UDC 004.41:378.4:005.336.2

Olena H. Glazunova
Doctor of Pedagogical Sciences, Professor, Dean of the Faculty of Information Technology
National University of Life and Environmental Sciences of Ukraine, Kyiv, Ukraine
ORCID ID 0000-0002-0136-4936
o-glazunova@nubip.edu.ua

Oleksandra V. Parkhomenko
PhD of Pedagogical Sciences, Assistant Professor of the Chair of Information Systems and Technologies
National University of Life and Environmental Sciences of Ukraine, Kyiv, Ukraine
ORCID ID 0000-0002-0136-4936
oleksa.par@nubip.edu.ua

Valentyna I. Korolchuk
PhD, Associate Professor of the Chair of Information Systems and Technologies
National University of Life and Environmental Sciences of Ukraine, Kyiv, Ukraine
ORCID ID 0000-0002-3145-8802
korolchuk@nubip.edu.ua

Tetiana V. Voloshyna
PhD of Pedagogical Sciences, Associate Professor of the Chair of Information Systems and Technologies
National University of Life and Environmental Sciences of Ukraine, Kyiv, Ukraine
ORCID 0000-0001-6020-5233
t-voloshina@nubip.edu.ua

BUILDING THE PROFESSIONAL COMPETENCE OF FUTURE PROGRAMMERS USING METHODS AND TOOLS OF FLEXIBLE DEVELOPMENT OF SOFTWARE APPLICATIONS

Abstract. To support their professional expertise, a modern programmer must constantly follow new technologies, learn new methods of solving problems (best practices), exchange experience, use auxiliary tools that accelerate the development process, should be able to work in a team and develop their knowledge and skills. The task of modern IT education is to meet the demands of the information technology market. Specialists should be provided with proper training, which will give them relevant professional competences. The present paper analyzes modern methodologies of flexible software development and tools, defines professional competences related to software development based on the standard of higher education for the specialists of the Bachelor's academic level in Software engineering specialty. The novelty of the research lies in the justification of the competency-based approach to the training of future programmers. This approach involves the use of methods and tools for flexible development of software applications in three stages of project tasks of different types and complexity, which are formed in accordance with certain professional competencies. At Phase 1, students studied flexible methodologies and tools for developing software applications in the Software Design academic discipline. In Phase 2, flexible methodologies and software development tools were used during academic and technological practical training, in particular, students performed a group project according to the Scrum methodology, using Kanban approaches. In Phase 3, students worked individually on the Bachelor's thesis under the guidance of a teacher. The article describes the organization of the work process on the principles of flexible development and flexible learning, presents the results of experimental research, which showed an increase in the level of professional competencies in software development. A statistical analysis of the results of the experiment has been carried out and their significance has been proved.

Keywords: flexible methods of software development; flexible software development tools; future programmers.

© Olena Glazunova, Oleksandra Parkhomenko, Valentyna Korolchuk, Tetiana Voloshyna, 2022.

1. INTRODUCTION

Problem statement. The current stage of development of the world society is closely connected with the rapid development of information technology. The complexity of information systems is constantly growing, respectively, the issues of effective management of the processes of creation, testing and implementation of such systems are becoming relevant. Scopes, software requirements, development approaches and tools are changing. Information technology covers almost all areas of human activity. There are more and more developments in the fields of BigData, Internet of Things, artificial intelligence, virtual reality, cloud technologies, increasing demand for specialists in programming, software engineering, design and development of information systems.

There is a need to train information technology specialists, possessing modern technologies that are in demand in the labor market. However, the traditional education system has certain problems with the rapid introduction of new technologies into the educational process, which is primarily due to the need to update the content and technological base. A study of articles, forums, professional communities and our own experience of working with IT-specialists suggest that to maintain professionalism, the programmer must be familiar with various new technologies, know the best solutions to certain tasks (best practices), use tools to speed up the process development, be able to work in a team, and replenish their knowledge and skills. The problem of training specialists in higher education institutions in accordance with the requirements of the labor market and the formation of their professional competences is becoming urgent. Thus, the task of modern education is to meet the demands of the information technology market and to train specialists who will have relevant knowledge and competences that are embedded in the educational standards of training.

It is also important that modern programming is collective, and the usefulness of an individual programmer is closely related to his/her usefulness to the whole team, and therefore requires teamwork skills, leadership skills, some knowledge of psychology and management. The collective nature of the development process and the complexity of the developed systems lead to the emergence and spread of specific technologies, methods, approaches to project management for software development. Managing the development and maintenance of complex information systems and software products under the conditions of uncertainty leads to the use of new approaches and tools. In addition, the feature of a successful programmer is not a fixed set of knowledge, skills and abilities in a particular field, but the formation of a range of competences. Building the educational process on the basis of the competency approach is the most effective way to ensure the training of a specialist in software engineering in accordance with modern requirements of higher education and society.

The competency approach, in addition to developing a certain set of professional knowledge, skills and abilities in software engineering, also focuses on the development of such qualities as teamwork, leadership, responsibility, the ability to reflect, to learn and to master new technologies throughout life, self-education, activity planning, logical and algorithmic thinking, purposefulness, persistence, ability to make independent decisions and to quickly adapt to new conditions and learn constantly. In addition, there is a demand for specific knowledge of psychology and management, in particular, project management in the framework of flexible software development methodologies.

Thus, there arises a **problem** of implementing a competency-based approach to training programmers using new methods and technologies that are widely used in real software development projects, among which the most important are the methods of flexible software development and tools.

The purpose of the study is to analyze the methods and tools of flexible software development, to develop approaches to the formation of professional competences using these methods and tools at different stages of training future programmers.

The review of recent papers and publications. Problems of training future programmers, in particular, the formation of their professional competences, are considered in the works by T. Vakaliuk, V. Kontsedaylo, D. Coppit, P. Kamthan, V. Kruglik, V. Osadchy, D. Saito, V. Sedov, A. Stryuk, A. Takebayashi, Ts. Yamaura and other scientists.

