LTIMindtree

Accelerate to Mule 4: Considerations for Migration while upgrading from MuleSoft 3

Table of Contents

1. Introduction

Integration has become an important ingredient in the success of any organization, as it increases the sprawl of applications, infrastructure, and the partner ecosystem. Businesses with the right integration strategies have raised the bar with lower operational costs, faster project delivery, and smart revenue streams. They are quickly leaving more traditional players far behind. An integration strategy allows organizations to connect in the right way, overcome obstructions, and drive tangible business value. By being able to quickly connect new information and operationalizing it across the entire enterprise, organizations can increase productivity, ensure tighter security, and be able to help the organization stay competitive in their industry.

MuleSoft's Anypoint Platform™ is a leading application network platform. It allows organizations to create composite applications that connect apps, data, and devices through API-led connectivity to form a flexible application network. Anypoint Platform is a single unified solution for iPaaS and full life cycle API management, across both on-premises and in the cloud.

MuleSoft provides a powerful technology platform that provides enterprises with robust API implementation solutions and strategies. Between its last major release 3.0 in 2010 and its current state, MuleSoft has evolved a lot, from being just an Integration/API platform to something which can aid in assisting legacy modernization, implementing secure SaaS integrations, and providing full API life cycle management. MuleSoft now allows organizations to implement and allow its IT Department to integrate, connect, and build its enterprise solutions in innovative new ways. The next-gen Mule 4 platform offers a broad range of new and improved features intended to enhance the capabilities of the platform along with developer experience. In this whitepaper, we try to address the benefits, challenges, and best practices for organizations looking to migrate from Mule 3.0 version to Mule 4.x.

Glossary

Abbreviation	Definition
API	Application Program Interface
C4E	Center for Enablement
CoE	Center for Excellence
DMZ	Demilitarized Zone
HA	High Availability
IT	Information Technology
LOB	Line Of Business
PCI	Payment Card Industry
RAML	RESTful API Modeling Language
SDK	Software Development Kit
SOA	Service Oriented Architecture

2. Why Do You Need to Migrate to Mule 4?

MuleSoft, in its journey from 3.0 to 4.x, tries to bring customers, business, and developers together, and help them innovate possibilities. With the launch of Mule 4, MuleSoft is offering several features to make integration easy, because of which many enterprises are already adopting Mule 4. Let's look at some of the features that Mule 4 offers and the differences between the Mule 3 and Mule 4 releases.

Mule 4 has improved on the following fronts, compared to Mule 3:


Exception handling


Self-tuning


Frictionless upgrades


Better application configurability


Repeatable streaming


New connectors


Seamless access to data


Triggers


Enhanced enrichers

2.1 Exception Handling

Mule is a high-end middleware tool for programming – which supports flows, business logic, data types, etc. A common gripe with the Mule 3 platform was that it had fallen short in managing Exception Handling. Mule 4 directly handles the exception with a seamless configurable error handling mechanism.

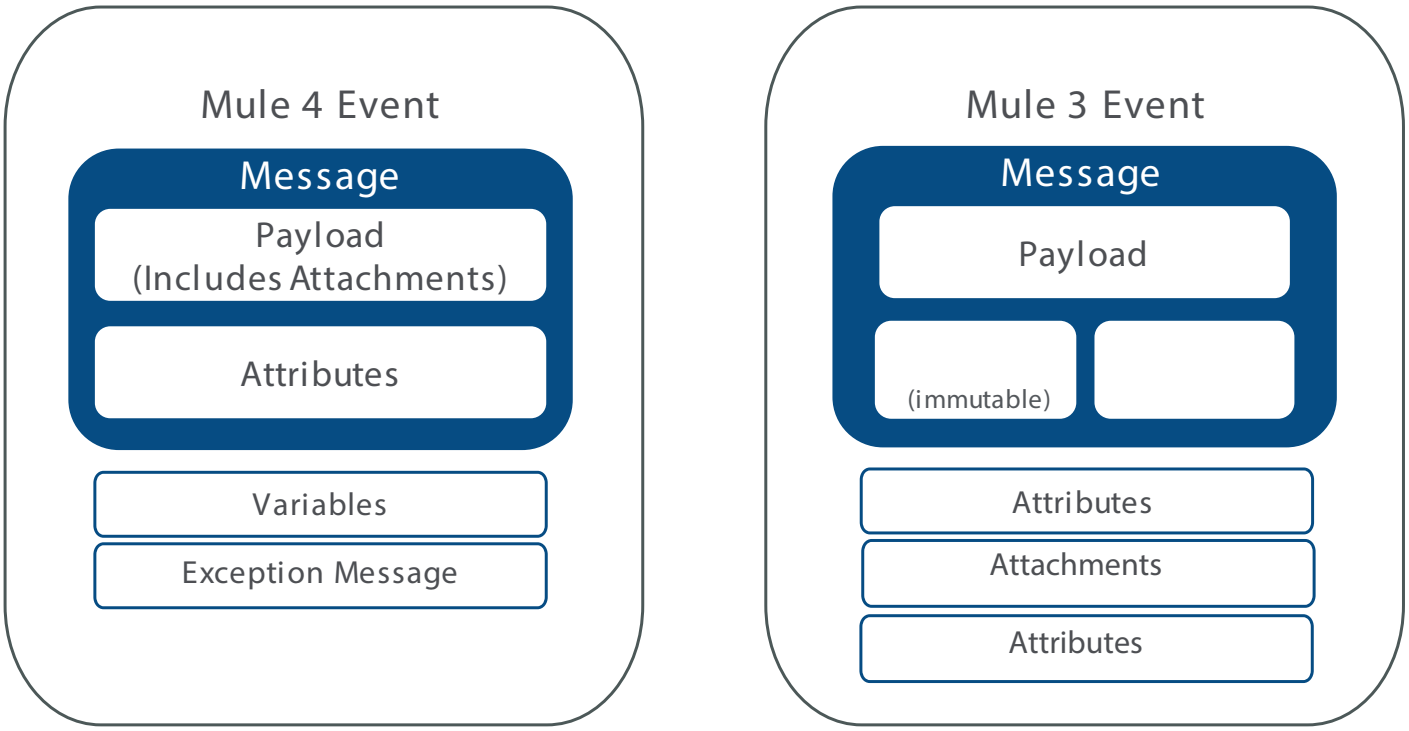
In points:

