**O. V. Prus; V. P. Maidaniuk, Cand. Sc. (Eng.), Associate Professor;**
**I. R. Arseniuk, Cand. Sc. (Eng.), Associate Professor**

# ANALYSIS OF TOOLS FOR MULTIPROJECT ENVIRONMENTS MANAGEMENT: OPTIMIZATION OF THE SOFTWARE DEVELOPMENT

*The paper considers the approach regarding the efficiency of multi-project environments management in the sphere of the software development. Main attention is paid to the detailed analysis and comparison of modern tools for monorepositories management, in particular, Lerna, Yarn Workspaces, Bazel, Rush, Pants, Bit and Nx. For the assessment of their functionality, performance, scalability. compatibility with different technological stacks and study of their impact on general efficiency of the application development, multicriterial analysis was used. Within the context of enhancing the efficiency of multi-project environment management in the sphere of the software development, where the resources contain, in particular, time and efforts of the programmers, technical infrastructure and financial facilities, when several project compete for these limited resources, there appears urgent need in complex decision-making regarding the priorities and distribution. That is why, the concept of applying complex multicriterial analysis for the assessment the tools for monorepositories management was put forward. Such approach enables to evaluate quantitively and compare different tools on the base of the previously determined criteria, using the utility formula. The work, dealing with the collection and assessment of the criteria data in the context of the software development in multi-project environments was carried out. This enabled not only to evaluate quantitively various instruments of the base of the previously determined criteria and their weights, by study the advantages and disadvantages of each of them. This study allows to reveal the most efficient variant for the enhancement of the performance and optimization of projects management processes, providing the developers of the software with necessary information. However, it is important to take into account the fact, that the selection of the specific instrument must be stipulated by the specific needs and context of a separate project. That is why, the results of this study must be considered as a prompt but not as an absolute single-valued decision.*

***Key words:*** *multiproject environment, monorepositories management, Lerna, Yarn Workspaces, Bazel, Rush, Pants, Bit, Nx, multicriterial analysis, optimization of the software development, flexibility of the software development, projects scaling.*

## Introduction

In the sphere of the software development, especially within the context of multiproject environments a numerous unique challenges emerge. Multi-project environment – it is a scenario, where the teams of programmers and developers work over several projects simultaneously, it often results in intersection of the tasks and resources [1].

One of the key issues is the resources distribution. In the context of the multiproject management, the resources include time and efforts of the programmers, technical infrastructure and financial facilities, when several projects compete for these limited resources, there emerges the need in taking complicated decisions regarding the priorities and distribution, that can lead to the delay and cost growth [2].

Another important task – coordination of the interproject work. In the multi-project environment, synchronization of the schedules, matching of interests and efficient communication between the teams is a decisive factor for achieving the integrity and efficiency of projects realization [3].

The last but not least is the task of the software quality provision. In the situation, when the attention of the teams is distributed between numerous projects, there exists high risk of worsening the quality of separate projects as a result of insufficient resources or limited time [1].

Thus, the need of studying and realization of efficient tools and managing methods in multi-project environment is relevant and requires deep analysis and comprehension from the point of view of the programmers and developers of the software.

### Analysis of the recent studies and publications

In modern world of programming, particularly in the context of multi-project environments, it is important to learn more about the recent research and trends to adequately meet the challenges, facing the software developers. Further key studies in this field, taking into account their advantages and potential faults, will be considered.

In the study [4] the author underlines the importance of keeping code cleanliness. However, this work can be perceived as too theoretical and not always suitable for specific challenges, facing the developers in modern multi-project systems.

Authors of the research [5] stress on the continuous delivery as the key element during software development. However, their approach can be complicated for the implementation in certain environments, where the infrastructure and processes are not fully prepared for such type of automation.

In the paper [6] the author concentrates the attention on the modularity in Web-design. But the suggested theory may not take into account the complex character of integration and management of dependences, often appearing in multimodule projects.

The author of the research [7] studies the usage of the networking technologies. But this work can have limited application in the context of general multi-project environments, where more complex approach to projects management is required.