V. Kontsedaylo developed an original model, which involves improving the content of the "Professional Practice of Software Engineering" course through the use of game simulators in the learning process. The author recommends forms, methods and teaching aids that should be used to form the professional competences of future software engineers. The author's methodology includes the use of selected game simulators (SimSE, Game Dev Tycoon, Software Inc.) [1] and the following teaching methods: project method, adaptive learning, situation modeling, testing. The author suggests the following forms of conducting educational classes with the use of selected game simulators: practical classes, trainings, consultations, independent work [2].

V. Sedov proposes to improve the formation of professional competence of future software engineers in the Master's program. The author has developed course packages in such academic disciplines as: "Internet of Things", "Cloud technologies in education", "Programming of microcontrollers and the Internet of Things." The paper describes the models of practice-oriented activities of students-programmers in the process of hackathons [3].

According to A. Stryuk, in the process of training bachelors in Software engineering, it is necessary to develop modules that provide effective support for the educational process in groups, which will be formed taking into account the organizational structure of the enterprise or educational institution, using the appropriate Agapa system [4].

In the initial stages of training future programmers A. Saito, A. Takebayashi, Ts. Yamaura [5] recommend the use of the Minecraft tool, because using this tool students can act as project participants, accordingly applying the project method of learning. The article describes the results of several years of gradual adaptation of Agile technology to the learning process, its introduction into the learning process and the results of the developed methodology approbation [6].

Many researchers consider the use of a competency-based approach in higher education. The effectiveness of this approach for vocational training is substantiated in the works of T. Lytvyn, K. Rajaram, Mollov M., Stoitsov G. and others.

In particular, according to T. Lytvyn, the main result of the competence approach is the comprehensive acquisition of knowledge and methods of practical activities, which allows future professionals to acquire competencies necessary for professional activities, and therefore successfully realize themselves in various fields [7].

K. Rajaram notes that universities and accreditation bodies need to develop curricula based on the competency-based approach, which will enable students to develop lifelong learning skills through self-directed learning and meet the challenges of the 21st century work environment [8].

M. Mollov and G. Stoitsov in their research demonstrate the effectiveness of the competence approach based on spiral competency development to be applied in the education modules "Introduction in object-oriented programming" (IOOP) and "Object-oriented programming" (OOP) in preparing students majoring in " Applied Programmer" of the National Program "Education for IT Careers " (NPEITC) in Bulgaria [9].

In today's world, the IT industry is facing new challenges in developing complex software systems in conditions of uncertainty associated with constant change, and the need to adapt to them. Technologies, approaches and tools for software application development are changing

and, accordingly, the requirements for their development are changing under the influence of external factors. Such conditions have led to the emergence and widespread use of flexible software methodologies, which have significantly influenced the software development process, making it iterative, modular and focused on product quality.

2. RESEARCH FINDINGS

Training IT specialists for IT-industry is a rather complicated process under the conditions of dynamic changes in the field of IT-products. Universities face problems of technological lag behind the level of the development of the IT-industry overall, lack of access to new technologies, the necessity of continuous improvement of curricula and training programs [10]. Future IT specialists need to develop the ability to solve complex specialized problems in the course of their studies in order to be leaders in their professional activities and meet the requirements of their clients, to apply modern methods and technologies of IT-solutions development [11].

One of the ways to improve the process of training future programmers is to use flexible methodologies in teaching and, accordingly, tools for flexible software development. The general advantage of using flexible methodologies for software development is that students not only get the opportunity to get acquainted with modern development methods, but also use them in the implementation of software development projects. This creates conditions close to real, for the acquisition of professional competences in software development directly during training. Project training of future programmers must be implemented using modern methodologies and real case studies.

The study on the formation of professional competences of future programmers focused on competences related to software development. The standard of higher education in the specialty 121 “Software Engineering” of the first (Bachelor’s) level of higher education defines the following competences for software development: C13, C14, C15, C16, C17, C19, C22, C23, C25 (Table 1). The selected competences correspond to the process of software development, their codes, names and place in the life cycle of the software product are given in Table 1. For the stages of the life cycle, the basic stages of the cascade model were selected as those that are common to all models of software development, namely: requirements analysis, design, development, testing.

Table 1

Competences of future programmers in programming, formed at the phases of software development

Competences	Software product life cycle phase
C13. Ability to identify, classify and formulate software requirements	requirements analysis
C14. Ability to participate in software design, including simulation (formal description) of its structure, behavior and operations	designing
C15. Ability to develop architectures, modules and components of software systems	development
C16. Ability to formulate and meet software quality requirements in accordance with customer requirements, specifications and standards	testing

C17. Ability to adhere to specifications, standards, rules and recommendations in the professional field in the implementation of life cycle processes	requirements analysis, designing, development, testing
C19. Knowledge of data information models, ability to create software for data storage, production and processing	designing, development
C22. Ability to accumulate, process and systematize professional knowledge on the creation and maintenance of software and recognition of the importance of lifelong learning	designing, development, testing
C23. Ability to implement phases and iterations of the life cycle of software systems and information technology based on appropriate models and approaches to software development	requirements analysis, designing, development, testing
C25. Ability to reasonably select and master software development and maintenance tools	designing, development, testing

The study analyzed flexible software development methodologies (Agile software development). The family of flexible methodologies of Agile development processes is based on the Agile Manifesto, which defines the values and principles that guide successful teams, mainly in software development. Ideological principles were formulated in four values and 12 principles documented in the Agile Manifesto, a manifesto of flexible development (Agile Manifesto). Agile embraces a set of techniques and methodologies that help the team think, work and make decisions more effectively. These methods and methodologies cover all areas of traditional programming, including project management, software design and architecture, and process optimization. All methods and methodologies consist of procedures that are as clear and optimized as possible and easy to apply. In practice, different Agile approaches can be combined based on the goals and conditions of the projects. Scrum is applied for cooperation and focus, Kanban – for efficiency and visibility of the process, extreme programming – for the best software development practices, economical production (Lean) – for waste disposal, function-driven development (FDD) – for process organization and reporting, dynamic system development method (DSDM) – for timing and budgets, Crystal Methods – for personalized process structure. For some years now the most popular Agile methodology among respondents has been Scrum (66%), ScrumBan – 9%, Scrum supplemented by Extreme Programming (Scrum / XP Hybrid) – 6%. That is, 81% of the surveyed companies chose Scrum (Fig. 1).

