International Journal of Business Quantitative Economics and Applied Management Research

Volume-5, Issue-9, February-2019 ISSN No: 2349-5677

ARCHITECTING AND IMPLEMENTING A MIDDLEWARE FRAMEWORK FOR SEAMLESS INTEROPERABILITY OF CUSTOM SYSTEMS IN OMNICHANNEL RETAIL ENVIRONMENTS

Devender Yadav

Abstract

The contemporary retail environment comprises a complex array of channels, touchpoints, and consumer expectations. Omnichannel strategies, which seek to deliver a cohesive and consistent experience across all channels, have become essential rather than optional for businesses. Achieving true omnichannel harmony presents significant technical challenges, especially in the integration of various systems, including both custom-built and standard off-the-shelf solutions. This paper examines the design and implementation of a custom middleware layer, aimed at integrating disparate systems. This solution enables the efficient transfer of data, allowing retailers to provide a cohesive customer experience. This study examines the architectural decisions, technological choices, and practical implications of implementing a middleware solution in a retail context. The focus extends beyond technical feasibility to include the development of a flexible, scalable, and adaptable solution capable of evolving with the dynamic demands of the retail sector.

Keywords: Omnichannel Retail, Middleware, System Integration, Data Synchronization, API, ESB, Microservices, Retail Technology, Customer Experience, Legacy Systems

I. INTRODUCTION

The retail industry is currently experiencing significant transformation. Consider the evolution of your shopping behaviors over the past ten years. Customers engage with brands through various channels, including physical stores, websites, mobile applications, social media, and emerging platforms such as voice assistants. The increase in touchpoints has led to the emergence of "omnichannel" retail, a strategy aimed at providing a seamless and integrated experience across all channels.

The concept of omnichannel presents significant advantages: it offers a comprehensive perspective of the customer, ensures uniform pricing and promotions, facilitates tailored interactions, and ultimately enhances customer loyalty and sales. However, the situation for numerous retailers is considerably more intricate. Their operations are typically underpinned by a diverse array of systems, including legacy systems that are decades old, modern and advanced technologies, custom-built solutions tailored to specific requirements, and standard software packages.

International Journal of Business Quantitative Economics and Applied Management Research
Volume-5, Issue-9, February-2019 ISSN No: 2349-5677

The integration of these disparate systems presents a considerable challenge. Consider the difficulties associated with synchronizing inventory between online and offline channels, maintaining consistent product information, and developing a comprehensive customer profile that encompasses interactions from all touchpoints. In the absence of seamless integration, retailers encounter data silos, operational inefficiencies, and a disjointed customer experience that fails to meet the standards of an omnichannel approach.

This paper examines the core of this challenge. This study examines the role of a well-designed and effectively implemented middleware layer as a critical component for achieving omnichannel success. The middleware functions as a strategic asset rather than merely a technological component. This middleware functions as a translator and facilitator, enabling communication and data exchange between systems that were not originally designed for interoperability.

II. PROBLEM STATEMENT

The primary issue we examine is the fundamental incompatibility and absence of interoperability among the various systems that support contemporary omnichannel retail operations. This incompatibility is evident in multiple significant aspects:

1. Data Silos: Data silos result in information being confined to isolated areas, causing an incomplete and fragmented understanding of customers, inventory, orders, and other essential business data.
2. Inconsistent Customer Experience: Inconsistent Customer Experience: Disparate systems frequently lead to inconsistencies in pricing, promotions, product information, and service levels across various channels, resulting in customer confusion and frustration. This leads to customers feeling undervalued, and they prefer a more comfortable shopping experience at their preferred retailers.
3. Operational Inefficiencies: Operational inefficiencies arise from manual data entry, reconciliation efforts, and the lack of process automation across channels, resulting in elevated costs and diminished operational agility [1].
4. Limited Real-time Visibility: Limited real-time visibility due to inadequate data synchronization impedes a retailer's capacity to swiftly adapt to evolving market conditions, customer requirements, or supply chain disruptions. One may place an online order, only to subsequently discover that the item is unavailable in the warehouse.
5. Difficulty in Adopting New Technologies: The integration of new technologies or channels presents challenges when existing systems lack interoperability, resulting in a complex and time-consuming process. This hinders numerous retailers from innovating and adapting to future developments [2].

