

TIBCO BusinessWorks to Spring Boot Migration Guide

Table of Contents









1. Overview
2. Architecture
3. Component Details
4. Setup Instructions
5. Usage Examples
6. Flow Diagrams
7. Sequence Diagrams
8. Generated Outputs
9. Troubleshooting

1. Overview

Project Summary

This project provides an **automated migration framework** to convert TIBCO BusinessWorks (BW) process definitions into modern Spring Boot microservices (REST and SOAP).

Key Features

-  Multi-agent AI architecture for intelligent code generation
-  Parallel processing support for multiple BW processes
-  Generates both REST and SOAP Spring Boot services
-  Complete Spring Boot project structure with Maven configuration
-  JAXB/XSD integration for type safety
-  JPA/Hibernate for database operations
-  JMS messaging support
-  Automated packaging and archiving

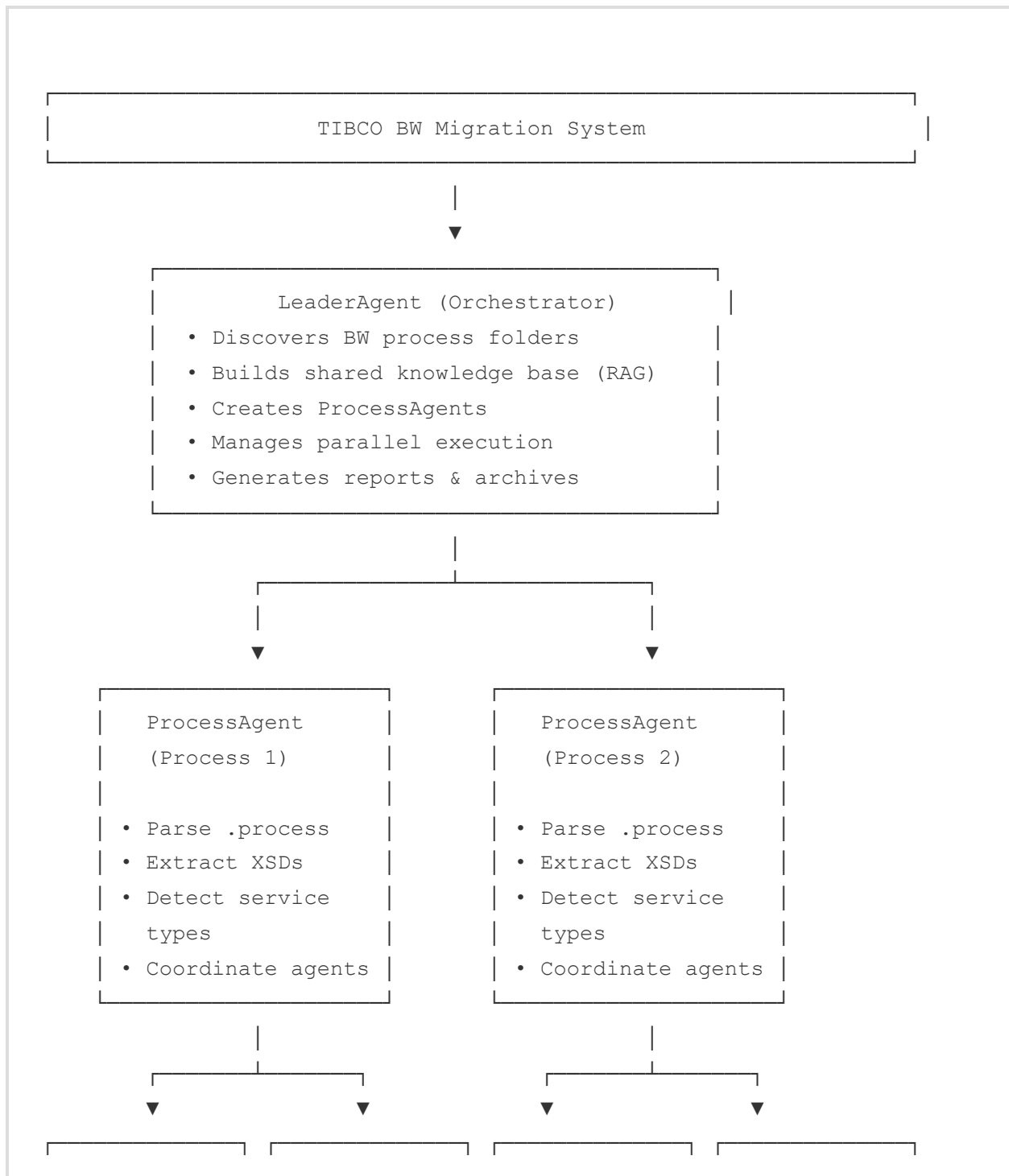
Technology Stack

Category	Technology	Version
Input Format	TIBCO BusinessWorks	.process, .xsd, .bwp files
Framework	Spring Boot	3.2.0
Language	Java	17

REST	Spring Web	Included in Spring Boot
SOAP	Spring Web Services	Included in Spring Boot
Database	Spring Data JPA	H2 / Oracle
Messaging	Spring JMS	Apache Artemis
XML Binding	JAXB	Jakarta XML Binding

2. Architecture

High-Level System Architecture



RestService	SoapService	RestService	SoapService
Agent	Agent	Agent	Agent
• REST API	• SOAP/WSDL	• REST API	• SOAP/WSDL
• Controller	• Endpoint	• Controller	• Endpoint
• DTOs	• JAXB DTOs	• DTOs	• JAXB DTOs
• Service	• Config	• Service	• Config

3. Component Details

3.1 LeaderAgent

Purpose: Orchestrates the entire migration process

Location: `generator/ai/leader.py`

Key Responsibilities

- Scan input directory for BW process folders
- Build shared RAG (Retrieval-Augmented Generation) knowledge base
- Create and manage ProcessAgents for each BW folder