The authors of the study [8] highlight complex problems, dealing with multi-project environment management, but do not take into consideration specific technological challenges, connected with the software development, such as provision of the compatibility and integration of different platforms and tools.

These studies set important directions for understanding and improvement of the processes in multi-project environments, however, it is important to take into account the potential challenges and limitations, they can set before the developers.

### Allocation of the non-solved part of the general task

In the context of modern multiproject programming environments, one of the key issues is the selection and optimization of monorepositories managing tools. Although there exists numerous developed solutions, such as Lerna, Yarn Workspaces, Bazel, Rush, Pants, Bit, Nx and others, not all the aspects of their usage, integration and optimization were analyzed in details and compared in scientific research [9]. The relevance of the selection and optimization of the monorepositories managing tools is that the selection of the correct tool for monorepositories management can greatly influence the performance of the development, efficiency of the team work and ability of the project to adapt to variable requirements. These tools vary according to the following parameters, namely, support of various programming languages, integration with available CI/CD (Continuous Integration and Continuous Delivery/Deployment) pipelines (automated sequence of actions, which enables to integrate, test and deliver the updated software most efficiently), convenience of the dependences management and efficiency in scalable projects [10].

Taking this into account, the need in the detailed comparative analysis of these tools becomes obvious. Such analysis must contain the assessment of their functionality, performance, scalability, compatibility with various technological stocks, it must study their impact on general efficiency of the development in multi-project environments. All this will help the developers take substantiated decisions, regarding the choice of the tool and contribute to general advance in the practice of monorepositories management in the sphere of software development [11].

**Objective of the article** is to improve the efficiency of the development and simplification of the processes of projects management as a result of performing multicriterial analysis, regarding the selection of the most suitable tool for monorepositories management, proceeding from the specific needs of scalability, speed of project deployment, integration flexibility as well as technical requirements concerning code purity, reliability and performance. The objective of the research is

achieved by means of carrying out the detailed analysis and comparison of modern managing tools of monorepositories, such as, Lerna, Yarn Workspaces, Bazel, Rush, Pants, Bit and Nx in the context of determining their advantages, drawbacks and optimal scenarios of usage.

**Main part**

Efficiency of the programming code management, improvement of the coordination between the teams and optimization of the process of integration and deployment of the software is achieved as a result of usage of the tools of microrepositories management. The following tools are widely used by the professional developers.

System of the collection and design Lerna is intended for the optimization of the work with projects, containing several packages, simplifies the processes of the dependences management and packages distribution but the system is known for its complexity for setting the projects and it has serious limitations in the sphere of usage. For instance, Lerna, manages each package in monorepository separately. This means, that for large repositories with many packages each package needs individual analysis, updating of dependences, installation and collection, this process is rather long. Besides, if similar dependences are used in different packages, Lerna can not cache these dependences efficiently between the packages, making each package perform separate enquires for installation dependences [12].

Working environment Yarn Workspaces enables the developers to manage dependences efficiently in monorepositories. This environment supports common usage of the packages , simplifies the installation process and updating but requires additional efforts for the integration with other tools [13].

System of collection and testing Bazel. This tool from Google is suitable for scalable monorepositories with high performance and efficiency. Bazel optimizes the process of collection, using caching and incremental collection. This means that Bazel collects only those parts of the project, which were changed, but not the whole project again, that greatly reduces the time of collection. Due to its caching algorithms and parallelization, Bazel is able to process rapidly large volumes of data, this provides rapid collection and testing. However, not all these tools and programming languages can be easily integrated with Bazel, this can limit its usage in some projects. Also Bazel has complicated curve of learning and requires large resources for setting and support as compared with the analogues [14].

Package for assembly and testing Rush proposes powerful possibilities for dependences management and monorepositories construction. It optimizes dependences management between packages, allowing easily install, update and manage dependences for all the packages in the repository. However, usage of this package can be complicated in the projects where other tools for dependences management are used. Rush integration into the available projects, especially large and complex, needs time and may require considerable changes in the structure of the project [15].