The challenges are exacerbated by the coexistence of custom-built systems, typically developed internally to meet specific business requirements, alongside standard off-the-shelf solutions, including Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) systems. Custom systems, although designed for specific needs, may pose challenges in integrating with external systems. Conventional systems often exhibit insufficient flexibility to adapt to distinct business processes. This analogy illustrates the difficulty of reconciling incompatible elements.

III. SOLUTION

The proposed solution involves designing and implementing a robust, adaptable middleware layer that serves as a central component for system integration in an omnichannel retail environment. This middleware is not designed as a monolithic solution implemented as a temporary measure. It is a meticulously organized collection of components, each designated a specific function, collaborating to ensure efficient data flow and process automation. It resembles a well-coordinated orchestra, in which each section plays a role in achieving overall harmony, rather than allowing one instrument to dominate.

Key Aspects and Functionalities

1. **API-Driven Architecture:** The middleware will provide a comprehensive array of Application Programming Interfaces (APIs). These APIs will function as the standardized medium for communication and data exchange between various systems [3]. It is essential to evaluate the most suitable API style for this integration.
   a) **RESTful APIs:** RESTful APIs are typically favored for their simplicity and scalability in most interactions. Systems will be enabled to request and exchange data through standard HTTP methods, including GET, POST, PUT, and DELETE. A website verifying item availability would likely utilize a RESTful API call.
   b) **GraphQL APIs:** GraphQL APIs will be implemented for scenarios that necessitate complex and flexible data retrieval. This enables clients to specify the exact data required, thereby minimizing over-fetching and enhancing efficiency. A mobile application may utilize GraphQL to obtain only the customer's name, recent orders, and loyalty points, instead of accessing the complete customer profile.
   c) **SOAP APIs:** SOAP APIs, while less prevalent in contemporary architectures, may be essential for integration with specific legacy systems that depend on this older standard. Integration with a system that is over 15 years old will likely require the use of SOAP. The middleware will be designed to manage these interactions, ensuring compatibility among all systems.

2. **Data Transformation and Mapping:** The middleware performs a crucial role in managing the complex processes of data transformation and mapping. Data exists in various formats and structures across different systems. It is unrealistic to anticipate that they will immediately communicate in a common language.
   a) **Format Conversion:** The middleware will convert data between multiple formats, including JSON, XML, CSV, and others. For instance, it may convert data from a JSON format utilized by an e-commerce platform to an XML format necessary for an older ERP system [4].

Volume-5, Issue-9, February-2019 ISSN No: 2349-5677

b) **Schema Mapping**: It will effectively map data fields between systems with varying schemas, regardless of differences in field names or data types. This guarantees accurate data transfer and interpretation, even among systems with fundamentally different data models. Schema mapping functions similarly to language translation, facilitating the conversion of data between different formats.
c) **Data Validation:** Data validation is conducted by the middleware prior to data transmission to verify its integrity and adherence to the receiving system's requirements.
d) **Event-Driven Processing:** The middleware will utilize an event-driven architecture to attain real-time responsiveness and agility. This indicates that actions within one system will elicit corresponding actions in other systems, resulting in a dynamic and interconnected environment.
e) **Event Producers and Consumers:** Systems will be classified as event producers, which generate events, and event consumers, which respond to events. The middleware will facilitate the event flow between the components.
f) **Real-time Updates:** The placement of an online order initiates a series of events: the inventory system is updated instantaneously, the warehouse is alerted to prepare the shipment, the customer's profile is revised to reflect the purchase history, and a confirmation email is dispatched, all coordinated by the middleware.

3. **Message Queuing:** Message queuing mechanisms will be integrated into the middleware to facilitate reliable and robust communication. Asynchronous communication, enabled by message queues, is essential [5].
   a) **Decoupling:** Decoupling occurs when message queues enable systems to function independently and asynchronously. This mitigates bottlenecks and enhances system resilience, ensuring that a failure in one system does not immediately propagate to others.
   b) **Guaranteed Delivery:** Message queues provide guaranteed delivery, ensuring that messages are retained and not lost during periods of system unavailability. They serve as a buffer, retaining messages until the receiving system is prepared to process them.
   c) **Popular Options:** Technologies such as RabbitMQ, Kafka, and ActiveMQ will be evaluated for the implementation of message queuing functionality.

