ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1311

Trends in Software Architecture Designs: Evolution and Current State

[1]Agwenyi C.A; [2] Mbugua S.M

Department of Information Technology
[1]Kibabii University
[2]Kibabii University

Publication Date: 2025/04/01

Abstract: Software architecture combines architectural style and quality attributes with the high-level structure of software system abstraction through composition and breakdown. In addition to meeting the system's primary functionality and performance needs, a software architectural design must also meet non-functional requirements including availability, scalability, portability, and dependability. The set of components, their relationships, how they interact, and how each component is deployed must all be described in a software architecture. There are numerous approaches to characterizing software architecture, including object-oriented modeling with UML (Unified Modeling Language), the Architecture View Model (4+1 view model), and ADL (Architecture Description Language). With an emphasis on microservices, serverless architecture, event-driven architecture, domain-driven design, cloud-native applications, zero trust security, and artificial intelligence integration, this paper reviews the latest developments in software architectural concepts, comparing their trends and contributions to modern computing. This paper reviews the evolution and current trends in software architecture designs from 2016 to 2024. It highlights key shifts, emerging paradigms, and factors influencing architectural decisions. The study is based on a systematic desktop review of existing literature, focusing on industry and academic contributions. Findings indicate a shift towards cloud-native architectures, microservices, event-driven models, and AI-enhanced frameworks. The paper synthesizes these trends and discusses their implications for future software development.

Keywords: *Software Architecture, Microservices, Cloud Computing, Event-Driven Architecture, AI-Driven Architecture, Evolution, Trends.*

How to Cite: Agwenyi C.A; Mbugua S.M (2025) Trends in Software Architecture Designs: Evolution and Current State. *International Journal of Innovative Science and Research Technology*, 10(3), 1725-1729. https://doi.org/10.38124/ijisrt/25mar1311

I. INTRODUCTION

Software architecture serves as the backbone of software systems, providing the structural framework that governs their functionality, scalability, and maintainability. Over the past decade, rapid technological advancements have influenced the evolution of software architecture, driving organizations toward more modular, scalable, and resilient designs. Software architecture directs the structure, behavior, and interactions of software components, acting as a blueprint for system design.

The evolution of software architecture continuously adapts to meet contemporary system demands and applications. This paper explores key architectural trends, their evolution, benefits, challenges, and future directions, highlighting how these paradigms improve scalability, maintainability, and performance. This paper explores recent trends in software architecture, analyzing key shifts and innovations between 2016 and 2024.

II. METHODOLOGY

This study employs a desktop research methodology, systematically reviewing academic papers, technical whitepapers, and industry reports published between 2016 and 2024. Sources include IEEE Xplore, ACM Digital Library, Springer, and leading industry reports from Gartner and Forrester. Data is analyzed to identify recurring patterns, emerging trends, and the impact of technological advancements on software architecture.

III. LITERATURE REVIEW

The literature review examines key developments in software architecture over the past decade, drawing from scholarly articles, industry reports, and case studies. The review is structured around the following themes:

- Monolithic vs. Microservices Architectures: Traditional monolithic architectures dominated software development for years but have been gradually replaced

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1311

by microservices for scalability and flexibility. (Vernon, V., & Jaskula, T. ,2021).

- Cloud-Native Architectures: The rise of cloud platforms (AWS, Azure, Google Cloud) has influenced the adoption of containerization, Kubernetes, and serverless architectures.
- Event-Driven Architectures: These architectures enable systems to react to real-time data changes, supporting scalable and loosely coupled applications. (Manchana, R. ,2021).
- AI and Automation in Architecture: The integration of AI and machine learning in architectural design has led to intelligent and self-adaptive systems. (Gheibi, O., Weyns, D., & Quin, F. ,2021).
- Security and Privacy Concerns: With increasing cyber threats, secure software architecture practices such as zero-trust models have gained prominence. Sarkar et al., 2022).

IV. MICROSERVICES ARCHITECTURE

Microservices architecture involves designing applications as a collection of loosely coupled, independently deployable services. Each service is responsible for a specific business capability and communicates with others via APIs. Microservices architecture divides applications into smaller, loosely linked services that can be independently developed, deployed, and scaled (Blinowski et al., 2022). This approach fosters flexibility, agility, and resilience. Scalability: Services scale independently, optimizing resource usage. Flexibility: Different technologies can be used for different services and Resilience: Fault tolerance is enhanced whereas pitfalls include; Complexity: Managing microservices requires robust orchestration and Data Management: Maintaining consistency is complex.

