A Comparative Analysis of Low-Code and Traditional Platforms: Assessing Effectiveness in Governmental Digital Transformation Initiatives

Prof. Dr. Mohamed Abdel-Fattah Belal* **Dr. Sara Mohamed Mosaad****
Moaiad Ismail Hawash+***

ABSTRACT

One of the most important factors in promoting digital transformation is accelerating the digital transformation process while ensuring quality and efficiency. This research therefore seeks to measure the effectiveness of Low-code Development Platforms (LCDPs), in driving government digital transformation, specifically through a comparative analysis of OutSystems and .NET MVC models. The central focus is on assessing various factors, with a keen eye on three essential metrics: Development Time (DT), Number of Lines of Code (NLC), and Number of Methods (NM).

The study aims to discern the efficiency needs of government, influencing the development of applications that align with organizational objectives. By using the Low-code Development Platforms (LCDPs), capabilities embedded in OutSystems and .NET MVC, a thorough comparison will be conducted to carefully examine precise differences and evaluate impact through DT, NLC, and NM.

On the other side of these technical aspects, the research tries to provide insightful guidance for decision-makers, aiding them in selecting the most fitting platform for digital transformation initiatives within the government and the public sector.

Keywords: Outsystems, .NET, MVC, LCDP, Low-Code, Development, Platforms, Government, Digital Transformation.

* Professor of Computer Science Dep. of Computer Science Faculty of Computers and Artificial Intelligence Helwan University

** Assistant professor in Business Information Systems Faculty of Commerce and Business Administration Helwan University

*** Masrer's Researcher in Business Information Systems Faculty of Commerce and Business Administration Helwan University

تحليل مقارن للمنصات منخفضة التعليمات البرمجية والمنصات التقليدية: تقييم الفعالية في مبادرات التحول الرقمي الحكومية

ملخص الدراسة:

إن أحد أهم العوامل في تعزيز التحول الرقمي هو تسريع عملية التحول الرقمي مع ضمان الجودة والكفاءة. لذلك يسعى هذا البحث إلى قياس مدى فعالية أطر العمل ذات التعليمات البرمجية المنخفضة في دفع التحول الرقمي الحكومي، وتحديداً من خلال تحليل مقارن لنماذج عمل على منصتي OutSystems وNET. MVC. حيث ينصب التركيز الرئيسي على تقييم العوامل المختلفة بين كل من بيئتي العمل، مع الاهتمام والتركيز الشديد بثلاثة مقاييس أساسية وهي: وقت التطوير وعدد سطور الكود أو التعليمات البرمجية وعدد الطرق أو الأساليب البرمجية.

تهدف الدراسة إلى تحديد احتياجات الكفاءة للمؤسسات الحكومية والعامة، مما يؤثر على تطوير التطبيقات التي تتوافق مع الأهداف التنظيمية. باستخدام إمكانات التعليمات البرمجية المنخفضة المضمنة في OutSystems وNET. MVC ، سيتم إجراء مقارنة شاملة لفحص الاختلافات بين كل من بيئتي العمل وتقييم وتحليل النتائج وتأثيرها من خلال ثلاثة عوامل رئيسية وهي وقت التطوير وعدد سطور الكود أو التعليمات البرمجية وعدد الطرق أو الأساليب البرمجية.

وعلى الجانب الآخر من هذه الجوانب التقنية، يحاول البحث تقديم إرشادات ثاقبة لصناع القرار، ومساعدتهم في اختيار المنصة الأنسب لمبادرات التحول الرقمي داخل الحكومة والقطاع العام.

الكلمات المفتاحية: اوت سيستمز، دوت نت، MVC، LCDP، التقنيات ذات التعليمات البرمجية المنخفضة، الكود المنخفض، التطوير، المنصات، الحكومة، التحول الرقمي.

INTRODUCTION

Government entities, amongst growing customer demands, are turning to Low-Code Development Platforms (LCDPs) for immediate and automated software application creation. Low code is chosen for its ecosystem that enables application creation with minimal hard code definition, focusing on visual interfaces for non-programming users [1, 7]. The main goal is to help Government entities create software applications quickly and easily without the need for hard coding and complex programming, which offers a rapid and lower-cost solution [1]. Due to technology affecting and impacting our lives in all aspects and businesses, customized software solutions become essential, so the need for LCDPs has increased as a result.

LCDPs are recognized by major IT entities as offering user-friendly environments for citizen developers, as these platforms accelerate application delivery by reducing development time, reducing manpower which affects project cost or budget, and enhancing scalability [1,4]. This can be achieved by filling gaps between domains, Low-Code Development Platforms (LCDPs) remove the need for highly advanced and specialized programming skills, that increase application delivery productivity [1,4]. At the same time, developers can focus on business needs or business process flow by automating code through the accelerated and generated code approach [4].

The low-code/no-code approach reduces barriers and complications in the software development workflow, which is critical in this era of digital transformation [12, 18, 26]. LCDP provides not only speed but efficiency significant increases also for businesses and individuals [15]. They provide a runtime environment for application logic, monitoring and orchestrating

the application lifecycle for reliability, scalability, security, and monitoring [13].

In the current market landscape, firms, regardless of size and industry, need to digitally transform their information systems [27,28]. LCDPs, characterized by visual interfaces and intuitive capabilities, play a crucial role in resolving challenges related to business innovation tasks in IT teams [2,29,37]. With model-driven and metadata-based programming languages, LCDPs support the creation of user interfaces, business logic, and data services [13].

As a result, the business world is increasingly moving toward transferring most businesses online, which is why modeling languages are gaining importance [14,15]. LCDPs, with integrated Domain-Specific Languages (DSLs), facilitate the development of online and mobile apps, enabling quicker and more agile development cycles and higher-quality products [15].

Also, LCDPs are defined as a "Set of tools for programmers and non-programmers. It enables the quick generation and delivery of business applications with minimum effort to write in a coding language and requires the least possible effort for the installation and configuration of environments, and training and implementation" [4].

Also, LCDP is defined as "Products and/or cloud services for application development that employs visual, declarative techniques instead of programming are available to customers at low- or no-cost in money and training time to begin, with costs rising in proportion of the business value of the platforms." [17].

1.1. OutSystems Platform:

OutSystems, as a company, offers a single product — the OutSystems platform — designed to craft enterprise and mobile

web applications that are maintainable, and scalable. Regarding developers, their previous experience is not important because they develop web applications on the technical platform while developing with the Visual Domain Language in an environment that is based on integrated development which is widely recognized [21].

