

Case Study

Modernizing Enterprise Middleware: A Decision Framework for Migrating BizTalk Orchestrations to Azure-Based Integration Architecture

Venkateswarlu Mulla

Independent Researcher, Atlanta, GA, USA.

Corresponding Author : venkatmulla@gmail.com

Received: 20 July 2026

Revised: 29 August 2026

Accepted: 12 September 2026

Published: 29 September 2026

Abstract - Migrating a mature enterprise integration environment to Azure involves more than replacing one middleware platform with another. Existing interfaces differ in orchestration complexity, business importance, technical dependencies, and suitability for cloud-based services. Applying one migration strategy to every interface can create unnecessary redesign work or carry legacy limitations into the target architecture. This paper presents an Integration Migration Decision Framework (IMDF) developed and applied during an enterprise migration from Microsoft BizTalk Server to Azure Integration Services. The framework assesses each interface using three dimensions: Orchestration Complexity, Business Criticality, and Cloud-Native Fitness. Based on these dimensions, each interface is assigned to one of four migration dispositions: lift-and-shift, re-architect, replace, or retain and reassess. The framework was applied to 200 interfaces processing approximately five million transactions per month across IBM AS/400, SAP S/4HANA, SaaS applications, SQL Server, and reporting platforms. The portfolio was classified into 25 lift-and-shift interfaces, 86 re-architected interfaces, 56 replacement candidates, and 33 retained or reassessed interfaces. All 200 interfaces were migrated through 16 waves over one year and eight months. The contribution of IMDF is not the introduction of new migration labels. Those labels overlap with established cloud migration strategies. The contribution is an integration-specific assessment method that connects technical complexity, business risk, cloud suitability, migration sequencing, and operational architecture. The paper also describes hybrid connectivity and staging database isolation patterns used to address legacy-system dependencies and recovery requirements.

Keywords - Enterprise Middleware Modernization, BizTalk Server Migration, Azure Integration Services, Hybrid Integration Architecture, Event-Driven Architecture, Serverless Integration, Lift-And-Shift, Legacy System Modernization, AS/400 Integration, SAP S/4HANA.

1. Introduction

Enterprise integration platforms often remain in production long after the applications around them have changed. New systems are connected to the existing middleware environment, business processes are modified, and temporary solutions become permanent parts of the integration landscape. Over time, the middleware platform becomes responsible for much more than message routing. It may also contain transformation logic, business rules, scheduling, exception handling, correlation, recovery procedures, and operational knowledge.

Microsoft BizTalk Server is an example of this type of long-running enterprise platform. BizTalk can support a wide range of integration requirements, but many implementations contain dependencies on older protocols, custom components, complex orchestrations, and platform-specific operational procedures. When an organization begins moving

toward Azure, the technical challenge is not limited to converting BizTalk artifacts into Logic Apps workflows.

The more important question is how each existing interface should be treated. Some interfaces may be simple enough to migrate with limited functional change. Others may contain complex orchestration or business-critical processing and require architectural redesign. Some may be better replaced by a native SaaS capability or a new API. Other interfaces may need to remain on BizTalk temporarily because their dependencies or migration risks are not yet acceptable.

General cloud migration frameworks provide useful categories such as rehost, replatform, refactor, replace, retire, and retain [3]. However, these categories are usually discussed in the context of applications or workloads. Enterprise integration interfaces have additional characteristics. They connect systems rather than operate independently, and their risk is influenced by source and



target behavior, message volume, transformation rules, business deadlines, downstream dependencies, and recovery requirements.

This creates a practical research and architecture gap. Existing migration categories explain the possible destinations, but they do not by themselves provide enough guidance for deciding which disposition is appropriate for an individual middleware interface. A simple file transfer and a multi-step order-processing orchestration may both be deployed in the same BizTalk application, yet their migration requirements can be very different.

The migration program described in this paper involved 200 interfaces processing approximately five million transactions per month. The connected environment included IBM AS/400 systems, SAP S/4HANA, SaaS applications, SQL Server databases, and reporting platforms. The portfolio had developed over more than ten years and included both straightforward integrations and workflows with significant technical and business dependencies.

