Modern Approaches to Software Optimization Methods

Igor Zhukov[1], Oleksii Synelnikov[1], Olena Chaikovska[2], and Serhiy Dorozhynskyi[1]

[1] National Aviation University, 1 Liubomyra Huzara ave., 03058, Kyiv, Ukraine
[2] Kyiv National University of Culture and Arts, 36 Y. Konovaltsya str, 01601, Kyiv, Ukraine

Abstract

Lately in the world some optimization methods have been developed surpassing traditional mathematical programming methods. These methods belong to the category of modern or unconventional optimization methods. Most of these methods are based on specific characteristics and behavior of the systems to which they are applied. This article provides a description of some of the common techniques used in many modern industries, not only in the traditional computing but also in everyday life areas. Most of these methods have been developed only in recent years and are becoming popular for solving complex engineering problems studying test object functions.

Keywords

Optimization approach, optimization methods, software optimization problem overview, architecture optimization.

1. Introduction

Dependence of software and equipment complexes using is steadily growing worldwide, which makes it more useful in all spheres of everyday life. Ukraine also observed persistent annual increase in the number of used software-hardware systems. In the conditions of the digital transformation for all spheres in Ukraine, this is the first from the key goals of the Ministry Education and Science, as it is the actual direction, not only because of the pandemic but also through global present day trends and national policies regarding the vector of digital transformation in whole country. Cost increasing and implementing systems complexity of the complexes themselves does not always allow to solve the tasks, since even the slightest specification inaccuracies or shortcomings allowed during the development process can lead to a decrease in the software performance. Thus, software optimization requires the most optimal approaches at all development stages even on testing and product implementation. Resources using is important criteria for the software complexes effectiveness [1].

A lot of researchers mentioned about impossibility to apply single formulation procedure for all engineering design problems, since the objective in a design modeling problem and associated therefore, design parameters vary product to product different techniques are used in different problems. In the meantime purpose of formulation is to create a mathematical model of the optimal design problem, which then can be solved using an optimization algorithm. Figure 1 shows an outline of the steps usually involved in an optimal design formulation.

2. Optimization Model

Different point of view describes constructing optimization model process determining using steps for it.

CPITS-II-2021: Cybersecurity Providing in Information and Telecommunication Systems, October 26, 2021, Kyiv, Ukraine
EMAIL: zhukov.ihor@sfk.nau.edu.ua (I. Zhukov); oleksii.synelnikov@npp.nau.edu.ua (O. Synelnikov); dorozhun1706@gmail.com (S. Dorozhynskyi); lena@knukim.edu.ua (O. Chaikovska)
ORCID: 0000-0002-9785-0233 (I. Zhukov); 0000-0002-4073-2594 (O. Synelnikov); 0000-0002-5395-6423 (S. Dorozhynskyi); 0000-0001-7769-1004 (O. Chaikovska)

The first step in the optimization process is constructing an appropriate model. Modeling is the process of identifying and expressing in mathematical terms the objective, the variables, and the constraints of the problem.

1. An objective is a quantitative measure of the performance of the system that we want to minimize or maximize. In manufacturing, we may want to maximize the profits or minimize the cost of production, whereas in fitting experimental data to a model, we may want to minimize the total deviation of the observed data from the predicted data.
2. The variables or the unknowns are the components of the system for which we want to find values. In manufacturing, the variables may be the amount of each resource consumed or the time spent on each activity, whereas in data fitting, the variables would be the parameters of the model.
3. The constraints are the functions that describe the relationships among the variables and that define the allowable values for the variables. In manufacturing, the amount of a resource consumed cannot exceed the available amount.

Having all parameters considered from different sides we tend to draw some best-practice steps in the optimization process classified for optimization model, since algorithms for solving optimization problems are tailored to a particular type of problem. An example with basic information for the various optimization problem types list is as following [2,3].

