Case study 1

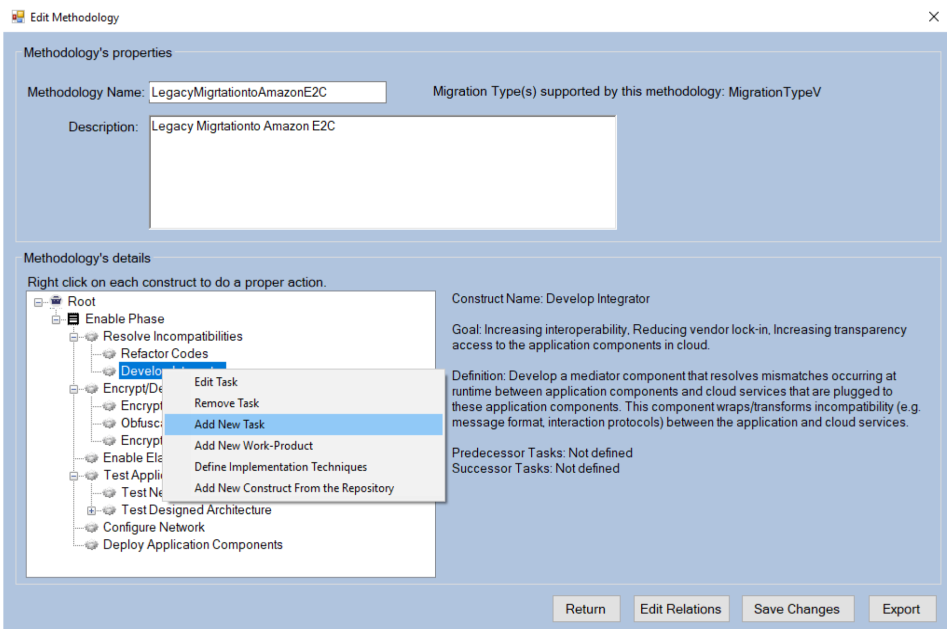
This document discusses tailoring steps of a reengineering method in MLSAC.

Step3.1 The method engineer may believe excluding some default elements in the method that are not necessary. For example, the method engineer recognises that there are strong data security mechanisms in Amazon servers. In addition, the database includes non-critical data for example temporary tables. Thus, there is no longer need for encrypting the application database. Then she remove the element *encrypt database* from the method to avoid application performance degradation.

Step3.2 The tailoring procedure allows adding of new elements such as phase, tasks, and work-products, which might not have been already supported by the meta-model. These elements can be added from various sources such as past developer experience or existing methods. For example, the method engineer may define a particular legacy system code refactoring to inform developers of updating the legacy system APIs to make them compatible with Amazon EC2 APIs to prevent semantical or syntactical incompatibilities. The method engineer adds a new element, as a subclass and named *update APIs and framework*, under the element *refactor codes* in enable phase (Figures 1 and 2).

Step3.3 MLSAC gives the ability of defining arbitrary implementation techniques to operationalise method elements. Suppose, the organisation wants moving the legacy database to Amazon Aurora database analytics engine. It is likely that meaning of some legacy columns are the same with target columns, however, the unit of measurement is different which causes a sematic change in the interpretation of the data. For example, both platforms may use different currencies or precision in saving decimal points. In this scenario, it is important to not lose or change the original legacy application data after integration with Amazon Aurora. To assure this, as shown in Figure 3, the method engineer firstly prescribes the following technique including two steps (1) Interoperability test should cover test cases for identifying sematic inconsistencies and incompatibilities between the legacy database and migrated data to Amazon Aurora database. The important columns that should be examined are those contain currency and decimal points, (2) developers should manually compare columns in both legacy and Amazon Aurora databases by looking at the main screen of the system. This technique is then assigned to the element *test interoperability* (Figure 4 and 5).

Step3.4 The method engineer can override existing sequences defined by the MLSAC to define a specific sequence of method elements. Figure 5, shows she defines that *recover legacy application knowledge* should be performed before a cloud platform is chosen via task *choose cloud provider*.


