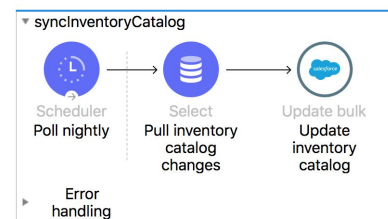
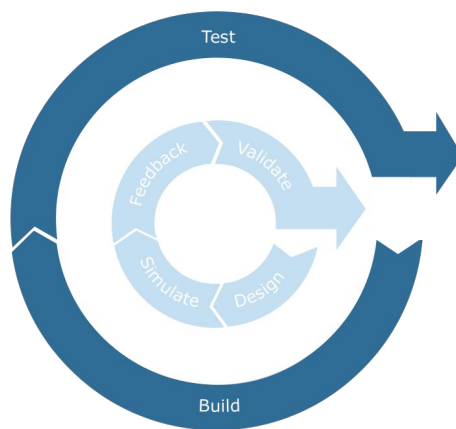
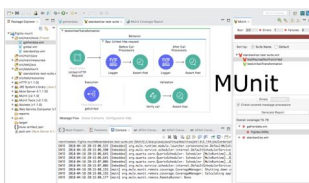


# PART 1: Architecting and Designing Integration Solutions

## Goal



At the end of this part, you should be able to

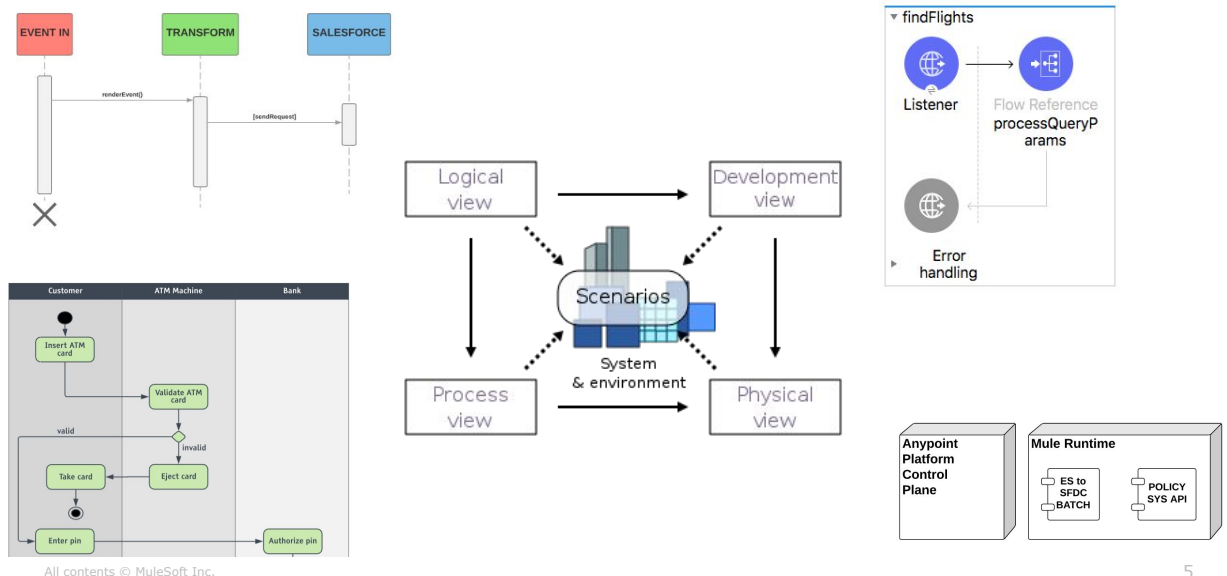


- Architect integration solutions
- Identify Anypoint Platform components and capabilities
- Design integration solutions using Mule applications
- Choose appropriate Mule processing models
- Choose Mule event transformation and routing patterns
- Design testing strategies for Mule applications



# Module 1: Introducing Integration Solutions Architectures






## At the end of this module, you should be able to

- Describe the objectives of enterprise integration solutions
- Summarize how to architect for customer success with Anypoint Platform
- Describe how integration solutions using Anypoint Platform are documented
- Start using an architecture template for the course case study

# Describing the objectives of enterprise integration solutions



## What characterizes integration solutions across an enterprise?



- **Integration solutions** broadly involve linking together different computing systems and software applications physically or functionally, to act as a coordinated whole
- Particular challenges arise when integration efforts are carried out across internal and external **organizational boundaries**
  - Various stakeholders across the enterprise have **multiple**, and **sometimes conflicting, goals**
  - These stakeholders also have different **assumptions, understandings, and language** to describe these goals
  - Enterprise systems often have additional **requirements** for **reliability, availability, and performance**
- Integration solutions within an enterprise must address these enterprise-wide concerns