The OutSystems Platform is considered as a high-efficiency development tool for creating rapid Enterprise Web Applications and mobile apps. The Integrated Development Environment, Service Studio, is essential in designing, maintaining, and deploying applications using a Visual Domain Specific Language. This language's graphical symbols streamline data modeling, UI composition, and business logic programming [21].

The OutSystems platform makes it possible to develop different application parts: data model, logic, and user interface. It also allows to manage the execution of various applications, their deployment process, and the life cycle of each application [5,11].

OutSystems provides some tools to help during development phases [5]:

- **Service Studio:** the environment where it is possible to build the different application parts; this tool uses drag-and-drop visual elements and allows applications to use SOAP and REST web services.
- **Experience Builder:** a set of specialized tools focused on the development acceleration by untangling the creation of applications.
- **Workflow Builder:** a tool directed to business experts, analysts, and process owners. By abstracting the development process from the workflow, internal productivity increases.

- **Integration Studio:** the environment where developers can create components that extend the OutSystems platform; it is with this tool that the integration with third-party systems and microservices is done; once deployed, the features developed here are reusable by all OutSystems applications.
- **Forge:** a repository that contains UI components, libraries, and connectors and provides existing modules that OutSystems applications can reutilize.

Managing applications during their development stages [5]:

- **Lifetime:** a console that performs central management of all development, Quality Assurance (QA), and production environments; this component also automates Development and Operations (DevOps) processes.
- **Service Center:** a console to perform operational management of an environment, including taking applications online and offline, reverting applications to previous versions, and configuring some properties such as batch job scheduling.
- **Architecture Dashboard:** a tool that conducts code inspection and runtime analysis. It recommends refinements for performance improvement, security, user experience, and architecture.

An overview of the different tools provided by the OutSystems platform is available in Figure 1:

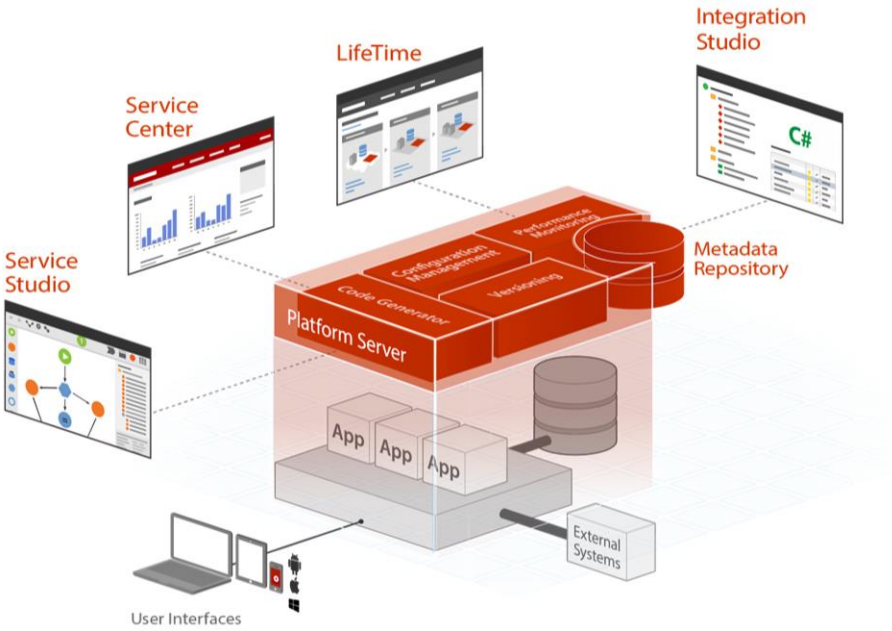

Figure 1 OutSystems Platform Overview [5]

1.2. Key features of the OutSystems Platform

OutSystems LCDP is characterized by essential aspects that make it one of the best choices of the low-code development platforms, some of these key aspects are [19]:

1) A sophisticated cloud-based low-code development platform prioritizing performance, scalability, and high availability.
2) Extension of the standard Web Application Server setup with additional services and repositories for application generation, building, packaging, and deployment.
3) Flexibility to deploy on Microsoft .NET stack, Oracle WebLogic, or JBOSS, allowing code in C# or Java with standard CLR or JVM technology.

4) Service Studio, a desktop application offering easy access to platform functions such as UI modeling, Business Processes, Databases, and more.
5) Integration Studio for technical developers to integrate external libraries, services, and databases with OutSystems.
6) Native Windows applications (Service Studio and Integration Studio) connecting to remote sessions on a Platform-as-a-Service (PaaS) platform.
7) Accessibility through web browsers, eliminating local software installation requirements and operating system limitations.

1.3. Advantages of Low-Code Development Platforms:

Low-Code Development Platforms (LCDPs) are acknowledged for their promising future, and as shown in Figure 2 experts emphasize several key advantages [6, 7, 26, 52]:

1) **Data Confidentiality:** As applications can be developed by users without an extensive technical background, businesses trust their internal staff for development tasks, enhancing data confidentiality.
2) **Speed:** With a significant portion of the code pre-developed, users visually configure applications, expediting development. A Forrester survey found that LCDPs accelerated development by 5 to 10 times.
3) **Cost Efficiency:** Reduced development timelines translate to lower costs, whether applications are developed in-house or by external developers.
4) **Simplified Complexity:** The non-scratch development approach simplifies the process, allowing a focus on customizing software to meet precise user requirements.

5) **Ease of Maintenance:** Low-code platforms, with minimal coding requirements, significantly reduce maintenance workloads.
6) **Engagement of Business Professionals:** LCDPs provide user-friendly interfaces, empowering end-users to take on a developer role without specialized technical knowledge. About 44% of LCDP users are business professionals collaborating with IT.

Based on 'The State of Application Development' survey [26], which surveyed over 3300 IT professionals worldwide, key motivations for adopting LCDP included accelerating digital transformation, increasing flexibility in among business needs (66% of the respondents), technical skills difficult to recruit (45% among the respondents) [1].