Edit Methodology

Methodology's properties

Methodology Name: LegacyMigrtationtoAmazonE2C

Migration Type(s) supported by this methodology: MigrationTypeV

Description: Legacy Migrtationto Amazon E2C

Methodology's details

Right click on each construct to do a proper action.

- Root
  - Enable Phase
    - Resolve Incompatibilities
      - Refactor Codes
      - Develo
    - Encrypt/De
      - Encrypt
      - Obfusc
      - Encrypt
    - Enable Ela
    - Test Appli
      - Test Ne
      - Test Designed Architecture
    - Configure Network
    - Deploy Application Components

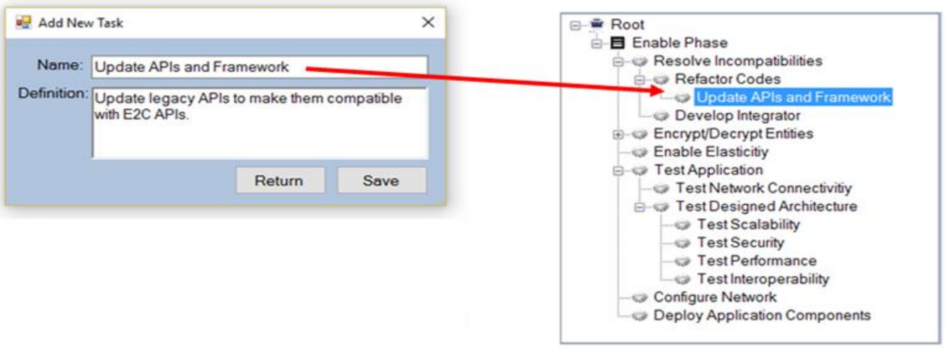
Construct Name: Develop Integrator

Goal: Increasing interoperability, Reducing vendor lock-in, Increasing transparency access to the application components in cloud.

Definition: Develop a mediator component that resolves mismatches occurring at runtime between application components and cloud services that are plugged to these application components. This component wraps/transforms incompatibility (e.g. message format, interaction protocols) between the application and cloud services.

Predecessor Tasks: Not defined
Successor Tasks: Not defined

Figure 1. Adding new task element to the method


Add New Task

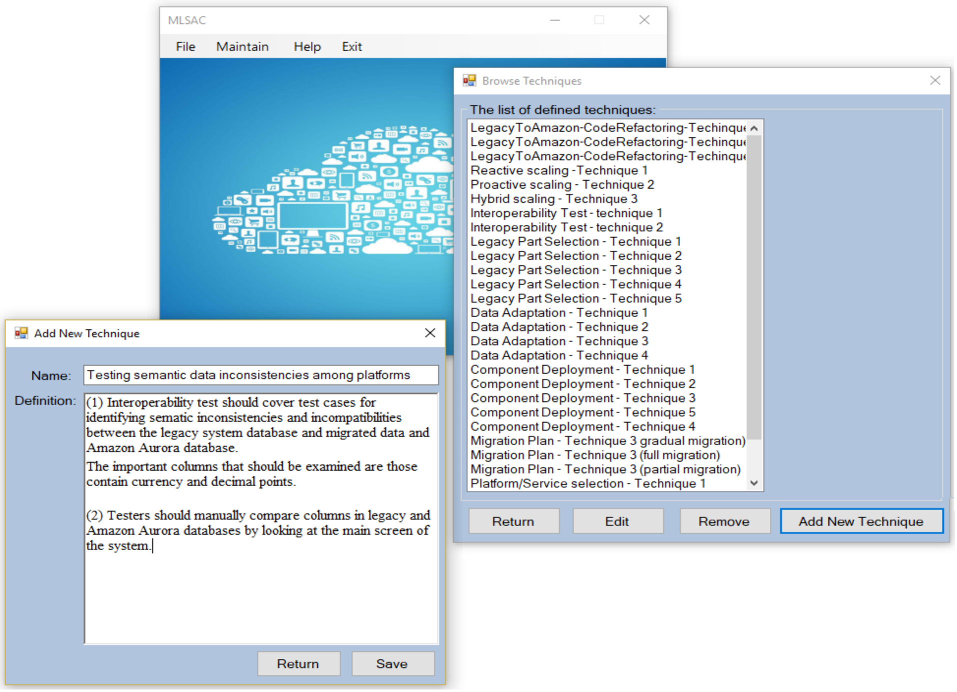
Name: Update APIs and Framework

Definition: Update legacy APIs to make them compatible with E2C APIs.

