International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

DyGAISP: Generative AI-Powered Approach for Intelligent Software Lifecycle Planning

Aryyama Kumar Jana[1], Srija Saha[2], Arnab Dey[3]

[1]Arizona State University, School of Computing, Informatics, and Decision Systems Engineering, Tempe, AZ, United States
Email: *akjana[at]asu.edu*

[2]Arizona State University, School of Computing, Informatics, and Decision Systems Engineering, Tempe, AZ, United States
Email: *ssaha35[at]asu.edu*

[3]Vice President, Leading Financial Services Provider, Jersey City, NJ, United States
Email: *adtuai[at]gmail.com*

Abstract: ***The paper presents DyGAISP, an innovative framework designed for dynamic generative AI-driven software planning that may be used in many industries. DyGAISP handles the task of reconciling constantly changing requirements, stakeholder preferences, and operational limitations that are inherent in software development. DyGAISP utilizes sophisticated mathematical formulations and state-of-the-art generative AI methods to provide a complete and fast solution for producing optimum software plans. The system starts with thorough preprocessing of incoming data, followed by advanced feature embedding to represent project needs and historical performance data. The generation of dynamic plans is accomplished using a Generative Adversarial Network (GAN), which allows for the generation of plans that can adjust to evolving project dynamics. Following that, optimization methods are used to enhance the produced plans by considering several goals, such as minimizing project completion time, maximizing resource usage, and ensuring stakeholder satisfaction. DyGAISP smoothly incorporates itself into the Software Development Life Cycle (SDLC), offering interfaces that provide effortless integration and interaction with current project management systems. The thorough assessment guarantees the efficiency and reliability of the generated software development plans and evaluating their performance based on predetermined criteria. The potential impact of this approach extends across several sectors, including banking, healthcare, and manufacturing, providing a scalable solution to the challenges faced in modern software projects.***

Keywords: Generative artificial intelligence (AI), Software Development Life Cycle (SDLC), Dynamic Optimization, Software Planning, Stakeholder management, Resource Utilization, Generative Adversarial Network (GAN), Adaptability

1. Introduction

In recent years, the field of software engineering has seen significant changes, mostly due to advancements in artificial intelligence (AI) and machine learning (ML) methodologies [1]. Generative AI has become a potent tool in software development, providing automated code creation, error discovery, and software optimization to tackle complicated tasks [2]. Considering the increasing complexity of modern software projects, and the growing need for swift delivery and innovation, there is an urgent need for mechanisms that can flexibly adjust to altering project requirements and stakeholder preferences. This research paper presents DyGAISP, a new framework for Dynamic Generative AI Software Planning. DyGAISP aims to transform software planning processes by using generative AI based techniques.

Conventional software planning approaches often depend on predetermined assumptions and human involvement, resulting in inefficiencies and suboptimal results. DyGAISP overcomes this issue by using sophisticated AI algorithms to build software plans that are optimized for individual project objectives and limitations. DyGAISP utilizes advanced techniques like Generative Adversarial Networks (GANs) and reinforcement learning to automate the creation and optimization of software plans in real-time. This enhances speed, flexibility, and effectiveness in the software development process [3].

DyGAISP is specifically intended to effortlessly incorporate into current Software Development Life Cycle (SDLC) processes, guaranteeing interoperability with widely accepted methods and tools in the industry. DyGAISP can be easily adjusted to many software development scenarios, such as web development, mobile app development, and corporate software solutions, thanks to its modular structure and configurable components. DyGAISP facilitates ongoing enhancement and optimization of software plans by integrating feedback mechanisms and iterative refinement procedures. This approach is in line with the concepts of agile development and DevOps approaches [4]. In this paper, we will dive deep into the algorithm underlying DyGAISP, its implementation in software development settings, and comments on its potential impact on the future of software engineering.

