Pak. j. life soc. Sci. (2024), 22(2):14040-14052 E-ISSN: 2221-7630;P-ISSN: 1727-4915

Pakistan Journal of Life and Social Sciences
www.pjlss.edu.pk

https://doi.org/10.57239/PJLSS-2024-22.2.001010

RESEARCH ARTICLE

Cross-Model Interaction Of Digital Twins Of Business Processes

Oleg Kazakov*

Bryansk State Engineering and Technology University, Bryansk, Russia

ARTICLE INFO

Received: Oct 12, 2024

Accepted: Nov 19, 2024

Keywords

Digital twin
Cross-model interaction
Business process
Orchestration
Management

***Corresponding Author**

ABSTRACT

The article describes an experiment to assess the effectiveness of different architectural styles for implementing the cross-model interaction of digital twins of business processes in real projects. An application for the experimental environment was developed to represent digital twins of business processes, facilitating cross-model interaction. The experiment results showed that event-driven orchestration of business processes is the most suitable style for developing the digital twin of the Warehouse Logistics business process. This architecture provides high scalability, flexibility, and performance.

INTRODUCTION

In current conditions, where competitiveness is determined not only by product quality but also by the efficiency of business processes (Gandolfi et al., 2024), digital twins enable the optimization of processes, prediction of system behavior, and increased production efficiency (Balova et al., 2022; Bagratuni et al., 2023; Abdullaev et al., 2023). Data flows between physical and virtual spaces become critically important for decision-making at all levels of management (Panasenko et al., 2024). However, implementing digital twins for business processes requires solving several complex issues related to scalability, flexibility, and integrating various systems (León-Valle et al., 2024). In Grieves & Vickers (2017), the authors emphasize the importance of data synchronization between the physical object and its digital twin to ensure effective management. S. Bhattacharya and R. Keller (2018) provide an overview of the technologies necessary to implement digital twins and discuss the key challenges developers face. Special attention is given to integrating data from different sources and formats, which is critical for cross-model interaction. A. Bernard and J. Rios conduct a systematic literature review on digital twin applications in business process management (Bernard & Rios, 2019). The authors highlight key research areas, including modeling, monitoring, and optimizing business processes, and stipulate the need to develop unified approaches to cross-model interaction. H. Niemann and H. Stuckenschmidt (2020) propose a semantic approach to modeling digital twins for business processes, allowing data integration from various sources and formats. The authors demonstrate how semantic models can ensure interaction and improve business process management. In recent years, two main approaches have been actively discussed: event-driven architecture (EDA) and microservices architecture (MSA). Gorton et al. (2017), Weyns et al. (2018), and Gao et al. (2022) compare these approaches in terms of their applicability to digital twins. However, the results of these studies do not provide a definitive answer regarding the effectiveness of implementing digital twins specifically for business processes. There is a need for further analysis and comparison of these architectural approaches and the development of new approaches to digital twin interaction in business processes (Kazakov et al., 2024; Goduni, 2024). Effective interaction between multiple digital twins significantly expands their functional capabilities. It can lead to the creation of more complex and better manageable systems (Eskerkhanova et al., 2023; Chumakova et al., 2024). Research into cross-model interaction approaches for digital twins in business processes opens new horizons for integrating and coordinating complex systems, presenting a relevant task that serves as the objective of our study.

1 MULTILAYER STRUCTURE OF THE DIGITAL TWIN OF BUSINESS PROCESSES

Before exploring different approaches to cross-model interaction of digital twins for business processes, it is essential to define a digital twin. We propose a solution that involves developing five layers at different levels of abstraction and virtual representation of the business process (Figure 1):

1. Spatio-temporal layer of the digital twin's virtual representation of the business process. This layer represents the execution of a business process within a specific coordinate system, considering time.

Purpose: To visualize the process participants' geographic location and temporal characteristics.