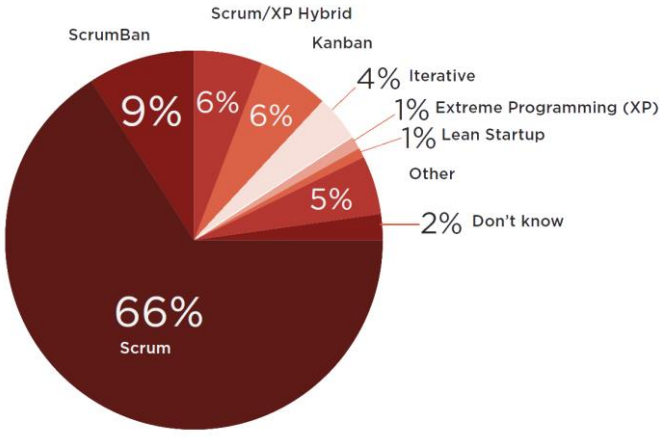

Fig. 1. Application of Agile methodologies (source: https://www.scrum.ua/blog/articles/chto-proishodit-v-mire-agile-v-2021-godu)

The introduction of flexible methodologies in the learning process optimizes the approaches to the training of future IT specialists. Implementation of flexible methodology of software development into the process of preparing the teaching and learning materials based on Agile methodology and adaptive approach in teaching not only contributes to students' better understanding and assimilation of theoretical and practical teaching material, but also helps students to understand production processes in IT-projects, sets them up to work within Scrum, gives an initial insight into the concept of complexity of the task, mandatory and additional product requirements – their knowledge within the academic discipline [12].

In her work, S. Leschuk [13] described selected methodological aspects of the use of Scrum methodology in the implementation of the software development project in the training of future information technology specialists. As a result, students gained practical experience in applying object-oriented programming principles; version control systems; web development software tools; MVC architectural template; working with JavaScript libraries; PHP programming using frameworks.

Y. Slyusarchuk, L. Javala, L. Uhryn investigated the problem of IT education quality, namely the improvement of professional competence of future specialists during project-oriented training based on Scrum-methodology using Moodle environment tools [14].

The use of flexible methodologies and approaches to software development modifies development process the most and affects all stages accordingly. This leads to the use of special programs and services, besides the development environments themselves – where programmers write a program code. The development process is divided into iterations, sprints, tasks and there arises a need to monitor compliance. Software project managers – services and applications – are used for such tasks. If in classical project management the main way to visualize the project is the Gantt charts, then in flexible methodologies it is Kanban boards, which are called "Kanban board", "project board", "agile-board" depending on the product interface.

The main purpose of such services is to visualize tasks, divide them into stages that look like columns on the board, prioritize tasks, assign project participants to perform certain tasks, limit tasks at work. In addition, there may be opportunities to link tasks with code in the repository; a defect tracking system is necessary for quality specialists and testers; future specialists also learn how to organize continuous integration and delivery of code, combining tasks into certain units: sprints, epics, graphical statistics and project metrics such as development speed, combustion charts. Among the popular services for working on agile projects are: Jira, Trello, Asana, Monday.com, Pivotal Tracker, HuBoard. In addition, there are special programs and services to ensure the continuous delivery and integration of software code, to test manual vs. automated ones, to work together on software code.

For IT projects, special attention should be paid to such cloud tools as version control systems [15], [16]. These include centralized version control systems (Subversion, Concurrent Versions System, Team Foundation Server) and distributed version control systems (Git, GitHub, GitLab, Gogs, and others).

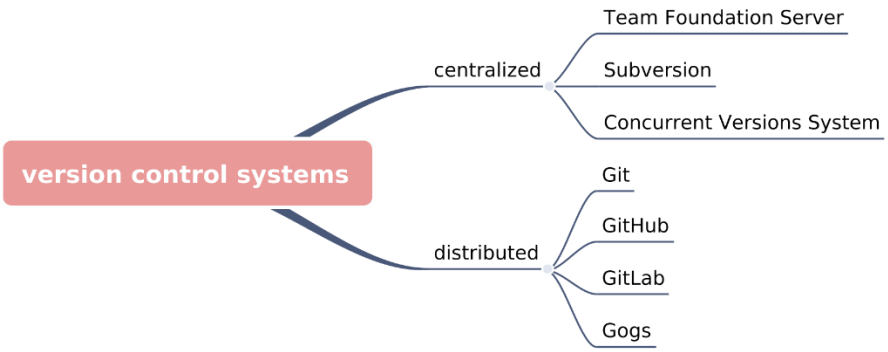

Fig. 2. Flexible software development tools

Distributed version control systems are most often used in software development because they are based on a distributed file storage model and do not require the use of a server, which allows each project developer to work with their own repository. That is why the formation of skills in working with flexible development tools for future programmers should start with distributed version control systems.

The following criteria and indicators should guide the selection of flexible software development tools for use in the educational process [17]:

- according to the structure-functional criterion: the presence of a repository; ability to manage versions; availability of a code editor; the presence of visualization of the work performed; availability of task boards; integration opportunities; platform support;
- according to the educational-communication criterion: the possibility of creating a project with other participants; the ability to assign tasks to users; possibility to discuss the code; community of users to share experiences; free version.

Comparative characteristics of tools for flexible software development according to the proposed criteria and indicators are given in Table 2.