## The most important objective of an integration solution architecture



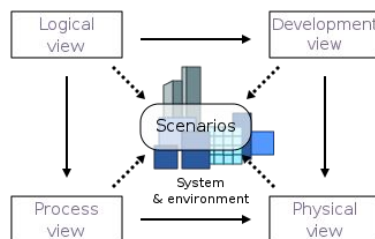
- To address the **in-scope requirements** of **stakeholders** related to the integration scenarios and use cases
  - This requires documenting **different views** to address these **goals** and **concerns**
  - Additional view and documentation may be required to align **stakeholders** from **different business units** or **job roles**

## Introducing the course case study



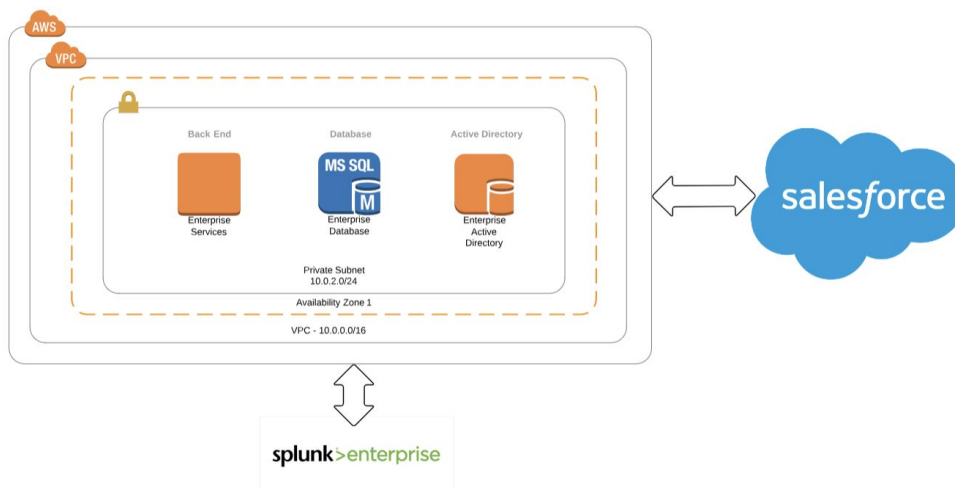
## This class focuses on the design phase

- This class assumes the **analysis phase** steps (scenarios) have already been **completed** and agreed upon
  - This initial **project phase** includes important decisions and agreements, but it is mainly independent of MuleSoft tools or Anypoint Platform
  - The class case studies **already incorporate these decisions**



## Case study

- Included in the course student files



# Identifying stakeholders involved in an integration project

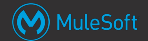


## Examples of stakeholders involved in integration solutions



Stakeholder type	Description and interests
<ul style="list-style-type: none"><li>• Project sponsor</li></ul>	<ul style="list-style-type: none"><li>• Drives the project</li></ul>
<ul style="list-style-type: none"><li>• Architects</li></ul>	<ul style="list-style-type: none"><li>• Responsible for implementing this and other integration solutions</li></ul>
<ul style="list-style-type: none"><li>• Systems integrators and other external stakeholders</li></ul>	<ul style="list-style-type: none"><li>• Responsible for external systems that interact with the integration systems</li></ul>
<ul style="list-style-type: none"><li>• Auditors</li></ul>	<ul style="list-style-type: none"><li>• Verify compliance, policies, and integrity of the company</li><li>• Often can be disruptive</li></ul>
<ul style="list-style-type: none"><li>• Users</li></ul>	<ul style="list-style-type: none"><li>• The business users (consumers) of the system</li></ul>

## Responsibilities related to Anypoint Platform



Stakeholder type	Description and responsibilities
<ul style="list-style-type: none"><li>• Users</li></ul>	<ul style="list-style-type: none"><li>• The business users of the system</li></ul>
<ul style="list-style-type: none"><li>• Administrators</li></ul>	<ul style="list-style-type: none"><li>• Users that manage the system</li></ul>
<ul style="list-style-type: none"><li>• Developers and Architects</li></ul>	<ul style="list-style-type: none"><li>• Responsible for implementing this and other integration solutions</li></ul>
<ul style="list-style-type: none"><li>• Systems integrators and other external stakeholders</li></ul>	<ul style="list-style-type: none"><li>• Responsible for external systems that interact with the integration systems</li></ul>
<ul style="list-style-type: none"><li>• Analysts and Managers</li></ul>	<ul style="list-style-type: none"><li>• Responsible for managing the scope, business value, cost, resource usage, etc. of the project, programs, initiatives, etc.</li></ul>

## Exercise 1-1: Identifying stakeholders and interests for an integration solution scenario



- Explore the course case study
- Describe the objectives of an integration solution scenario
- Identify stakeholders for an integration solutions scenario



## An integration architecture must communicate with the various stakeholders



- An integration architecture must carefully **document** and **balance** the needs and **requirements** from all the different, and sometimes opposing, **stakeholders** and their **viewpoints**
- Different architectural **views** are constructed to build consensus with and between
  - Non-technical business level vs. technical level stakeholders
  - End users
  - Development vs. deployment vs. runtime operations stakeholders
  - Implementers vs. managers vs. executives