The key components are as follows:

– Integration with mapping services: Utilization of map service APIs to display the geographic coordinates of process participants,

– Timestamps: Monitoring and displaying timestamps for events, phase durations, delays, and other time-based characteristics of the process,

– Visualization of BPMN (Business Process Model and Notation) elements: Development of a method in the business process class interface that returns images of BPMN elements executed up to a specific time based on real process data.

2. Logical-structural layer of the digital twin's business process virtual representation. This layer implements the process model.

Purpose: To represent the logical structure and relationships between the elements of the business process.

The key components are identified according to the selected modeling notations using DMN (Decision Model and Notation) and BPMN.

3. Resource-infrastructure layer of the digital twin's business process representation. This layer represents all types of resources involved in executing the business process.

Purpose: To visualize the resources and infrastructure used in the process.

The key components are as follows:

– Resource identification,

– Resource modeling: Creating models of resources, considering their properties and parameters,

– Resource visualization: Displaying resources on process diagrams and using infographics to show resource availability and load.

4. Data flow layer of the business process. At this level, input and output information is tracked within the context of data flows generated by business process execution.

Purpose: To visualize data flows within the process.

The key components are as follows:

– Data flow identification,

– Data flow visualization: creating models of data flow.

5. Metric-analytic layer of the digital twin's representation of the business process. At this level, the monitoring and analysis of business process execution are conducted using dashboards and other visualization tools.

Purpose: To visualize key performance indicators (KPIs) of the process.

The key components are as follows:

– KPIs: Identifying KPIs to be monitored,

– Data collection subsystem: Gathering data for KPI calculation from various sources, such as accounting systems, sensors, event logs, etc.,

– Dashboard creation: Developing interactive dashboards where KPIs are displayed through graphs, charts, tables, and other formats,

– Data analysis: Providing tools for data analysis to identify trends, gaps, and issues.

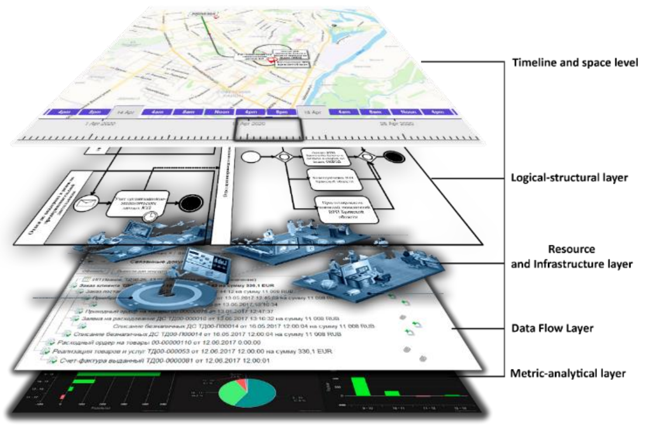


Figure 1: Multilayer structure of the digital twin of business processes

This multilayer representation of business processes helps consider all aspects of process execution, and implementing this approach provides an effective environment for managing all facets of the business process.

2 ARCHITECTURAL STYLES APPLICABLE FOR THE CROSS-MODEL INTERACTION OF DIGITAL TWINS

Let us consider approaches to the cross-model interaction of digital twins based on the following styles:

1. Layered architecture,

2. Service-oriented architecture (SOA),

3. MSA,

4. EDA,

5. Event-driven orchestration of business processes.

Starting from the multilayer structure of the digital twin of business processes, it is logical to examine the approach to the cross-model interaction of digital twins based on layered architecture. Layered architecture is a powerful tool for designing complex and scalable systems (Chumakova et al., 2023). It provides modularity, flexibility, and code reusability. However, when designing a layered architecture, it is important to consider its complexity and potential impact on performance. The main elements of layered architecture can be represented as follows:

1. Presentation layer: This layer is responsible for user interaction. The main components include user interfaces, web pages, and API interfaces. The key functions implemented at this level include data display, user input processing, and data validation.

2. Business logic layer: This layer contains business rules and application logic. The main components include business objects, services, and controllers. The key functions implemented at this level are the execution of business operations, data processing, and decision-making.