Table 2

Comparative characteristics of flexible software development tools

Indicators	Git	GitHub	GitLab	Gogs
Structure-functional criterion				
the presence of a repository	-	+	+	-
ability to manage versions	+	+	+	+
availability of a code editor	-	+	+	-
the presence of visualization of the work performed	-	+	+	+
availability of task boards	-	-	-	-
integration opportunities	GitHub, GitLab, Visual Studio	Jira, HuBoard, Visual Studio	Jira, Confluence, Visual Studio, GitHub	GitHub, GitLab, Google+, QQ, Weibo
platform support	Windows, Mac	Windows, Mac, web application	Linux, Ubuntu, web application	Ubuntu, web application
Educational-communication criterion				
the possibility of creating a project with other participants	+	+	+	+

the ability to assign tasks to users	-	-	-	-
possibility to discuss the code	-	+	+	+
community of users to share experiences	-	a large number of open code repositories	predominantly closed private company repositories	-
free version	+	+	+	+

One of the most common in modern development is the Git-distributed system of file version management and collaboration, which is indispensable for the vast majority of IT developments worldwide. Working with Git requires teamwork and promotes accountability to other members of the team, while facilitating the collection of code fragments and ensuring reliability. Any changes are recorded and you can always track which user made the changes and when.

This tool is effectively used for collaborative and individual work with program code, which is also used for collaborative work on documents. There are currently no analogues of Git with its system for storing molded versions of files instead of sequences of changes.

The GitHub Flexible Software Development Tool is one of Git's online services that hosts Git repositories and provides other features such as tracking software development issues. GitHub is the most common platform for hosting open source projects. The software developer community identifies the GitHub cloud service as the primary platform for managing software projects and supporting collaborative IT software development [18]. As viewed by S. Gunnarsson, P. Larsson, S. Mansson, E. Martensson, J. Sonnerup, GitHub can be used in a variety of ways, but students need a certain amount of knowledge about Git [19]. In the process of training IT students, teachers use GitHub as a learning tool for programming courses, posting code samples and managing student tasks, organize collaborative work [20]. A. Zagalsky, J. Feliciano, M. Storey, Y. Zhao, W. Wang in their work [21] describe the use of the GitHub cloud service as a shared learning platform that promotes effective collaboration and openness of students in the learning process.

GitHub's cloud-based version control service in programming training allows you to receive an electronic information support for the project, simplifies joint work with information, allows online communication, dissemination of results to the general public, simplifies the formatting and planning of activities, and increases the motivation of participants.

In addition to GitHub, GitLab is another widely used service for working with Git repositories. It is a web application and Unix management system of software code repositories for Git. GitLab offers solutions for storing its own code and joint development of large-scale programs. L. Haaranen, T. Lehtinen [22] conducted a study in which the flexible development tools Git and GitLab were used in a training course in computer science. According to researchers, the use of such tools helps to develop students' skills in working with appropriate services for collective IT development of software products, which is now needed for successful work in the field of information technology.

Gogs is a good alternative to GitHub and GitLab. It is much more economical in terms of the consumption of system resources than any similar solution. Gogs is a self-hosted open-source Git server written in Go, which includes a repository file editor, a project tracking system, and a built-in wiki.

It is expedient to develop skills of software development in future programmers with the use of flexible methodologies and the use of appropriate tools in three phases. At the first (preparatory) stage, students were offered to study flexible methodologies of software development within the Software Design academic discipline. Theoretical material for students was placed in the e-learning course (ELC) of the relevant academic discipline, which is available to students online anytime from anywhere. The training was based on the principles

of blended and flexible learning (agile-learning). Initially, to get acquainted with the material, students were offered the appropriate resources of the ELC: resource Book for the study of theoretical materials, resource Tasks for independent work to choose one of the proposed by MOOC. In the classroom, students updated their knowledge, mostly by doing teamwork. In order to determine the level of mastery of theoretical material, the Kanban board with card-themes was proposed, in which the basic concepts to be mastered were described in each topic. After mastering the theoretical material, the students moved the card to the "studied" column, and when the team finished working with all the cards, they performed the tasks: the case-project. The assignment offered a description of the specific situation – the customer's requirements for software development, namely: to analyze the requirements, to find out which ones should be clarified, and, after proposing and justifying the choice of a software development model for its task, as a result, to present a task developed on the basis of the relevant requirements in the form of the methodology chosen by the team.

Fig. 3. Example of ELC in Software Design academic discipline

At this phase, students formed competences related to the analysis of requirements, selection of tools and technologies for software development, as well as compliance with the recommendations of the profession and the accumulation of necessary professional knowledge, namely: C13, C16, C17, C22, C23, C25.

During Phase 2 (practical), future programmers used flexible software development methods during academic and technological practical training, where they performed a group project using the Scrum methodology, using Kanban tools and approaches, tools for flexible software development. The following tools were prepared to start working on the project: setting up Git, registering in GitHub, creating repositories, setting up HuBoard, providing

access to team members and the teacher to the main repository (Master), testing the created environment. To support the project, students also had to use a web service to organize the project, implement management (Kanban board) and monitor tasks and progress. Students could choose the tools for support of their choice or from the ones offered: GitHub, Jira, Trello, HuBord, and use them based on the tasks they have to assign to the respective roles: scrum master, developers, testers. An example of using the GitHub service while working on a project is shown in Fig. 4.

Since during Phase 2 students are offered the task of a team project that is as close as possible to the actual working conditions of a programmer, they will develop the following professional competences in the process of its realization: C13, C14, C15, C16, C17, C19, C22, C23, C25.