## An integration architecture must communicate across contexts



- These different views communicate the intended architecture within **contexts** understood by each type of the stakeholders
  - This enables all stakeholders to **verify** and **accept** that the proposed systems will address their **requirements** and **concerns**
- Some architecture documents must translate or bridge between the contexts of different stakeholders
  - Business focused vs. technical focused stakeholders
  - Executive level stakeholders vs. managers vs. implementers

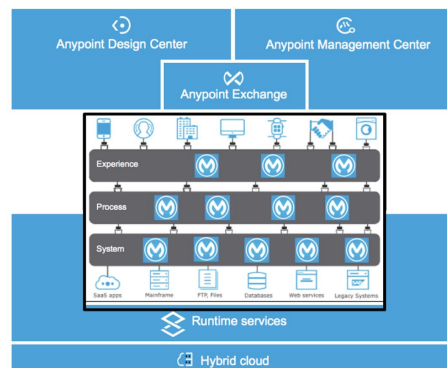
# Architecting for customer success with Anypoint Platform

## What is Anypoint Platform?



- A **unified**, highly productive, **hybrid** enterprise **integration platform** that
  - Creates a seamless collection of **integration applications**
  - Manages their **complete software development lifecycles**

### Anypoint Platform



## MuleSoft's point of view on integration solution architectures vs. enterprise architectures



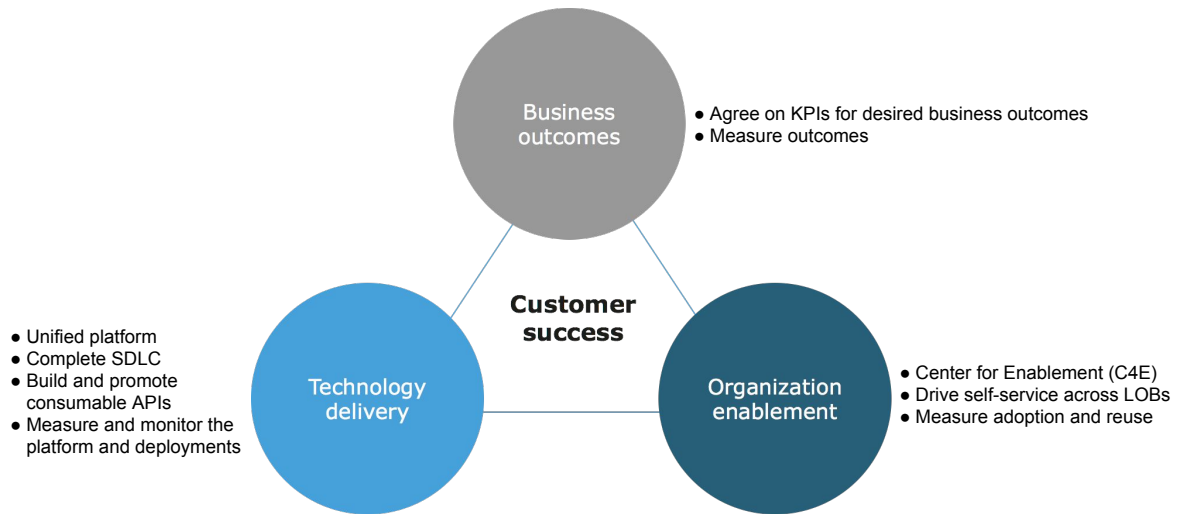
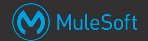
- MuleSoft generally divides architecture projects into two broad areas
- **Integration solutions architecture** (covered in this course)
  - Documents individual integration initiatives
    - For example message-based integration, batch processing, ETL, often in the form of direct integration between two systems
- **Enterprise architecture**
  - Continually builds up an organization's **application network**
    - Covered in the **Anypoint Platform Architecture: Platform Architecture course** and the corresponding **MuleSoft Certified Platform Architect** exam
  - Often involves **layering, reusing, and combining API-Led** initiatives **across organization boundaries**
  - Provides platforms and frameworks to guide individual solution architectures

