ISSN 2278-3091
Volume 14, No.6, November – December 2025
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse011462025.pdf
https://doi.org/10.30534/ijatcse/2025/011462025

Architectural Approaches to Designing Scalable Information Systems in E-Commerce

Andrey Berezhnoy
Bachelor's degree, Peter the Great St. Petersburg polytechnic university, St. Petersburg, Russia

Received Date: October 20, 2025 Accepted Date: November 22, 2025 Published Date: December 06, 2025

ABSTRACT

The article examines modern architectural approaches to designing scalable information systems in e-commerce. Key principles are discussed, including modularity, loose coupling of components, and support for early detection of failures and automatic recovery of the services. Microservice architecture, containerization, clustering, and event-driven models are focused on, which support flexibility and high availability. Real-life scenarios involving the large e-commerce giants such as Amazon, Walmart, and Wayfair are described, testifying to how efficiently they are able to handle enormous data and heavy loads. Architectural fault tolerance and flexibility are set forth as the key to maintaining competitiveness and fitting in with sustainable business development in the ever-changing digital business landscape.

Key words : e-commerce, information systems, architectural approaches, system design, scalability, fault tolerance.

1. INTRODUCTION

Modern electronic commerce is under the stringent availability, performance, and flexibility demands of information systems. Online sales growth, customized customer support needs, periodic peak loads (e.g., for sales periods), and ongoing refreshing of product lines necessitate the use of solutions having the ability for dynamic scalability, fault tolerance, and modularity principles support. For this purpose, microservice architecture-based solutions, clustering, containerization, and cloud technologies are the most applicable since they enable systems to adapt rapidly to respond to shifting business requirements.
The aim of this article is to analyze contemporary architectural approaches to the design of scalable information systems in the field of electronic commerce, with a focus on their principles, implementation models, and practical application.

2. GENERAL REQUIREMENTS FOR E-COMMERCE INFORMATION SYSTEMS

E-commerce continues to be one of the fastest-growing areas of the digital economy, showing consistent growth and a widening global footprint. The scale of this segment is confirmed by analytical data. According to research conducted by Verified Market Research, the global e-commerce software market reached $10,15 billion in 2024 (figure 1).

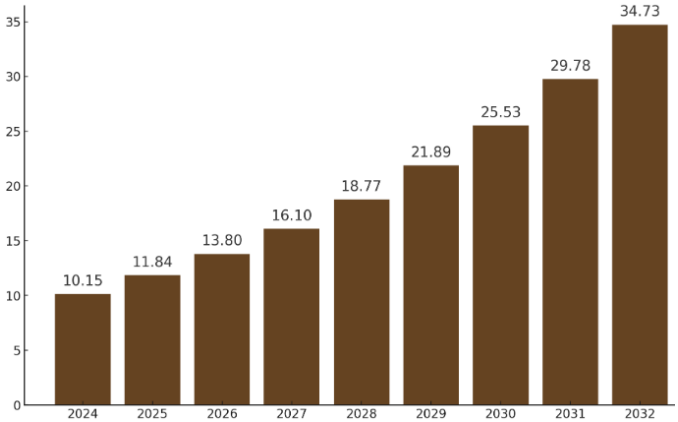

Figure 1: Global e-commerce software market size, billion dollars [1]

Such expansion is closely associated with the rising sophistication of digital platforms, which need to be able to handle high volumes of transactions, offer round-the-clock access to services, and maintain customer confidence. In this regard, information systems applied in e-commerce not only provide core functionality for product, order, and user management, but also fulfill a number of essential requirements. These requirements arise from the specifics of online trade, where delays, system failures, or data loss may lead to significant financial losses and reduced customer loyalty (table 1).

Table 1: Generalized requirements for e-commerce information systems [2, 3]

Requirement category	Description	Examples of implementation
Scalability	Ability of the system to efficiently process increasing volumes of data and requests.	Horizontal scaling via Kubernetes.
Fault tolerance	Ability to continue functioning despite failures of individual components.	Replication, redundancy, circuit breaker patterns.
High availability	Guaranteed level of service uptime, often not lower than 99,95%.	Load balancing architecture with redundant zones.