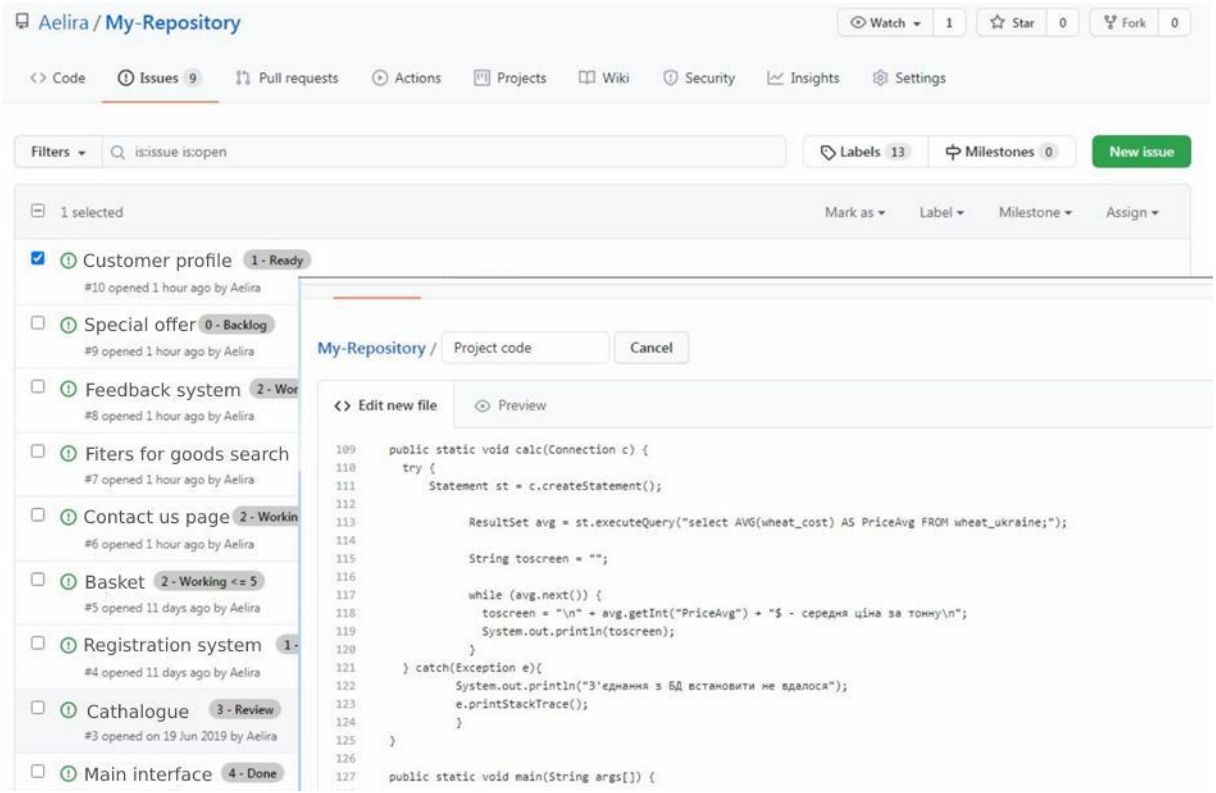

Fig. 4. Project code in GitHub service

At Phase 3 – Qualification, students worked independently on their Bachelor's thesis under the guidance of a teacher using flexible methodologies and tools for software development. The work was organized on the principles of flexible development and agile learning. Since this is a complex project, it is advisable to use flexible methodologies during its implementation, rather than a cascading or similar life cycle model, which is likely to fail the Bachelor's thesis in a timely manner, which is a significant drawback. The student can choose the methods and tools for developing software applications, the main features of which are: constant feedback from the teacher, iterative-incremental approach, breaking the whole project into separate tasks, development of main tasks, clarity of development process, use of systems version control, retrospective. Feedback time – we chose 2 weeks of sprint time. Each student had to independently determine the main objectives of their project, organize them in the form of cards on the Kanban board. Special attention was paid to the prioritization of tasks, because the main limitation is the time of implementation. The tools for organizing the Kanban board were chosen by the students themselves, such as HuBoard, Trello, Jira, because after the

training and technology practice they will have already mastered the HuBoard tool and they can easily use any other similar tool, usually in web service format to provide access to the scientific supervisor. Figure 5 shows the work on the Bachelor's/Master's thesis divided into separate tasks. The process is designed in such a way that if a student fails to complete all the work, he/she can proceed to the final steps, to the presentation design, without losing the essence of the work, since the most important tasks will be executed in the first place.

Phase 3 presupposes the development of the following professional competences: C13, C14, C15, C16, C17, C19, C22, C23, C25, as the students independently develop a software application that includes all phases of its development.

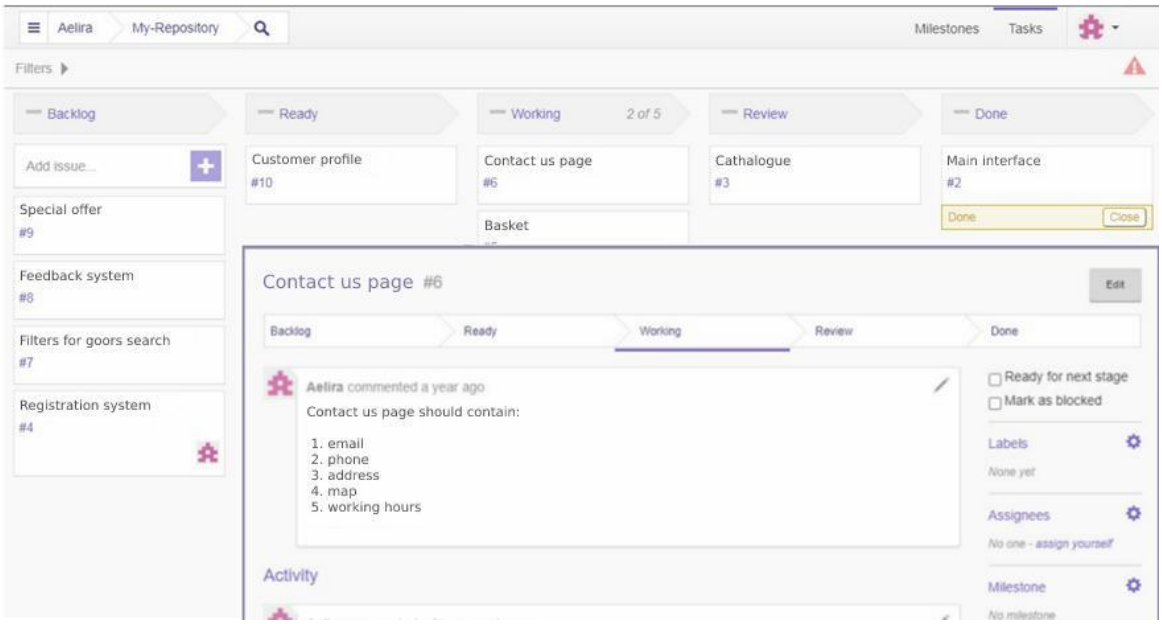

Fig. 5. Tasks in HuBoard on Bachelor's thesis performance

In order to test the impact of the use of flexible methodologies and software development tools on the formation of programming competence in future programmers, a pedagogical experiment was conducted in the process of preparing students of the Bachelor's academic level, specialty 121 "Software Engineering", which involved 379 students (experimental group – 197 students and control group – 182) of the National University of Life and Environmental Sciences of Ukraine during three phases.

