Scholastic Research Publications, SRP Group®
International Journal of Humanities Science Innovations and Management Studies
Volume 1 Issue 1, Pg. No. 43-56, November-December, 2024
IJHSIMS-V1I1P104, Doi: https://doi.org/10.64137/30508509/IJHSIMS-V1I1P105

Original Article

A Review on Model-Driven Development with a Focus on Microsoft PowerApps

Aniruddha Arjun Singh Singh[1], Rami Reddy Kothamaram[2], Dinesh Rajendran[3], Venkata Deepak Namburi[4], Vetrivelan Tamilmani[5], Vaibhav Maniar[6]

[1] ADP, Agile Team Leader.
[2] California University of management and science, MS in Computer Information systems.
[3] Coimbatore Institute of Technology, MSC. Software Engineering.
[4] University of Central Missouri, Department of Computer Science.
[5] Principal Service Architect, SAP America.
[6] Oklahoma City University, MBA / Product Management.

Received Date: 22th September 2024 Revised Date: 11th October 2024 Accepted Date: 3rd October 2024 Published Date: 27th November 2024

Abstract - The paradigm shifts in software engineering that Model-Driven Development (MDD) and low-code/no-code (LCNC) systems, including Microsoft PowerApps, offer is a focus on high-level models in the form of visual development environments instead of code-centric development models. MDD is concerned with the development of abstract models of system behavior, structure and logic to be converted into executable applications using automated tools. Research-based MDD techniques investigate frameworks, application domain languages, and formal methodologies, and commercial tools, such as Mendix, IBM Rational Rhapsody, and PowerApps, offer viable application development platforms. Low-code platforms go further to allow citizen developers and professional programmers to create applications using visual interfaces with limited code. Although they have their benefits, such as enhanced abstraction, productivity, reusability, and platform insensitivity, such things as tool dependency, performance constraints, complexity in debugging, constraints in delegation, and long-term maintainability remain. Effective data management, integration with diverse sources, and attention to both developer and user experience are critical for successful adoption and the development of scalable, enterprise-grade applications. Future trends indicate the growing adoption of AI-assisted model-driven tools, automated testing, and enhanced integration with DevOps pipelines, which further accelerate software development and innovation. Understanding both the technical and organizational implications of MDD and low-code platforms is essential for maximizing their benefits in real-world enterprise environments.

Keywords - Software Engineering, Developer Experience Low-Code / No-Code (LCNC) Platforms, Model-Driven Development, Microsoft PowerApps.

1. Introduction

The development of software has undergone considerable changes over the last several decades, evolving into high-level, model-driven software development, as opposed to low-level, hand-written procedures. The traditional software development process is also usually characterized by the use of a lot of code, debugging, and maintenance, which is time-consuming and prone to error. The model-driven development (MDD) is a software development paradigm that employs models as the main development artefacts [1]. Modelling tools in software production are now common and popular now that they have matured to a stage where their availability and execution is well known even by non-technical users [2]. Many modelling tools, such as business process management systems and low-code/no-code platforms, have been created to support MDD in the last twenty years [3]. The major goals of these modelling tools and platforms are to increase productivity and decrease time to market by enabling development at a higher degree of abstraction and applying concepts closer to the issue domain. This is in contrast to programming languages, which dictate lower levels of abstraction.

The application of MDD principles is mainly implemented by Microsoft Power Apps, namely, its model-driven app capabilities [4]. The developers can create applications within a limited period of time with a low-code platform where they can specify data models, relationships and business logic to create applications with minimal hand-coding [5]. In addition to speeding up development, this method guarantees consistency and maintainability across all applications.

Microsoft has developed PowerApps as a No-Code programming option. The new Microsoft Access replacement is PowerApps. Microsoft Access is a web app and database management application. Office 365, Microsoft's most recent software as a service platform, serves as a foundation for PowerApps. Users could only access data stored in its own database, but PowerApps are compatible with a wide range of external systems. With PowerApps, and access numerous data sources, which gives more freedom [6]. The PowerApps NoCode development environment makes it easy to integrate PowerApps with current enterprise architecture by providing access to these data sources.

One collection of tools that both programmers and non-programmers can use is the low-code platform. It permits the rapid development and distribution of commercial applications with little coding work, necessitating the bare minimum in terms of installation, configuration, training, and implementation [7]. Large corporations in the conventional sense manage their infrastructure and employees with enterprise-level software [8]. The development of business solutions that save time and money on consumer-beneficial chores has been largely driven by computer applications [9]. Customers in a conventional company setting have limited say over the degree to which the software they purchase may grow with their needs. An increasing number of businesses are realizing that low-code solutions are the way to go when it comes to developing mission-critical software.

This review explores the history and application of Model-Driven Development, with particular interest in Microsoft Power Apps. With the examination of the principles of MDD and its place within the Power Apps ecosystem, the review is likely to contribute to the provision of a detailed study of how model-driven solutions are shaping the future of software development.

1.1. Structure of the Paper

This paper is organized in the following way: Model-Driven Development (MDD) is defined and introduced in Section II. Section III discusses low-code platforms, their concepts, comparisons, and impact on developer experience. Section IV examines data management within model-driven Microsoft PowerApps. Section V highlights key challenges in model-driven and low-code development. Section VI reviews related literature in this domain. Finally, Section VII concludes the paper with key insights and directions for future work.

2. Model-Driven Development (MDD)

Software engineering methodology known as MDD places an emphasis on abstract models as main artefacts in the process of creating applications. Rather than writing source code directly, developers construct models representing system behavior, structure, and logic and then leverage automated tools to transform them into executable code. MDD is fundamentally based on raising the level of concept so that developers can focus on business requirements and system design without worrying about the details of implementation [10]. MDD provides consistency, automates manual coding, and accelerates application development.

2.1. Model-Driven Development (MDD) Approaches

MDD encompasses a range of approaches designed to enhance the level of software engineering abstraction, with models serving as the primary artifacts. Models are written by developers to express the structure, behaviour and constraints of the system in place of writing actual source code that is, in turn, sliced into executable implementations through automated or semi-automated tools. Overall, the approaches of MDD can be classified into two categories: research-based approaches and marketable solutions.

2.1.1. Research-Oriented Approaches

Academic and research-oriented approaches primarily focus on exploring new methodologies, frameworks, and transformation techniques. These include:

- MD2: One model-driven framework for creating data-driven mobile applications that function across platforms is MD2. The application is initially described in MD2 using a textual DSL in a platform-

independent paradigm. After that, a code generator takes the PIM and turns it into platform-specific source code, app logic, and the files needed to build the user interface.