- It projects the Java exception into Mule 4 error objects
- Custom Error functionality can be used to differentiate errors in the application and various points
- It allows mechanism to catch groups of error/exception objects together

2.2 Simplified Event Processing and Messaging

Mule 4 brings a more compact event-processing model by optimizing unwanted hierarchies and workflows. Inbound and outbound properties are merged as attributes in Mule 4 event architecture's message section. Unlike Mule 3, the inbound and outbound properties are combined under one section. They are used to carry the payload's metadata information such as - any file content, query parameters, inbound properties, outbound properties, etc.

Message handling in Mules 3 vs Mule 4 is different due to simplification of complicated message structures. Earlier a simple transformation required a creation of java objects, etc. whereas now they happen by default.



Event Models

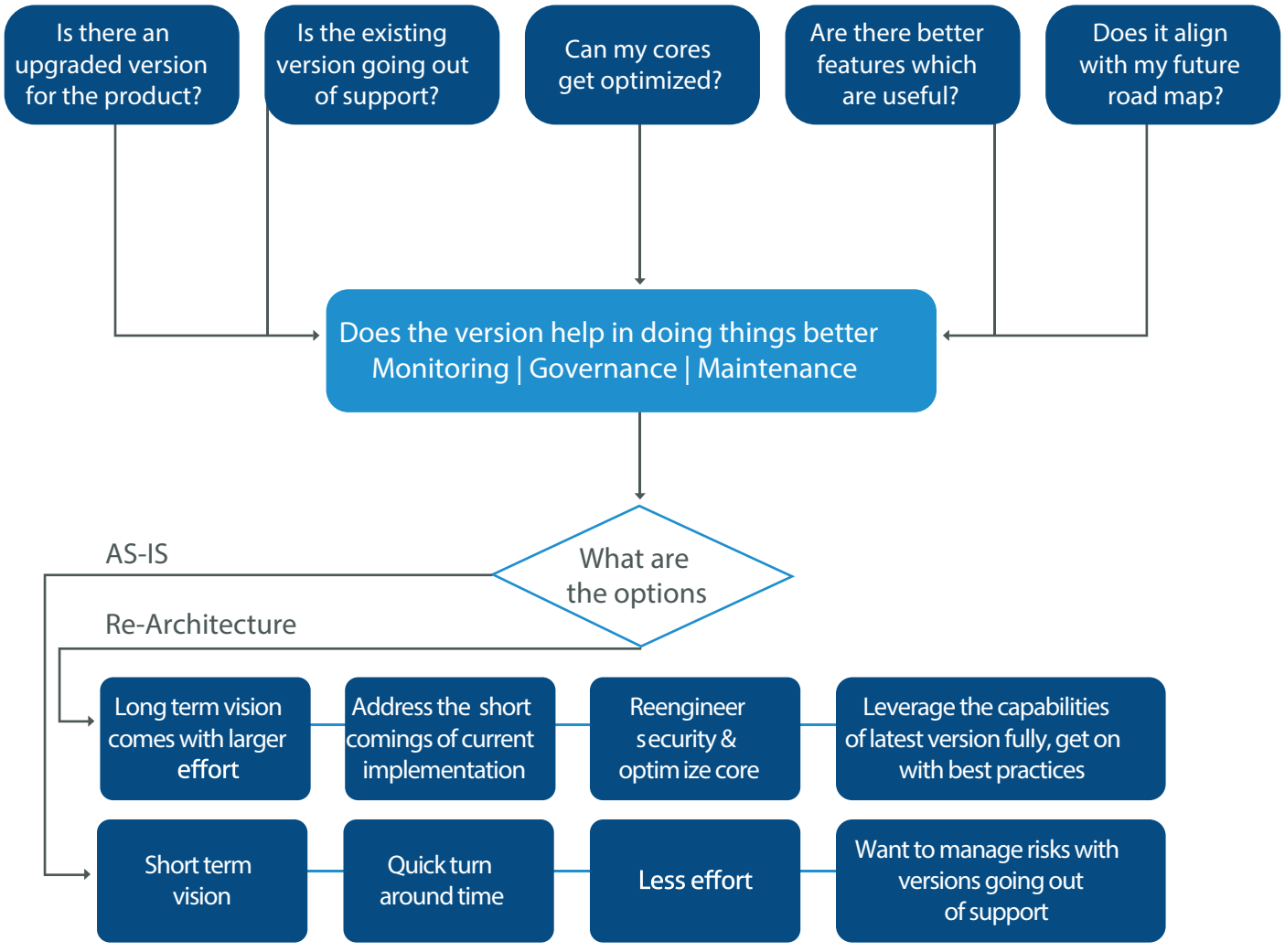
2.3 Self-Tuning Capabilities

While the ease of implementation, look and feel, features and simplicity are important considerations, performance and scalability are of paramount importance – regardless of the use case. Mule 4 brings in a new approach altogether with the adoption of a reactive and non-blocking character of threads (meaning threads will no longer wait for responses; instead they will process parallelly), which makes it scalable. In addition to ensuring that the developers and support teams do not have to do performance tuning, the platform has been blessed with self-tuning capabilities. Implementers will no longer need to worry about thread pools, threading profiles, and processing strategies in order to achieve a high performing implementation. Mule 4 can now perform self-analyses and auto-scale based on runtime condition needs.

3. When to Migrate from Mule 3 to 4?

Migration of an already working implementation demands a lot of support from the senior management and will obviously need an investment, as suggested earlier. However, when to migrate is a very vital question.