To support migration planning, an Integration Migration Decision Framework was developed and applied at the interface level. The framework uses three dimensions:

1. Orchestration Complexity
2. Business Criticality
3. Cloud-Native Fitness

The assessment produces one of four dispositions:

1. Lift-and-shift
2. Re-architect
3. Replace
4. Retain and reassess

The paper makes three main contributions. First, it adapts established cloud migration thinking to the specific characteristics of enterprise integration interfaces. In addition, it provides explicit scoring guidance and decision rules that can be used during architecture review. A further consideration, it documents two architectural patterns used during the migration: hybrid on-premises-to-cloud connectivity and staging database isolation.

The paper does not claim that IMDF is a statistically validated prediction model. It is a structured architecture decision aid developed from an actual enterprise migration program. The analysis focuses on how the framework was used, what the classification revealed, and what additional architecture and operational issues became visible during implementation.

2. Related Work and Research Gap

2.1. Enterprise Integration

Enterprise Application Integration has traditionally addressed the coordination of heterogeneous applications

through message exchange, transformation, routing, and process coordination [1], [2]. Enterprise Integration Patterns established a widely used vocabulary for message channels, routers, translators, process managers, correlation, and reliable delivery [1].

These concepts remain relevant when modernizing BizTalk environments. A migration still needs to answer questions about message delivery, transformation, routing, state, failure handling, and recovery. The difference is that the target architecture may distribute these responsibilities across several cloud services instead of providing them through one centralized middleware platform.

Service-oriented architecture and microservice approaches further emphasized independently deployable services, APIs, and decentralized processing [2]. These approaches can support modernization, but they also introduce additional decisions concerning service boundaries, messaging, state management, and operational ownership.

2.2. Cloud Migration and Modernization

Cloud migration research and industry frameworks commonly describe migration as a set of possible strategies rather than one universal process. Rehosting, replatforming, refactoring, replacing, retiring, and retaining are often used to classify application modernization decisions [1], [2].

These categories are useful for an integration program, but the categories alone do not explain how an interface should be assessed. For example, rehosting an application may be relatively straightforward when the application is self-contained. An integration interface is different because it may depend on the availability, protocol, timing, and behavior of multiple external systems.

Legacy modernization research also identifies the importance of dependency analysis, business risk, technical debt, migration effort, and incremental delivery. These concerns are particularly relevant to middleware because the integration layer often contains undocumented dependencies between otherwise independent applications.

2.3. Cloud and Hybrid Integration

Azure Integration Services provides several capabilities relevant to BizTalk modernization, including Logic Apps, Azure Functions, Service Bus, API Management, Event Grid, and Azure Monitor. These services support workflow execution, custom processing, messaging, API exposure, and monitoring.

However, moving to Azure does not eliminate on-premises dependencies. Enterprise environments may continue to rely on IBM AS/400, SAP systems, databases, file exchanges, and other systems that cannot be modernized at the same time as the integration platform. Hybrid

integration, therefore, requires more than network connectivity. It requires decisions about data extraction, buffering, processing state, retry behavior, reconciliation, and operational ownership.

2.4. Research Gap

The existing literature and migration guidance provide important foundations, but three practical gaps remain.

Broad cloud migration categories do not sufficiently distinguish among individual integration interfaces within the same application or platform — a single BizTalk application can contain both trivial and highly complex workflows, yet general frameworks treat it as one unit. A related gap is that integration-specific migration decisions need technical complexity, business criticality, and cloud suitability to be weighed together rather than separately; considering only one factor can produce misleading results. Finally, the migration decision cannot be separated from the target operating model. An interface that appears suitable for direct migration may still require a different recovery, monitoring, or state-management design once it runs in Azure.

IMDF addresses these gaps by applying established migration dispositions through an integration-specific assessment process. Its novelty is therefore not the creation of new disposition names. The novelty is the combination of interface-level assessment, three integration-focused dimensions, explicit decision guidance, migration sequencing, and architecture patterns for hybrid operation and recoverability.

3. Integration Migration Decision Framework

3.1. Interface-Level Assessment

The IMDF uses the interface or logical workflow as the unit of assessment.

A BizTalk application may contain receive locations, send ports, maps, schemas, pipelines, orchestrations, and shared components. These artifacts do not necessarily have the same migration characteristics. Assessing the entire BizTalk application as one unit can conceal important differences.

For example, a scheduled file transfer may share an application with a complex orchestration that correlates messages from several systems. The file transfer may be suitable for a relatively direct migration, while the orchestration may require redesign and more extensive testing.