- MobML: One collaborative framework for creating data-intensive mobile apps is Model-Based Mobile Language (MobML) [11]. Three parts make up the framework: modelling languages, code generators called synthesizers, and a mechanism for collaboration.
- Mobile Multimodality Creator (MIMIC): Mobile Multimodality Creator (MIMIC) is one model-driven framework that automates the process of modelling and generating code for multimodal mobile applications. The foundation of MIMIC is M4L, a state-machine-based language for describing input/output multimodal mobile interfaces.
- Applause: A toolbox for making mobile apps that work on several platforms is Applause. An application description language (DSL) and a suite of code generators are the building blocks of the framework. These generators use the models to create native apps for different platforms, including Android, iOS, Windows Phone, and Google App Engine. According to expectations, Applause2, the upcoming framework version, would address every facet of a mobile app.
- Model Transformation Techniques: Transformations are defined between source models and target models (e.g., UML class diagrams to C# code). This requires explicit meta-model definitions for both input and output models.
- Domain-Specific Languages (DSLs): Research efforts have often emphasized the design of DSLs that capture the semantics of a specific problem domain, thereby improving expressiveness and reducing complexity for developers.
- Formal Methods Integration: Some approaches integrate formal verification methods into the modeling process to ensure correctness and reliability.
- Experimental Studies: Research prototypes are often validated through controlled experiments comparing MDD with code-centric development (CcD), evaluating factors such as correctness, efficiency, maintainability, and user satisfaction.

2.1.2. Commercial Solutions

Industrial and commercial solutions are more focused on providing practical tool support to improve productivity and software quality. These include:

- Mendix App Platform: Developers and business users alike can create and launch multi-channel apps using the Mendix App Platform, which is an MDD platform. A collection of common services makes up the platform's core library, and users can build and reuse widgets that incorporate the libraries they require for an optimal user interface and experience.
- IBM Rational Rhapsody: Rational Team Concert and IBM Rational Rhapsody 5 work together to let developers model Android apps and see the API for the Android framework right in Rational Rhapsody.
- WebRatio Mobile Platform: A model-driven approach is at the heart of WebRatio Mobile Platform 6, a platform for creating mobile applications. This tool is based on the IFML standard, specifically the mobile-extended version. Three smooth options are available in WebRatio Mobile.
- Appian Mobile: Appian Mobile is a crucial part of the Appian BPM Suite and was built for mobile apps using the write-once, deploy anywhere architecture. The Appian Mobile Process Modeller makes it easy for application designers to create mobile process patterns with a simple drag-and-drop interface.
- Modeling Tools: The popular Modeling Tools, such as IBM Rational Software Architect, Enterprise Architect and MagicDraw, are modeled on the modeling languages of UML and SysML and generate code as models.
- Model-Driven Architecture (MDA): The Object Management Group (OMG) created MDA, which advocates for the systematic transfer of models from one platform to another based on the assumptions made by Platform-Independent Models (PIMs) and Platform-Specific Models (PSMs).
- Low-Code/No-Code Platforms: Mendix, OutSystems, and Microsoft Power Apps are just a few examples of the many commercial modern MDD systems that have moved to low-code or no-code platforms. Users with less technical knowledge are able to contribute to software development on these platforms, which allows for rapid application development through automation of a large percentage of the coding process.
- Integration with DevOps Pipelines: The use of commercial MDD solutions with CI/CD pipelines is on the rise, which helps with agile and iterative development.

2.2. Benefits and Challenges of MDD

MDD offers several advantages:

MDD has become a well-known research and practice topic, and the term has frequently been used in research studies to compare it with traditional Code-centric Development (CcD) [12]. The development technique (MDD vs. CcD) is typically selected as the independent variable in experimental investigations, while correctness, efficiency, and user satisfaction are commonly associated with the dependent factors [13]. The efficiency of a system is defined as the ratio of its correctness to the time it takes to develop it. The constructs of user satisfaction, such as PEOU, PU, and ITU, are typically measured through the TAM. Automated testing or correction templates are usually used to determine correctness.

2.2.1. Benefits of MDD

MDD is said to have a number of reported benefits:

- Increased abstraction: Developers do not work on low-level code; instead, they work on system design and business requirements.
- Greater productivity and efficiency: Automated model-to-code transformations save labor, which usually results in a shorter time to develop.
- Consistency and Standardization: Standardized models guarantee consistency in requirements, design and implementation.
- Enhanced communication: Models provide a visual representation that can help technical experts and non-technical stakeholders work together more effectively.
- Early detection of errors: validation during the modeling phase enables the detection of errors prior to implementation, which reduces the cost of correction.
- Reusability and scalability: Models can be used across other projects and be scaled by the changing needs.
- Platform independence: One more way transformations help with portability is by making it possible to use the same model on different systems.
- Positive user perception: In controlled studies, participants tend to perceive MDD as more useful and as easier to use than code-centric methods.

2.2.2. Challenges of MDD

However, MDD adoption is not without limitations, and several challenges have been consistently reported in the literature:

- Tool dependency: Effective MDD makes the intensive use of specialized modeling and transformation tools, which are not necessarily interoperable.
- Expensive implementation: Training, the purchase of tools, and integration in the current processes demand a large investment.
- High learning curve: To gain benefits, developers have to get acquainted with modeling language (UML or DSLs), and this process may slow initial productivity.
- Complicated transformation: It is hard to make model-to-code transformations accurate, efficient, and maintainable.
- Integration problems: Easing of MDD into existing systems and other development environments may pose a challenge.
- Risk of over-engineering: On small projects, it may prove to be costly to construct and support large-scale models, which might not be worth it.
- Divided user satisfaction: Although most users enjoy the abstraction and productivity, others consider modeling less intuitive than direct code, especially for beginners.
- Sluggish adoption in industry: Largely, industrial adoption of MDD remains on a small scale relative to traditional development methods, even though there has been extensive research interest.

3. Low-Code Platforms: Concepts and Trends

Rapid application development technologies like low-code and no-code platforms have recently arisen, allowing organizations to speed up their digital transformation and rely less on traditional software engineering resources.

3.1. Low-Code/No-Code Platforms Overview

Recently, development platforms that automate development chores and decrease code have begun to gain widespread adoption; now, with the COVID-19 outbreak, that adoption is only increasing. A bundle of tools for programmers and non-programmers is one way to describe these low-code platforms [7]". Thanks to this, commercial applications may be quickly created and delivered with minimal coding work, and environments can be set up, trained, and implemented with minimal effort as well [14]. As a result of streamlining the code-to-goal process, developers may now focus on rapidly constructing new functionality rather than spending time figuring out the intricate coding needs of a system. Automating and standardizing code makes it easy to implement in a visual coding environment, which is how this is accomplished. A Comparison of Low-Code and No-Code Approaches:

- **Low-Code:** Quickly design apps with the use of automated workflows, pre-built components, and drag-and-drop interfaces; minimal coding expertise is required. Ideal for professional developers or technically inclined individuals.
- **No-Code:** Established with the intention of empowering average individuals, also known as "citizen developers," to build their own applications from the ground up with no coding expertise and a focus on visual tools.