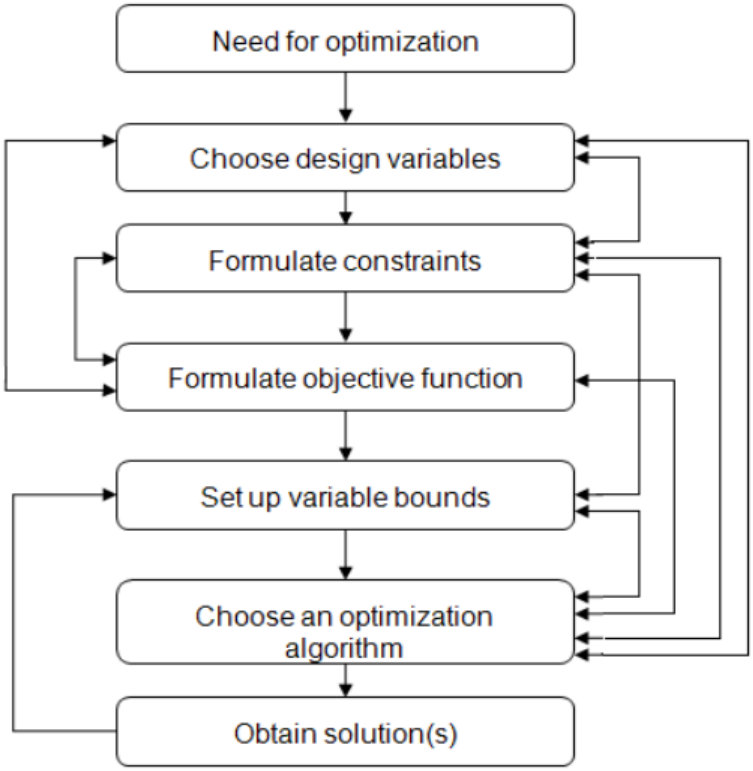

Figure 1: Solution diagram [4]

- Continuous Optimization versus Discrete Optimization. Some models only make sense if the variables take on values from a discrete set, often a subset of integers, whereas other models contain variables that can take on any real value. Continuous optimization problems tend to be easier to solve than discrete optimization problems; the smoothness of the functions means that the objective function and constraint function values at a point x can be used to deduce information about points in a neighborhood of x. However, improvements in algorithms coupled with advancements in computing technology have dramatically increased the size and complexity of discrete optimization problems that can be solved efficiently. Continuous optimization algorithms are important in discrete optimization because many discrete optimization algorithms generate a sequence of continuous subproblems.
- Unconstrained Optimization versus Constrained Optimization. Another important distinction is between problems in which there are no constraints on the variables and problems in which there are

constraints on the variables. Unconstrained optimization problems arise directly in many practical applications; they also arise in the reformulation of constrained optimization problems in which the constraints are replaced by a penalty term in the objective function. Constrained optimization problems arise from applications in which there are explicit constraints on the variables. The constraints on the variables can vary widely from simple bounds to systems of equalities and inequalities that model complex relationships among the variables. Constrained optimization problems can be furthered classified according to the nature of the constraints (e.g., linear, nonlinear, convex) and the smoothness of the functions (e.g., differentiable or non-differentiable).

- None, One or Many Objectives. Most optimization problems have a single objective function, however, there are interesting cases when optimization problems have no objective function or multiple objective functions. Feasibility problems are problems in which the goal is to find values for the variables that satisfy the constraints of a model with no particular objective to optimize. Complementarity problems are pervasive in engineering and economics. The goal is to find a solution that satisfies the complementarity conditions. Multi-objective optimization problems arise in many fields, such as engineering, economics, and logistics, when optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives. For example, developing a new component might involve minimizing weight while maximizing strength or choosing a portfolio might involve maximizing the expected return while minimizing the risk. In practice, problems with multiple objectives often are reformulated as single objective problems by either forming a weighted combination of the different objectives or by replacing some of the objectives by constraints.
- Deterministic Optimization versus Stochastic Optimization. In deterministic optimization, it is assumed that the data for the given problem are known accurately. However, for many actual problems, the data cannot be known accurately for a variety of reasons. The first reason is due to simple measurement error. The second and more fundamental reason is that some data represent information about the future (e. g., product demand or price for a future time period) and simply cannot be known with certainty. In optimization under uncertainty, or stochastic optimization, the uncertainty is incorporated into the model. Robust optimization techniques can be used when the parameters are known only within certain bounds; the goal is to find a solution that is feasible for all data and optimal in some sense. Stochastic optimization models take advantage of the fact that probability distributions governing the data are known or can be estimated; the goal is to find some policy that is feasible for all (or almost all) the possible data instances and optimizes the expected performance of the model [5,6].

