

General Introduction to the use of X3ML toolkit

G. Bruseker
ICS-FORTH

Workshop Content

1. X3ML Design Principles
2. Basic Interface Operation
3. Setting up a New Mapping
4. Matching Table Operation
5. Mapping Patterns
6. Instance Generators

1. X3ML DESIGN PRINCIPLES

X3ML: A Mapping Language

- **X3ML** is a declarative, XML based language which describes schema mappings in such a way that they can be collaboratively created and discussed by experts.
- Mappings have been done in very many custom ways in the past.
- In practice mappings are produced manually by Domain/IT experts:
 - labor-intensive
 - error prone
 - time consuming
- Emphasis is on establishing a **standardized mapping description** which lends itself to collaboration and the building of a **mapping memory** to accumulate knowledge and experience.

X3ML toolkit

- the **X3ML Toolkit** is a set of small, open source, microservices that follow the SYNERGY Reference Model. They are designed with open interfaces and they can be easily customized and adapted to complex environments. The key components of the toolkit are:
 - Mapping Memory Manager
 - 3M Editor
 - X3ML Engine
- FORTH's open access service is found at:
<https://www.ics.forth.gr/isl/3M>

3M : Mapping Memory Manager

- **3M** is a tool for managing mapping definition files. It's based on [FIMS](#) management system for the administration of the files and also on the [3MEditor](#) for editing and viewing the files. It provides a number of administrative actions that assist the experts to manage their mapping definition files.
- The source code is open source available on github <http://github.com/isl/Mapping-Memory-Manager>

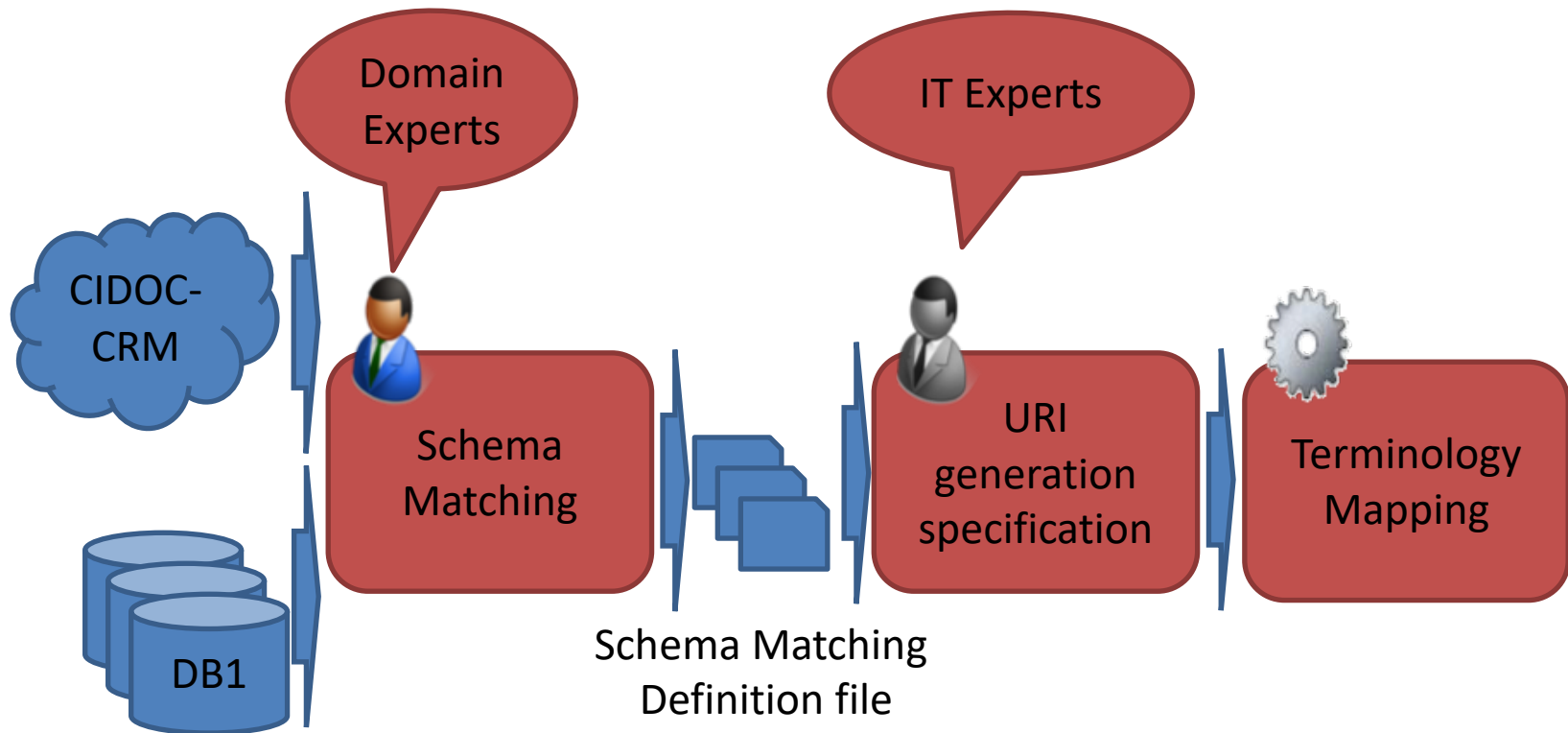
3M Editor: A Mapping Editor

- **3MEditor** is a software that allows domain experts to build and discuss mappings with little resource to any particular software skills. It is the interface tool envisioned to allow domain experts to build mappings.
- It provides:
 - Source and target agnostic mapping facility
 - Guided mapping according to deployed ontology's logic
 - Comment and justification facility
 - Mapping storage
 - Separated instance generation practice for IT professionals
- The source code is open source available on github <https://github.com/isl/3MEditor>

X3ML engine: A Transformation Tool

- The **X3ML Engine** realizes the transformation of the source records to the target format. The engine takes as input the source data (currently in the form of an XML document), the description of the mappings in the X3ML mapping definition file and the URI generation policy file and is responsible for transforming the source document into a valid RDF document which corresponds to the input XML file, with respect to the given mappings and policy.
- The source code is open source available on github <https://github.com/isl/x3ml>

X3ML Workflow



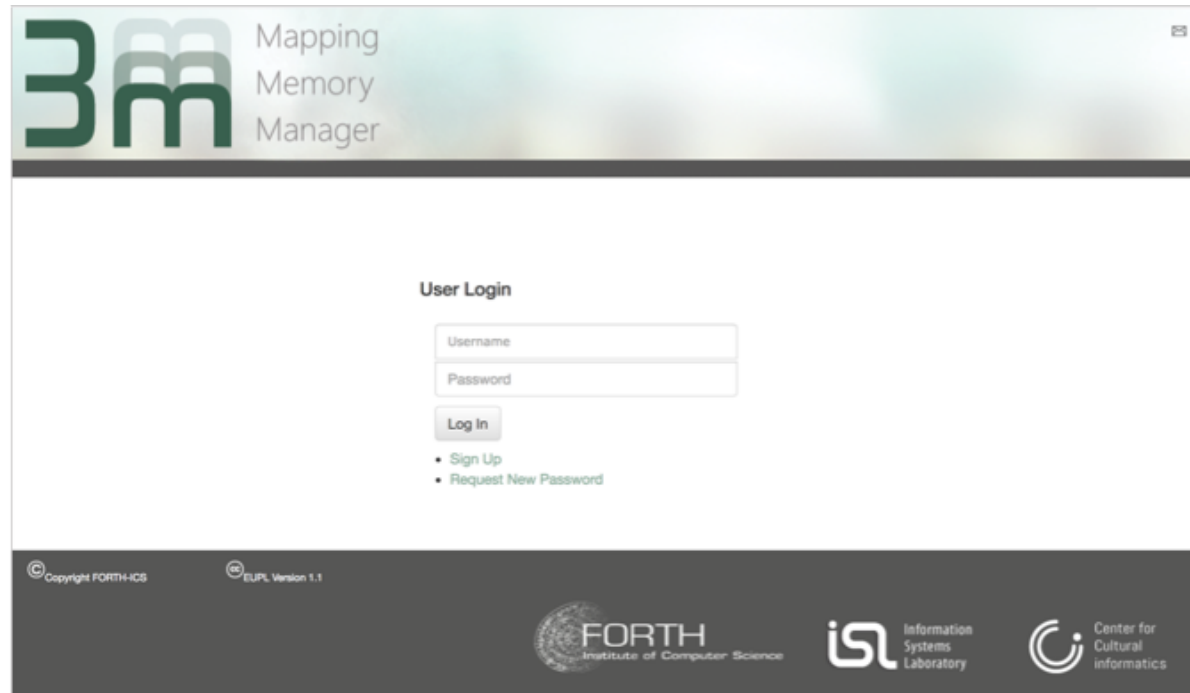
2. BASIC INTERFACE OPERATION

Basic Interface Operation

Section Content

1. Logging In / New User
2. Control Tools and List
3. 'More' Feature in Control Tools
(e.g. Copy / Give Rights)
4. Search / Filter options
5. Manual

1. Login



The screenshot shows the 'Mapping Memory Manager' (3M) login interface. At the top left is the '3M' logo and the text 'Mapping Memory Manager'. The main content area is titled 'User Login' and contains a 'Username' input field, a 'Password' input field, and a 'Log In' button. Below the button are two links: 'Sign Up' and 'Request New Password'. The footer contains copyright information for FORTH-ICS and EUPL Version 1.1, along with logos for FORTH Institute of Computer Science, Information Systems Laboratory (isl), and the Center for Cultural Informatics.