The interface-level assessment includes:

- Source and target systems
- Message and transaction characteristics

- Transformation requirements
- Orchestration behavior
- Dependencies on other interfaces
- Error and recovery procedures
- Business purpose
- Current ownership
- Potential Azure implementation options

3.2. Orchestration Complexity

Orchestration Complexity, or OC, measures the technical difficulty of reproducing or redesigning an interface.

The following factors are considered:

- Number of processing steps
- Conditional branches
- Correlation requirements
- Transformation complexity
- Custom pipelines or components
- Number of participating systems
- Retry and exception behavior
- Stateful processing
- Manual recovery requirements

The score is assigned using the following guidance.

OC score	Assessment guidance
1 – Low	Single-step or limited-step flow; basic routing or transformation; no significant state or correlation
2 – Medium	Multiple processing steps; moderate transformation; some exception handling or system dependencies
3 – High	Complex orchestration; correlation, branching, custom components, durable state, or multi-system recovery

The score does not indicate whether an interface is well or poorly designed. It indicates the amount of technical analysis and migration effort likely to be required.

3.3. Business Criticality

Business Criticality, or BC, measures the effect of failure or prolonged interruption.

The assessment considers:

- Revenue or order-processing impact
- Customer or partner visibility
- Financial reporting relevance
- Regulatory or compliance implications
- Recovery time requirements
- Transaction volume and frequency
- Dependency on other business processes
- Risk of incomplete or duplicate processing

The scoring guidance is:

BC score	Assessment guidance
1 – Low	Internal or low-impact process; failure can be handled through normal support procedures
2 – Medium	Important operational process; failure affects a department, partner, or scheduled business activity
3 – High	Revenue, order, financial reporting, customer-facing, regulatory, or time-sensitive process

This score is deliberately kept separate from technical complexity — a file transfer that is trivial to rebuild can still carry a high BC score if, for example, it feeds daily order confirmations to customers.

3.4. Cloud-Native Fitness

Cloud-Native Fitness, or CF, measures how naturally an interface can be implemented and operated using Azure services.

The assessment considers:

- Availability of Azure connectors
- Availability of APIs
- Fit with Logic Apps
- Need for Azure Functions
- Messaging requirements
- Hybrid connectivity
- Latency sensitivity
- Persistent state requirements
- Monitoring and operational support
- Security and identity requirements

The scoring guidance is:

CF score	Assessment guidance
1 – Low	Limited connector or API support; strong legacy dependency; difficult cloud operating fit
2 – Medium	Partial Azure fit; hybrid connectivity or custom processing required
3 – High	Suitable APIs or connectors; good fit with Logic Apps, Functions, Service Bus, or API Management

Complexity and cloud fitness are not the same thing here. Even a highly complex interface can score well on CF if Azure already provides the patterns and services it needs — durable orchestration, messaging, and connectors, for instance.

3.5. Migration Dispositions

The three dimensions lead to four practical dispositions.

3.5.1. Lift-and-Shift

Used when the interface is relatively straightforward, business risk is manageable, and Azure provides a reasonable technical fit. The goal is to migrate with limited functional redesign while still adapting deployment, security, monitoring, and operational procedures to Azure.

3.5.2. Re-Architect

Re-architecture is appropriate when reproducing the existing workflow would carry forward limitations or create operational risk.

Typical reasons include:

- High orchestration complexity
- High business criticality
- Weak recovery design
- Excessive coupling
- Need for asynchronous processing
- Need for a durable state
- Poor observability
- Dependence on custom BizTalk behavior

A re-architected solution may divide responsibilities among Logic Apps, Azure Functions, Service Bus, Azure SQL, and API Management.

3.5.3. Replace

Replacement is appropriate when the original integration can be removed or provided through another capability.

Examples include:

- Native SaaS integration
- A new API replacing a file-based exchange
- A business application absorbing a previous middleware function
- A business process that is no longer required

Replacement reduces the number of workloads that need to be reproduced in Azure.

3.5.4. Retain and Reassess

Retention is appropriate when immediate migration is not justified by the current risk and benefit assessment.

Possible reasons include:

- Unusual technical dependencies
- Limited business value
- Insufficient Azure support
- High migration risk
- Dependence on a system scheduled for replacement
- Lack of a suitable owner or target design

Retention is not a permanent exemption. The reason for retention and a future review condition should be recorded.

3.6. Decision Rules

The following rules were used as practical guidance during the architecture review.