3. Data access layer: This layer is responsible for interaction with data storage. Main components include DAO objects, ORM frameworks, and SQL queries. The key functions implemented at this level include reading and writing data, transaction management, and data caching.

4. Data layer: This layer is responsible for data storage. The main components include databases, file systems, and external APIs. The key functions at this level are data storage, data access provisioning, and data management.

Layered architecture is a powerful tool for designing complex and scalable systems. It provides modularity, flexibility, and the ability to reuse code. However, when designing a layered architecture, it is important to consider its complexity and the potential decrease in performance.

The next architectural style for the cross-model interaction of digital twins of business processes is SOA. This approach to software development structures the application as a set of loosely coupled services. Each service performs a specific function and can be invoked independently of other services. SOA allows for increased flexibility, scalability, and code reuse. The main elements of this style include:

1. Services: The fundamental building blocks of SOA. Each service is an autonomous component that performs a specific business function,

2. Loose coupling: Services should be loosely coupled, meaning they should interact with each other through clearly defined interfaces and should not depend on each other's internal implementations,

3. Unified interface: All services should provide a unified interface that defines how they can be accessed and how they interact,

4. Message-based services: Services typically interact with each other through messages that are transmitted over the network,

5. Open standards: SOA encourages open standards for service interaction, simplifying integration and improving compatibility.

MSA is a software development approach where an application is built as a set of small, autonomous services, each performing a narrowly specialized function. Each microservice operates independently and can be developed, deployed, and scaled separately from others.

Here are the key components of MSA:

1. Small and autonomous services: Each service performs a specific task and can be developed, deployed, and scaled independently of other services. Services may use different technologies, programming languages, and databases based on the requirements.

In addition, depending on the patterns of MSA, the following components can be identified:

1. API gateway: Provides a single entry point for clients by routing requests to the appropriate services. It distributes requests among service instances and ensures security by verifying client access rights,

2. Service discovery: Allows services to find each other in a dynamic environment, where services can be deployed and removed at any time. It stores information about available services and their locations.

A popular architectural style used in implementing digital twins for products is EDA, which focuses on creation, detection, consumption, and response to events. In EDA, the system is divided into independent components that interact with each other via events. The main elements of this style include:

1. Event: Any significant change in state that may affect the system (e.g., a new order, a change in customer status, a payment failure),

2. Event source: The component that generates the event (e.g., a web server, application, sensor),

3. Event consumer: The component that responds to the event (e.g., an order processing system, notification system, analytics system),

4. Event channel: The means of transmitting events from the source to consumers (e.g., message broker, data bus),

5. Event stream: A sequence of events that are generated and processed within the system.

Event-driven business process orchestration is an approach to automating and managing business processes based on reacting to events occurring within the system. This approach allows one to create dynamic, flexible, and responsive systems that adapt to changing business conditions and requirements.

The main components are as follows:

1. Events: These are signals or notifications that something has happened in the system. Events can be triggered by user actions, data changes, messages from other systems, etc.,

2. Business processes: These are sequences of tasks and actions that need to be performed to achieve a specific business goal. Business processes may include multiple steps and involve interactions with systems and services,

3. Orchestrator: This is the central component responsible for managing and coordinating business processes based on events. The orchestrator determines which processes should be triggered or continued in response to specific events. It also monitors the state of processes and manages their lifecycle,

4. Services and systems: These are components that perform specific tasks within business processes. They can be implemented as microservices, web services, functions, etc.,

5. Event broker: This component is responsible for transmitting events between different systems and services. It ensures reliable event delivery and the system's scalability.

3 METHODS

The main goal of the experiment is to assess the effectiveness of applying different architectural styles for implementing the cross-model interaction of digital twins of business processes in real-life projects. The experiment aims to study the impact of architectural styles on the following aspects:

1. Performance: Evaluating how different architectural styles affect response time and throughput of services,

2. Reliability: Investigating how resilient the implemented systems are to failures and how quickly they recover,

3. Scalability: Determining how well different architectural styles handle increased load and the ability to add new services.