Here is a flow chart for simplifying the decision-making:



Decision Process

Before initiating the migration from Mule 3 to Mule 4, it is important to check if the organization is ready for the upcoming changes. Organizations can consider below points in their decision-making process:

- How comfortable are the IT (development and Supports) teams in using Mule 4?
- Do they understand the components that have to be changed in Mule 4 in comparison to Mule 3? (understand the difference between Mule 3 vs Mule 4)
- What is the organization strategy to support all currently running MuleSoft versions along with Mule 4?
- Since Mule 3 and its variants' supports are expiring in 2021, would they extend?
- Do you want to upgrade all applications to Mule 4?

4. Methodologies for Mule 3 to Mule 4 Migration

As described earlier, Mule 4 offers several new features that can make integration easy and cost-effective. However, just like any other migration, migrating from Mule 3 to Mule 4 has its own challenges. To assist businesses in secured migration, we have detailed a few measures that you need to take prior to the migration process:

4.1 Rationalization of Existing Mule 3 Implementation

For most organizations, the existing Mule 3 implementation would have matured over a period of time and could have been customized by multiple implementers. Thus, there is often a need to rationalize the existing integrations in order to support a smooth migration. Integration rationalization can help organizations mature their integration landscape, and improve LOB management capabilities and delivery of mission-critical business services. However, it requires buy-in from stakeholders across the enterprise - including senior leaders, technical teams, LOBs, enterprise teams, etc.

The process for rationalization is shown below:

- Group based on functionality
- Future state
- Device strategy

- Finalize scope
- Identify requirements
- Establish governance parameters

- Bring in scoring methodology
- Review interface scoring

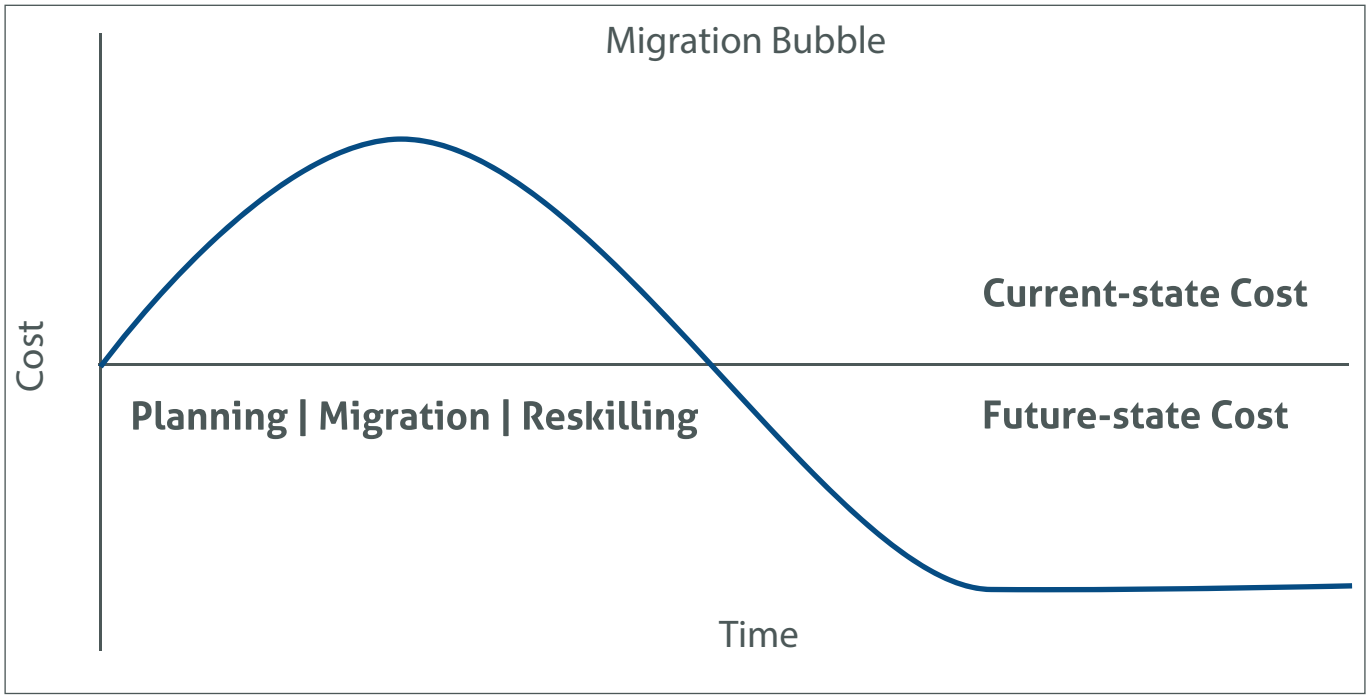
- Finalize inventory
- Categorize the inventory
- Isolate end points