While microservices provide significant advantages, I believe organizations should be cautious about over-segmenting their applications. The complexity of managing multiple services can lead to challenges in orchestration and monitoring. Striking a balance between modularity and manageability is crucial. I propose developing a microservices governance framework that outlines best practices for service design, communication, and data management. This framework could include guidelines for when to decompose services, ensuring teams maintain a clear understanding of inter-service dependencies.

➢ *Serverless Architecture*

Serverless computing allows developers to build and deploy applications without managing servers. Cloud providers (AWS Lambda, Azure Functions, Google Cloud Functions) dynamically allocate resources as needed. Serverless architecture isolates server management, allowing developers to focus on writing code. Functions are executed in response to events (Jonas et al., 2019). The accrued benefits are as follows; Pay-as-you-go pricing reduces operational costs, platforms scale automatically to handle workloads and Faster Development: Reduces time-to-market. Whilst challenges include the following; Dependence on cloud providers and Cold Start Latency hence delays in infrequently executed functions.

Serverless computing is a powerful paradigm, but it can lead to cold start latency issues, particularly for time-sensitive applications. Additionally, vendor lock-in can hinder flexibility. Tools that facilitate multi-cloud deployments. These tools could help organizations manage serverless functions across different providers, enabling seamless integration and minimizing cold start impacts through proactive caching strategies.

➢ *Event-Driven Architecture (EDA)*

EDA enables applications to respond to events (changes in state) asynchronously. It decouples event producers from consumers, enhancing system responsiveness and scalability enhances responsiveness by enabling systems to react instantly to events, supporting real-time applications and dynamic interactions (Kommera, 2020). The greatest Benefits include Real-Time Processing this enables rapid responses. Scalability which handles high-throughput applications and flexibility that supports decoupled components, whilst challenges are enormously such as Managing event flows requires advanced tools and difficult due to decoupling.

While EDA can significantly improve application performance, it can also introduce complexities in managing event flows and debugging. Many teams may struggle to visualize and understand event interactions effectively. I advocate for creating visualization tools for event flows that enable developers to map and track events in real-time. Such tools could provide insights into event dependencies and streamline troubleshooting processes, making it easier for teams to manage complex event-driven systems.

➢ *Cloud-Native Design*

Cloud-native applications are designed to leverage cloud computing benefits, ensuring high availability, scalability, and resilience. Cloud-native design utilizes containers, microservices, and orchestration tools like Kubernetes to build scalable and resilient applications (Raj et al., 2022). The profound benefits include the following: Automated scaling, high resilience and faster software delivery. On the other hand, shortfalls include but not limited to the following; Architectural complexity and Security concerns.

➢ *Zero Trust Architecture (ZTA)*

Zero Trust enforces strict identity verification and least-privilege access, ensuring that no entity inside or outside the network is trusted by default. ZTA enhances security by assuming threats can arise from both internal and external sources, enforcing strict access controls (Stafford, 2020). Benefits that accrued include. Granular access control, Improved compliance and Resilience against insider threats. Whereas negative effects include; Implementation complexity and Integration with legacy systems.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1311

- *Domain-Driven Design (DDD)*

DDD focuses on designing software based on the complexities of real-world business domains. It emphasizes collaboration between developers and domain experts. DDD promotes software modeling based on core business concepts, ensuring alignment with business goals (Kapferer, 2020). Benefits: Business-driven development and improved modularization. Whereas challenges include; Requires significant upfront investment and Demands organizational cultural shifts.

DDD is essential for aligning software solutions with business goals, but it often requires a cultural shift within organizations. Many teams may find it challenging to fully embrace DDD principles without proper support and training. I propose establishing DDD training programs and workshops that focus on practical applications of DDD concepts. These programs could include case studies and real-world scenarios, helping teams effectively implement DDD in their projects and fostering a culture of collaboration between technical and non-technical stakeholders.

- *Integration of Artificial Intelligence (AI)*

AI-driven architectures integrate AI and machine learning models into software systems for intelligent decision-making. AI is increasingly integrated into software architecture to enable intelligent decision-making and automation (Gill et al., 2022). The tangible impact includes predictive analytics, enhanced automation and data-driven insights, whilst challenges are complexity and ethical considerations.

