iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** XII **Month of publication:** December 2021

DOI: https://doi.org/10.22214/ijraset.2021.39328

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue XII Dec 2021- Available at www.ijraset.com

Analysis of Low Code-No Code Development Platforms in comparison with Traditional Development Methodologies

Shreyas Shridhar[1], Siddharth Bose[2]
[1]*Student, Department of Computer Engineering, SIES GST, University of Mumbai,*
[2]*Student, Department of Electronics & telecommunication, PCCOE, Pune, Savitribai Phule Pune University(SPPU)*

Abstract: ***This paper examines the overview of low-code/no-code development platforms in comparison with traditional development methodologies and examines the benefits and limitations of the same. For several decades, businesses have had multiple options when they demanded new information systems. They could develop a new system using in-house developers, or they could order a system from an external merchant. This offers a close fit to business obligations. However, nowadays, there is a new alternative that is becoming increasingly prevalent. Low code/no code (LC/NC) applications can cater to business requirements efficiently, can be implemented instantly, and the cost is much less than systems developed in-house. Few, if any, programming skills are required.***

Keywords: Traditional development, No code development, low code development, Low code No code development, Software development life cycle (SDLC)

I. INTRODUCTION

Low code/no-code development platforms provide a familiar and apt solution to business specifications at a fraction of a cost in contrast to traditional development equivalents and also is twice if not faster as well. With the help of pre-built modules and an intuitive interface that usually incorporates drag and drop functionality for setting up the process models and framework of the app, it makes app development straightforward, faster and extensible. It also subdues the difficulty of learning various programming languages, technology stacks and helps people with less experience in the technology domain to develop and create applications/products for their businesses without much hassle. However they are not meant to be a replacement for traditional development practices or methodologies in any way, rather they serve as an alternative for specific use cases and business requirements by organizations.

II. METHODOLOGY

In this paper, we intend to analyze various aspects of low code/no-code development platforms and compare them with traditional development methodologies in terms of agility, cost of development, functionality, customization and many other aspects.
The different stages of traditional development methodology are shown in the given figure below

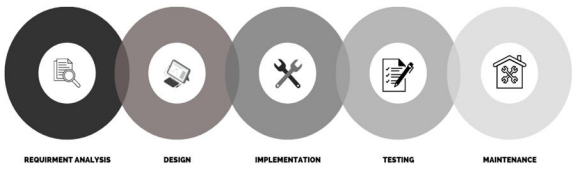


Figure 1. Traditional Development Life Cycle

No-Code/Low Code Development Platforms work in a different way. They tend to have pre-built modules for different functions and these modules are integrated and configured together to develop applications and software.

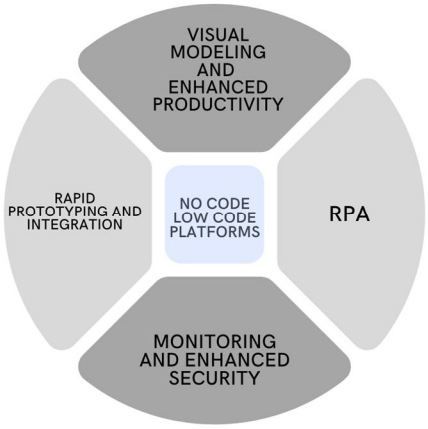


Figure 2. No-Code/Low Code Development Platform Overview

III.OVERVIEW

A. *Ease of Development*

1) *Low-Code/ No-Code Development Platforms:* These platforms typically possess a small set of tools, which helps one to develop various varieties of applications without considerable trouble. Application development using (LC/NC) approach becomes way more manageable for non-technical personnel without the knowledge of skills in this domain. The prominent advantage of low-code no-code development, particularly for proficient developers, is that it's quick. Prebuilt modules reduce the time taken to implement application functionality, so developers can spend time on tasks that need more originality or that have greater precedence for the business. Low-code development can also help developers integrate a function with an external platform without learning about the ins and outs of that external platform.
2) *Traditional Development Platforms:* Traditional app development utilises a well-defined, established development process. With programming, you normally expect the implementation of Agile methodology and DevOps best practices. The DevOps procedures ensure your application may be easily managed, seamlessly updated, and delivered to the general public. However, to develop a standard app, developers must be at home with the ecosystem, frameworks, technologies and technology stack utilized by an organisation. there's a good range of tools and technologies to choose from. Developers encounter software errors and bugs which may be a nuisance to repair.