Return Save

- Root
  - Enable Phase
    - Resolve Incompatibilities
      - Refactor Codes
        - Update APIs and Framework
      - Develop Integrator
    - Encrypt/Decrypt Entities
    - Enable Elasticitiy
    - Test Application
      - Test Network Connectivitiy
      - Test Designed Architecture
        - Test Scalability
        - Test Security
        - Test Performance
        - Test Interoperability
    - Configure Network
    - Deploy Application Components

Figure 2 Adding a subclass to the method element *refactor codes*


MLSAC

File Maintain Help Exit

Browse Techniques

The list of defined techniques:

LegacyToAmazon-CodeRefactoring-Techinque
LegacyToAmazon-CodeRefactoring-Techinque
LegacyToAmazon-CodeRefactoring-Techinque
Reactive scaling -Technique 1
Proactive scaling - Technique 2
Hybrid scaling - Technique 3
Interoperability Test - technique 1
Interoperability Test - technique 2
Legacy Part Selection - Technique 1
Legacy Part Selection - Technique 2
Legacy Part Selection - Technique 3
Legacy Part Selection - Technique 4
Legacy Part Selection - Technique 5
Data Adaptation - Technique 1
Data Adaptation - Technique 2
Data Adaptation - Technique 3
Data Adaptation - Technique 4
Component Deployment - Technique 1
Component Deployment - Technique 2
Component Deployment - Technique 3
Component Deployment - Technique 5
Component Deployment - Technique 4
Migration Plan - Technique 3 gradual migration)
Migration Plan - Technique 3 (full migration)
Migration Plan - Technique 3 (partial migration)
Platform/Service selection - Technique 1

Return Edit Remove Add New Technique

Add New Technique

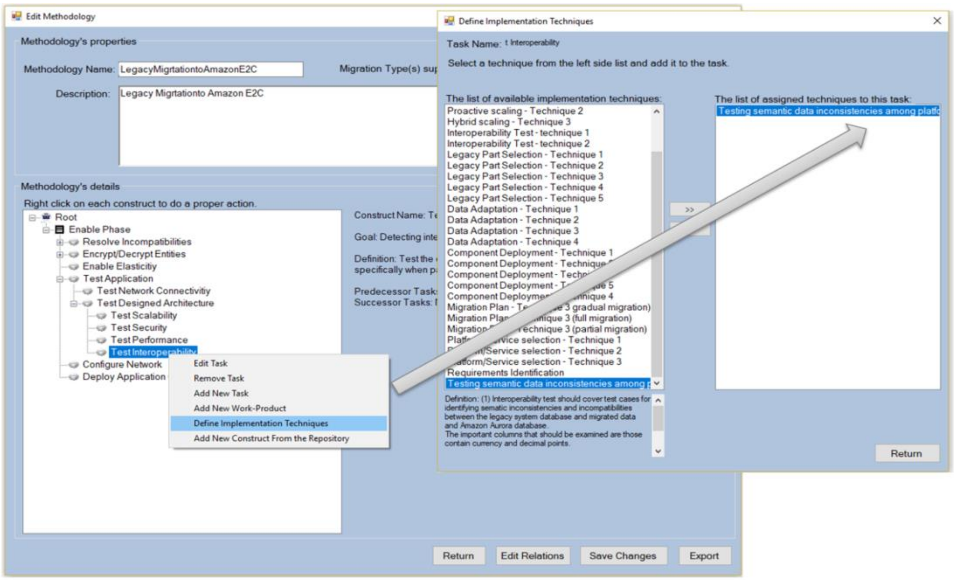
Name: Testing semantic data inconsistencies among platforms

Definition: (1) Interoperability test should cover test cases for identifying sematic inconsistencies and incompatibilities between the legacy system database and migrated data and Amazon Aurora database.
The important columns that should be examined are those contain currency and decimal points.

(2) Testers should manually compare columns in legacy and Amazon Aurora databases by looking at the main screen of the system.

