ISSN 2664-9969

8. Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T. et al. (2013). Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms. *IEEE Robotics & Automation Magazine*. Available at: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=09db526ea45bf5829f049f69100eb86322fe44fb
9. Price, I. C. (2006). *Evolving Self-Organized Behavior for Homogeneous and Heterogeneous UAV or UCAV Swarms*. USAF. Air Force Institute of Technology. Available at: https://scholar.afit.edu/cgi/viewcontent.cgi?article=4466&context=etd
10. Albrekht, Y., Pysarenko, A. (2023). Unknown location targets searching system in known environment using reinforcement learning. *Adaptive systems of automatic control, 1 (42)*, 9–14. doi: https://doi.org/10.20535/1560-8956.42.2023.278920

✉ ***Yosyp Albrekht,*** *Postgraduate Student, Department of Information Systems and Technologies, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, e-mail: yosyp.albrekht@gmail.com, ORCID: https://orcid.org/0000-0003-0093-6397*

Andrii Pysarenko, *PhD, Associate Professor, Department of Information Systems and Technologies, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, ORCID: https://orcid.org/0000-0001-7947-218X*

✉ *Corresponding author*

UDC 004.02
DOI: 10.15587/2519-4798.2023.293067

**Kostyantyn Kharchenko,
Oleksandr Beznosyk,
Bogdan Bulakh,
Ganna Ishchenko,
Vadym Yaremenko**

THE DEVELOPMENT OF THE METHOD OF OPTIMIZING COSTS FOR SOFTWARE TESTING IN THE AGILE MODEL

The object of research in the article is the process of testing and operating software with cost minimization. In the Software Development Life Cycle, depending on the chosen option of the flexible methodology, special attention is focused on testing software versions both in the process of passing iterations and in the process of releasing alpha, beta and production versions.

This article is devoted to the problem of developing a method for software testing cost optimization method that estimates the test cost function and the losses cost function from the occurrence of an error.

Using the optimization method (for example, the first-order descent method) from the two functions of testing costs and estimating the losses caused during operation, it is possible to calculate the optimal cost of testing and operating the software product.

The results obtained show that with the correct assessment of a cost function and a loss function such calculations allow to significantly save money and time for the production of the next version of the software product.

These results are explained by the fact that the method of optimizing the cost function finds the optimum point and allows to pre-estimate the budget and risks during the development and operation of the software.

The article provides several examples of the calculation and optimization of testing costs within the proposed concept for one iteration in a flexible software development cycle.

The results of the study can be used in practice, provided that the functions of estimating costs for testing and compensation for losses caused during the operation of the software are set correctly. Experienced managers and project supervisors determine these functions quite accurately for a certain number of iterations, which makes it possible to apply the method of finding the minimum budget costs for testing and operating a software product.

Keywords: *agile, SCRUM, software development life cycle, testing, QA, risk management.*

Received date: 18.10.2023
Accepted date: 12.12.2023
Published date: 15.12.2023

How to cite
Kharchenko, K., Beznosyk, O., Bulakh, B., Ishchenko, G., Yaremenko, V. (2023). The development of the method of optimizing costs for software testing in the Agile model. Technology Audit and Production Reserves, 6 (2 (74)), 10–14. doi: https://doi.org/10.15587/2519-4798.2023.293067

1. Introduction

The constant development of IT and programming methodologies requires new methods of planning and forecasting the quality of the resulting software product and information system.

One of the key aspects of software development is testing. In flexible methodologies, such as Agile, software testing stages play an important role, which directly affect the quality of the proposed solution and, accordingly, the cost of operating the information system [1–3].

Considering the importance and complexity of the testing process, it is necessary to have a certain strategy for minimizing costs both directly for the testing stage itself and for risk compensation [4–6]. This should take place in the context of an iterative approach and Agile methodology in order to achieve an optimal balance between testing costs and operating costs and covering possible losses associated with downtime (or incorrect operation) of the information system, loss of the company's image, moral damages, etc. [7].

In this context, the authors try to solve the actual problem of optimization of software testing costs by proposing an appropriate calculation method based on the evaluation of the function of testing costs and the function of the cost of damages that may occur during the operation of the information system due to detected errors.