- *Sustainable Software Design*

Green software engineering prioritizes energy-efficient practices in software architecture. (Mahmoud, S. S., & Ahmad, I. ,2013). More companies prioritize power-efficient computing, using serverless and optimized data processing to minimize cloud costs and carbon footprints**.** Carbon-Aware Architectures for adaptive workload scheduling based on real-time energy consumption metrics is gaining traction. (Centofanti, C., Santos, J., Gudepu, V., & Kondepu, K. ,2024). Benefits include: Reduces carbon footprint and Enhances system longevity where challenges are limited standardization and trade-offs with performance.

- *Integration of Artificial Intelligence (AI)*

AI integration into software architecture allows applications to learn from data and provide intelligent functionalities. The integration of AI presents exciting opportunities, but it also raises ethical concerns regarding data privacy, bias, and transparency. As organizations increasingly adopt AI, they must prioritize responsible AI practices.I recommend developing a responsible AI toolkit that offers resources for evaluating AI models for fairness, transparency, and accountability. This toolkit could include guidelines for data sourcing, model evaluation, and best practices for communicating AI-driven decisions to users, helping organizations align their AI initiatives with ethical standards.

V. FINDINGS

The analysis of reviewed literature reveals the following key findings:

- Rise of Microservices: Companies increasingly adopt microservices for modular, scalable applications, reducing dependency on monolithic structures.
- Cloud-Native Approaches: Cloud computing drives the shift towards containerization, DevOps, and continuous integration/continuous deployment (CI/CD) pipelines.
- Event-Driven and Serverless Architectures: Organizations leverage event-driven models to enhance real-time processing capabilities and serverless computing to optimize cost and resource allocation.
- AI and Automation: AI-driven architectural patterns, such as self-healing systems and automated code generation, are gaining traction.
- Security Enhancements: Emphasis on zero-trust architectures and secure DevOps (DevSecOps) is increasing to counter growing cybersecurity threats.

- *Trends in Software Architecture Designs*

The figures here in show the most recent and recent trends in software architecture designs.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1311

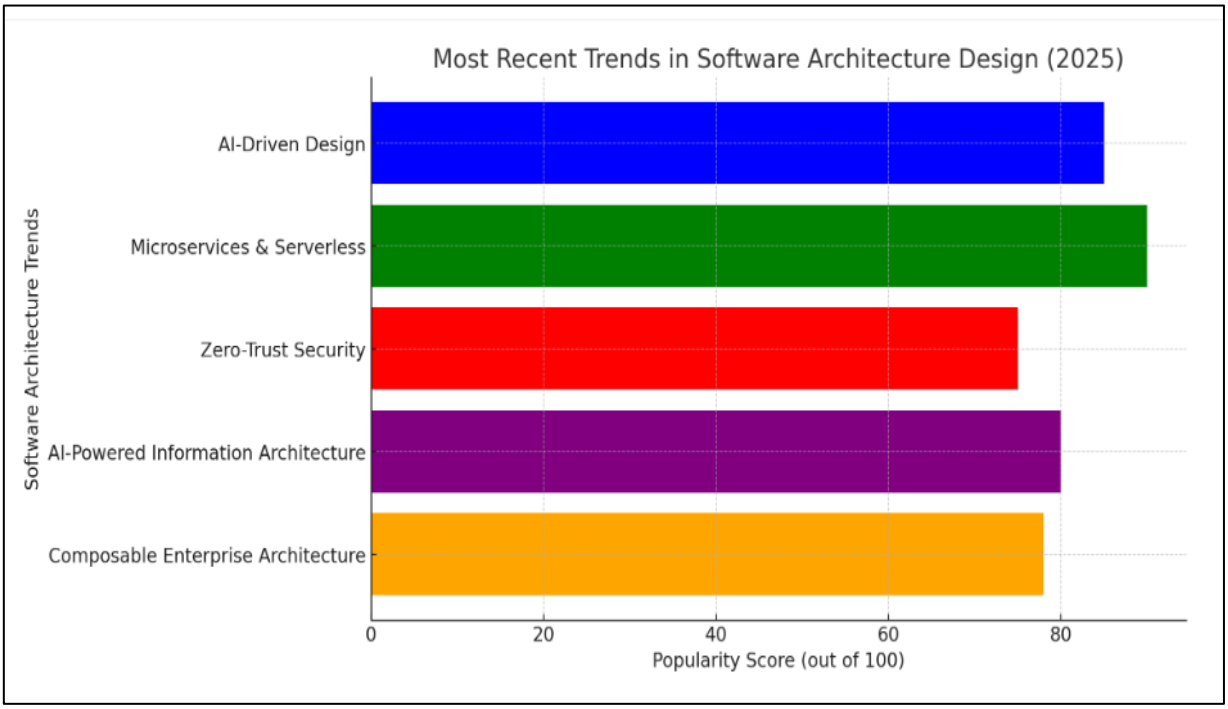

Fig 1: Most recent Trends in software architecture design
Source: Researcher *Compisition 2025*