Including DyGAISP into software development processes has the potential to optimize project management, foster cooperation among development teams, and expedite the time it takes to bring software products to market. DyGAISP enhances the efficiency and creativity of software engineers by automating laborious and time-consuming processes like analyzing requirements, allocating resources, and scheduling projects. This allows engineers to concentrate on higher-level design and innovation. DyGAISP's adaptability and flexibility make it ideal for tackling the obstacles presented by quickly evolving market demands, technical breakthroughs, and regulatory landscapes. This ensures that software projects can effectively adapt and respond to dynamic contexts, maintaining their agility.

In the future, the implementation and development of DyGAISP has the potential to have a significant impact on the field of software engineering, leading to a new age of intelligent and data-driven techniques in software development. DyGAISP provides a look into how AI-driven methods might potentially transform software planning and development in several industries. DyGAISP shows potential as a fundamental framework for promoting sustainable innovation and competitiveness in the software industry. More research and development are being conducted to improve its capabilities, overcome scalability issues, and enhance its integration with emerging technologies like blockchain and edge computing.

2. Background

The advancement of software engineering techniques has been characterized by an ongoing pursuit for ways that improve efficiency, quality, and agility in software development processes [5]. Conventional software planning techniques, which are often inflexible and hierarchical, and need human intervention, have encountered difficulties in adjusting to the dynamic and intricate nature of contemporary software projects. As software systems become more complex, consisting of a wide range of capabilities, technologies, and stakeholder needs, there is a growing demand for planning approaches that are more flexible and intelligent.

The combination of artificial intelligence (AI) and software engineering has produced novel methods to automate and enhance several stages of the software development life cycle (SDLC) to address these difficulties. Generative AI has gained considerable interest due to its capacity to independently produce data, code, or other objects based on learned patterns and objectives [6]. Generative AI has the potential to transform software planning processes by using advanced techniques like neural networks, reinforcement learning, and evolutionary algorithms. This enables dynamic decision-making based on data.

Recent advancements in generative AI have made it possible to develop new frameworks and tools designed explicitly for increasing the efficiency of the software planning process. These frameworks can use machine learning algorithms to examine project needs, past data, and stakeholder preferences. They can provide optimal software plans that strike a compromise between conflicting goals such as time-to-market, resource usage, and stakeholder satisfaction. Through the automation of laborious planning processes and the ability to adjust plans in response to changing project circumstances, these AI-powered methods have the potential to fundamentally transform software development techniques and accelerate innovations in the industry.

The paper presents DyGAISP, a dynamic generative AI framework specifically developed to tackle the challenges associated with software planning in the SDLC. By seamlessly integrating into pre-existing software development workflows, DyGAISP provides a comprehensive solution that streamlines resource allocation, automates planning tasks, and accommodates the constantly changing requirements of projects. By virtue of its adaptable components and modular design, DyGAISP is capable of being modified to suit a wide range of software development environments, including but not limited to finance, healthcare, and manufacturing. DyGAISP signifies a substantial advancement in the pursuit of rational, adaptable, and robust software engineering methodologies by enabling data-driven decision-making and facilitating ongoing enhancement.

3. Methodology

The approach used in DyGAISP is an extensive framework carefully crafted to coordinate a series of interrelated processes. These processes are closely interconnected with the main objective of dynamically creating software plans that are optimized to meet unique project objectives and limitations. Within this complex structure, there are several crucial steps, each carefully designed to contribute to the overall goal of enabling efficient, adaptable, and high-quality software planning methods. The phases include a range of tasks, starting from preparing data to evaluating and validating it. These stages are the foundation of DyGAISP's innovative approach to software planning in the constantly evolving landscape of modern software development projects. The comprehensive approach of DyGAISP guarantees that it is not just a standalone tool, but a versatile and adaptable framework that can easily be incorporated into existing software development workflows.

3.1 Data Preprocessing

Data preprocessing is the first stage in the DyGAISP method. It is responsible for ensuring that the incoming data is of high quality, consistent, and compatible. This crucial phase encompasses several complex procedures, such as data cleaning, normalization, and feature engineering, with the goal of preparing the unprocessed data for later analysis and modeling.

Input:
The data preprocessing step receives input in the form of diverse datasets that include project needs, historical performance metrics, stakeholder preferences, and other useful details. The datasets may come from many sources, such as structured databases, textual records, and user surveys. They may have differences in data types, formats, and quality.