There are some approaches improving problems solving quality with the allocation of relevant computing systems resources. Algorithm verification for solving the most computational task reduces its solution time, which also has a positive effect on performance as a whole. For selecting one or another approach, scientific research is proposed to make a decision on the basis of a possible type of variable model values, thus correlating optimization problems as discrete or continuous.

Continuous optimization problems concern the case when models variables can take any value allowed by some specified limitations. At the same time, the method of discrete optimization (combinatorial) uses combinational variables related to a discrete set (from integers subset) of values. Discrete optimization is aimed at problems associated with the optimal choice solution from the final possibilities number. In the discrete optimization, rely on the correct solution is possible only when in the selected solutions there is a deliberately correct result [7].

Heuristic approaches have an advantage in the form of a solution rate at the expense of approximation, but only in cases where the exact solution exists. These approaches show almost optimal solutions on a certain time, which gives an increase in the solution rate. Although purely heuristic optimization is designed to solve a specific task, metabolic can be applied to a wider range of tasks. They can be used to find an almost optimal solution for optimization problem based on the possible knowledge of calculation area.

3. Optimization Technique

Generally speaking, the optimization techniques in used approaches with specific context described in Table 1. However, this is a technically complex and almost resource-proof procedure, and those obtained in this way may not often be true and requiring additional time for a real solution. One of the

machine learning founder gives to world significant contribution to the development and theoretical analysis of concepts and technique such as multi-class classification, semi-supervised learning, latent factor models, and sequential models. Machine training is a technique that allows you to calculate systems for learning (ie, improved) from experience (available data). Programs for machine learning method are based on creating a prediction model from a set of preparatory data, which are subsequently used to create predicted data, and not after preliminary static instructions. You can select the most used machine learning algorithms for that: regression, solutions tree and artificial networks. Software optimization is possible only during the life cycle of the software itself. Optimization techniques can be classified as follows:

1. Message headers (System Memory Savings);
2. Variables initiation (assigning variable variables at the same time with their ad);
3. Reducing variables number (cleaning temporary variables after their use);
4. Select data types (streamlining the simultaneous use of different data types);
5. Removing unnecessary assignment operators (integration of assignment operators);
6. identifying variables (replacement of variables by another variable or numerical variable value);
7. Removing identical operators (may appear in the optimization process);
8. Elimination of non-fulfilled operators (which are not performed for any sets of initial data);
9. I/O using (excluding this type of operations due to their duration);
10. Selecting procedures (functions) (transformation sequence of identical operators in the procedure or function);
11. Reducing the number of procedures (functions) (allocation to a separate procedure of operators that run many times);
12. Alternatives (streamlining conditions-in order to descend their probability to be true);
13. Arithmetic operations (replacement of operations from different speeds);
14. Transformation of expressions (preliminary simplification of expressions);
15. Optimization of expressions (replacement of complex semantically equivalent subsections per simple);
16. Pre-calculation of arithmetic subsection (reducing the number of operations performed);
17. Elimination of unnecessary brackets (eliminate pairs of brackets in arithmetic or logical expressions);
18. Elimination of extra labels (removing labels in front of operators to which there is no control transmission);
19. Elimination of unnecessary transition operators (removal of unconditional transition operators with a non-existent label);
20. Implicit operations (simplifying operators with frequently used variables);
21. Cleaning cycles (reduces the cycle time by removing arithmetic expressions from its body, independent of the cycle control variable);
22. Using cycles (replacement not repeatedly repeated sequences of operators on successive);
23. Combining cycles (connecting multiple cycles with the same headlines);
24. Cycle loop branches removal (reduce validation checks number);
25. Deleting empty cycles (eliminating empty cycles saves the program execution time);
26. Cycles compression (body cycle logical transition operators removal);
27. Management selection (the location commands group in selection operator, descending the frequency using probability) [8,9].