Performance	Fast processing of user requests even under peak loads.	CDN usage, caching with Redis.
Requirement category	**Description**	**Examples of implementation**
Modularity and flexibility	Capacity to alter and enhance features without complete system overhaul.	Architecture of microservices, approach prioritizing APIs.
Integration with external systems	Support for interaction with external services via APIs and protocols.	REST, GraphQL, Webhooks.
Security	Protection of data and transactions against external and internal threats.	TLS, OAuth 2.0, CSRF/XSS protection.
Monitoring and management	Ability to track system status and manage incidents.	Prometheus, Grafana, centralized logging.
Usability and UX	Ease of use, responsiveness, and efficiency of user interfaces.	Optimized frontend, SPA, responsive design.

Thus, designing information system architecture in e-commerce requires a holistic approach in view of technical, functional, and operational aspects. All the requirements mentioned above directly affect the platform's competitiveness and the ability of the platform to operate effectively under the highly dynamic market environment.

3. ARCHITECTURAL PRINCIPLES OF SCALABLE SYSTEM DESIGN

Creation of information systems scalable for e-commerce requires rigid adherence to architectural principles ensuring flexibility, fault tolerance, and potential for evolutionary development. Of these principles, most essential are modularity, loose coupling of the subsystems, and the «fail fast» policy, which ensure rapid detection of faults and containment along with automated recovery measures.

Modularity, as a design principle in architecture, refers to compartmentalizing a system into segregated logical components (modules), each performing a precisely defined function. This facilitates simpler maintenance, simple testing, and modification or expansion of individual components without affecting the system as a whole. As a study by Broadleaf Commerce (2024) observes, organizations applying modular or composable approaches reduce time-to-market for new functionality by 30-50 % compared to traditional monolithic implementations.

The modularity approach allows architectural flexibility, allowing systems to adapt as business requirements and technology landscapes change. It enables technical debt removal, improved resilience against failures, and accelerates the deployment of new functional components. Modularity is particularly critical in e-commerce since it allows isolated development and scaling of significant subsystems. Figure 2 illustrates a typical modular architecture structure, demonstrating the principle of organizational separation into independent functional blocks.

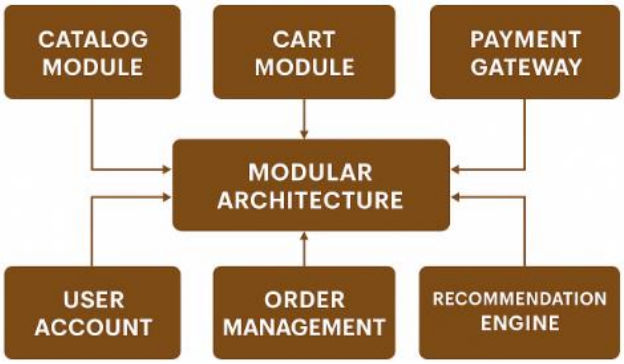

Figure 2: Modular architecture of an e-commerce information system

Modular patterns form the foundation for more intricate patterns such as microservices, serverless, and event-driven architecture. Modular architectures extend the benefits of modularity to enable services to be run independently, to scale decentralization, and to react swiftly to changing workloads. In high-demand e-commerce situations, these patterns are necessary in order to achieve resilience, efficiency, and responsiveness in development.

Another design principle for scalable systems is loose coupling, or low interdependence of system modules. This type of property proves useful in enabling flexibility as well as scalability: reconfiguring or reinitializing a single module does not have to affect others. Loose coupling can be facilitated by shared interfaces, message brokers such as Kafka or RabbitMQ, and event-driven and publish–subscribe patterns. In e-commerce, this approach is particularly relevant for event handling –for instance, the placement of an order can trigger a sequence of actions (inventory deduction, customer notification, delivery calculation), each implemented independently and executed asynchronously (figure 3).