The aim of the research is to apply optimization methods to identify the best-case scenario for allocating funds between testing expenses and covering potential risks during the operation of software. Relevant functions have been developed to assess these costs, and a solution to the posed problem has been developed using the Python language and mathematical libraries.

From a practical standpoint, this approach can be utilized in both large and small projects. It is expected that the application of such an approach will result in significant cost savings during the operation of software, while, on the other hand, enabling the sensible expenditure of the budget on software testing.

2. Materials and Methods

Let's consider an approach to optimizing the costs of testing and operating a software product.

Let's suppose that the function P of software testing cost estimation has the form of a piecewise linear function and depends on the number of tests performed.

Let's suppose that the function Q of estimating the costs of compensation for damages caused by errors in the software operation has the form of a piecewise linear function and also depends on the number of software tests.

To begin *w=sith*, consider an example when P and Q depend on only one parameter. Let's Suppose that one version of the software is tested for x iterations. Then the performance of this test can be described by the function $f(x)$.

Let's suppose that the cost estimation function for software testing has the form:

$$P=f(x), \quad (1)$$

where P is the cost of testing, x is the number of test iterations.

Assuming that the cost estimation function for compensating losses from errors in the software operation has the form:

$$Q=g(x), \quad (2)$$

where Q are expenses for compensation of losses during the operation of the software product for a separate part of the functionality, x is the number of test iterations.

Then the total cost of testing, taking into account compensation for damages for software errors, is estimated as:

$$R=P+Q, \quad (3)$$

or

$$R=f(x)+g(x). \quad (4)$$

Let's set the test cost function as piecewise linear (Fig. 1). Let's plot the number of testing iterations on the x-axis, and the cost of testing on the y-axis. Let's also set the cost function for compensation of losses during the operation of the software product as piecewise linear (Fig. 2). On the x-axis, let's set aside the number of testing iterations, and on the y-axis the cost of compensation for losses during the operation of the software product (for example, loss of revenue from customers during system unavailability). Such a function can be empirically estimated due to previous operating data.

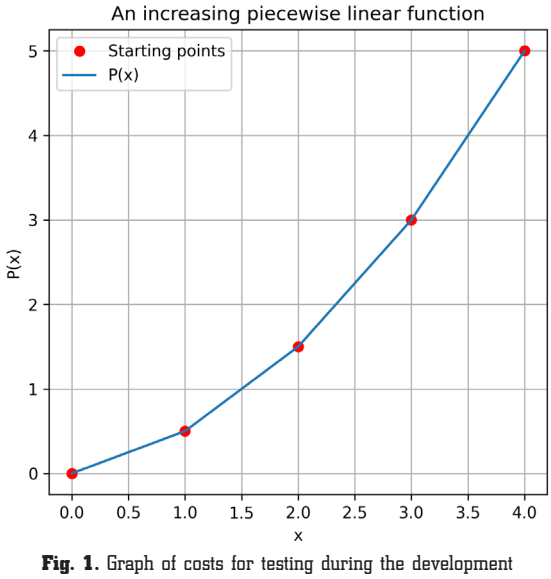

Fig. 1. Graph of costs for testing during the development of a software product

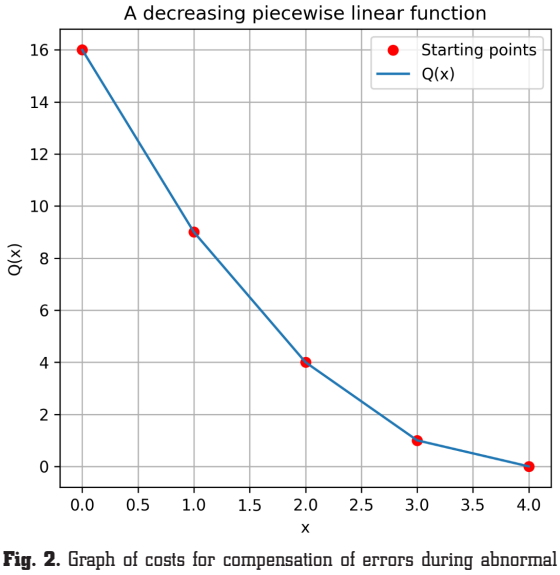

Fig. 2. Graph of costs for compensation of errors during abnormal operation of the software product