However, in practice few are used.

Table 1
Optimization techniques [10]

Method	Features	Disadvantages
Heuristic	The practical ability to find new unexplored solutions	No guarantee for better solution Incorrect decision in some cases No guarantee for solution even in case his existing
Metaheuristic	Applicable in cases with complete absence studied object properties Allow find solutions without correctness proof	High computational using costs
Machine Learning	Workaround solution, learning due applying a plurality on a set of similar problems	Select optimal starting point

Another equally important part of software optimization is the right choice of architecture. The quality of the architectural solution evaluated by a set of criteria, and his choice is always a compromise because for given set of functional requirements and quality requirements there is no obvious one the best solution and improvement of some indicators leads to the deterioration of others and vice versa. And because here you need to take into account relationships between architecture quality criteria and PS, the initial task must be formulated as a multi-criteria hierarchical task optimization. The most widely used are two standard approaches to quality assessment architecture. The first approach is evaluation of its characteristics by the method simulation. For example, for performance simulation results estimated bandwidth and scalability of the system. In other cases, connectivity and adhesion characteristics used to assess ability to changes and convenience of system maintenance. In the methods of the second approach architecture characteristics are evaluated by the results of a survey of experts. For example, in the ATAM method (Architecture Trade-off Analysis Method) risks are assessed that the architecture does not satisfy conceptual requirements. Conceptual requirement of one of the interested parties experiment in the ATAM technique is described with using scripts. Then alternative architecture options are analyzed for the subject support for each of the scenarios. The method of analysis of Saati hierarchies allows get relative estimates of architectures as per to each criterion, and on their set. However, because MAI is an expert technology in essence, then obtained solutions may be unstable to inconsistencies of matrices of pairwise comparisons, and also to change the priorities of quality criteria. Changing the requirements for the aircraft leads to a change prioritize quality criteria as a system in general, and its architecture. To detect critical decisions and analysis sensitivity of decisions to changes in requirements is necessary perform analysis of conflicts and possible compromises between criteria. It will give opportunity to identify potential problems and ensure the adaptation of the architecture to future changes in requirements. Issues of sensitivity in methods architecture evaluation was the first delivered in. But they did not provide no solutions to this problem. This is partial due to what they used quantitative evaluation methods, such as ATAM. The approach proposed in the article is an additional analysis results of the method of analysis of hierarchies. As example of the study of the sensitivity of solutions to changing requirements and analyzing trade-offs between quality criteria were held for architectures taken from the GlasBox project [11,12].

One of the most important methods of optimizing the workflow in software development is the use of cluster techniques. The movement of flexible Agile techniques is revolutionary. Teams that use this technology systematically report improvements (sometimes abrupt) in the ability to create better Software. Those who have successfully implemented Agile create high-quality products and do so faster than before. Agile has turned from an outsider into a working institute. Currently, there is almost none left Doubts that agile methodologies are an effective way to create software. In 2008 it was conducted a study, which showed that more than half of all surveyed teams involved software development, use agile methodologies, practices or principles. Traditionally, companies have used software development projects Waterfall approach, according to which the team first determines the requirements for the

product, plans a project in in general, develops a software solution and then generates code and tests the product. Much of the software provision - both grand and very small - for years was created in this way. However, for decades, different teams in different companies have faced the same problems. And some of them suspected that the main cause of failure - the Waterfall approach itself. The use of Canban to date is this the most common method (and effective for many agile practitioners) is the introduction of frugal thinking organization [13].

When using flexible techniques are guided by 12 principles of flexible software development:

1. Our highest priority is customer satisfaction through frequent and continuous deliveries valuable software for him.

2. Changes in requirements are accepted even at later stages of project implementation. Agile processes allow use the changes to increase the competitiveness of the product.

3. The desire to supply fully working software every few weeks, at the very least case - every few months. The more often, the better.