Output:
The result is a dataset that has been standardized and harmonized, making it suitable for analysis and modeling. The processed dataset is used as the input for the next phases of the DyGAISP approach, which include feature embedding, dynamic plan creation, and optimization.

Mathematical Formulations:

Data Cleaning
Data cleaning is the process of detecting and correcting inconsistencies, mistakes, and missing values in the input dataset. This procedure involves using methods such as outlier identification, imputation, and error correction to guarantee the integrity and comprehensiveness of the data.

Mathematically, the process of data cleaning may be expressed in the following manner:

$$CleanedData(D) = d_i | d_i \in D, Quality(d_i) = True$$

where D = original dataset,
d_i = individual data points,
Quality(.) = function that assesses the quality of each data point according to preset criteria

Normalization
Normalization is crucial in normalizing the scale of input features, which helps reduce the impact of varying magnitudes and improves the performance of subsequent modeling techniques. Two often used normalizing approaches are min-max scaling and z-score normalization. Mathematically, normalization may be expressed as:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

where X = original feature values,
X_{min} = minimum value of the feature,
X_{max} = maximum value of the feature

Feature Engineering
Feature engineering is the process of modifying and generating additional features from the original dataset in order to improve the prediction capability of the modeling algorithms. These strategies may include methods like reducing dimensionality, selecting features, and generating derived features. Mathematically, the process of feature engineering may be expressed as:

$$F_{new} = Transform(F)$$

where F = original feature space,
F_{new} = transformed feature space,
Transform(.) = feature engineering function

3.2 Feature Embedding

Feature embedding is an important stage in the DyGAISP process. It helps convert different types of input data into compact vector representations that can be easily analyzed and modeled. This method entails transforming project requirements, historical performance metrics, and other pertinent information into high-dimensional vector spaces. This allows for the representation of semantic linkages and contextual information that are inherent in the data.

Input:
The input for the feature embedding step consists of several datasets that include textual descriptions of project requirements, structured historical performance measurements, and categorical or numerical variables that reflect stakeholder preferences and limitations. These datasets may differ in terms of their size, complexity, and format, which requires the use of specialized methodologies to effectively include them.

Output:
The result of the feature embedding step is a collection of compact vector representations that correspond to the input data. The embeddings capture the semantic and contextual information included in the original data, making it easier to do future analysis and modeling activities, such as generating dynamic plans and optimizing them.

Mathematical Formulations:

Word Embedding (Word2Vec)
Word2Vec embedding methods are often utilized for capturing the semantic links between words and phrases in textual data, such as project requirements. Mathematically, Word2Vec entails training a neural network to forecast the context (adjacent words) of a given word using its input vector representation. The Skip-gram and Continuous Bag of Words (CBOW) architectures are often used for this objective.

$$Word2Vec(w_i) = v_i$$

where w_i = individual word or phrase,
d_i = corresponding dense vector representation

Graph Embedding (Graph Neural Networks)
Graph Neural Networks (GNNs) are used to capture the intrinsic relational structure of structured data, such as historical performance metrics. Graph Neural Networks (GNNs) function on graph topologies, enabling the transmission of information between nodes and edges to produce embeddings that include both local and global context. Mathematically, the concept of graph embedding may be expressed in the following manner:

$$GNN(G) = v_i | v_i \in R^d, \forall\, v_i \in V$$

where G = input graph,
V = set of nodes in the graph,
v_i = dense vector representation of i^{th} node

Hybrid Embedding
Occasionally, it may be advantageous to use a hybrid methodology that combines Word2Vec and GNN embeddings to effectively capture different features of the input data. This process entails combining or merging the embeddings produced from textual and structured data sources to develop cohesive representations that include both semantic and structural information. Mathematically, hybrid embedding may be represented as:

$$HybridEmbedding(X) = [Word2Vec(X_{text}), GNN(X_{structure})]$$

where X_{text} = text data,
$X_{structure}$ = structured data,
[.] = concatenation

3.3 Dynamic Plan Generation

The DyGAISP technique is centered on dynamic plan generation which allows for the automated software development plans that are customized to meet unique project objectives and limitations. This methodology utilizes sophisticated machine learning methods, namely Generative Adversarial networks (GANs), to produce software plans and

maximize predetermined goals such as project completion time, resource utilization and satisfaction.