- Execute parallel or sequential processing
- Generate migration reports and ZIP archives

Configuration Example

```
leader = LeaderAgent(  
    input_base='input_artifacts',      # BW process folder  
    output_base='output',              # Generated code output  
    package_root='com.example.tibco',  # Java package  
    parallel=True,                    # Enable parallel processing  
    max_workers=4                     # Thread pool size  
)
```

3.2 ProcessAgent

Purpose: Analyzes individual BW process and coordinates service generation

Location: generator/ai/process_agent.py

Key Responsibilities

1. Parse .process files using process_parser
2. Extract activities (JDBC, JMS, HTTP, SOAP)
3. Identify XSD schemas
4. Determine service type (REST vs SOAP)
5. Generate shared artifacts (Entities, Repositories, Services, DTOs)

3.3 RestServiceAgent

Purpose: Generates Spring Boot REST API implementation

Generated Files

- pom.xml - Maven with Spring Web dependencies
- *Controller.java - @RestController with endpoints
- *Service.java - @Service with business logic
- *Repository.java - JpaRepository interface
- *Entity.java - JPA Entity classes
- *Request.java / *Response.java - DTOs
- application.yml - Spring Boot configuration

3.4 SoapServiceAgent



Purpose: Generates Spring Boot SOAP Web Service (Spring-WS)

Generated Files

- `SoapServiceApplication.java` - `@SpringBootApplication`
- `WebServiceConfig.java` - WSDL/XSD configuration
- `*Endpoint.java` - `@Endpoint` with `@PayloadRoot`
- JAXB-annotated DTOs with `@XmlElement`
- `pom.xml` - Maven with Spring-WS dependencies
- Unified XSD schema files

4. Setup Instructions

4.1 Prerequisites

Required Software

- Python 3.8 or higher
- Java 17 or higher (JDK)
- Maven 3.6+ (optional, for building generated projects)