Assessment condition	Recommended disposition
OC 1–2, BC 1–2, and CF 2–3	Lift-and-shift
OC 2–3 with high BC or significant state/recovery requirements	Re-architect
Native SaaS/API capability removes the need for the existing flow	Replace
OC 3 with CF 1, or low business value with high migration risk	Retain and reassess

An interface may be assigned a different disposition after considering security, ownership, licensing, dependency analysis, regulatory requirements, or business timing.

3.7. Decision Process

The decision process can be summarized as follows:

1. Identify the interface and its source and target systems.
2. Document the workflow, transformations, dependencies, and recovery behavior.
3. Assign OC, BC, and CF scores.
4. Check whether a replacement capability exists.
5. Apply the decision rules.
6. Review the proposed disposition with application and operations owners.
7. Record the decision, rationale, dependencies, and next action.
8. Assign the interface to a migration wave or reassessment backlog.

This process makes the framework reproducible without suggesting that the scores provide mathematical certainty.

4. Enterprise Case Study

4.1. Environment

The framework was applied in a large manufacturing and distribution environment. BizTalk Server 2020 served as a central integration platform supporting 200 interfaces and approximately five million transactions per month.

The connected landscape included:

- IBM AS/400 systems
- SAP S/4HANA
- SaaS applications
- SQL Server databases
- Business intelligence and reporting platforms
- Other enterprise applications and partner systems

The BizTalk environment has evolved over more than ten years. Some interfaces were straightforward data

exchanges, while others contained complex orchestration, custom processing, or dependencies that were not fully visible from the BizTalk configuration alone. The target architecture used Azure Integration Services, including Logic Apps, Azure Functions, Service Bus, Azure SQL, and Azure monitoring capabilities as appropriate for each interface.

4.2. Portfolio Classification

The 200 interfaces were classified as follows.

Migration disposition	Number of interfaces	Percent age
Lift-and-shift	25	12.5%
Re-architect	86	43.0%
Replace	56	28.0%
Retain/reassess	33	16.5%
Total	200	100%

The distribution is significant for two reasons.

First, 25 interfaces, or 12.5 percent of the portfolio, were classified for lift-and-shift. This indicates that a direct migration was not considered the default choice for the portfolio.

Second, 86 interfaces, or 43 percent, were classified for re-architecture. This reflects the presence of complex workflows, business-critical processing, legacy dependencies, or operational requirements that could not be addressed adequately through a direct conversion.

The 56 replacement interfaces represented 28 percent of the portfolio. These interfaces were candidates for removal or substitution through another application capability, native integration option, or revised business process. The remaining 33 interfaces were retained or placed under reassessment. Retention was treated as a managed decision rather than as an indefinite postponement.

4.3. Migration Execution

All 200 interfaces were migrated through 16 waves over one year and eight months.

The wave approach allowed the migration team to establish common patterns and improve the target operating model over time. It also reduced the risk associated with moving the entire integration portfolio in one cutover.

The migration was not treated as a single replacement event. BizTalk and Azure capabilities had to be considered together during planning and execution. Interface ownership, testing, deployment, monitoring, recovery, and downstream coordination were part of the migration work.

The available case evidence confirms the number of interfaces, the migration waves, and the overall duration. It does not provide a complete set of measured values for

migration cost, individual interface effort, production failure rate, performance change, or recovery-time improvement. Those metrics are therefore not presented as quantitative outcomes in this paper.

5. Analysis of the Framework Application

5.1. Lift-and-Shift

Only 25 of the 200 interfaces were classified for lift-and-shift.

Based on the results, the portfolio contained relatively few interfaces for which technical simplicity, business risk, and Azure suitability aligned sufficiently to support a mostly direct migration.

A direct migration can appear attractive because it reduces functional change. However, the approach becomes less attractive when the existing interface contains custom components, undocumented dependencies, complex recovery behavior, or a strong dependency on a legacy system.

The classification, therefore, avoided treating lift-and-shift as the automatic starting point. It was selected when the existing behavior could be understood and reproduced without carrying unacceptable limitations into the target architecture.

5.2. Re-Architecture

Re-architecture was the largest category, with 86 interfaces.

This result is consistent with the characteristics of a mature middleware portfolio. Over time, interfaces often accumulate responsibilities that were not part of their original purpose. A single orchestration may extract data, apply business rules, call multiple systems, perform transformations, handle exceptions, and coordinate recovery.