4. The most effective and efficient way to transfer information is to meet the members of the software development team.

5. Business representatives and the development team should work on the project together.

6. Projects are built around motivated people. It is necessary to create a suitable environment for them environment, provide everything you need and entrust to do your job.

7. Working software is the main measure of project progress.

8. Flexible processes contribute to continuous development. Sponsors, developers and users should be able to maintain a constant pace of work indefinitely.

9. Constant attention to technical excellence and quality architecture promotes flexibility.

10. Simplicity represents for the art team not to do unnecessary work.

11. The best architecture, requirements and design are created in self-organized teams.

12. The team is constantly looking for ways to become more effective by adjusting and correcting their actions [14].

The first principle includes three important conditions: early release of the software product, continuous realization of value and customer satisfaction. To understand the essence of this principle, you need to know how these three ideas work together. Project teams exist in the real world, where nothing perfect happens. Even a brilliantly working team will be missing something when collecting and recording project requirements, because it is impossible to fully and accurately display the requirements for each system. That doesn't mean teams shouldn't try, but agile-methodologies are based on unprecedented methods of communication and fixation of requirements. But, until customers get their hands on working software, it's hard for them to imagine exactly how it will work [15,16].

In this case, if you want to receive from the client a real, reasonable return communication after viewing the working software product, it is best to do so through early delivery: "Ship" to the customer the first working version of the software as soon as possible. Even if implemented only one work function that the client can use - is to achieve some success for the team. Because now the consumer is able to provide feedback that will help move the project in the right direction. It is also a success for customers because by having this work program, they can realize that previously only planned. We can say that the team has created real value, because the software works and customers can use it. Let this be only a small part of what the consumer needs, but still it is better than waiting without software, especially if the alternative is to wait a long time before receiving software product. The disadvantage of early delivery is that the software is far from perfect. It is a challenge for some customers and stakeholders: if some users are used to thinking that see the software early, it is harder for others to get used to the idea. Many employees very much experience when their software is not perfect. In companies (especially large ones, where years are spent working with teams that develop software) these teams must agree very carefully on terms of software delivery to interested parties. In the absence of a partnership between the customer. The incoming software product supplied by the development team can be evaluated strictly and even terrify the user if it really doesn't live up to expectations. Basic agile values solve similar ones question: cooperation with the customer is more important than agreeing on the terms of the contract. A team tightly tied to specifications, cannot make changes to the software during its creation. Working in such conditions, a new process of total change management needs to be launched, which will require new negotiations under the contract with the customer. But the team that really works with customers has the opportunity to make all the necessary changes during the work [17].

That's what uninterrupted delivery means. That is why flexible methodologies are usually iterative. Agile commands plan iterations of the project by selecting indicators and requirements that ensure maximum return. Only the way in which the team can find out which indicators implement this value-cooperation with client and embedding feedback from a previous iteration. This allows the team to satisfy requests consumer in the short term, demonstrating the value of the product at an early stage cooperation, and in the long run to provide him with a finished product that has all possible valuable qualities [18].

The team uses iterations to break the project into pieces with regular delivery deadlines. During iterations team supplies the working software. At the end of each iteration, the team conducts a demonstration, showing the customer the created product as well as previous options to see what lessons can be learned from this situation. Then begin a planning session to find out what they will create next iteration. A predictable schedule and constant checkpoints help the team keep track of these changes early on stages and create an atmosphere in which it is not customary to look for the culprit when everyone can discuss change and agree on a strategy to include it in the project.

At this point, Agile becomes attractive to the traditional command-and-control manager project. Such a manager wants to control the term. Setting limits on the duration of iterations gives it this possibility. In addition, one of the main tasks of the manager is solved - to work with the changes that occur at the very end of the project.

One of the most difficult moments in the work of a traditional manager is the monitoring of changes. Daily reports and retrospectives of iterations allow the project manager to gain the support of all team members, because they are now the eyes and ears of the leader, helping to identify the need for change before they become the cause more serious problems in the project. The role of the project manager is to initiate the replacement of the command-and-control system, in which he needs to give tasks to team members on a daily basis and constantly adjust the work plan to direct them in the right direction. Now the manager interacts with the team just to make sure: each specialist sees a single picture and works on common goals. The easiest way to do this is when the team is running in short iteration modes that allow you to deliver work Software. This sets specific goals for each participant and gives him a comprehensive idea of what working team, as well as a sense of responsibility not only for their work, but also for the overall result after end of iteration [19].