4.2 Installation Steps

Step 1: Navigate to Project Directory

```
cd c:\IBM\DBS-TIBCO-SPRINGBOOT\PY29OCT\tibco_migration
```

Step 2: Set Up Python Virtual Environment

```
# Create virtual environment
python -m venv .venv

# Activate virtual environment (Windows PowerShell)
.venv\Scripts\activate

# Install dependencies (if requirements.txt exists)
pip install -r requirements.txt
```

Step 3: Verify Directory Structure

```
tibco_migration/  
├── generator/  
│   ├── generator/  
│   │   ├── ai/  
│   │   │   ├── leader.py  
│   │   │   ├── process_agent.py  
│   │   │   ├── service_agents.py  
│   │   │   └── run.py  
│   │   └── process_parser.py  
│   ├── input_artifacts/  
│   └── output/  
└── MIGRATION_GUIDE.md
```

5. Usage Examples


5.1 Basic Migration

Input Structure

```
input_artifacts/  
├─ LoanApp/  
│   ├── LoanApplication.process  
│   ├── loan_request.xsd  
│   └─ loan_response.xsd
```

Run Migration

```
cd generator  
python -m generator.ai.run --input-dir input_artifacts --output-dir output
```



Expected Output

```
output/  
├─ rest/                # Spring Boot REST project  
├─ soap/               # Spring Boot SOAP project  
├─ src_rest.zip         # REST source archive  
├─ src_soap.zip         # SOAP source archive  
└─ ai_migration_report.json
```

5.2 Testing REST Service

Build and Run

```
cd output\rest
mvn clean package
mvn spring-boot:run
```

Sample REST Request

```
curl -X POST http://localhost:8080/loan/apply \
-H "Content-Type: application/json" \
-d '{
  "customerID": "CUST001",
  "loanAmount": 50000,
  "loanType": "Personal",
  "loanTermMonths": 36,
  "applicantName": "John Doe",
  "contactEmail": "john.doe@example.com"
}'
```

Expected Response

```
{
  "loanID": "550e8400-e29b-41d4-a716-446655440000",
  "status": "APPROVED",
  "message": "Loan application approved",
  "approvedAmount": 50000
}
```

5.3 Testing SOAP Service

Build and Run

```
cd output\soap
mvn clean package
```

```
mvn spring-boot:run
```