System of assembly automation and dependences management Pants is oriented, first of all, at high speed and efficiency, especially for large code bases. This system supports many languages and platforms, providing flexibility for various projects. It enables to perform compilation tasks and testing parallelly reducing time of assembly. It is easily expanded and adapted to specific needs of the project. But as compared with other tools Pants may have less branched community of users and limited resources for learning and support. Besides, efficient operation of Pants in large projects may need considerable computational resources [16].

Managing package and reuse of code components Bit is concentrated on the reuse of the components and modularity. It enables to scale easily the projects, add new components and adapt to changes without altering the whole project, provides integration with cloud services and communities, promoting cooperation and exchange of components. But this package is not ideal for very large systems due to potential limitations in dependences management. For instance Bit depends on its own platform for the components management, this may create the dependence on

the specific service and limit flexibility in the selection of the technological solutions [17].

System of automatic assembling and dependences management Nx from Narwhal Technologies is a powerful tool for monorepositories management with high performance, especially for Angular- and React-projects. It uses incremental assembling and caching, that enables to increase the speed of development and reduce time of assembling. It supports integration with different development tools and programming languages, providing flexibility in the selection of technologies. However, for certain specific scenarios of development Nx can not provide sufficient flexibility or possibilities, needed for projects or processes management (for instance, Nx has strong integration with JavaScript- and TypeScript-systems and if the project is based on other programming languages or technological stacks, usage of Nx can be less efficient or even complicated) [18].

Comparative analysis of monorepositories management tools is presented in Table 1.

Using data from the comparative analysis of the instruments for monorepositories management, presented in [21 – 22], and the updated information with the help of the studies, presented in [12 – 18] we suggested the criterial analysis of different tools, the results of such analysis are presented in Table 2. This analysis comprises the wide range of parameters, from efficient processing of the dependences to integration aspects with other tools of the development. Great attention is paid to the assessment of scalability, flexibility of the configuration and cost parameters of each tool. Such an approach enables to provide all-round understanding of the advantages and limitations of each decision, promoting the optimization of the tools selection for the efficient management of multiproject environments.

For the assessment of the tools for multirepositories management the method of multicriterial analysis, based on R. Keeney and B. Howard concepts [19] was used. Application of this methodology allows to assess quantitively and compare different tools on the base of the already determined criteria, using the utility formula:

$$U_i = \sum_{i=1}^{n} x_{il} \times P_l \qquad (1)$$

where $U_i$ is the utility of the $i^{th}$ tool; $x_{il}$ – represents the assessment by the $l^{th}$ criterion; $i = 1, \ldots, 7$ – is the number of tools. $l = 1, \ldots, 10$ – is the number of criteria, $P_i$ – is the weight of $l^{th}$ criterion.

The utility of each tool will be determined by the formula (1):

$U_1 = 0.7\times1.0+0.6\times0.9+0.8\times0.8+0.7\times0.7+0.8\times0.6+0.7\times0.9+0.6\times0.5+0.7\times0.6+0.6\times0.5+0.7\times0.4 = 4.51.$

$U_2 = 0.8\times1.0+0.7\times0.9+0.9\times0.8+0.8\times0.7+0.7\times0.6+0.7\times0.9+0.7\times0.5+0.8\times0.6+0.7\times0.5+0.8\times0.4 = 4.95.$

$U_3 = 0.9\times1.0+0.9\times0.9+0.6\times0.8+0.6\times0.7+0.5\times0.6+0.9\times0.9+0.8\times0.5+0.6\times0.6+0.8\times0.5+0.6\times0.4 = 4.83.$

$U_4 = 0.8\times1.0+0.8\times0.9+0.7\times0.8+0.7\times0.7+0.6\times0.6+0.8\times0.9+0.7\times0.5+0.7\times0.6+0.7\times0.5+0.7\times0.4 = 4.79.$

$U_5 = 0.7\times1.0+0.9\times0.9+0.6\times0.8+0.6\times0.7+0.7\times0.6+0.9\times0.9+0.7\times0.5+0.6\times0.6+0.6\times0.5+0.6\times0.4 = 4.75.$

$U_6 = 0.6\times1.0+0.6\times0.9+0.7\times0.8+0.7\times0.7+0.8\times0.6+0.6\times0.9+0.6\times0.5+0.8\times0.6+0.7\times0.5+0.8\times0.4 = 4.61.$