Moving such a workflow directly into a cloud workflow engine may reproduce the original complexity without improving its architecture.

The re-architecture decision provided an opportunity to separate responsibilities. For example:

- Logic Apps could coordinate workflow steps.
- Azure Functions could handle specialized processing.
- Service Bus could provide buffering and asynchronous delivery.
- Azure SQL could provide a durable business-level state.
- API Management could govern API exposure.
- Azure Monitor could support operational visibility.

5.3. Replacement

Replacement accounted for 56 interfaces, or 28 percent of the portfolio.

This category demonstrates that modernization is not always a matter of reproducing existing middleware behavior. An interface may exist because of a limitation or requirement that has since changed.

Replacement may be appropriate when:

- A SaaS application provides a native integration capability.
- A modern API replaces a legacy file exchange.
- A business process is moved into the target application.
- An old interface is no longer required.
- Duplicate integration functionality exists elsewhere.

The replacement decision can reduce both migration effort and long-term operational responsibility. It also requires business and application-owner confirmation because removing an interface may affect processes outside the middleware team's direct control.

5.4. Retain and Reassess

Thirty-three interfaces were retained or reassessed.

Some interfaces may have high technical complexity and low cloud-native fitness. Others may depend on applications that are themselves scheduled for replacement. However, retention creates a governance risk. Without an owner, rationale, and review date, retained interfaces can remain indefinitely on the legacy platform.

The framework treats retention as a temporary or conditional disposition. A retained interface should have:

- A documented reason
- A responsible owner
- A known dependency or constraint
- A reassessment condition
- A target review date

5.5. What the Classification Does and Does Not Prove

The classification provides evidence that the framework was applied to a complete portfolio of 200 interfaces. It also shows that the migration program used more than one migration strategy. The classification does not prove that one disposition was faster, cheaper, or more reliable than another. The available evidence does not include complete effort, cost, performance, or incident data for each category. The contribution of the case study is therefore primarily architectural and process-oriented. It demonstrates how the framework was used to organize a real migration portfolio and how the classification supported different migration paths.

6. Architectural Patterns

6.1. Hybrid On-Premises-to-Cloud Connectivity

A significant portion of the portfolio involved systems that could not be modernized at the same time as the integration layer. Approximately 40 percent of the interfaces involved IBM AS/400 or SAP-related connectivity.

The AS/400 environment could expose data through DB2, IBM MQ, ODBC/JDBC, or file-based mechanisms. SAP S/4HANA introduced its own APIs and integration mechanisms. These differences made it difficult to connect every source directly to every Azure workflow.

The architecture used a separation between source extraction and cloud processing:

Source system → Azure SQL staging → Azure integration layer → target system

The staging layer acted as a boundary between systems with different connectivity, scheduling, and performance characteristics. The source extraction process could run according to the constraints of the source platform. Azure-based processing could then validate, transform, route, and deliver the data independently.

Depending on the interface, the Azure integration layer could include:

- Azure Logic Apps
- Azure Functions
- Azure Service Bus
- API Management
- Azure monitoring services

The pattern was useful for high-volume and burst-oriented workloads. Pricing-related processing, for example, could generate a large number of records during a scheduled business event. Separating extraction from downstream delivery reduced the requirement for both sides of the process to be available at exactly the same time. The design also reduced the effect of downstream failures. If the target system was temporarily unavailable, the extracted records could remain in staging while delivery was investigated or restarted.

6.2 Staging Database Isolation

Traditional BizTalk environments provide platform-level capabilities for tracking, suspended instances, message context, and recovery. When a workflow is decomposed into Logic Apps, Functions, and Service Bus, equivalent business-level state may need to be designed explicitly.

A message may be retried or moved to a dead-letter queue, but those mechanisms do not always provide the complete business record needed for reconciliation. Operations may still need to know:

- Which source record was received
- Which processing steps were completed
- Whether a record was delivered
- Why a record failed
- Whether the replay is safe
- Whether duplicate processing occurred

The staging database isolation pattern addressed this need. A simplified flow is:

Source → staging record → validation and enrichment → Service Bus → transformation and delivery → target → completed status

The staging record represents the processing lifecycle. It can contain a business identifier, processing status, timestamps, error information, and other fields required for reconciliation. A successful record can be marked as completed. A failed record can remain available with an error status and supporting information. This allows selected records to be investigated or replayed without repeating the entire source extraction.