- Confirm cost and distribution
- Compare Value against cost
- Suggest new ways to bring cost distribution

- Review value & Technical Fit
- Identify dependencies
- Identify duplication and versions

Rationalization Process

Organizations generally face a 'migration bubble' which is an increase in IT costs due to the migration. However, migration brings long term value in the form of increased worker productivity, greater scalability and agility, and operational resilience, which establishes a new cost baseline and results in cost savings in the long term.



Migration Bubble

4.2 Choosing the Right Migration Options

While Center for Enablement (C4E) is the driving force for many organizations, independent LOB's may have specific implementations with different architecture flavors, leading to multiple patterns and localized frameworks. During the migration, customers should look at addressing these concerns in addition to looking at deployment architecture, HA, DevOps, etc.

First things first, evaluate your business needs and road maps and consider the options of As–is migration vs. re-architecture. The recommended approach though is mostly to re-architecture (based on factors such as cost, time, benefits, roadmap, etc.).

As-Is Migration: this strategy involves rebuilding the application as-is on the new platform. Here, implementation is done with none or very little modification in logic. While it is a simplistic approach to start with As-Is migration of APIs or implementation optimization, this approach ensures to be quick and easy, as outcomes are achieved with minimal application disruption and effort. However, the platform may not deliver the latest features and benefits, which can lead to core optimization. Note that organizations may leverage the MuleSoft-provided migration utility for the same too, which can help to a certain extent.

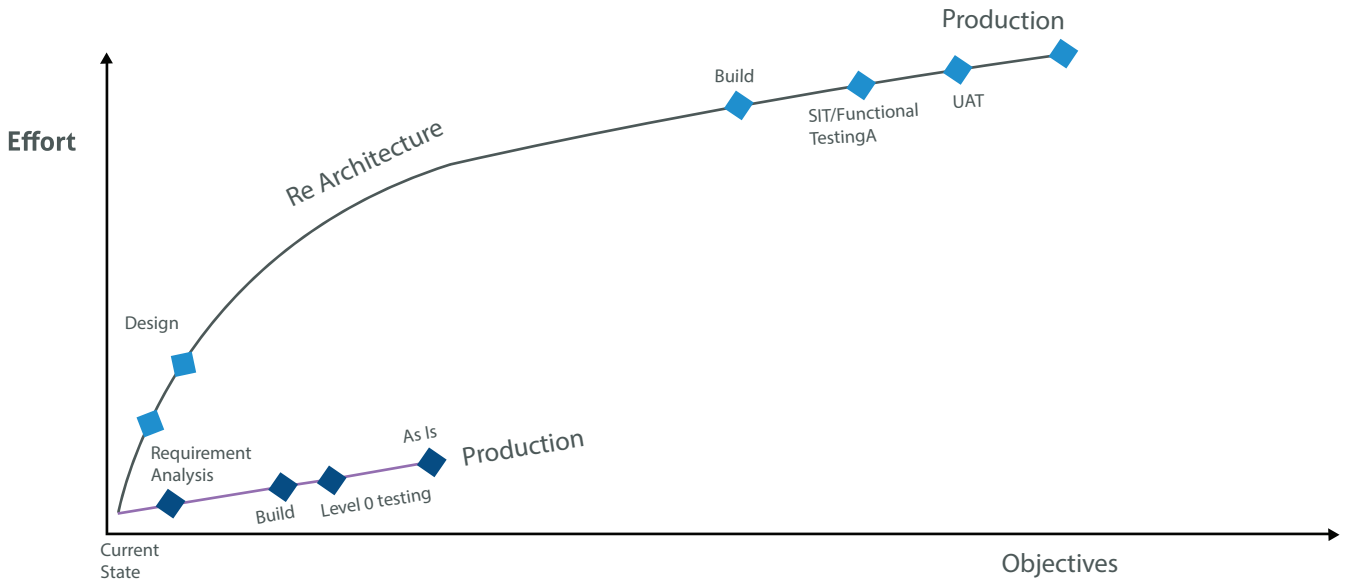
Re-Architecture: this strategy entails making major changes in the integration implementation. This is a complex approach in comparison to As-Is migration. We need to ensure that there is no impact on external behavior of the middleware layer while implementing changes.