B. *Cost*

1) *Low-code/ No-code Development Platforms:* Low-code/no-code development platforms support businesses to develop applications for immediate use speedily and at a reduced expense. Traditional development demands an in-house team or outsourced software development. With low-code development, one doesn't need a team: the organization can hire part-time developers to work on an appropriate project. It also depends on the subscription costs of the platforms used. Usually way cheaper than hiring professional developers who demand higher pay in accordance with their skills and knowledge.
2) *Traditional Development Platforms:* For traditional app development, businesses must hire developers who are well versed with the technology stack and pay them. Supplemental expenses of developing an app with the traditional approach include costs for App development tools and libraries, Third-party service integration and Infrastructure services like servers, data storage and content development networks (CDN). These services should be established by companies on their own or outsourced to cloud services like AWS, GCP or Azure. Although costs will vary depending on the extent of the project and particular requirements of the organization, it still is significantly costlier than low-code no-code platforms.

C. Maintenance

1) *Low-code/ No-code Development Platforms:* With no-code platforms and their pre-built and pre-tested modules, implementing changes is more straightforward and efficient. Plus, the risks of incompatibility are low. Low-code platforms being aPaaS (application platform as a service), are typically governed and managed by the organisation that maintains the platform. This simplifies the procedure for maintenance as all the updates and enhancements to the software are controlled by the hosting company rather than the business advocating the service. This assists businesses with inadequate or non-existent IT departments because it enables them to take advantage of well-maintained software without the requirement to spend on extra resources.
2) *Traditional Development Platforms:* With traditional development, there's a requirement for in house or third party development teams for application support. Updating applications developed in a traditional approach needs IT teams, to feature new functionalities and fix bugs. With conventional programming, maintenance may be a big headache as the updates or revision would require a dedicated in-house or third-party development team to implement changes. Furthermore, software revision could be a cumbersome process that the teams should plan, design, and test, followed by training users on new changes.

D. Agility

1) *Low-code/ No-code Development Platforms:* These platforms are much simpler to use thanks to their intuitive interface and also because of their drag and drop features. Unlike custom development, there's no requirement for coding to develop the applications/software since one can simply utilise the drawing techniques for the same.
2) *Traditional Development Platforms:* Traditional development platforms usually have an advanced configuration system, making them less agile. Further, complex coding requires a lot of time to learn and implement accurately.

E. Reusability/Robust

1) *Low-code/ No-code Development Platforms:* Distinguished low-code systems provide OOTB (out-of-the-box) functionality which precludes the responsibility to develop kernel modules for apps from scratch. Gist modules like customer service management, data management, etc. are pre-built in some platforms. Multi-device adaptability is one of the most sought-out characteristics of low-code/no-code platforms. Cross-platform compatibility also enables users to make apps that could run on all platforms and devices. An added distinguished feature of low-code development is the use of pre-configured modules and functionality for applications.
2) *Traditional Development Platforms:* Traditional development allows code reusability. This not only increases productivity but also includes a definite impact on the condition and maintainability of software products. Reusable components are often developed using frameworks like React Native, AngularJS etc. Still, it takes time to develop these components to be used later in contrast to low-code systems which have built-in reusable components thus saving time. Although traditional development allows code reusability, it doesn't contain pre-configured modules and functions for apps like low-code/no-code systems. These modules and functionality must be created beforehand for reusability.