First, the Python code defines two data sets representing x and y coordinates for two different functions [8]: $x1$ and $y1$ are coordinates for a piecewise linear increasing function, and $x2$ and $y2$ – for a decreasing one, where $x1$, $y1$ define the function Q, and $x2$, $y2$ define the function R. Then the points and the line connecting them are constructed

ISSN 2664-9969

for the first data set ($x1$, $y1$). A red color is used for the points and a descriptive legend is added. The construction of the second graph takes place similarly. Then *plt.show()* displays the final figure with two graphs. Each of these graphs shows piecewise linear functions of P and Q.

```
import numpy as np
import matplotlib.pyplot as plt

# Increasing piecewise linear function
x1=np.array([0, 1, 2, 3, 4])
y1=np.array([0, 0.5, 1.5, 3, 5])

# Decreasing piecewise linear function
x2=np.array([0, 1, 2, 3, 4])
y2=np.array([16, 9, 4, 1, 0])

# Display graphs
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.plot(x1, y1, 'ro', label='Starting points')
plt.plot(x1, y1, label='P(x)')
# plt.plot(x1_interp, y1_interp, label='P(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.title('A increasing piecewise linear function')
plt.legend()
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(x2, y2, 'ro', label='Starting points')
plt.plot(x2, y2, label='Q(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.title('A decreasing piecewise linear function')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

Now let's search for the minimum of the function $R=f(x)+g(x)$ using the optimization method:

$$R(x)\to\min,\ x\in X, \tag{5}$$

where X is an admissible set, each point x of this set is an admissible point of the problem [9].

The value of $x_{\min}$ in this case will correspond to the minimum costs for testing and compensation for losses during the operation of the software product, and the possible total costs for testing and operation together will be $R(x_{\min})$.

This Python code demonstrates the optimization process, namely finding the minimum for the sum of two piecewise linear functions $R=P+Q$. The *numpy*, *matplotlib*, and *scipy* libraries are used for visualization and optimization. First, let's create an objective function. The *objective(x)* function determines the sum of two piecewise linear functions. This is achieved using the *np.interp* function, which is used to perform a linear interpolation of each function, and the results are summed. The *minimize_scalar* function from *scipy.optimize* is used to search for the minimum of the sum of two functions [10]. This function automatically selects an optimization method and looks for the value of x at which *objective(x)* reaches its minimum value. To illustrate the results, a graph of the total function and the minimum point is constructed. Finally, *plt.show()* is called to display the final plot showing the function and the minimum point found.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize_scalar

# Increasing piecewise linear function
x1=np.array([0, 1, 2, 3, 4])
y1=np.array([0, 0.5, 1.5, 3, 5])

# Decreasing piecewise linear function
x2=np.array([0, 1, 2, 3, 4])
y2=np.array([16, 9, 4, 1, 0])

# Function, the sum of two piecewise linear
functions
def objective(x):
y1_interp=np.interp(x, x1, y1)
y2_interp=np.interp(x, x2, y2)
return y1_interp+y2_interp

# We find the minimum point using the
optimization algorithm
result=minimize_scalar(objective)

# Output the result
if result.success:
print("Minimum point: x =", result.x)
print("The value of the function at the minimum
point: f(x) =", result.fun)
else:
print("The optimization algorithm could not find
the minimum point.")

# Graph of the function
x_interp=np.linspace(x1.min(), x1.max(), num=100)
y_interp=objective(x_interp)

plt.plot(x_interp, y_interp, label='Function R(x)')
plt.scatter(result.x, result.fun, color='red',
label='Minimum point')
plt.xlabel('x')
plt.ylabel('R(x)')
plt.title('Graph of the function')
plt.legend()
plt.grid(True)
plt.show()
```

3. Results and Discussion

In our case, the minimum value of the *R(x)* function, which is found with a small error using the *numpy.minimize_scalar()* function, equals to 3.0000000103012523 (the calculation considers the calculation error), while the total costs for testing and operation will be 4.000000010301252. The minimum of the function is shown on the graph in Fig. 3.