## MuleSoft's point of view on integration solutions

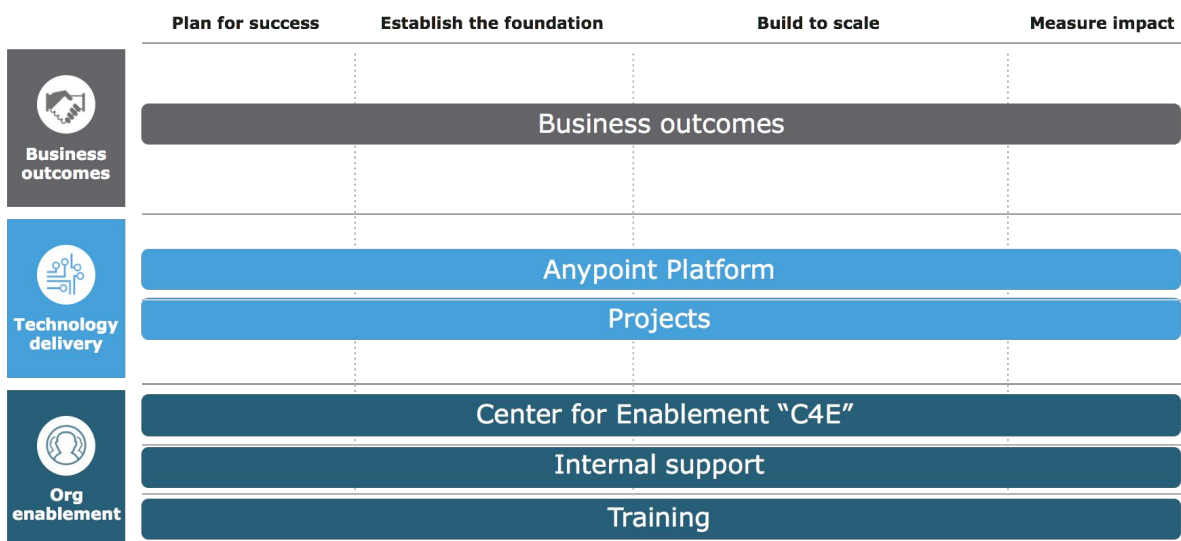


- For both types of architectures, MuleSoft has a strong **point of view** on how to effectively deliver business value across the enterprise using **integration solutions**
  - This includes **proven technology, delivery frameworks, and architectural styles**, focused on customer outcomes
  - Many MuleSoft customers are moving towards building up an application network while still moving fast towards delivering critical individual integration solutions
- MuleSoft can help you decide when and how to build up an enterprise architecture and application network along with your ongoing integration solutions

# The three aspects of MuleSoft's **outcome based delivery** (OBD) approach



## Outcome based delivery timelines



# MuleSoft has a blueprint for you to follow



	Plan for success	Establish the foundation	Build to scale	Measure impact
<b>Business outcomes</b>	Agree on business outcomes and KPIs Develop the overall success plan	Monitor and manage	Refresh the success plan	Measure business outcomes
<b>Technology delivery</b>	Define Anypoint platform vision and roadmap Design Anypoint platform architecture and implementation plan	Deploy Anypoint Platform	Refine and scale Anypoint Platform	Measure Anypoint platform KPIs
	Prioritize IT projects and quick wins Staff and onboard the project teams	Define reference architecture Launch initial projects and quick wins	Onboard additional project teams Launch additional projects	Measure project KPIs
<b>Org enablement</b>	Assess integration capabilities Establish the C4E operating model	Build and publish foundational assets Evangelize	Drive consumption	Measure C4E KPIs
	Onboard MuleSoft Determine the internal support operating model	Staff, train and launch team Publish support guidance and self-serve materials	Monitor Anypoint Platform	Measure support KPIs
	Agree on initial roles Train the initial team(s)	Develop the broader training plan Launch experiential learning opportunities	Update training plan	Conduct skills assessment

*Note: For details about individual steps, contact your MuleSoft Customer Success Manager*

All contents © MuleSoft Inc.

26

## Documenting integration solutions



## Identifying the types of audiences addressed in an integration solution architecture



- The developers
- The operations staff
- Business stakeholders

## Identifying the types of documentation targeted to **developers** in an integration solution architecture



- Required use cases
- Logical views of systems, sub-systems
- Logical views of data and interfaces
- Non-functional requirements (NFRs) and service level agreements (SLAs)
- Process views of interactions and design decisions
- Key decisions, requirements, and tradeoffs
  
- Part 1 of the class focuses on these views and documents

## Identifying the types of documentation targeted to **ops** in an integration solution architecture



- Development view of projects
- Physical view of systems and networks
- Part 2 and 3 of the class focuses on these views and documents

## Documenting use cases for integration solutions



## How scenarios are typically documented



- **User stories** are often used in more modern Agile development, and have a specific format
- They are place keepers to begin conversations about requirements
  - **Use cases** are another option, but do not have a specific format

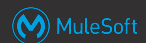
User story template:

- **As a** <what type of user>
- **I want/need to** <goal>
- **So that I can** <business objective>

Example:

- **As a** Market Data Analyst
- **I need to** run the Salesforce & Google analytics reports
- **So that I can** build the monthly media campaign plans

## User stories need detailed acceptance criteria



- **Acceptance criteria** communicate to the development team how to decide when the user story is considered completed (delivered)
- This helps limit the project scope and set expectations

### User story

- **As a** Market Data Analyst
- **I need to** run the Salesforce & Google analytics reports the
- **So that I can** build the monthly media campaign plans