F. Advantages

1) *Low-code/ No-code Development Platforms*

a) *Economical:* With low-code/no-code, the event time and expenses get decreased considerably, and enterprises needn't hire as many developers. Additionally, once you choose no-code development for your commodity, one enjoys the flexibility that the Agile method renders as these platforms support versioning for any edits that were addressed within the application.
b) *Agility:* It empowers organizations to advance and deliver revisions quicker. the talents required for making low-code/no-code products are lesser compared to custom application development because they supply configuration-based design expertise.
c) *Improved Risk Management:* Enterprises can make modifications quickly just in case of a no-code/low-code approach to remain compliant with invariably evolving regulations Further, all of the configuration processes during this case, including code review, compiling, and debugging, are much accelerated compared to traditional development methods.
d) *Augmented customer Experience:* Low-code/no-code platforms automate multiple transactions that are essential for user experience. The promptness in app improvement and robust business process features help in developing far better apps
e) *Improved Productivity:* Low-code/no-code platforms help bridge the gap between IT and business teams, allowing them to unravel real issues that impact the corporate. Using this approach, business organisations can create their own applications without having to attend to developers. It eliminates the requirement for complex code that increases access to more team members, resulting in enhanced productivity.

f) Enhanced Security: If there's a patch or security upgrade, it'll be automatically installed within the low-code/no-code platforms rather than the developer going into the code to install the patch.

2) Traditional Development Platforms

a) Endless Functionality and Customization: Companies can alter custom code any way they require. Nearly any feature is often built and practically any integration is feasible. While making a custom app, you're absolved to choose any technology, tooling, hosting, and APIs of your choice, whatever gets the duty done and whatever works best to attain the required functionality.

b) Complete Ownership and Control: Absolute control over every aspect of the software that's developed is another perk of traditional development. there's complete control of the app's architecture, security, and integrability, among other things since Businesses choosing custom apps own the ASCII text file. Developers haven't any limitations or restrictions over how their code is written and the way applications are structured.

c) Flexibility: Traditional application development can be accomplished using any language and development platform.

d) Deploy Anywhere: Apps developed with a standard approach is deployed to whichever destinations developers value more highly to support.

e) Complex Functionality and Business Logic: Developers don't seem to be limited to pre-existing features and modules. this can be the most important advantage of Traditional application development compared with low-code/no-code development. Low-code/no-code systems are good for little scale applications except for bigger projects that need scaling and complicated custom logic, Traditional application development is the best.

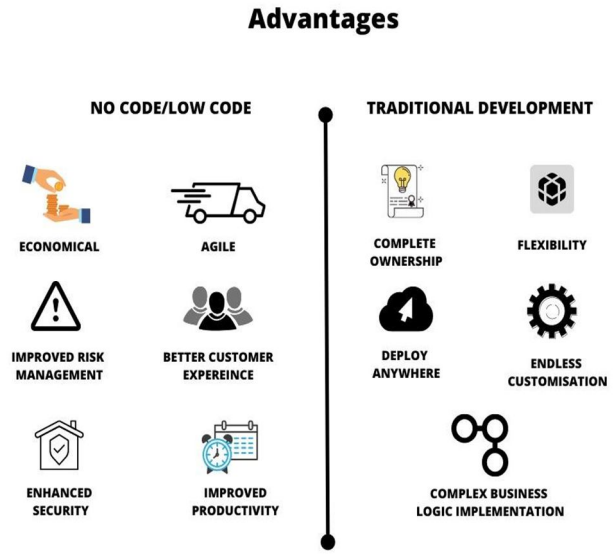


Figure 3. Advantages

G. Disadvantages

1) Low-code/ No-code Development Platforms

a) Third-Party Reliance: With a low-code or no-code platform, one would mostly depend upon the merchant to mitigate risks and secure vulnerabilities, together with changing their schedule for updates to align the vendor.

b) Limited Customisation: Low-code/no-code development platforms usually offer surprisingly limited options for companies to develop customized or bespoke software applications/products.

c) Limited Integration Options: Options for creating applications on a low-code/no-code development platform limits integration options for developers. This might be a notable challenge for businesses with legacy systems that are vital to their business procedures.

d) Scarcity of Developers: Since low-code/no-code development is not an easily seen area of expertise, it's challenging for companies to get developers who are skilled in low-code/no-code development.