Access WSDL

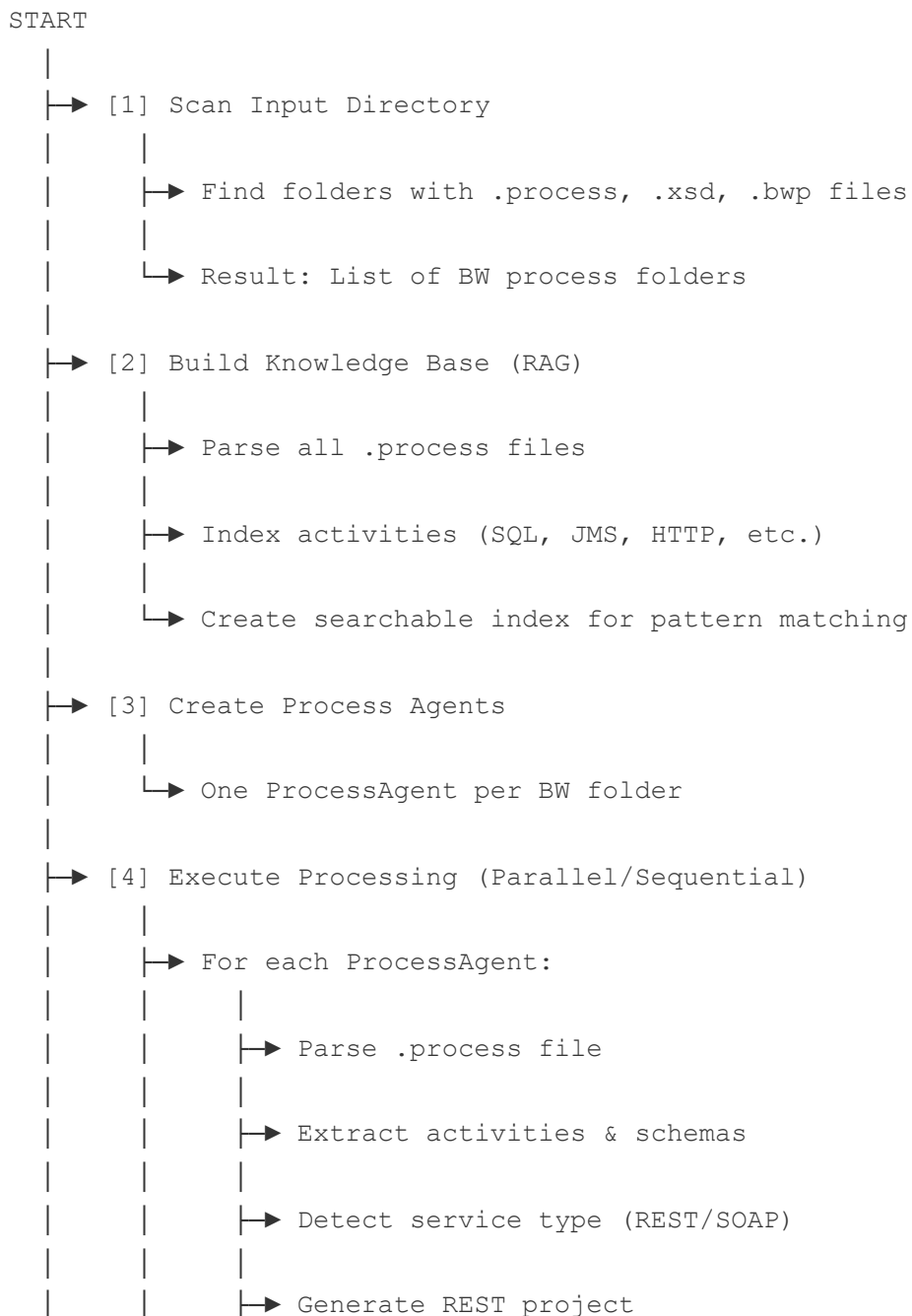
```
http://localhost:8081/ws/loanApplication.wsdl
```

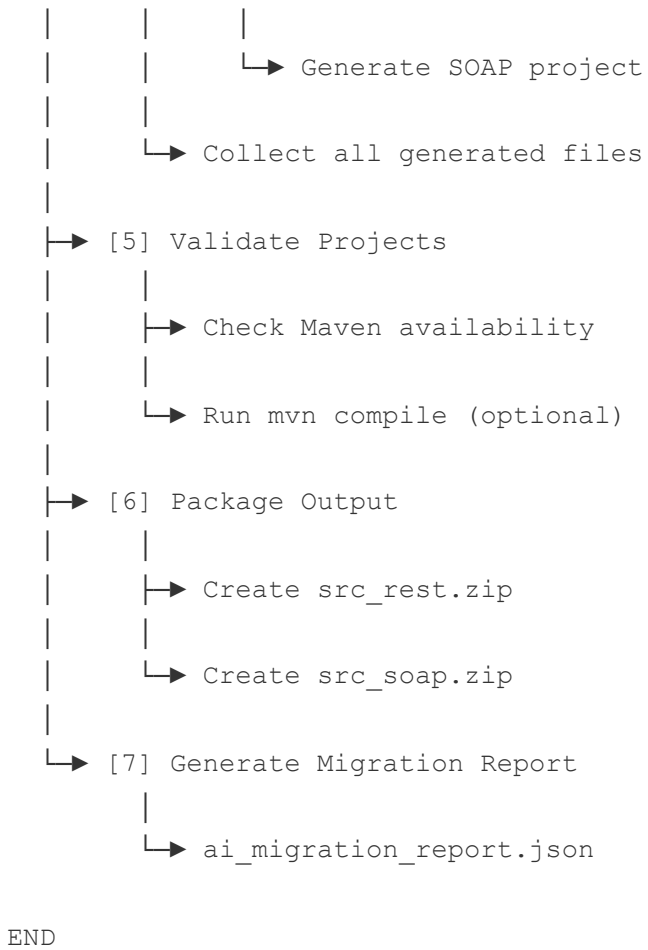
Sample SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:loan="http://example.com/tibco_migration/loan">
  <soapenv:Header/>
  <soapenv:Body>
    <loan:LoanApplicationRequest>
      <loan:customerID>CUST001</loan:customerID>
      <loan:loanAmount>50000</loan:loanAmount>
      <loan:loanType>Personal</loan:loanType>
      <loan:loanTermMonths>36</loan:loanTermMonths>
      <loan:applicantName>John Doe</loan:applicantName>
      <loan:contactEmail>john.doe@example.com</loan:contactEmail>
    </loan:LoanApplicationRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