The experiment was conducted using a mixed approach, combining quantitative metrics and qualitative feedback from developers. Developer teams were formed, with each team responsible for implementing an architectural approach. The teams consisted of 2-4 people and were balanced in terms of experience and knowledge.

During the experiment, data was collected based on the following parameters:

- Implementation time: Time tracking was done using built-in tools in the YouGile project management system,
- Performance: Load testing was conducted using Apache JMeter,
- Resource usage: CPU and memory consumption were monitored using Prometheus and Grafana tools.

To compare various architectural styles for implementing the cross-model interaction of digital twins of business processes, the task was to develop an application to represent the digital twins of business processes. Figure 2 presents two digital twins within the domain of Warehouse Logistics. The processes being considered must include the following actions:

1. Registration of goods receipt,

2. Storage of goods information,

3. Tracking the location of goods in the warehouse,

4. Registration of goods shipment.

For each architecture, a separate implementation of the digital twin was developed. Let us examine the details of the implementation:

1. **Layered architecture**. The technological stack used includes Java, Spring, and MySQL. The system was divided into three main layers:

Presentation layer: Implemented using Spring MVC to handle HTTP requests and display data to the user.

Business logic layer: Implemented using Spring Services, which encapsulates the core application logic.

Data access layer: Implemented using Spring Data JPA to interact with the MySQL database.

2. **SOA**. The technological stack used includes Java, Apache Camel, and MySQL. The system was divided into several independent services, each performing a specific function:

Goods arrival service: Responsible for registering the arrival of goods.

Information storage service: Responsible for storing information about the goods.

Location tracking service: Responsible for tracking the location of goods in the warehouse.

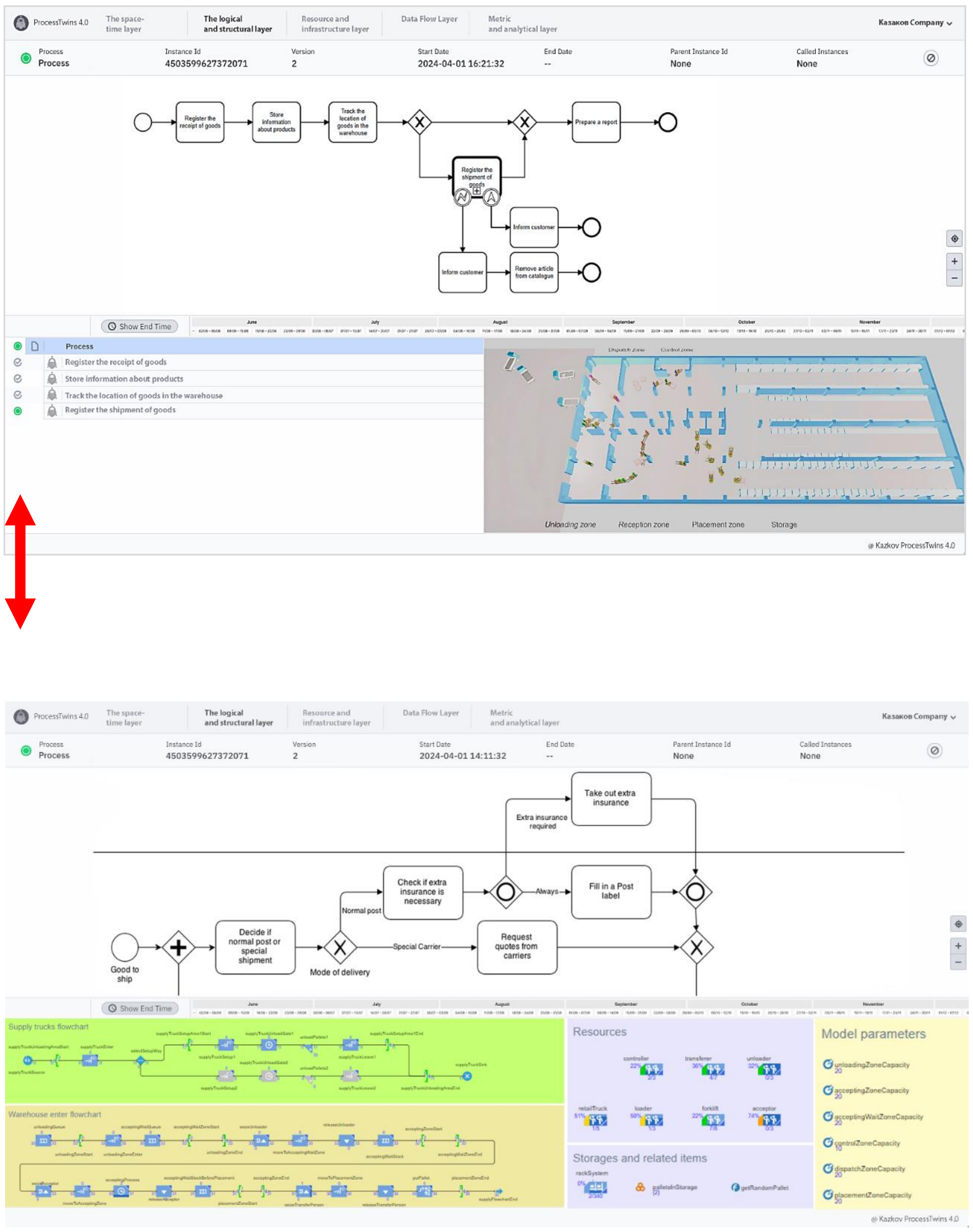
Goods shipment service: Responsible for registering the shipment of goods.

