# Software Reuse Approach Based on Review and Analysis of Reuse Risks from Projects Uploaded to GitHub

Olena Chebanyuk[1,2(✉)]

[1] Department of Informatics, New Bulgarian University, Sofia, Bulgaria
Chebanyuk.olena@gmail.com
[2] Software Engineering Department, National Aviation University, Kyiv, Ukraine

**Abstract.** Modern and large software systems usually are not developed from scratch. Reuse operations for small modules are not complicated activities. When reuse is organized on level of algorithms or software features, practices of many companies show that reuse procedures are performed on low maturity levels (analysis of software bugs and reuse risks are often performed approximately). According to researches of IBM and many other companies the later you will define any kind of error the more expensive and large scale will be cost of improving your software.

Paper proposes an approach based of reverse engineering activities aimed to estimate reuse risks of existing projects on GitHub before their further reuse. Proposed approach is designed by analysis of typical activities performed in research laboratories of software companies and considers specific of working with GitHub. Model for estimating of reuse risks is proposed. Model covers reuse risks for multilayer applications, but can be extended for other types of projects. Recommendations for developers for extension of the model are outlined.

**Keywords:** Software Reuse Risks · GitHub · Software Reuse Activities · Software Product Lines · Multi-layered Architecture

## 1 Introduction

Today development of software projects does not start from scratch. Software development teams expect to reduce development efforts by means of reusing existing software code, interface prototypes, documentation, and other software development artifacts.

From the first look, efforts to adopt architecture and functionality of existing projects to new ones could be estimated as not so difficult. However, in real projects efforts may become so great [1]. Often developer team thinks that the current changes are the last. And then, the procedures of changing are repeated many times. After that new area of adaptation are defined – for example – structure of data sets or necessary interface changing.


T. Zlateva and G. Tuparov (Eds.): CSECS 2023, LNICST 514, pp. 144–155, 2023.
https://doi.org/10.1007/978-3-031-44668-9_11

## 2 Review of Papers

Approaches that are devoted to software reuse and estimation of reuse risks are developed as two adjacent branches. Summarizing of reuse practices gives a background for development of risks estimation standards and formal approaches of software reuse considering potential development risks.

Paper [2] proposes domain model for considering different reuse risks. It allows to estimate the next factors:

- risk level and its effect on the final subjective and objective project performance;
- the ineffective performance of software projects and the consequential costs
- time overruns,
- missed business prospects;
- errors in software project management activities;
- evaluation and estimation of project performance;
- other related aspects.

It is difficult to use the proposed model because the research questions for validation of the model allowing to estimate different risks were formulated clearly, but answered only partially [2].

Andres Orrego, Tim Menzies, and Oussama El-Rawas have classified reuse activities from different points of view using models COCOMO and COQUALMO. Authors have proposed reuse models and have compared the benefits of software reuse to other activities [3]. Based on this research, project managers have to weigh drawbacks and advantages of software development artifacts reuse. These drawbacks and advantages should be evaluated using results from Software Product Line (SPL) approach practices. They are project-by-project estimation and comparing of reuse tactics with other project development strategies.

Classification of activities is based on ranking and connected with a value of different efforts, schedule, and strategies of using models with competing influences that have not been precisely tuned using local data [3].

Russ Cox has examined the possible negative effects of software reuse. The author has ten years of experience with Google's source code system, which recognizes software dependencies as a first-class reuse problems [4].

Software dependencies are a significant and often overlooked risk. The move to the cheap and granular software reuse has happened so quickly that developers don't yet understand the best practices for choosing and using dependencies properly. The goal is to raise awareness of the danger and encourage further research into possible protections.

The simple models estimating the total cost for reuse can be represented as the sum of the cost of all bad decisions multiplied by the probability of their occurrence:

$$expectedcosts = \sum_{a \epsilon badoutcomes} cost(a) \times probability(a) \quad (1)$$

***Conclusions from the Review.*** Review of the papers shows that investigations in the reuse area deeply focused on some aspects of software reuse. Results often are focused on proposing some models or analytical fundamentals as well as developing common recommendations to answer the question – "How to manage reuse risks?".