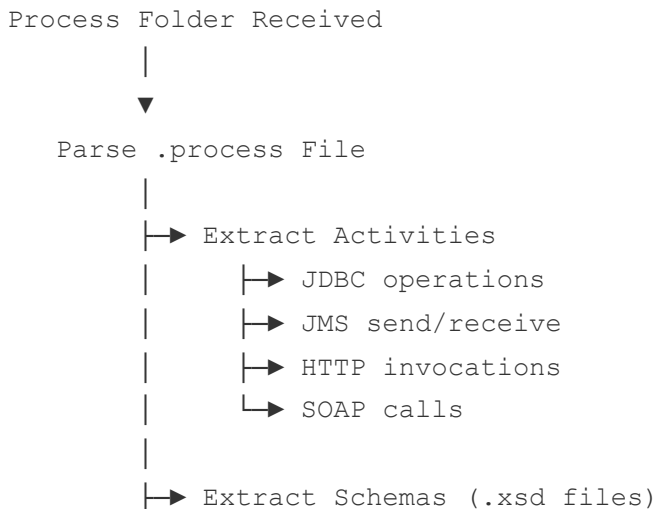
6. Flow Diagrams

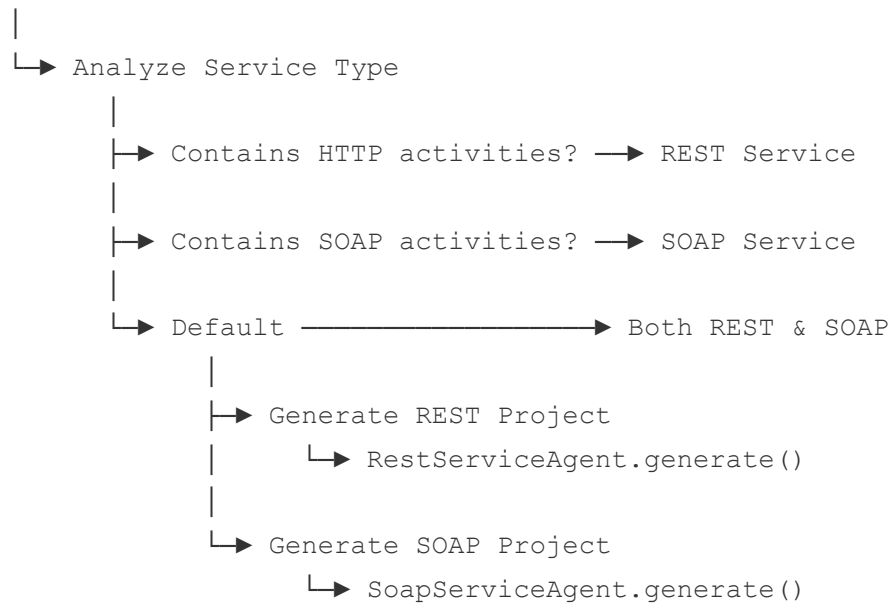
6.1 Overall Migration Flow





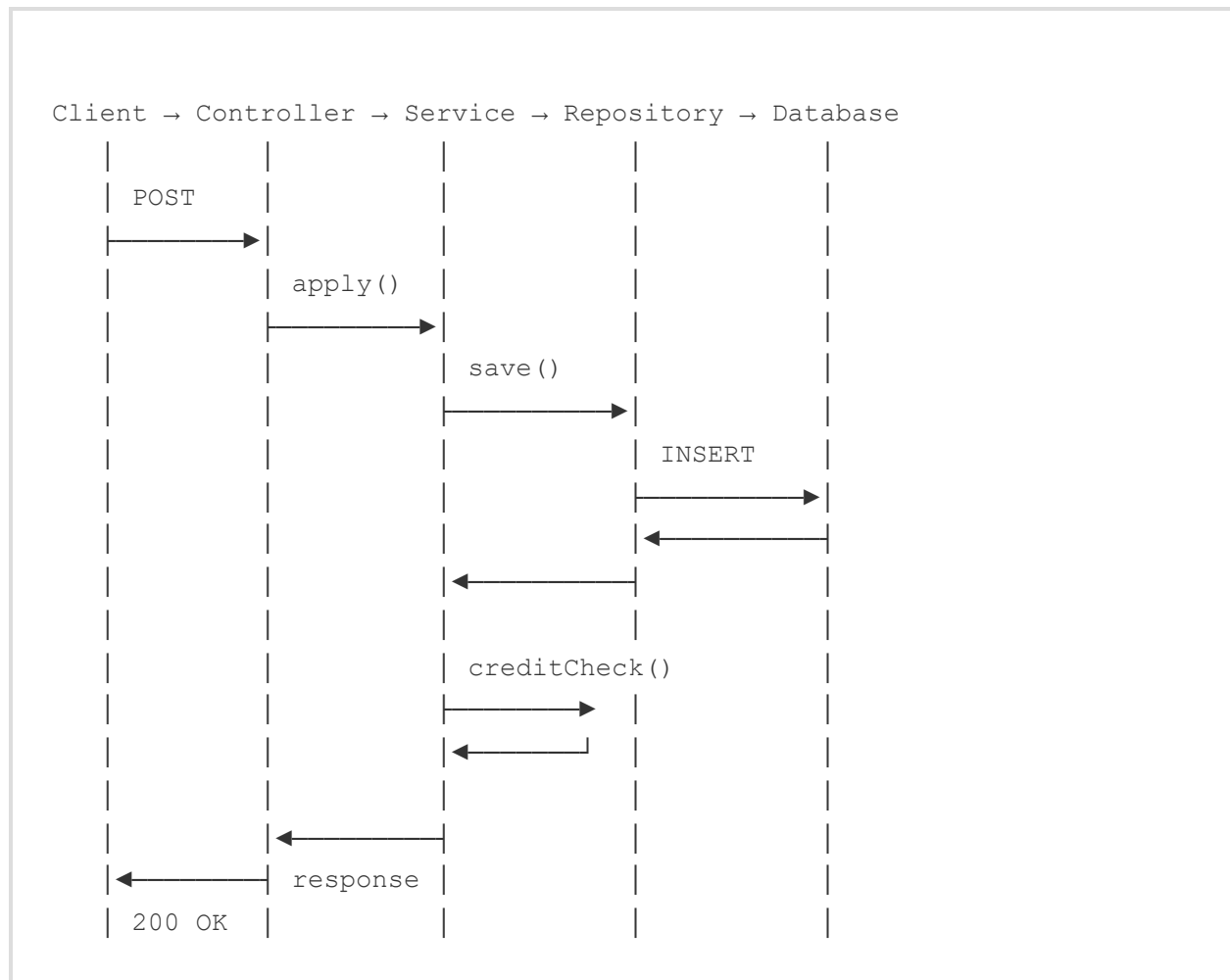
6.2 Process Agent Decision Flow





7. Sequence Diagrams

7.1 REST Service Request Flow



8. Generated Outputs

8.1 REST Service Structure

```
output/rest/  
├─ pom.xml  
├─ src/main/java/com/example/tibco_migration/  
│   ├─ controller/  
│   │   └─ LoanApplicationController.java  
│   ├─ service/  
│   │   └─ LoanApplicationService.java  
│   ├─ repository/  
│   │   └─ LoanRepository.java  
│   ├─ entity/  
│   │   └─ LoanEntity.java  
│   └─ dto/  
│       ├─ LoanApplicationRequest.java  
│       └─ LoanApplicationResponse.java  
└─ src/main/resources/  
    ├─ application.yml  
    └─ xsd/  
        ├─ loan_request.xsd  
        └─ loan_response.xsd
```

8.2 SOAP Service Structure

```
output/soap/  
├─ pom.xml
```

```
|─ src/main/java/com/example/tibco_migration/
|   |─ SoapServiceApplication.java
|   |─ config/
|   |   |─ WebServiceConfig.java
|   |   └─ WebClientConfig.java
|   |─ endpoint/
|   |   └─ LoanApplicationEndpoint.java
|   |─ service/
|   |   └─ LoanApplicationService.java
|   |─ repository/
|   |   └─ LoanRepository.java
|   |─ entity/
|   |   └─ LoanEntity.java
|   └─ dto/
|       |─ LoanApplicationRequest.java (JAXB)
|       └─ LoanApplicationResponse.java (JAXB)
└─ src/main/resources/
    |─ application.yml
    └─ xsd/
        └─ loan_request.xsd
```

9. Troubleshooting

Common Issues and Solutions

Issue	Symptom	Solution