Moreover, approximately 3 units of the budget were spent on testing and 1 unit of the budget was spent on compensation for losses during operation.

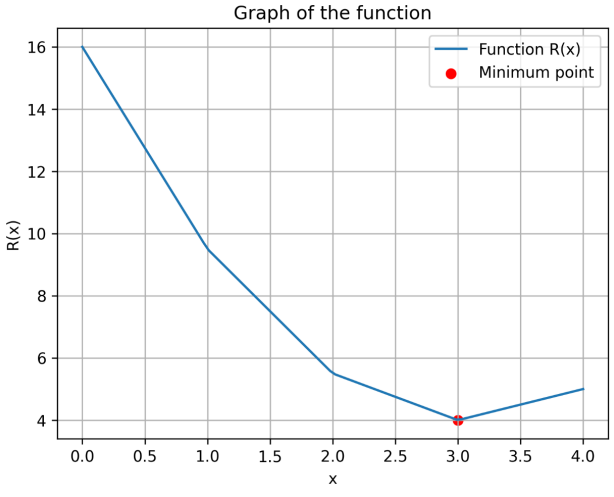

Fig. 3. Graph of the total testing cost and operation of the software product

In this case, when the number of tests increases ($x > 3$), the budget for general operation also increases, but when the number of tests decreases, the budget for general operation also increases, because the cost of compensation for operational risks increases.

In the Agile methodology, when developing software, the project is executed in iterations. Then, at each iteration, before starting work, it is possible to estimate the function P of software testing costs and the function Q of costs to compensate for losses due to errors in the software operation process.

During each iteration, the project manager and system architect, along with a team of testers, can conduct peer review and set parameters for P and Q functions.

In further work, the authors propose to develop a method of forecasting the next values of the R function depending on the accuracy of previous estimates and the amount of work performed on the software of the information system. It is assumed that at each cycle of the iterative methodology, it is possible to refine the dependence graphs P and Q and build a prediction function using, for example, the extrapolation method, the autoregressive and moving average (ARMA) model, or neural networks.

The estimated cost Q of software downtime losses can be quite significant if to consider losses from services such as, say, public transport ticketing, where an hour of downtime can lead to millions in losses.

A significant drawback of the proposed method is the influence of the human factor on the assessment of the scope of work and risks for the construction of P and Q dependencies, but in the end, this method provides a stable assessment metric and the ability to analyze how the calculated values after optimization correspond to real data. Particular attention should be paid to the construction of the Q function, that is, the compensation of losses during the operation of the software product on the number of test iterations. Although experienced project managers quickly master such an assessment and can apply the method in practice.

Among the disadvantages of the method, it is also worth noting that in certain critical information systems, software failure is not allowed in any case (medical information systems, strategic object management systems, etc.) and then the proposed method will not work, since P will always be several orders of magnitude higher than Q.

The limitations of the proposed method are that the optimization of the R function may depend on the form of this function, which can lead to falling into a local minimum when a better solution exists. It is advisable to review the graph of the R function and the calculated optimal value when making a decision, which will allow for a more careful analysis of the optimization results.

If the actual data of the evaluation of functions in the process of execution of iterations is significantly underestimated, then the project managers can determine this and make corrections for risk assessments in the next iterations, which as a result will give a useful effect in the process of execution of the entire project.

The impact of war on research. The authors claim that there was no influence of the war while conducting this research.

4. Conclusions

The article highlights the actual problem of cost optimization for software testing in the context of Agile methodologies with iterative development models. The authors propose a mathematical method for minimizing the total costs of testing and compensation for losses during the operation of the software product. The method is based on the estimation of two main functions – the function of testing costs (P) and the function of the cost of damage caused by errors (Q).

Optimization methods such as first-order descent or Newton's method can be used to find the optimal balance between these two types of costs. The experimental data presented in the article confirm the effectiveness of the proposed method.