Data from this figure shows that the most trending software architecture designs are, Microservices & Serverless, A.I-Driven and AI-powered Information Architecture with an average popularity index value of 90, 85 and 80 respectively. From this its apparent that Microservices & Serverless is much hyped design, followed by A.I-Driven, A.I-powered Information Architecture, Composable Enterprise Architecture and Zero Trust Security. Here the most recent and much hyped trend being Microservices & Serverless architectural design and Zero Trust Security is least trending.

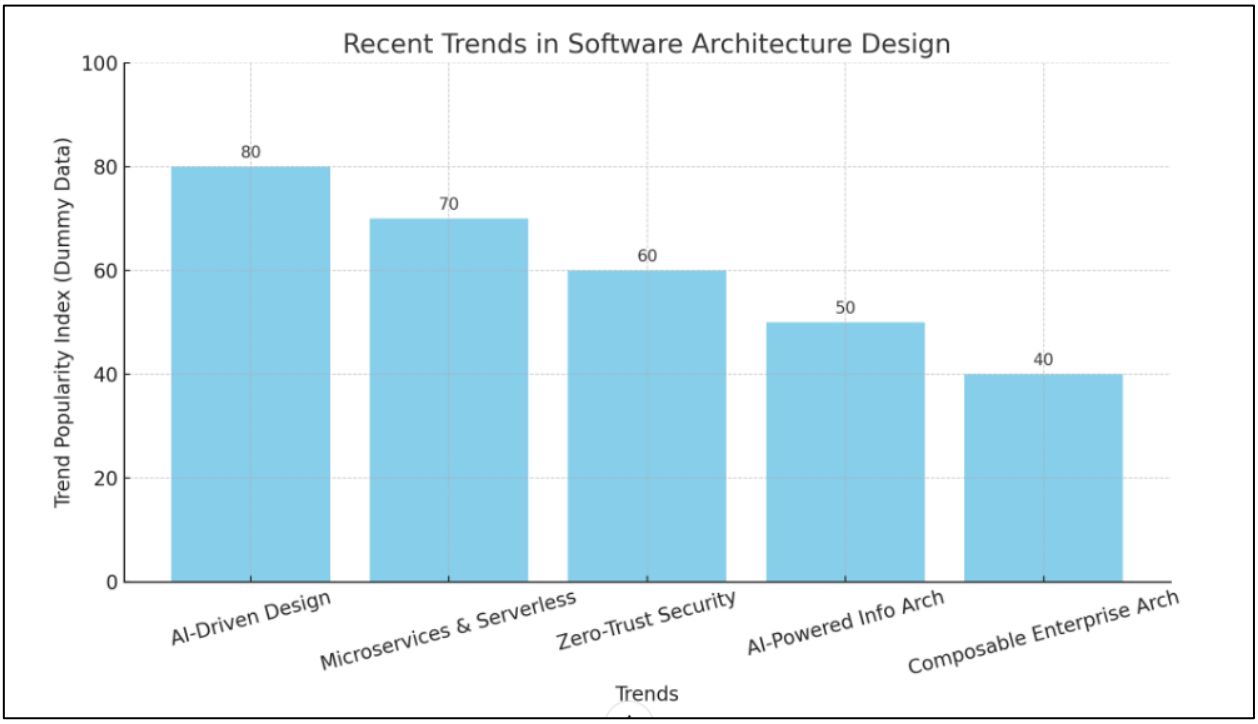

Fig 2: Recent Trends in software architecture design
Source: Researcher Composition 2025

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25mar1311

This figure shows that the most recent trending software architecture designs are AI-Driven, Microservices & Serverless and Zero-Trust Security with an average popularity index value of 80,70 and 60 respectively. From this its apparent that AI-Driven Design is much hyped design, followed by Microservices & Serverless and Zero Trust Security. The least hyped is Composable Enterprise Architecture.