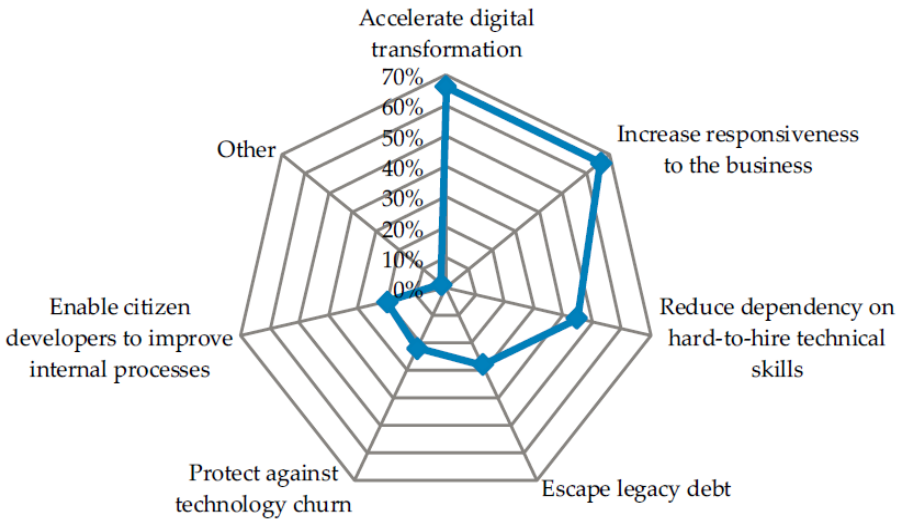

Figure 2 Main reasons for using low-code development platforms [1]

Literature Review

The literature review underscores the urgency for governments to swiftly adapt to technological changes as a critical challenge. Emphasizing the significance of low-code application development platforms emerges as a viable solution to enhance service delivery, streamline operations, and foster active engagement from both developers and citizens. The incorporation of low-code development, visual interfaces, and pre-built components becomes pivotal in achieving these objectives. The versatility of this tool is evident, as even individuals with limited programming experience can actively participate in a straightforward and efficient manner during such evaluations. The primary aim is to assess the extent to which low-code platforms align with the requirements of public digital transformations.

Contemporary attention is directed towards employment-related issues, proposing a strategy that involves leveraging low-code development platforms to address the rapidly evolving landscape of modern work. The authors advocate for the utilization of low-code platforms, enabling individuals to swiftly build applications through visual interfaces and pre-existing modules. The paper introduces an ideal method for enhancing employment skills using the low-code development technique, encompassing the identification of necessary skills, selection of an appropriate platform, training, and certification acquisition, and active participation in practical assignments. The authors posit that this approach can yield several positive outcomes, including increased competitiveness in employment, career development, adaptability to fast-changing innovations, and the recognition of individuals as valuable assets in the labor market [3].

Delves into the challenges arising from shifting employment trends and proposes a remedy centered on a low-

code development platform. In the context of a rapidly evolving labor market, the study underscores the necessity for individuals to swiftly adapt to new technologies to enhance employability. Criticizing traditional software development for its time-consuming nature and difficult programming skills, the authors believe in the strategic use of low-code development platforms because these platforms enable users with limited programming experience to quickly build applications through visual interfaces and pre-built components and provide benefits such as accelerating development timelines, simplifying learning, and enhancing collaboration between business and IT teams. The authors outline a strategic employability enhancement approach involving steps like identifying in-demand skills, choosing a suitable low-code platform, acquiring training and certifications, and engaging in practical projects. They stress the positive impacts on employability dimensions, including competitiveness, career growth, and adaptability to evolving technology, positioning individuals as valuable assets highly sought after by potential employers [30].

Recognizing the challenges faced by designers and practitioners when collaborating on low-level development highlights the need to bridge the gap between these activities for improved software development outcomes. The results suggest that addressing this gap is crucial, as designers bring unique strengths that can be leveraged and enhanced through enhanced efficiency and effective communication skills.

Historically, traditional software development processes have grappled with challenges arising from the separation of artists and producers, leading to misunderstandings and development negotiations hampered by insufficient communication. The authors advocate for the early involvement of designers in the low-code process, integrating them seamlessly into the development lifecycle. This approach

harnesses their expertise to create user-centered, portable applications that are both concise and elegant. The paper explores barriers to effective collaboration and aims to facilitate inter-functional collaboration, particularly in low-regulation ecosystems where design principles play a crucial role.

The central argument of the authors emphasizes the significance of fostering collective cultural reasoning among individuals to achieve a shared understanding. The key takeaway is to underscore the importance of developing a collaborative mindset to enhance communication and collaboration in software development processes. [31].

Exploring the factors influencing the reliability and adoption of low-code development platforms has garnered attention for expediting application development and diminishing the reliability constraints associated with traditional coding practices. This investigation involved surveys conducted among developers, IT professionals, and decision-makers with expertise in low-code development. The focus of the study encompassed aspects such as decision-makers' roles, perceived benefits and challenges, organizational support, and individual proficiency.

The findings underscore significant determinants affecting the utilization of low-code platforms, including considerations of time and cost efficiency, ease of use and learning curve, customization and flexibility, integration capabilities, and the impact on organizational support and cultural factors. Participants acknowledged the rapid development capabilities, user-friendly interfaces, adaptability to business needs, compatibility with existing systems, and the crucial role of organizational support in ensuring the successful utilization of low-code platforms. The paper offers valuable insights for organizations, aiding them in making well-informed decisions regarding the adoption and implementation of low-

code systems. Factors such as time efficiency, ease of use, customization options, integration capabilities, and organizational support are crucial considerations highlighted in the study [32].

The author delves into the application of low-code development techniques at a granular level to facilitate digital transformation processes. The paper provides an overview of these techniques, emphasizing their pivotal role in the successful implementation of digital transformation by making it more accessible. With user-friendly interfaces, pre-built components, and drag-and-drop functionalities, low-code platforms are tailored to cater to non-technical users, ensuring the advantages of low-code development in the digital landscape. The authors strive to expedite application delivery to address evolving market needs, emphasizing flexibility and collaboration between business and IT teams.

In the process of choosing a low-code development platform, the paper underscores key considerations such as scalability, integration, security, and vendor support. Real-world case studies and examples are incorporated to illustrate the effective implementation of low-code development processes in driving digital innovations and achieving business objectives. The intention is to provide insights for informed decision-making, acknowledging the diverse values and paths organizations may encounter. In summary, the paper offers a comprehensive exploration of the digital transformation journey, portraying low-code platforms as valuable tools for organizational engagement. It analyzes content, enumerates benefits, discusses decision-making factors, and provides practical examples of implementation [33].