Input:
The dynamic plan generation step receives input consisting of project requirements, historical performance data, stakeholder preferences, and other pertinent information gathered from the data preprocessing and feature embedding phases. These inputs provide the essential context and limitations for creating viable and optimum software plans.

Output:
The outcome of the dynamic plan generation step is a collection of candidate software plans, each representing a possible solution to the project requirements. These plans include the distribution of resources, organization of tasks, and delegation of duties, all designed to fulfill the project goals while meeting the specified limitations.

Mathematical Formulations:

Generative Adversarial Networks (GANs)
GANs are the primary framework used for generating dynamic plans in DyGAISP. GANs are composed of two neural networks, namely a Generator and Discriminator, which are involved in a minimax game. The Generator network receives a random noise vector from a latent space and generates potential software plans while the Discriminator network assesses the quality and viability of these plans. The Generator network aims to minimize the following loss function:

$$min_G max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}\left[log\left(1-D\left(G(z)\right)\right)\right]$$

where D(x) = discriminator's output for real data x,
G(z) = generator's output for noise vector z,
$p_{data}(x)$ = data distributions,
$p_z(z)$ = noise distributions

During an adversarial training process, the Generator acquires the ability to develop software plans that are almost identical to real plans, while the Discriminator develops the skill to differentiate between real plans and produced plans.

Dynamic Optimization
Dynamic optimization approaches are used to improve and enhance the quality of the produced software plans in addition to GANs. This may include using methodologies such as genetic algorithms, particle swarm optimization, or gradient-based optimization approaches to explore the solution space given by the generator network and find the most effective solutions. Mathematically, optimization is the process of finding the best possible value for an objective function while considering certain restrictions and needs represented by constraints.

3.4 Optimization

Optimization is a crucial step in the DyGAISP approach. Its purpose is to improve and increase the quality of software plans developed, ensuring they satisfy established goals and project restrictions. This procedure entails using sophisticated optimization techniques to explore the solution space defined by the generator network to find the most optimum solutions. It requires iteratively enhancing the performance and efficiency of the software development plans.

Input:
The optimization step receives candidate software development plans developed by the dynamic plan generation process, as well as project requirements, historical performance data, stakeholder preferences, and other relevant information gathered from prior phases of the DyGAISP approach. These inputs provide the essential context and limitations to direct the optimization process.

Output:
The optimization step produces a modified and enhanced set of software development plans that are more aligned with the set project goals and limitations. These optimized plans are practical and high-quality solutions that are customized to meet the individual needs and preferences of the project stakeholders.

Mathematical Formulations:

Objective Function

The optimization process hinges on the objective function, which measures the performance of a software development plan using predetermined metrics and criteria. The objective function is a mathematical representation that combines many project performance measures, including project completion time, resource utilization, cost reduction, and stakeholder satisfaction. It is designed to accurately reflect the overall aims and priorities of the project. The objective function may be mathematically stated as:

$$min_{Plan} f(Plan)$$

where f(Plan) = Objective function

Constraint Handling
Apart from improving the objective function, the optimization process must also adhere to the constraints set by project needs, resource limits, and stakeholder preferences. These limitations may include financial restrictions, administrative limitations, availability of resources, and regulatory obligations, among other factors. Mathematically, constraints may be included into the optimization process by using techniques such as linear programming, quadratic programming, or constraint satisfaction methods. This guarantees that the optimized software development plans comply with the defined constraints.

Optimization Algorithms
Different optimization techniques may be used to search for the best solutions within the solution space determined by the objective function and constraints. The algorithms include gradient-based optimization techniques like gradient descent and stochastic gradient descent, evolutionary algorithms like genetic algorithms and particle swarm optimization, and metaheuristic algorithms like simulated annealing and tabu

search. Every optimization technique has distinct benefits and compromises in terms of how quickly it converges, the quality of the solutions it produces, and the computing complexity it requires. This allows for the development of flexible and adaptable optimization strategies that are customized to the features of the problem domain.

