UCMIS M2M Transformation Project - Comprehensive Documentation

Overview

ucmism2m is a professional Eclipse-based Model-to-Model (M2M) transformation system built with QVTo (Query/View/Transformation Operational). It provides a complete framework for transforming UML models with JSON-based configuration, supporting both Eclipse IDE development and headless command-line execution for CI/CD integration.

Technology Stack

Component	Version	Purpose
Eclipse Platform	2025-12	Base runtime environment
QVTo	3.11.1	Transformation engine
UML2	5.0.0	UML metamodel support
Tycho	5.0.0	Maven/OSGi build system
Java	21 (LTS)	Runtime and black-box operations
org.json	20240303	JSON configuration parsing

Key Features

- **QVTo Transformations**: Model-to-model UML transformations with full QVTo 3.11.1 support
- **JSON Configuration**: External configuration files for runtime parameterization
- **Black-box Operations**: Java bridge for custom operations accessible from QVTo
- **Multi-platform**: Single build generates executables for Linux, Windows, and macOS
- **Headless Execution**: Pure command-line operation, no GUI dependencies
- **Eclipse IDE Integration**: Full development support in Eclipse Modeling Tools
- **CI/CD Ready**: Maven-based builds suitable for automation
- **Build Profiles**: Fast development builds (Linux-only) or complete releases (all platforms)

Project Structure

```
ucmism2m/
│
├── pom.xml                                  # Parent Maven POM (multi-module
coordinator)
├── README.md                                # Project documentation
├── .gitignore                               # Git ignore rules
│
├── ucmism2m.target/                         # Eclipse Target Platform Definition
│   ├── pom.xml                              # Maven configuration
│   ├── .project                             # Eclipse project file
│   └── ucmism2m.target.target               # Target platform definition (Eclipse
2025-12, QVTo 3.11.1)
│
```

```
├── ucmism2m.blackbox/                    # Black-box Java Operations
│   ├── pom.xml                           # Maven configuration with org.json
dependency
│   ├── .project                          # Eclipse project file
│   ├── .classpath                        # Eclipse classpath
│   ├── build.properties                  # Tycho build configuration
│   ├── META-INF/
│   │   └── MANIFEST.MF                   # OSGi bundle manifest
│   ├── plugin.xml                        # QVTo black-box unit registration
(namespace: blackboxlib)
│   └── src/
│       └── blackbox/
│           ├── JSONConfigLoader.java     # Java implementation (org.json-based)
│           └── JSONConfigLoaderLib.qvto # QVTo library wrapper
│
├── ucmism2m.transformation/              # QVTo Transformation Scripts
│   ├── pom.xml                           # Maven configuration
│   ├── .project                          # Eclipse project file
│   ├── .classpath                        # Eclipse classpath
│   ├── build.properties                  # Tycho build configuration
│   ├── META-INF/
│   │   └── MANIFEST.MF                   # OSGi bundle manifest
│   └── transforms/
│       └── m2m.qvto                      # Main transformation script (UML →
UML)
│
├── ucmism2m.feature/                     # Eclipse Feature (plugin grouping)
│   ├── pom.xml                           # Maven configuration
│   ├── .project                          # Eclipse project file
│   ├── feature.xml                       # Feature definition
│   └── build.properties                  # Build configuration
│
├── ucmism2m.app/                         # Headless Application Launcher
│   ├── pom.xml                           # Maven configuration
│   ├── .project                          # Eclipse project file
│   ├── .classpath                        # Eclipse classpath
│   ├── build.properties                  # Tycho build configuration
│   ├── META-INF/
│   │   └── MANIFEST.MF                   # OSGi bundle manifest
│   ├── plugin.xml                        # Application extension point
│   └── src/
│       └── ucmism2m/
│           └── app/
│               └── UCMISTransformationApp.java  # Main application class
│
├── ucmism2m.product/                     # Product Configuration (executable
export)
│   ├── pom.xml                           # Maven configuration with p2-director
plugin
│   ├── .project                          # Eclipse project file
│   ├── ucmism2m.product                  # Product definition (plugins,
launchers)
│   └── build.properties                  # Build configuration
│
├── examples/                             # Example configurations
│   └── config.json                       # Sample JSON configuration
│
├── scripts/                              # Build automation scripts
│   ├── build.sh                          # Standard build script
│   ├── build-all-platforms.sh           # Multi-platform build
│   ├── build-linux.sh                    # Linux-only build
│   ├── build-windows.sh                  # Windows-only build
│   ├── build-macos.sh                    # macOS-only build
│   └── run-transformation.sh             # Transformation runner
```

```
└── docs/                                  # Documentation
    ├── INSTALLATION.md                    # Installation guide
    ├── USAGE.md                           # Usage guide
    └── DEVELOPMENT.md                     # Development guide
```