Another research paper explores the relationship between low-code development and model-driven engineering (MDE) methods in software development, aiming to discern whether

these approaches complement each other or represent distinct paradigms. The authors scrutinize their fundamental principles and concepts, noting that while low-code development platforms simplify the software development process through visual interfaces and abstraction layers, MDE emphasizes the use of modeling languages and code generation to automate development.

The paper underscores both similarities and differences between the two methods, recognizing their shared objectives of enhancing productivity and bridging the gap between business requirements and software implementation. Low-code development aims at rapid software development for non-technical users, relying on pre-built components, while MDE focuses on modeling requirements and code generation for complex systems. Consequently, the authors advocate for a hybrid approach that harnesses the strengths of both methods. They propose using low-code platforms for rapid prototyping and user interface design, coupled with MDE techniques for generating efficient, maintainable, and scalable code from formal models.

In conclusion, the authors suggest that low-code development and MDE can be integrated, offering a balanced approach that enhances productivity, flexibility, and scalability within the software development process itself [34].

When delving into the concept of low-code systems and their importance in software development, the paper thoroughly explores how these platforms streamline the development process. This is achieved by providing user-friendly visual interfaces, pre-built components, and drag-and-drop functionality. The authors consistently underscore the advantages of low-code platforms, emphasizing increased development speed, rapid prototyping, and improved collaboration between business stakeholders and IT teams.

The paper also addresses challenges that organizations encounter, such as scalability, customization flexibility, and integration capabilities, and highlights how low-code platforms have effectively mitigated these issues. Real-world case studies and examples demonstrate the tangible success of low-code systems across various industries, showcasing their adaptability and effectiveness in creating applications that cater to specific business needs.

The conclusion of the paper emphasizes the value of low-code platforms in application development, offering an overview of their key characteristics, benefits, and practical applications. For organizations considering the use of low-code platforms for software development projects, this paper serves as a valuable and informative resource [35].

There is a research study in software engineering that seeks to integrate agile software engineering processes, such as Scrum, with low-code development platforms for creating educational materials. Acknowledging the importance of agility in educational contexts, the authors argue that low-code platforms offer an ideal learning environment due to their visual nature, speed, and collaborative features. The study explores the integration of Scrum into low-code development, aligning it with core agile principles like iterative development and cross-functional teaming. It delves into low-code environments showcasing user stories and a Sprint Planning board.

The paper focuses on providing students with practical experience in utilizing the Agile Project Management approach and low-code platforms for rapidly developing applications, establishing a connection between theory and practice. The authors stress the significance of hands-on experience, emphasizing that low-code platforms contribute to transforming agile programming into a tangible reality. Consequently, the paper recommends the integration of Scrum and Low-Code as

an effective method for teaching Agile software engineering to students, fostering teamwork, and enhancing their adaptability [36].

Generally, traditional automation involves much expensive and time-consuming work due to the need for extensive custom coding. However, low-code platforms provide a solution with visual interfaces and pre-formed parts for fast process automation. Low-code platforms are highlighted in this paper as having benefits for manufacturing companies that involve non-technical users to be part of automation using simplified workflows and data flow. For example, case studies of process automation in inventory management and quality control display better efficiencies, fewer errors, and faster decision-making. This is because low-code platforms flexibly and easily adapt to changing needs. This paper presents challenges like data security and system integration giving insights on how the organization can effectively adopt it. It concludes that low code platform provides users who are not technical with the ability to create applications which increases operational efficiency [7].

A study talks about the new era of no-code/low code with NO experience NEEDED in CODE! The latest platforms with friendly user interfaces and modular elements intend to rewrite software programming from scratch in order to provide it for everyone (non-coders as well). These include a reduction in the entry barrier, making access easier, and easy to implement. However, this paper points out the weaknesses associated with customization and upgradeability and adds that complex requirements might involve complex coding or many programmers. Likewise, no-code and low-code development must also consider maintaining software quality and security. In summary, it means that alongside such platforms, there is a need to pair them up with conventional coding abilities since

they are likely to become more sophisticated in the coming years which may end up straining some organizations [16].

COMPARISON

In the dynamic realm of software development, two prominent platforms, the .NET environment, and the OutSystems environment, have emerged as leading choices for organizations. Each platform has different strengths and serves varied development needs. The .NET environment, developed by Microsoft, is polymorph and strong, supporting a wide kind of applications. It offers flexibility in programming languages and has a huge and strong development community. On the other hand, OutSystems is a low-code platform created with a focus on ready-made components and visual tools for quick application development.

In this comparison, we explore the key features and use cases of both environments. The .NET environment, known for its traditional approach, provides full control but demands higher technical expertise and development time. On the other hand, OutSystems, with its low-code approach, offers a faster and more accessible development process, making it preferred for organizations with priority in agility and cost effectiveness. The comparison extends to building a sample of a small store website with the MVC framework, highlighting the significance of trusted code quality, as OutSystems is based on .NET and adheres to software engineering requirements while the comparison dimensions or factors will be about the Development Time (DT) factor, Number of Lines of Code factor and Number of Methods (NM).

RESULT

In comparing .NET MVC and OutSystems across the DT, NLC and NM factors, the following table records the result as shown in Table 1.

Note: Assuming that who will do the development is a highly expert person and his development and coding skills are very high with a significant number of years of software development experience on both platforms also.

Table 1 Result of comparison Outsystems vs .NET

	DT Development Time	NLC Number of Lines of Code	NM Number Of Methods
Traditional (.NET MVC)	16 hours	3,700 Lines of code	312 Methods
Low-Code (Outsystems)	1 hour	314 Lines of code	76 Methods

Result Analysis:

Certainly, let's analyze the provided results for both Traditional (.NET MVC) and Low-Code (Outsystems) based on development time, number of lines of code, and number of methods:

Development Time:

- Traditional (.NET MVC): 16 hours
- Low-Code (Outsystems): 1 hour

Analysis: Outsystems, being a low-code platform, significantly outperforms traditional .NET MVC in terms of development time. The development time for Outsystems is much shorter (1 hour) compared to the traditional approach (16 hours). This suggests that developing the same application or functionality using Outsystems is considerably faster as shown in Figure 3.