AI Libraries Not Available	Warning: "No module named 'numpy'"	Normal operation. System uses fallback mode. Optional: <code>pip install numpy scikit-learn</code>
Maven Not Found	"Maven executable not found"	Install Maven or skip validation: <code>choco install maven</code>
Port Already in Use	"Port 8080 is already in use"	Change port in <code>application.yml</code> : <code>server.port: 8090</code>
JAXB Classes Not Generated	"Cannot find generated JAXB classes"	Run: <code>mvn clean compile</code>
WSDL Not Accessible	"404 Not Found for WSDL"	Verify XSD exists in <code>src/main/resources/xsd/</code>

Debug Mode

Enable detailed logging in `application.yml` :

```
logging:  
  level:  
    org.springframework.ws: DEBUG  
    com.example.tibco_migration: DEBUG  
    org.springframework.jms: DEBUG  
    org.springframework.data.jpa: DEBUG
```

Quick Reference

Command Summary

Action	Command
Run Migration	<code>python -m generator.ai.run --input-dir input_artifacts --output-dir output</code>
Build REST Service	<code>cd output/rest && mvn clean package</code>
Build SOAP Service	<code>cd output/soap && mvn clean package</code>
Run REST Service	<code>mvn spring-boot:run</code> (in rest folder)
Run SOAP Service	<code>mvn spring-boot:run</code> (in soap folder)

Service URLs

Service	URL
REST API	<code>http://localhost:8080</code>
SOAP WSDL	<code>http://localhost:8081/ws/loanApplication.wsdl</code>
H2 Console (REST)	<code>http://localhost:8080/h2-console</code>

H2 Console (SOAP)	http://localhost:8081/h2-console
-------------------	---

Document Version: 1.0

Last Updated: October 30, 2025

Project: TIBCO BusinessWorks to Spring Boot Migration Framework