3.2. User Experience

The interaction and adoption of low-code platforms by users are directly influenced by the user experience (UX), which is a critical component. The worldwide standard on the ergonomics of human-system interaction refers to an individual's impressions and responses that are a result of an object, system, or service's actual or projected use as UX [15]. The way a person uses a user interface to engage with various products, systems, services, and objects is another possible definition of user experience.

This definition suggests that although general experience may encompass events, physical environments, social experiences, and artistic experiences, it is not user experience unless they occur within a designed, man-made user interface. In the case of low-code platforms, UX focuses on the ease and simplicity with which users can navigate the platform's graphic development tools, procedures, and auto-generated products to achieve their goals.

3.3. Software and Designing for Appropriation

Software and Designing to Appropriate is the process of making software change in ownership to suit and suit personal context, needs and preferences, usually in ways that were not originally intended by the developers. The latter is especially critical with low-code platforms since the developers (amateurs and professionals) can carve their own experiences by customizing resources that better fit their needs. Appropriation is not limited to technical modification and it is more about the personal and contextual use of software. Conversely, designing with appropriation focuses on intentional designing of systems that promote user extensions, adaptation and customization of applications in the unique ways [16]. This point of view plays a pivotal role in a low-code environment as flexibility and adaptability are the key to innovating and guaranteeing long-term adoption.

3.4. Developer Experience in Low-Code Platforms

Developer Experience (DX) is a term used in Low-Code Platforms to describe an extension of the user experience (UX) in software development with a focus on programmers as both makers and consumers of applications [17]. As opposed to the traditional UX that focuses on the interaction with the end-users, DX encompasses all that pertains to the software development, as procedures, software models and routines of a cycle [18]. The development of low-code platforms and their tools is dynamic and requires developers to be constantly flexible, and DX is essential to productivity, adoption, and the quality of applications. Regardless of its significance, the body of research on DX in low-code platforms is still small, and little research is done on how developers engage with visual developments or model-driven development tools. The comprehension of DX plays a key role in achieving the low-code processes optimization, increasing developer satisfaction, and improving the development and maintenance of applications.

3.5. Market Trends and Adoption

The growing need in digital solutions, the lack of highly trained developers, and the necessity of a shortened time-to-market have all led to the rapid expansion of low-code/no-code tools on a worldwide scale. Industry reports

by Gartner and Forrester have indicated that the world is witnessing a massive adoption of LCNC platforms by businesses, and organizations are becoming more and more dependent on these platforms as a way of achieving digital transformation. The adoption trends of these tools in industries are different: in the financial sector, these tools are applied to the automation of reporting, risk management dashboards, and customer onboarding applications, in healthcare sectors, they are applied to patient portals, appointment booking systems, and internal workflow automation, in educational settings, they are applied to learning management systems, course enrollment systems and analytics dashboards, and in the government, they are applied to citizen services, permit application processes, and citizen Web portals. On the whole, these tendencies showcase the flexibility of LCNC platforms to solve domain-related issues and improve efficiency, innovation, and service delivery in various sectors.

4. Data Management in Model-Driven Microsoft Powerapps

PowerApps is a part of the Power Platform ecosystem of applications launched by Microsoft in 2015, including Power BI, Power Automate, and Power Virtual Agents. Originally designed as a mobile app builder, PowerApps has gradually grown to be an entire low-code application development platform of its own, as a result of the need to support citizen development and fast enterprise solutions. One of the crucial steps in this development was the addition of Microsoft Dataverse (previously known as Common Data Service) that made it possible to create model-driven applications that had a strong connection with services of Dynamics 365 and Azure. This has seen Power Apps become a foundational app to MDD, and it offers enterprise capabilities.

The application architecture in model-driven apps is based on data management. The data model describes the entities, attributes, relationships and business rules that are going to be consistent and that ensure governance throughout the system. PowerApps facilitates seamless connection to multiple data sources, including Dataverse, SharePoint, SQL Server, and Microsoft 365, thereby enabling organizations to create robust data-centric applications [19]. However, despite this flexibility, developers face challenges related to delegation limits, query scalability, performance optimization, and security enforcement.

4.1. Types of Applications in PowerApps

Here are some types of applications in PowerApps are as follows:

4.1.1. Canvas Apps

Canvas apps provide a drag-and-drop design environment, allowing developers and citizen developers to exercise complete control over the application's layout and user experience. These apps are highly customizable, making them particularly suitable for use cases where flexibility in design and interface optimization is critical. Additionally, canvas apps can link to more than 300 data sources, such as third-party APIs, Microsoft SharePoint, SQL Server, Office 365, Dynamics 365, and SQL Server. This extensive connectivity enhances their applicability for creating lightweight, task-specific, and user-centric applications.

4.1.2. Model-Driven Apps

A data-first approach is the hallmark of model-driven apps, as opposed to canvas apps. These apps are developed automatically according to the data model, and they are built on Microsoft Dataverse. The dynamic creation of views, dashboards, and forms guarantees a uniform and standardised user experience [20]. Model-driven apps are particularly effective for complex business processes such as CRM, ERP, and other structured workflows. Their ability to enforce business rules, role-based security, and process automation makes them highly suitable for enterprise-scale deployments.

4.1.3. Portals

PowerApps portals extend the scope of applications by enabling the creation of external-facing websites. These portals allow customers, partners, and suppliers to securely interact with organizational data stored in DataVerse. By bridging internal systems with external stakeholders, portals play a crucial role in facilitating collaboration, providing self-service functionality, and ensuring secure data sharing. This feature significantly broadens the impact of PowerApps beyond internal enterprise users, supporting digital engagement strategies across organizational boundaries.

4.2. Data Sources and Integration in Powerapps

The data sources used by Power Apps for storing and retrieving information are tables. A data source, either local or linked, such SQL, SharePoint, or Excel, can be used to build the table. Although other data sources are also acceptable, tables remain the most popular. For instance, can use organization's data to build the app need with Microsoft 365's many connectors [21]. Power Apps' Patch, Collect, and Remove functions allow users to modify the tables inside a data source. Using a OneDrive Excel workbook as an example, can see how straightforward and simple it is to integrate a data source into application. Here is how Power Apps integration works: A table in an Excel workbook can be used to read and publish data after the data source is connected with Power Apps (Figure 1).