$U_7 = 0.8\times1.0+0.8\times0.9+0.8\times0.8+0.8\times0.7+0.7\times0.6+0.8\times0.9+0.8\times0.5+0.9\times0.6+0.8\times0.5+0.7\times0.4 = 5.03.$

$\max U_l = U_7$

Table 1

**Comparative analysis of monorepositories managing tools**

| Tool | Advantages | Disadvantages | Optimal scenario of usage |
|---|---|---|---|
| Lerna | Efficient management of the dependences between management.<br>Automation of the processes of packages publication.<br>Ability to manage the versioning in complex projects. | Complex process of setting and maintenance.<br>Potential complication at project scaling.<br>Requires certain level of technical understanding for efficient usage. | Projects with several packages. |
| Yarn Workspaces | Centralized dependences management support of the common.<br>Usage of the code between the projects.<br>Optimization of the installation processes and packages updating. | Limited support non -JavaScript programming languages.<br>Complicated integration with certain CI/CD tools.<br>May need additional time for setting in complex projects. | Monorepositories with the stress on JavaScript. |
| Bazel | High performance and assembling efficiency.<br>Support of various programming languages and platforms.<br>Scaling for large and complex projects. | High curve of learning for new users<br>Configuration and support complexity.<br>May be overloading for small or average projects. | Large monorepositories, corporate level. |
| Rush | Efficient management of the dependences and projects assembling.<br>Integration with wide range of CI/CD tools.<br>Support of project insulation to avoid conflicts. | Need of deep understanding of the configuration.<br>Possible problems with integration in the available working processes.<br>Mau require considerable resources for large projects. | Multiplatform monorepositories. |
| Pants | Assembly rate for large code bases is optimized.<br>Support of various programming languages and frameworks.<br>Flexibility of setting and usage.. | Complexity of primary setting and management.<br>Potential challenges for integration with other systems.<br>May be complicated for understanding. | Projects with complex structure and large code base. |
| Bit | Concentrated on reusability and modularity of the components.<br>Support of decomposition and common usage of code between projects.<br>Intuitive interface and friendly management. | Limitations during the work with very large monorepositories.<br>May require additional time for the integration with the existing projects.<br>Certain limitations in management with complex dependences<br>. | Projects with the need in high reusability of the components. |
| Nx | High performance and efficiency especially for Angular and React projects.<br>Possibilities for scaling and projects optimization are built-in.<br>Support of the modernized working processes and tools. | May need some time for learning and adaptation, especially for unexperienced users.<br>Potential challenges during integration with other languages and frameworks.<br>Requirements, concerning the resource for the efficient usage in large projects. | Projects on Angular and React, medium and large monorepositories. |

Table 2

**Assessments of the tools for monorepositories management**

| № | $l^{th}$ criteria, $l = 1,l$ | Lerna | Yarn Workspaces | Bazel | Rush | Pants | Bit | Nx | weight of $l^{th}$ criterion |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 1 | Performance and efficiency | 0.7 | 0.8 | 0.9 | 0.8 | 0.7 | 0.6 | 0.8 | 1 |
| 2 | Scalability | 0.6 | 0.7 | 0.9 | 0.8 | 0.9 | 0.6 | 0.8 | 0.9 |
| 3 | Simplification of the dependences management | 0.8 | 0.9 | 0.6 | 0.7 | 0.6 | 0.7 | 0.8 | 0.8 |
| 4 | Integration with other tools | 0.7 | 0.8 | 0.6 | 0.7 | 0.6 | 0.7 | 0.8 | 0.7 |
| 5 | Flexibility of the configuration | 0.8 | 0.7 | 0.5 | 0.6 | 0.7 | 0.8 | 0.7 | 0.6 |
| 6 | Security and reliability | 0.7 | 0.7 | 0.9 | 0.8 | 0.9 | 0.6 | 0.8 | 0.9 |
| 7 | Support of various languages and frameworks | 0.6 | 0.7 | 0.8 | 0.7 | 0.7 | 0.6 | 0.8 | 0.5 |
| 8 | Intuitivity of the interface and convenient usage | 0.7 | 0.8 | 0.6 | 0.7 | 0.6 | 0.8 | 0.9 | 0.6 |
| 9 | Support of the community and documentation | 0.6 | 0.7 | 0.8 | 0.7 | 0.6 | 0.7 | 0.8 | 0.5 |
| 10 | Cost and licensing | 0.7 | 0.8 | 0.6 | 0.7 | 0.6 | 0.8 | 0.7 | 0.4 |