Kanban is a method of improving the processes used by flexible teams. Teams that apply it, begin to understand how they create software, and gradually improve him. Kanban, like Scrum and XP, affects the way a person thinks. In particular, it requires thrift thinking. In turn, Lean is a way of thinking, values and principles. Teams using Kanban began with the application of the worldview of Lean. This provides a solid foundation, which in combination with Kanban gives opportunity to improve processes. When teams use Kanban to improve, they are focused to eliminate losses from processes. Kanban is a technological term adapted by David Anderson for software development.

Here is how he describes his relationship with Lean in his book Kanban. An alternative path in Agile. Kanban method is a complex adaptive system designed to activate lean solutions within organizations. There are teams that use Lean and Thrifty thinking to develop programs without using them kanban, but today it is the most common method (and effective for many agile practitioners) introduction of frugal thinking in the organization. Kanban differs from flexible methodologies such as Scrum and XP. Scrum is mainly focused on project management: the amount of work to be done, that this work was done, and the result needed by users and stakeholders. XP is focused on Software development, its values and practices are built around creating an enabling environment for developing and forming habits that help the developer write simple and easily modifiable code. Kanban helps the team improve software development methods. The team that uses this method, has a clear idea of what actions it does when creating a software product, how it interacts with another the company, how losses are caused by inefficiency and unevenness, and how over time rectify the situation by eliminating the root cause of these losses. When a team improves the way they create software, it is traditionally called the process of improvement. Kanban is a good example of the application of flexible ideas (such as a last resort) to create a process improvement method that teams can easily adopt [20].

CANBAN practices allow to stabilize and improve the software development system software. An up-to-date list of basic practices can be found in the Kanban Yahoo! First, it should adhere to the basic principles:

• Start with what you are doing now, keep existing roles, responsibilities and job descriptions.

• Agree on evolutionary development.
• Encourage leadership at all levels.

Then you should apply the basic practices: visualization; workflow limit (WIP); flow management; make the rules clear; enter feedback loops; develop together and experiment (using modeling or a scientific approach). At the initial stage, it is not necessary to implement all six practices. Partial implementation is recognized superficial and involves a gradual deepening, as most practices are accepted and performed with great potential. Kanban is not a software development methodology or a project management system. One of the most common mistakes at the beginning of studying CANBAN is to try to use it as methodology for software development.

Kanban is a way to improve processes. It will help to understand the technique used, and find ways to improve it. Kanban does not operate with processes, but with working elements. It's called that a stand-alone unit of work that needs to be tracked throughout the system. Typically, the requirements are designed for user history or a similar fragment of the general pool of works. There is a difference between a task board and a kanbandoshka: while tasks move across the task board, work items are not tasks. Task - this is what people do to move the working elements within the system, ie these are the "cogs" of the mechanism, which pushes the working elements [21, 22].

That is why you can use systems thinking to understand the workflow when creating software, without degrading the human dignity of developers by thinking that they are part of some cars. Tasks are "cogs," and people are unique individuals with their own character, desires and motivation. The columns on the kanban board may seem like stages in the value stream. However, many kanban experts share the concepts of "value systematization" and "kanbandoshka." They reflect on the board the state of the working elements in the workflow regardless of the flow values and call it a life cycle map. The difference is that the value creation stream is a tool careful thinking, helps to understand the operation of the system, and the life cycle map is like a kanban method identifies the specific stages that each working element goes through [23,24].