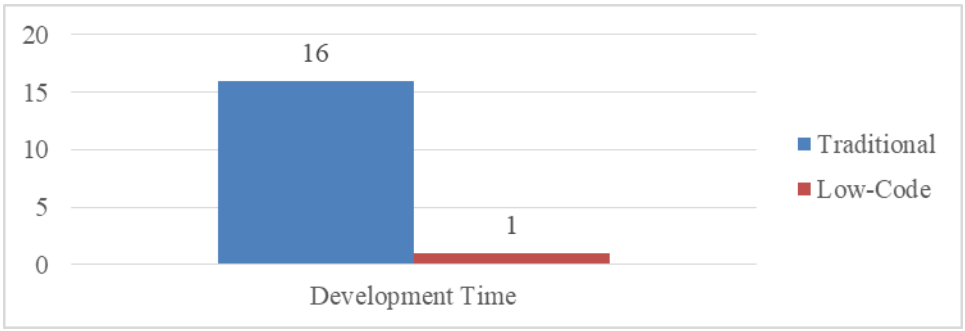

Figure 3 Development Time Factor

Number of Lines of Code:

- Traditional (.NET MVC): 3,700 lines of code
- Low-Code (Outsystems): 314 lines of code

Analysis: Outsystems result in fewer lines of code (314) compared to the traditional .NET MVC approach (3,700). Low-code platforms typically abstract a lot of the complexity, resulting in concise and efficient code as shown in Figure 4.

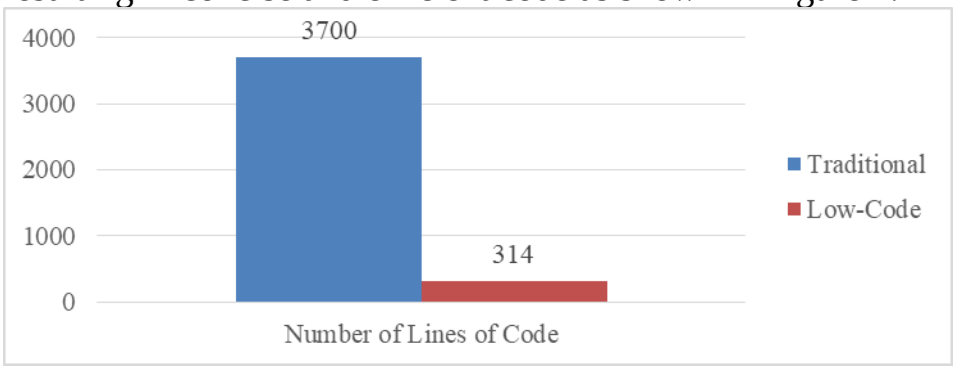

Figure 4 Number of Lines of Code Factor

Number of Methods:

- Traditional (.NET MVC): 312 methods
- Low-Code (Outsystems): 76 methods

Analysis: The low-code approach again demonstrates a notable reduction in the number of methods (76) compared to the traditional approach (312). This indicates that the low-code

platform abstracts and simplifies the logic, resulting in a more streamlined and modular structure as shown in Figure 5.

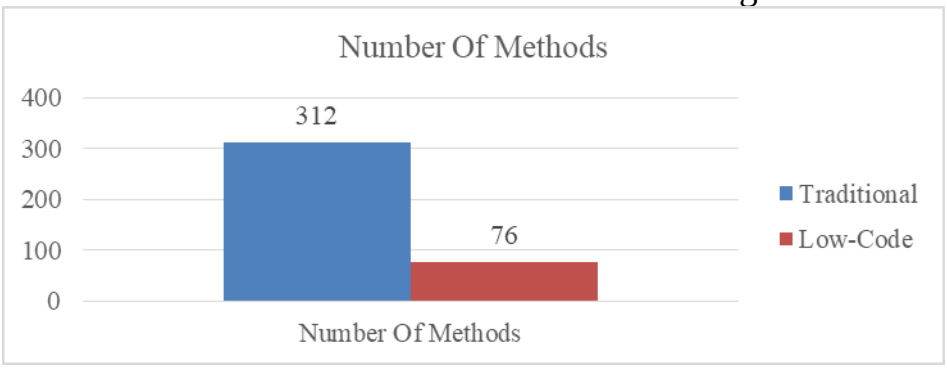

Figure 5 Number of Methods Factor

Overall Analysis:

Outsystems, as a low-code platform, excels in terms of development speed and code efficiency. It allows for the creation of the same functionality with significantly less manual coding effort. This can be advantageous in scenarios where rapid development and reduced maintenance overhead are critical factors.

Traditional .NET MVC, while offering control and flexibility, requires more time and lines of code to achieve the same result. The higher number of methods in the traditional approach might indicate a more granular and detailed implementation compared to the more abstracted low-code solution.

As trying to measure and analyze the overall factor with each platform to find the comparison and differentiation between platforms for the same factor by using a stacked column chart to visualize the result.

A stacked column chart is a type of data visualization that displays multiple data series as stacked columns, where each column represents a category or a group, and each segment within the column represents a sub-category or a value

within that group. This chart type is useful for comparing the total sizes across different groups and understanding the composition of each group.

As shown in Figure 6 note that the blue area expresses the Traditional platform and the red area expresses the Low-Code platform with the provided numbers recorded as a result in Table 1.

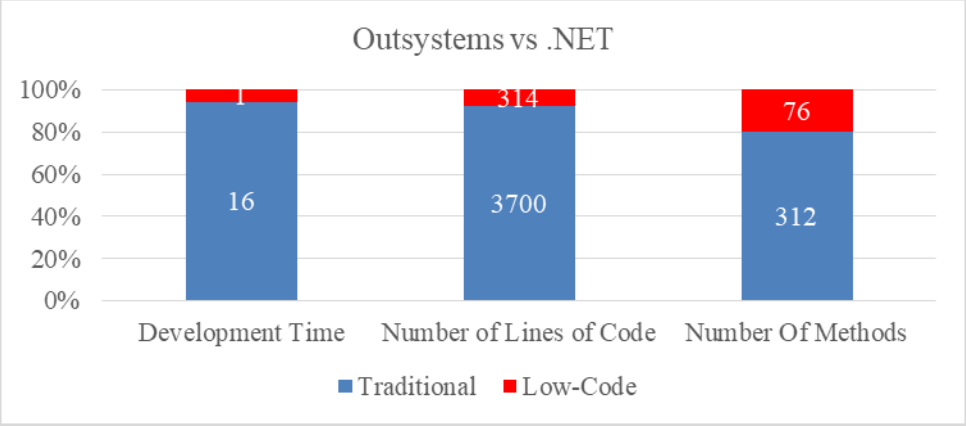

Figure 6 Comparing results as 100% stacked column chart

To get better indications we will use these values for each factor and get its percentage for the whole factor as shown in Figure 7 using a 100% stacked column, to facilitate analysis of the provided percentages in two ways for both Traditional and

Low-Code platforms across development time, number of lines of code, and number of methods.