For example, let us consider that different LOB's have implemented customer information API's. As part of re-architecting, if the organization decides to move to API-led connectivity with experience layers for each LOB along with few best practices and common frameworks embedded in the implementation, it may cause a few blips in the experience. Hence, testing is mandatory and has to be conducted end-to-end.

The organization has to choose between As-Is or re-architecture, depending on the following factors.

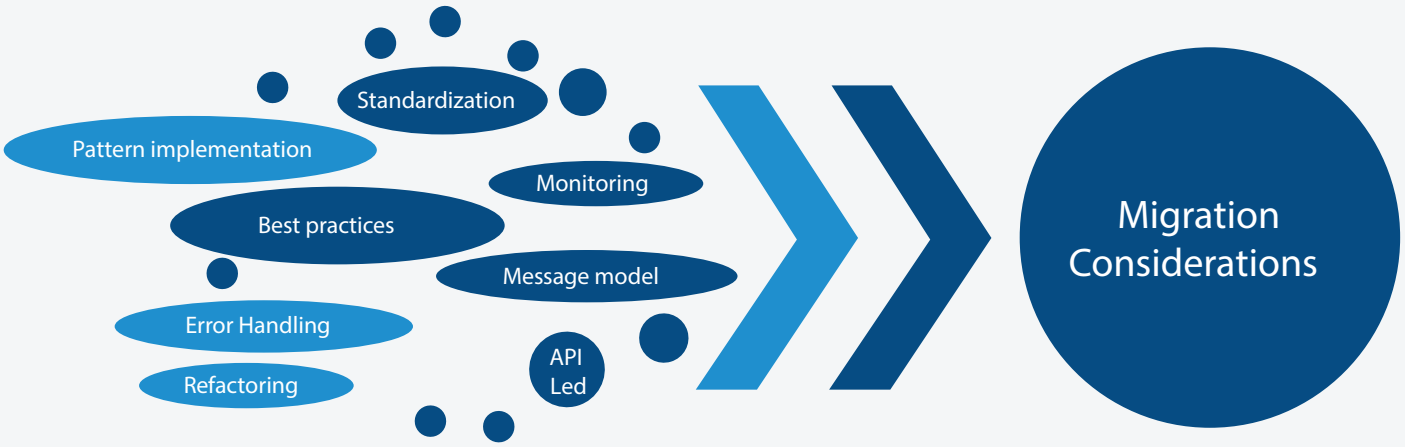
Options	As-Is	Re-Architecture
Cost	Low	High
Time	Low	High
Version Compliance	Yes	No
Platform Benefits	No	Yes
Core Optimization	No	Yes
Long-term benefits	No	Yes
Short-term plan	Yes	No

Based on our experience working on Mule migration projects and the objectives achieved, the comparison of effort vs. objectives is depicted below. Note that the graph could change based on multiple parameters, foresight of the organization, number of applications to be migrated, maturity of the implementation, etc. The below comparison is for organizations having a significant investment in MuleSoft over time with multiple implementation approaches (API-led, Customizations, Central/Federated, etc.)



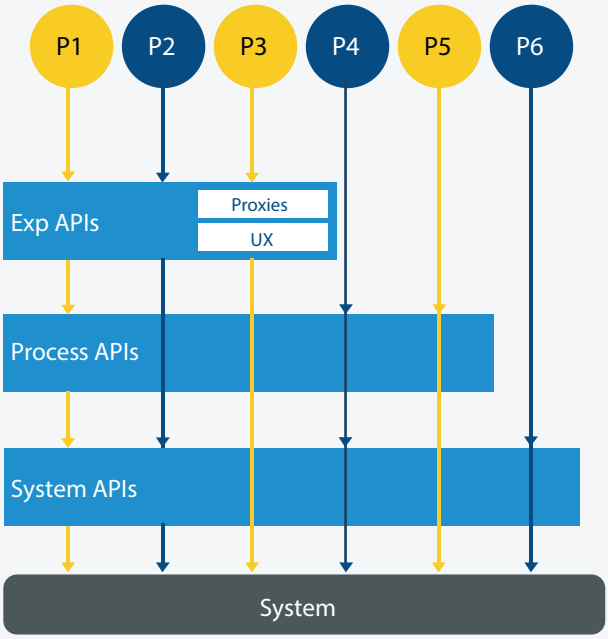
Effort vs Objective Comparison

Modern architecture sets a platform to support the future road map and should therefore be a key consideration while migrating. A well-planned and executed integration migration will result in a modern platform that meets business needs for agility along with objectives and produces cost savings while aligning to the product roadmap.



Migration Consideration

One of the considerations as part of the migration is API-led connectivity and the identification of patterns for implementation. We believe there are generally six patterns in a synchronous world.