From the other side, many efforts and experiments are needed to gather proposed ideas to one complex approach with possibility to organize reuse procedure considering peculiarities of project types, reuse errors and stack of technologies.

It becomes a motivation for author to propose an approach that consider peculiarities of reuse activities for GitHub (because it is a popular software repository) that is used in everyday activities of different stakeholders teams from all over the World.

## 3 Task and Research Questions

In order to propose the approach for estimation of software reuse risks for GitHub projects it is necessary to solve the next Research Questions (RQs):

RQ1: Investigate typical bugs for GitHub projects with multilayer architecture and summarize, which types of bugs are specific for these projects.
RQ2: Propose a model of reuse risk estimation.
RQ3: Summarize peculiarities for activities, performed in research laboratories, as well as activities aimed to analyze GitHub projects.
RQ4: Propose an approach for estimation of reuse risks for GitHub projects that is based on analysis of source modules semantics, considering specifics of work with GitHub. In order to do this, systematize knowledge about the next aspects from theoretical and practical backgrounds:
RQ4.1: gather and analyze information about the reuse activities from research labs of companies that follow Software Product Lines approach;
RQ4.2: conduct the experimental research, summarizing peculiarities of activities for investigating of different reuse risks of newly downloaded projects. The aim of this research is to define typical bugs for different projects and estimate a dependency between reuse risks, number, and quality of types of bugs for different projects.

## 4 Gathering of Experimental Data

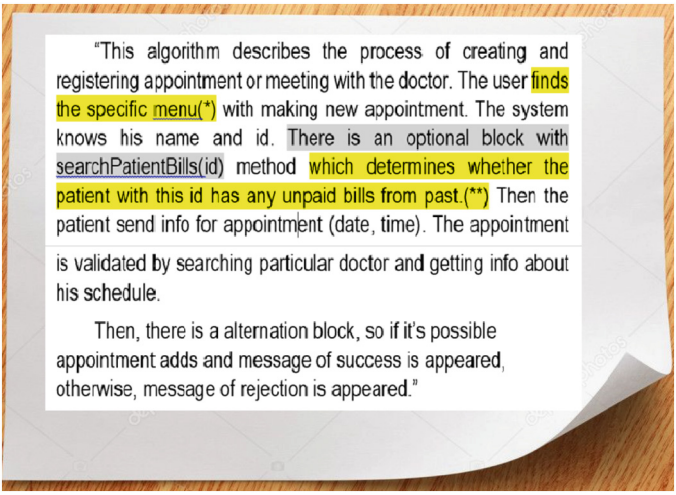
Aim of this chapter is to define which bugs may be expected in project with multilayer architecture. Projects with multilayer architecture contain three (or more) levels of architecture representation. Examples –web applications (MVC, MVP, MVVM, multilayer applications with monolithic architecture etc.).

The first reuse risk is based on unclear understanding of requirements by all stakeholders.

Consider a part of a requirement specification from a real project (Fig. 1). Text is given without editors' changes.

The first part of specification, marked in yellow and (*), may be understood in different ways by front-end developer, user, tester, and other stakeholders. The seconds part marked in yellow and (**) is an error, because analysis of payment practices for medical services shows that it is necessary to provide payment for current medical service immediately, not later. Grey-marked part contains extra technical detail that is not necessary for user.

As a result of analysis of this requirement specification the first risk is defined.


"This algorithm describes the process of creating and registering appointment or meeting with the doctor. The user finds the specific menu(*) with making new appointment. The system knows his name and id. There is an optional block with searchPatientBills(id) method which determines whether the patient with this id has any unpaid bills from past.(**) Then the patient send info for appointment (date, time). The appointment is validated by searching particular doctor and getting info about his schedule.

Then, there is a alternation block, so if it's possible appointment adds and message of success is appeared, otherwise, message of rejection is appeared."