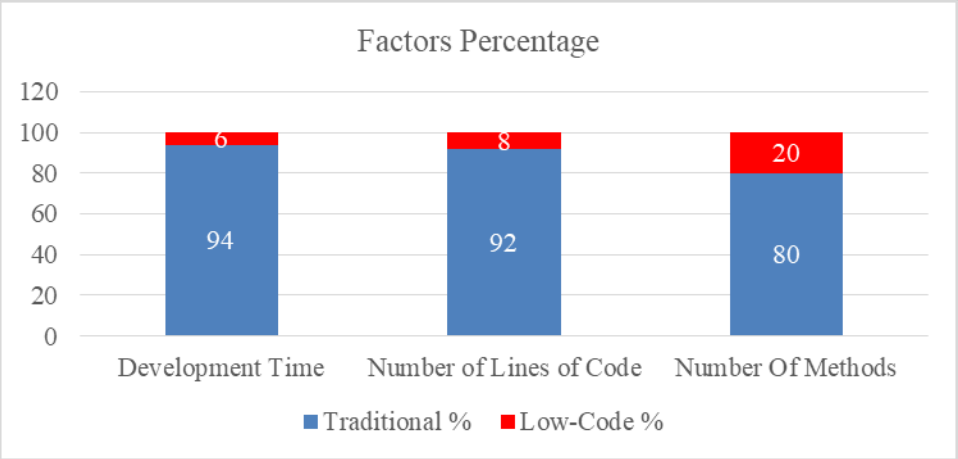

Figure 7 Comparing percentage as 100% stacked column chart

Factors percentage to each other on the same platform

In evaluating software development methodologies, it is crucial to understand the relative importance assigned to different factors within a given platform. The distribution of percentages across key metrics provides insights into the priorities and emphasis placed on various aspects of the development process. This analysis aims to explore how factors such as development time, number of lines of code, and number of methods are weighted against each other within the context of a specific platform as shown in Table 2.

Table 2 Factors percentage for each other on the same platform

	DT Development Time	**NLC Number of Lines of Code**	**NM Number Of Methods**	**%**
Traditional (.NET MVC)	35%	35%	30%	100%
Low-Code (Outsystems)	18%	24%	59%	100%

Traditional Platform:

For the traditional development approach, the allocation of percentages reflects the platform's focus on balancing development speed, code volume, and code structure. Each factor plays a distinct role in shaping the evaluation, with development time and lines of code sharing a similar level of significance, and the number of methods contributing to a lesser extent as shown in Figure 8.

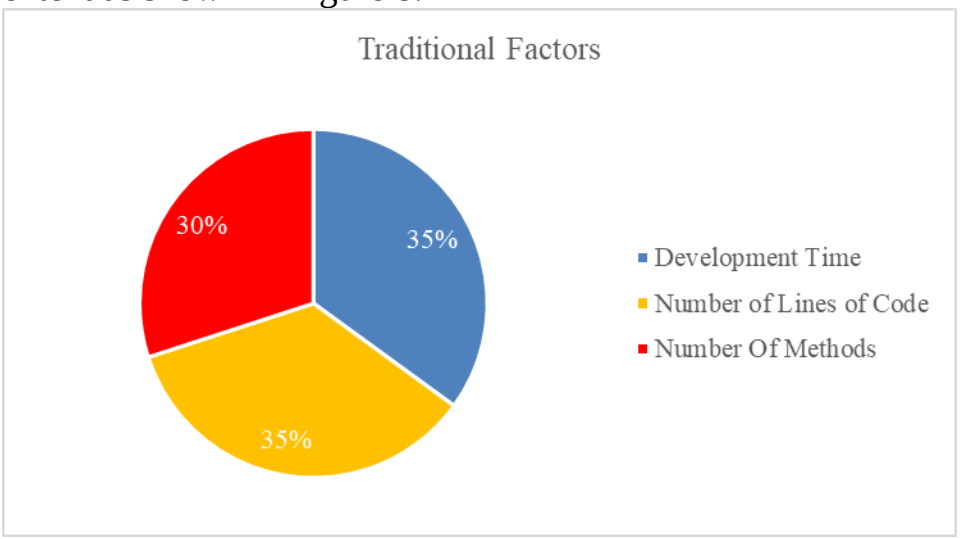

Figure 8 Traditional factors percentage to each other as a pie chart

- **Development Time: 35%**

This indicates that the time spent on development contributes the most to the overall evaluation of the traditional platform.

- **Number of Lines of Code: 35%**

The number of lines of code also carries a significant weight in the evaluation, suggesting that code volume is an important factor.

- **Number of Methods: 30%**

The number of methods contributes the least to the evaluation, implying that the structure or modularity of the code is not as critical in this context.

Low-Code platform:

In the low-code development paradigm, the distribution of percentages conveys a different set of priorities. Notably, the number of methods takes precedence, highlighting the platform's emphasis on code modularity, structure, and complexity. Development time, while still considered, holds less weight, suggesting that the efficiency of code production is not the primary concern as shown in Figure 9.

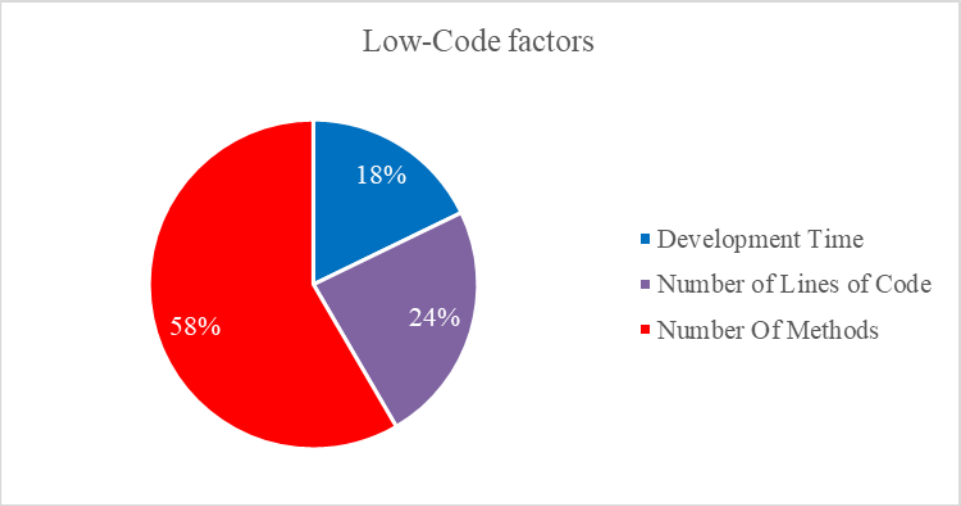

Figure 9 Low-Code factors percentage to each other as a pie chart

- **Development Time: 18%**

Development time contributes the least to the overall evaluation, suggesting that the speed of development is not the primary focus in this context.