Detailed Module Descriptions

1. ucmism2m.target - Target Platform Definition

Purpose: Defines the Eclipse platform and all dependencies required for the project.

Key File: `ucmism2m.target.target`

Contents:

- Eclipse 2025-12 platform features
- QVTo 3.11.1 SDK and runtime
- org.json library (from Maven Central, wrapped as OSGi bundle)
- UML2 5.0.0 metamodel
- EMF (Eclipse Modeling Framework) core

Usage: This must be set as the active target platform in Eclipse for proper dependency resolution.

---

2. ucmism2m.blackbox - Black-box Java Operations

Purpose: Provides Java-based operations accessible from QVTo transformations, specifically for JSON configuration loading.

Architecture:

```
Java Layer (JSONConfigLoader.java)
    ↓ exports operations
QVTo Library (JSONConfigLoaderLib.qvto)
    ↓ imports via namespace
QVTo Transformation (m2m.qvto)
```

Key Components:

`JSONConfigLoader.java` - Java implementation with methods:

- `loadConfig(String jsonPath)` - Loads complete JSON as Map
- `getStringValue(String jsonPath, String key)` - Retrieves string value
- `getStringList(String jsonPath, String key)` - Retrieves string array
- `getBooleanValue(String jsonPath, String key, Boolean defaultValue)` - Retrieves boolean
- `hasKey(String jsonPath, String key)` - Checks key existence

`JSONConfigLoaderLib.qvto` - QVTo library wrapper:

- Imports Java operations using `implements` keyword

- Namespace: `blackboxlib`
- Makes Java methods callable from QVTo

`plugin.xml` - Registers the black-box unit with QVTo:

```
<unit name="JSONConfigLoaderLib" namespace="blackboxlib">
    <library class="blackbox.JSONConfigLoader" name="JSONConfigLoaderLib"/>
</unit>
```

Dependencies:

- `org.json:json:20240303` (Maven dependency)
- `org.eclipse.m2m.qvt.oml` (QVTo runtime)
- `org.eclipse.emf.ecore` (EMF core)

---

3. ucmism2m.transformation - QVTo Transformations

Purpose: Contains the actual model-to-model transformation logic written in QVTo.

Main File: `transforms/m2m.qvto`

Transformation Structure:

```
import blackbox.JSONConfigLoaderLib;

modeltype UML uses 'http://www.eclipse.org/uml2/5.0.0/UML';

transformation m2m(in inModel : UML, out outModel : UML) {
    configuration property configPath : String;

    main() {
        // Load JSON configuration
        var config := JSONConfigLoaderLib::loadConfig(configPath);

        // Transform model elements
        inModel.rootObjects()[Package]->map transformPackage(config);
    }

    // Mapping operations for UML elements
    mapping Package::transformPackage(config) : Package { ... }
    mapping Class::transformClass(config) : Class { ... }
    mapping Property::transformProperty(config) : Property { ... }
    // ... additional mappings
}
```

Features:

- Configuration-driven transformation logic
- Selective element mapping based on JSON criteria
- Preservation of UML semantic relationships
- Support for:
    - Packages
    - Classes (with attributes, operations)
    - DataTypes
    - Associations

- Generalizations (inheritance)
- Properties with multiplicities
- Operations with parameters

Extensibility: Add new mapping operations as helper functions or additional mapping rules.

---

4. ucmism2m.feature - Eclipse Feature

Purpose: Groups related plugins into a deployable unit for Eclipse.