- Create New User
- Reset Password

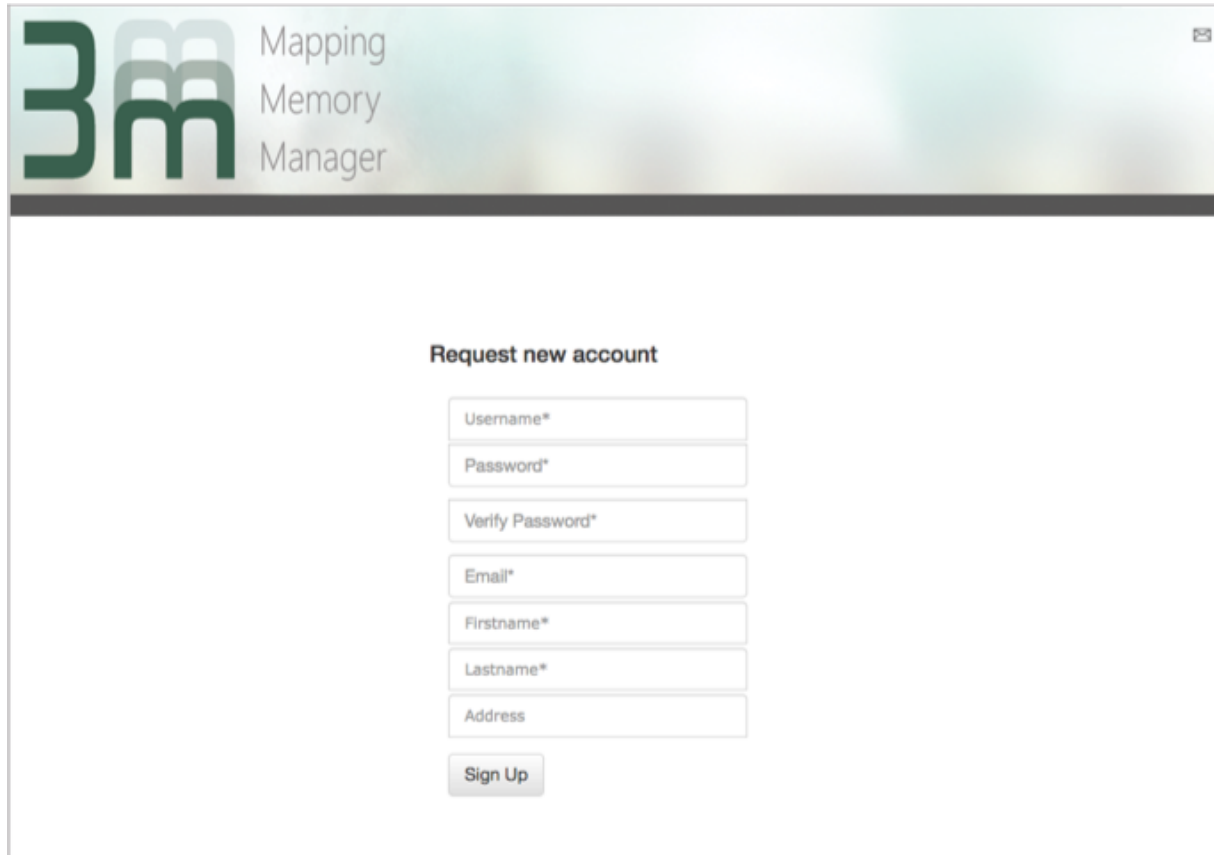
OR

- Login

Login at:

<https://www.ics.forth.gr/isl/3M>

1. New User



The screenshot shows the 'Mapping Memory Manager' web application. The header features a logo with the letters '3m' in green and grey, followed by the text 'Mapping Memory Manager' and a small envelope icon. The main content area is titled 'Request new account' and contains a registration form with the following fields: 'Username*', 'Password*', 'Verify Password*', 'Email*', 'Firstname*', 'Lastname*', and 'Address'. A 'Sign Up' button is located at the bottom of the form.

Mapping
Memory
Manager

Request new account

Username*

Password*

Verify Password*

Email*

Firstname*

Lastname*

Address

Sign Up

- Only need a valid email to sign up!
- System will send a verification email and you can begin

2. Control Tools and List

New View Edit

Mapping Memory Manager

George Bruseker

Main Menu

Mappings

Help

Quick Start Guide

Manual

Mappings

Showing: All

Filter Table

Showing 10 entries

Title	Source Schema	Target Schema	General Description	Creator	Card Status	Id	
LIDO to CIDOC (version 2.0)	LIDO	CIDOC-CRM	The mapping was implemented within the ATHENA project (www.athenaeurope.org). The ATHENA project is bringing together re...	admin	Unpublished	Mapping/1	

Control Tools

Mapping List

- Tools for working with maps located in top Control Tools bar
- to view or edit a map, first select it in Mapping List, then click the appropriate icon

3. Control Tools 'More'

The screenshot shows the Mapping Memory Manager interface. The 'More' dropdown menu is open, displaying various options. The 'Mappings' table is visible below the menu, showing a mapping from LIDO to CIDOC-CRM. The 'More' dropdown menu options are: Publish, Import/Export, Versions, Delete, Unlock File, Copy XML, Generators, Analysis, Compare, and Rights. The 'Copy XML' and 'Rights' options are highlighted with red boxes. An arrow points from the 'More' dropdown menu to the 'Copy XML' option.

Main Menu

- Mappings
- Help
 - Quick Start Guide
 - Manual

Mappings

Showing: All

Filter Table

Showing 10 entries

Title	Source Schema	Target Schema	General Description	Creator	Card Status	Id	
LIDO to CIDOC (version 2.0)	LIDO	CIDOC-CRM	The mapping was implemented within the ATHENA project (www.athena-europe.org). The ATHENA project is bringing together re...	admin	Unpublished	Mapping/1	🔒

Publish

- Request for publishing

Import/Export

- Export Template
- Export to XML
- Export Schema
- Import from XML

Versions

- Create Version
- View Versions

Delete

Unlock File

Copy XML

Generators

Analysis

Compare

Rights

- 'More' dropdown menu gives many options for working on a mapping.
- 'Copy XML' allows you to take a full copy of the selected mapping (useful when not sure of changes)
- 'Rights' allows you to share edit rights to other users.

4. Search and Filter

Filter

Search

The screenshot displays the Mapping Memory Manager web application. At the top, the header includes the '3m' logo and the text 'Mapping Memory Manager'. A user profile 'George Bruseker' is visible in the top right. On the left, a 'Main Menu' sidebar lists 'Mappings', 'Help', 'Quick Start Guide', and 'Manual'. The main content area is titled 'Mappings' and shows 'Showing All'. A search bar with a magnifying glass icon and the word 'Search' is highlighted with a red box and an arrow labeled 'Search'. Below the search bar, a 'Filter Table' input field is highlighted with a red box and an arrow labeled 'Filter'. The table below lists mappings with columns: Title, Source Schema, Target Schema, General Description, Creator, Card Status, Id, and an action icon. The first entry is 'LIDO to CIDOC (version 2.0)' with source 'LIDO' and target 'CIDOC-CRM'. The table is set to 'Showing 10 entries'.

Title	Source Schema	Target Schema	General Description	Creator	Card Status	Id	
LIDO to CIDOC (version 2.0)	LIDO	CIDOC-CRM	The mapping was implemented within the ATHENA project (www.athenaeurope.org). The ATHENA project is bringing together re...	admin	Unpublished	Mapping/1	🔒

- There is a search feature allowing you to find maps
- There is a filter feature to narrow down within a viewed search set

5. Manual

Manual

The screenshot shows the 3M Mapping Memory Manager interface. The sidebar on the left has a 'Main Menu' section with 'Mappings', 'Help', 'Quick Start Guide', and 'Manual'. The 'Manual' item is highlighted with a red box. An arrow points from the 'Manual' text above to this box. The main content area shows a 'Mappings' section with a search bar, a table of mappings, and a filter table.

Title	Source Schema	Target Schema	General Description	Creator	Card Status	Id
LIDO to CIDOC (version 2.0)	LIDO	CIDOC-CRM	The mapping was implemented within the ATHENA project (www.athenaeurope.org). The ATHENA project is bringing together re...	admin	Unpublished	Mapping/1

- The complete documentation for 3M including general mapping instructions and advice is always accessible directly from the sidebar.

3. SETTING UP A NEW MAPPING

Setting up New Mapping

Section Content

1. Give Title and choose Target Schemas
2. Edit/View Info Tab
3. Add Generic Mapping Metadata
4. Adding Source Schema
5. Adding Sample Data
6. Adding Target Schemas

1. Creating a New Map: Title & Targets

Mapping Memory Manager

George Bruseker

Main Menu

Mappings

Help

Quick Start Guide

Manual

Mappings - Create New

Title:

Target Schema:

CIDOC-CRM 6.0

CRMdig 3.2

CRMgeo 1.2

CRMsci 1.2.2

CRMarchaeo 1.2.1

FRBR 2.1

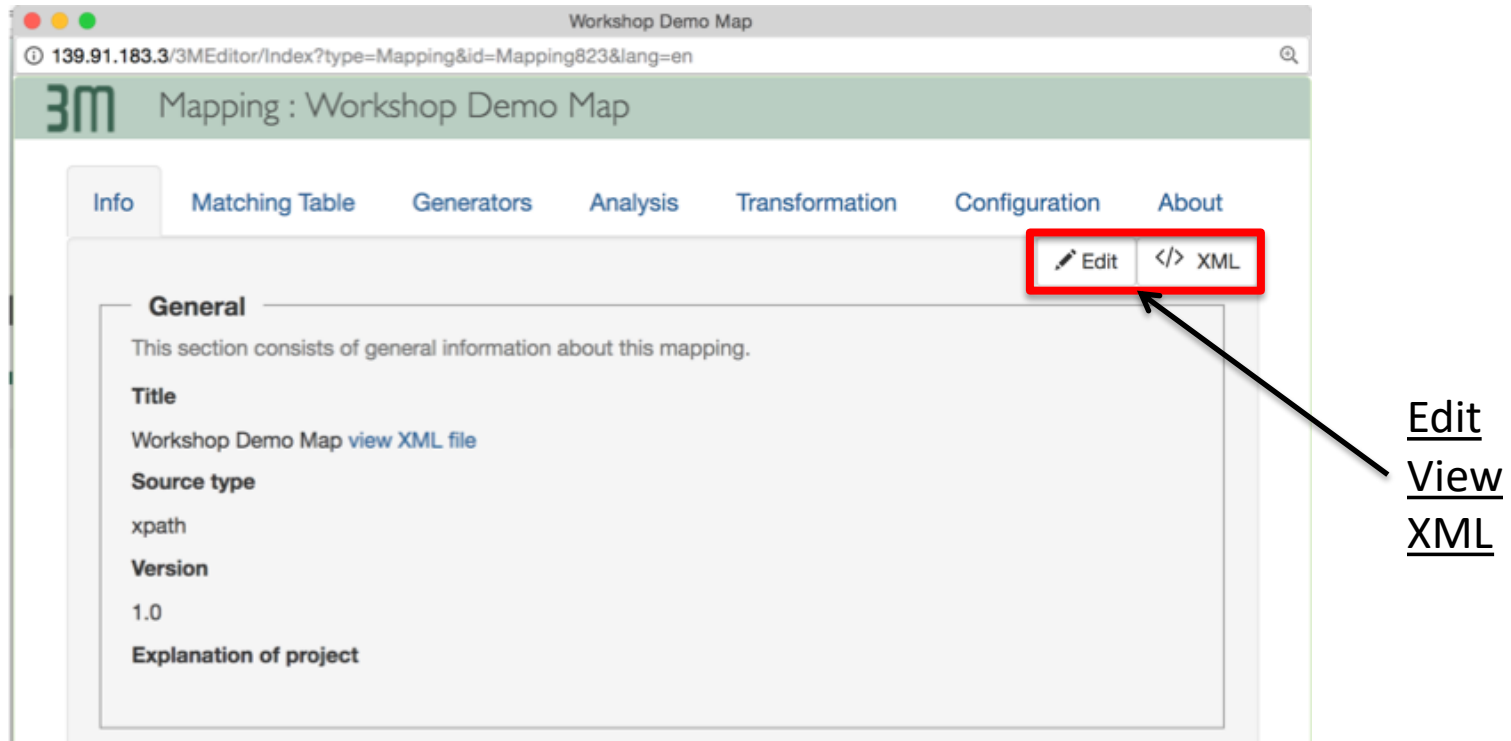
Copyright FORTH-ICS

EUPL Version 1.1

Creating a new mapping involves:

1. Click the '+' button (Create New) in 3M to open a create new map dialogue
2. Assign a title so it can be found again
3. Choose target schema(s) you wish to use
4. Finish the creation of the new map
5. Once map saved, find it again from list and open using 'edit function' (see above)

2. Filling the Info Tab: Edit/View/XML Modes



- To edit the mapping, you will need to enter edit mode. This can be done by choosing the 'Edit' button.
- To leave edit mode, you can choose the opposite 'View' button. Experts can also edit the XML directly by clicking the 'XML' button.

3. Adding Generic Mapping Metadata

Mapping

This section consists of information about who creates and supports this mapping.

Created by (Organization)

Contact person(s)

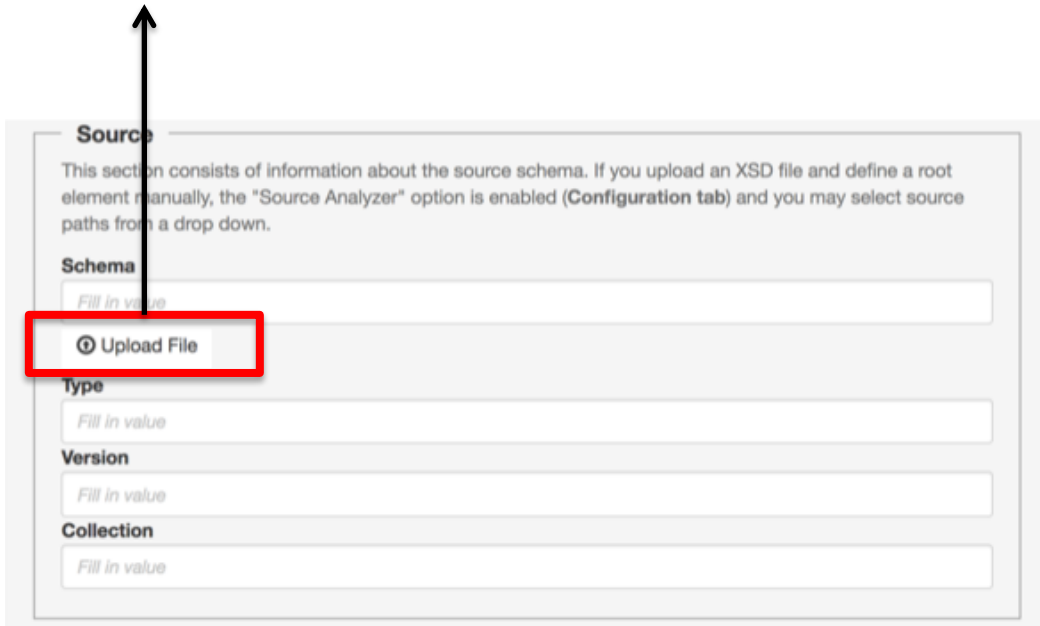
In collaboration with

Mapping Metadata provides crucial provenance information to the mapping so that it can be re-used in the future.

Who did this mapping? How can they be contacted?

4. Adding Source Schema

Upload Source Schema



The screenshot shows a web form titled 'Source'. It contains a text area with instructions, followed by input fields for 'Schema', 'Type', 'Version', and 'Collection'. The 'Schema' field has a red box around the 'Upload File' button, and an arrow points from this button to the 'Upload Source Schema' header.

Source

This section consists of information about the source schema. If you upload an XSD file and define a root element manually, the "Source Analyzer" option is enabled (**Configuration tab**) and you may select source paths from a drop down.

Schema

Fill in value

Upload File

Type

Fill in value

Version

Fill in value

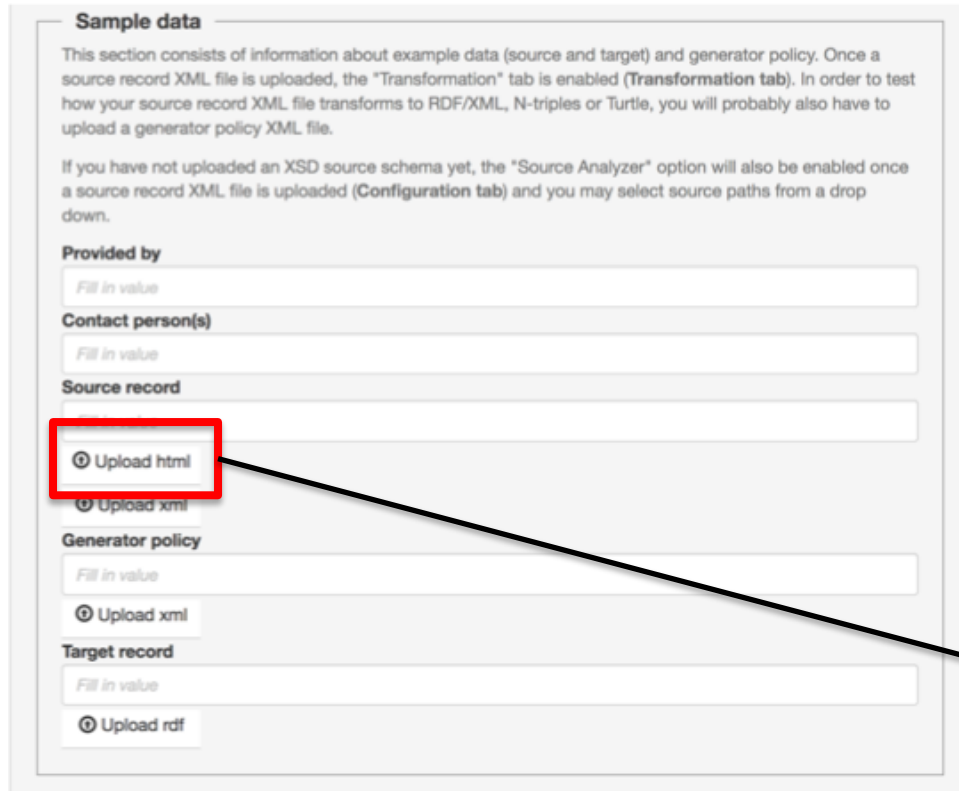
Collection

Fill in value

- Upload a source schema using 'upload button'
- Source can be in XML or XSD.
- Additional metadata can be added for
 - the type of schema
 - the version of the schema
 - what collection schema used for

Adding source schema allows 3M to help guide mapping by only allowing valid paths to be selected from source.

5. Adding Source Schema Sample Data



Sample data

This section consists of information about example data (source and target) and generator policy. Once a source record XML file is uploaded, the "Transformation" tab is enabled (**Transformation tab**). In order to test how your source record XML file transforms to RDF/XML, N-triples or Turtle, you will probably also have to upload a generator policy XML file.

If you have not uploaded an XSD source schema yet, the "Source Analyzer" option will also be enabled once a source record XML file is uploaded (**Configuration tab**) and you may select source paths from a drop down.

Provided by

Contact person(s)

Source record

Generator policy

Target record

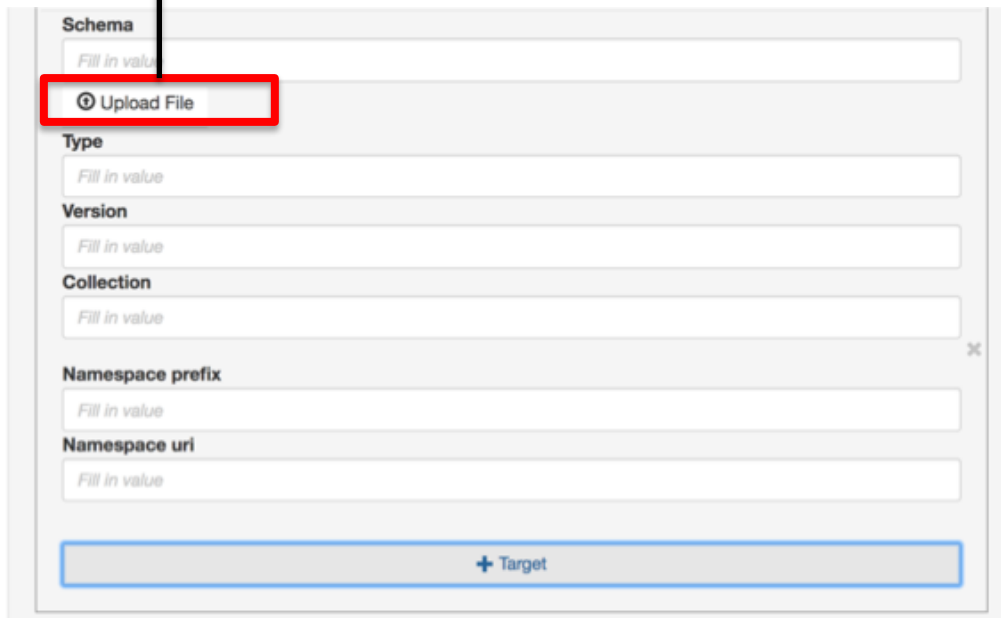
- Upload Source Data by choosing 'Upload xml' button
- Add provenance information of who gave data and how they can be contacted.