Return Save

Figure 3 Defining an implementing technique for *test interoperability* method element avoiding semantic data inconsistency


Edit Methodology

Methodology's properties

Methodology Name: LegacyMigrtationtoAmazonE2C

Migration Type(s) sup

Description: Legacy Migrtationto Amazon E2C

Methodology's details

Right click on each construct to do a proper action.

- Root
  - Enable Phase
    - Resolve Incompatibilities
    - Encrypt/Decrypt Entities
    - Enable Elasticitiy
    - Test Application
      - Test Network Connectivitiy
      - Test Designed Architecture
        - Test Scalability
        - Test Security
        - Test Performance
        - Test Interoperability
    - Configure Network
    - Deploy Application

Edit Task
Remove Task
Add New Task
Add New Work-Product
Define Implementation Techniques
Add New Construct From the Repository

Construct Name: Te

Goal: Detecting inte

Definition: Test the
specifically when p

Predecessor Task:
Successor Tasks: I

Return Edit Relations Save Changes Export

Define Implementation Techniques

Task Name: t Interoperability

Select a technique from the left side list and add it to the task.

The list of available implementation techniques:

Proactive scaling - Technique 2
Hybrid scaling - Technique 3
Interoperability Test - technique 1
Interoperability Test - technique 2
Legacy Part Selection - Technique 1
Legacy Part Selection - Technique 2
Legacy Part Selection - Technique 3
Legacy Part Selection - Technique 4
Legacy Part Selection - Technique 5
Data Adaptation - Technique 1
Data Adaptation - Technique 2
Data Adaptation - Technique 3
Data Adaptation - Technique 4
Component Deployment - Technique 1
Component Deployment - Technique
Component Deployment - Techn
Component Deployment - T que 5
Component Deployme nique 4
Migration Plan - Te 3 gradual migration)
Migration Pla nique 3 (full migration)
Migration echnique 3 (partial migration)
Platf vice selection - Technique 1
P n/Service selection - Technique 2
atform/Service selection - Technique 3
Requirements Identification
Testing semantic data inconsistencies among p

>>

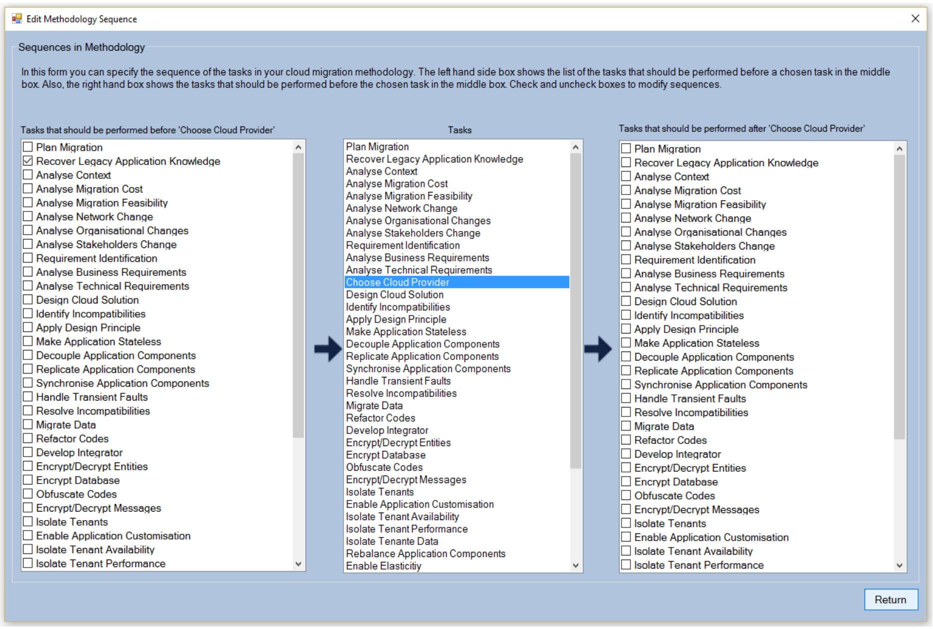
The list of assigned techniques to this task:

Testing semantic data inconsistencies among platfo

Definition: (1) Interoperability test should cover test cases for identifying sematic inconsistencies and incompatibilities between the legacy system database and migrated data and Amazon Aurora database.
The important columns that should be examined are those contain currency and decimal points.

