A Comprehensive Review of Interactive System Architecture

Mohammed Swaleh Mohammed

Technical University of Mombasa

DOI: https://doi.org/10.51244/IJRSI.2024.1102010

Received: 13 January 2024; Revised: 01 February 2024; Accepted: 05 February 2024; Published: 02 March 2024

ABSTRACT

This paper presents a detailed review of interactive software architecture (ISA). ISA has emerged as an important aspect in software engineering offering a framework that prioritizes user interaction, real-time responsiveness and adaptability.

The paper explores various aspects crucial for the design, development, and maintenance of robust and scalable software systems. The review encompasses architectural patterns, design principles, scalability, maintainability, security, and emerging trends. The paper also examines the significance of ISA and challenges for the same.

Objective: The objective of the paper is to provide a comprehensive understanding of the key considerations in interactive system architecture and how they impact the overall success of a software project. The paper also goes on to discuss the challenges and benefits of implementation of ISA

Methodology: the study adopted a desktop approach by reviewing existing literature on the topic at hand so as to provide a simplified understanding of the interactive system architecture.

Recommendations: at the end of the paper it provides recommendation on future developments in this field.

Keywords: interactive system architecture (ISA), design, principles, challenges of ISA

INTRODUCTION

Brief Background of ISA

Interactive Software Architecture (ISA) represents a significant advancement in software engineering, highlighting the increasing need for software systems that prioritize user interaction, adaptability, and user-centric design. Interactive systems architecture refers to the design and organization of computer systems characterized by significant amounts of interaction between humans and the computer (*Interactive Systems | Encyclopedia.com*, n.d.).

The origins of ISA may be traced back to the initial stages of software architecture research. This research sought to identify the fundamental rules that govern the design and structure of intricate software systems (Bass et al., 2020). ISA has become a significant architectural paradigm known for its emphasis on enabling smooth user interactions, immediate reactivity, and ability to adapt to changing requirements.

The primary objective of interaction-oriented architecture, a subset of interactive systems architecture, is to separate the interaction of the user from data abstraction and business data processing (*Interaction-Oriented*

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XI Issue II February 2024

Architecture, n.d.-b). This architecture decomposes the system into three major partitions: the data module, control module, and view presentation module (*Interaction-Oriented Architecture*, n.d.-b).

ISA is a design paradigm that aims to separate user interaction from data abstraction and business data processing (*Interaction-Oriented Architecture*, n.d.-c). This architecture decomposes the system into three major partitions: the data module, control module, and view presentation module (*Interaction-Oriented Architecture*, n.d.-c).

The data module provides data abstraction and all business logic, while the control module identifies the flow of control and system configuration actions. The view presentation module is responsible for the visual or audio presentation of data output and provides an interface for user input (*Interaction-Oriented Architecture*, n.d.-c).

Two major styles of interaction-oriented architecture are Model-View-Controller (MVC) and Presentation-Abstraction-Control (PAC). Both MVC and PAC propose a three-component decomposition and are used for interactive applications such as web applications with multiple tasks and user interactions (*Interaction-Oriented Architecture*, n.d.-c). However, they differ in their flow of control and organization. PAC is an agent-based hierarchical architecture, while MVC does not have a clear hierarchical structure (*Interaction-Oriented Architecture*, n.d.-c).

Component-based software architecture, a subset of interaction-oriented architecture, divides a problem into sub-problems each associated with component partitions (*Software Design and Architecture Lecture 15*, n.d.-b). The interfaces of the components play important roles in the component-based design, with the main motivation behind this design being component reusability (*Software Design and Architecture Lecture 15*, n.d.-b). This approach increases overall system reliability since the reliability of each individual component enhances the reliability of the whole system via reuse (*Software Design and Architecture Lecture 15*, n.d.-b).

In conclusion, interaction-oriented software architecture offers a structured approach to separating user interaction from data processing, promoting component reusability, and enhancing system reliability. However, it's important to note that the choice between MVC and PAC, or other styles, should be guided by the specific needs and context of the software project.

Evolution of ISA

ISA has evolved significantly over the years, driven by the increasing complexity of software systems and the need for more effective ways to manage user interactions. The core of interactive systems is based on the functional requirements for the system, which usually remains stable. However, user interfaces are often subject to change and adaptation, requiring architectures that support the adaptation of user interface parts without causing significant disruption (Seffah et al. 2008).