Thus, on the base of the performed calculations of the utility for various tools of monorepositories management, the conclusion can be made, that the best tool, according to the determined criteria and weights is Nx **(**$U_7$**)**. This tool demonstrates the highest general utility with the value of 5.03, that shows its efficiency in many key aspects, including the performance, scalability integration with other tools and community support, documentation.

Besides high utility, tool Nx has important advantages, which make it the ideal solution for monorepositories management. Key advantages of Nx comprise:

1. High performance: Nx is optimized for rapid work with large code bases, providing high performance even in complex projects.

2. Flexible management of dependences: Nx proposes extended possibilities for efficient dependences management that enables to control easily complex interconnections between different parts of the project.

3. Integration with popular frameworks and languages: Nx supports wide range of programming languages and frameworks, it is very efficient, working with Angular and React.

4. Scalability of the projects: Nx is an ideal choice for scaling the projects, providing flexibility and efficience, working with large and multifunctional projects.

5. Support of the community and documentation: Nx has active community of the developers and well supported documentation, promoting easy integration and usage of the tool.

6. Easy implementation and usage: Nx proposes intuitively understandable interface and simple setting making it accessible for the developers of different levels of qualification.

These and other characteristics make Nx the tool which meets the requirements of modern developers and teams, looking for efficient and flexible solutions for monorepositories management [20].

## Conclusions

Within the context of enhancing the efficiency of multi-projects environments management in the sphere of the software development, new concept was suggested, the essence of the concept is application of complex multicriterial analysis for the assessment of the tools for monorepositories management. The concept enabled to perform the detailed quantitative assessment of various tools, taking into account the determined criteria and their weights, efficiently underlying the advantages and limitations of each of them. The suggested approach differs from the available by complex and system assessment of the microrepositories managing tools. Unlike the conventional approaches, which can be relied on the subjective assessment or partial analysis, this concept uses detailed analysis by numerous criteria, providing quantitative and objective base for the tool assessment. Thus, this approach, enables to obtain balanced assessment, taking into consideration various factors, decisive for the efficient development of the software in multiproject environments, that leads to more substantiated decision making, regarding the selection and usage of the corresponding tools.

The research, carried out, demonstrated that that Nx tool can be considered as the most efficient variant, providing the developers of the software with necessary information, regarding the enhancement of the performance and optimization of project management.

It should be noted, that in general case the choice of the tools depends on the specific requirements and context of each project, that is why, the suggested results should be considered as the reference point but not as a definitive solution.

## REFERENCES

1. Achieving success in large, complex software projects [Electronic resource] / M. Bloch, S. Blumberg, J. Laartz // McKinsey Digital. – 2014. – Access mode : https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/achieving-success-in-large-complex-software-projects.

2. Managing complex software projects [Electronic resource] / Atlassian. – Access mode : https://www.atlassian.com/team-playbook/plays/ludicrously-complex-software-projects.

3. Transfer learning : a comprehensive introduction [Electronic resource] / A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, M. Abdul Azim // Journal of Big Data. – 2022. – № 102 (2022). – Access mode : https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00652-w.

4. Clean Code : A Handbook of Agile Software Craftsmanship [Electronic resource] / C. Martin. // Prentice Hall. – 2008. – Access mode: https://github.com/jnguyen095/clean-code/blob/master/Clean.Code.A.Handbook.of.Agile.Software.Craftsmanship.pdf.

5. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation [Electronic resource] / D. Farley J. Humble // Addison-Wesley Professional. – 2010. – Access mode : https://github.com/aaquresh/CITraining/blob/master/Ebooks/Continuous%20Delivery%20-%20Reliable%20Software%20Releases%20Through%20Build,%20Test%20And%20Deployment%20Automation.pdf.