### Acceptance criteria

**Ensure the Marketing Data Analyst is able to:**

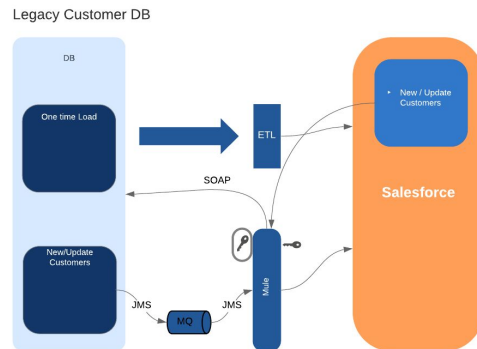
- Access the Salesforce & Google Analytics reports
- Create the monthly media campaign plan for a specified region (e.g. Region 9)
- Access a Contacts list
- Email the prepared monthly media campaign to one or more selected contact(s)



## Use case realizations



- It can be helpful to collect the user stories together into an **integration stories map** to show the information flow between all the user stories
- This is baseline project milestone to scope the project so development can proceed



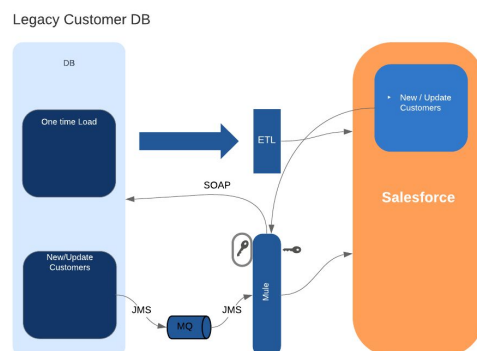
All contents © MuleSoft Inc.

34

## Exercise 1-2: Document user stories for an integration solution



- Identify user stories for a case study
- Collect user stories together into an integration stories map



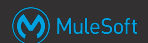
All contents © MuleSoft Inc.

35

# Documenting requirements



## Requirements are often discovered from user stories



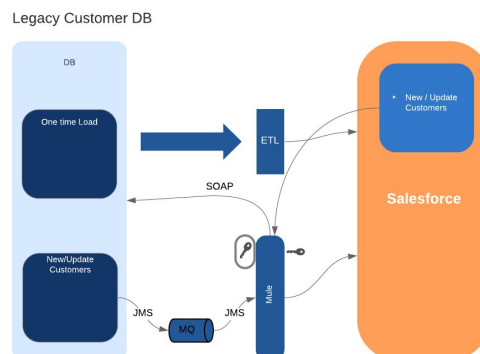
- Functional requirements are often discovered from user stories
  - Triggering events
  - Acceptance criteria or other expected outcomes
  - Expected errors and error handling
- Non-functional requirements may exist beyond or outside a user story
  - May be invented by industry or other external authorities
  - Are constraints on other requirements
    - "Single sign-on is centrally supported by the enterprise LDAP server"
    - "The system must survive Mule runtime restarts"
    - "If the system is offline for 30 seconds, a status page must appear"

## Identifying the types of documentation in an integration solution architecture

- Required use cases
- Views of systems, sub-systems
- Views of data and interfaces
- Functional requirements (FRs), non-functional requirements (NFRs), and service level agreements (SLAs)
  - **NFRs** must carefully specify criteria and tolerances
  - For example: "All data at rest must be encrypted per corporate standards"
- Views of interactions and design decisions
- Key decisions, requirements, and tradeoffs

## Exercise 1-3: Document requirements

- Identify functional requirements for use cases
- Collect and identify non-functional requirements for an integration solution



# Documenting interactions



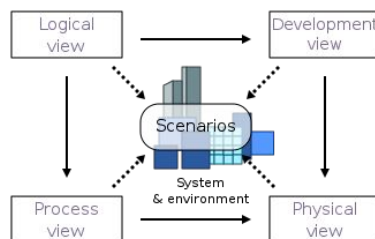
## How to identify and document interactions



- For each use case, the details of all the interactions between systems and stakeholders must be defined
- To work efficiently, use cases must be prioritized and the details specified first for the most important use case(s)

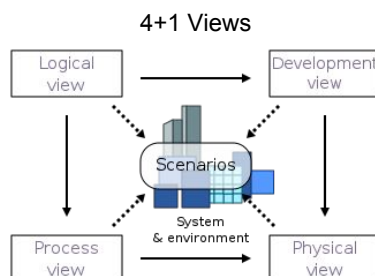
## The **4+1 views** model is one common approach to illustrate views for software intensive systems

- Standard templates and best practices should be agreed upon and used for architecture documents
- The 4+1 views describe software systems
  - From the viewpoint of different stakeholders
    - Such as end-users, developers, and project managers
  - At various technical levels and different phases of a project's lifecycle
- [https://en.wikipedia.org/wiki/4%2B1\\_architectural\\_view\\_model](https://en.wikipedia.org/wiki/4%2B1_architectural_view_model)



## The architecture methodology used in this course

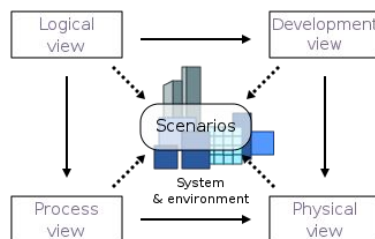
- In the course exercises, you will be filling in parts an architecture specification
- There are many widely accepted architectural approaches and styles
- In this course, the architecture templates are based on **4+1 views**



## The 1 in the 4+1 view model is a set of **scenarios**



- The scenarios illustrate the behavior of the system from an **end user** perspective
  - Helps to **validate** the **architectural design** and **requirements** with key stakeholders
  - In typical integration solutions, the end user might not be a person
    - May be another system, sub-system, scheduler, or cron job



All contents © MuleSoft Inc.