ID	FirstName	LastName
1	John	Watson
2	Mary	Smith
3	George	Doe

Fig. 1 Example of an Excel Table

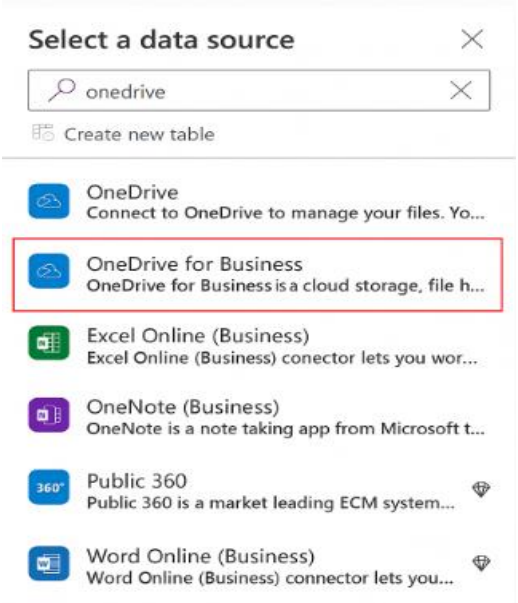

Fig. 2 Data Source Selection in Power Apps Studio

A data source can easily be incorporated into a Power App. Once one is ready to choose the preferred data connector, in this case, OneDrive for Business, the table or dataset inside it can be selected and used in the app (Figure 2). Functions such as Patch, Collect and remove can be used to alter tables in a data source.

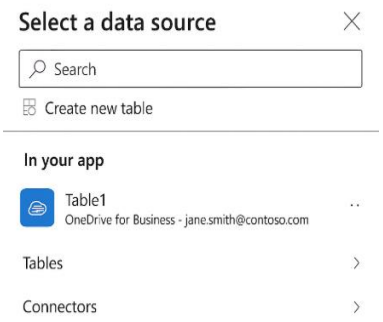

Fig. 3 Data source selection within a Gallery in Power Apps studio

Fig. 4 Example of Data Visualization in A Gallery

Once connected, the data can be visualized using components such as Galleries, Forms, or Data Tables, enabling efficient interaction with the information. For example, an Excel table saved in OneDrive can be connected to a Gallery in Power Apps to display its contents, and modifications in the app are automatically reflected in the original data source (Figures 1, 3, and 4).

This integration process enables the creation of dynamic, data-driven applications with minimal coding effort, while leveraging the rich ecosystem of Microsoft connectors to meet various organizational needs.

4.3. Data Source Limitations and Considerations

The selection of a data source in Microsoft PowerApps requires careful evaluation of its inherent limitations and constraints. For instance, while Microsoft Lists (SharePoint) supports approximately 15 data types, Microsoft Dataverse provides more extensive data modeling capabilities, offering richer functionality for complex business applications. Beyond the choice of data types, several additional factors must be considered, including security requirements, data storage capacity, and data movement capabilities. One of the most important elements is the idea of delegation that defines the way the data queries are handled between the client and the server [22]. A query that retrieves or makes amends to a specific number of records is only possible through one action because of the limitations of delegation of the data sources such as SharePoint and Excel. These constraints can significantly affect the speed of the applications, scalability, and user experience when handling huge amounts of data [23]. It is therefore important to consider these limitations prior to their use and ensure that they are aligned to the application requirement. Proper assessment ensures that the selected data source meets short term functional needs and the long-term needs of the model driven application which include scalability, performance and governance of the application.

5. Challenges in Model-Driven and Low-Code Development

The advantages of the MDD and low-code development platforms such as Microsoft Power Platform are many and enormous. These are simplified logic abstraction, rapid application deployment and rapid software development. Nevertheless, these strategies have a number of drawbacks that have the potential to affect the application performance, adoption by the users and sustainability.

- **Scalability and Delegation Limits:** Applications such as Power Apps also have limits on how many records it can process or show at a given time, particularly when utilizing such connectors as SharePoint or Excel. Such limitations in delegation can affect the work with big datasets and even require restructuring of the data model or use of other sources of data.
- **Performance:** Low-code-built applications might have worse performance than custom-coded applications, particularly when dealing with complicated logic, huge data volumes, or with repeated interconnections with numerous services.
- **Debugging and Troubleshooting:** The debugging can be hard when there is limited access to the underlying code. Developers usually use the built-in logs and visual debugging tools that platform provides, which might fail to present all the issues, especially in the complicated processes or integrations.
- **Security and Compliance Issues:** Organizational sensitive data is usually accessed through model-driven applications. The incorporation of a large number of data sources greatly complicates the task of maintaining a sufficient level of access controls, data privacy, and adhering to such rules as GDPR and HIPAA.

- **Vendor Lock-In and Interoperability:** Dependence on one vendor platform may lead to inflexibility due to dependencies. Proprietary formats and connectors may make it difficult to integrate with external systems, migrate data or switch platforms.
- **Skills and Training for Developers:** Although MDD has the benefit of reducing the technical barrier to application creation, non-technical citizen developers might still need to be trained on how to comprehend data modeling, relationships, and implementation of business logic. Absence of appropriate directions may result in optimal or faulty applications.
- **Long-Term Maintainability:** Applications can also be changed over time as business needs change or platforms can change. To avoid technical debt and keep the applications robust and flexible, a clear model structure, documentation and modular design are important.

6. Literature Review

In this section, a detailed review of the research in Model-Driven Development using a particular focus of Microsoft PowerApps is to be offered, and some brief summary is to be presented in Table I.

Sahay et al. (2020) A conceptual comparison framework is suggested, which was used to provide an overview of different LCDPs. Namely, the functionalities and the services that any of the considered platforms might facilitate have been identified in terms of the set of attributes that correlate with them by analyzing a set of eight exemplary LCDPs. The final aim is to simplify the process of understanding and comparing different low-code platforms and selecting the most appropriate one depending on the needs of the users. In order to enable a citizen developer who lacks a programming background to develop software systems, major players in the IT sector are marketing low-code development platforms (LCDPs), visual and easy-to-use visual environments [24].

Liu et al. (2020) offer an example-based strategy to develop service-oriented I4.0 system capable of addressing these issues. To model SOA, they modified and implemented a comprehensive I4.0 service metamodel that was based on the Reference Model of Service Architecture (RM-SA) of I4.0. The system can be connected with the generated model which can then be used in code generation. The suggested method speeds up the reconfiguration process in the event that client needs change and helps to make service and system designs more reusable [25].

Hou, Zhang and Rana (2019) offered an explanation of a model for software architecture that included algebraic requirements, process algebra, and category theory. The software components and their relationship were characterized using category objects and morphism respectively. The semantics of the component relationships were depicted by different morphisms. With such it was the category factors which would give the mapping relationships among the different architectural models, and the typed category diagram was to give the full representation of the software architecture model. A collaborative authoring system offers an example of how to implement the formal framework to a transformation in order to identify whether it satisfies particular features or constraints [26].