[Upload Sample Source Data](#)

Adding source sample data allows testing of the mapping with real data using the 3M transformation tool.

Upload Target Schema

6. Adding Target Schema



The screenshot shows a form titled 'Add Target' with several input fields and a button. An arrow points from the 'Upload File' button to the 'Upload Target Schema' header. The form fields are:

- Schema**: Fill in value
- Type**: Fill in value
- Version**: Fill in value
- Collection**: Fill in value
- Namespace prefix**: Fill in value
- Namespace uri**: Fill in value

At the bottom of the form is a button labeled '+ Target'.

- Click '+Target' to add new target
- Upload target schema using 'upload' button
- For each target schema specify
 - it's namespace prefix e.g.: 'crm'
 - It's full namespace URI
- Additional metadata can be added for
 - the name of the schema
 - the type of schema
 - the version of the schema
 - what collection schema used for

Adding target schema allows 3M to help guide mapping by only allowing valid paths to be selected from target.

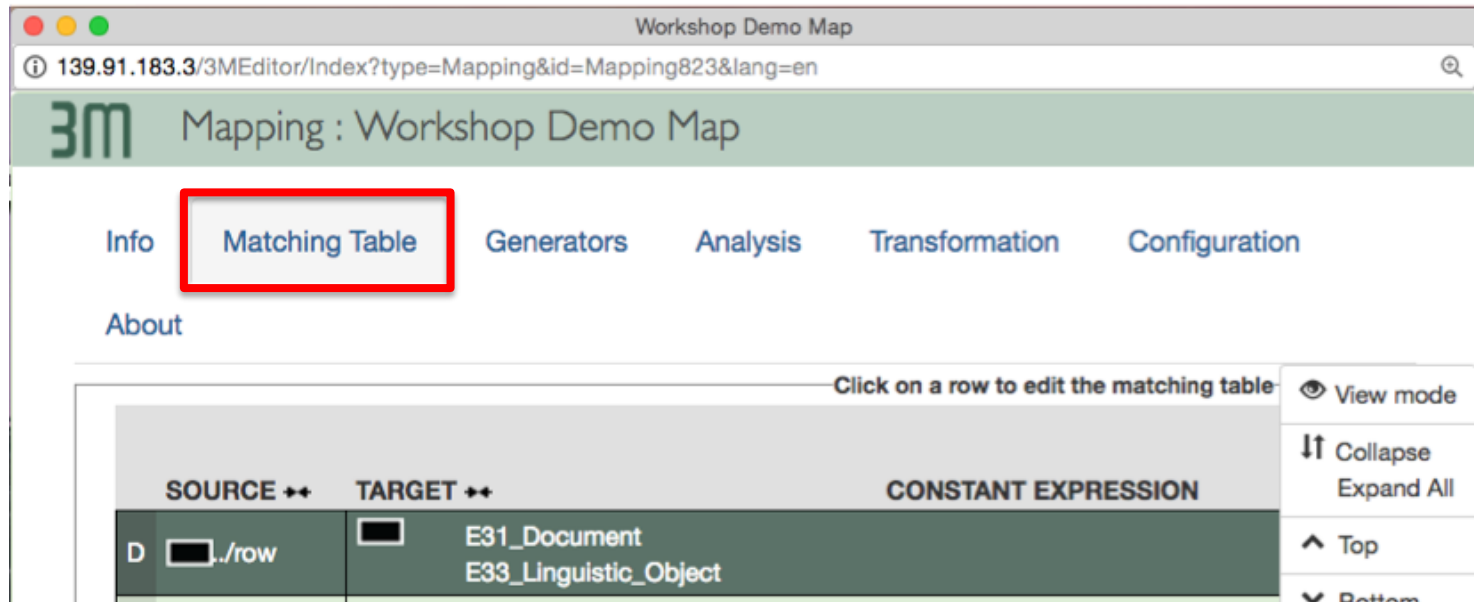
4. MATCHING TABLE OPERATION

Matching Table Operation

Section Content

1. Accessing Matching Table
2. Adding a Map
3. Adding a Link
4. Copy / Delete Maps and Links
5. How to comment
6. View Controls

1. Accessing the Matching Table



The screenshot shows a web browser window titled "Workshop Demo Map" with the URL `139.91.183.3/3MEditor/Index?type=Mapping&id=Mapping823&lang=en`. The page header displays the 3M logo and the title "Mapping : Workshop Demo Map". A navigation bar contains several tabs: "Info", "Matching Table" (highlighted with a red rectangle), "Generators", "Analysis", "Transformation", and "Configuration". Below the navigation bar is an "About" link. The main content area features a table with the following structure:

Click on a row to edit the matching table		
SOURCE ↔	TARGET ↔	CONSTANT EXPRESSION
D <input type="checkbox"/> ./row	<input type="checkbox"/>	E31_Document E33_Linguistic_Object

On the right side of the table, there is a vertical menu with the following options: "View mode", "Collapse/Expand All", "Top", and "Bottom".

- From within your mapping, click on 'matching table' tab

2. Adding a New Map

The screenshot shows the '3M Mapping: Workshop Demo Map' interface. The 'Matching Table' tab is active, displaying a table with columns: SOURCE **, TARGET **, IF RULE **, and COMMENTS **. A red box highlights the '+ Map' button in the top row of the table. Below the table, there is a detailed view of a map configuration, including a 'Source Node' field, a 'Target Entity' dropdown, and a 'Select a value' input field. The interface also features a sidebar with 'View mode', 'Collapse', 'Expand All', 'Top', 'Bottom', and 'XML' options.

- Maps are used for mapping a domain (root node) from the source to a specific class in the target ontology
- If your data structure is complex (not flat) you can use multiple maps to map multiple domains (root nodes) in the source data structure
- Use '+ Map' button to add additional maps

1. Adding Map Examples

```
<root>
  <row>
    <Type>Text</Type>
    <Title>Protocols of Proceedings of Crimea Conference</Title>
    <Subtitle>Declaration of Liberated Europe</Subtitle>
    <Date>11/2/1945</Date>
    <Creator>
      <name>Joseph Stalin</name>
      <role>The Premier of the Union of Soviet Socialist Republics</role>
    </Creator>
    <Publisher>
      <name>Winston Churchill</name>
      <role>The Prime Minister of the United Kingdom</role>
    </Publisher>
    <Subject>Postwar Division of Europe</Subject>
  </row>
</root>
```

Mapping: Workshop Demo Map

SOURCE **	TARGET **	CONSTANT EXPRESSION	IF RULE **	COMMENTS **
D [] ./row	E31_Document E33_Linguistic_Object			

+ Link + Map

SOURCE **	TARGET **	CONSTANT EXPRESSION	IF RULE **	COMMENTS **
D [] ./Creator	E39_Actor			

+ Link + Map

- We add maps for significant nodes in the source XML that have many relations to re-express in the target ontology. These are primary foci of interest.
- In this example 'row' is the root node under which individual source records in this schema are documented, attributing title, subtitle etc. This is our focus of interest and becomes the source domain in a map.
- In this example there is a subnode of 'row' called 'creator' which has its own list of properties. We can create a new map for this more complex node and later link the two maps. This simplifies mapping and makes the mapping definition easier to read.

3. Adding a New Link

The screenshot shows the '3M Mapping: Workshop Demo Map' interface. The 'Matching Table' tab is active, displaying a table with columns: SOURCE **, TARGET **, IF RULE **, and COMMENTS **. The table has two rows: 'D' (Domain) and 'P' (Property). The 'P' row is highlighted with a red box. Below the table, a red box highlights the '+ Link' button.

	SOURCE **	TARGET **	IF RULE **	COMMENTS **
D				
P	<div>Source Relation</div> <div>↓</div> <div>Fill in value</div> <div>Add intermediate</div> <div>Set blank source relation</div>	<div>Target Relation</div> <div>↓</div> <div>Select a value</div> <div>Add intermediate</div>	<div>Add rule ▼</div>	<div>Add comment about ▼</div>
R	<div>Source Node</div> <div>□</div> <div>Fill in value</div> <div>Add additional class</div> <div>Add instance info ▼</div>	<div>Target Entity</div> <div>□</div> <div>Select a value</div> <div>Add additional class</div> <div>Add instance info ▼</div>	<div>Add constant expression</div> <div>Add rule ▼</div>	<div>Add comment about ▼</div>

+ Link + Map

- A link is used for mapping one field/node under a domain (root node) from the source to a semantic path in the target ontology
- For each field/node in the source that should be mapped, there should be at least one link in the map
- Use '+ Link' button to add additional links

3. Adding Link Examples

```

▼<root>
  ▼<row>
    <Type>Text</Type>
    <Title>Protocols of Proceedings of Crimea Conference</Title>
    <Subtitle>Declaration of Liberated Europe</Subtitle>
    <Date>11/2/1945</Date>
    ▼<Creator>
      <name>Joseph Stalin</name>
      ▼<role>
        The Premier of the Union of Soviet Socialist Republics
      </role>
    </Creator>
    ▼<Creator>
      <name>Winston Churchill</name>
      <role>The Prime Minister of the United Kingdom</role>
    </Creator>
    ▼<Creator>
      <name>Theodore Roosevelt</name>
      <role>President of the United States of America</role>
    </Creator>
    ▼<Publisher>
      <name>State Department</name>
    </Publisher>
    <Subject>Postwar Division of Europe</Subject>
  </row>
</root>