Return

Figure 4 Assigning the defined technique to method element *test interoperability*


Edit Methodology Sequence

Sequences in Methodology

In this form you can specify the sequence of the tasks in your cloud migration methodology. The left hand side box shows the list of the tasks that should be performed before a chosen task in the middle box. Also, the right hand box shows the tasks that should be performed before the chosen task in the middle box. Check and uncheck boxes to modify sequences.

Tasks that should be performed before 'Choose Cloud Provider'

- ☐ Plan Migration
- ☑ Recover Legacy Application Knowledge
- ☐ Analyse Context
- ☐ Analyse Migration Cost
- ☐ Analyse Migration Feasibility
- ☐ Analyse Network Change
- ☐ Analyse Organisational Changes
- ☐ Analyse Stakeholders Change
- ☐ Requirement Identification
- ☐ Analyse Business Requirements
- ☐ Analyse Technical Requirements
- ☐ Design Cloud Solution
- ☐ Identify Incompatibilities
- ☐ Apply Design Principle
- ☐ Make Application Stateless
- ☐ Decouple Application Components
- ☐ Replicate Application Components
- ☐ Synchronise Application Components
- ☐ Handle Transient Faults
- ☐ Resolve Incompatibilities
- ☐ Migrate Data
- ☐ Refactor Codes
- ☐ Develop Integrator
- ☐ Encrypt/Decrypt Entities
- ☐ Encrypt Database
- ☐ Obfuscate Codes
- ☐ Encrypt/Decrypt Messages
- ☐ Isolate Tenants
- ☐ Enable Application Customisation
- ☐ Isolate Tenant Availability
- ☐ Isolate Tenant Performance

Tasks

- Plan Migration
- Recover Legacy Application Knowledge
- Analyse Context
- Analyse Migration Cost
- Analyse Migration Feasibility
- Analyse Network Change
- Analyse Organisational Changes
- Analyse Stakeholders Change
- Requirement Identification
- Analyse Business Requirements
- Analyse Technical Requirements
- Choose Cloud Provider
- Design Cloud Solution
- Identify Incompatibilities
- Apply Design Principle
- Make Application Stateless
- Decouple Application Components
- Replicate Application Components
- Synchronise Application Components
- Handle Transient Faults
- Resolve Incompatibilities
- Migrate Data
- Refactor Codes
- Develop Integrator
- Encrypt/Decrypt Entities
- Encrypt Database
- Obfuscate Codes
- Encrypt/Decrypt Messages
- Isolate Tenants
- Enable Application Customisation
- Isolate Tenant Availability
- Isolate Tenant Performance
- Isolate Tenante Data
- Rebalance Application Components
- Enable Elasticity

Tasks that should be performed after 'Choose Cloud Provider'

- ☐ Plan Migration
- ☐ Recover Legacy Application Knowledge
- ☐ Analyse Context
- ☐ Analyse Migration Cost
- ☐ Analyse Migration Feasibility
- ☐ Analyse Network Change
- ☐ Analyse Organisational Changes
- ☐ Analyse Stakeholders Change
- ☐ Requirement Identification
- ☐ Analyse Business Requirements
- ☐ Analyse Technical Requirements
- ☐ Design Cloud Solution
- ☐ Identify Incompatibilities
- ☐ Apply Design Principle
- ☐ Make Application Stateless
- ☐ Decouple Application Components
- ☐ Replicate Application Components
- ☐ Synchronise Application Components
- ☐ Handle Transient Faults
- ☐ Resolve Incompatibilities
- ☐ Migrate Data
- ☐ Refactor Codes
- ☐ Develop Integrator
- ☐ Encrypt/Decrypt Entities
- ☐ Encrypt Database
- ☐ Obfuscate Codes
- ☐ Encrypt/Decrypt Messages
- ☐ Isolate Tenants
- ☐ Enable Application Customisation
- ☐ Isolate Tenant Availability
- ☐ Isolate Tenant Performance

Figure 5 Specifying sequences among method elements, i.e. tasks