- **Number of Lines of Code: 24%**

The number of lines of code has a moderate weight in the evaluation, indicating that code volume is more relevant than development time.

- **Number of Methods: 59%**

The number of methods carries the most significant weight, implying that the structure, modularity, and complexity of the code are crucial factors in evaluating the low-code platform.

Conclusion

For governmental digital transformation projects, the choice between .NET MVC and OutSystems should be consistent with the digital transformation criteria, taking into account factors such as speed of development time, development programming effort represented by the number of lines of code, and the number of methods, while each platform has its strengths and advantages, and the decision should follow the specific needs, business objectives and meeting the digital transformation aspect issues. Therefore, the decision must be a strategic decision guided by a comprehensive understanding of the digital transformation goals and project requirements in parallel. Both systems offer powerful solutions, and the optimal choice depends on striking the right balance between customization, development speed, and long-term scalability to meet digital transformation and project-specific goals.

Consideration Factors:

- **Project Complexity:** Assess project complexity for customization requirements.
- **Development Team Skill Set:** Evaluate the team's expertise and coding proficiency.

- **Business Requirements:** Align with business goals and priorities for speed or customization.

After a thorough comparison between .NET MVC and OutSystems, the nomination for the "Best Fit for Optimal Time and Code Effort Utilization" goes to OutSystems.

For projects prioritizing rapid development, reduced coding effort, and efficient utilization of coding efforts and manpower, OutSystems emerges as the best fit.

Its low-code nature, visual development environment, and built-in scalability make it an optimal choice for projects where time-to-market and productivity are critical factors, as a result of its ability to streamline development, reduce coding effort, and enhance overall efficiency.

References

1. Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2020). Low-Code as Enabler of Digital Transformation in Manufacturing Industry. **Applied Sciences**. 10, 12. [Online] Available: https://doi.org/10.3390/app10010012.

2. Pantelimon, S. G., et al. (2019). Towards a seamless integration of IoT devices with IoT platforms using a low-code approach. In 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). **IEEE**.

3. Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. **In 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**, Portoroz, Slovenia, pp. 171-178. doi: 10.1109/SEAA51224.2020.00036.

4. Dahlberg, D. (2020). **Developer experience of a low-code platform: An exploratory study**.

5. Silva, A. T. D. (2022). **Quality Assurance Framework for Low-Code Development Platforms**.

6. Richardson, C., & Rymer, J. (2014). New Development Platforms Emerge For Customer-Facing Applications. **Forrester**.

7. Waszkowski, R. (2019). Low-code platform for automating business processes in manufacturing. **IFAC-PapersOnLine**, 52(10), pp. 376-381.

8. Lundell, B., & Lings, B. (2004). Changing perceptions of CASE technology. **Journal of Systems and Software**, 72(2), pp. 271-280.

9. Fuggetta, A. (1993). A classification of CASE technology. **Computer**, 26(12), pp. 25-38.

10. Schallmo, D. R., et al. (2018). History of digital transformation. **In Digital Transformation Now! Guiding the Successful Digitalization of Your Business Model**, pp. 3-8.

11. Lopes, B., Amorim, S., & Ferreira, C. (2021). Solution discovery over feature toggling with built-in abstraction in OutSystems. **In 2021 ACM/IEEE International Conference on Model Driven**

Engineering Languages and Systems Companion (MODELS-C), pp. 47–56.

12. Beranic, T., Rek, P., & Heričko, M. (2020). Adoption and usability of low-code/no-code development tools**. In Central European Conference on Information and Intelligent Systems**. Faculty of Organization and Informatics Varazdin.

13. Vincent, P., Iijima, K., Driver, M., Wong, J., & Natis, Y. (2019). Magic quadrant for enterprise low-code application platforms. **Gartner report**.

14. Zolotas, C., Chatzidimitriou, K. C., & Symeonidis, A. L. (2018). RESTsec: a low-code platform for generating secure by design enterprise services. **Enterprise Information Systems**, 12(8-9), pp. 1007-1033.

15. Henriques, H., et al. (2018). Improving the developer experience with a low-code process modelling language. **Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems**.

16. Woo, M. (2020). The rise of no/low code software development—no experience needed? **Engineering** (Beijing, China), 6(9), 960.

17. Rymer, J., & Kony Appian. (2017). **The Forrester Wave™: Low-code development platforms for ad&d pros**, q4 2017. Cambridge, MA: **Forrester Research**.

18. Hecht, L. E. (2019). Low-code platform adoption gets a boost from digital transformation. **The New Stack**.

19. Golovin, D. (2017). **OutSystems as a rapid application development platform for mobile and web applications**.

20. Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021). Characteristics and challenges of low-code development: The practitioners perspective. **In Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)**, pp. 1–11.

21. Pires, M. C. (2014). **Incremental compilation and deployment for OutSystems Platform** (Doctoral dissertation).

22. Elshan, E., Dickhaut, E., & Ebel, P. A. (2023). **An investigation of why low code platforms provide answers and new challenges**.

23. Clay, & Rymer, J. R. (2016). **The Forrester Wave™: low-code development platforms**, Q2 2016. **Forrester**, Washington DC.

24. Salgueiro, J., Ribeiro, F., & Metrôlho, J. (2021). Best practices for OutSystems development and its influence on test automation. In A. Rocha, H. Adeli, G. Dzemyda, F. Moreira, & A. M. R. Correia (Eds.), **Trends and Applications in Information Systems and Technologies** (Vol. 4, pp. 85–95). **Springer International Publishing**.