44

## Reflection questions



- Have you used 4+1 views before?
- What other architecture approaches have you used?
- Have you used user stories or use cases, and if so, how did they compare with the user stories template presented earlier?

All contents © MuleSoft Inc.

45

# Architecting with 4+1 views



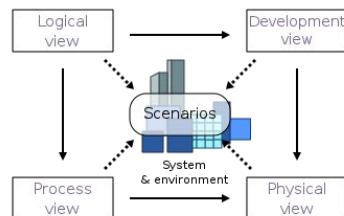
## Defining architectural views for Mule applications



- Combining various architectural views related to a Mule application provides a complete realization of an integration solution using MuleSoft
  - Often use various UML diagrams
- **Sequence** and/ **activity** diagrams are typically associated with use case realizations of the systems
  - Mule flows are similar to activity diagrams, so mocked Mule flows can also be used to help design integration solutions
- Later in the project lifecycle, **deployment diagrams** can be associated with use case realizations
- Per use case, architects may provide other views at various detail levels (zooming in or out) to help clarify the integration solution

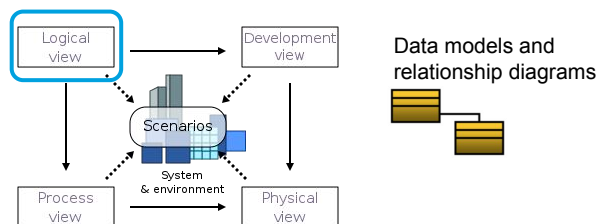
## How 4+1 views are tied to common project lifecycle phases

- The analysis phases focuses more on the business requirements
  - Usually focuses on the **scenarios** and **logical views**
- The design phases refines the logical views into more actionable views
  - Oriented to more technical stakeholders
  - Design actual interfaces and interactions to realize the user stories
  - **Process views**, **development views**, and **physical views**
- The rest of the project lifecycle (implementation, deployment, maintenance) is informed by the design phase
  - Earlier phases are continuously revisited and refined in an **iterative manner**



## The 4+1 **logical view** illustrates end-to-end system functionality

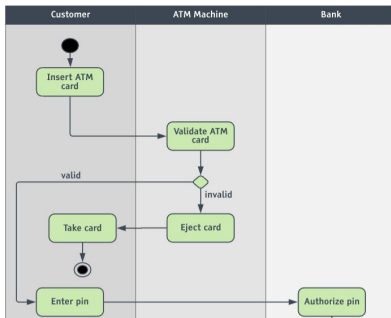
- Usually defines and documents **system**, **stakeholders**, and **interfaces**, and their **relationships**
  - Major big picture view
  - Diagrams show the **relationships** between systems and stakeholders and existing relationships and connectivity
  - Includes **data models** for the integration domain and systems of record/reference
  - These views are included in the larger enterprise distributed architecture



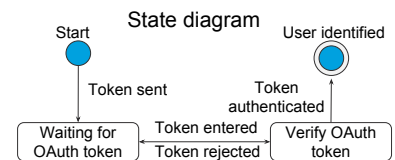
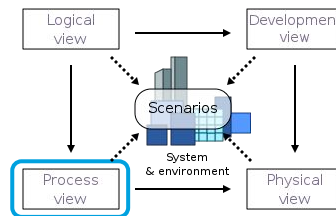


## The 4+1 **process view** illustrates the runtime behavior of systems

- Give more detail of a particular movement of information, usually per use case
- Specifies how SLAs and NFRs are met, including security policies
- Documents both success and failure paths
- Documents human interactions in the business processes



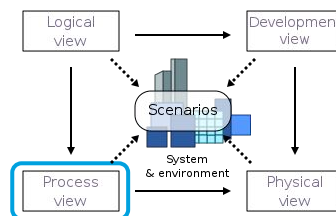
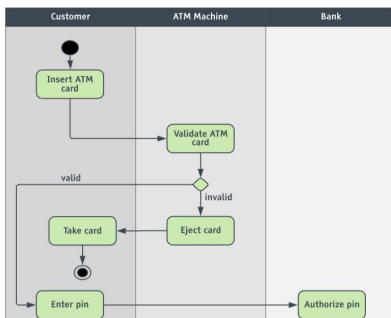
All contents © MuleSoft Inc.