Sosa-Reyna, Tello-Leal and Lara-Alabazares (2018) propose an MDD solution to deal with the challenges involved in developing IoT systems. The technique has four stages, each of which has a distinctive combination of the abstraction, perspective, and granularity. The stages are assisted by methods to transform models on the basis of MDD. In addition, a four-layer architecture of IoT systems is presented, and it is based on the Service-Oriented design (SOA) approach. This design links the virtual and the physical worlds of Internet of Things (IoT) domain by offering various ways of interoperability [27].

Tanaka et al. (2017) present a paradigm-based method in software development which entails self-adaptive power consumption. With an element model of Software Product Line development incorporated with state-machine schemas of an Executable UML, the software is able to modify its behaviour during execution in line with the recommended method. This strategy can be used to set software power consumption to the overall power consumption of the target device. The proposed program has the potential to offer the most desirable service with the limited resources available. Consequently, a balance between the quality of the services and energy efficiency can be obtained in the desired program [28].

Chansuwath and Senivongse (2016) suggest a UML profile for AngularJS that can be used to model AngularJS web applications and then a series of transformations that can be used to produce code templates from these models. Once the developer has the template, they can easily fill it out to create a fully functional web application. A

transformation tool is also built to help with the code template construction. Results from the transformation rate evaluation show that the automatically generated code covers 87% of the whole code, which might drastically reduce development time, as shown in the case study application [29].

Table 1: Summary of Model-Driven Development (MDD) with Microsoft PowerApps

Reference	Focus On	Key Findings	Challenges	Limitations
Sahay et al. (2020)	Low-Code Development Platforms (LCDPs): A Comparative Analysis	Proposed a conceptual comparative framework; identified features across 8 LCDPs to support better understanding and platform selection.	Evaluating diverse LCDPs with varying architectures and use cases.	Limited to the selected 8 platforms; may not generalize to newer/emerging LCDPs.
Liu et al. (2020)	Industry 4.0 service-oriented systems model-driven development	Developed a service metamodel (RM-SA based); improved reusability and reconfiguration; enabled faster deployment with code generation.	Handling dynamic and complex Industry 4.0 requirements.	Applied mainly to I4.0 context; generalization to other domains is limited.
Hou, Zhang & Rana (2019)	Formal software architecture modeling using category theory	Introduced typed category diagrams and functors for semantic description; improved precision in architecture transformations.	Complexity of applying abstract mathematical frameworks in practice.	Framework tested with limited case studies; scalability remains untested.
Sosa-Reyna, et.al. (2018)	MDD methodology for IoT systems	Proposed 4-phase MDD approach with SOA-based IoT architecture ensuring interoperability between heterogeneous devices.	Ensuring interoperability across diverse IoT devices.	Evaluation scope is narrow; real-world adoption challenges not fully explored.
Tanaka et al. (2017)	Developing applications with adjustable power consumption using models	Linked Executable UML with feature models; achieved runtime adaptation of power consumption with high adaptive rate (~87.6%).	Balancing power efficiency with software performance (QoS).	Focused on specific devices; applicability to broader software ecosystems not validated.
Chansuwath et.al. (2016)	UML profile for AngularJS with automated code generation	Proposed UML profile and transformations for AngularJS; achieved 87% code coverage in case study; reduced development time.	Managing framework-specific constraints and evolving JavaScript frameworks.	Framework-specific; limited to AngularJS; may not adapt well to modern JS frameworks (e.g., React, Vue).

7. Conclusion and Future Work

MDD and low-code/no-code platforms have significantly transformed software engineering by emphasizing high-level abstraction, visual modeling, and rapid application delivery over traditional code-centric methods. Automated model-to-code transformations decrease manual coding, increase productivity, and guarantee consistency while MDD enables developers to concentrate on system architecture, business requirements, and process logic. Low-code platforms, such as Microsoft PowerApps, extend these capabilities to both professional and citizen developers, enabling the creation of scalable, data-driven applications with minimal technical expertise. Despite the benefits, including improved reusability, platform independence, and enhanced user and developer experience, challenges such as tool dependency, performance limitations, delegation constraints, debugging

complexity, and long-term maintainability remain. Proper planning, training, and awareness of platform limitations are essential for leveraging the full potential of MDD and low-code solutions. Moreover, ongoing research and innovation in AI-assisted modeling, integration with DevOps, and advanced security measures are expected to further enhance the efficiency and adoption of these platforms across industries.

Future research needs to explore the interplay between artificial intelligence and MDD, particularly in understanding how auto-generating model transformations can enhance developer productivity. Additionally, future research studies should focus on conducting more empirical research on the use and enterprise adoption of PowerApps, as well as governance frameworks for citizen development adoption and frameworks that establish guarantees of scale, interoperability, and security in low-code environments.

References

[1] M. Staron, *Model Driven Engineering Languages and Systems*, vol. 4199. in Lecture Notes in Computer Science, vol. 4199. Berlin, Heidelberg, Heidelberg: Springer Berlin Heidelberg, 2006. doi: 10.1007/11880240.

[2] L. García-Borgoñón, M. A. Barcelona, J. A. García-García, M. Alba, and M. J. Escalona, "Software process modeling languages: A systematic literature review," *Inf. Softw. Technol.*, vol. 56, no. 2, pp. 103–116, Feb. 2014, doi: 10.1016/j.infsof.2013.10.001.

[3] S. Sendall and W. Kozaczynski, "Model transformation: The heart and soul of model-driven software development," *IEEE Softw.*, vol. 20, no. 5, pp. 42–45, 2003, doi: 10.1109/MS.2003.1231150.

[4] K. Falkner, C. Szabo, R. Vivian, and N. Falkner, "Evolution of Software Development Strategies," *Proc. - Int. Conf. Softw. Eng.*, vol. 2, pp. 243–252, 2015, doi: 10.1109/ICSE.2015.153.

[5] J. P. Tolvanen and S. Kelly, "Model-driven development challenges and solutions: Experiences with domain-specific modelling in industry," *Model. 2016 - Proc. 4th Int. Conf. Model. Eng. Softw. Dev.*, no. Modelsward, pp. 711–719, 2016, doi: 10.5220/0005833207110719.

[6] E. Yigitbas, I. Jovanovikj, K. Biermeier, S. Sauer, and G. Engels, "Integrated model-driven development of self-adaptive user interfaces," *Softw. Syst. Model.*, vol. 19, no. 5, pp. 1057–1081, Sep. 2020, doi: 10.1007/s10270-020-00777-7.

[7] R. Waszkowski, "Low-code platform for automating business processes in manufacturing," *IFAC-PapersOnLine*, vol. 52, no. 10, pp. 376–381, 2019, doi: 10.1016/j.ifacol.2019.10.060.