Key File: `feature.xml`

Includes:

- `ucmism2m.blackbox`
- `ucmism2m.transformation`
- `ucmism2m.app`

Note: This is primarily for Eclipse-based distribution. The product configuration (ucmism2m.product) uses a plugin-based approach for better control.

---

5. ucmism2m.app - Headless Application

Purpose: Provides the entry point for command-line execution of transformations outside Eclipse IDE.

Main Class: `UCMISTransformationApp.java`

Functionality:

1. **Command-line argument parsing**:
   - `-input <path>` - Input UML model file
   - `-output <path>` - Output UML model file
   - `-config <path>` - JSON configuration file
2. **Initialization**:
   - Registers UML2 resource factory
   - Sets up EMF resource set
   - Loads input model
3. **Transformation execution**:
   - Loads QVTo transformation from plugin
   - Sets configuration property (config file path)
   - Executes transformation with QVTo engine
   - Handles diagnostics and error reporting
4. **Output generation**:
   - Saves transformed model
   - Resolves proxies

- Applies save options

Key Features:

- Completely headless (no GUI)
- Proper error handling and diagnostics
- Platform URI resolution for plugin resources
- EMF/UML2 resource management

Example Implementation Pattern:

```
// Load transformation
URI transformationURI = URI.createURI(
    "platform:/plugin/ucmism2m.transformation/transforms/m2m.qvto");
TransformationExecutor executor = new TransformationExecutor(transformationURI);

// Set up context
ExecutionContextImpl context = new ExecutionContextImpl();
context.setConfigProperty("configPath", configPath);

// Execute
ExecutionDiagnostic result = executor.execute(context, inputExtent,
outputExtent);
```

---

6. ucmism2m.product - Product Configuration

Purpose: Defines the complete executable product with all required plugins and configurations.

Key File: `ucmism2m.product`

Configuration:

- **Application**: `ucmism2m.app.ucmism2m`
- **Product Name**: UCMIS M2M Transformation
- **Launcher Name**: `ucmism2m`
- **VM Arguments**:

  ```
  -Xms256m-Xmx1024m-Dosgi.requiredJavaVersion=21
  ```

Included Plugins (plugin-based approach):

- Core application plugins (ucmism2m.*)
- Eclipse runtime (org.eclipse.core.runtime, org.eclipse.equinox.*)
- QVTo runtime (org.eclipse.m2m.qvt.oml.*)
- EMF core (org.eclipse.emf.*)
- UML2 (org.eclipse.uml2.*)

Build Output: Creates platform-specific executables in:

```
target/products/ucmism2m/
├── linux/gtk/x86_64/ucmism2m
├── win32/win32/x86_64/ucmism2m.exe
├── macosx/cocoa/x86_64/ucmism2m.app
└── macosx/cocoa/aarch64/ucmism2m.app
```

Distribution Archives:

```
target/products/
├── ucmism2m-linux.gtk.x86_64.tar.gz
├── ucmism2m-win32.win32.x86_64.zip
├── ucmism2m-macosx.cocoa.x86_64.tar.gz
└── ucmism2m-macosx.cocoa.aarch64.tar.gz
```

Configuration Files

JSON Configuration Format

Location: External file passed via `-config` argument

Example (`examples/config.json`):

```
{
  "transformationName": "UCMIS M2M Transformation",
  "version": "1.0.0",
  "excludedElements": [
    "TemporaryClass",
    "DebugInfo"
  ],
  "mappingRules": {
    "preserveComments": true,
    "flattenInheritance": false,
    "includeAbstractClasses": true
  },
  "datatypeMapping": {
    "String": "Text",
    "Integer": "Number",
    "Boolean": "Flag"
  },
  "options": {
    "verbose": true,
    "validateOutput": true
  }
}
```

Access from QVTo:

```
var config := JSONConfigLoaderLib::loadConfig(configPath);
var excludedList := config->get('excludedElements');
var preserveComments := JSONConfigLoaderLib::getBooleanValue(configPath,
'preserveComments', true);
```

Build System

Maven Profiles

The project uses Maven profiles for flexible build configurations:

1. linux-only (Default)

```
mvn clean verify
# OR explicitly
mvn clean verify -P linux-only
```

- **Purpose**: Fast development builds
- **Output**: Linux x86_64 executable only
- **Build Time**: ~2-3 minutes
- **Use Case**: Development, testing, Linux-only deployments

2. all-platforms

```
mvn clean verify -P all-platforms
```

- **Purpose**: Complete release builds
- **Output**: Linux, Windows, macOS (Intel + Apple Silicon) executables
- **Build Time**: ~5-8 minutes
- **Use Case**: Releases, multi-platform distribution

3. windows-only

```
mvn clean verify -P windows-only
```

- **Purpose**: Windows-specific builds
- **Output**: Windows x86_64 executable only

4. macos-only

```
mvn clean verify -P macos-only
```

- **Purpose**: macOS-specific builds
- **Output**: macOS Intel and Apple Silicon executables

Build Configuration

Parent POM (`pom.xml`):

- Tycho 5.0 configuration
- Java 21 compiler settings
- Target platform reference
- Profile definitions
- m2e lifecycle mapping (Eclipse-only)

Tycho Plugins:

- `tycho-maven-plugin` - Core Tycho functionality
- `tycho-compiler-plugin` - Java compilation with OSGi
- `target-platform-configuration` - Platform and environment setup
- `tycho-p2-director-plugin` - Product materialization and archiving

Usage

Command-Line Execution

Basic Syntax:

```
./ucmism2m -input <input.uml> -output <output.uml> -config <config.json>
```

Example:

```
cd ucmism2m.product/target/products/ucmism2m/linux/gtk/x86_64/

./ucmism2m \
  -input /home/user/models/source.uml \
  -output /home/user/models/target.uml \
  -config /home/user/configs/transformation.json
```

Output:

```
UCMIS M2M Transformation Application
Eclipse 2025-12 | QVTo 3.11.1 | Java 21
====================================
Input model: /home/user/models/source.uml
Output model: /home/user/models/target.uml
Configuration: /home/user/configs/transformation.json

Loading input model...
Input model loaded: 5 root elements
Loading transformation...
Transformation loaded successfully

Executing transformation...
UCMIS M2M Transformation started
Configuration file: /home/user/configs/transformation.json
Configuration loaded successfully
Transforming package: MyModel
Transforming class: Person
  Transforming property: name
  Transforming property: age
...
UCMIS M2M Transformation completed
Transformation executed successfully

Saving output model...
Output model saved: /home/user/models/target.uml

Transformation completed successfully!
```

Exit Codes

Code	Meaning
0	Success
1	Transformation failed or error occurred

Eclipse IDE Usage

Import Projects

1. **File → Import → Maven → Existing Maven Projects**
2. **Browse to**: `ucmism2m` directory
3. **Select all 6 projects**:
    - ucmism2m.target

- ucmism2m.blackbox
- ucmism2m.transformation
- ucmism2m.feature
- ucmism2m.app
- ucmism2m.product

4. **Click Finish**

Set Target Platform

Critical Step - Must be done after import:

1. **Window → Preferences**
2. **Plug-in Development → Target Platform**
3. **Select**: `ucmism2m.target`
4. **Click**: "Reload" (if needed)
5. **Click**: "Apply and Close"
6. **Wait** for dependency resolution (2-5 minutes)

Run QVTo Transformation in Eclipse

1. **Open**: `ucmism2m.transformation/transforms/m2m.qvto`
2. **Right-click → Run As → QVTo Transformation**
3. **Configure Run Configuration**:
    - **Transformation**: Browse to m2m.qvto
    - **Input Models**: Add UML model (ModelType: UML)
    - **Output Models**: Specify output location
    - **Configuration Properties**: Add `configPath` = path to JSON
4. **Click Run**

Test Application in Eclipse

1. **Open**: `ucmism2m.product/ucmism2m.product`
2. **Click**: "Launch an Eclipse application" (or "Debug")
3. **OR Create Run Configuration**:
    - **Run → Run Configurations → Eclipse Application**
    - **Program Arguments**: `-input ... -output ... -config ...`