The services interact with each other via SOAP or REST API.

3. **MSA**. The technological stack used includes Node.js, Express, and MongoDB. The system was divided into multiple small, autonomous services:

Goods arrival service: Implemented using Node.js and Express.

Information storage service: Implemented using Node.js and Express.


ProcessTwins 4.0
The space-time layer
The logical and structural layer
Resource and infrastructure layer
Data Flow Layer
Metric and analytical layer
Казаков Company
Register the receipt of goods
Store information about products
Track the location of goods in the warehouse
Prepare a report
Register the shipment of goods
Inform customer
Remove article from catalogue
Show End Time
Dispatch zone
Control zone
Unloading zone
Reception zone
Placement zone
Storage
© Kazkov ProcessTwins 4.0
Take out extra insurance
Extra insurance required
Check if extra insurance is necessary
Normal post
Always
Fill in a Post label
Decide if normal post or special shipment
Special Carrier
Request quotes from carriers
Good to ship
Mode of delivery
Supply trucks flowchart
Warehouse enter flowchart
Resources
Storages and related items
Model parameters
unloadingZoneCapacity
acceptingZoneCapacity
acceptingWaitZoneCapacity
controlZoneCapacity
dispatchZoneCapacity
placementZoneCapacity
© Kazkov ProcessTwins 4.0

Process	Instance Id	Version	Start Date	End Date	Parent Instance Id	Called Instances
Process	4503599627372071	2	2024-04-01 16:21:32	--	None	None

Process
Register the receipt of goods
Store information about products
Track the location of goods in the warehouse
Register the shipment of goods

Process	Instance Id	Version	Start Date	End Date	Parent Instance Id	Called Instances
Process	4503599627372071	2	2024-04-01 14:11:32	--	None	None

Figure 2: Digital twins implemented for the experiment in the domain of Warehouse Logistics

Location tracking service: Implemented using Node.js and Express.

Goods shipment service: Implemented using Node.js and Express.

Each microservice interacts with the MongoDB database and can be independently deployed, scaled, and updated without affecting the other services.

4. EDA. The technology stack used includes Node.js, Kafka, and MongoDB. The system was designed around event processing:

Goods arrival event: Generated when goods arrive at the warehouse.