During the experiment, the students of the experimental group (EG) deepened their knowledge and applied it in practice, carrying out a team project on software development. The development was based on flexible methodologies of software development using selected tools of flexible development. In the control group (CG) the development was carried out according to traditional methods (linear development).

Tests based on previously defined competency indicators were developed to measure students' attainment of a certain level of programming competence. The thesis [17] provides a table of indicators to the structural components of competence in software development. A 100-point scale was used to assess the level of achievement of the indicator. All the students who participated in the pedagogical experiment were evaluated on each indicator. The average value for all indicators was taken to form a contingency table. The clustering of values was carried out according to the following principle: students who received an average score from 60 to 73 have a low level of programming skills, from 74 to 89 – average, from 90 to 100 – the high one.

Prior to the start of the experiment, the assessment of the observational section in the control and experimental groups was performed, and the same evaluation was performed after the experiment (final section). The obtained results of the formation of programming competences at three levels in two groups are presented in Table 3.

To test the effectiveness of the proposed approach to building programming competence in future programmers, it was suggested that the use of flexible methodologies and software development tools in the training of future programmers influences the formation of programming competence. Frequency tables (contingency tables) and $\chi 2$ criterion were used to statistically support this assumption. These statistical methods are used when two variables (samples) are categorical and ordinal variables, which corresponds to the conditions of the assumption.

Table 3

Distribution of students in control and experimental groups according to the levels of the formation of programming competences based on the results of statement and final assessments

Level	Statement assessment (number of students)		Final assessment (number of students)	
	CG	EG	CG	EG
low	24	32	18	14
average	86	88	80	59
high	72	77	84	124
Total	**182**	**197**	**182**	**197**

Thus, Table 4 presents the grouped results of the assessment of programming competences at three levels, as well as the expected (theoretical) frequencies in the control and experimental groups according to the results of the statement assessment.

To test the hypothesis, the null hypothesis H0 was formed: no differences between the levels of competences in software development in the students of control and experimental groups and the alternative hypothesis (H1): the presence of significant differences between levels of programming competences formation in students of control and experimental groups. To test these hypotheses, Pearson's criterion was calculated based on the results of the assessment. Table 4 offers the results of the calculations.

Table 4

Calculation of $\chi 2$ Pearson's criterion, statement assessment

Level	Group	Empirical frequency	Theoretical frequency	$(f_E - f_T)$	$(f_E - f_T)^2$	$(f_E - f_T)^2 / f_T$
low	CG	24	26.89	-2.89	8.35	0.311
	EG	32	29.11	2.89	8.35	0.287
average	CG	86	83.56	2.44	5.95	0.071
	EG	88	90.44	-2.44	5.95	0.066
high	CG	72	71.55	0.45	0.2	0.003
	EG	77	77.45	-0.45	0.2	0.003
	sums	379	379	-	-	0.741

The differences between the two distributions can be considered significant if χ^{2Emp} reaches or exceeds χ^2 0.05, and even more significant if χ^{2Emp} reaches or exceeds χ^2 0.01. The

calculated value of the Pearson test according to the results of the statement section (for the significance level of 0.05 and 2 degrees of freedom) is less than the critical: χ^{2Emp} = 0,741. Therefore, since at the beginning of Phase 2 χ^{2Emp} is less than the critical value (5,991), the differences between the distributions are not statistically significant (null hypothesis is correct).

The same calculation was performed after the experiment (final section) (Table 5), and the obtained calculated value of Pearson's criterion (for the level of significance of 0.05 and 2 degrees of freedom) is more than critical: χ^{2Emp} = 10.783. Therefore, according to the results of the experiment χ^{2Emp} is more critical, so the differences between the distributions are statistically significant (alternative hypothesis is correct).

Table 5

Calculation of χ2 Pearson's criterion, final assessment

Level	Group	Empirical frequency	Theoretical frequency	$(f_E - f_T)$	$(f_E - f_T)^2$	$(f_E - f_T)^2 / f_T$
low	CG	18	15.37	2.63	6.92	0.45
	EG	14	16.63	-2.63	6.92	0.416
average	CG	80	66.75	13.25	175.56	2.63
	EG	59	72.25	-13.25	175.56	2.43
high	CG	84	99.88	-15.88	252.17	2.525
	EG	124	108.12	15.88	252.17	2.332
	sums	379	379	-	-	10.783

If at the beginning of the experiment according to the results of the statement assessment the characteristics (levels of competence formation of software development in students) of control and experimental groups coincide, and the null hypothesis is confirmed, then at the end of the experiment the results of the final assessment confirm the alternative hypothesis (the calculated value of Pearson's criterion is greater than the critical one and makes 10,783). The comparative distribution of students in control and experimental groups according to the levels of competences formation in software development is presented in Fig. 6.