Maven Build from Eclipse

1. **Right-click** parent project (`ucmism2m`)
2. **Run As → Maven build...**
3. **Goals**: `clean verify`
4. **Profiles**: `linux-only` or `all-platforms`
5. **Click Run**

Development Workflow

Adding a New QVTo Transformation

1. **Open**: `ucmism2m.transformation/transforms/m2m.qvto`
2. **Add mapping operation**:

```
mapping MyElement::transformMyElement(config : Dict(String, OclAny)) : MyElement
{
    name := self.name;
    // transformation logic
}
```

3. **Call from main or other mapping**:

```
self.myElements->map transformMyElement(config);
```

4. **Test in Eclipse** (Run As → QVTo Transformation)
5. **Build**: `mvn clean verify`

Adding a New Black-box Operation

1. **Edit**: `ucmism2m.blackbox/src/blackbox/JSONConfigLoader.java`
2. **Add Java method**:

```
@Operation(contextual = false)
public static Integer getIntegerValue(String jsonPath, String key, Integer
defaultValue) {
    // implementation
}
```

3. **Edit**: `ucmism2m.blackbox/src/blackbox/JSONConfigLoaderLib.qvto`
4. **Add QVTo query**:

```
query getIntegerValue(jsonPath : String, key : String, defaultValue : Integer) :
Integer
    implements blackboxlib::JSONConfigLoaderLib::getIntegerValue;
```

5. **Use in transformation**:

```
var maxItems := JSONConfigLoaderLib::getIntegerValue(configPath, 'maxItems',
100);
```

6. **Build**: `mvn clean verify`

Modifying Application Behavior

1. **Edit**: `ucmism2m.app/src/ucmism2m/app/UCMISTransformationApp.java`
2. **Make changes** (e.g., add new command-line arguments, change output format)
3. **Test in Eclipse** (Run As → Eclipse Application)
4. **Build**: `mvn clean verify`

CI/CD Integration

Jenkins Pipeline Example

```
pipeline {
    agent any

    tools {
        jdk 'JDK-21'
        maven 'Maven-3.9'
    }

    stages {
        stage('Checkout') {
            steps {
                git 'https://github.com/your-org/ucmism2m.git'
            }
        }

        stage('Build') {
            steps {
                sh 'mvn clean verify -P all-platforms'
            }
        }

        stage('Archive') {
            steps {
                archiveArtifacts artifacts:
'ucmism2m.product/target/products/*.tar.gz,ucmism2m.product/target/products/
*.zip'
            }
        }

        stage('Test') {
            steps {
                sh '''
                    cd
ucmism2m.product/target/products/ucmism2m/linux/gtk/x86_64
                    ./ucmism2m -input test/input.uml -output test/output.uml -
config test/config.json
                '''
            }
        }
    }
}
```

GitLab CI Example

```
image: maven:3.9-eclipse-temurin-21

stages:
  - build
  - test
  - package

build:
  stage: build
  script:
    - mvn clean verify -P all-platforms
  artifacts:
    paths:
      - ucmism2m.product/target/products/
```

```
test:
  stage: test
  script:
    - cd ucmism2m.product/target/products/ucmism2m/linux/gtk/x86_64
    - ./ucmism2m -input $CI_PROJECT_DIR/test/input.uml -output /tmp/output.uml -
config $CI_PROJECT_DIR/test/config.json

package:
  stage: package
  script:
    - cd ucmism2m.product/target/products
    - ls -lh *.tar.gz *.zip
  artifacts:
    paths:
      - ucmism2m.product/target/products/*.tar.gz
      - ucmism2m.product/target/products/*.zip
```

GitHub Actions Example

```
name: Build and Test

on: [push, pull_request]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4

    - name: Set up JDK 21
      uses: actions/setup-java@v4
      with:
        java-version: '21'
        distribution: 'temurin'

    - name: Build with Maven
      run: mvn clean verify -P all-platforms

    - name: Test Transformation
      run: |
        cd ucmism2m.product/target/products/ucmism2m/linux/gtk/x86_64
        ./ucmism2m -input $GITHUB_WORKSPACE/test/input.uml -output
/tmp/output.uml -config $GITHUB_WORKSPACE/test/config.json

    - name: Upload Artifacts
      uses: actions/upload-artifact@v4
      with:
        name: ucmism2m-executables
        path: ucmism2m.product/target/products/*.tar.gz
```