51

## The 4+1 **process view** illustrates the runtime behavior of systems

- Activity diagrams illustrate communication across **swimlanes** at a more technical level, while still skipping implementation details
  - Swimlanes visually partition ownership by person, group, system, or sub-system
  - Document **concurrency, distribution, integrators, performance, and scalability** of systems and processes

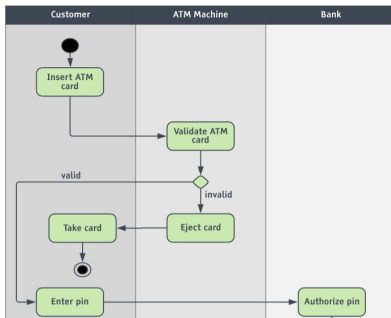


52

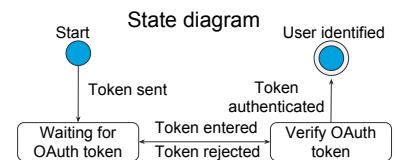
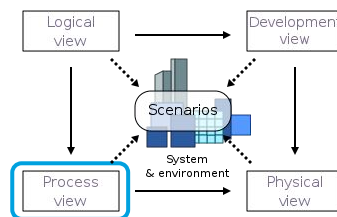
# Anypoint Studio can quickly represent process views



- Flows are similar to activity diagrams
- They do not need class diagrams or state diagrams



All contents © MuleSoft Inc.

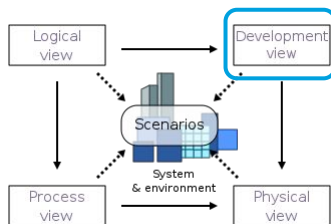


53

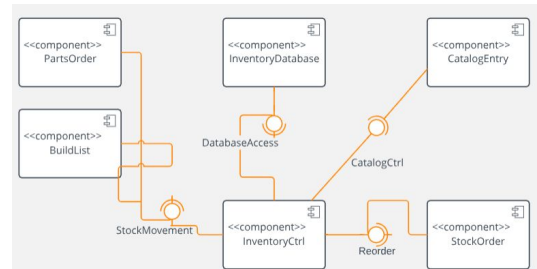
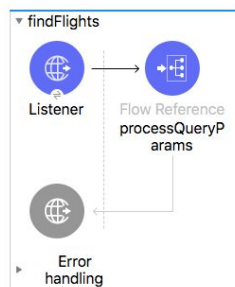
# The 4+1 **development view** illustrates systems from a developer perspective



- Documents **component** and **packages** with diagrams to illustrate how to code the solution
- Often the development view does not directly relate to Mule apps
  - MuleSoft's drag-and-drop development tools often shield the developer from these coding details
  - But they may be included in the larger enterprise distributed architecture



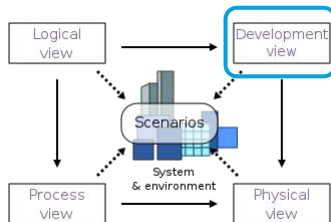
All contents © MuleSoft Inc.



54

## The 4+1 **development view** categories

- Development processes to deliver solutions
  - Test automation and strategies, performance testing, release management, branch/merge strategies, Continuous Integration (CI)
- Project scoping and reuse
  - Microservice architecture, API-Led, reusable libraries, version and platform dependencies and decisions
- Solution quality
  - Standards and frameworks, protocols, logging, and instrumentation

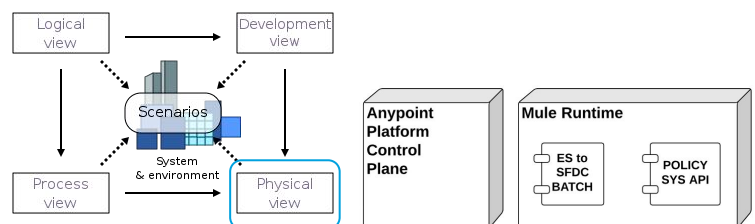


All contents © MuleSoft Inc.

55

## The 4+1 **physical view** documents systems deployments for system integrators

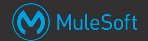
- Uses **deployment diagrams** to illustrate the physical view of systems, including showing
  - Components ("nodes") that currently exist or will exist
    - For example enterprise databases, JMS servers, or SaaS systems (like Salesforce)
  - What software components ("artifacts") run on each node
    - For example Mule applications, custom Java code, or Apex APIs
  - How the different pieces are connected (e.g. JDBC, REST, SOAP)



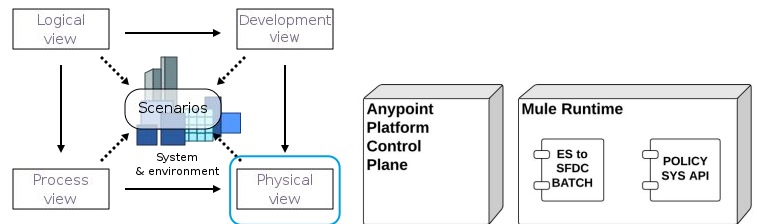
All contents © MuleSoft Inc.

56

## Common 4+1 **physical view** documents



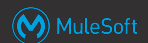
- High level network topology
  - Database instances, Mule runtimes, CloudHub workers, DLBs, firewalls, VPCs and network tunnels
- Licensing
- Infrastructure, including runtime and control planes
- Continuous Delivery (CD) and other environments
  - Mule environments must coexist and conform with existing environments



All contents © MuleSoft Inc.

