7={8,23}, g8={16}	{10}
4	g1={3,4,6,9,11,17}, g2={5,19,21,24}, g3={10,14}, g4={8,12,15}, g5={1,7}, g6={18}, g7={16}, g8={13,22}	{2,20,23}
5	g1={2,4,7,9,14,16}, g2={3,6,10,13,17,21,24}, g3={12,22}, g4={1,5,8,23}, g5={11,15}, g6={18}	{19,20}
6	g1={1,2,4,5,8,14,15}, g2={12,16,20}, g3={10,11}, g4={19,22,23}, g5={18}	{3,6,7,9,13, 17,21,24}
7	g1={2,9,10,11,13,14,22,23}, g2={7,12,15,16}, g3={18}	{1,3,4,5,6,8, 17,19,20,21,24}
8	g1={7,10,11,12,15}, g2={18}	{1,2,3,4,5,6,8, 9,13,14,16,17, 19,20,21,22, 23,24}

CONCLUSION

This paper has presented a method of assessing the qualifications of development teams that takes into account their knowledge and skills in programming technologies and tools. We have developed a genetic algorithm to optimize the size, staff and number of teams, which maximizes the overall qualification of the teams and minimizes the number of programmers not involved in them. Further research will focus on the integration of the qualification aspects and the performances of programmers and teams of programmers, which participate in executing big projects.

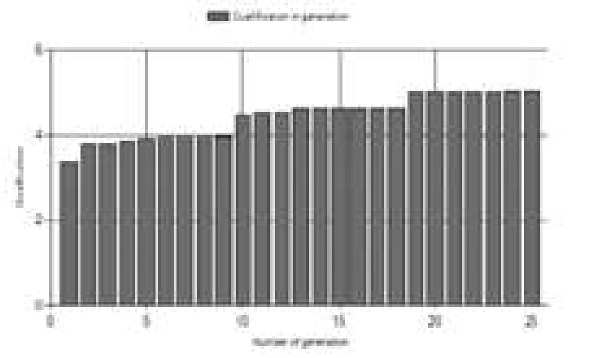

Figure 6. Weighted total qualification of all teams of programmers vs. generation number in GA.

REFERENCES

[1] Joshi, S. Agile Development - Working with Agile in a Distributed Team Environment / S. Joshi // MSDN Magazine, 2012, Vol.27, No.1, pp.1-6.

[2] Prihozhy, A. Lecture notes on "Modeling and Optimization for Engineering Systems Design" / A. Prihozhy // BNTU, Software for Computers and Automated Systems Dpt., 2013, pp. 58-69.

[3] Barricelli, N.A. Symbio genetic evolution processes realized by artificial methods / N.A. Barricelli // Methodos, 1957, pp. 143-182.

[4] Muller, J.P., Rao, A.S., Singh, M.P. A-Teams: An Agent Architecture for Optimization and Decision-Support, Proceedings 5th International Workshop, ATAL'98 Paris, France, July 4-7, 1998, pp. 261-276.

[5] Prihozhy, A.A. Evolutionary Method of Software Teams Optimization for Reducing Time and Resources of Project Execution / A. Prihozhy, A. Zhdanouski // Proc. Conf. "Information Technologies in Engineering and Business", Minsk, RIHS, 2016, pp.16-20.

[6] Prihozhy A., Method of qualification estimation and optimization of professional teams of programmers / A. Prihozhy, A. Zhdanouski // System analysis and applied information science, No 2, 2018, pp. 4-12.

[7] Red Monk, [electronics resource] / the developer-focused industry analyst firm Red Monk. – Access mode: http://redmonk.com/sogrady/2016/07/20/language-rankings-6-16/. – Date of access: 26.02.2017.

[8] Cass, S. The 2016 Top Programming Languages [electronics resource] / IEEE Spectrum, 2016. —— Access mode: http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages. – Date of access: 26.02.2017.

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ОПТИМИЗАЦИИ ЧИСЛЕННОСТИ, ПЕРСОНАЛА И КОЛИЧЕСТВА ПРОФЕССИОНАЛЬНЫХ КОМАНД ПРОГРАММИСТОВ

Прихожий А.А., Ждановский А.М.

В статье рассматривается проблема создания и оптимизации команд программистов с учетом их квалификации и уровня навыков в технологиях и инструментах программирования. Она предлагает метод формализации и оценки квалификации отдельных программистов и целых команд программистов, а также предлагает генетический алгоритм оптимизации численности, персонала и количества команд. Экспериментальные результаты на множестве программистов, окончивших вузы Беларуси, показывают способность системы находить команды программистов, которые повышают суммарную квалификацию коллектива разработчиков на 30%. Полученные результаты подтверждают практическую значимость модели, генетического алгоритма и программного обеспечения в области технологий и инструментов для управления профессиональными командами программистов.

Received 10.01.19