Information storage event: Generated when product information is stored.

Location tracking event: Generated when tracking the location of goods within the warehouse.

Goods shipment event: Generated when goods are shipped from the warehouse.

The system uses Kafka for event handling and enabling real-time processing and asynchronous communication between components and MongoDB for data storage.

5. Event-driven business process orchestration. The technology stack used includes Node.js, Camunda, Kafka, and MongoDB. The system was built using Camunda to orchestrate business processes and Kafka for event handling:

Goods arrival process: Defines the sequence of actions when goods arrive at the warehouse.

Information storage process: Specifies the steps involved in storing product information.

Location tracking process: Outlines the actions required for tracking the location of goods within the warehouse.

Goods shipment process: Establishes the sequence of actions for shipping goods from the warehouse.

4 RESULTS

After collecting the metrics, the data were analyzed, and the results are presented in Tables 1 and 2.

Table 1: Experiment progress data

Pattern	Average time to implement (hours)	Average throughput (requests/sec)	Average memory usage (MB)
Layered architecture	15	110	200
SOA	20	130	150
MSA	25	120	250
EDA	30	140	300
Event-driven business process orchestration	35	180	350

Table 2. Advantages and disadvantages of using different architectural styles for the cross-model interaction of digital twins

Approach	Pros	Cons
Layered architecture	1. Separation of concerns: each layer performs its specific functions, facilitating better code organization and making the system easier to understand. 2. Simplified testing: it is easy to test individual layers, enabling modular testing and increasing confidence in code quality. 3. Flexibility and scalability: individual layers can be modified without affecting others, allowing one to integrate new technologies and approaches. 4. Code reusability: layers or their components can be reused across different projects or applications. 5. Simplified maintenance: errors can be found and corrected more easily, and system updates are more manageable.	1. Performance: the multi-layer structure can reduce performance due to the overhead of data transfer between layers.
SOA	1. Increased flexibility: SOA allows for easy modification and extension of applications, as services can be developed, deployed, and updated independently. 2. Scalability: applications can be scaled by adding more service instances as needed. 3. Enhanced reusability: services can be reused across various applications and contexts. 4. Improved integration: SOA facilitates application integration, as services can provide shared functionalities and data.	1. Overhead: SOA can increase overhead related to message transmission and service coordination. 2. Performance: interactions between services may lead to performance degradation.
MSA	1. Independent development: different teams can work on separate microservices independently, speeding up development. 2. Frequent releases: microservices can be deployed individually, allowing for more frequent releases of new features and fixes. 3. Horizontal scalability: only high-load microservices can be scaled, avoiding the need to scale the entire application.	1. Network latency: communication between microservices over the network can introduce latency and reduce performance. 2. Transaction management: managing distributed transactions is complex and requires specialized approaches.

	4. Vertical scalability: each microservice can be optimized for its specific resource requirements. 5. Fault isolation: failure in one microservice does not affect the entire application if the impact is isolated. 6. Quick recovery: a failed microservice can be swiftly replaced or restored. 7. Technology diversity: different microservices can use various technologies and programming languages, enabling the best solution for each task.	3. Distributed data: data spread across multiple microservices makes consistency and integrity challenging. 4. Data duplication: data may be duplicated across microservices, necessitating extra effort for synchronization.
EDA	1. Flexibility: the system can adapt to real-time changes, enabling rapid responses to new opportunities and challenges. 2. Speed: events are processed instantly, reducing request processing time and accelerating product and service delivery. 3. Scalability: easy scale to handle large event volumes, supporting business growth. 4. Resilience: independent components can continue operating even when others fail. 5. Transparency: events and actions are logged, enhancing system transparency and simplifying tracking and analysis. 6. Openness: designed for integration with other systems and technologies, increasing flexibility and extensibility.	1. Complexity: developing and maintaining complex EDA-based systems can be time-consuming and costly. 2. Dependency on technology: EDA requires reliable, scalable technologies for real-time event processing. 3. Unpredictability risk: unexpected event sequences may result in unforeseen outcomes and errors. 4. Training requirements: team members need training to work with new EDA technologies and methodologies. 5. Implementation cost: EDA may require significant investment in technology, training, and business process adjustments. 6. Event stream management: managing and routing large volumes of events can be challenging. 7. Debugging and monitoring: debugging and monitoring EDA systems can be complex due to asynchronous component interactions.
Event-driven business	1. Flexibility and adaptability: business processes can be easily modified and expanded without significant code changes.	1. Design and management complexity: the system can become challenging to manage