Significance of studying interactive systems architecture

The development of ISA has been closely linked with technological advancements and the evolving field of software engineering. The widespread adoption of the internet, mobile devices, and ubiquitous computing has required a transition towards software systems that are highly dynamic and responsive (Butani, 2020). As a result, ISA has been utilized in various fields such as web applications, gaming, multimedia systems, and IoT (Internet of Things) ecosystems. The versatility of this technology highlights its significance in a constantly evolving technological environment.

Therefore, studying interactive systems architecture is important because it provides a structured approach to designing systems that are user-centric and adaptable. It allows for the creation of systems that can

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XI Issue II February 2024

effectively manage user interactions, process data, and adapt to changing user needs and system requirements (*Interaction-Oriented Architecture*, n.d.-b). Furthermore, understanding this architecture can lead to the development of more efficient, reliable, and scalable systems.

The importance of ISA has grown in recent years due to the widespread use of microservices architectures and the rising focus on user-centered design methods (Chalupsky & Osterweil, 2017). The architectural patterns and concepts that form the foundation of ISA strongly correspond to the principles of modularity and encapsulation that are central to microservices, making it a seamless match for contemporary software development paradigms.

Ultimately, ISA has emerged as a robust architectural paradigm that effectively caters to the requirements of modern software engineering. It specifically emphasizes the creation of interactive, user-centered, and flexible software systems. In order to advance the field and fulfill the changing needs of users and stakeholders, it is crucial to comprehend and establish the principles and practices of Information Systems Architecture (ISA), as software becomes more and more crucial in our lives.

Overview of the research paper's objectives and structure

The objective of the paper is to provide a comprehensive review of interactive system architecture.

To do this we will examine the below

- Brief background on ISA
- Principles of ISA
- Architectural Components and patterns of ISA
- Benefits of ISA
- Challenges in implementation of ISA
- Recommendation of future studies
- Conclusion

PRINCIPLES OF INTERACTIVE SOFTWARE ARCHITECTURE

Interactive Software Architecture (ISA) is distinguished from other architectural paradigms by its design and execution, which are based on a set of fundamental principles. These principles together steer the creation of software systems that give priority to human engagement, immediate reactivity, and adaptability. Avgeriou et al. (2019) identified a set of principles that characterize ISA, utilizing well-established research and industry standards.

Modular decomposition

The key concepts and principles in ISA include the separation of concerns, modular decomposition, loose coupling, and support for multiple views (Seffah et al. 2008; Bass et al, 2020). The primary objective of interaction-oriented architecture, a subset of ISA, is to separate the interaction of the user from data abstraction and business data processing (Amini et al. 2019). This idea improves the capacity to reuse and maintain components, as they may be developed and updated separately, resulting in systems that are more adaptable and can handle larger workloads.

This architecture decomposes the system into three major partitions: the data module, control module, and view presentation module (Amini et al. 2019). Each module has its own responsibilities, promoting modularity and loose coupling (Seffah et al. 2008).

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XI Issue II February 2024

User-Centric Design:

ISA prioritizes the user experience as the primary focus of architectural concerns. The authors argue for a comprehensive comprehension of user requirements, inclinations, and actions during the entirety of the software development procedure (Avgeriou et al., 2019). ISA incorporates user-centered design approaches, such as user personas and usability testing, to guarantee that software systems are intuitive and promptly responsive to user input.

Ability to respond quickly

ISA is characterized by its ability to quickly respond in real-time, which is made possible by the use of event-driven communication protocols. Elements within an Instruction Set Architecture (ISA)-based system interact through events, enabling prompt reactions to user inputs or modifications in the system's surroundings (Soares et al. 2021). This guarantees that the software maintains a high level of interactivity and adaptability.

Smooth and consistent interactions

The ISA framework prioritizes the orchestration of user interfaces, allowing for smooth and consistent interactions. It entails the synchronization of diverse interface elements and components to deliver a coherent and captivating user experience (Soares et al. 2021). This orchestration guarantees that user interactions are seamless and instinctive, irrespective of the underlying intricacy.

ISA systems are intentionally built to be adaptable and capable of evolving. According to Chalupsky and Osterweil (2017), software systems have the ability to adapt and adjust to new needs, technologies, and user expectations without the need for significant redesign. This idea allows software systems to stay current and competitive in rapidly changing contexts.

These principles collectively establish the basis of ISA, providing guidance to architects and developers in designing software systems that excel in user engagement, real-time responsiveness, and adaptability. ISA promotes the development of software that not only fulfills functional requirements but also delivers an exceptional user experience, in line with the changing needs of modern software engineering (Gonçalves et al. 2023).