VI. CONCLUSION

Software architecture has undergone significant transformation between 2016 and 2024, influenced by technological innovations and business demands. The transition from monolithic to microservices, adoption of cloud-native and event-driven models, and integration of AI in architecture design are shaping modern software development. As organizations continue to seek scalable, resilient, and secure solutions, these trends will further evolve, potentially giving rise to new architectural paradigms. Software architecture trends are shifting towards decentralization, automation, and scalability to meet modern business needs. Organizations must choose architectures based on their specific performance, security, and scalability requirements.

This paper extends existing knowledge by identifying intersections between AI-driven software architectures, sustainability in design, and security advancements. Future research should explore AI-assisted software design for optimized architectures, enhanced developer productivity and explore the long-term impact of these trends and assess emerging technologies such as quantum computing and blockchain in software architecture.

ACKNOWLEDGEMENT

It is my humble pleasure and opportunity to appreciate the guidance and support given by my course lecturer, Samuel Mbugua, throughout this paper. His vast and scholarly experience in Information Technology research and article writing came in handy to make this a success and also Kibabii University for allowing me to enroll for the course.

REFERENCES

[1]. Blinowski, G., Ojdowska, A., & Przybyłek, A. (2022). Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access*, *10*, 20357-20374.

[2]. Centofanti, C., Santos, J., Gudepu, V., & Kondepu, K. (2024). Impact of power consumption in containerized clouds: A comprehensive analysis of open-source power measurement tools. *Computer Networks*, *245*, 110371.

[3]. Chi, H. L., Kang, S. C., & Wang, X. (2013). Research trends and opportunities of augmented reality applications in architecture, engineering, and construction. *Automation in construction*, *33*, 116-122.

[4]. Cleland-Huang, J., Gotel, O. C., Huffman Hayes, J., Mäder, P., & Zisman, A. (2014). Software traceability: trends and future directions. In *Future of software engineering proceedings* (pp. 55-69).

[5]. Dobrica, L., & Niemela, E. (2002). A survey on software architecture analysis methods. *IEEE Transactions on software Engineering*, *28*(7), 638-653.

[6]. Garlan, D. (2000, May). Software architecture: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (pp. 91-101).

[7]. Gheibi, O., Weyns, D., & Quin, F. (2021). Applying machine learning in self-adaptive systems: A systematic literature review. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, *15*(3), 1-37.

[8]. Gill, S. S., Xu, M., Ottaviani, C., Patros, P., Bahsoon, R., Shaghaghi, A., ... & Uhlig, S. (2022). AI for next generation computing: Emerging trends and future directions. *Internet of Things*, *19*, 100514.

[9]. Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C. C., Khandelwal, A., Pu, Q., ... & Patterson, D. A. (2019). Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383*.

[10]. Kapferer, S. (2020). *A Modeling Framework for Strategic Domain-driven Design and Service Decomposition* (Doctoral dissertation, HSR Hochschule für Technik Rapperswil).

[11]. Kommera, A. R. (2020). The Power of Event-Driven Architecture: Enabling Real-Time Systems and Scalable Solutions. *Turkish Journal of Computer and Mathematics Education (TURCOMAT) ISSN*, *3048*, 4855.

[12]. Mahmoud, S. S., & Ahmad, I. (2013). A green model for sustainable software engineering. *International Journal of Software Engineering and Its Applications*, *7*(4), 55-74.

[13]. Manchana, R. (2021). Event-Driven Architecture: Building Responsive and Scalable Systems for Modern Industries. *International Journal of Science and Research (IJSR)*, *10*(1), 1706-1716.

[14]. Nasr, L., & Khalil, S. (2024). Development of Scalable Microservices: Best Practices for Designing, Deploying, and Optimizing Distributed Systems to Achieve High Performance, Fault Tolerance, and Seamless Scalability. *Eigenpub Review of Science and Technology*, *8*(7), 86-113.

[15]. Raj, P., Vanga, S., & Chaudhary, A. (2022). *Cloud-Native Computing: How to Design, Develop, and Secure Microservices and Event-Driven Applications*. John Wiley & Sons.

[16]. Sarkar, S., Choudhary, G., Shandilya, S. K., Hussain, A., & Kim, H. (2022). Security of zero trust networks in cloud computing: A comparative review. *Sustainability*, *14*(18), 11213.

[17]. Stafford, V. (2020). Zero trust architecture. *NIST special publication*, *800*, 207

[18]. Vernon, V., & Jaskula, T. (2021). *Strategic monoliths and microservices: driving innovation using purposeful architecture*. Addison-Wesley Professional.