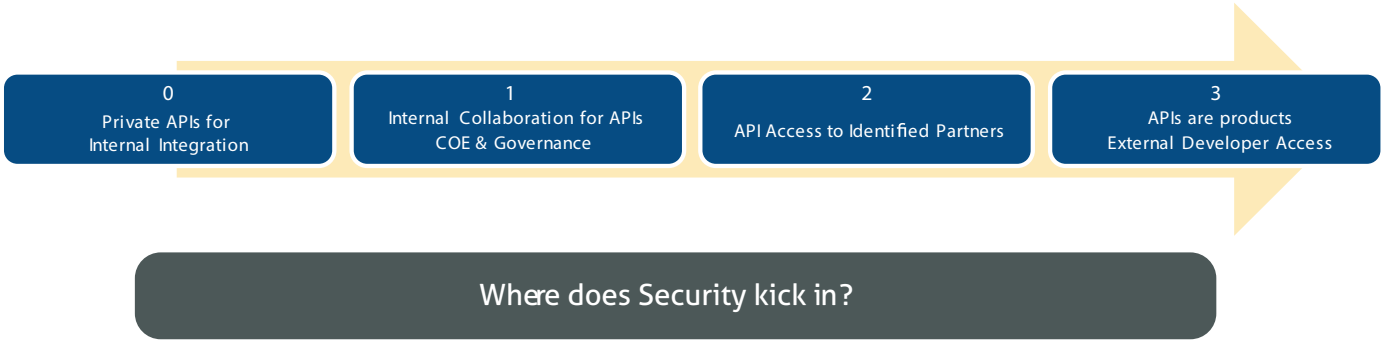
- Patterns P1, P2 align to the API Led Approach
- Pattern P3 is proxy an external API
  - Connecting using native protocols in P3 is an anti pattern
- Pattern P4 is valid unless process needs tight security
- Pattern P5 is not recommended unless
  - Requirment needs a complex transaxtional flow
  - Exception is obtained from Governance team
- Pattern P6 is valid for internal consumption unless systems need a tight security

API Patterns

As part of refactoring and re-architecting, it would be recommended to group the integrations with similar functionalities and fit them in one of the patterns above, thus making the implementation more template-driven and structured.

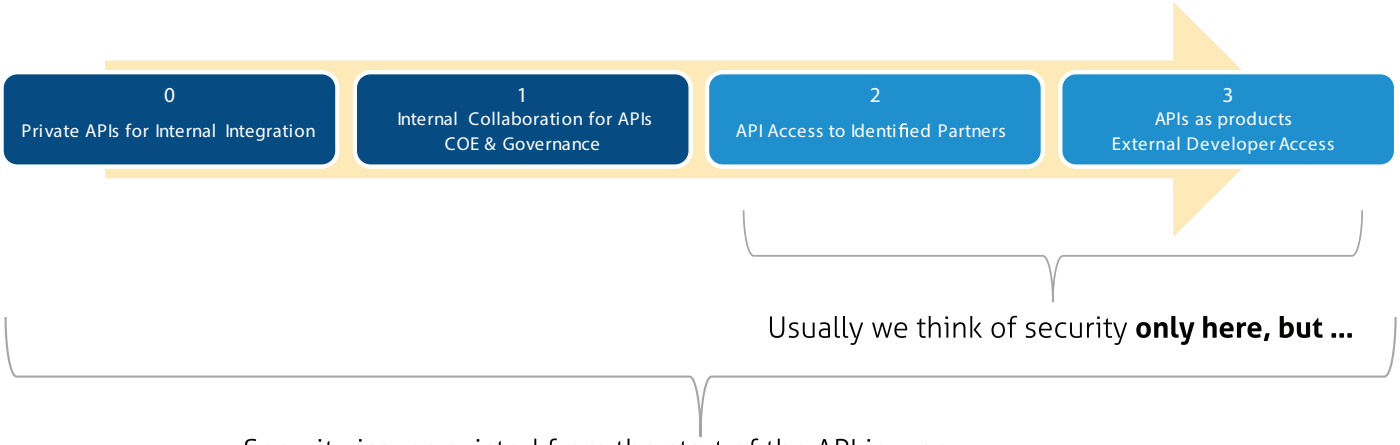
4.3 Security

Security is crucial in any type of implementation. When an organization decides to migrate to a new platform, it provides an opportunity to relook at the implementation and ensure security for applications, data, and infrastructure. The first step is to understand where to start.



Security Thought

The answer is that it kicks in right from the inception.



Security issues existed from the start of the API journey.

- Internal users → Internal teams → External partners → Public access
- **Bad guys may be present anywhere (not intentional always)**
- **Accessible infrastructure, not only API access**

Security Understanding

There are multiple facets of API security and it is the responsibility of the migration implementers to understand the current implementation, identify the gaps, and address them across:

5. Conducting the Mule 3 to Mule 4 Migration Process

While we elaborated on methodologies of migration along with when to take the call, the part which details how is extremely important.