6. Modular Web Design: Creating Reusable Components for User Experience Design [Electronic resource] / N. Curtis // Peachpit Press. – 2009. – Access mode : https://dokumen.pub/modular-web-design-creating-reusable-components-for-user-experience-design-and-documentation-9780321601353-0321601351-9780321638311-032163831x.html.

7. Yeping Z. The Interactive Design of Library Information Sharing in View of Network Communication Technology / Z. Yeping // Advanced Pattern Recognition Systems for Multimedia Data. – 2022. – Vol. 2022. – P. 1 – 14. – URL: https://doi.org/10.1155/2022/5342645.

8. Managing complex projects in multi-project environments [Electronic resource] / G. Hagan, D. Bower, N. Smith // Procs 27th Annual ARCOM Conference, 5 – 7 September 2011, Bristol, UK, Association of Researchers in Construction Management. – P. 787 – 796. Access mode : https://www.researchgate.net/publication/267423601_Managing_complex_projects_in_multi-project_environments/.

9. Monorepo in action for library maintenance [Electronic resource] / N. Nguen // Access mode: https://namnguyen.design/blog/2023-07-03-monorepo-in-action-for-library-maintenance-%F0%9F%92%9E.

10. Prus O. V. Usage of graph neural networks for automatic detection of the dependences between the components in monorepositories / O. V. Prus, V. P. Maidaniuk // IIIrd All-Ukrainian scientific-technical conference of young scientists, post graduates and students «Computer games and multimedia as the innovation approach to communication – 2023». – 2023. – P. 211 – 214. (Ukr).

11. Prus O. V. Analysis of the basic principles of monorepositories operation: characteristic features, advantages and drawbacks / O. V. Prus, V. P. Maidaniuk // XVI International scientific practical conference «Information technologies

and automation – 2023». – 2023. – P. 267 – 270. (Ukr).

12. Documentation. Lerna [Electronic resource] : Access mode : https://lerna.js.org/.
13. Yarn Workspaces [Electronic resource] : Access mode : https://classic.yarnpkg.com/lang/en/docs/workspaces.
14. Bazel build system [Electronic resource] : Access mode : https://bazel.build/.
15. Rush: a scalable monorepo manager for the web [Electronic resource] : Access mode : https://rushjs.io/.
16. Pants 2: The ergonomic build system [Electronic resource] : Access mode : https://www.pantsbuild.org/.
17. Painless Monorepo Dependency Management with Bit [Electronic resource] / Z. Kochan // Access mode : https://bit.dev/blog/painless-monorepo-dependency-management-with-bit-l4f9fzyw/.
18. Nx: Smart, FastExtensibleBuild System. system [Electronic resource] : Access mode : https://nx.dev/.
19. Decisions with Multiple Objectives. Preferences and Value Tradeoffs / R. Keeney, R. Howard // Cambridge University Press. – 2014. – 592 p. – URL : https://doi.org/10.1017/CBO9781139174084.
20. Read B. 6 reasons why we chose Nx as our monorepo management tool [Electronic resource] – 2021 : Access mode : https://medium.com/purplebricks-digital/6-reasons-why-we-chose-nx-as-our-monorepo-management-tool-1fe5274a008e/.
21. Monorepo Part 3: The benchmark [Electronic resource] / R. Tahar // Medium. – 2022. – Access mode : https://engineering.combohr.com/part-3-the-benchmark-adb90f416151.
22. Monorepo vs Microrepo: How to Choose the Best Repository Structure for Your Code [Electronic resource] / K. Nirav // DEV. – 2023. – Access mode : https://dev.to/kanani_nirav/monorepo-vs-microrepo-how-to-choose-the-best-repository-structure-for-your-code-4pce.

Editorial office received the paper 29.01.2024.
The paper was reviewed 06.03.2024.

***Prus Oleg*** – Post Graduate with the Software Department.

***Maidaniuk Volodymyr*** – Cand. Sc. (Eng.), Associate Professor with the Software Department.

***Arseniuk Igor*** – Cand. Sc. (Eng.), Associate Professor with the Department of computer science, e-mail: igrosars@gmail.com.
Vinnytsia National Technical University.