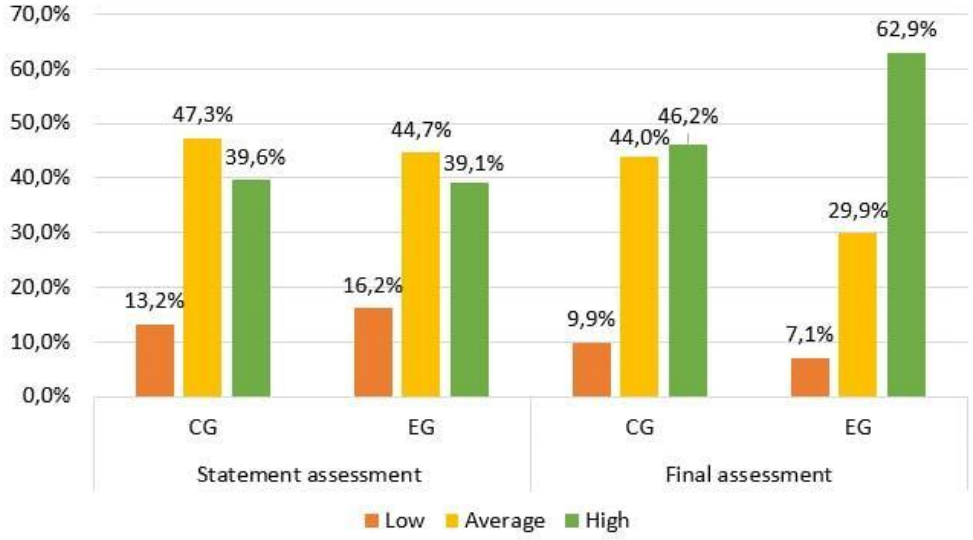

Fig. 6. Diagram of comparative distribution of students in CG and EG by levels of competences in software development

Comparing the levels of competences in software development of future programmers in experimental groups at the beginning and end of the pedagogical experiment, we see an increase in the share of students with high and average levels of these competences, namely by 9.1%.

3. CONCLUSIONS AND PROSPECTS OF FURTHER RESEARCH

As a result of the research, the problem of realization of the competence approach in the training of programmers involving the use of methods and technologies of flexible development of software applications, which are widely used in real projects on software development, has been solved.

The authors have developed an approach to the phased development of the professional competences of future programmers using methods and tools for flexible software development in three phases: preparatory, practical and qualifying. The preparatory phase includes the study of flexible software development methodologies based on the principles of blended and flexible learning in order to develop competences related to requirements analysis, selection of tools and technologies for software development, as well as compliance with recommendations and accumulation of necessary professional knowledge. To achieve the goal of the study, an e-learning course with various types of digital materials was developed, which could be processed remotely, and the tasks could be performed differently according to the level of personal competencies. The practical phase consisted in a group project based on the Scrum methodology combined with Kanban tools and approaches, as well as flexible software development tools during academic and technological practical training. During the qualification phase, students worked on their Bachelor's theses under the tutor's supervision, using flexible methodologies and tools to develop software applications.

At the same time, the experimental study confirmed that the number of students with a high level of professional competencies in software development in the experimental groups compared to the control ones increased by 32%. The second and third phases gave the possibility to build all the selected professional competences in software development.

We see the prospect of further research in the theoretical justification and development of methods for flexible training of future programmers based on the technology of "agile-learning."

REFERENCES (TRANSLATED AND TRANSLITERATED)

[1] V. V. Kontsedailo, T. A. Vakaliuk, "Selection criteria of games simulation used to develop professional competencies of the future software engineers". *Information Technologies and Learning Tools*, vol. 65, no. 3, pp. 133-151, 2018. doi:https://doi.org/10.33407/itlt.v65i3.2039. (in Ukrainian)

[2] V. V. Kontsedailo, "The use of simulation games in the development of professional competencies of future software engineers", Ph.D. dissertation, Institute of Information Technologies and Learning Tools NAES of Ukraine Kyiv, 2018. (in Ukrainian)

[3] V. E. Sedov, "Formation of professional competence of future engineersprogrammers under conditions of master degree", Ph.D. dissertation, Kherson state University, Kherson, 2016. (in Ukrainian)

[4] A. M. Striuk, "The «Agapa» system as a learning tool of System Programming for Software Engineering BA students", Ph.D. dissertation, Institute of Information Technologies and Learning Tools NAES of Ukraine Kyiv, 2012. (in Ukrainian)

[5] D. Saito, A. Takebayashi, Ts. Yamaura, "Minecraft-based preparatory training for software development project", in *International Professional Communication Conference (IPCC)*, Pittsburgh, PA, USA, 2014. doi: 10.1109/IPCC.2014.7020393.

[6] I. I. Harko, M. B. Pyroh, V. L. Mironova, "Application of agile-methodology in teaching algorithmization and programming basics for computer sciences speciality students". *Information Technologies and Learning Tools*, vol. 85, no. 5, pp. 147-162, 2021. doi:https://doi.org/10.33407/itlt.v85i5.4024. (in Ukrainian).