[8] A. N. Alonso *et al.*, "Building a Polyglot Data Access Layer for a Low-Code Application Development Platform," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12135 LNCS, no. 2, 2020, pp. 95–103. doi: 10.1007/978-3-030-50323-9_6.

[9] A. U. Alrubaee, D. Cetinkaya, G. Liebchen, and H. Dogan, "A Process Model for Component-Based Model-Driven Software Development," *Information*, vol. 11, no. 6, p. 302, Jun. 2020, doi: 10.3390/info11060302.

[10] A. Rodrigues da Silva, "Model-driven engineering: A survey supported by the unified conceptual model," *Comput. Lang. Syst. Struct.*, vol. 43, pp. 139–155, Oct. 2015, doi: 10.1016/j.cl.2015.06.001.

[11] M. Franzago, I. Malavolta, and H. Muccini, "Stakeholders, Viewpoints and Languages of a Modelling Framework for the Design and Development of Data-Intensive Mobile Apps," 2015.

[12] G. Taentzer and S. Vaupel, "Model-Driven Development of Mobile Applications: Towards Context-Aware Apps of High Quality," *CEUR Workshop Proc.*, vol. 1591, pp. 17–29, 2016.

[13] M. Brambilla, E. Umuhoza, and R. Acerbis, "Model-driven development of user interfaces for IoT systems via domain-specific components and patterns," *J. Internet Serv. Appl.*, vol. 8, no. 1, p. 14, Dec. 2017, doi: 10.1186/s13174-017-0064-1.

[14] M. Woo, "The Rise of No/Low Code Software Development—No Experience Needed?," *Engineering*, vol. 6, no. 9, pp. 960–961, Sep. 2020, doi: 10.1016/j.eng.2020.07.007.

[15] H. P. Kapadia, "Cross-Platform UI/UX Adaptions Engine for Hybrid Mobile Apps," *Int. J. Nov. Res. Dev.*, vol. 5, no. 9, pp. 30–37, 2020.

[16] P. Tchounikine, "Designing for Appropriation: A Theoretical Account," *Human–Computer Interact.*, vol. 32, no. 4, pp. 155–195, Jul. 2017, doi: 10.1080/07370024.2016.1203263.

[17] D. Graziotin, X. Wang, and P. Abrahamsson, "Happy software developers solve problems better: psychological measurements in empirical software engineering," *PeerJ*, vol. 2, Mar. 2014, doi: 10.7717/peerj.289.

[18] A. Fontão, O. M. Ekwoge, R. Santos, and A. C. Dias-Neto, "Facing up the primary emotions in Mobile Software Ecosystems from Developer Experience," in *Proceedings of the 2nd Workshop on Social, Human, and Economic Aspects of Software*, in WASHES '17. New York, NY, USA, NY, USA: ACM, May 2017, pp. 5–11. doi: 10.1145/3098322.3098325.

[19] S. S. S. Neeli, "Real-Time Data Management with In-Memory Databases: A Performance-Centric Approach," *J. Adv. Dev. Res.*, vol. 11, no. 2, 2020.

[20] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, "Data-Driven Evolutionary Optimization: An Overview and Case Studies," *IEEE Trans. Evol. Comput.*, 2019, doi: 10.1109/TEVC.2018.2869001.

[21] G. Fusco and L. Aversano, "An approach for semantic integration of heterogeneous data sources," *PeerJ Comput. Sci.*, vol. 6, Mar. 2020, doi: 10.7717/peerj-cs.254.

[22] S. S. S. Neeli, "Serverless Databases: A Cost-Effective and Scalable Solution," *Int. J. Innov. Res. Eng. Multidiscip. Phys. Sci.*, vol. 7, no. 6, p. 7, 2019.

[23] T. Logan, T. Williams, A. Nisbet, K. Liberman, C. Zuo, and S. Guikema, "Evaluating urban accessibility: leveraging open-source data and analytics to overcome existing limitations," *Environ. Plan. B Urban Anal. City Sci.*, vol. 46, no. 5, pp. 897–913, Jun. 2019, doi: 10.1177/2399808317736528.

[24] A. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, "Supporting the understanding and comparison of low-code development platforms," in *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, Aug. 2020, pp. 171–178. doi: 10.1109/SEAA51224.2020.00036.

[25] B. Liu, T. Glock, V. P. Betancourt, M. Kern, E. Sax, and J. Becker, "Model Driven Development Process for a Service-oriented Industry 4.0 System," in *2020 9th International Conference on Industrial Technology and Management (ICITM)*, 2020, pp. 78–83. doi: 10.1109/ICITM48982.2020.9080344.

[26] J. Hou, Y. Zhang, and A. D. Rana, "Describing Approach for Model-Driven Collaborative Application Development," in *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, 2019, pp. 336–343. doi: 10.1109/AIAM48774.2019.00073.

[27] C. M. Sosa-Reyna, E. Tello-Leal, and D. Lara-Alabazares, "An Approach Based on Model-Driven Development for IoT Applications," in *2018 IEEE International Congress on Internet of Things (ICIOT)*, 2018, pp. 134–139. doi: 10.1109/ICIOT.2018.00026.

[28] F. Tanaka, K. Hisazumi, S. Ishida, and A. Fukuda, "A methodology to develop energy adaptive software using model-driven development," in *TENCON 2017 - 2017 IEEE Region 10 Conference*, 2017, pp. 769–774. doi: 10.1109/TENCON.2017.8227963.

[29] W. Chansuwath and T. Senivongse, "A model-driven development of web applications using AngularJS framework," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, 2016, pp. 1–6. doi: 10.1109/ICIS.2016.7550838.

[30] Polam, R. M., Kamarthapu, B., Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., & Vangala, S. R. (2021). Big Text Data Analysis for Sentiment Classification in Product Reviews Using Advanced Large Language Models. *International Journal of AI, BigData, Computational and Management Studies*, *2*(2), 55-65.

[31] Gangineni, V. N., Tyagadurgam, M. S. V., Chalasani, R., Bhumireddy, J. R., & Penmetsa, M. (2021). Strengthening Cybersecurity Governance: The Impact of Firewalls on Risk Management. *International Journal of AI, BigData, Computational and Management Studies*, *2*, 10-63282.

[32] Pabbineedi, S., Penmetsa, M., Bhumireddy, J. R., Chalasani, R., Tyagadurgam, M. S. V., & Gangineni, V. N. (2021). An Advanced Machine Learning Models Design for Fraud Identification in Healthcare Insurance. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(1), 26-34.

[33] Kamarthapu, B., Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., Vangala, S. R., & Polam, R. M. (2021). Advanced Machine Learning Models for Detecting and Classifying Financial Fraud in Big Data-Driven. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(3), 39-46.