4. **Microservices Approach**: The middleware can be designed using a microservices approach to improve flexibility and scalability.
   a) **Independent Services:** The functionality of the middleware will be divided into smaller, independent services that can be developed, deployed, and scaled autonomously. This facilitates enhanced agility and simplifies maintenance [6].
   b) **Technology Diversity:** Various microservices may be developed using distinct technologies, selected according to the specific needs of each service.

5. **Enterprise Service Bus (ESB) Capabilities:** The middleware functions as a central component of a comprehensive Enterprise Service Bus (ESB) strategy. An ESB offers a

International Journal of Business Quantitative Economics and Applied Management Research
Volume-5, Issue-9, February-2019 ISSN No: 2349-5677

centralized framework for the management and orchestration of interactions among services. The middleware is capable of managing essential ESB functions, including:

a) **Routing:** Routing involves the intelligent direction of messages to relevant services according to established rules or content criteria.
b) **Mediation**: Mediation involves the adaptation of communication protocols and message formats between incompatible services.
c) **Process Orchestration:** Process orchestration involves the definition and management of intricate business processes that encompass various services and sequential steps [7].
d) **Service Management:** Service Management involves the assessment of service performance and availability, alongside the provision of tools for effective lifecycle management.

IV. ARCHITECTURE

The middleware layer's architecture is modular, scalable, and adaptable to the changing requirements of omnichannel retail businesses. It utilizes a combination of architectural patterns and technologies to facilitate seamless integration among diverse systems.

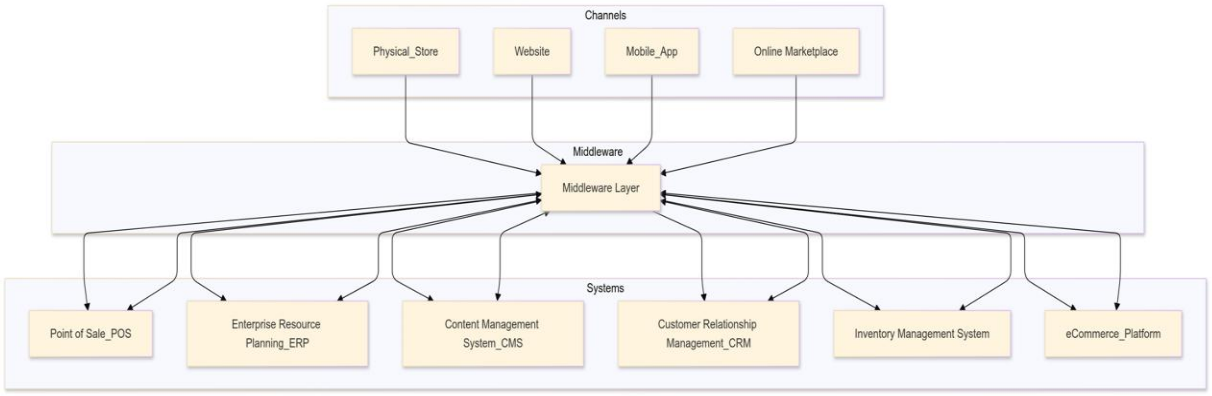

Figure 1: High Level Solution Design of a Middleware

1. **API Gateway:**
   a) Functions as the initial interface for all external requests directed to the middleware.
   b) Facilitates interaction between various systems through the provision of RESTful and GraphQL APIs.
   c) Manages authentication, authorization, and rate limiting.
   d) Directs requests to the relevant services within the middleware.

2. **Middleware Layer**:
   a) Data Transformation and Mapping: Data transformation and mapping involve converting data across various formats, such as JSON and XML, and aligning data fields between systems that utilize distinct schemas.
   b) Event-Driven Processing: This component oversees the transmission of events among systems. The system employs a message queue, such as RabbitMQ or Kafka, to enable asynchronous communication and guarantee reliable message delivery.

c) Microservices (Optional): The middleware may be decomposed into smaller, independent microservices to enhance flexibility, scalability, and maintainability. Each microservice is designated to perform a specific function, including order processing, inventory management, or customer profile synchronization [8].
d) ESB Capabilities: This component offers routing, mediation, process orchestration, and service management functionalities, serving as a central hub for service interactions.