**Fig. 1.** A fragment of a requirement specification

***Risk1 - "Unclear Requirement Specification" and May Be Estimated from 0 to 20.*** Other risks are defined on the base on analysis of different multilayer application projects. Let's consider results of analysis for different projects.

Project 1: Web-application, which allows to people who have limited vision to book flight tickets.

System goal: possibility to sound text labels and zoom components while booking tickets. Also, possibility to recognize key words using Speech to Text module to perform some part of user scenario. (For example system recognizes a word "reject" and rejects the last booking).

The GitHub project with the closest functionality [5], namely project allowing to book tickets without adaptation needs for weak-eyed people, was investigated. Link to the video, describing project, is represented in [6].

*Summary of analysis of bugs, defined in the project.*

Project has some testing bugs, namely errors that were not improved during testing. Examples of such bugs are the next: it is impossible to input correct information about child age and cities names by two words. Other errors are related to main functionality of project. One of them is a limited functionality for search and representation of information.

As a result of analysis of bugs for this project the next risk is defined: ***Risk2 – "Incorrect input processing"***. This risk is based on limiting of some program features because initial information for performing some features is absent. Usually such a risk partially blocks executing of some features of the project.

Bugs that are related to this risk:

– risk 2 bug 1 – "*Incorrect data range*". Examples from the considered project are: there is no possibility to input a child age or input a city name by two words.

– risk 2 bug 2 – *"Interface errors"*. Examples from the considered project: representation only four fields for searching of flights.
– risk 2 bug 3 – "*REST errors"*. Sometimes the source of these errors is that http requests are not sent from interface elements to business logic level. One of the example from the user point of view: user logs out from the system and continues to see pages, specific to his account.

*Description of the scheme of risks estimation.*

Estimation of all bugs was done by several software development teams the expert evaluations approaches. Calculated resulting values for bugs' estimations are represented below in this paper.

Risk value is defined as a sum of the values for all its bugs.

For example, value of risk2: *risk2 = risk2 bug1 + risk2 bug2 + risk2 bug 3.*

***Risk2 May Be Estimated from 0 to 15.*** Every bug in this risk may be estimated maximum to five points.

Consider the next project "System supporting CRUD operations for library".

The following project from GitHub about this topic was investigated [7]. Link for the video describing project is represented in [8].

As a result of analysis of bugs for this project the next risk is defined:

***Risk3 – Resource Missing.*** Such a risk may be estimated from 0 to 20. Every bug in this risk may be estimated maximum to ten points.This project sometimes has "no database response" situations. This bug is named *"missing database access" (risk 3 bug 1).* It happens when user can't get access to database (SQL server is not installed when it is needed or there is another business logic error).

Another bug is a *"missing of third-party service" (risk 3 bug 2)*, for example service from some cloud platform is not found.

Consider the next project: Game for study foreign languages.

Game goal: Improve user comprehension. Levels are started from listening words, then sentences, then reading of stories, and writing some words.

The following project from GitHub about this topic was investigated [9]. Link to the video is represented in [10].

During investigation of this project the next testing errors were defined: after switching the task, the window with previous task did not closed. As a result, user may confuse what task need to be completed. Thus, user's score may be calculated in a wrong way.

As a result of analysis of bugs for this project the next risk is defined:

***Risk4 – "User Story Error". Such a Risk May Be Estimated from 1 to 5.*** Usually such a risk may partially confuse user to perform some actions with software.

Consider the next project: Medical system for remote consultations.

System goal: to register a patient and a doctor. Possibility to search doctor and time for appointment to doctor. Doctor can reject or change day or time of appointment. Video conference for appointment. Possibility to send messages and transmit files between doctor and patient. Doctor fills forms with parameters about patient health. System can visualize information about dynamic of patient health.

The next project from GitHub about this topic was investigated [11]. Link to the video is represented in [12].