[34] Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., Penmetsa, M., Bhumireddy, J. R., & Chalasani, R. (2021). Enhancing IoT (Internet of Things) Security Through Intelligent Intrusion Detection Using ML Models. *International Journal of Emerging Research in Engineering and Technology*, *2*(1), 27-36.

[35] Vangala, S. R., Polam, R. M., Kamarthapu, B., Kakani, A. B., Nandiraju, S. K. K., & Chundru, S. K. (2021). Smart Healthcare: Machine Learning-Based Classification of Epileptic Seizure Disease Using EEG Signal Analysis. *International Journal of Emerging Research in Engineering and Technology*, *2*(3), 61-70.

[36] Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., Vangala, S. R., Polam, R. M., & Kamarthapu, B. (2021). Big Data and Predictive Analytics for Customer Retention: Exploring the Role of Machine Learning in E-Commerce. *International Journal of Emerging Trends in Computer Science and Information Technology*, *2*(2), 26-34.

[37] Penmetsa, M., Bhumireddy, J. R., Chalasani, R., Tyagadurgam, M. S. V., Gangineni, V. N., & Pabbineedi, S. (2021). Next-Generation Cybersecurity: The Role of AI and Quantum Computing in Threat Detection. *International Journal of Emerging Trends in Computer Science and Information Technology*, *2*(4), 54-61.

[38] Polu, A. R., Vattikonda, N., Gupta, A., Patchipulusu, H., Buddula, D. V. K. R., & Narra, B. (2021). Enhancing Marketing Analytics in Online Retailing through Machine Learning Classification Techniques. *Available at SSRN 5297803*.

[39] Polu, A. R., Buddula, D. V. K. R., Narra, B., Gupta, A., Vattikonda, N., & Patchipulusu, H. (2021). Evolution of AI in Software Development and Cybersecurity: Unifying Automation, Innovation, and Protection in the Digital Age. *Available at SSRN 5266517*.

[40] Polu, A. R., Vattikonda, N., Buddula, D. V. K. R., Narra, B., Patchipulusu, H., & Gupta, A. (2021). Integrating AI-Based Sentiment Analysis With Social Media Data For Enhanced Marketing Insights. *Available at SSRN 5266555*.

[41] Buddula, D. V. K. R., Patchipulusu, H. H. S., Polu, A. R., Vattikonda, N., & Gupta, A. K. (2021). INTEGRATING AI-BASED SENTIMENT ANALYSIS WITH SOCIAL MEDIA DATA FOR ENHANCED MARKETING INSIGHTS. *Journal Homepage: http://www. ijesm. co. in*, *10*(2).

[42] Gupta, A. K., Buddula, D. V. K. R., Patchipulusu, H. H. S., Polu, A. R., Narra, B., & Vattikonda, N. (2021). An Analysis of Crime Prediction and Classification Using Data Mining Techniques.

[43] Rajiv, C., Mukund Sai, V. T., Venkataswamy Naidu, G., Sriram, P., & Mitra, P. (2022). Leveraging Big Datasets for Machine Learning-Based Anomaly Detection in Cybersecurity Network Traffic. *J Contemp Edu Theo Artific Intel: JCETAI/102*.

[44] Sandeep Kumar, C., Srikanth Reddy, V., Ram Mohan, P., Bhavana, K., & Ajay Babu, K. (2022). Efficient Machine Learning Approaches for Intrusion Identification of DDoS Attacks in Cloud Networks. *J Contemp Edu Theo Artific Intel: JCETAI/101*.

[45] Bhumireddy, J. R., Chalasani, R., Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., & Penmetsa, M. (2020). Big Data-Driven Time Series Forecasting for Financial Market Prediction: Deep Learning Models. *Journal of Artificial Intelligence and Big Data*, *2*(1), 153–164.DOI: 10.31586/jaibd.2022.1341

[46] Nandiraju, S. K. K., Chundru, S. K., Vangala, S. R., Polam, R. M., Kamarthapu, B., & Kakani, A. B. (2022). Advance of AI-Based Predictive Models for Diagnosis of Alzheimer's Disease (AD) in Healthcare. *Journal of Artificial Intelligence and Big Data*, *2*(1), 141–152.DOI: 10.31586/jaibd.2022.1340

[47] Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., Penmetsa, M., Bhumireddy, J. R., & Chalasani, R. (2022). Designing an Intelligent Cybersecurity Intrusion Identify Framework Using Advanced Machine Learning Models in Cloud Computing. *Universal Library of Engineering Technology*, (Issue).

[48] Vangala, S. R., Polam, R. M., Kamarthapu, B., Kakani, A. B., Nandiraju, S. K. K., & Chundru, S. K. (2022). Leveraging Artificial Intelligence Algorithms for Risk Prediction in Life Insurance Service Industry. *Available at SSRN 5459694*.

[49] Polam, R. M., Kamarthapu, B., Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., & Vangala, S. R. (2021). Data Security in Cloud Computing: Encryption, Zero Trust, and Homomorphic Encryption. *International Journal of Emerging Trends in Computer Science and Information Technology*, *2*(3), 70-80.

[50] Gangineni, V. N., Pabbineedi, S., Penmetsa, M., Bhumireddy, J. R., Chalasani, R., & Tyagadurgam, M. S. V. Efficient Framework for Forecasting Auto Insurance Claims Utilizing Machine Learning Based Data-Driven Methodologies. *International Research Journal of Economics and Management Studies IRJEMS*, *1*(2).

[51] Vattikonda, N., Gupta, A. K., Polu, A. R., Narra, B., Buddula, D. V. K. R., & Patchipulusu, H. H. S. (2022). Blockchain Technology in Supply Chain and Logistics: A Comprehensive Review of Applications, Challenges, and Innovations. *International Journal of Emerging Research in Engineering and Technology*, *3*(3), 99-107.

[52] Narra, B., Vattikonda, N., Gupta, A. K., Buddula, D. V. K. R., Patchipulusu, H. H. S., & Polu, A. R. (2022). Revolutionizing Marketing Analytics: A Data-Driven Machine Learning Framework for Churn Prediction. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(2), 112-121.

[53] Polu, A. R., Narra, B., Buddula, D. V. K. R., Patchipulusu, H. H. S., Vattikonda, N., & Gupta, A. K. BLOCKCHAIN TECHNOLOGY AS A TOOL FOR CYBERSECURITY: STRENGTHS, WEAKNESSES, AND POTENTIAL APPLICATIONS.