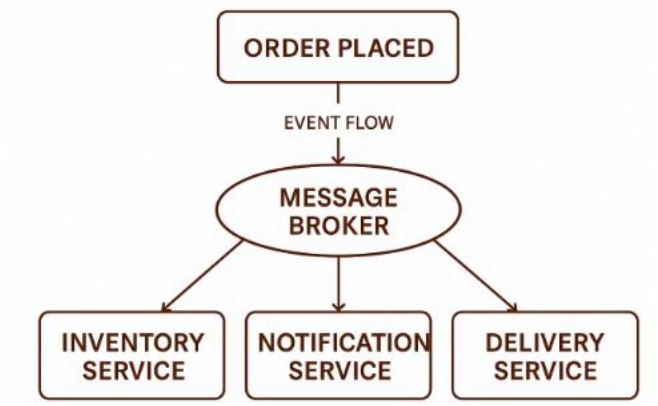

Figure 3: Loose coupling and event-driven communication between microservices in an e-commerce system

The diagram illustrates how event-driven communication enables the decoupling of services in a scalable e-commerce architecture. Each service reacts to a specific event using a message broker without relying on other components. This pattern not only increases the resilience of systems and fault isolation but also accomodates independent deployment and

scaling of services. Asynchronous communication also achieves additional performance benefit under heavy load such that the system is able to handle numerous transactions simultaneously with efficiency without bottlenecks and service conflicts. Such architectural strategies are particularly relevant for high-load environments, where distributed components must maintain operational independence and responsiveness despite fluctuating demand and complex interaction patterns [4].

In addition to loose coupling, scalable architectures rely on mechanisms that ensure rapid error detection. Another key principle is the fail fast approach, which means that the system should immediately detect and report any error, minimizing the risk of defect propagation [5]. This approach enables prompt reaction to failures and helps prevent the accumulation of hidden issues. It is especially vital in high-loaded e-commerce systems where errors in transaction processing, unavailability of external services, or logical failures can lead to data loss and interruption of service within a limited time (figure 4).

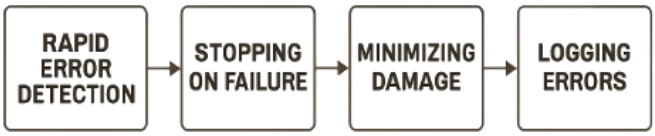

Figure 4: Fail fast implementation steps in e-commerce architecture

Fail fast implementation typically includes input validation, dependency checking, timeouts for external API calls, and required error logging. These phenomena guarantee that failures are contained and separated at the point of lowest level, and additional system interference is prevented.

Also, big distributed systems increasingly use self-healing mechanisms that attempt to repair failures with no human intervention. This is done through patterns such as circuit breakers, health-check APIs, replication, and automatic reboots of components once faults have been identified. The orchestrators of today, such as Kubernetes, enable us to maintain containerized self-healing infrastructure where failing components can be restarted, scaled, or replaced without dropping the user's experience. According to the CNCF (2024), there are over 80 % of enterprises already putting Kubernetes to use in production, and 52 % utilize containerization for the vast majority or all applications, which suggests high architectural maturity.

The combined application of these principles – modularity, loose coupling, fail fast, and self-healing – enables the development of a resilient, flexible, and scalable architecture capable of operating effectively under the high dynamism and unpredictable load patterns characteristic of e-commerce systems.

4. ARCHITECTURAL MODELS AND PATTERNS

The development of scalable information systems for e-commerce requires the application of proven architectural models and design patterns. These models enable the system to be structured in such a way that its components remain isolated, easily scalable, and capable of evolving independently (table 2).

Table 2: Architectural models and design patterns in e-commerce systems [6, 7]

Model / pattern	Brief description	Advantages	Implementation considerations
Microservice architecture (MSA)	Every component carries out a distinct business function and can be scaled individually.	Component isolation, flexible scaling, elimination of monolithic bottlenecks.	Requires DevOps culture, CI/CD pipelines, and orchestration (e.g., Kubernetes).
Serverless architecture	Running functions in the cloud without the need for server oversight. Developers concentrate exclusively on business logic.	Automatic scaling, reduced costs, fast deployment.	Limited execution time, debugging challenges, and stateless execution model.
CQRS (Command Query Responsibility Segregation)	Separates read and write operations into distinct models optimized for their respective tasks.	Simplified scaling, improved performance, flexible data handling.	Requires separate data stores: e.g., PostgreSQL for writes and ElasticSearch for reads.
Event sourcing	Stores a sequence of events leading to the current state instead of the state itself.	Full audit trail, rollback capability, supports event-driven integration.	Requires event logs, replay facilities, typing, and validation.
API gateway	Single point of entry to microservices, for routing, aggregation, authentication, and security.	Hides internal complexity, provides unified access, enhances scalability and security.	Requires support for SSL, CORS, rate limiting; common solutions include Kong, NGINX, AWS Gateway.