ARCHITECTURAL COMPONENTS AND PATTERNS

According to Avgeriou et al. (2019), Interactive Software Architecture (ISA) is based on a collection of architectural components and patterns that enable the development of software systems that are optimized for user interaction, real-time responsiveness, and adaptability. The components and patterns serve as the foundational elements of ISA, establishing the structural framework for the creation of interactive software. This part delves into the fundamental architectural components and patterns related to ISA, utilizing well-established research and industry standards.

Components of the User Interface (UI): The fundamental components of ISA consist of user interface (UI) elements, including buttons, forms, and widgets (Avgeriou et al., 2019). These components have the responsibility of displaying information to users and recording their input. UI components are meticulously crafted to guarantee usability, responsiveness, and an aesthetically pleasant user experience.

Event-Driven Mechanism – ISA significantly relies on event handling algorithms to effectively handle user interactions and system responses (Bass et al., 2020). Occurrences, such as user clicks or keyboard

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XI Issue II February 2024

inputs, initiate particular operations within the software. Event-driven programming paradigms, such as the observer pattern, are frequently used to facilitate immediate responsiveness.

Middleware and Communication Layers: Middleware and communication layers are integrated into ISA to facilitate the smooth transmission of data and events among components (Soares et al., 2021). These layers ease the interchange of data and coordination between components by permitting inter-component communication, while also ensuring that the components remain loosely coupled.

Component Containers: ISA encourages the use of modularity and componentization, which are made easier by employing component containers or containership patterns (Chalupsky & Osterweil, 2017). Containers serve the purpose of enclosing and overseeing collections of interconnected elements, hence improving the capacity to reuse and maintain them. Typical container types encompass modules, services, and microservices.

Model-View-Controller (MVC) and Variations: The concept of Model-View-Controller (MVC) and its several adaptations: The MVC architectural pattern, including its versions like Model-View-Presenter (MVP) and Model-View-View Model (MVVM), is commonly used in ISA (*What were the major problems with MVC Framework*? – Geeks for Geeks. 2022) The patterns facilitate the segregation of the user interface (view) from the application logic (controller or presenter), hence fostering a well-defined and sustainable design.

State Management: State management is crucial in interactive systems due to their dynamic character, as stated by Bass et al. (2020) in their research on ISA. Architectural patterns, such as the Finite State Machine (FSM), aid in the representation and control of the several states that a system or component can undergo as a result of user interactions.

Adaptive Components: ISA prioritizes adaptability and achieves it through the use of adaptive components and patterns (Chalupsky & Osterweil, 2017). These components have the ability to adapt their behaviour and appearance in response to user preferences, environmental conditions, or system requirements.

The inclusion of these architectural components and patterns is crucial for the effectiveness of ISA-based systems, as they empower architects and developers to design software that is exceptionally interactive, responsive to user activities, and capable of adapting to dynamic circumstances. ISA strikes a balance between modularity, usability, and performance by utilizing these components and patterns. This aligns with the key principles of user-centric design and flexibility.

BENEFITS OF INTERACTIVE SYSTEMS ARCHITECTURE (ISA)

The use of ISA provides numerous benefits, as emphasized by Chalupsky and Osterweil (2017):

1. Improved User Satisfaction: ISA prioritizes user-centric design, resulting in enhanced user satisfaction and usability, crucial factors in the current competitive software market.

2. The ease of maintenance of ISA is enhanced by its modularity, which allows for localized adjustments to certain components without impacting the overall system.

3. Scalability: The modular design of ISA facilitates the expansion of systems by adding or substituting components as required.

4. Components produced following ISA principles can be repurposed in many projects, resulting in time and resource savings.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XI Issue II February 2024

Challenges and Issues in implementation of ISA

ISA faces several challenges and issues that can impact the effectiveness and efficiency of the system:

Scalability and performance considerations

As interactive systems grow in complexity and user base, scalability becomes a critical challenge. The architecture must be able to handle increased loads without performance degradation. This includes ensuring that the backend infrastructure can scale to meet demand and that the user interface remains responsive (Rzayev, T. 2019; Lard et al. 2004).

Security and privacy concerns

With the increasing amount of sensitive data processed by interactive systems, security and privacy issues are important to consider during the implementation of the ISA. The architecture must incorporate robust security measures to protect against unauthorized access and data breaches. Privacy concerns also dictate that the system must ensure user data is handled in compliance with regulations and ethical standards (Rzayev, 2019; Ward & DeVault, 2016).