25. Tisi, M., et al. (2019). Lowcomote: Training the next generation of experts in scalable low-code engineering platforms. In STAF 2019 Co-Located Events Joint Proceedings: **1st Junior Researcher Community Event, 2nd International Workshop on Model-Driven Engineering for Design-Runtime Interaction in Complex Systems, and 1st Research Project Showcase Workshop co-located with Software Technologies: Applications and Foundations** (STAF 2019).

26. Outsystems. (n.d.). **The State of Application Development Report**. Retrieved from https://www.outsystems.com/1/state-app-development-trends/

27. Bowman, D. D. (2018). Declining Talent in Computer Related Careers. **Journal of Academic Administration in Higher Education**, 14(1), 1-4.

28. Yang, W., Zhang, L., & Yu, L. (2018). An analysis of the talents shortage in the present labor market: A case study of China. DEStech Transactions on Economics, **Business and Management**, (icssed).

29. Caspin-Wagner, K., Massini, S., & Lewin, A. Y. (2018). The changing structure of talent for innovation: on-demand online marketplaces. International business in the information and digital age (pp. 245-272). **Emerald Publishing Limited**.

30. Metrolho, J. C., Ribeiro, F., & Araujo, R. (2020). A strategy for facing new employability trends using a low-code development platform. **INTED2020 Proceedings**. **IATED**.

31. Bexiga, M., Garbatov, S., & Seco, J. C. (2020). Closing the gap between designers and developers in a low code ecosystem. **In Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings**.

32. Alsaadi, H. A., et al. (2021). Factors that affect the utilization of low-code development platforms: survey study**. Romanian Journal of Information Technology & Automatic Control/Revista Română de Informatică și Automatică**, 31(3).

33. Phalake, V. S., & Joshi, S. D. (2021). Low code development platform for digital transformation. **In Information and Communication Technology for Competitive Strategies (ICTCS 2020) Intelligent Strategies for ICT**. **Springer Singapore**.

34. Di Ruscio, D., et al. (2022). **Low-code development and model-driven engineering: Two sides of the same coin? Software and Systems Modeling**, 21(2), 437-446.

35. Talesra, K., & Nagaraja, G. S. (2021). **Low-code platform for application development. International Journal of Applied Engineering Research**, 16(5), 346-351.

36. Metrôlho, J. C., Ribeiro, F. R., & Passão, P. (2020). Teaching agile software engineering practices using scrum and a low-code development platform–a case study. **In Proceedings of the 15th Conference on Software Engineering Advances**.

37. Rogojanu, T., et al. (2018). Netiot: A versatile IoT platform integrating sensors and applications**. In 2018 Global Internet of Things Summit (GIoTS)**. **IEEE**.

38. Basciani, F., Iovino, L., & Pierantonio, A. (2014). Mdeforge: an extensible web-based modeling platform. **In Proceedings of the 2nd International Workshop on Model-Driven Engineering on and for the Cloud co-located with the 17th International Conference on Model Driven Engineering Languages and Systems**, CloudMDE@ MoDELS 2014, Valencia, Spain, September 30, 2014 (Vol. 1242). **CEUR-WS**.

39. Hoogsteen, D., & Borgman, H. (2022). **Empower the workforce, empower the company? Citizen development adoption**.

40. Lichtenthäler, R., et al. (2022). A Use Case-based Investigation of Low-Code Development Platforms. **ZEUS**.

41. Martinez, E., & Pfister, L. (2023). Benefits and limitations of using low-code development to support digitalization in the construction industry. **Automation in Construction**, 152, 104909.

42. Bucaioni, A., Cicchetti, A., & Ciccozzi, F. (2022). Modelling in low-code development: a multi-vocal systematic review. **Software and Systems Modeling**, 21(5), 1959-1981.

43. Bratincevic, J., & Koplowitz, R. (2021). **The Forrester Wave™: Low-Code Development Platforms For Professional Developers**, Q2 2021.

44. Basciani, F., Iovino, L., & Pierantonio, A. (2014). Mdeforge: an extensible web-based modeling platform. In Proceedings of the 2nd International Workshop on Model-Driven Engineering on and for the Cloud co-located with the 17th International Conference on Model Driven Engineering Languages and Systems, CloudMDE@ MoDELS 2014, Valencia, Spain, September 30, 2014 (Vol. 1242). **CEUR-WS**.

45. Oltrogge, M., et al. (2018). The rise of the citizen developer: Assessing the security impact of online app generators. 2018 IEEE Symposium on Security and Privacy (SP). **IEEE**.

46. Rowles, D. (2017). Mobile marketing: how mobile technology is revolutionizing marketing, communications and advertising. **Kogan Page Publishers.**

47. Tarasiev, A., et al. (2018). Developing Prototype of CASE-Tool to Create Automation Systems Based on Web Applications Using Code Generation. 2018 Dynamics of Systems, Mechanisms and Machines (Dynamics). **IEEE**.

48. Al-Ashwal, D., Al-Sewari, E. Z., & Al-Shargabi, A. A. (2018). A CASE tool for JAVA programs logical errors detection: Static and dynamic testing. 2018 International Arab Conference on Information Technology (ACIT). **IEEE**.

49. Glover, N., & Dudley, T. (1991). **Practical error correction design for engineers**.

50. Lundell, B., & Lings, B. (2004). **Changing perceptions of CASE technology. Journal of Systems and Software**, 72(2), 271-280.

51. Fuggetta, A. (1993). A classification of CASE technology. **Computer**, 26(12), 25-38.

52. Richardson, C., & Rymer, J. R. (2016). Vendor landscape: The fractured, fertile terrain of low-code application platforms. **Forrester**.

53. Bock, A. C., & Frank, U. (2021). Low-code platform. **Business & Information Systems Engineering**, 63, 733-740.

54. Wang, Y., et al. (2021). The necessity of low-code engineering for industrial software development: a case study and reflections. In 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). **IEEE**.

55. Lugovsky, V. (2021). A Guide To Low-Code/No-Code Development Platforms In 2021. **Forbes, Council Post**.

56. Tariq, H. (2021). Low-Code Versus No-Code And The Future Of Application Development. **Forbes, Council Post**.

57. Lethbridge, T. C. (2021). Low-code is often high-code, so we must design low-code platforms to enable proper software engineering. **In International Symposium on Leveraging Applications of Formal Methods** (pp. 202–212).

58. Gartner Inc. (2022). Enterprise LCAP (Low-Code Application Platforms) Reviews 2022 | Gartner Peer Insights. **Gartner**.