```

3M Mapping : Workshop Demo Map

Info Matching Table Generators Analysis Transformation Configuration About

Click on a row to edit the matching table

SOURCE **	TARGET **	CONSTANT EXPRESSION	IF RULE **	COMMENTS **
D <input type="checkbox"/> /row	E31_Document E33_Linguistic_Object			
P <input type="checkbox"/> Type	P2_has_type			
R <input type="checkbox"/> Type	E55_Type			
P <input type="checkbox"/> Creator	P94i_was_created_by E65_Creation [create1] P14_carried_out_by			
R <input type="checkbox"/> Creator	E39_Actor			
P <input type="checkbox"/> Date	P94i_was_created_by E65_Creation [create1] P4_has_time-span E52_Time-Span P82_at_some_time_within			
R <input type="checkbox"/> Date	rdf-schema#Literal			
P <input type="checkbox"/> Title	P1_is_identified_by			
R <input type="checkbox"/> Title	E41_Appellation			
P <input type="checkbox"/> Subtitle	P1_is_identified_by			
R <input type="checkbox"/> Subtitle	E41_Appellation	[P2_has_type a] [E55_Type = "Subtitle"]		

+ Link + Map

EXAMPLE:

- The domain (root node) 'row' contains all information about our base record in the source
- 'row' has five properties 'Title', 'Subtitle', 'Date', 'Creator', 'Publisher' and 'Subject'
- For each property we want to map, we need one link in the map

4. Copying and Deleting Maps/Links

The screenshot shows the '3M Mapping: Workshop Demo Map' interface. The 'Matching Table' tab is active, displaying a table with columns: SOURCE, TARGET, IF RULE, and COMMENTS. The table has two rows, 'D' and 'P'. The 'P' row is expanded, showing details for 'Source Relation' and 'Target Relation'. A red box highlights the 'COPY' (two overlapping sheets) and 'DELETE' (X) icons in the 'COMMENTS' column of the 'P' row. Below the table are '+ Link' and '+ Map' buttons. On the right, a sidebar contains 'View mode', 'Collapse/Expand All', 'Top/Bottom', and 'XML' options.

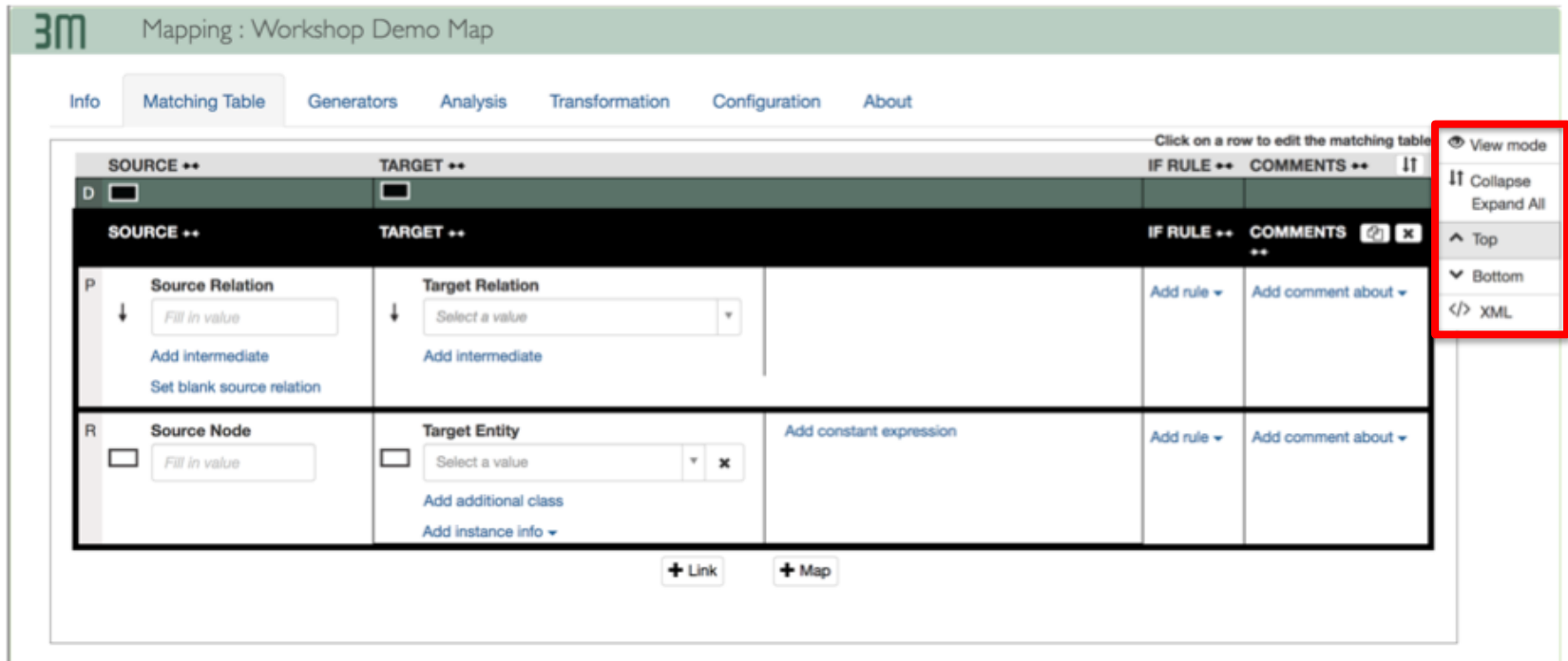
- Longer maps/links can take time to generate. The copying feature allows one to quickly duplicate existing maps/links.
- To copy a map or link click the 'duplicate' icon, followed by the 'clipboard icon'
- To delete a map/link, select the appropriate map/link and then click the 'x'

5. Commenting

The screenshot displays the 3M Editor interface for a 'Workshop Demo Map'. The top navigation bar includes tabs for Info, Matching Table (selected), Generators, Analysis, Transformation, Configuration, and About. The main workspace is divided into a table with columns for SOURCE, TARGET, IF RULE, and COMMENTS. The table is currently empty, with headers for SOURCE and TARGET. Below the headers, there are two rows of input fields: 'Source Relation' and 'Target Relation' for the first row, and 'Source Node' and 'Target Entity' for the second row. Each input field has a 'Fill in value' placeholder. To the right of the input fields, there are buttons for 'Add intermediate', 'Set blank source relation', 'Add additional class', and 'Add instance info'. A dropdown menu is open in the COMMENTS column, showing options: 'Add comment about', 'Rationale', 'Alternatives', 'Typical mistakes', 'Local Habits', 'Link to Cook Book', 'Example Source', 'Example Target', 'Comments Last Update Person', and 'Comments Last Update Date'. The dropdown menu is highlighted with a red box. At the bottom of the interface, there are buttons for '+ Link' and '+ Map'.

- 3MEditor allows you to make comments on the mapping of individual fields/nodes in links. This is a useful feature to explain/remember mapping decisions
- To add comment click 'Add comment about'

6. View Controls



3MEditor also offers a series of control tools to help navigate and work with your mapping

- **View Mode:** toggles map to read only
- **Collapse/Expand All:** Opens or Closes all Links in Maps
- **Top/Bottom:** Provides scrolling over long maps
- **</>XML:** Allows direct editing of the underlying X3ML definition of the mapping

5. MAPPING PATTERNS

Mapping Patterns Section Content

1. Mapping Source Root to Target Domain
2. Simple Field Mapping (One to One)
3. Introducing Intermediate Nodes
4. Adding constants
5. Using variables
6. Joining Maps
7. Multiple instantiation

1. Mapping Source to Target Domain

The screenshot shows a mapping interface with two main columns: SOURCE and TARGET. Each column has a header with a double-headed arrow icon. Below the headers, there are two rows. The first row is highlighted with a black background. The second row is white and contains the mapping configuration. In the SOURCE column of the second row, there is a 'Source Node' dropdown menu with the value '/root/row' selected. In the TARGET column of the second row, there is a 'Target Entity' dropdown menu with the value 'E31_Document' selected. Both dropdown menus are highlighted with red rectangles. Below the 'Target Entity' dropdown, there are two links: 'Add additional class' and 'Add instance info'.

SOURCE ↔	TARGET ↔
SOURCE ↔	TARGET ↔
D Source Node /root/row	Target Entity E31_Document Add additional class Add instance info

- Click top row in a map
- On Source Side, choose appropriate domain (root node) from source
- On Target Side, choose appropriate class in target ontology

This now says:

“for each instance of source domain (root node) generate one instance of target class”

1. Mapping Source to Target Domain Example

Mapping Instruction

Source

```
<root>
  <row>
    <Type>Text</Type>
    <Title>Protocols of Proceedings of Crimea Conference</Title>
    <Subtitle>Declaration of Liberated Europe</Subtitle>
    <Date>11/2/1945</Date>
    <Creator>
      <name>Joseph Stalin</name>
      <role>
        The Premier of the Union of Soviet Socialist Republics
      </role>
    </Creator>
    <Creator>
      <name>Winston Churchill</name>
      <role>The Prime Minister of the United Kingdom</role>
    </Creator>
    <Creator>
      <name>Theodore Roosevelt</name>
      <role>President of the United States of America</role>
    </Creator>
    <Publisher>
      <name>State Department</name>
    </Publisher>
    <Subject>Postwar Division of Europe</Subject>
  </row>
</root>
```

SOURCE ++	TARGET ++
<div>Source Node</div> <div>/root/row x ▾</div>	<div>Target Entity</div> <div>E31_Document x ▾</div> <div>Add additional class</div> <div>Add instance info ▾</div>

Generates
Output

E31 Document
Row/1

So this mapping instruction says,

“for each instance of ‘row’ in source, create an RDF instance of type ‘E31 Document’”