As a result of analysis of this project the next bugs were defined:

- risk3 bug 1, namely "*missing database access*". Project depends upon database version. The same bug is met one more time when new information is added or changed. It is implemented as "database updating error".
- risk4, namely *"User story risk"*, is defined. It is represented as unexpected closing of program.

As a result of the analysis of this project new risks and bugs were not defined. Bugs that are repeated will be considered in statistics.

Consider the next project: "Task management system for IT-company with communication module".

System goal: to obtain tasks from the project management, distribute tasks between team members. Possibility to organize meetings, and estimate characteristics of performed tasks (time. Software quality and other parameters). System has a statistics module.

The next project from GitHub about this topic was investigated [13]. Link to the video is represented in [14].

As a result of analysis of this project the next bugs were defined:

- risk2 bug 2, namely *"Interface errors"*. Some elements do not re-drawn correctly when interfaces are changed;
- risk2 bug 3, namely *"REST errors"*. Error of database updating after new information added or changed.
- risk4, namely *"User story risk"*. Sometimes there is an unexpected closing of program.

In such a manner the other projects for different topics were considered.

## 5 Summarizing of experimental research

Table 1 shows a classification of risks and defined bugs for considered projects. Numeric values of vulnerabilities of the risks and bugs are estimated by experts from different software developers' teams. In order to estimate maximum values of risks (in sentences from previous chapter: "such a risk may be estimated from…") and bugs, from three to five developer teams were involved. Resulting maximum values for bugs are represented in the previous chapter. Table 1 contains concrete values that are based on estimation of defined bugs for described projects.

Due to lack of place in the paper, only unique bugs and risks were described and recorded to video clips. If some risks and bugs were repeated more than one time, they were not being commented and were represented only in the table,

Figure 2 represents the summary of research for different projects. It shows results for fifteen projects with multilayer architecture, some of which were represented in the Table 1.

**Table 1.** Estimations of reuse risks for different projects

| Type of risk and its code | Bugs for this risk and their codes | Names of the projects | | | | |
|---|---|---|---|---|---|---|
| | | Weak eyed people | CRUD Library | Game | Medical system | Bug tracing system |
| R1 Unclear requirement specification | R1_B1 Non correct representation of requirements | 1 | 0 | 2 | 8 | 4 |
| R2 Incorrect input processing | R2_B1 Incorrect data range | 3 | 1 | 1 | 5 | 0 |
| | R2_B2 interface errors | 3 | 2 | 0 | 4 | 3 |
| | R2_B3 REST Errors | 4 | 1 | 0 | 5 | 3 |
| R3 Resource missing | R3_B1 missing database access | 0 | 3 | 0 | 7 | 10 |
| | R3_B2 missing third-party service | 4 | 4 | 0 | 2 | 0 |
| R4 User story error | R4_B1 User is confused what to do | 3 | 1 | 4 | 5 | 1 |

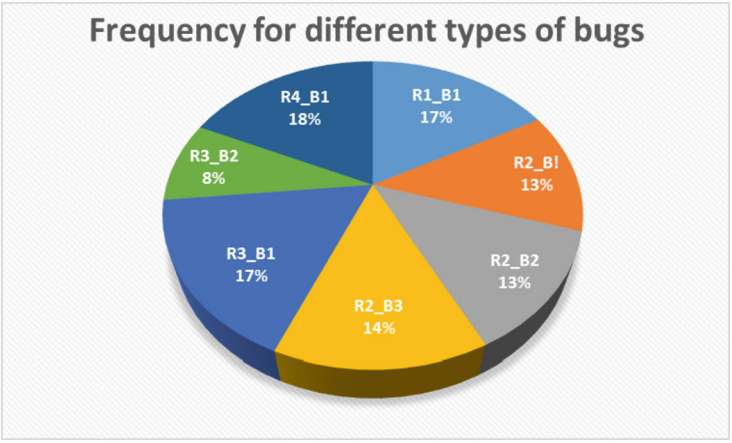



**Fig. 2.** Summary of bugs for GitHub projects with multilayer architecture.

## 6 Model for estimating of reuse risks

Every risk is estimated as a sum of values for all its bugs. Denote $r_i$ as the current risk.

$$r_i = \sum_{j=1}^{n} b_j \tag{2}$$

where $b_j$ - value of bug with number $j$ for current risk.