[54] Bhumireddy, J. R., Chalasani, R., Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., & Penmetsa, M. (2022). Big Data-Driven Time Series Forecasting for Financial Market Prediction: Deep Learning Models. *Journal of Artificial Intelligence and Big Data*, *2*(1), 153–164.DOI: 10.31586/jaibd.2022.1341

[55] Nandiraju, S. K. K., Chundru, S. K., Vangala, S. R., Polam, R. M., Kamarthapu, B., & Kakani, A. B. (2022). Advance of AI-Based Predictive Models for Diagnosis of Alzheimer's Disease (AD) in Healthcare. *Journal of Artificial Intelligence and Big Data*, *2*(1), 141–152.DOI: 10.31586/jaibd.2022.1340

[56] Pabbineedi, S., Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., Tyagadurgam, M. S. V., & Gangineni, V. N. (2023). Scalable Deep Learning Algorithms with Big Data for Predictive Maintenance in Industrial IoT. International Journal of AI, BigData, Computational and Management Studies, 4(1), 88-97.

[57] Chalasani, R., Vangala, S. R., Polam, R. M., Kamarthapu, B., Penmetsa, M., & Bhumireddy, J. R. (2023). Detecting Network Intrusions Using Big Data-Driven Artificial Intelligence Techniques in Cybersecurity. International Journal of AI, BigData, Computational and Management Studies, 4(3), 50-60.

[58] Vangala, S. R., Polam, R. M., Kamarthapu, B., Penmetsa, M., Bhumireddy, J. R., & Chalasani, R. (2023). A Review of Machine Learning Techniques for Financial Stress Testing: Emerging Trends, Tools, and Challenges. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 4(1), 40-50.

[59] Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., Tyagadurgam, M. S. V., Gangineni, V. N., & Pabbineedi, S. (2023). A Survey on Regulatory Compliance and AI-Based Risk Management in Financial Services. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 4(4), 46-53.

[60] Bhumireddy, J. R., Chalasani, R., Vangala, S. R., Kamarthapu, B., Polam, R. M., & Penmetsa, M. (2023). Predictive Machine Learning Models for Financial Fraud Detection Leveraging Big Data Analysis. International Journal of Emerging Trends in Computer Science and Information Technology, 4(1), 34-43.

[61] Gangineni, V. N., Pabbineedi, S., Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., & Tyagadurgam, M. S. V. (2023). AI-Enabled Big Data Analytics for Climate Change Prediction and Environmental Monitoring. International Journal of Emerging Trends in Computer Science and Information Technology, 4(3), 71-79.

[62] Polam, R. M. (2023). Predictive Machine Learning Strategies and Clinical Diagnosis for Prognosis in Healthcare: Insights from MIMIC-III Dataset. Available at SSRN 5495028.

[63] Narra, B., Gupta, A., Polu, A. R., Vattikonda, N., Buddula, D. V. K. R., & Patchipulusu, H. (2023). Predictive Analytics in E-Commerce: Effective Business Analysis through Machine Learning. Available at SSRN 5315532.

[64] Narra, B., Buddula, D. V. K. R., Patchipulusu, H. H. S., Polu, A. R., Vattikonda, N., & Gupta, A. K. (2023). Advanced Edge Computing Frameworks for Optimizing Data Processing and Latency in IoT Networks. JOETSR-Journal of Emerging Trends in Scientific Research, 1(1).

[65] Patchipulusu, H. H. S., Vattikonda, N., Gupta, A. K., Polu, A. R., Narra, B., & Buddula, D. V. K. R. (2023). Opportunities and Limitations of Using Artificial Intelligence to Personalize E-Learning Platforms. International Journal of AI, BigData, Computational and Management Studies, 4(1), 128-136.

[66] Madhura, R., Krishnappa, K. H., Shashidhar, R., Shwetha, G., Yashaswini, K. P., & Sandya, G. R. (2023, December). UVM Methodology for ARINC 429 Transceiver in Loop Back Mode. In 2023 3rd International Conference on Mobile Networks and Wireless Communications (ICMNWC) (pp. 1-7). IEEE.

[67] Shashidhar, R., Kadakol, P., Sreeniketh, D., Patil, P., Krishnappa, K. H., & Madhura, R. (2023, November). EEG data analysis for stress detection using k-nearest neighbor. In 2023 International Conference on Integrated Intelligence and Communication Systems (ICIICS) (pp. 1-7). IEEE.

[68] KRISHNAPPA, K. H., & Trivedi, S. K. (2023). Efficient and Accurate Estimation of Pharmacokinetic Maps from DCE-MRI using Extended Tofts Model in Frequency Domain.

[69] Krishnappa, K. H., Shashidhar, R., Shashank, M. P., & Roopa, M. (2023, November). Detecting Parkinson's disease with prediction: A novel SVM approach. In 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE) (pp. 1-7). IEEE.

[70] Shashidhar, R., Balivada, D., Shalini, D. N., Krishnappa, K. H., & Roopa, M. (2023, November). Music Emotion Recognition using Convolutional Neural Networks for Regional Languages. In 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE) (pp. 1-7). IEEE.

[71] Madhura, R., Krishnappa, K. H., Manasa, R., & Yashaswini, K. P. (2023, August). Slack Time Analysis for APB Timer Using Genus Synthesis Tool. In International Conference on ICT for Sustainable Development (pp. 207-217). Singapore: Springer Nature Singapore.

[72] Krishnappa, K. H., & Gowda, N. V. N. (2023, August). Dictionary-Based PLS Approach to Pharmacokinetic Mapping in DCE-MRI Using Tofts Model. In International Conference on ICT for Sustainable Development (pp. 219-226). Singapore: Springer Nature Singapore.

[73] Krishnappa, K. H., & Gowda, N. V. N. (2023, August). Dictionary-Based PLS Approach to Pharmacokinetic Mapping in DCE-MRI Using Tofts Model. In International Conference on ICT for Sustainable Development (pp. 219-226). Singapore: Springer Nature Singapore.

[74] Madhura, R., Krutthika Hirebasur Krishnappa. et al., (2023). Slack time analysis for APB timer using Genus synthesis tool. 8th Edition ICT4SD International ICT Summit & Awards, Vol.3, 207–217. https://doi.org/10.1007/978-981-99-4932-8_20

[75] Shashidhar, R., Aditya, V., Srihari, S., Subhash, M. H., & Krishnappa, K. H. (2023). Empowering investors: Insights from sentiment analysis, FFT, and regression in Indian stock markets. 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE), 01–06. https://doi.org/10.1109/AIKIIE60097.2023.10390502

[76] Jayakeshav Reddy Bhumireddy, Rajiv Chalasani, Mukund Sai Vikram Tyagadurgam, Venkataswamy Naidu Gangineni, Sriram Pabbineedi, Mitra Penmetsa. Predictive models for early detection of chronic diseases in elderly populations: A machine learning perspective. Int J Comput Artif Intell 2023;4(1):71-79. DOI: 10.33545/27076571.2023.v4.i1a.169