3. **Integrated Systems:**
   a) POS System: The in-store point-of-sale system transmits sales data and inventory updates to the middleware.
   b) E-commerce Platform: The online store engages with the middleware to synchronize orders, customer data, product information, and inventory levels.
   c) ERP System: The enterprise resource planning system supplies the middleware with inventory data, financial information, and additional back-office data [9].
   d) CRM System: The customer relationship management system integrates customer data, interaction history, and loyalty program information with the middleware.
   e) WMS: The warehouse management system offers real-time updates regarding inventory levels, order fulfillment status, and shipping information.
   f) Marketing Automation Platform: The Marketing Automation Platform interfaces with middleware to tailor marketing campaigns according to customer data and behavior.
   g) Mobile App: The native mobile application utilizes the middleware's APIs to facilitate a seamless shopping experience for mobile users, encompassing access to product information, order tracking, and personalized recommendations.

V. EXAMPLE OF INTEGRATION WORKFLOW: ONLINE ORDER PROCESSING

1. A customer submits an order on the e-commerce platform.
2. The e-commerce platform transmits a "Order Created" event to the middleware through the API Gateway.
3. The event-driven processing component of the middleware receives the event and subsequently places it in the message queue.
4. The data transformation and mapping component converts order data from the e-commerce platform's format to a standardized format utilized within the middleware.
5. The middleware directs the modified order data to the relevant microservices or ESB components.
6. The order processing microservice updates the ERP system with new order information.
7. The inventory management microservice synchronizes inventory levels across the ERP and WMS systems.
8. The microservice for customer profile synchronization updates the CRM system with the purchase history of customers.
9. The WMS is notified to prepare the order for shipment.
10. The middleware transmits order status updates to both the e-commerce platform and the mobile application, thereby ensuring customer awareness.

International Journal of Business Quantitative Economics and Applied Management Research
Volume-5, Issue-9, February-2019 ISSN No: 2349-5677

This architecture facilitates an uninterrupted data flow among all interconnected systems, thereby ensuring data consistency, real-time visibility, and a cohesive customer experience across all channels. It offers the capability to incorporate new systems and adjust to evolving business needs.

VI. USES

The implemented middleware layer serves various functions within the omnichannel retail environment, including:

1. **Unified Customer Profile:** Integrating customer data from various touchpoints into a singular, comprehensive view to facilitate personalized marketing and service.
2. **Real-time Inventory Synchronization:** Maintaining precise and current inventory data across all channels to minimize stockouts and prevent overselling [10].
3. **Order Management:** Order Management involves optimizing the order fulfillment process through the integration of online and offline channels, facilitating efficient order routing, tracking, and delivery.
4. **Consistent Pricing and Promotions:** Consistent pricing and promotions are essential for maintaining clarity across all channels, thereby reducing customer confusion and fostering trust.
5. **Personalized Recommendations:** Utilizing customer data and browsing history to deliver tailored product recommendations and offers across various channels.
6. **Seamless Returns and Exchanges:** Ensuring a uniform and efficient return and exchange process across all purchasing channels.
7. **Click-and-Collect/Buy Online, Pickup In-Store (BOPIS):** It involves the integration of online ordering with in-store inventory, facilitating convenient pickup options for customers.
8. **Loyalty Program Integration:** Establishing connections among loyalty programs across various channels to create a cohesive perspective on customer rewards and benefits.

VII. IMPACT

The effective implementation of this middleware layer significantly influences a retailer's operations and enhances its capacity to provide a compelling omnichannel experience:

1. **Enhanced Customer Experience:** An integrated experience promotes customer loyalty and satisfaction. Customers experience a sense of value and comprehension, resulting in increased repeat business and favorable word-of-mouth referrals.
2. **Improved Operational Efficiency**: Enhanced operational efficiency is achieved through automation and real-time data flow, which streamline processes, minimize manual effort, and elevate overall effectiveness.
3. **Increased Sales and Revenue:** Consistent pricing, personalized offers, and a seamless shopping experience can lead to substantial increases in sales and revenue.
4. **Better Decision-Making:** A unified view of data enhances decision-making for retailers, enabling more informed choices regarding inventory, marketing, and other essential business domains.