The pattern also supports duplicate control. A business identifier and processing status can be checked before a record is delivered again.

6.3. Qualitative Evaluation

The following comparison summarizes the architectural trade-off.

Evaluation factor	Direct source-to-cloud flow	Staging-based flow
Source and target coupling	Higher	Reduced
Buffering for volume spikes	Limited unless separately designed	Explicit staging buffer
Source extraction independence	Lower	Higher
Selected-record replay	Depends on implementation	Supported through staging status
Reconciliation visibility	May depend on platform logs	Business-level status can be retained
Target outage handling	May block or trigger retries	Records can remain staged
Additional infrastructure	Lower	Higher
Data lifecycle responsibility	Less explicit	Requires retention and cleanup design

The staging pattern introduces additional database infrastructure, data lifecycle responsibilities, and operational work. It should therefore not be applied automatically to every interface.

The pattern is most useful when the business requires recoverability, reconciliation, buffering, or controlled replay. For a simple, low-risk interface, the additional staging layer may not justify its cost and complexity.

7. Practical Findings

7.1. Hybrid Operations Require a New Support Model

A hybrid architecture can appear simple on a diagram, but its support model is more complicated.

During migration, support teams may need to work with:

- BizTalk administration tools
- Azure workflow history
- Service Bus queues and topics
- Azure Functions logs
- Azure SQL data
- Application telemetry
- Downstream application logs

The failure models are also different. A BizTalk suspended instance is not operationally identical to a failed Logic Apps action or a Service Bus dead-letter message. The investigation process, available diagnostic information, retry behavior, and recovery procedure may vary.

Migration planning, therefore, needs to address monitoring ownership, alert routing, incident procedures, replay, access, training, and escalation.

7.2. Lift-and-Shift Still Requires Operational Redesign

A lift-and-shift decision limits functional redesign, but it does not eliminate operational work. Azure workflows have their own retry behavior, trigger configuration, run history, action-level failures, deployment process, and security requirements. These characteristics need to be understood before production support begins. Certificates, secrets, connection settings, permissions, and environment-specific configuration also need to be managed according to the Azure operating model. The practical meaning of lift-and-shift is therefore limited functional change, not zero change.

7.3. Retention Requires Governance

Retention is often easy to approve and difficult to manage.

An interface may remain on BizTalk because its dependencies are difficult to resolve or because its business value does not justify immediate migration. Without governance, it may remain indefinitely and continue to consume support effort. A retained interface should have a documented reason, an owner, and a reassessment condition. Changes in connected applications, business processes, or Azure capabilities may make a later migration more appropriate.

8. Discussion

8.1. Difference from Existing Migration Frameworks

The four IMDF dispositions overlap with established cloud migration strategies. Lift-and-shift is related to rehosting or replatforming. Re-architect is related to

refactoring. Replace corresponds to repurchasing or replacement. Retaining and reassessing corresponds to retaining a workload temporarily.

IMDF applies these dispositions to integration interfaces using three dimensions that are particularly relevant to middleware:

- Orchestration complexity
- Business criticality
- Cloud-native fitness

The framework also connects the disposition to migration sequencing and operational architecture. This is important because an interface cannot be evaluated only by its implementation. Its source system, target system, consumers, business purpose, and recovery requirements also affect the decision.

8.2. Why the Three Dimensions Are Separated

Combining all factors into one general migration score would hide important trade-offs.

For example:

- High complexity does not always mean high business criticality.
- High business criticality does not always mean poor cloud fitness.
- Low complexity does not guarantee a safe migration.
- Strong cloud fitness does not remove the need for recovery and monitoring design.

Keeping the dimensions separate allows the architecture review to explain why an interface received a particular disposition.

8.3. Interpretation of the Case Results

The portfolio classification provides several observations.

The relatively small lift-and-shift category indicates that direct migration was not sufficient for most interfaces. The large re-architecture category indicates that many interfaces required changes beyond technology conversion. The replacement category shows that a significant part of modernization involved removing or substituting existing integration functionality. The retain/reassess category shows that some interfaces required a controlled exception to the immediate migration plan.

These findings support the use of an interface-level decision process. A single migration strategy would not have reflected the actual composition of the portfolio.

8.4. Limits of the Analysis

The case study confirms that all 200 interfaces were migrated through 16 waves over one year and eight months.

