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](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](http://fims.org) management system for the administration of the files and also on the [3MEditor](http://3meditor.org) 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](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](https://github.com/isl/3MEditor)
• 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](https://github.com/isl/x3ml)
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

- Create New User
- Reset Password

OR

- Login

Login at: https://www.ics.forth.gr/isl/3M
1. New User

- Only need a valid email to sign up!
- System will send a verification email and you can begin
2. Control Tools and 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’

- ‘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

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

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 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 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

• Upload Source Data by choosing ‘Upload xml’ button

• Add provenance information of who gave data and how they can be contacted.

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

- Click ‘+Target’ to add new target
- Upload target schema using ‘upload’ button
- For each target schema specify
  - its 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

- From within your mapping, click on ‘matching table’ tab
2. Adding a New Map

- 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

- 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

- 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

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

- 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

- 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

- 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

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

- 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
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

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

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”. 

```
<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>
      </Publisher>
    </Root>
```
4. Adding Constant Data

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

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

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

So this mapping instruction says,

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”
6. Joining Maps

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

<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>
      <Subject>Postwar Division of Europe</Subject>
    </Publisher>
  </row>
</root>

Mapping Instruction

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>TARGET</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>E31_Document</td>
</tr>
<tr>
<td>P</td>
<td>E39_Actor</td>
</tr>
<tr>
<td>R</td>
<td>E41_Apellation</td>
</tr>
</tbody>
</table>

Generates Output

P94i was created by
E65_Creation
create1

P4 has time-span
P14_carried_out_by

E2 Time Span
11/2/1945

P82 at some time within
P14_carried_out_by

E31 Document
Row/1

E39 Actor

E41 Appellation
Joseph Stalin

E74 Group
Prime Minister of Soviet Union

P107i is current or former member of

P1 is identified by

RDF Literal

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

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
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 you source schema) be added to each node as well. This is also done through the instance generator.
2. Defining Instance Generators (Offline)

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
<?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

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

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

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 an instance generator specified.

- Click on ‘Transform’ tab
- Click ‘Run Transformation’
- Analyze resulting RDF and/or error messages
THANK YOU!

Any questions?

G. Bruseker
ICS-FORTH
bruseker@ics.forth.gr