5. **Greater Agility and Innovation**: The middleware offers a flexible foundation for integrating new technologies and channels, allowing retailers to swiftly adapt to changing customer expectations and market trends.
6. **Cost Savings:** Automation diminishes the necessity for manual data entry and reconciliation, leading to cost savings.
7. **Competitive Advantage:** Retailers that effectively implement omnichannel integration achieve a notable competitive advantage in the current market landscape.

VIII. SCOPE

This middleware solution integrates various systems commonly present in a retail environment, including:

1. **Point of Sale (POS) Systems:** Integration of traditional in-store POS systems with mobile POS solutions.
2. **E-commerce Platforms:** E-commerce platforms facilitate the connection of online stores developed on diverse systems, such as Shopify, Magento, and WooCommerce.
3. **ERP Systems:** ERP systems integrate with essential business systems that oversee finances, inventory, and various back-office functions, such as SAP and Oracle.
4. **CRM Systems:** CRM systems facilitate the integration of customer relationship management platforms to consolidate customer data and interactions, such as Salesforce and Microsoft Dynamics.
5. **Warehouse Management Systems (WMS):** WMS facilitate the integration of systems that oversee warehouse operations and inventory management.
6. **Order Management Systems (OMS):** OMS facilitate integration with systems that oversee the complete order lifecycle.
7. **Marketing Automation Platforms**: Marketing Automation Platforms: Integration with systems that facilitate the automation of marketing campaigns and customer segmentation.
8. **Social Media Platforms**: Social media platforms facilitate the capture of customer interactions and feedback through engagement with various channels.
9. **Mobile Applications:** Integrating with native mobile applications to ensure a seamless experience for mobile shoppers.
10. **Third-party Marketplaces**: Integration with third-party marketplaces such as Amazon and eBay facilitates the management of product listings and orders.

IX. CONCLUSION

The pursuit of genuine omnichannel retail is intricate; however, it is a worthwhile endeavor. The design and implementation of a robust middleware layer outlined in this paper establish a crucial foundation for success. This solution serves as a strategic enabler, allowing retailers to address system integration challenges, maximize data potential, and provide a cohesive and engaging customer experience across all channels.

This approach enables retailers to overcome the constraints of fragmented systems, facilitating the development of stronger customer relationships, operational optimization, and success in the dynamic environment of contemporary commerce. The middleware layer serves as a crucial connection between systems, facilitating advancements in the retail sector. Connecting systems extends beyond technical integration; it involves establishing a deeper engagement with customers. This approach enables retailers to fulfill current customer expectations while also anticipating future needs.

REFERENCES

1. A. D. A. M. S. Project, A Reference Model for Computer Assisted Personalized Approach (CAPPA): A Framework for Designing and Implementing Personalized Services, 2018.
2. E. Brynjolfsson and A. McAfee, The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies. New York, NY: W. W. Norton & Company, 2014.
3. A. Zimmermann, K. Schmidt, S. Kahl, L. Wolf, and K. Hinkelmann, "A design method for service-oriented architectures," in Proceedings of the 16th European Conference on Pattern Languages of Programs, 2011, pp. 1–12.
4. C. Pautasso, O. Zimmermann, and F. Leymann, "RESTful web services vs. big' web services: Making the right architectural decision," in Proceedings of the 17th International Conference on World Wide Web, 2008, pp. 805–814.
5. D. Linthicum, Cloud Computing and SOA Convergence in Your Enterprise: A Step-by-Step Guide. Addison-Wesley, 2009.
6. E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, 2004.
7. B. W. Boehm, "A spiral model of software development and enhancement," Computer, vol. 21, no. 5, pp. 61–72, 1988.
8. F. P. Brooks Jr., The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley, 1995
9. D. L. Parnas, "On the criteria to be used in decomposing systems into modules," Communications of the ACM, vol. 15, no. 12, pp. 1053–1058, 1972.
10. D. Rigby, J. Sutherland, and H. Takeuchi, "Embracing agile," Harvard Business Review, vol. 94, no. 5, pp. 40–50, 2016.