2. Simple Field Mapping: One to One

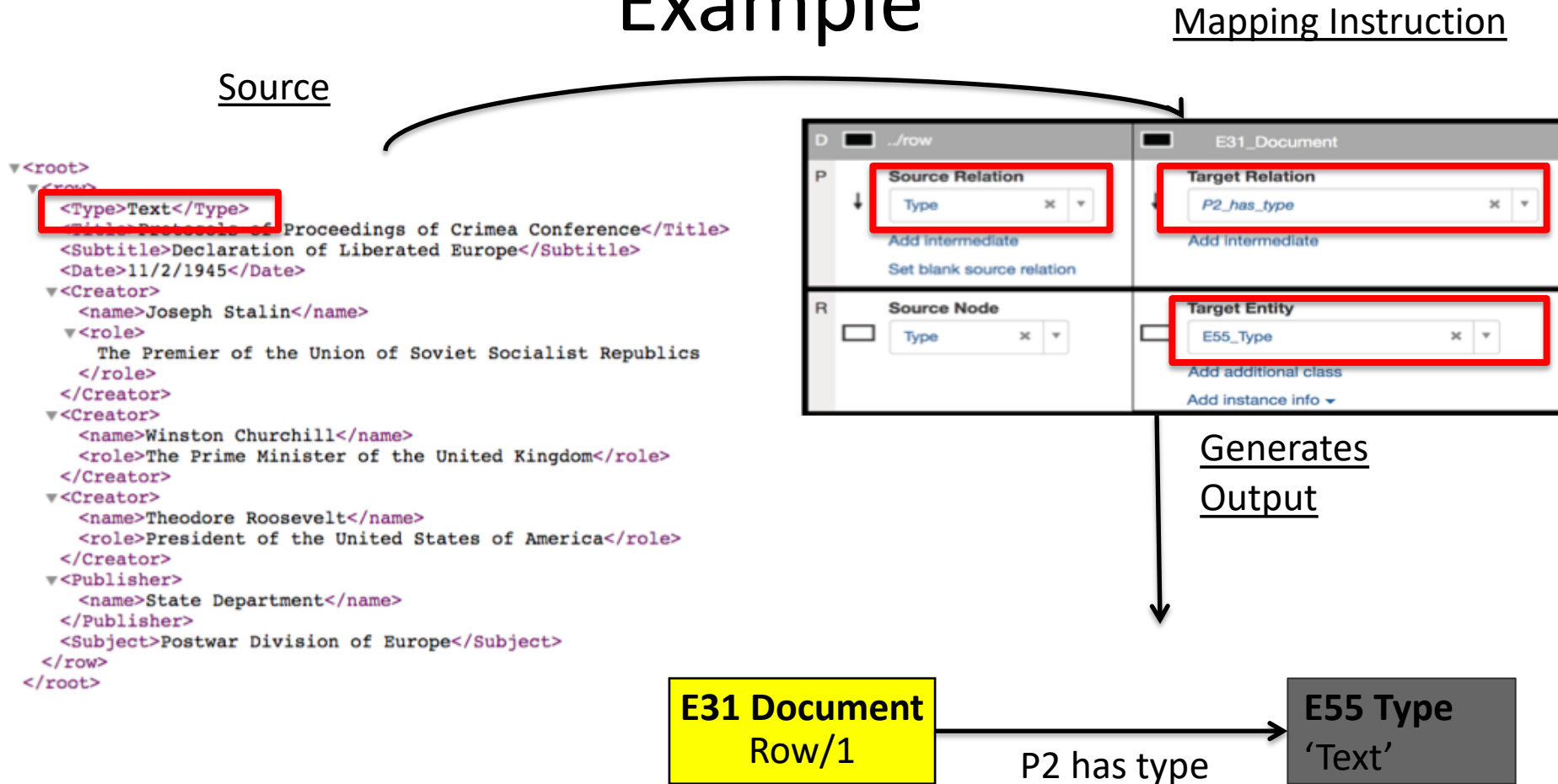
D <input type="checkbox"/> ../row	E31_Document
<p>P <input type="checkbox"/></p> <p>Source Relation</p> <p><input type="text" value="Type"/> <input type="button" value="x"/> <input type="button" value="v"/></p> <p>Add intermediate</p> <p>Set blank source relation</p>	<p>Target Relation</p> <p><input type="text" value="P2_has_type"/> <input type="button" value="x"/> <input type="button" value="v"/></p> <p>Add intermediate</p>
<p>R <input type="checkbox"/></p> <p>Source Node</p> <p><input type="text" value="Type"/> <input type="button" value="x"/> <input type="button" value="v"/></p>	<p>Target Entity</p> <p><input type="text" value="E55_Type"/> <input type="button" value="x"/> <input type="button" value="v"/></p> <p>Add additional class</p> <p>Add instance info <input type="button" value="v"/></p>

- Choose field for mapping from Source
- Add one relation and one target class in target

This creates a simple triple statement in RDF of form S-V-O

2. Simple Mapping One to One

Example



So this mapping instruction says,

“for each instance of ‘type’ in ‘row’, create an RDF triple stating that this instance of ‘E31 Document’” has type ‘E55 Type’

3. Introducing Intermediate Nodes

P Source Relation ↓ <div>Date × ▾</div> Add intermediate Set blank source relation	Target Relation ↓ <div>P94i_was_created_by × ▾</div> Target Entity ■ <div>E65_Creation × ▾</div> Add additional class Add instance info ▾ Target Relation ↓ <div>P4_has_time-span × ▾</div> Target Entity ■ <div>E52_Time-Span × ▾</div> Add additional class Add instance info ▾ Target Relation ↓ <div>P82_at_some_time_within × ▾</div> Add intermediate
R Source Node <div><input type="checkbox"/> Date × ▾</div>	Target Entity <div><input type="checkbox"/> rdf-schema#Literal × ▾ ×</div> Add additional class Add instance info ▾

For more complex paths, intermediate nodes are necessary.

- ‘Click ‘Add Intermediate Node’ for as many intermediate relations as you may need to build

In this example, the date of the creation of the Crimea document is expressed.

3. Introducing Intermediate Nodes

Example

Source

```
<root>
  <row>
    <Type>Text</Type>
    <Title>Protocols of Proceedings of Crimea Conference</Title>
    <Subtitle>Declaration of Liberated Europe</Subtitle>
    <Date>11/2/1945</Date>
    <Creator>
      <name>Joseph Stalin</name>
      <role>The Premier of the Union of Soviet Socialist Republics</role>
    </Creator>
    <Creator>
      <name>Winston Churchill</name>
      <role>The Prime Minister of the United Kingdom</role>
    </Creator>
    <Creator>
      <name>Theodore Roosevelt</name>
      <role>President of the United States of America</role>
    </Creator>
    <Publisher>
      <name>State Department</name>
    </Publisher>
    <Subject>Postwar Division of Europe</Subject>
  </row>
</root>
```

Mapping Instruction

Source Relation Date X ▼ Add intermediate Set blank source relation	Target Relation P94i_was_created_by X ▼ Target Entity E65_Creation X ▼ Add additional class Add instance info ▼
	Target Relation P4_has_time-span X ▼ Target Entity E52_Time-Span X ▼ Add additional class Add instance info ▼
Source Node <input type="checkbox"/> Date X ▼	Target Entity <input type="checkbox"/> rdf-schema:Literal X X Add additional class Add instance info ▼

Output

E31 Document
Row/1

P94i was
created by

E65 Creation
*

P4 has time-span

E52 Time Span
*

P82 at some time within

RDF Literal
11/2/1945

So this mapping instruction says,
“for each instance of ‘date’ in ‘row’,
create a series of RDF triples stating that this instance
of ‘E31 Document’” was created by ‘E65 Creation’
has time span “E52 Time Span” at some time within “Some Date”

4. Adding Constant Data

The screenshot shows a mapping tool interface with three main sections: Source Node, Target Entity, and Relation. The Source Node section has a dropdown menu with 'Subtitle' selected. The Target Entity section has a dropdown menu with 'E41_Appellation' selected. The Relation section is highlighted with a red border and contains a dropdown menu with 'P2_has_type' selected. Below the Relation dropdown, there is an 'Entity' section with a dropdown menu showing 'E55_Type'. Below the Entity section, there is a 'Constant' section with a text input field containing 'Subtitle'. At the bottom of the Relation section, there is a link 'Add constant expression'.

Sometimes, we want to add additional constant information to our mapping.

- Click 'Add Constant Expression' (to the right of target)
- Add appropriate links and constant info

In this example, we want to express for each node representing a subtitle in the target graph is of type 'subtitle'.

4. Adding Constant Data Example

Mapping Instruction

Source

```
<root>
  <row>
    <Type>Text</Type>
    <Title>Proceedings of Crimea Conference</Title>
    <Subtitle>Declaration of Liberated Europe</Subtitle>
    <Date>11/2/1945</Date>
    <Creator>
      <name>Joseph Stalin</name>
      <role>
        The Premier of the Union of Soviet Socialist Republics
      </role>
    </Creator>
    <Creator>
      <name>Winston Churchill</name>
      <role>The Prime Minister of the United Kingdom</role>
    </Creator>
    <Creator>
      <name>Theodore Roosevelt</name>
      <role>President of the United States of America</role>
    </Creator>
    <Publisher>
      <name>State Department</name>
    </Publisher>
    <Subject>Postwar Division of Europe</Subject>
  </row>
</root>
```

Target Entity
☐ E41_Appellation x v
Add additional class
Add instance info v

Relation
→ P2_has_type x v
Entity
E55_Type x v
Constant
: Subtitle
Add instance info v
Add constant expression

Generates Output

P1 is identified by

E31 Document
Row/1

E41 Appellation
'Declaration of.'