It does not provide a complete quantitative comparison of the four dispositions.

The following measures were not consistently available:

- Effort per interface
- Cost per migration category
- Deployment duration per interface
- Production failure rate
- Performance change
- Recovery-time improvement
- Operational incident reduction
- Long-term operating cost

Hence, the paper does not claim that IMDF produced a specific percentage improvement in cost, speed, reliability, or performance.

The absence of these measurements limits the strength of the empirical conclusions. At the same time, the documented portfolio classification and migration execution provide a useful basis for future quantitative study.

9. Limitations and Future Research

The framework was developed and applied in one enterprise environment. The organization's systems, business processes, technical standards, and operating model may differ from those of other organizations.

The scoring model uses three levels for each dimension. The scores provide consistency during architecture review, but they do not represent statistically calibrated probabilities. Different organizations may need different thresholds or additional factors.

The case study also lacks complete outcome measurements for each interface category. Future research could collect:

- Actual migration effort
- Cost and resource consumption
- Defect and incident rates
- Deployment frequency
- Recovery time
- Processing latency
- Message failure and replay rates
- Long-term support cost

A larger study could compare several organizations and determine whether the three IMDF dimensions consistently explain migration decisions and outcomes.

References

- [1] Gregor Hohpe, and Bobby Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*, Addison-Wesley, 2004. [[Google Scholar](#)]

Another area for future work is automated assessment. BizTalk metadata, interface catalogs, dependency information, message volume, operational logs, and application ownership data could be used to create an initial assessment before a detailed architecture review.

The framework could also be extended with additional dimensions, such as security sensitivity, regulatory exposure, data residency, or organizational readiness. Such extensions should be introduced only when they improve decision quality without making the assessment too difficult to use.

10. Conclusion

Migrating a mature BizTalk environment to Azure is not simply a matter of converting orchestrations into Logic Apps workflows. A large integration portfolio contains interfaces with different levels of technical complexity, business importance, cloud suitability, and operational risk.

The Integration Migration Decision Framework described in this paper uses three dimensions—Orchestration Complexity, Business Criticality, and Cloud-Native Fitness—to guide four dispositions: lift-and-shift, re-architect, replace, and retain and reassess.

The framework was applied to 200 interfaces processing approximately five million transactions per month. The portfolio included 25 lift-and-shift interfaces, 86 re-architected interfaces, 56 replacement interfaces, and 33 retained or reassessed interfaces. All interfaces were migrated through 16 waves over one year and eight months.

The case study demonstrates that direct migration was not the default choice. Re-architecture was the largest category, while replacement also represented a substantial part of the portfolio. This distribution supports the need for interface-level assessment rather than one migration strategy for an entire BizTalk application.

The work also shows that modernization extends beyond workflow conversion. Hybrid connectivity, durable staging, replay, monitoring, recovery, and ownership must be considered as part of the target architecture.

The main contribution of IMDF is therefore a practical decision structure for applying established migration strategies to enterprise integration interfaces. IMDF makes migration decisions more explicit, comparable, and easier to discuss across architecture, application, business, and operations teams.

- [2] Dirk Krafzig, Karl Banke, and Dirk Slama, *Enterprise SOA: Service-Oriented Architecture Best Practices*, Prentice Hall, 2004. [[Google Scholar](#)] [[Publisher Link](#)]
- [3] Gartner, Gartner's Cloud Migration Framework, Gartner Research, 2010. [Online]. Available: <https://www.gartner.com/en>
- [4] Microsoft, Migration Approaches for BizTalk Server to Azure Logic Apps Standard, Microsoft Documentation, 2022. [Online]. Available: <https://docs.azure.cn/en-us/logic-apps/biztalk-server-migration-approaches>
- [5] Microsoft, Azure Logic Apps Documentation, Microsoft Documentation, 2023. [Online]. Available: <https://learn.microsoft.com/en-us/azure/logic-apps/>
- [6] Microsoft, Microsoft BizTalk Server Product Lifecycle Update, Microsoft Documentation, 2023. [Online]. Available: <https://techcommunity.microsoft.com/blog/integrationsonazureblog/microsoft-biztalk-server-product-lifecycle-update/4478559>
- [7] Software Modernization Services, BizTalk End of Life: Support Dates and Migration Options, Software Modernization Services Documentation, 2026. [Online]. Available: <https://softwaremodernizationservices.com/insights/biztalk-end-of-life/>