Existing design trends and architectures used during e-commerce information system development can provide high fault tolerance, flexibility, and scalability. Together, they enable digital platforms with better resilience to evolving business needs, highest load handling, and accelerated growth without massive architectural reconstruction and better user experience. In competitive e-commerce, the features are instrumental in ensuring business continuity and a quality user experience.

5. IMPLEMENTATION OF SCALABLE ARCHITECTURAL SOLUTIONS IN HIGH-LOAD E-COMMERCE SYSTEMS: AN ANALYSIS OF PRACTICAL CASES

The use of scalable architectural solutions to real e-commerce conditions not only means compliance with theoretical requirements, but also their adjustment to specific business processes and the patterns of workload. The most exemplary here are established by multi-scale e-commerce companies,

where high demands on performance, fault tolerance, and rapid provision of innovation drive the implementation in real-world conditions of cutting-edge architectural solutions.

Amazon, being among the top e-commerce players globally, actively utilizes microservice architecture and event-based designs to achieve scalability and system fault tolerance. Amazon Web Services (AWS) reports that the company processes over 600 million transactions daily on a distributed container-based platform, like Amazon ECS (Elastic Container Service) and Amazon EKS (Elastic Kubernetes Service). A key building block of such an architecture is serverless computation – here, AWS Lambda, employed to trigger functions in response to events such as order placement, delivery status events, or notice generation [8]. Event-driven architecture enables Amazon to dynamically scale resources without pre-booking, which is critical during times of peak demand such as seasonal sales.

As highlighted on the AWS Architecture Blog, event-driven processing pipelines have been implemented across business domains at Amazon through the use of Lambda, S3, Step Functions, and Amazon EventBridge to achieve complete automation and high performance in systems handling billions of units of data. Besides maximizing fault tolerance and scalability, this method also provides fine-grained event traceability essential for real-time observability and operational control across business processes.

Another example of real-world utilization of scalable architectural patterns is that of Walmart, which applies Event Sourcing and Command Query Responsibility Segregation (CQRS) patterns to support its high-traffic distributed systems. The Inventory Availability System was designed through an event-driven approach, permitting the platform to scale dynamically when increasing transaction volumes. Segregation of read and write operations based on CQRS standards improved transactional efficiency, and event sourcing ensured end-to-end traceability and state consistency. The system notably went through a throughput of over 5,000 operations per second on inventory reservation APIs under high robustness in maximum load scenarios and fault scenarios [9].

In addition, to optimize search infrastructure, Walmart also implemented several low-latency architectural strategies to reduce response time and increase reliability. Following the rollout of a multi-tier caching infrastructure (e.g., Memcached, Redis, and FastCache), the platform achieved an average search latency of around 120 ms, with query cache hit rates of 40% and as much as 80% for autocomplete suggestions. These results were obtained with heavy loads of operation and numerous asynchronous simultaneous requests, further confirming the effectiveness of architectural patterns in designing large-scale e-commerce sites.

One good case study of scalable architecture in the e-commerce sector is that of Wayfair, which is one of the largest US-based internet retailers of home goods. While part of its transition to microservices and event-driven architecture, Wayfair implemented a product data processing system that would handle over 80 million records daily, with its scalability up to 200 million records during peak loads, using Apache Kafka. According to Google Cloud, the company manages over 30 million SKUs from more than 20,000 suppliers, and its analytics platform – powered by Looker and BigQuery – processes over a million business intelligence queries each week. To attain high availability and modularity, the company has split its business logic into independent microservices regulated using Kubernetes in order to deploy updates speedily and scale individual components selectively without interfering with overall performance [10]. The design enables Wayfair to enjoy operational resilience amid explosively growing volumes of data and bursts of seasonal traffic, while delivering a consistent user experience and business operation agility.