[7] T. Lytvyn, "Competence approach in the system of higher education of Ukraine: analysis of basic concepts". *Pedagogy and psychology of vocational education*, no. 2, pp. 9–14, 2012. (in Ukrainian)
[8] K. Rajaram, "Flipped Classrooms: Scaffolding Support System with Real-time Learning Interventions". *Asian Journal of the Scholarship of Teaching and Learning,* vol. 9, no. 1, pp. 30–58, 2019.
[9] M. Mollov, G. Stoitsov, "Competency development in the object-oriented programming style education". *TEM Journal*, vol. 10, no 4, pp. 1938-1944, 2021. doi:10.18421/TEM104-59.
[10] O. G. Glazunova, T. V. Voloshyna, N. Dorosh, "Development of professional and soft skills of future IT specialists in cooperation with leading IT companies". *Information Technologies and Learning Tools*, vol. 60, no. 4, pp. 141-154, 2017.
[11] O. G. Glazunova, T. V. Voloshyna, V. I. Korolchuk, "Hybrid cloud-oriented learning environment for IT student project teamwork", *Information Technologies and Learning Tools*, vol. 77, no. 3, pp. 114-129, 2020. doi:https://doi.org/10.33407/itlt.v77i3.3210.
[12] V. Mironova, M. Pyroh, I. Harko, "Methodology of Building Agile-Education Processes in Higher Education Institutions", in *IT&I Workshops*, 2020, pp. 409-417. [Online]. Available: http://ceur-ws.org/Vol-2845/Paper_38.pdf Accessed on: Dec. 12, 2021.
[13] S. O. Leshchuk, "Some methodological aspects of training IT professionals", *Information Technologies in Education*, vol. 1, no. 30, 2017. pp. 81–96. doi: 10.14308/ite000621. (in Ukrainian).
[14] Y. Slyusarchuk, L. Dzhavala, L. Uhryn, "Competence approach to it specialists training based on training project", pp. 29-34, 2015. [Online]. Available: https://science.lpnu.ua/sites/default/files/journal-paper/2019/jun/16830/06-29-34.pdf. Accessed on: Oct. 28, 2021. (in Ukrainian).
[15] M. Cochez, V. Isomottonen, V. Tirronen and J. Itkonen, "The Use of Distributed Version Control Systems in Advanced Programming Courses", in *ICT in Education, Research and Industrial Applications*, pp.221-235, 2013.
[16] M. Ch. Barukh, B. A. Benatallah, "Toolkit for Simplified Web-Services Programming", in *International Conference on Web Information Systems Engineering*, vol 8181, pp. 515–518, 2013. doi: https://doi.org/10.1007/978-3-642-41154-0_42.
[17] O. V. Parkhomenko, "The Use of Agile Software Development Methodologies in the Training of Future Software", Ph.D. dissertation, National University of Life and Environmental Sciences of Ukraine, Kyiv, 2021 (in Ukrainian).
[18] J. Feliciano, M. Storey, A. Zagalsky, "Student experiences using GitHub in software engineering courses: a case study", in *38th International Conference on Software Engineering Companion*, 2016, pp. 422-431. doi: 10.1145/2889160.2889195.
[19] S. Gunnarsson, P. Larsson, S. Mansson, E. Martensson, J. Sonnerup, "Enhancing Student Engagement Using GitHub as an Educational Tool", Lund: Genombrottet, Lunds tekniska hogskola, 2017.
[20] M. A. Angulo, O. Aktunc, "Using GitHub as a Teaching Tool for Programming Courses", *in Proceedings of the 2018 ASEE Gulf-Southwest Section Annual Conference the University of Texa*s. 2018. doi: http://dx.doi.org/10.26153/tsw/6948.
[21] A. Zagalsky, J. Feliciano, M.-A. Storey, Y. Zhao, W. Wang, "The emergence of github as a collaborative platform for education", in *Proc. of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, 2015, pp. 1906-1917.
[22] L. Haaranen, T. Lehtinen, "Teaching git on the side: Version control system as a course platform", in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, 2015, pp. 87-92.

Text of the article was accepted by Editorial Team 02.02.2022

ФОРМУВАННЯ ПРОФЕСІЙНОЇ КОМПЕТЕНТНОСТІ МАЙБУТНІХ ПРОГРАМІСТІВ З ВИКОРИСТАННЯМ МЕТОДІВ ТА ІНСТРУМЕНТІВ ГНУЧКОЇ РОЗРОБКИ ПРОГРАМНИХ ДОДАТКІВ

Глазунова Олена Григорівна
доктор педагогічних наук, професор, декан факультету інформаційних технологій
Національний університет біоресурсів і природокористування України, м. Київ, Україна
ORCID 0000-0002-0136-4936
o-glazunova@nubip.edu.ua

Пархоменко Олександра В'ячеславівна
кандидат педагогічних наук, асистентка кафедри інформаційних систем і технологій
Національний університет біоресурсів і природокористування України, м. Київ, Україна
ORCID ID 0000-0002-0136-4936
oleksa.par@nubip.edu.ua

Корольчук Валентина Ігорівна
доктор філософії, старша викладачка кафедри інформаційних систем і технологій
Національний університет біоресурсів і природокористування України, м. Київ, Україна
ORCID 0000-0002-3145-8802
korolchuk@nubip.edu.ua

Волошина Тетяна Володимирівна
кандидат педагогічних наук, доцент кафедри інформаційних систем і технологій
Національний університет біоресурсів і природокористування України, м. Київ, Україна
ORCID 0000-0001-6020-5233
t-voloshina@nubip.edu.ua

Анотація. Сучасний програміст для підтримки професіоналізму повинен постійно слідкувати за новими технологіями, дізнаватись про нові методики вирішення завдань (best practice), обмінюватись досвідом, використовувати допоміжні інструменти, що пришвидшують процес розробки, уміти працювати в команді та поповнювати свої знання і вміння. Відповідно до запитів ринку завдання сучасної ІТ-освіти - забезпечення підготовки фахівців, що володітимуть актуальними професійними компетентностями. У статті проаналізовано сучасні методології гнучкої розробки програмних додатків та відповідний інструментарій. Визначено професійні компетентності, які стосуються розробки програмного забезпечення на основі стандарту вищої освіти для фахівців освітнього ступеня "Бакалавр" спеціальності "Інженерія програмного забезпечення". Новизна наукового дослідження полягає в обґрунтуванні компетентнісного підходу до підготовки майбутніх програмістів, який передбачає використання методів та засобів гнучкої розробки програмних додатків на трьох етапах виконання проєктних завдань різного типу та складності, які формуються відповідно до професійних компетентностей. На першому етапі студенти вивчали гнучкі методології та інструменти розробки програмних додатків у межах дисципліни «Конструювання програмного забезпечення». На другому етапі гнучкі методології та інструменти розробки програмного забезпечення були застосовані під час навчально-технологічної практики, зокрема студенти виконували груповий проєкт за методологією Скрам, використовуючи підходи Канбан. На третьому етапі студенти працювали індивідуально над виконанням бакалаврської роботи під керівництвом викладача. У статті описано організацію робочого процесу на принципах гнучкого розвитку та гнучкого навчання, представлено результати експериментальних досліджень, які показали підвищення рівня професійних компетенцій з розробки програмного забезпечення. Здійснено статистичний аналіз результатів експерименту та доведено їх значущість.

Ключові слова: гнучкі методології розробки програмних додатків; гнучкі засоби розробки програмних додатків; майбутні програмісти.

(cc) BY-NC-SA

This work is licensed under Creative Commons Attribution-Non Commercial-Share Alike 4.0 International License.