Iterative Refinement

The optimization process is usually iterative, consisting of multiple cycles of generating, evaluating, and adjusting solutions to gradually enhance the quality of the software development plans. By repeatedly improving and adjusting, the optimization process gradually reaches high-quality solutions that achieve a balance between conflicting goals and limitations. This finally results in software development plans that are optimized for performance, efficiency, and stakeholder satisfaction.

3.5 SDLC Integration

Integrating with the Software Development Life Cycle (SDLC) is a vital component of the DyGAISP technique, enabling smooth integration of dynamically produced software development plans into current development processes. Essentially, this connection guarantees that the results of DyGAISP are in line with the several stages of the SDLC, starting with project initiation and continuing through deployment and maintenance. The input to this stage comprises the software plans that have been created, project requirements, comments from stakeholders, and any other pertinent information gathered throughout the planning and modeling stages. These inputs are then aligned with the respective phases of the SDLC, guaranteeing that the resulting plans are practical and consistent with established practices for development.

The result of the integration process is a unified software development roadmap that includes the plans, tasks, milestones, and deliverables developed in a dynamic manner, all within the framework of the SDLC. This roadmap functions as a comprehensive plan for development teams, clearly detailing the order of operations, allocation of resources, and interdependencies necessary to effectively carry out the project. By incorporating the outputs of DyGAISP into the Software Development Life Cycle (SDLC), businesses can take use of the advantages of AI-powered planning while still adhering to industry standards and best practices. This eventually results in more efficient, transparent, and successful software development projects.

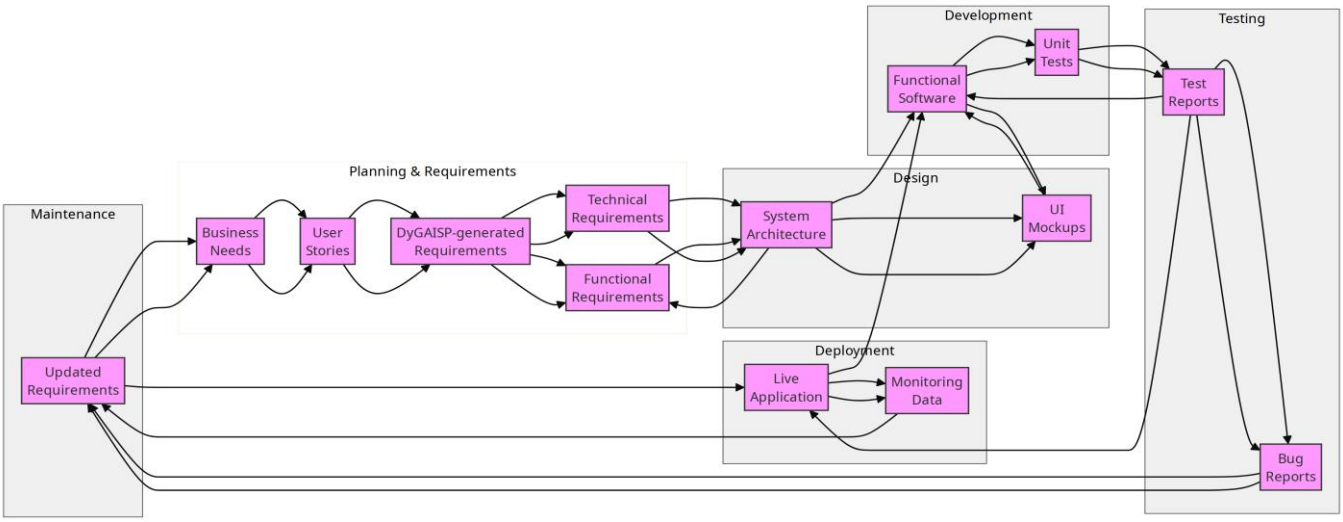

Figure 1: A Visual Overview of DyGAISP integrated into SDLC.