Taken as a whole, these practical examples demonstrate that implementing scalable architecture solutions in e-commerce is not a theoretical ideal, but is irrevocably tied to the run-time concerns of massive-scale digital platforms. Microservices, event-driven patterns, distributed data management, and multi-tier caching enable global retailers to achieve high availability, maintain transactional consistency, and respond flexibly to varying demand. The evidence from Amazon, Walmart, and Wayfair indicates that fault tolerance and scalability are not merely technology requirements, but also strategic issues that have immediate consequences for customer satisfaction, market competitiveness, and the ability of businesses to sustain growth in the very dynamic e-commerce ecosystem. This strategic viewpoint of resilience and scalability in e-commerce architecture appeals to practices within financial management as well, where techniques such as portfolio immunization are employed so as to lower risks and ensure stability when exposed to dynamic environments [11]. Finally, such similarities help to reaffirm the agelessness of resilience-based strategies and communicate their usefulness to ensure long-term sustainability of digital commerce systems.

6. CONCLUSION

Architectural styles to e-commerce information system design are the foundation for ensuring their robustness, scalability, and flexibility in the dynamic virtual marketplace. Modularity, loose coupling of the modules, reactive and event-driven style enable systems to withstand peak loads, enable high availability of services, and provide business process continuity. It should be pointed out that efficiency of such systems also relies, aside from the choice of architectural patterns, on their comprehensive integration into operational practices like monitoring, automation, and the employment of cloud infrastructures.

The experience of e-commerce champions confirms that fault tolerance focus and architectural flexibility have become drivers of strategic competitiveness. In a context of growing data volumes and rising customer expectations, information system reliability and scalability are essential to ensure long-term business sustainability, minimize operations risk,

and facilitate innovative growth. Thus, architectural solutions in e-commerce need to be considered not just as a technology choice but as part of the integral strategy of digital transformation.

REFERENCES

1. Global E-commerce Software Market Size / Verified Market Research // URL: https://www.verifiedmarketresearch.com/product/ecommerce-software-market (date of application: 19.09.2025).
2. S. Bolgov. **Development of high-load backend systems for banking products: problems and solutions,** *Proceedings of the LIII International Multidisciplinary Conference «Innovations and Tendencies of State-of-Art Science». Mijnbestseller Nederland, Rotterdam, Nederland,* pp. 44-51, 2025.
3. M. Roilian. **Designing fault-tolerant distributed systems for supply chain management: architectural patterns and manufacturing scenarios,** *International Journal of Engineering in Computer Science,* vol. 7(2), pp. 12-17, 2025.
4. A. Bogutskii. **Architecture of high-load distributed systems: the case of a web crawler,** *Cold Science,* no. 19, pp. 41-52, 2025.
5. J. Willard, J. Hutson. **Fail Fast, Fail Small: Designing Resilient Systems for the Future of Software Engineering,** *SSRG International Journal of Recent Engineering Science,* vol.11(5), pp. 55-58, 2024.
6. I. N. Nasyrova. **Microservice architectures for financial platforms: challenges and solutions,** *Professional Bulletin: Information Technology and Security,* no. 2, pp. 49-55, 2025.
7. T. Mukayev. **Development of intelligent workflow automation systems based on the integration of RPA and AI models,** *Cold Science,* no. 17, pp. 15-27, 2025.
8. AWS Lambda / AWS Amazon // URL: https://aws.amazon.com/ru/lambda/ (date of application: 20.09.2025).
9. J. Lin, S. Yadav, F. Liu, N. Rossi, P. Suram, S. Chembolu, P. Chandran, H. Mohapatra, T. Lee, A. Magnani, C. Liao. **Enhancing Relevance of Embedding-based Retrieval at Walmart,** *InProceedings of the 33rd ACM International Conference on Information and Knowledge Management,* pp. 4694-4701, 2024.
10. Streamlining Access to Product Data with the MarTech Product Service / Wayfair // URL: https://www.aboutwayfair.com/careers/tech-blog/streamlining-access-to-product-data-with-the-martech-product-service (date of application: 20.09.2025).
11. Ye. Zharmagambetov. **Portfolio Immunization and Debt Maturity Management Strategies,** *Innovations and Investments*, no. 1, pp. 451-456, 2025/1.