P2 has type


E55 Type
'Subtitle'

So this mapping instruction says,
“for each instance of ‘subtitle’ in ‘row’, create a series of RDF
triples stating that:
“E31 Document” is identified by a ‘E41 Appellation’ AND
‘E41 Appellation’ has type “E55 Type” (Subtitle)

5. Using Variables

[Add instance info](#) ▼

 Is same as (map)

 Is same as (global)

D	../row	E31_Document
P	Source Relation ↓ Date	Target Relation ↓ P94i_was_created_by
	Add intermediate Set blank source relation	Target Entity E65_Creation
		Add additional class Is same as (map)
		[create1]
		Add instance info ▼

P	↓ Creator	↓ P94i_was_created_by E65_Creation [create1]
R	□ Creator	↓ P14_carried_out_by E39_Actor
P	↓ Date	↓ P94i_was_created_by E65_Creation [create1]
R	□ Date	↓ P4_has_time-span E52_Time-Span ↓ P82_at_some_time_within rdf-schema#Literal

When two nodes in a mapping refer to the same real world entity, we can tell the mapping engine to generate only one node for this entity

For each node we wish to merge,

- click 'Add instance info'
- Click 'is same as'
- Add variable (random name) to both nodes

In this example there is one creation of the Crimean Protocols

5. Using Variables Example

Source

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <row>
    <Type>Text</Type>
    <Title>Protocols of Proceedings of Crimea Conference</Title>
    <Subtitle>Declaration of Liberated Europe</Subtitle>
    <Date>11/2/1945</Date>
    <Creator>
      <name>Joseph Stalin</name>
      <role>The Premier of the Union of Soviet Socialist Republics</role>
    </Creator>
    <Creator>
      <name>Winston Churchill</name>
      <role>The Prime Minister of the United Kingdom</role>
    </Creator>
    <Creator>
      <name>Theodore Roosevelt</name>
      <role>President of the United States of America</role>
    </Creator>
    <Publisher>
      <name>State Department</name>
    </Publisher>
    <Subject>Postwar Division of Europe</Subject>
  </row>
</root>
```

Mapping Instruction

P	↓ Creator	P94i_was_created_by	E65_Creation [create1]
R	□ Creator	P14_carried_out_by	E39_Actor
P	↓ Date	P94i_was_created_by	E65_Creation [create1]
R	□ Date	P4_has_time-span	E52_Time-Span
		P82_at_some_time_within	rdf-schema#Literal

Generates

Output

so this mapping instruction says,

E31 Document
Row/1

P94i was
created by

E65 Creation
create1

The Creation in the 'Creator' link is the same
with the Creation in the 'Date' link.

The series of RDF triples that this will generate are:

'E31 Document' was created by 'E65 Creation' (create1) AND
'E65 Creation' (create1) has time span "E52 Time Span" at
some time within "Some Date" AND
'E65 Creation' (create1) carried out by "E21 Person"

P14 carried out by

E39 Actor

P4 has time-span

E52 Time Span
*

P82 at some time within

RDF Literal
11/2/1945

6. Joining Maps

SOURCE ↔		TARGET ↔	C
D	../row	E31_Document	
P	↓ Type	↓ P2_has_type	
R	Type	E55_Type	
P	↓ Creator	↓ P94i_was_created_by	
		E65_Creation [create1]	
		↓ P14_carried_out_by	
R	Creator	E39_Actor	

SOURCE ↔		TARGET ↔	CONSTRAINT
D	../Creator	E39_Actor	
P	↓ name	↓ P1_is_identified_by	
R	name	E41_Appellation	
P	↓ role	↓ P107i_is_current_or_former_member_of	
R	role	E74_Group	

Two maps can be joined, just in case the first maps a node which is the domain of a second map. This is often done for convenience purposes, to make mapping easier and simple to read.

E.g.: There is a subnode in the XML that has many fields describing an actor. In map 1, reference Actor. In map 2, map all fields for Actor node.

6. Joining Maps Example

Source

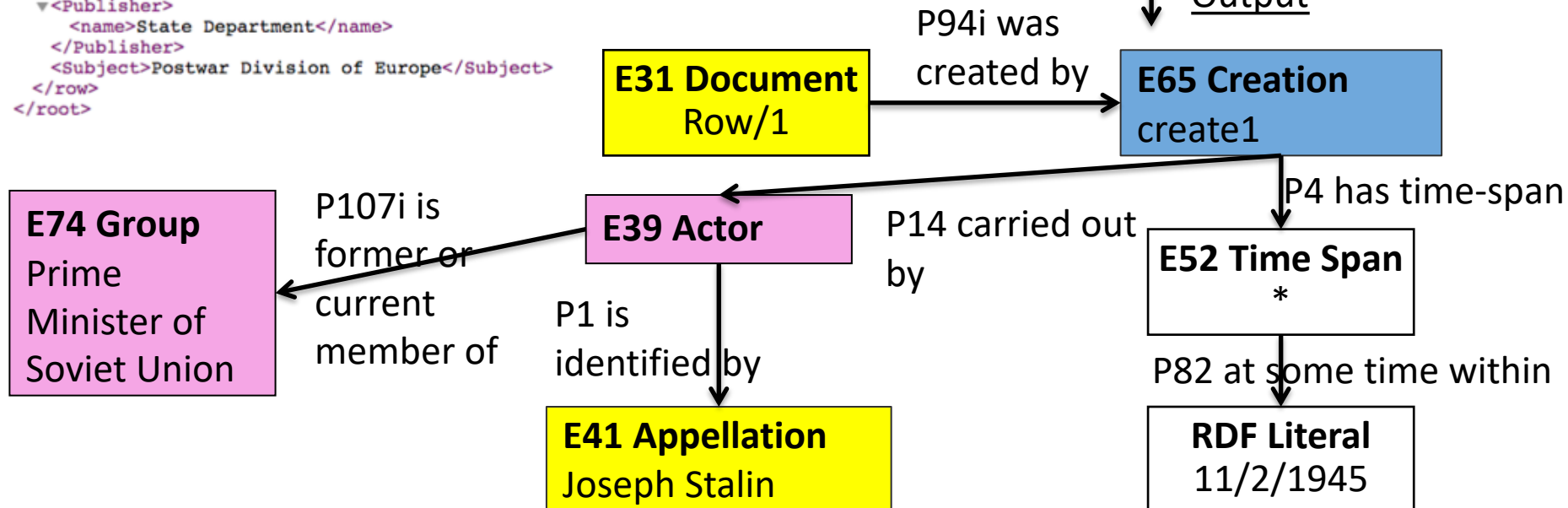
Mapping Instruction

```
<root>
  <row>
    <Type>Text</Type>
    <Title>Protocols of Proceedings of Crimea Conference</Title>
    <Subtitle>Declaration of Liberated Europe</Subtitle>
    <Date>11/2/1945</Date>
    <Creator>
      <name>Joseph Stalin</name>
      <role>The Premier of the Union of Soviet Socialist Republics</role>
    </Creator>
    <Creator>
      <name>Winston Churchill</name>
      <role>The Prime Minister of the United Kingdom</role>
    </Creator>
    <Creator>
      <name>Theodore Roosevelt</name>
      <role>President of the United States of America</role>
    </Creator>
    <Publisher>
      <name>State Department</name>
    </Publisher>
    <Subject>Postwar Division of Europe</Subject>
  </row>
</root>
```

SOURCE **	TARGET **	C
D <input type="checkbox"/> ./row	E31_Document	
P <input type="checkbox"/> Type	P2_has_type	
R <input type="checkbox"/> Type	E55_Type	
P <input type="checkbox"/> Creator	P94i_was_created_by	
	E65_Creation [create1]	
R <input type="checkbox"/> Creator	P14_carried_out_by	
	E39_Actor	

SOURCE **	TARGET **	CONSTRAINT
D <input type="checkbox"/> ./Creator	E39_Actor	
P <input type="checkbox"/> name	P1_is_identified_by	
R <input type="checkbox"/> name	E41_Appellation	
P <input type="checkbox"/> role	P107i_is_current_or_former_member_of	
R <input type="checkbox"/> role	E74_Group	

Generates
Output



7. Multiple Instantiation

SOURCE ↔	TARGET ↔
SOURCE ↔	TARGET ↔
<div>D</div> <div><div>Source Node</div><div><div>/root/row</div><div>×</div><div>▼</div></div></div>	<div>Target Entity</div> <div><div>E31_Document</div><div>×</div><div>▼</div><div>×</div></div> <div><div>E33_Linguistic_Object</div><div>×</div><div>▼</div><div>×</div></div> <div><div>Add additional class</div></div> <div><div>Add instance info ▼</div></div>

For any node, if it is an instance of two or more classes, we can express this in 3M by using the 'add additional class' feature.

In this example, we say each 'row' from source both is an E31 Document and an E33 Linguistic Object. This allows the use of relations from both classes.

7. Multi Instantiation Example

Mapping Instruction

Source

```
<?xml version="1.0"?>
<root>
  <row>
    <Type>Text</Type>
    <Title>Protocols of Proceedings of Crimea Conference</Title>
    <Subtitle>Declaration of Liberated Europe</Subtitle>
    <Date>11/2/1945</Date>
    <Creator>
      <name>Joseph Stalin</name>
      <role>
        The Premier of the Union of Soviet Socialist Republics
      </role>
    </Creator>
    <Creator>
      <name>Winston Churchill</name>
      <role>The Prime Minister of the United Kingdom</role>
    </Creator>
    <Creator>
      <name>Theodore Roosevelt</name>
      <role>President of the United States of America</role>
    </Creator>
    <Publisher>
      <name>State Department</name>
    </Publisher>
    <Subject>Postwar Division of Europe</Subject>
  </row>
</root>
```

SOURCE ↔	TARGET ↔
SOURCE ↔	TARGET ↔
D	Target Entity
Source Node	E31_Document ✕ ▼ ✕
/root/row ✕ ▼	E33_Linguistic_Object ✕ ▼ ✕
	Add additional class
	Add instance info ▼

Generates
Output

E33 Linguistic Object
E31 Document
Row/1

So this mapping instruction says,

“for each instance of ‘row’ in source, create an RDF instance of both type ‘E31 Document’ and ‘E33 Linguistic Object’”

6. INSTANCE GENERATORS

Instance Generator Section Content

1. Why Instance Generators, what do they do?
2. Defining instance generation functions/patterns (offline)
3. Adding generator file
4. Opening up the generator editor
5. Specifying Instance Generators
6. Testing Transform

1. Why instance generators?

The mapping table allows you to make a translation between a source schema and an RDFS encoded schema like CIDOC CRM.

Each node specified in the target will become a separate data entity in the semantic graph that is created through the X3ML transformation engine.

This separate data entity will need a unique identifier by which it can be found in the system (like a unique key in a relational database)

The instance generators allow specifying patterns for building unique identifiers for instances called 'URIs'.

Because a URI is often unreadable to the uninitiated, it is highly recommended that for each node a label (usually the actual data value from your source schema) be added to each node as well. This is also done through the instance generator.

2. Defining Instance Generators (Offline)

Prefix will be used to define base path of URI

The instance generator is a small piece of simple XML code. This is generated with a simple XML file outside X3ML defining convenient patterns. It can then be uploaded to 3MEditor and used to assign the instance generators.