Maximum risk value for project is estimated by the following way:

$$R = \sum_{i=1}^{n} r_j \tag{3}$$

Then the next question becomes actual – "What is the max value of risk from the range $0 \leq R \leq R_{\max}$, where $R_{\max}$ is the value when all values of bugs are maximum?".

In order to answer to this question, the same teams of developers have performed the experimental researches. Their tasks were to realize software systems that correspond to system goals descripted for different projects (see chapter 3). Before realization of these projects, the value R for every project was estimated.

The experiment gives the basis for the next conclusion: if the value R for current project is inside of the range $0 \leq R \leq 0,25R_{\max}$ reuse activities may be performed successfully. One more recommendation was defined from reuse practices that usually successful reuse is done when a new project is developed on the base of one existing project. Link to the video for medical system project that was successfully realized by means of adding functionality is represented in [15].

***Conclusion from this chapter.*** Represented experimental research should help to software teams to understand the background of estimation of reuse risk for unknown projects. Some of risks were analyzed in this chapter. Centrally, other types of risk and bugs, which were not considered here due to of lack of place of the paper, may be added.

It is recommended to use statistical information, represented on Fig. 1, and start to investigate projects from risks that are met more often. When researcher will come.

to the limit $0,25R_{\max}$, the estimation of a project reuse risks should stop.

## 7 Comparison of peculiarities of reuse activities in research labs and GitHub

The aim of this chapter is to investigate peculiarities of activities performed in research laboratories of different companies and GitHub reuse activities. It will represent a background for proposing an approach for estimation of reuse risks for software projects from GitHub. Table 2 summarizes defined peculiarities.

**Table 2.** Peculiarities of reuse activities in research laboratories and GitHub repositories.

| Approaches that are implemented by research labs when software is prepared for reuse | Approach considering specific of GitHub |
|---|---|
| Initial data | |
| More or less structured information about source code from previous projects | GitHub system for searching of information about project using GitHub meta-attributes |
| Reuse conditions and peculiarities | |
| Employers work in the same technology stack (or inheritance technology stack)<br>Company follows code quality standards, design patterns, standards for naming of variables, methods, and other approaches [16]<br>Re-user may obtain real information about software (for example through investigating of bug-tracking systems or getting information from company chats) and consider projects details<br>Components for reuse were tested while previous projects were developed [17]<br>Previously developed services are integrated with Infrastructure of the company<br>Projects are implemented following Software Product Line approach [16] | Similar functionality may appear in projects that are realized in different stacks of technologies<br>Re-user may not understand approaches for designing code, comments, interface components and other projects' elements<br>Some of projects may be not finished and functionality may be realized partially<br>Some of projects may depend upon external sources, for example databases or cloud services that are located on customer resources<br>Licensing policies may limit a reuse<br>Sometimes it is no possible to adopt architectural solutions for reuse, for example, when it is necessary to reuse several projects simultaneously |

## 8 Software Reuse Approach Based on Review and Analysis of Reuse Risks From Projects Uploaded to GitHub

The experimental results and the analysis of peculiarities of the activities, performed both in research laboratories and in GitHub, provide a background for designing of the new software reuse approach, allowing to estimate the reuse risks for projects that were uploaded to GitHub. The approach is designed on the base of activities that are performed in research laboratories of software companies. These activities were taken as a basis, because they provide the background to perform reuse operations on higher levels of maturity, as repositories of companies store not only software, but UML diagrams, documentation, and other software development artifacts, allowing to get knowledge about their semantic. Reuse activities are represented in the Table 3.

How to analyze this table?

Activities, mentioned in the middle, are common for both types of reuse.