57

## Other required documentation



- **Maintenance, operations, and security** may need specialized documents
- These stakeholders may need separate documents and manuals culled and refined from the 4+1 views

All contents © MuleSoft Inc.

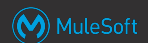
58

## Exercise 1-4: Document 4+1 views for a use case



- Add 4+1 view sections to an architecture document
- Locate 4+1 views for an existing MuleSoft-enabled solution
- Create a process view for a use case
- Explore completed 4+1 views for a use case

## Reflection questions



- Which 4+1 views often use use case diagrams or user stories?
- Which 4+1 views often use sequence diagrams?
- Which 4+1 views often use activity diagrams?
- Which 4+1 views often use class diagrams?
- Which 4+1 views often use state diagrams?
- What is the difference between a state diagram and an activity diagram?

- Identify uses cases and requirements
  - Open the case study for the course
  - Discuss the use cases to be implemented by this integration project
  - Identify and discuss functional requirements
  - Identify and discuss non-functional requirements
  - Identify deliverable documentation that will explain how all requirements of the case study will be realized

- Begin to document a use case for the integration solution
  - Open the system integration architecture document template
  - Add the case study description and associated use cases to the architecture template
  - Add functional requirements to the architecture document
  - Add non-functional requirements to the architecture document
  - Identify the types of views required to complete the architecture document
  - Identify the types of optional views that may also be included to complete the architecture document

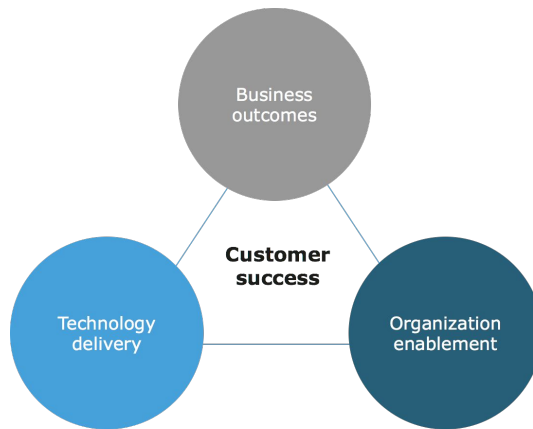
- Have each student present their solution, with group discussion guided by the instructor
- Compare and discuss the different solutions presented
- Fill in the solution architecture document with an agreed solution

# Summary



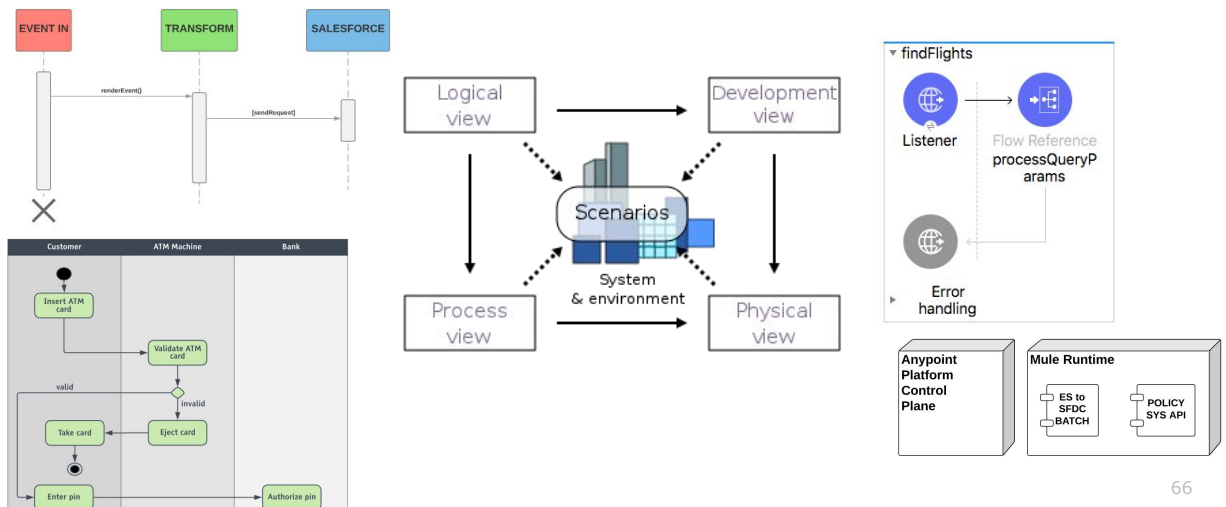
## Summary

- MuleSoft uses an outcome based delivery approach to deliver integration solutions



## Summary

- MuleSoft integration architectures document various views





- The development views can be simplified by using the MuleSoft toolset
  - May not need to create class and state diagrams for the logical view
  - May not need to create component and package diagrams for the development view
- MuleSoft has a proven point of view on how to successfully architect integration solutions