Figure 1 shows the phases: Planning & Requirements, Design, Development, Testing, Deployment, and Maintenance, with arrows indicating the flow from one phase to the next. Each phase is represented with blocks or symbols to show AI involvement in the Software Development Life Cycle. The diagram emphasizes the integration of DyGAISP in the SDLC, providing a professional and clear visual representation.

3.6 Validation

The DyGAISP approach includes evaluation and validation to guarantee the reliability, efficiency, and appropriateness of the software plans produced for practical execution. During the review process, the developed plans are thoroughly examined and analyzed based on predetermined performance indicators, benchmarks, and stakeholder needs. This entails evaluating variables such as the duration of project completion, the efficient use of resources, the satisfaction of stakeholders, and the adherence to project limitations. The input to this step comprises the software plans that have been developed, the assessment criteria, and previous performance data, which are used as reference points for comparison.

The outcome of the evaluation and validation step is a thorough assessment report that provides a detailed analysis of the performance, strengths, and limits of the developed software development plans. Furthermore, validation activities include the process of seeking input from end-users, domain experts, and project stakeholders to confirm the feasibility and applicability of the created plans in real-life situations. DyGAISP guarantees the quality of the produced plans by exposing them to thorough review and validation procedures. This increases trust in their usefulness and efficacy for successful software development projects.