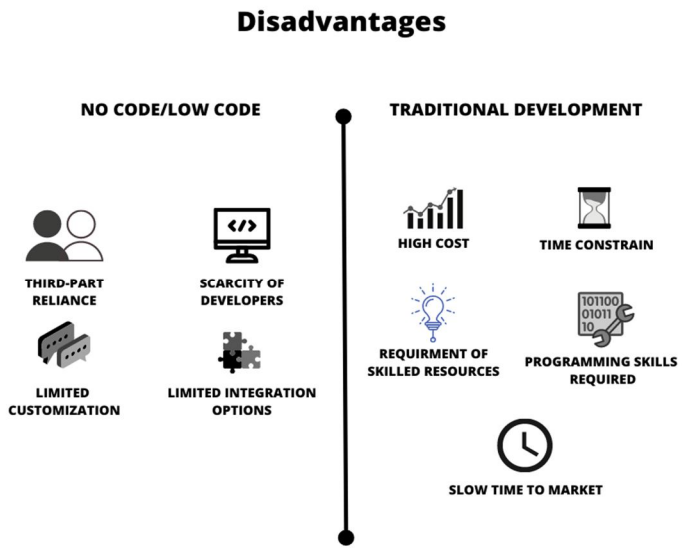


Figure 4. Disadvantages

2) *Traditional Development Platforms*

a) *Higher Costs:* There's no way around it: you'll build in-house, outsource, or near-source, but traditional custom programming requires serious budgeting.

b) *Time Constraints:* Manual programming requires longer than auto-generated code from low-code/no-code tools. Also, manually making changes within the code to support varied platforms and develop sockets and other interfaces between services and applications are prolonged in traditional application development.

c) *Requirement of Skilled Resources:* An entire team of developers, QA engineers, designers, and other specialists are required to make a custom application. Additionally, hiring an app developer that treats your product as their own could be a task in itself. Coding may be a hard skill to master, and it's unlikely that an entrepreneur would have enough time to check the craft to the purpose where he/she is comfortable making functioning apps as and when required.

d) *Programming Skills Required:* Traditional application development demands more expertise with a programing language or development platform. People have to be trained with appropriate skill sets before starting a conventional application development project.

e) *Comparatively Slow Time to Market:* Compared to no-code programming, custom app development takes longer — somewhere between a pair of months and a year. Even when programmers use pre-existing components to hurry up development, the majority of an app is created manually and takes time.