5.1 Re-Architecture

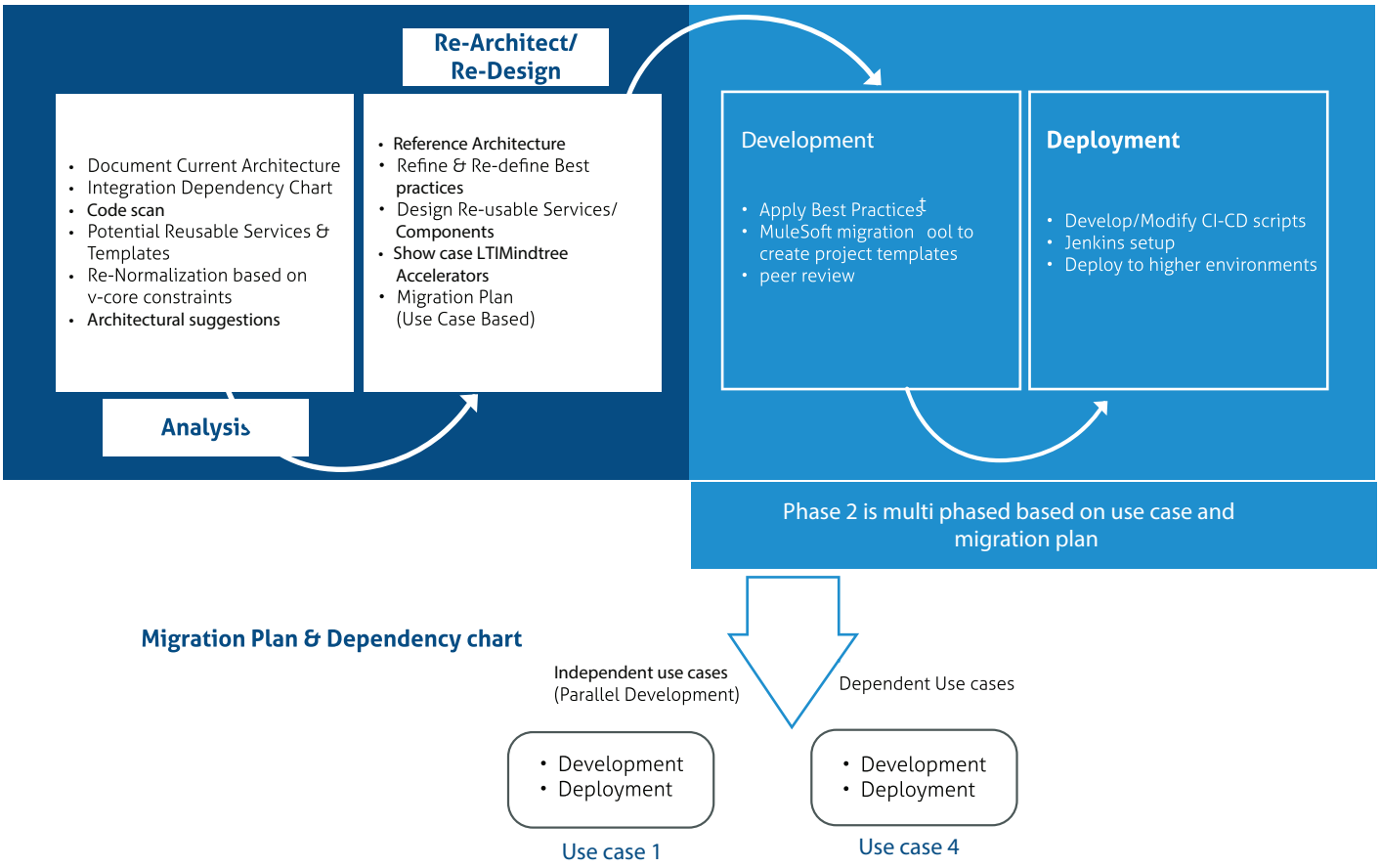
Migration is a very complex process, specifically when the organization chooses the re-architecture path. The organization not only has to rewrite the implementation, but also needs to ensure that this opportunity that requires significant investment is not wasted.

Re-architecture provides opportunities to relook at implementation from various fronts and questions future viability. A few aspects to consider are:

Category	Description
Platform	Is the API hosted on-premise or on cloud? Can it be hosted on cloud? If not, why?
API-led	Is the API/integration designed as per API-led thought process? Can they be relooked again, as sometimes organizations would have over-engineered the implementation and now could relook and optimize?
RAML	Is there a refactoring of RAML required? Do we increase readability, implements types?
Custom	Are there custom Java implementations for file handling, parsing, reading messages from queues, backend applications, etc.? Are there Groovy Scripts? Are there any customer components used?
Naming Conventions	Are there naming standards and are the implementations following them? Did they evolve over time and need standardization?
Logging & Exception handing	Are the standards and formats followed as per guidelines? Is it API kit default and do you want to relook?
End Connectors	Are there any system-specific connectors? Are there any object stores used?
Reusability	Is there any scope for reusability? Are there any common processes across LOBs?
Patterns	Have you identified the patterns applicable for your organization? Are the implementations following those patterns?
Auto Discovery	Is auto-discovery implemented?
General Optimizations	General conditions, hard coding, certificates, removal of custom implementations with latest platform capabilities, optimize variable usage, conversion optimization
Compliance	Does the implementation need to be compliant with PCI etc.?
Security	Is the security implemented across layers as per standards? Can someone hack, if they pass through the DMZ?