Activities in left columns are realized in research laboratories. If effective performance of such activities is impossible for GitHub, specific steps for GitHub are represented in the right column. When activities from middle and right column will be read together the reader will receive all steps of the proposed approach.

**Table 3.** Software reuse approach based on review and analysis of reuse risks from projects uploaded to GitHub

| | |
|---|---|
| 1, Compose of requirement specification | |
| 2. Select existing projects with the same functionality | |
| 3. Obtain an information about functionality of existing working projects and corresponding documentation (textual description, UML diagrams, licensing policy, screenshots etc.) | |
| 3.1.1 Explore bug tracking systems and (or) task management system (Jira or others) for this projects<br>3.1.2 Get information about frameworks, skipped and improved bugs, software architecture. Unit tests etc | 3.1.1 Avoid breaking licensing policies<br>3.1.2 Explore review of other users |
| 4 Take decision about possibility of reusing of this code in new project [18] | |
| 4.1 Investigate the functionality of previous realized systems<br>4.2 Get information about users' feedback and maintenance activities<br>4.3 Find view models (front-end elements), unit test and other software development artifacts from other projects for reuse<br>4.4 Get information about statistic of their reuse | 4.1 Perform black-box testing and calculate of value R using (2) and (3) for current project<br>4.2 Estimate common reuse risk |
| 5. Analyze software architecture<br>5.1 Verify whether architecture follows SOLID design principles<br>5.2 Investigate level of dependencies between components<br>6. Perform other activities of software development lifecycle processes to Implement source code with new functionality | |
| | 6.1 Improve the source code of the project, using static code analysis approaches |

## 9 Conclusions

The paper proposes an approach of software projects reuse considering peculiarities of GitHub. Actuality of this approach is approved by fact that GitHub is used by most of software development companies around the World. The proposed approach is based on practices of companies that are performed in research laboratories of great companies (for example IBM, Motorola, Hewlett Packard and many others) [18]. Peculiarity of the proposed approach is that it considers additional steps, allowing to manage reuse risks for unfamiliar projects. These reuse risks were gathered experimentally during analysis of GitHub projects' bugs for projects with multilayer architecture.

Estimation of reuse risks is based on the risk assessment model. Proposed model is flexible and may be extended and reused for different types of projects.

The proposed approach and the model allow to provide a background for raising of maturity level for reuse process, considering peculiarities of reuse activities for projects

downloaded from GitHub. The higher maturity levels (fourth and fifth) expect analysis before performing activities, taking conclusions after activities, analyzing mistakes, and considering them in further operations. Performing reuse processes on high maturity level allows to save efforts and resources as well as to avoid situations when reusing may create more problems than advantages.

## 10 Further Research

It is planned to design the approach of different types of software development artifacts reusing (interface elements, 3D models, software services, and data structures) for multilayer applications.

In order to organize reuse activities it is planned to consider reuse procedures according to layers of multilayer architecture.

For UI and presentation layers, user-friendly and intuitive interfaces may be designed considering factors that influence to human cognitive and emotional abilities [19]. Such interfaces may reduce the next risks: R4 User story error and R2_B2 interface errors.

For business logic layer, collaboration of software services on different cloud platforms (for example Google Cloud Platform and Microsoft Azure) allows "to construct" applications that may help to solve actual tasks from the digital society development, for example, economics or public policy and public administration [20]. Other application of reusing services is education, namely implementing new visualization techniques [21] or development of new algorithms [22]. Reuse of services from reliable cloud platforms may reduce the next risks: R3_B2 missing third-party service.