```
<?xml version="1.0" encoding="UTF-8"?>
<generator_policy>
  <generator name="SimpleLabel">
    <pattern>{label}</pattern>
  </generator>
  <generator name="CompositeLabel">
    <pattern>{label} {text}</pattern>
  </generator>
  <generator name="LocalTermURI" prefix="parthenos">
    <pattern>{hierarchy}/{term}</pattern>
  </generator>
  <generator name="LocalTermURI-2" prefix="parthenos">
    <pattern>{level1}/{hierarchy}/{term}</pattern>
  </generator>
  <generator name="GermanDateTime">
    <custom generatorClass="gr.forth.GermanDate">
      <set-arg name="bound" type="constant"/>
      <set-arg name="text"/>
    </custom>
  </generator>
  <generator name="URIorUUID">
    <custom generatorClass="gr.forth.URIorUUID">
      <set-arg name="text"/>
    </custom>
  </generator>
</generator_policy>
```

3. Adding generator file

Schema **Type** **Version** **Collection**

① Upload File

Namespaces

Namespace prefix **Namespace uri**

+ Target

Sample data

This section consists of information about example data (source and target) and generator policy. Once a source record XML file is uploaded, the "Transformation" tab is enabled (**Transformation tab**). In order to test how your source record XML file transforms to RDF/XML, N-triples or Turtle, you will probably also have to upload a generator policy XML file.

If you have not uploaded an XSD source schema yet, the "Source Analyzer" option will also be enabled once a source record XML file is uploaded (**Configuration tab**) and you may select source paths from a drop down.

Provided by **Contact person(s)** **Source record** **Generator policy** **Target record**

view xml ✕

1. Add a new 'target' scheme, and specify the same namespace prefix as in your generator file. Use 'namespace uri' field to specify your base namespace
2. Upload the generator file

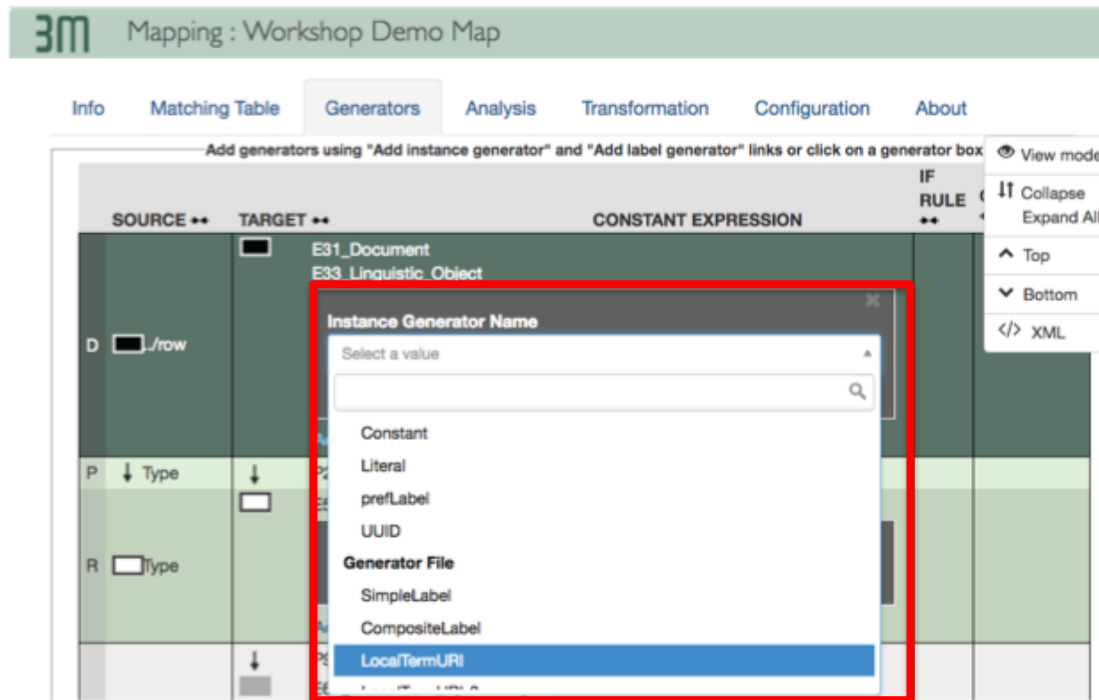
4. Opening Generator Editor

The screenshot shows the '3M Mapping : Workshop Demo Map' interface. The 'Generators' tab is highlighted with a red box. Below the tabs, a message states: 'Add generators using "Add instance generator" and "Add label generator" links or click on a generator box'. The main area displays a table with columns: SOURCE ↔, TARGET ↔, CONSTANT EXPRESSION, and IF RULE ↔. The table contains two rows. The first row has a source of 'D' with a box containing './row', a target of 'E31_Document' and 'E33_Linguistic_Object', and links for 'Add instance generator' and 'Add label generator'. The second row has a source of 'P' with a box containing '↓ Type', a target of 'P2_has_type' and 'E55_Type', and a link for 'Add label generator'. A modal window is open over the second row, titled 'Instance Generator Name', with a text input field containing 'UUID'. On the right side, a context menu is visible with options: 'View mode', 'Collapse', 'Expand All', 'Top', 'Bottom', and 'XML'.

SOURCE ↔	TARGET ↔	CONSTANT EXPRESSION	IF RULE ↔
D <input type="text" value="./row"/>	<input type="text" value="E31_Document"/> <input type="text" value="E33_Linguistic_Object"/> Add instance generator Add label generator		
P <input type="text" value="↓ Type"/>	<input type="text" value="P2_has_type"/> <input type="text" value="E55_Type"/> <div>Instance Generator Name <input type="text" value="UUID"/> Add label generator</div>		

Opening generator editor is easy, just click to the generators tab.

5. Specifying Instance Generators



1. Click Add Instance Generator
2. Select Instance Generator Type

5. Specifying Instance Generators

The screenshot shows a configuration window for 'E31_Document' and 'E33_Linguistic_Object'. It features two main sections for defining arguments. The top section has an 'Instance Generator Name' dropdown set to 'LocalTermURI'. Below it, an 'Argument' section contains a 'Name' field with 'hierarchy' and a 'Value' field with the placeholder 'Fill in value'. The bottom section has a 'Name' field with 'term' and a 'Value' field with 'text()'. A 'Type' dropdown menu is open, showing options: 'xpath' (selected), 'xpathPosition', 'constant', and 'position'. An 'Add argument' button is at the bottom left.

1. Each argument in a generator specifies a piece of the URI string
2. Each argument in the URI can be of different types, chiefly 'Constant' (a value you specify) or 'xpath' (a value from the source)
3. The unique portion of the URI should usually be drawn from the source data. It is accessed from a node using the command 'text()'

6. Testing Transforms

To test transform all mapped entities must have a instance generator specified.

- Click on 'Transform' tab
- Click 'Run Transformation'
- Analyze resulting RDF and/or error messages

The screenshot shows the 'Mapping: Workshop Demo Map' interface. The 'Transformation' tab is selected and highlighted with a red box. Below the tab, the 'Metadata Transformation' section contains instructions and a 'Run Transformation' button, which is also highlighted with a red box. The 'Source Record XML File' and 'Generator Policy XML File' sections are visible, showing XML code. The 'UUID Size' is set to 2, and the 'Output Format' is set to 'RDF/XML'. The 'Target Record RDF File' section is at the bottom.

Mapping : Workshop Demo Map

Info Matching Table Generators Analysis **Transformation** Configuration About

Metadata Transformation

If you have a Generator Policy xml file available, please upload it (Info tab). Otherwise you are provided with a generic one. You may modify both source and generator files on the fly, but changes will be lost next time you click on the Transformation tab. (You may read instructions about generators by clicking [Notes about Generators](#)). Choose UUID size and output format and then click "Run Transformation". View metadata transformed (or x3ml exception messages) inside "Target Record RDF File" textarea.

Source Record XML File

```
<?xml version="1.0" encoding="UTF-8"?><root>
<row>
  <Type>Text</Type>
  <Title>Protocols of Proceedings of Crimea
Conference</Title>
  <Subtitle>Declaration of Liberated Europe</Subtitle>
  <Date>11/2/1945</Date>
  <Creator>
    <name>Joseph Stalin</name>
    <role>The Premier of the Union of Soviet Socialist
Republics
  </role>
  </Creator>
  <Creator>
    <name>Winston Churchill</name>
    <role>The Prime Minister of the United Kingdom
  </role>
  </Creator>
  <Creator>
    <name>Theodore Roosevelt</name>
  </Creator>

```

Generator Policy XML File

```
<?xml version="1.0" encoding="UTF-8"?><generator_policy>
  <generator name="SimpleLabel">
    <pattern>[label]</pattern>
  </generator>
  <generator name="CompositeLabel">
    <pattern>[label] {text}</pattern>
  </generator>
  <generator name="LocalTermURI" prefix="parthenos">
    <pattern>[hierarchy]/[term]</pattern>
  </generator>
  <generator name="LocalTermURI-2"
prefix="parthenos">
    <pattern>[level1]/[hierarchy]/[term]</pattern>
  </generator>
  <generator name="GermanDateTime">
    <custom generatorClass="gr.forth.GermanDate">

```

UUID Size: 2

Output Format: RDF/XML N-triples Turtle

Run Transformation

Target Record RDF File (Using x3ml engine version 1.7.3)

THANK YOU!

Any questions?

G. Bruseker
ICS-FORTH

bruseker@ics.forth.gr