We suggest organizations to take a two-step approach before decommissioning the existing applications

- Foundation (Covered as Analysis and Re-Architecture in the Figure Implementation Process)
- Productionized (Covered as Development and Deployment in the Figure Implementation Process)



Implementation Process

5.2 As-Is Migration

Based on your needs and time available, if you need to migrate As-Is from Mule 3 to Mule 4, it requires all the modules to be added in the Anypoint Studio palette. The sequence to follow when migrating is as follows:

- Migrate Patterns
  - Migrate Message Properties
  - Migration Re-connection Strategies
- **Migrate Secure Property Placeholders**
- **Migrate Watermarks**
- Migration Core Components
  - Migrate Batch Components
  - Migrate Choice Router
  - Migrate Exception Strategies to Error Handlers
  - Migrate Enrichers to Target Parameters
  - Migrate Filters
  - Migrate the 'For Each' Component
  - Migrate Poll Component
  - Migrate Scatter-Gather Router
  - Migrate Transformers
- Migrate Connectors
  - Migrate Anypoint Enterprise Security (AES) Module
  - Migrate to the AMQP Connector
  - Migrate Database Connector
  - Migrate Email Connector
  - Migrate File Connector
  - Migrate FTP and SFTP Connector
  - Migrate HTTP Connector
  - Migrate JMS Connector
  - Migrate Object Store Connector
  - Migrate Scripting Module
  - Migrate Spring Module
  - Migrate Validate Module
  - Migrate VM Module
  - Migrate Web Service Consumer Module
  - Migrate XML Module

- **MuleSoft Connector Migration from 3 to 4**
  In Mule 4 the architecture for MuleSoft connector has changed completely. Connectors are developed in Mule 3 using Dev Kit, whereas in Mule 4 they are built using Mule 4 SDK. Each processor built, is added as a different connector (Pallet) in Anypoint Studio.

To migrate the connector to Mule 4, you can use the conversion tool to:

- **In pom file Enable Connector as Mule 4 extension**
- **Modify package element: <packaging>mule-extension</packaging>**
- **Modify parent element**

  <parent>
  <groupId>org.mule.extensions</groupId>
  <artifactId>mule-modules-parent</artifactId>
  <version>1.0.0</version>
  </parent>
- **Modify the folder structure**
- **The typical folder structure for Mule 4 is:**

  <Module>/api
  <Module>/internal
- Update annotations and Params Classes
- **Delete/Add/Update Classes based on Mule 4 SDK (Please refer Java Docs)**

Additionally, the below parameters need to be considered for Mule 4 migration.

- **Auto Discovery**
- DataWeave Header Content
- Standardize RAML Template

5.3 Leverage Testing and DevOps framework

Most of the Mule 3.X implementation would have DevOps and Testing frameworks in place. Few of the good things that can be reused in a MuleSoft 3.X to 4.X migration are:

- **DevOps frameworks with minor enhancements**
- **Testing framework, if built using tools (not Munit)**

DevOps implementation will need changes in few places like the polling directory, project structures, and scripts, while the configuration for the DevOps pipelines may change. Probably this could be a good time to see if there are any changes needed in the overall pipeline and approval process aligning to the organization roadmap.

Automated testing, if developed using Munit, cannot be reused. Only if the test cases are developed using tools such as Postman or SOAPUI projects (assuming that the API contracts/Definitions/Schema have not changed) it can be reused.

6. Conclusion

At a time when the migration from MuleSoft 3 to 4 is imminent, you only have two options. Hence, sitting on the sidelines is no longer an option. Organizations need to ensure that the continuity of the services is essential, costs are lower, and the need for migration is considered as an opportunity towards ensuring long-term benefits. Based on the options available, we have created a model that can provide organizations the means to take the optimized and safest path towards the latest MuleSoft 4 platform.

For more details, contact us.

7. Reference

- https://docs.MuleSoft.com/mule-runtime/4.2/intro-overview
- https://docs.MuleSoft.com/mule-runtime/4.1/migration-cheat-sheet
- https://docs.MuleSoft.com/mule-runtime/4.1/mule-runtime-updates
- https://www.mulesoft.com/press-center/network-graph-machine-learning

8. About the Authors

Srinivas Kanduri has about 19+ years' experience working as Program Architect at LTIMindtree Digital practice, specializing on enterprise application integration. Experience in major integration technologies like MuleSoft, TIBCO and Software AG WebMethods. TOGAF 9.1 Certified enterprise Architect providing Enterprise Digital Transformation consulting, Architecture Consulting, Solution architecture, technology consulting, and leadership for technology groups within organization and IT strategy for client organizations.

Vijay Chakka possesses close to 17+ years of IT industry experience, primarily in the integration space, and engaged in consulting, managing delivery and practice leads. Currently, he heads the 'Mulesoft CoE' within the Enterprise Application Integration service line under LTIMindtree Digital. He has been part of the growth path for multiple organizations across industries such as banking, entertainment, energy, manufacturing etc.

Surendra Thekkatte is a CoE head within Enterprise Application Integration service line under LTIMindtree Digital practice with over 22+ years of experience specializing in Integration and Architecture consulting offerings from LTIMindtree, helping customers defining API Strategy, setting up of API CoE and implementation roadmap for customers transitioning from SOA centric middleware products to Microservices based architecture involving API management platforms.