## References

1. Li, D., Tian, P.: Early prediction method of software reliability based on reuse analysis. In 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Con-trol Conference (IAEAC), vol. 1, pp. 545–550. IEEE (2019)
2. Mohammadi, N., Goeke, L., Heisel, M., Surridge, M.: Systematic risk assessment of cloud computing systems using a combined model-based approach. In: Proceedings of the 22nd International Conference on Enterprise Information Systems ICEIS, vol. 2, pp. 53–66 (2020). ISBN: 978-989-758-423-7
3. Orrego, A., Menzies, T., El-Rawas, O.: On the relative merits of software reuse. In: Wang, Q., Garousi, V., Madachy, R., Pfahl, D. (eds) Trustworthy Software Development Processes. ICSP 2009. Lecture Notes in Computer Science, vol. 5543, pp. 186-197. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01680-6_18
4. Cox, R.: Surviving software dependencies: software reuse is finally here but comes with risks. Queue **17**(2), 24–47 (2019)
5. NebbAirline GitHub repository https://github.com/Talevska/NebbAirline. Accessed 07 May 2023
6. Lab1_AM https://youtu.be/x6I3drPgM6I. Accessed 07 May 2023
7. Book library repository https://github.com/BnSalahFahmi/book-store. Accessed 07 May 2023
8. Video with bugs of library project https://drive.google.com/drive/folders/1rdHzulQE2UJ4-V6oxS49X9MleD0G2YZC?usp=share_link. Accessed 07 May 2023
9. English study game. https://drive.google.com/file/d/1LhHfuJzhHJ1OWSJxFjwkzdnXcYDJrQ8g/view. Accessed 07 May 2023

10. Game video. https://drive.google.com/drive/folders/1YWSSWEw3wiiq-EY2QbNbXfYQnuPiFY7z?usp=share_link. Accessed 07 May 2023
11. System for rehabilitation. https://github.com/heshanera/HealthPlus. Accessed 07 May 2023
12. Bugs for system HealphPlus. https://drive.google.com/drive/folders/1y8XCFSLoUjpa6e0nRJtu5jl_hO4hIuXc?usp=share_link. Accessed 07 May 2023
13. Bugtacker. https://github.com/connorleee/BugTracker. Accessed 07 May 2023
14. Video for bug-tracking project. https://drive.google.com/drive/folders/1f6fBZy73-crsofdliyFKKdHg68ZVwrqP?usp=share_link. Accessed 07 May 2023
15. Videos of improvement medical system.https://drive.google.com/drive/folders/1Y0DjzihFo5qmJwAy5toYxIDWj5ARtMSQ?usp=share_link. Accessed 07 May 2023
16. Rajakumari, K.E.: Towards a novel conceptual framework for analyzing code clones to assist in software development and software reuse. In: 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 105–111. IEEE (2020)
17. Griss, M.L.: Software reuse: from library to factory. IBM Syst. J. **32**(4), 548-566 (1993). https://doi.org/10.1147/sj.324.0548
18. Dabhade, M., Shivam, S., Manjula, R.: A systematic review of software reuse using domain engineering paradigms. In: Online International Conference on Green Engineering and Technologies (IC-GET). IEEE (2016)
19. Yakovytska, L., Lych, O., Horskyi, O., Khokhlina, O.: Psychological features of emotional stability as a safety factor of air traffic specialists. Transp. Res. Proc. **63**, 294–302 (2022). https://doi.org/10.1016/j.trpro.2022.06.016
20. Semenchenko, A., Gurkovskyi, V., Romanenko, Y., Sydorenko, V., Kudrenko, S., Polozhentsev, A.: Ukraine on the road to the european digital market: status and tools for implementing the European digital economy and society index in Ukraine. In: 1st International Workshop on Social Communication and Information Activity in Digital Humanities SCIA-2022, October 20, Lviv, Ukraine, pp. 186–197 (2022)
21. Mavrevski, R., Traykov, M., Trenchev, I.: Interactive approach to learning of sorting algorithms. Int.J. Online Biomed. Eng. (iJOE) **15**(08), 120-133 (2019). https://doi.org/10.3991/ijoe.v15i08.10530
22. Mavrevski, R., Traykov, M.: Visualization software for Hydrophobic-polar protein folding model. Sci. Vis. **11**(1), 11–19 (2019). https://doi.org/10.26583/sv.11.1.02