Teams continue to deliver functionality, they identify workflow issues and adjust the WIP limit so that the feedback loops provide a sufficient amount of information, no causing slips. Flow is the speed at which working elements move through the system. When the team finds the optimal pace for delivery combined with the convenient feedback it has maximizes flow. Reduction of uneven work and overloads, the ability to complete one task and only then move on to the next increases flow. When due to uneven work accumulate task, this interrupts the workflow and reduces the flow. The Kanban team uses flow management practices by measuring and doing active steps to improve it. You already know how inspiring it is when the work progresses. You feel that you have time to do a lot, do not waste time and do not wait for someone else to finish their part of the work and will give it to you. When you work in a team with a large flow, you constantly feel that you are doing something valuable. Everyone aspires to this. You also know how it is when work stalls. It feels like you're booted into mud and can barely move. It seems like you're always waiting for someone to finish creating what you need, or make a decision that affects work, agree on a card, or find something else that interferes with work - even if you know that no one is intentionally interfering. You feel inconsistency and inconsistency and spend a lot of time explaining why you are waiting. This does not mean that the team is not enough loaded, maybe people are laid out 100%, and maybe even overwhelmed. But while the plan of the project says that the work is done by 90%, it seems to you that, on the contrary, you still need to do 90% work. An effective tool for measuring flow - cumulative flow diagram (cumulative flow diagram, CFD). CFD is similar to a WIP chart, but has one important difference: instead of disappearing, working elements accumulate in the bottom bar of the chart. And if on the WIP-diagram strips correspond to states in the value creation flow chart, the CFD (and WIP chart) in this section has the bands corresponding to the columns on kanban board [25].

4. Conclussions

In general, most of mentioned methods give various improvements in productivity and speed, but there are a lot of issues that require improvements and improvements. Among the disadvantages can be allocated: requirement to use additional comments in the code, restriction on a specific programming language or model or ability to use only one of the optimization techniques.

There is exigency for software optimization that is applicable to a large spectrum of programs and systemic architecture. Many existing solutions are limited unfortunately to a specific programming language and model, application type or system architecture. Future efforts should focus on developing solutions for widely used general-purpose languages.

5. References

[1] Chantieva, M. E., et al., Software Optimization Methods for Composite Materials, 2019 Systems of Signals Generating and Processing in the Field of on Board Communications, 2019, pp. 1-4, doi: 10.1109/SOSG.2019.8706771.

[2] S. Shkarlet, Digital Transformation of Education and Science is one of the key objectives of MES for 2021, 2021. URL: https://mon.gov.ua/ua/news/cifrova-transformaciya-osviti-i-nauki-ye-odniyeyu-z-klyuchovih-cilej-mon-na-2021-rik-sergij-shkarlet.

[3] Interview with Ministry and Digital Transformation Committee of Ukraine, 2021. URL: https://thedigital.gov.ua/news/ogolosheno-tender-na-analiz-ta-rozrobku-metodichnikh-rekomendatsiy-shchodo-rozvitku-sistemi-e-poslug.

[4] Memorandum Mikhail Fedorov from Ministry and Digital Transformation Committee of Ukraine, 2021. URL: https://thedigital.gov.ua/news/ogolosheno-tender-na-analiz-ta-rozrobku-metodichnikh-rekomendatsiy-shchodo-rozvitku-sistemi-e-poslug.

[5] Florian Robic, Daniel Micallef, Simon Paul Borg, Brian Ellul, Implementation and fine-tuning of the Big Bang-Big Crunch optimisation method for use in passive building design, Building and Environment, 10.1016/j.buildenv.2020.106731, (106731), (2020).

[6] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek and I. Meedeniya, "Software Architecture Optimization Methods: A Systematic Literature Review," in IEEE Transactions on Software Engineering, vol. 39, no. 5, pp. 658-683, May 2013, doi: 10.1109/TSE.2012.64.

[7] ISO/IEC Standard for Systems and Software Engineering—Recommended Practice for Architectural Description of Software-Intensive Systems, Int'l Standards Organization, ISO/IEC 42010 IEEE Std 1471-2000, first ed. 2007-07-15, p. c1-24, 2007.

[8] Czyzyk, J., Mesnier, M. P., and Moré, J. J. 1998. The NEOS Server. IEEE Journal on Computational Science and Engineering 5(3), 68-75. This paper discusses the design and implementation of the NEOS Server.