Compatibility and interoperability challenges

Interactive systems often need to work with a range of other systems and technologies. Ensuring compatibility and interoperability is a significant challenge, particularly when integrating new technologies or updating legacy systems. The architecture must support standard communication protocols and be designed with interoperability in mind (Lard et al; 2004; *Information*, n.d.).

Proposed Solutions to address the Challenges in implementation of ISA

Existing research and literature on interactive systems architecture address these challenges, offering insights into potential solutions and best practices. For example, the paper "Advances in Software Architecture Design Applied to Human Computer Interaction Processing" discusses distributed architectures that propose high-level services oriented toward interaction for all applications (Lard et al. 2004). The article "Challenges in Building Highly Interactive Dialogue Systems" identifies key challenges relating to modeling, systems architecture, and development methods, emphasizing the importance of addressing these issues for the success of interactive systems (Ward & DeVault, 2016). Additionally, the special issue on Interactive Systems from the journal "Information" highlights the problems of human-computer interactions and the complexities that arise in professional and daily life due to the evolving nature of computerized environments [(*Information*, n.d.). These resources contribute to a deeper understanding of the challenges faced in designing and implementing interactive systems architecture.

Future Directions and Research Opportunities

Emerging trends and technologies, potential areas of improvement, and research gaps present numerous opportunities for the future of interactive systems architecture:

Emerging trends and technologies in ISA

As interactive systems evolve, they open the door for more seamless human-computer interaction and collaboration (Rzayev, T. 2019). For instance, the rise of mixed reality, artificial intelligence, and machine learning technologies can significantly enhance the capabilities of interactive systems (Rzayev, T. 2019).

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XI Issue II February 2024

Additionally, the concept of interactive architecture, which involves designing spaces that can change and adapt in real-time response to people and the environment, presents exciting possibilities for the future (Wikipedia contributors, 2023a).

Potential areas of improvement and innovation

There are several areas where interactive systems architecture can be improved. For instance, enhancing the scalability and performance of interactive systems, particularly as they grow in complexity, is a critical area of focus [Amini, et al. 2019). Additionally, addressing security and privacy concerns, particularly in the context of increasing amounts of sensitive data being processed by interactive systems, is another important area for improvement (Ward & DeVault, 2016b).

Research gaps and avenues for further exploration

Despite the significant progress in interactive systems architecture, there are still many research gaps and avenues for further exploration. For example, the complexity of computerized environments has increased year by year, leading to problems in professional activities as well as in daily life (*Information*, n.d.-b). Addressing these complexities and designing more user-friendly and intuitive systems is a key research challenge. Furthermore, integrating interaction design into the software development process and exploring the potential of domain-specific languages for interactive systems are other promising research directions (*Information*, n.d.-b).

The future of ISA is promising, with numerous opportunities for research and innovation. By addressing the current challenges and leveraging emerging trends and technologies, we can expect to see significant advancements in this field.

CONCLUSION

Summary of key findings and insights from the research paper

This research paper has explored the key principles, components, challenges, and future directions of interactive systems architecture. The study has highlighted the importance of separation of concerns, modularity, loose coupling, and support for multiple views in the design and implementation of interactive systems (Achten, 2019; Taleb, 2008). It has also underscored the critical role of user interface design, backend infrastructure, and communication protocols in facilitating effective user interaction and system functionality (Amini et al. 2019; Seffah et al. 2008).

The research has identified scalability, security, and compatibility as major challenges in interactive systems architecture. It has also pointed out the potential of emerging trends and technologies, such as mixed reality, artificial intelligence, and machine learning, in enhancing the capabilities of interactive systems (Taleb, 2008; Seffah et al. 2008).

Implications of the study on the field of interactive systems architecture

The implications of this study are significant for both researchers and practitioners in the field of interactive systems architecture. It provides a comprehensive overview of the current state of the field and highlights areas for future research and innovation. The study also underscores the need for continuous improvement in the design and implementation of interactive systems to address the evolving needs of users and the increasing complexity of software systems (Achten, 2019; Taleb, 2008; Amini et al. 2019; Seffah et al. 2008).

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XI Issue II February 2024

Recommendations for future research and industry practitioners.

For future research, the study recommends a focus on enhancing the scalability and performance of interactive systems, addressing security and privacy concerns, and ensuring compatibility and interoperability. It also suggests exploring the potential of emerging trends and technologies and addressing the complexities of computerized environments (Taleb, 2008; Seffah et al. 2008).