It is important that with the correct estimation of the cost functions, the optimal value will be found. In several Agile iterations, the estimation of cost functions can be selected more accurately.

The article pays special attention to the calculation of the total cost of testing and operating the software product. In the proposed example, based on the optimization results, the authors find that the optimal number of tests corresponds to approximately 3 units of the budget, while the compensation of operating costs requires approximately 1 unit of the budget.

These data confirm the importance of a comprehensive approach to cost optimization in the software development process and demonstrate how to balance costs to minimize risks during the operation of the software product.

This method is proposed for use in the field of software testing and quality management in Agile environments. The method can be used both in scientific research and in practical activities to reduce costs and increase the efficiency of software development projects.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this study, including financial, personal, authorship, or any other, that could affect the study and its results presented in this article.

Financing

The study was conducted without financial support.

ISSN 2664-9969

Availability of data

The manuscript has no associated data.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the presented work.

References

1. Sadiq, Mohd., Khalid Imam Rahmani, Mohd. Wazih Ahmad, Jung, S. (2010). Software risk assessment and evaluation process (SRAEP) using model based approach. *2010 International Conference on Networking and Information Technology*. Manilam, 171–177. doi: https://doi.org/10.1109/icnit.2010.5508535
2. Mohamud Sharif, A., Basri, S.; Zain, J. M., Wan Mohd, W. M. B., El-Qawasmeh, E. (Eds.) (2011). Software Risk Assessment: A Review on Small and Medium Software Projects. *Communications in Computer and Information Science*. Berlin, Heidelberg: Springer, 214–224. doi: https://doi.org/10.1007/978-3-642-22191-0_19
3. McGraw, G. (2004). *Risk analysis in software design*. Available at: https://www.synopsys.com/blogs/software-security/software-risk-analysis/ Last accessed: 20.10.2023
4. Taylor, L., Shepherd, M. (2007). *Performing a System Risk Assessment*. FISMA Certification and Accreditation Handbook, 275–294. doi: https://doi.org/10.1016/b978-159749116-7/50022-6
5. Seniv, M. M., Roik, O. O. (2021). Means of calculating the reliability of software based on models, taking into account imperfect debugging. *Scientific Bulletin of UNFU, 31 (6)*, 87–91. doi: https://doi.org/10.36930/40310613
6. Yakovyna, V. S., Fedasiuk, D. V., Seniv, M. M., Nytrebych, O. O. (2015). *Modeli, metody ta zasoby analizu nadiinosti prohramnykh system*. Lviv: Vydavnytstvo Lvivskoi politekhniky, 220.
7. *Software Risk Analysis Tutorial: Comprehensive Guide With Best Practices*. Available at: https://www.lambdatest.com/learning-hub/software-risk-analysis Last accessed: 15.10.2023
8. *NumPy. The fundamental package for scientific computing with Python*. Available at: https://numpy.org/ Last accessed: 25.10.2023
9. Zhaldak, M. I., Tryus, Yu. V. (2005). *Osnovy teorii i metodiv optymizatsii*. Cherkasy: Brama-Ukraina, 608.
10. *Scypy. Fundamental algorithms for scientific computing in Python*. Available at: https://scipy.org/ Last accessed: 27.10.2023

Kostyantyn Kharchenko, *PhD, Associate Professor, Department of System Design, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, ORCID: https://orcid.org/0000-0002-7334-8038*

✉***Oleksandr Beznosyk,*** *PhD, Associate Professor, Department of System Design, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, ORCID: https://orcid.org/0000-0003-2775-6070, e-mail: beznosyk.oleksandr@lll.kpi.ua*

Bogdan Bulakh, *PhD, Associate Professor, Department of System Design, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, ORCID: https://orcid.org/0000-0001-5880-6101*

Ganna Ishchenko, *Senior Lecture, Department of System Design, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, ORCID: https://orcid.org/0000-0001-5086-5991*

Vadym Yaremenko, *Assistant, Department of System Design, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, ORCID: https://orcid.org/0000-0001-8557-6938*

✉*Corresponding author*