[9] Dolan, E. 2001. The NEOS Server 4.0 Administrative Guide. Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory. This technical report, which discusses the implementation of the server and its use in detail, is available for download in PDF format.

[10] Gropp, W. and Moré, J. J. 1997. Optimization Environments and the NEOS Server. Approximation Theory and Optimization, M. D. Buhmann and A. Iserles, eds., Cambridge University Press, pages 167-182. This paper discusses the NEOS Server as a problem-solving environment that simplifies the formulation of optimization problems and the access to computational resources.

[11] R. Faia, J. Soares, T. Pinto, F. Lezama, Z. Vale and J. M. Corchado, "Optimal Model for Local Energy Community Scheduling Considering Peer to Peer Electricity Transactions," in IEEE Access, vol. 9, pp. 12420-12430, 2021, doi: 10.1109/ACCESS.2021.3051004.

[12] Chicco, G.; Mazza, A. Metaheuristic Optimization of Power and Energy Systems: Underlying Principles and Main Issues of the 'Rush to Heuristics'. Energies 2020, *13*, 5097. https://doi.org/10.3390/en13195097.

[13] Vale, Zita & Lee, Kwang. (2020). Applications of Modern Heuristic Optimization Methods in Power and Energy Systems. 10.1002/9781119602286.

[14] Rao, Singiresu. (2019). Modern Methods of Optimization. 10.1002/9781119454816.ch13.

[15] I.S. Lamprianidou, T.A. Papadopoulos, G.C. Kryonidis, E. Fatih Yetkin, K.D. Pippi, A.I. Chrysochos, Assessment of load and generation modelling on the quasi-static analysis of distribution networks, Sustainable Energy, Grids and Networks, 2021, 100509, ISSN 2352-4677, https://doi.org/10.1016/j.segan.2021.100509.

[16] G.N. Vanderplaats, Thirty years of modern structural optimization, Advances in Engineering Software, Volume 16, Issue 2, 1993, Pages 81-88, ISSN 0965-9978, https://doi.org/10.1016/0965-9978(93)90052-U.

[17] Wilson Trigueiro de Sousa Junior, José Arnaldo Barra Montevechi, Rafael de Carvalho Miranda, Afonso Teberga Campos, Discrete simulation-based optimization methods for industrial engineering problems: A systematic literature review, Computers & Industrial Engineering, Volume 128, 2019, Pages 526-540, ISSN 0360-8352, https://doi.org/10.1016/j.cie.2018.12.073.

[18] Leyffer, Sven & Mahajan, Ashutosh. (2011). Software For Nonlinearly Constrained Optimization. 10.1002/9780470400531.eorms0570.

[19] Memeti, S., Pllana, S., Binotto, A. et al. Using meta-heuristics and machine learning for software optimization of parallel computing systems: a systematic literature review. Computing 101, 893–936 (2019). https://doi.org/10.1007/s00607-018-0614-9.

[20] I.A. Farhat, M.E. El-Hawary, Optimization methods applied for solving the short-term hydrothermal coordination problem, Electric Power Systems Research, Volume 79, Issue 9, 2009, Pages 1308-1320, ISSN 0378-7796, https://doi.org/10.1016/j.epsr.2009.04.001.

[21] Anh-Tuan Nguyen, Sigrid Reiter, Philippe Rigo, A review on simulation-based optimization methods applied to building performance analysis, Applied Energy, Volume 113, 2014, Pages 1043-1058, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2013.08.061.

[22] Jennifer Davis and Ryn Daniels. 2016. Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale (1st. ed.). O'Reilly Media, Inc.

[23] Bogachuk, I., Sokolov, V., Buriachok, V., Monitoring subsystem for wireless systems based on miniature spectrum analyzers, in: 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (2018). https://doi.org/10.1109/infocommst. 2018.8632151.

[24] Kipchuk, F., et al. Investigation of Availability of Wireless Access Points based on Embedded Systems. 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 2019. https://doi.org/10.1109/picst47496.2019.9061551

[25] Zhukov I.A. Implementation of integral telecommunication environment for harmonized air traffic control with scalable flight display systems //Aviation. 2010, 14(4), 117–122.