For industry practitioners, the study recommends adopting the key principles of interactive systems architecture in the design and implementation of systems. It also suggests focusing on user interface design, investing in robust backend infrastructure, and adopting standard communication protocols to facilitate effective user interaction and system functionality (Amini et al. 2019; Seffah et al. 2008).

REFERENCES

1. Avgeriou, P., Kruchten, P., & Staron, M. (2019). Architectural Patterns Revisited—A Pattern Language. Springer.
2. Aymen Daoudi, Ghizlane ElBoussaidi, Naouel Moha, and Sègla Kpodjedo. 2019. An exploratory study of MVC-based architectural patterns in Android apps. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19). Association for Computing Machinery, New York, NY, USA, 1711–1720. https://doi.org/10.1145/3297280.3297447
3. Bass, L., Clements, P., & Kazman, R. (2020). Software architecture in practice. Addison-Wesley.
4. Bhadwal, P. (2022, March 4). MVC Architecture – Definition, Advantages, Disadvantages – TechStrot. TechStrot. https://www.techstrot.com/mvc-architecture/
5. Butani, A. (2020, December 16). 5 essential patterns of software architecture. Enable Architect. https://www.redhat.com/architect/5-essential-patterns-software-architecture
6. Chalupsky, H. P., & Osterweil, L. J. (2017). Software architecture: A roadmap. In Software Architecture (pp. 3-12). Springer.
7. Dissanayake, N. R., & Dias, K. (2017). Balanced abstract web-MVC style: an abstract MVC implementation for web-based applications. GSTF Journal on Computing (JoC), 5(3).
8. Garud, S. (2023, July 20). Mastering MVC architecture: A comprehensive guide for web developers. LinkedIn. https://www.linkedin.com/pulse/mastering-mvc-architecture-comprehensive-guide-web-developers-garud
9. How To Build MVC Architecture in Node Js Applications? (2019, September 10). CronJ. https://www.cronj.com/blog/mvc-architecture-in-node-js/
10. Interaction-Oriented architecture. (n.d.). https://www.tutorialspoint.com/software architecture design/interaction oriented architecture.htm
11. Martin, M. (2023, September 27). MVC Framework tutorial for beginners: What is, Architecture & Example. Guru99. https://www.guru99.com/mvc-tutorial.html
12. Model View Presenter Styles · Los Techies. (n.d.). Model View Presenter Styles &Middot; Los Techies. https://lostechies.com/derekgreer/2008/11/23/model-view-presenter-styles/
13. MVC Framework Introduction – Geeks for Geeks. (2022, March 21). Geeks for Geeks. https://www.geeksforgeeks.org/mvc-framework-introduction/
14. Sera, B. (2018, April 21). MVC is obsolete. Medium. https://medium.com/@balint_sera/mvc-is-obsolete-99203eba5247
15. Sheldon, R. (2023, September 12). What is model-view-controller (MVC)?: Definition from TechTarget. WhatIs.com. https://www.techtarget.com/whatis/definition/model-view-controller-MVC
16. Soares, M. M., Rosenzweig, E., & Marcus, A. (Eds.). (2021). Design, User Experience, and Usability:

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XI Issue II February 2024

UX Research and Design: 10th International Conference, DUXU 2021, Held as Part of the 23rd HCI International Conference, HCII 2021, Virtual Event, July 24–29, 2021, Proceedings, Part I (Vol. 12779). Springer Nature.

17. Software Design and Architecture Lecture 15. Review Interaction-Oriented Software Architectures – MVC. – ppt download. (n.d.). Software Design and Architecture Lecture 15. Review Interaction-Oriented Software Architectures – MVC. – Ppt Download. https://slideplayer.com/slide/9454303/
18. Soni, N. (2023, October 26). Benefits and drawbacks of MVC Architecture. S2 Labs by Shrey Sharma. https://shreysharma.com/benefits-and-drawbacks-of-mvc-architecture/
19. The History of MVC." Microsoft Developer, https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/history-and-evolution-of-mvc
20. Geeks for Geeks. (2022, April 14). What were the major problems with MVC Framework. https://www.geeksforgeeks.org/what-were-the-major-problems-with-mvc-framework
21. Verdecchia, R., Kruchten, P., Lago, P., & Malavolta, I. (2021). Building and evaluating a theory of architectural technical debt in software-intensive systems. Journal of Systems and Software, 176, 110925.
22. What are the advantages and disadvantages of using the MVC pattern for web development? (n.d.) Linkedin. Retrieved on 30th January 2024 from https://www.linkedin.com/advice/3/what-advantages-disadvantages-using-2e