process orchestration	2. Responsiveness: the system can quickly respond to events, ensuring the fast execution of business processes. 3. Scalability: it can handle many events and processes in parallel, making it highly scalable. 4. Enhanced fault tolerance: the orchestrator can automatically restart processes in case of failure, increasing system reliability. 5. Simplified integration: events provide a unified method for interactions between different systems and services.	and monitor, especially with many events and processes. 2. Risk of event storm: a high volume of events can lead to system overload, reducing performance. 3. Data consistency challenges: in a distributed system with numerous events and processes, ensuring data consistency can be complex.

5 DISCUSSION

Each approach to using different architectural styles for the cross-model interaction of digital twins has advantages and disadvantages. The results confirmed the hypothesis that selecting the right MSA significantly impacts the performance and reliability of the application:

1. Scalability

– Layered architecture: Low scalability. The system cannot efficiently handle many requests,

–SOA: Average scalability. Enables scaling of individual services but with high overhead costs,

– MSA: High scalability. Allows independent scaling of each microservice,

– EDA: High scalability. Capable of handling asynchronous events,

– Event-driven orchestration: High scalability. Facilitates automation and scaling of complex business processes.

2. Flexibility

– Layered architecture: Low flexibility. It is difficult to make changes within the system,

–SOA: Average flexibility. Enables reuse of services,

– MSA: High flexibility. Allows for the use of different technologies for various microservices,

– EDA: High flexibility. Capable of handling asynchronous events,

– Event-driven orchestration: High flexibility. Facilitates automation and modification of business processes.

3. Performance

– Layered architecture: Average performance. Limited capacity to handle many requests,

–SOA: Average performance. High overhead for service interaction,

– MSA: High performance. Allows optimization of each microservice individually,

– EDA: High performance. Capable of processing asynchronous events,

– Event-driven orchestration: High performance. Enables optimization of business processes.

4. Development complexity

– Layered architecture: Low complexity. Simple to develop and understand,

–SOA: Average complexity. Challenges in service management and monitoring,

– MSA: High complexity. It is difficult to manage and monitor multiple microservices,

– EDA: High complexity. Complexity in debugging and monitoring,

– Event-driven orchestration: High complexity. Complex to design and manage the orchestrator.

6 CONCLUSIONS

The experiment results proved that the event-driven orchestration of business processes is the most suitable style for developing the digital twin of the Warehouse Logistics business process. This architecture provides high scalability, flexibility, and performance. As part of this study, an application was developed to implement an approach to cross-model interaction, which includes data integration from various sources, model synchronization, and interaction between different levels of abstraction.

The next step in our research is a more in-depth study of use cases for applying the event-driven orchestration of business processes for the cross-model interaction of digital twins in industrial companies.

ACKNOWLEDGEMENTS

The study was supported by a grant from the Russian Science Foundation No. 23-28-00180, URL: https://rscf.ru/project/23-28-00180/.

AUTHORS' CONTIBUTIONS

OK was responsible for all parts of the research.

REFERENCES

Abdullaev I, Prodanova N, Ahmed MA, Joshi GP, Cho W, 2023. Leveraging metaheuristics with artificial intelligence for customer churn prediction in telecom industries. Electronic Research Archive, 2023, 31(8): 4443-4458. doi: 10.3934/era.2023227.