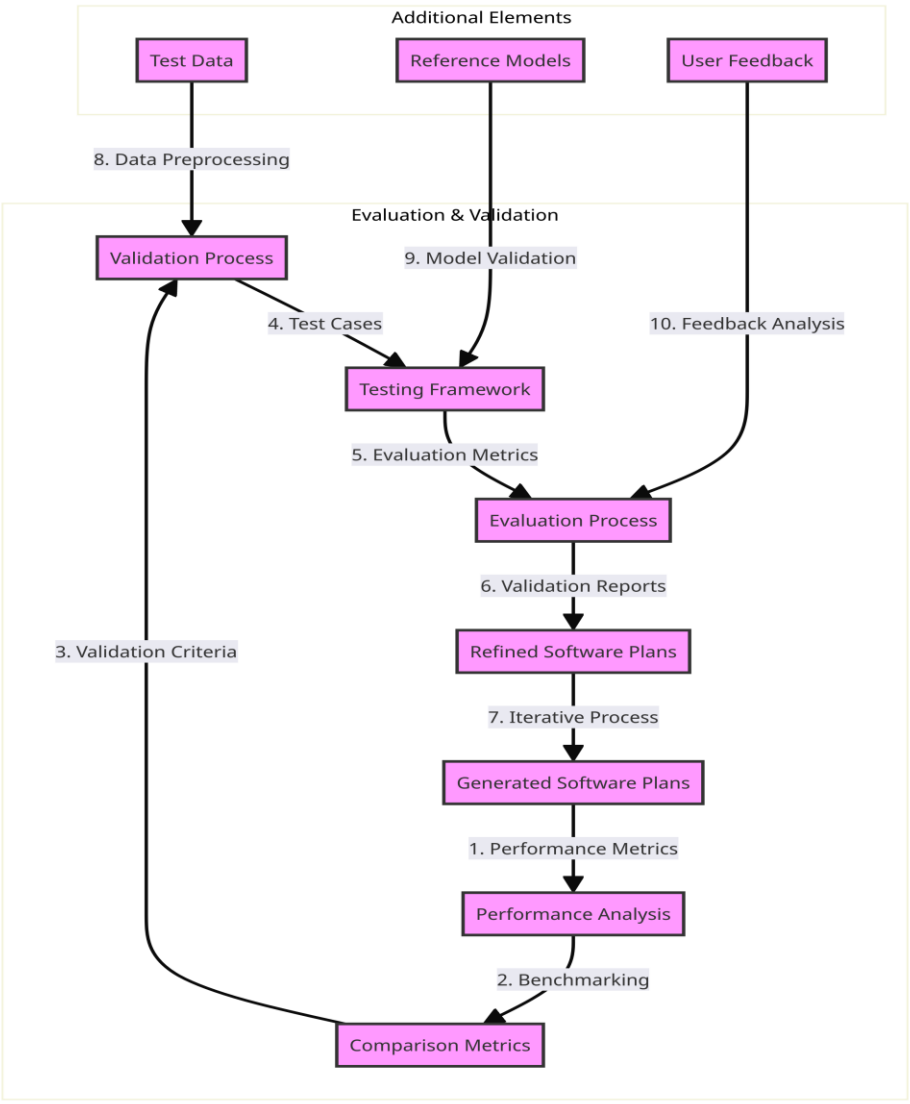

Figure 2: DyGAISP Evaluation & Validation Block Diagram.

Figure 2 represents the extensive evaluation and validation process in the DyGAISP approach, guaranteeing the reliability, effectiveness, and suitability of software designs for actual implementation. The block doagram describes a systematic process that includes analyzing performance, comparing against benchmarks, setting validation criteria, performing thorough testing, applying evaluation metrics, and making incremental improvements. Moreover, it incorporates essential components including test data, reference models, and user input to enhance the robustness and efficiency of the validation process. This comprehensive strategy aims to foster the creation of software designs of superior quality that fulfill predetermined criteria and demonstrate optimum performance in real-life situations.

4. Software Engineering Use Case

The use of DyGAISP in software development signifies a fundamental change in the way projects are strategized, implemented, and supervised. It provides a complete framework for dealing with the intricacies that are inherent in contemporary software engineering undertakings. DyGAISP, with its advanced algorithms and dynamic features, has a broad range of applications across the whole software development lifecycle (SDLC). It can transform conventional methods of project management, resource allocation, and risk reduction.

DyGAISP plays a crucial role in the early phases of the Software Development Life Cycle (SDLC), namely in gathering and analyzing project requirements. It utilizes its AI-driven capabilities to examine, prioritize, and refine the requirements. It aids development teams in establishing a solid foundation for future design and development tasks by creating software development plans that align with stakeholder expectations, market trends, and regulatory needs. In addition, DyGAISP promotes cooperation and agreement among individuals or groups involved by offering neutral observations and evaluations of potential trade-offs, resulting in a mutual understanding of project objectives and limitations.

As development advances, DyGAISP continues its influence by dynamically optimizing the allocation of resources, scheduling tasks, and distributing workloads. It utilizes real-time monitoring and analysis of project data to identify

possible bottlenecks, resource conflicts, and schedule deviations. This allows for proactive interventions to be done to reduce risks and assure the success of the project. In addition, it enables development teams to adjust to changing requirements and stakeholder input by constantly improving and revising software development plans in response to changing needs, market realities, and technology improvements.

The outputs of DyGAISP, which consist of software development plans and performance insights created in real-time, are extremely valuable resources for development teams. These outputs provide guidance for decision-making and actions at every stage of the Software Development Life Cycle (SDLC). By using DyGAISP, software development companies may attain unparalleled levels of agility, efficiency, and creativity, leading to a competitive edge and the production of better software products for the market.

5. Conclusion

The incorporation of DyGAISP into software development techniques marks a notable progress in the area, providing a robust set of tools to tackle the inherent difficulties and intricacies of modern software engineering projects. In this paper, we have examined the methodology, applications, and impact of DyGAISP, emphasizing its capacity to transform project planning, execution, and management.

We have analyzed the many ways in which DyGAISP affects software development. This includes its involvement in requirements analysis, resource allocation, risk management, and adaptability to changing project circumstances. DyGAISP enables development teams to negotiate the complexities of software projects more effectively by developing software development plans that optimize project goals while considering restrictions and uncertainties.

Furthermore, our investigation into the uses of DyGAISP across the whole software development process has shown its flexibility and capacity to adapt to various project situations and fields. DyGAISP provides a versatile framework that can be used for both small-scale agile projects and large-scale business efforts. It effectively addresses the changing requirements and challenges of current software development initiatives.