Troubleshooting

Common Issues

Build Fails with "Target Platform Resolution Error"

Symptom: Cannot resolve dependencies from Eclipse repositories

Solution:

1. Check internet connectivity
2. Verify Eclipse 2025-12 repository is accessible
3. Try: `mvn clean verify -U` (force update)
4. Check proxy settings in Maven `settings.xml`

Eclipse Shows "org.json package not found"

Symptom: MANIFEST.MF shows error on `Import-Package: org.json`

Solution:

1. Ensure target platform is set: Window → Preferences → Plug-in Development → Target Platform → Select `ucmism2m.target`
2. Maven → Update Project (all projects, force update)
3. Project → Clean → Clean all projects

QVTo Transformation Shows Compilation Errors

Symptom: m2m.qvto shows red errors in Eclipse

Solution:

1. Verify QVTo plugins installed: Help → About → Installation Details
2. Check import statement: `import blackbox.JSONConfigLoaderLib;`
3. Ensure blackbox project has no errors
4. Clean and rebuild workspace

Executable Fails with "Class Not Found"

Symptom: Running executable shows ClassNotFoundException

Solution:

1. Rebuild with: `mvn clean verify -P all-platforms`
2. Check `ucmism2m.product` includes all required plugins
3. Verify MANIFEST.MF dependencies are correct

Transformation Fails with "Configuration file not found"

Symptom: Runtime error about missing JSON file

Solution:

1. Verify config file path is absolute or relative to execution directory
2. Check file permissions (readable)

3. Validate JSON syntax with online validator

Performance Considerations

Build Performance

Build Type	Duration	Output Size
Linux-only	~2-3 min	~150 MB
All platforms	~5-8 min	~600 MB

Optimization Tips:

- Use `linux-only` profile for development
- Use `-o` (offline) flag if dependencies cached: `mvn clean verify -o`
- Increase Maven memory: `export MAVEN_OPTS="-Xmx2048m"`

Runtime Performance

Typical Transformation:

- Small model (< 100 elements): < 1 second
- Medium model (100-1000 elements): 1-5 seconds
- Large model (> 1000 elements): 5-30 seconds

Memory Usage:

- Default: 256 MB min, 1024 MB max
- Adjust in `ucmism2m.product` VM arguments if needed

Extension Points

Custom Black-box Operations

The architecture supports adding custom black-box operations for:

- Database access
- Web service calls
- File system operations
- External tool integration
- Custom algorithms

Pattern:

1. Add Java method to `JSONConfigLoader.java` or new class
2. Annotate with `@Operation`
3. Register in `plugin.xml`
4. Expose via QVTo library
5. Use in transformations

Additional Transformations

The system can be extended with:

- Multiple transformation scripts
- Chained transformations
- Bidirectional transformations
- Model validation rules

Alternative Input/Output Formats

With modifications, support additional formats:

- XMI models
- Ecore models
- Custom DSL models
- Database schemas

Version Information

Current Version: 1.0.0-SNAPSHOT

Compatibility:

- Eclipse: 2025-12 (may work with 2024-12)
- Java: 21+ required
- QVTo: 3.11.1
- UML2: 5.0.0
- Tycho: 5.0.0

Tested Platforms:

- Linux (Ubuntu 22.04, x86_64)
- Windows (10/11, x86_64)
- macOS (Ventura+, Intel)
- macOS (Ventura+, Apple Silicon)

License

[Specify your license here - e.g., Apache 2.0, MIT, GPL, proprietary]

Support and Contact

Project Repository: [Your repository URL] **Issue Tracker**: [Your issue tracker URL]
Documentation: [Your documentation URL] **Contact**: [Your contact information]

Acknowledgments

Built with:

- Eclipse Modeling Framework (EMF)
- Eclipse QVTo
- Eclipse Tycho
- Apache Maven
- org.json library

This comprehensive documentation provides everything needed to understand, build, use, and extend the ucmism2m transformation system.