Bagratuni K, Kashina E, Kletskova E, Kapustina D, Ivashkin M, Sinyukov V, Karshalova A, Hajiyev H, Hajiyev E, 2023. Impact of socially responsible business behavior on implementing the principles of sustainable development (experience of large business). International Journal of Sustainable Development and Planning, 18(8): 2481-2488. https://doi.org/10.18280/ijsdp.180819

Balova S, Orlova I, Konovalova E, Repina M, Shichkin I, 2022. Social Media Marketing (SMM) Impact on Hotel Business Development: Private Mini Hotel Experience. Anais Brasileiros De Estudos Turísticos, 12 (Special Issue). DOI: https://doi.org/10.5281/zenodo.7154757

Bernard A, Rios J, 2019. Digital Twin for Business Process Management: A Systematic Literature Review. Business & Information Systems Engineering, 61(5): 577-594.

Bhattacharya S, Keller R, 2018. Digital Twin: Enabling Technologies, Challenges, and Open Research. IEEE Access, 6: 19652-19671.

Chumakova E, Korneev D, Gasparian M, Titov V, Makhov I, 2024. Developing a neural network to assess staff competence and minimize operational risks in credit organizations. International Research Journal of Multidisciplinary Scope, 5(2): 461-471.

Chumakova EV, Korneev DG, Gasparian MS, Ponomarev AA, Makhov IS, 2023. Building a neural network to assess the level of operational risks of a credit institution. Journal of Theoretical and Applied Information Technology, 101(11): 4205.

Eskerkhanova LT, Beloglazova LB, Masyutina NM, Romanishina TS, Turishcheva TB, 2023. Increasing the competitiveness of future economists for work in industry 4.0. Perspektivy nauki i obrazovania, 62(2): 158-173. doi: 10.32744/pse.2023.2.9.

Gandolfi F, Stone S, Gano M, Nasrah SKM, 2024. Toxic Engagement: Identifying Toxic Leadership through Employee Engagement to Benefit the SDGs. Journal of Lifestyle and SDGs Review, 5(1): e03164. https://doi.org/10.47172/2965-730X.SDGsReview.v5.n01.pe03164

Gao Y, Chang D, Chen CH, Xu Z, 2022. Design of digital twin applications in automated storage yard scheduling. Advanced Engineering Informatics, 51: 101477.

Goduni J, 2024. Empowering a Knowledge-Based Economy: An Assessment of the Influence on Economic Development. Theoretical and Practical Research In Economic Fields, 15(3): 754-763. doi:10.14505/tpref.v15.3(31).19.

Gorton G, 2017. The history and economics of safe assets. Annual Review of Economics, 9(1): 547-586.

Grieves M, Vickers J, 2017. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In: Transdisciplinary Perspectives on Complex Systems. Springer, Cham, pp: 85-113.

Kazakov O, Azarenko N, Kozlova I, 2024. Developing a Method for Building Business Process Models Based on Graph Neural Networks in the Absence of Task Identifier Data. Quanah Academic Journal, 4(1): 19-25. https://doi.org/10.58429/qaj.v4n1a333

León-Valle BW, Gómez MKB, Rodríguez EAP, Vélez WZ, Cobos-Alvarado F, Peñaherrera-León M, 2024. ICTS as Tools for Local Development: Santa Elena Province, A View from the SDGs. Journal of Lifestyle and SDGs Review, 5(1): e02611. https://doi.org/10.47172/2965-730X.SDGsReview.v5.n01.pe02611

Niemann H, Stuckenschmidt H, 2020. Towards a Digital Twin for Business Processes: A Semantic Modeling Approach. In: Proceedings of the 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, pp: 1123-1127.

Panasenko S, Fedyunin D, Osipova E, Star I, Zotova A, Iavdoliuk E, 2024. Features, factors, and risks of the influence of metaverses on business development in the digital economy: Russian and international context. Revista Relações Internacionais do Mundo Atual, 1(43).

Weyns D, 2020. An introduction to self-adaptive systems: A contemporary software engineering perspective. John Wiley & Sons.