To summarize, DyGAISP is an innovative step forward in software engineering that provides a means to implement more intelligent and data-driven project management methods. DyGAISP utilizes sophisticated artificial intelligence approaches and dynamic optimization algorithms to empower development teams in creating practical software plans that are in line with project goals, stakeholder requirements, and market conditions. Further research and development efforts are necessary to improve the capabilities and scalability of DyGAISP. This will ensure that it remains relevant and impactful in the constantly changing field of software engineering. As enterprises become more aware of the benefits of using AI in software development, DyGAISP has the potential to have a significant impact on the industry by driving innovation and providing value to stakeholders in many industries and domains.

6. Future Directions

The trajectory of DyGAISP has the potential for a promising future, characterized by ongoing innovation and investigation in several areas of software engineering. An important opportunity for progress is to enhance DyGAISP's AI-driven capabilities, namely by emphasizing predictive analytics and decision assistance. Using sophisticated machine learning models, DyGAISP has the potential to transform into a proactive planning tool. It would possess the ability to anticipate project trends, risks, and opportunities, therefore enabling stakeholders to make well-informed choices in real-time.

Additionally, the incorporation of multi-objective optimization methods is a potential avenue for the further development of DyGAISP. DyGAISP can potentially use Pareto optimization and evolutionary algorithms to effectively manage the challenges of balancing competing project objectives, such as cost, quality, and time-to-market. This approach would result in Pareto-optimal solutions that provide the most favorable trade-offs among conflicting goals.

In addition, the potential of DyGAISP can be greatly enhanced by exploring its integration with future technologies such as blockchain, IoT, and edge computing. By employing the decentralized nature of blockchain and the real-time data processing capabilities of IoT and edge computing, DyGAISP could enable the creation of intelligent, autonomous software systems that can adapt to changing environments and seamlessly interact with distributed networks of devices and sensors.

References

[1] Giray, Görkem. "A software engineering perspective on engineering machine learning systems: State of the art and challenges." Journal of Systems and Software 180 (2021): 111031.
[2] Nguyen-Duc, Anh, et al. "Generative Artificial Intelligence for Software Engineering--A Research Agenda." arXiv preprint arXiv:2310.18648 (2023).
[3] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).
[4] Winters, Titus, Tom Manshreck, and Hyrum Wright. Software engineering at google: Lessons learned from programming over time. O'Reilly Media, 2020.
[5] Boehm, Barry. "A view of 20th and 21st century software engineering." Proceedings of the 28th international conference on Software engineering. 2006.
[6] Haleem, Abid, Mohd Javaid, and Ravi Pratap Singh. "An era of ChatGPT as a significant futuristic support tool: A study on features, abilities, and challenges." BenchCouncil transactions on benchmarks, standards and evaluations 2.4 (2022): 100089.

Author Profile

Aryyama Kumar Jana holds a bachelor's degree in electrical engineering from Jadavpur University and a master's degree in computer science from Arizona State University. During his academic tenure, he conducted research at several esteemed institutions, focusing on mathematical optimization, computer science, and artificial intelligence. Aryyama has also contributed significantly to industry giants such as Amazon, Siemens, Intel, and ABB, where he played critical roles in shaping the digital landscape and advancing technological innovation. His remarkable achievements, including numerous research publications and prestigious international awards, underscore his commitment to driving transformative advancements in technology and engineering.

Srija Saha graduated with a bachelor's degree in Electronics & Telecommunication Engineering from the Indian Institute of Engineering Science and Technology, Shibpur, and holds a master's degree in computer science from Arizona State University. With a strong academic background, Srija ventured into research, focusing on Artificial Intelligence and Cybersecurity. Throughout her career, she has played pivotal roles at industry leaders like OCI, Intel and PwC, contributing to transformative projects. Her accomplishments, including research publications and international awards, highlight her commitment to innovation in Computer Science.

Arnab Dey, an accomplished Vice President at JPMorgan Chase & Co., brings over 20 years of expertise in software management, architecture, and development. With a strong focus on banking technology, Arnab has led numerous innovative projects, driving efficiency and revenue growth. His dedication to leveraging cutting-edge technologies ensures that his work consistently delivers exceptional value and impact.