IV. CONCLUSIONS

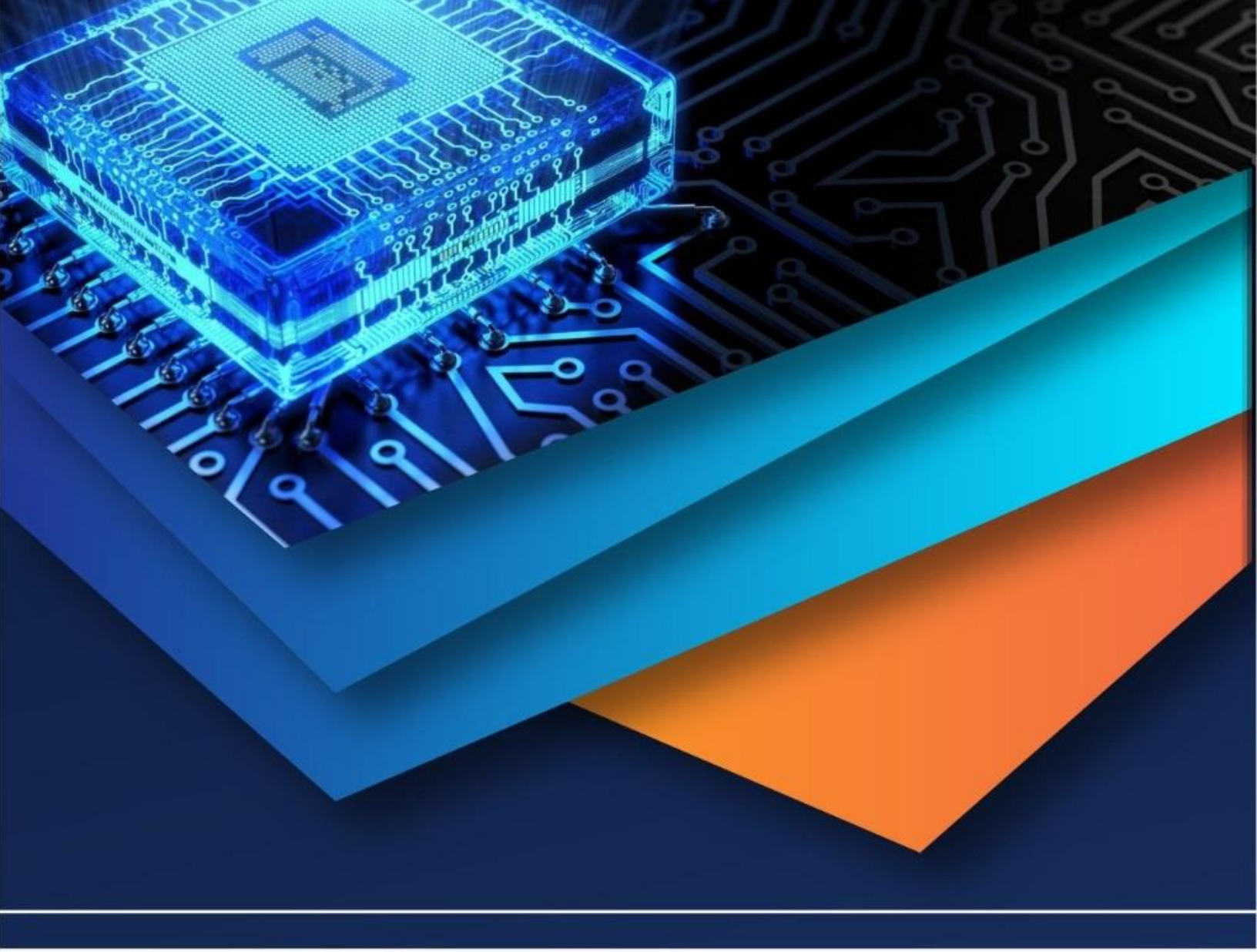
From the above analysis, we are able to conclude that both No code/Low Code Development platforms and traditional development methodologies have their own advantages and disadvantages. depending upon the business requirement, an appropriate technology stack could be chosen. That being said, it's crucial to keep in mind that (LC/NC) platforms aren't meant to totally eliminate the necessity for traditional/custom development methodologies. They're largely meant to equip each member of a collaborative team, with the required tools and resources they could use. It goes without saying that developers can't be replaced.

V. ACKNOWLEDGEMENT

Working on "Analysis of Low Code/No-Code Development Platforms in comparison with Traditional Development Methodologies" helped me understand the nuances of both technologies. It was a pleasure to research in this field as it would impact the world of development in the imminent future and enable more non-technical people to enter the development world.

REFERENCES

[1] Den Haan J. Introducing AI-assisted development to elevate low-code platforms to the next level [Internet]. Boston: Mendix; 2018 Jun 19 [cited 2020 Jun 19]. Available from: https://www.mendix.com/blog/introducing-ai-assisteddevelopment-to-elevate-low-code-platforms-to-the-next-level/.
[2] Marcus Woo, The Rise of No/Low Code Software Development—No Experience Needed? https://www.researchgate.net/publication/342951159_The_Rise_of_NoLow_Code_Software_Development-No_Experience_Needed
[3] Appian - Crunchbase Company Profile & Funding. (n.d.). Crunchbase. Retrieved July 10, 2021, from https://www.crunchbase.com/organization/appian
[4] Manifesto for Agile Software Development. (2021). Agile Manifesto. http://agilemanifesto.org/
[5] The Forrester Wave™: Low-Code Development Platforms For Professional Developers, Q2 2021. (n.d.). Forrester. Retrieved July 7, 2021, from https://www.forrester.com/report/The+Forrester+Wave+LowCode+Develo pment+Platforms+For+Professional+Developers+Q2+2021/-/ERES161668
[6] No-code/Low-code vs. Programming: How to choose? from https://topflightapps.com/ideas/no-code-low-code-vs-traditional-development/
[7] A practical take on low-code vs. Traditional development
from https://searchsoftwarequality.techtarget.com/tip/A-practical-take-on-low-code-vs-traditional-development



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY