A practical Example:

Model-driven Semantic Interoperability using

- Open Standards

and

- MXV (Model-driven XML Vocabulary) Design and Productivity Tools

VERSION
2.2 (NDR 2.1)

© Data Management Solutions, 2007-2012

Copyright © Data Management Solutions [DMS] 2007-2012. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself must not be modified in any way, such as by removing the copyright notice or references to DMS, except as needed for the purpose of developing DMS specifications.

The limited permissions granted above are perpetual and will not be revoked by DMS or its successors or assigns.

This document and the information contained herein is provided on an AS IS basis and DMS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Document Title: A practical Example:
Model-driven Semantic Interoperability using Open Standards and MXV (Model-driven XML Vocabulary) design

Publication Date: 7th of December 2007

Author:

Table of Contents

Document Overview **3**
Purpose 3
Scope 3
Audience 3
In this document 3
Document history 4
Status 4
References 4

Defining a UML semantic data model **5**
Background 5
In this section 5
UML class diagram 6
UML domains 6
UML classes 7
UML attributes 7
UML associations 8
UML generalisations 8
UML naming of classes and attributes 9
UML meta model extensions 9
UML model validation 11

Transitioning a UML model to an XML library model **12**
Background 12
In this section 12
Unqualified Data Types UDT 13
Qualified Data Types QDT 14
Common Basic Components CBC 14
Common Aggregate Components CAC 15
XML meta model extensions 16
Model administration automation 19

Composing the XML document schema model **21**
Background 21
In this section 21
Document Schema model DOC 22
Model administration automation 23

Validating an XML model **25**
Background 25
In this section 25
XML model validation 26

Generating code from XML models **27**
Background 27
In this section 27
Generating XSD files 28
Generating Genericode skeleton 29
Generating Context Value Association file skeleton 32
Generating Business Constraints file skeleton 33
Generating Forward Compatibility Filter 34

Sample XML Schemas and outputs **35**
Background 35
In this section 35
Sample XML Document Schema 36

Schema Documentation 36
Sample XSD files 36
Sample XPath report 37
Sample XPath XML file 37

UML and XML model integration benefits 38
Background 38
In this section 38
Centralised management 39
Tool enforced validation and consistency 39
Cascading class and attribute definitions 39
Automation of repeating tasks 40
Impact analysis 40

Appendix A: Glossary 41
Semantic model 41
OASIS 41
OASIS Universal Business Language (UBL) 41
OASIS Universal Business Language Naming and Design Rules (UBL NDR) 41
Model-driven XML Vocabulary Naming and Design Rules (MXV NDR) 41
UN/CEFACT 42
UML 42
XML 2-pass content validation 42
Genericode 42
Context Value Association 42
ISO/IEC 11179 Part 5 42

Document Overview

Purpose

This document provides a made up and simplified example to illustrate the paper 'Model-driven Semantic Interoperability Using Open Standards: A Case Study'. The example is MXV NDR 2.1 and thus ESL NDR 2.1 compliant.

The document aims to demonstrate that this solution is applicable to

- any data exchanges between computer systems and organisations
- using W3C XML Schema
- when no existing XML standard is meeting the requirements
- and hand-coded XML Schemas are not acceptable because of well-known lack of re-use, consistency, versioning and change management issues

A demo video is also available for free download at
http://www.d-m-s.co.nz/Download_Demo.htm

A trial download package, including all sample models, MXV Productivity Tools and generated artefacts is available at
http://www.d-m-s.co.nz/serv_xmlschema.htm#download

Scope

This document provides simplified artefact snapshots of the described methodology:

- UML semantic data model (aka ontology or canonical model)
- XML library module models
- XML document schema model
- Optional supplementary XML 2-pass content validation skeleton artefacts
- Optional automation tools
- XML Schema files (XSD)

This document visualises the benefits of a model-driven approach from UML to XML.

Audience

Anyone who wishes to gain a deeper insight *after* reading the paper 'Model-driven Semantic Interoperability Using Open Standards: A Case Study':

- Business Managers
- Strategic Business Users
- IT Managers
- IT Architects
- IT Technical Staff

In this document

The sections of this document are:

Topic	See Page
Defining a UML semantic data model	5
Transitioning a UML model to an XML library model	12
Composing the XML document schema model	21
Validating an XML model	25
Generating code from XML models	27
Sample XML Schemas	35
UML and XML model integration benefits	38
Appendix A: Glossary	41

Continued on next page

Document Overview, Continued

Document history

Version	Date	Status	Edited By	Revision Description
1.0	7th Dec 2007	Final	Tschumperlin Juerg	
1.1	20th Dec 2008	Final	Tschumperlin Juerg	Updated automation tool aspects, replaced references to UMCLVV with XML 2-pass content validation
1.2	26th Mar 2009	Final	Tschumperlin Juerg	Minor updates and clarifications, added references
1.3	3rd Nov 2009	Final	Tschumperlin Juerg	Minor updates and clarifications
1.4	5th Nov 2009	Final	Tschumperlin Juerg	Updated the example to match the MXV Example and MXV trial download.
1.5	20th Jan 2010	Final	Tschumperlin Juerg	Updated screenshots to match latest MXV enhancements
2.0	15th Mar 2010	Final	Tschumperlin Juerg	Aligned to UBL NDR 2,1
2.1	27th July 2011	Final	Tschumperlin Juerg	Updated some screen shots
2.2	20th Nov 2012	Final	Tschumperlin Juerg	Added generators for forward compatibility filters / XPath outputs

Status

Final

References

OASIS UBL NDR 2.0:[1]
http://www.oasis-open.org/specs/index.php#ublv2.0

Crane Softwrights:
http://cranesoftwrights.com/

OASIS Genericode:
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=codelist

OASIS Context Value Association:
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=codelist

New Zealand Education Sector methodology documentation:
http://d-m-s.co.nz/serv_xmlschemaDocs.asp

This document references the registered trademarks of SAP / Sybase PowerDesigner and Microsoft Excel.

[1] As of November 2012, OASIS has not yet published the UBL NDR 2.1 documentation. MXV emulates UBL NDR 2.1.

Defining a UML semantic data model

Background

The UML semantic data model, also known as canonical model or ontology, is a single enterprise model that drives the XML schema implementation with its many document schema models (one for each message type).

Versioning rules allow for the UML model to evolve over time.

There are no restrictions on the business scope of a UML semantic model.

There are however formal requirements that the UML model must meet for it to be fit and suitable for an XML Schema implementation.

Semantic interoperability is business objective, not a technology objective.

Long-term business ownership of the UML semantic model is vital for success.

In this section

The building blocks of the UML semantic model are:

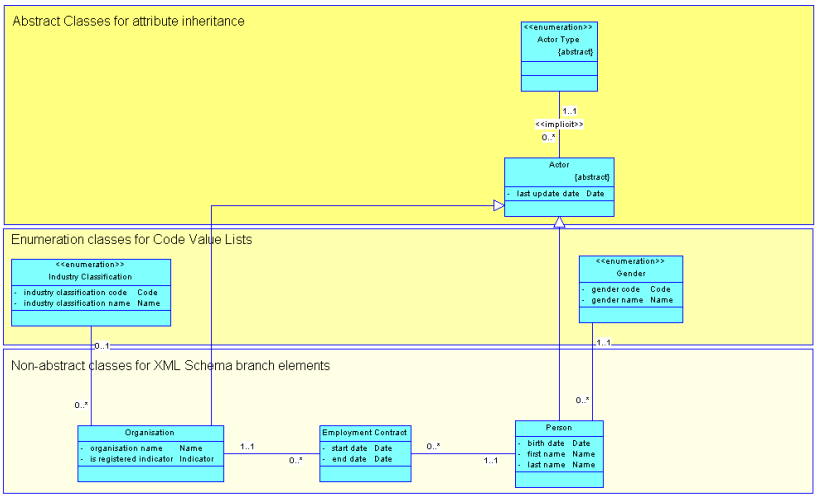
Topic	See Page
UML class diagram	6
UML domains	6
UML classes	7
UML attributes	7
UML associations	8
UML generalisations	8
UML naming of classes and attributes	9
UML meta model extensions	9
UML model validation	11

Continued on next page

Defining a UML semantic data model, Continued

UML class diagram

The UML class diagram is the graphic representation of the semantic data model. A simplified example:



The individual model components are described below.

UML domains

The UML domains are used to enforce the consistent handling of data types. They therefore exactly mirror the XML data types defined by UN/CEFACT and published as file 'UnqualifiedDataTypeSchemaModule-2.0.xsd'. This XSD file is shipped with the OASIS UBL 2.0 package.

- Amount
- Binary Object
- Code
- Date
- DateTime
- Graphic
- Identifier
- Indicator
- Measure
- Name
- Numeric
- Percent
- Picture
- Quantity
- Rate
- Sound
- Text
- Time
- Value
- Video

Additional domains should not be added to ascertain UN/CEFACT data type compliance, but may be added if necessary, provided some rules are adhered to.

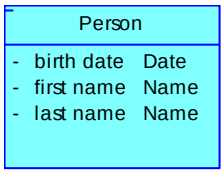
The UML domains are assigned to attributes and can be seen in the above diagram after each attribute name.

Continued on next page

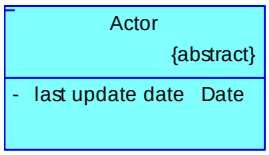
Defining a UML semantic data model, Continued

UML classes

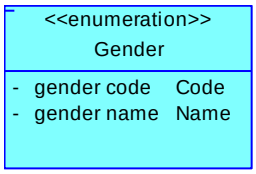
The UML class is a description of a set of objects that have a similar structure and share the same attributes, associations and semantics.



A class may be marked {abstract} if it is deemed insufficiently specific to be correctly interpreted. It therefore must not appear in an XML document Schema, but enables inheritance behind the scenes.



A class may have a stereotype of <<enumeration>> to indicate it is a list of named values, AKA code value list.



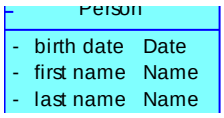
The structure of a class is described by attributes and associations.

Each class must have a clearly defined name that adheres to UBL/MXV NDR 2.1 rules.

Each class must have a clear business definition, stating its meaning without overlapping other classes meanings.

UML attributes

The UML attribute is a property of a class.



Each attribute must have a domain assigned.

Each attribute must have a clearly defined name that adheres to UBL/MXV NDR 2.1 rules.

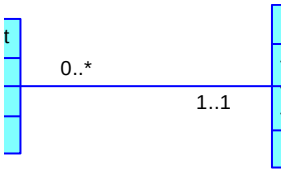
Each attribute must have a clear business definition, stating its meaning without overlapping other attribute meanings of this class.

Continued on next page

Defining a UML semantic data model, Continued

UML associations

A UML association represents a structural relationship between two classes.



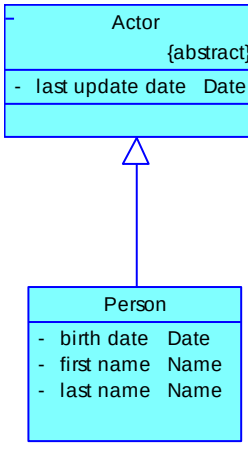
An association has cardinalities, AKA multiplicity, at either side.

Associations must be self-explanatory without a name. If a name was required to express their meaning, a separate class must be defined instead. Hence, 1-to-1 associations are not uncommon in a semantic data model.

Associations cannot have attributes. If attributes were required, a separate class must be defined instead.

UML generalisations

A UML generalisation is a relationship between a more general class (the parent) and a more specific class (the child). The parent class describes the common attributes and associations. The child class is fully consistent with the parent class, but contains additional attributes and/or associations.



The parent class may or may not be abstract.

Several layers of generalisations may be stacked.

Continued on next page

Defining a UML semantic data model, Continued

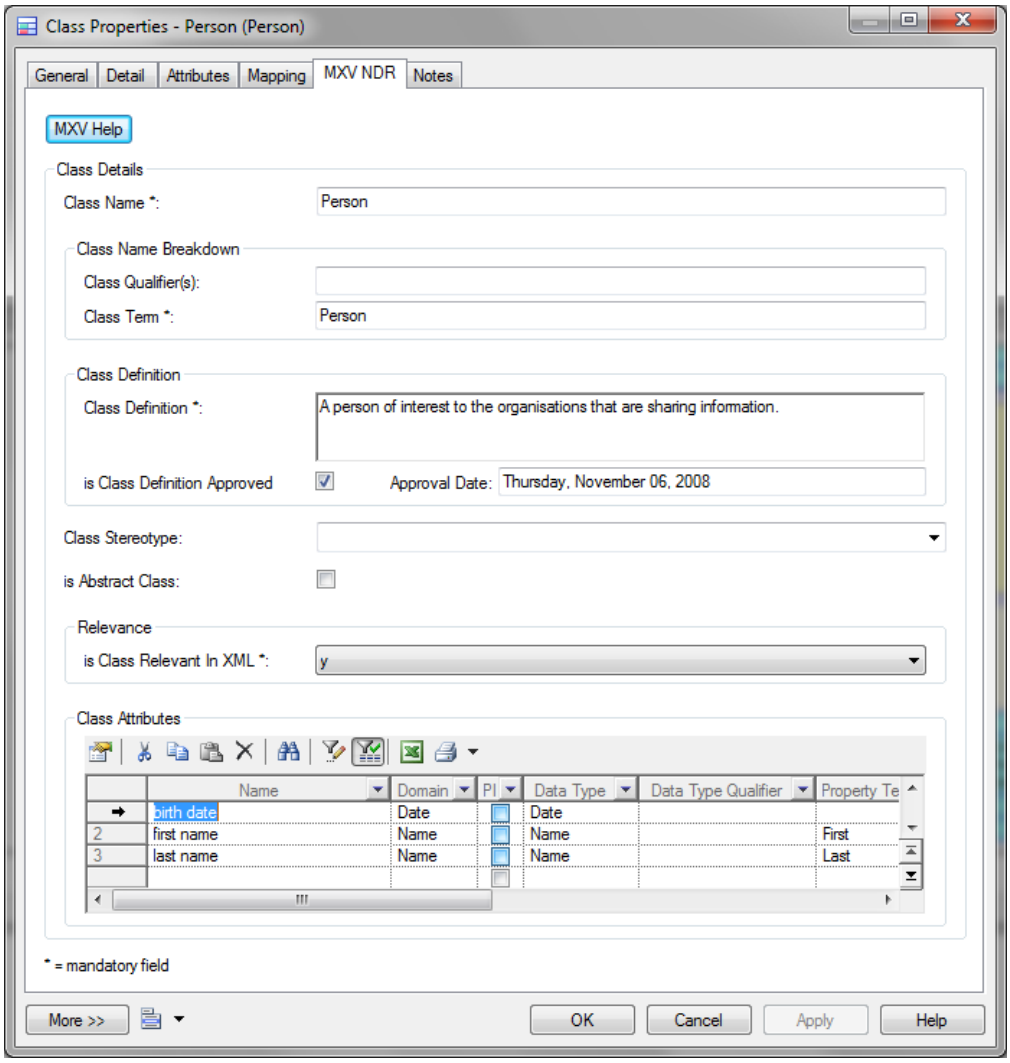
UML naming of classes and attributes

The naming of UML classes and attributes must adhere to UBL/MXV NDR 2.1 rules, which is based on the UN/CEFACT data types and ISO11179-5 standards.

UML meta model extensions

In order to record all information required for implementing the UML semantic model as UBL/MXV NDR 2.1 compliant XML library schemas, the UML model requires meta model extensions.

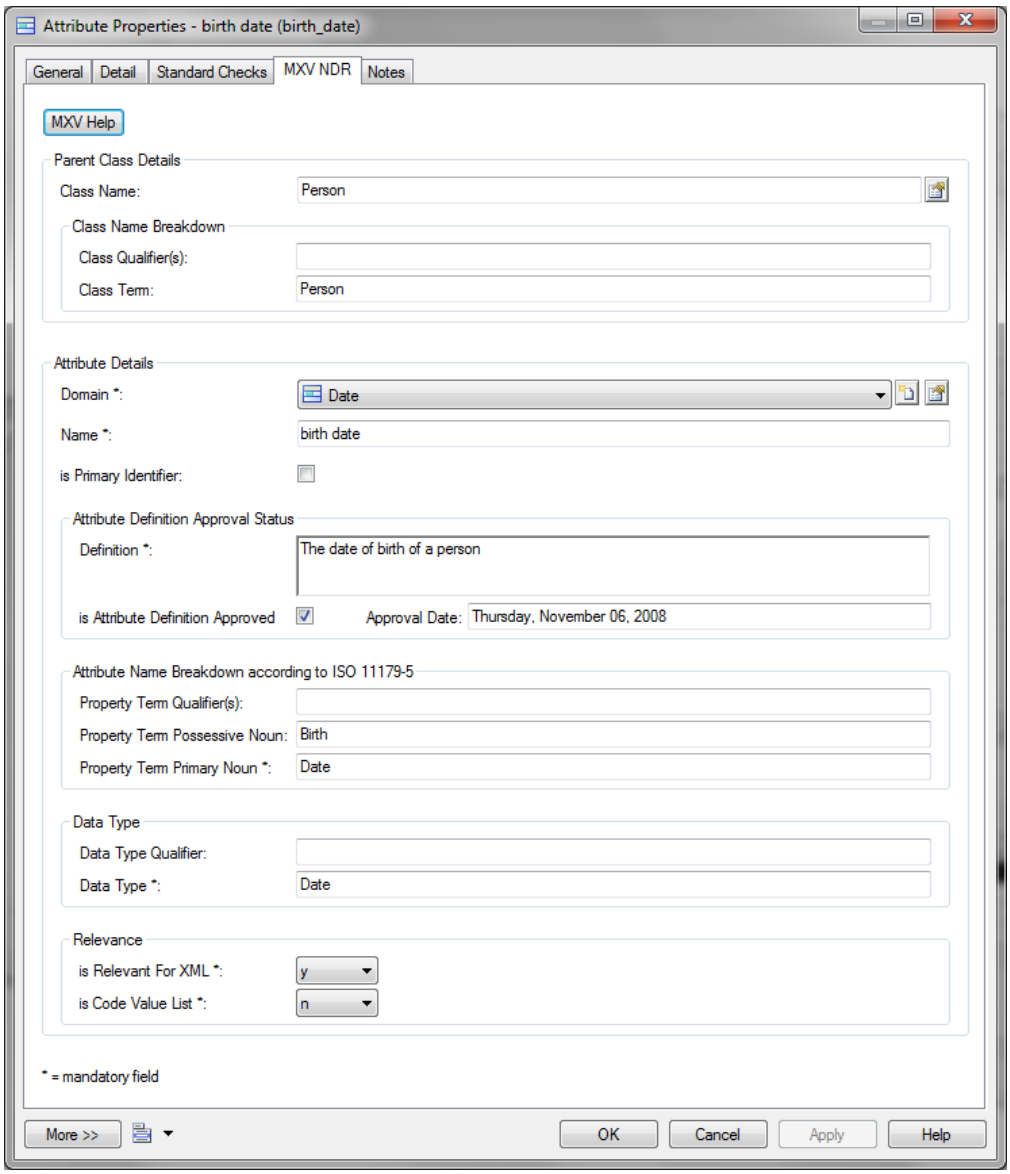
For classes, the following extensions have been defined:



Continued on next page

Defining a UML semantic data model, Continued

For attributes, the following extensions have been defined:



These meta model extensions have been implemented by Data Management Solutions, www.d-m-s.co.nz, using Sybase PowerDesigner's customisation feature.

These meta model extensions can be used in any industry, with only minor initial setup adjustments required. For more details, contact Data Management Solutions.

Continued on next page

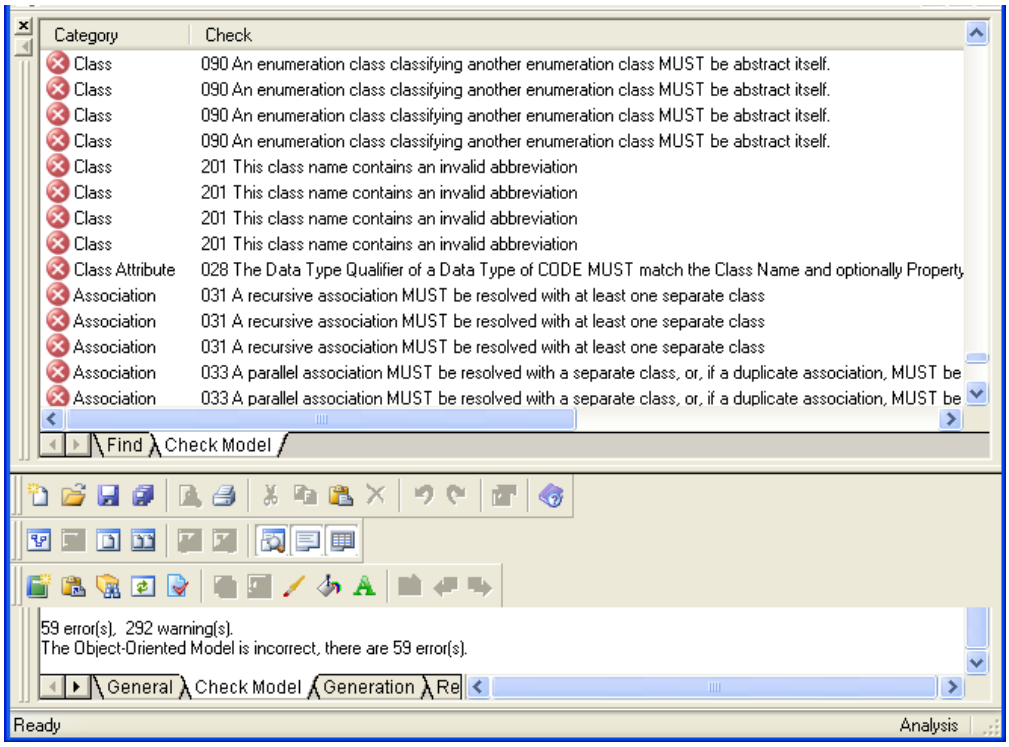
Defining a UML semantic data model, Continued

UML model validation

The UBL/MXV NDR 2.1 rules are focusing on the XML implementation, but many rules have an upstream effect on the UML semantic model: The earlier a rule violation is detected, the less administrative iterations are required to produce functioning XML deliverables.

The number and complexity of rules for a UML model to comply with quickly becomes difficult to manage manually.

To ease this burden on the model administrator, Data Management Solutions www.d-m-s.co.nz has to date developed over 70 automated UML model checks. These checks are executed at the push of a button, integrated into the PowerDesigner user interface.



These model checks can be used in any industry, with only minor initial setup adjustments required. For more details, contact Data Management Solutions.

Transitioning a UML model to an XML library model

Background

The UML semantic model drives the population of the XML library models, or at least large parts of it.

The XML library consists of 4 main modules:

Unqualified Data Types (UDT) library module: supplied by UN/CEFACT
Qualified Data Types (QDT): compliant with UBL/MXV NDR 2.1 rules
Common Basis Components (CBC): compliant with UBL/MXV NDR 2.1 rules
Common Aggregate Components (CAC): compliant with UBL/MXV NDR 2.1 rules

The latter three XML library modules are populated according the UML semantic data model content.

In this section

The transition from the single UML semantic model to the four XML library module models:

Topic	See Page
Unqualified Data Types UDT	13
Qualified Data Types QDT	14
Common Basic Components CBC	14
Common Aggregate Components CAC	15
XML meta model extensions	16
Model administration automation	19

Continued on next page

Transitioning a UML model to an XML library model,
Continued

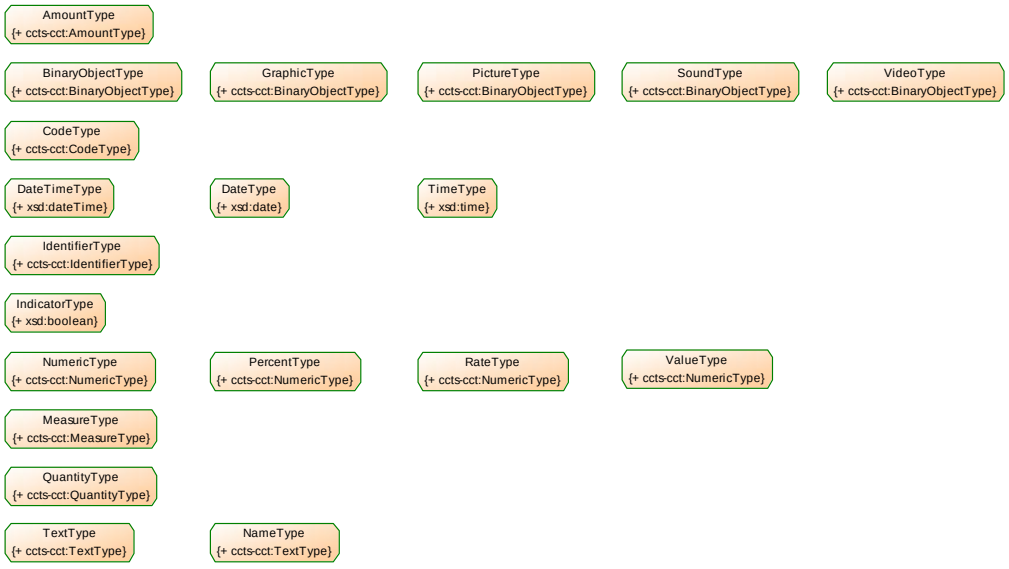
Unqualified Data Types UDT

The UBL UDT XML library module is maintained and supplied by OASIS in form of an XSD file and extends the UN/CEFACT CCTS Core Component Types CCT 2.1.

The UDT XSD file has been reverse-engineered using PowerDesigner in order to allow other XML models to reference the UDT model.

The UDT XSD file is the master for keeping the UDT model and the UML domains in synch.

The UDT XML model diagram:



Continued on next page

Transitioning a UML model to an XML library model,
Continued

Qualified Data Types QDT

The QDT XML library module is maintained and supplied by the organisation that maintains and owns the UML semantic data model.

The QDT XML library module model is created from scratch, based on input from the UML semantic model:

All qualified data types (a meta model extension) in the UML semantic model are transitioned into the QDT XML library module model and must be kept in synch.

Unless specialised sub-domains are defined for a UN/CEFACT domain in the UML model, the QDT model remains empty.

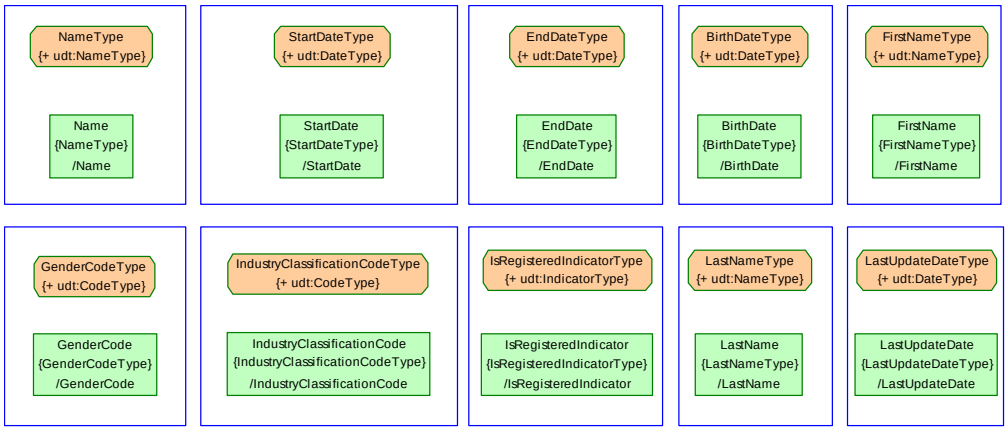
Common Basic Components CBC

The CBC XML library module is maintained and supplied by the organisation that maintains and owns the UML semantic data model.

The CBC XML library module model is created from scratch, based on input from the UML semantic model:

All attributes that are marked XML relevant (a meta model extension) in the UML semantic model are transitioned into the CBC XML library module model and must be kept in synch.

The example CBC XML model diagram:



Continued on next page

Transitioning a UML model to an XML library model,
Continued

Common Aggregate Components CAC

The CAC XML library module is maintained and supplied by the organisation that maintains and owns the UML semantic data model.

The CAC XML library module model is created from scratch, based on input from the UML semantic model:

All classes that are marked XML relevant in the UML semantic model are transitioned into the CAC XML library module model and must be kept in synch.

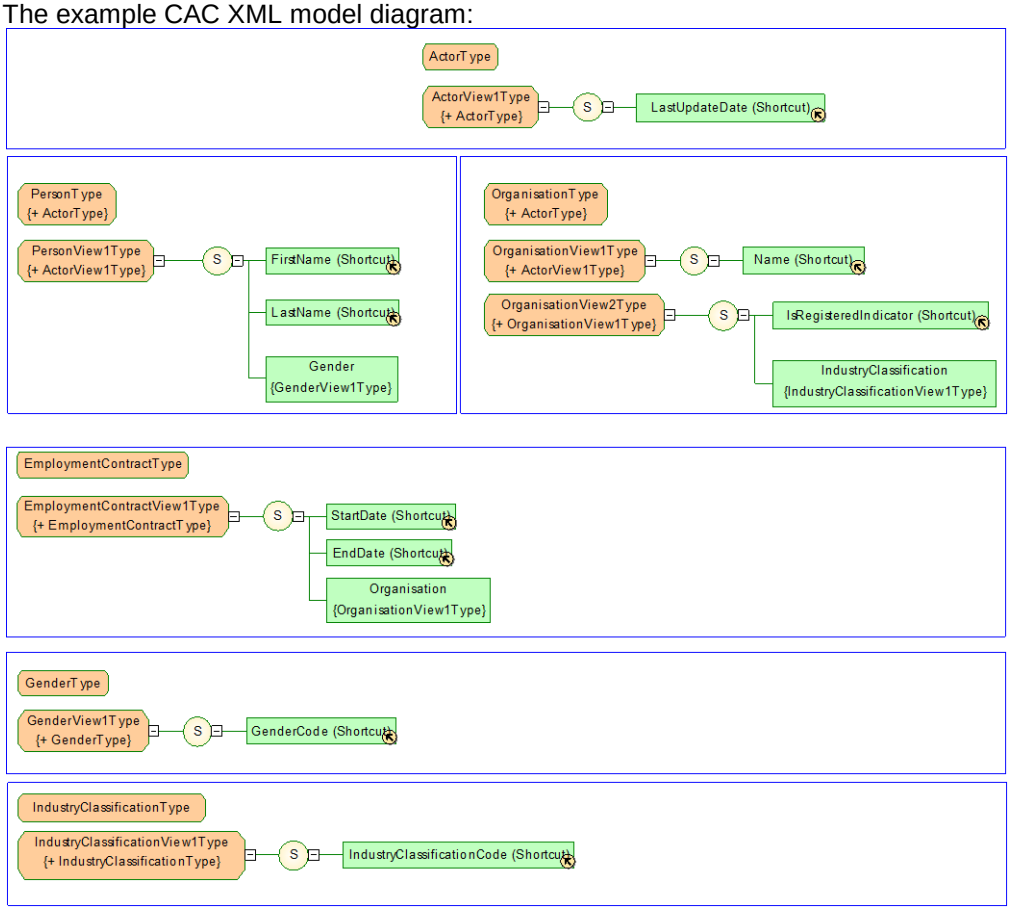
Abstract classes become abstract complexTypes.

The parent class of a specialisation child class becomes the extension base of the child complexType.

Additional 'complexType views' may be created to define complexTypes with varying, but prescriptive content for the same base UML class.[2]

Any view that is deemed re-usable may or should be defined in the CAC model.

The example CAC XML model diagram:



Continued on next page

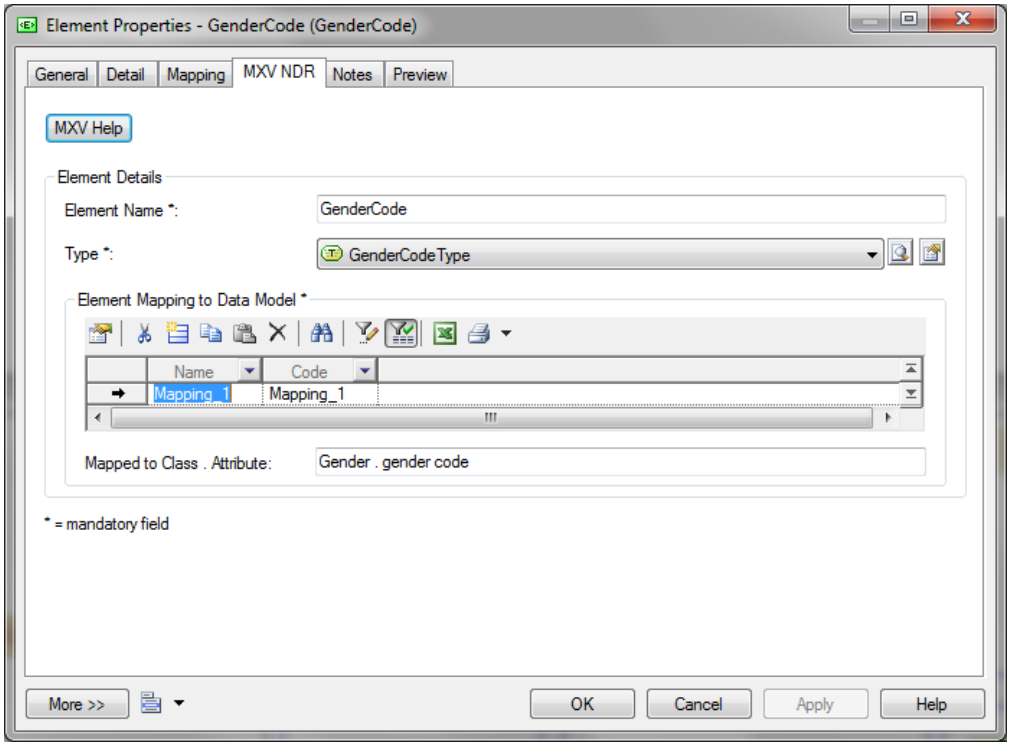
[2] This feature allows various prescriptive constructs for a single class (not part of UBL NDR)

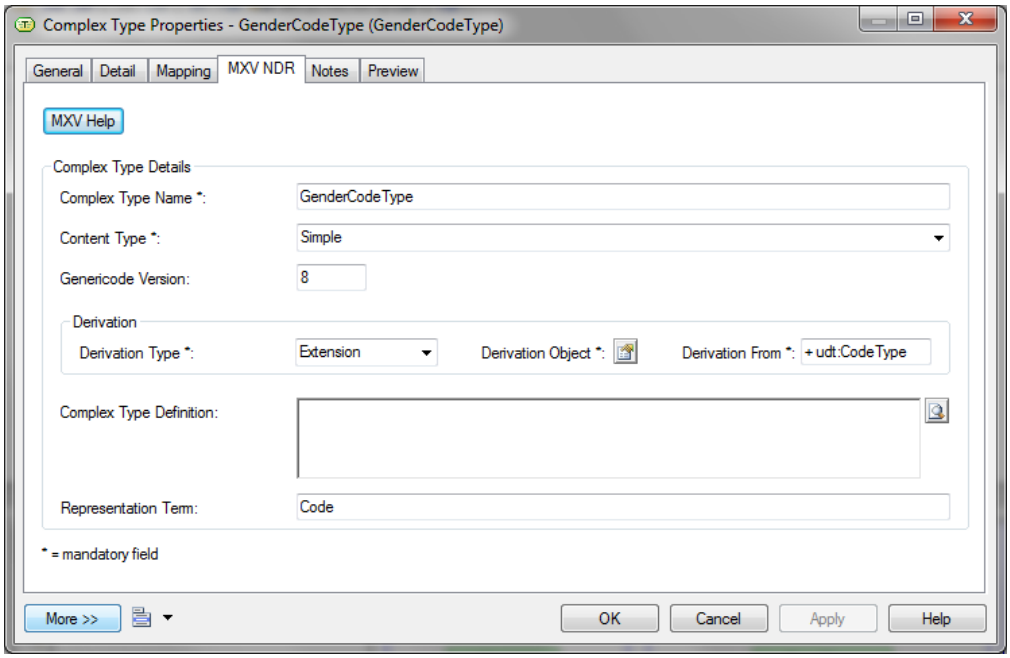
Transitioning a UML model to an XML library model,
Continued

XML meta model extensions

In order to record all information required for implementing UBL/MXV NDR 2.1 compliant XML library and schemas, the XML models require meta model extensions depending on their model type.

Examples:
The following extensions have been defined for a CBC model:



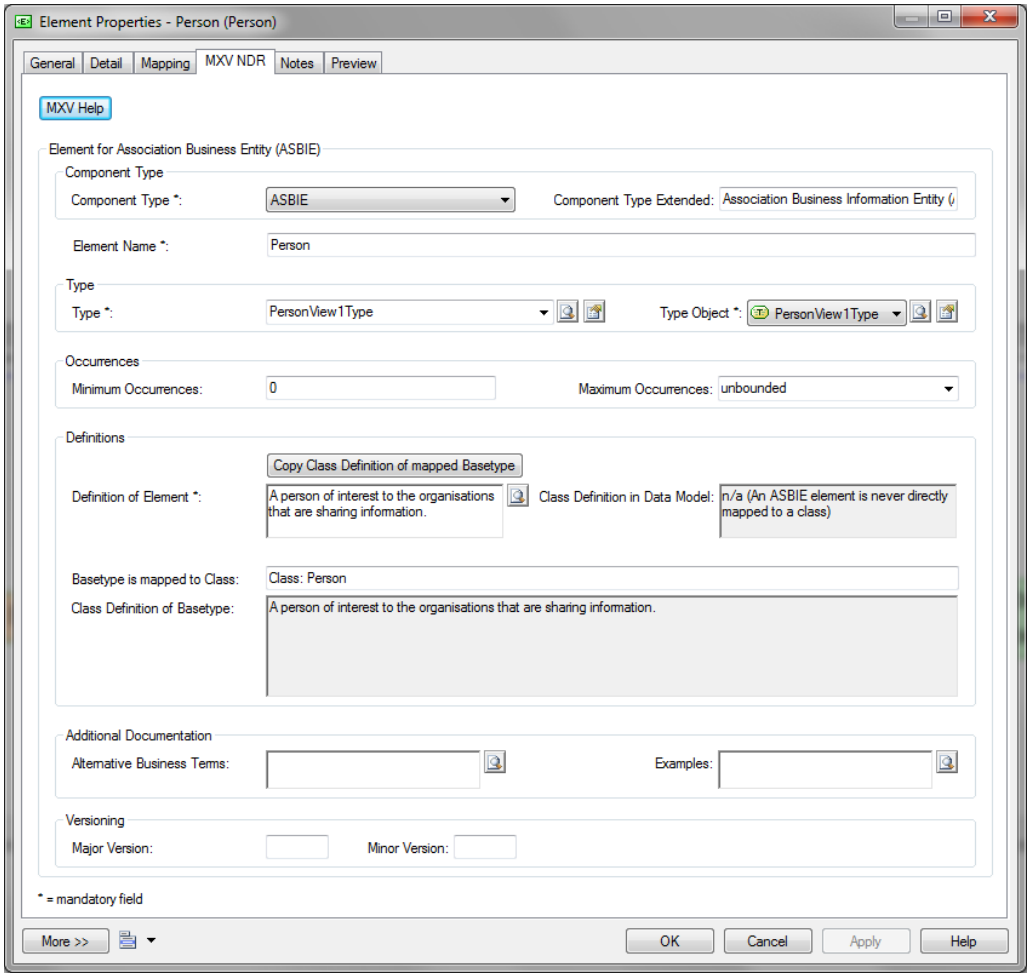


Continued on next page

Transitioning a UML model to an XML library model,
Continued

The following element extensions have been defined for a CAC or DOC model:

For example, for an Association Business Information Entity (ASBIE) element:

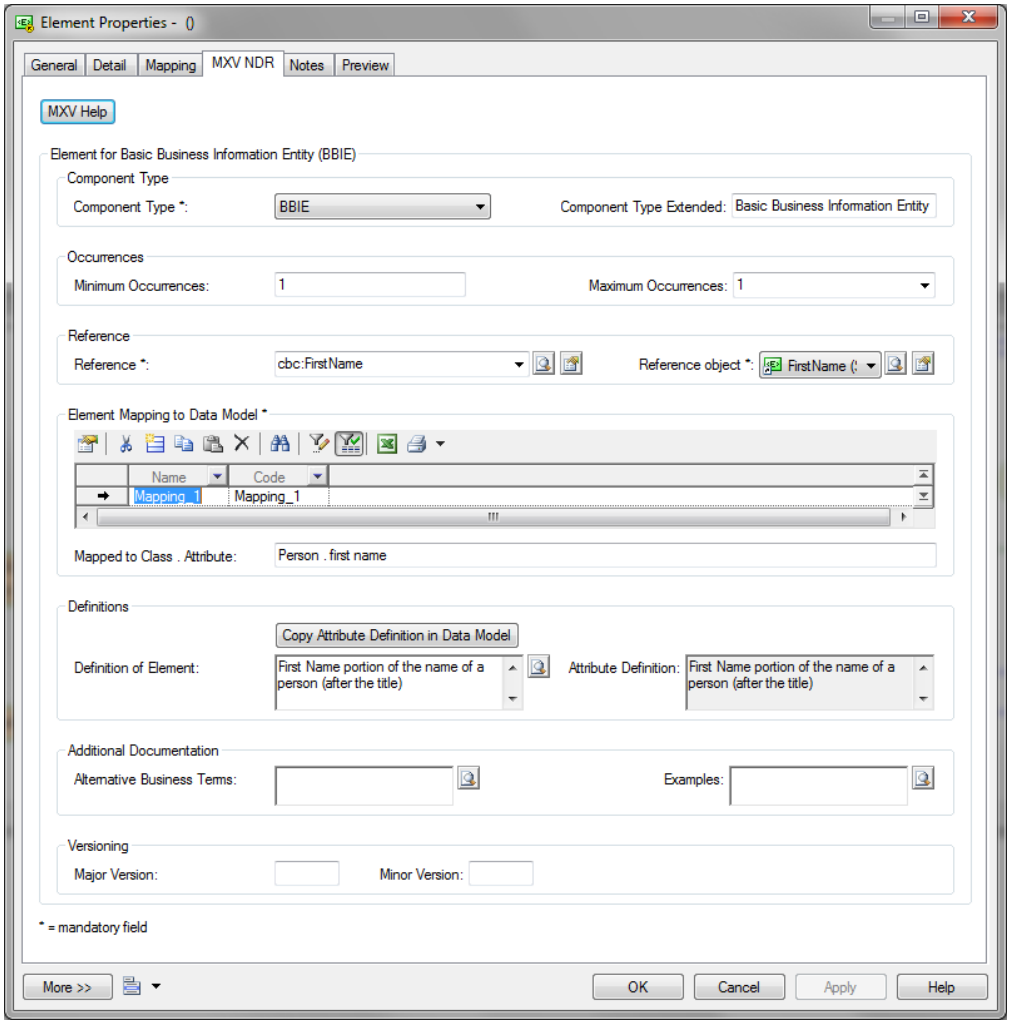


and …

Continued on next page

Transitioning a UML model to an XML library model,
Continued

For a Basic Business Information Entity (BBIE) element:



Continued on next page

Transitioning a UML model to an XML library model,
Continued

Model administration automation

Data Management Solutions, www.d-m-s.co.nz has developed automation tools for as many of the model administration tasks as is feasible. The software is being deployed by the New Zealand Ministry of Education since December 2007. Automated tasks for library models are:

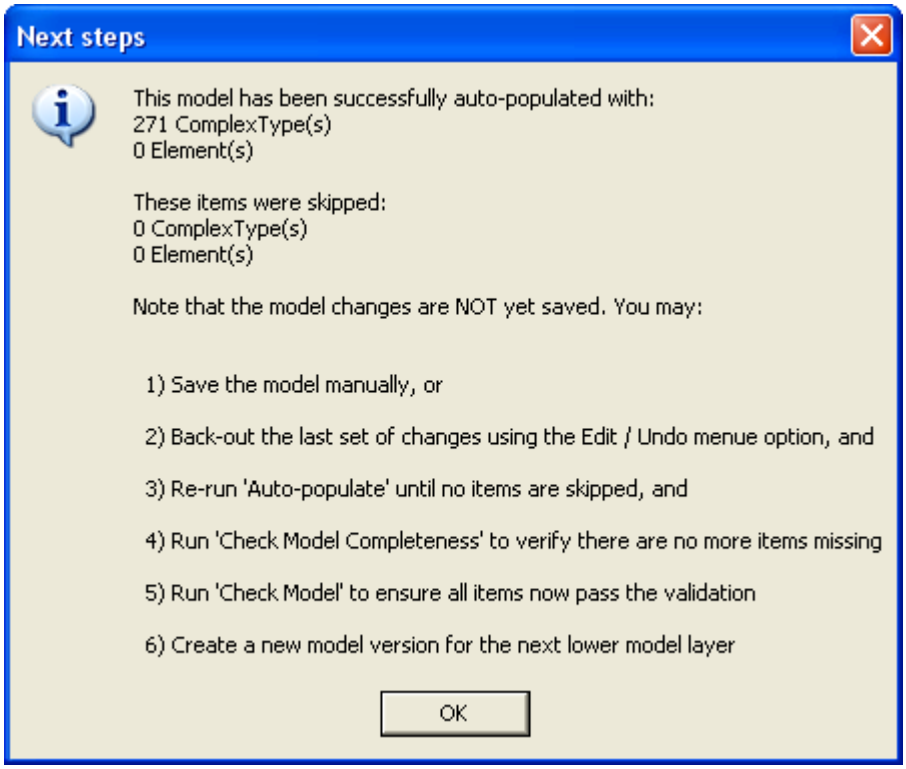
- Checking a QDT, CBC and CAC model
- Checking the completeness of mandatory objects of QDT, CBC and CAC models
- Sorting of QDT, CBC and CAC model objects
- Auto-populating mandatory objects of CBC and CAC models
- Generating QDT, CBC, CAC and Document XML Schema files
- Migrating draft models to specification status
- Creating a new UML or XML model version
- Cloning a CAC View

These tools are integrated into the PowerDesigner user interface. For example, the MXV NDR CAC Tools menu:

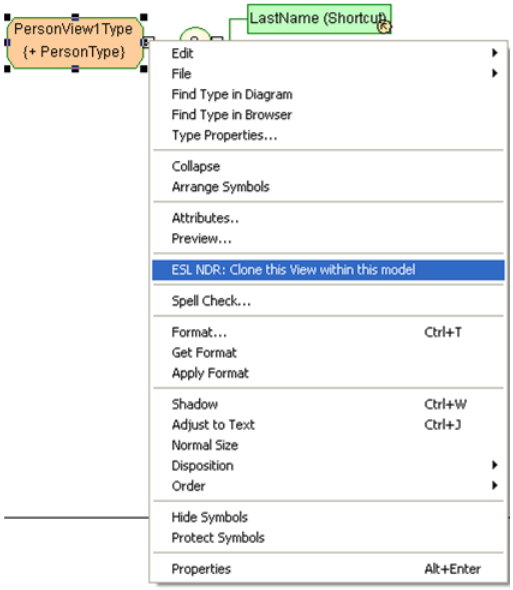
Continued on next page

Transitioning a UML model to an XML library model,
Continued

An example of a summary window after auto-populating a CAC model:



An example of the integrated menu for cloning a CAC view:



These automation tools can be used in any industry, with only minor initial setup adjustments required. For more details, contact Data Management Solutions.

Composing the XML document schema model

Background

The messaging business requirements define the scope of every XML document schema model.

However, no XML tag names may be 'invented' at this stage. Instead, the XML document schema is composed by cherry-picking XML components from the XML library, which have been populated according to the agreed UML semantic model.

Should the XML document schema require components that are not defined yet in the UML model, then firstly the UML model needs to be agreed and expanded, secondly the XML library modules need to reflect that UML expansion, and thirdly only now can the XML document schema be composed using approved components. The price of interoperability!

Such evolution naturally requires adequate model and schema versioning. UBL NDR 2.1 defines the versioning rules for XSD files, which obviously also affect the models that generate the XSD files.

In this section

The composition of a XML document schema model:

Topic	See Page
Document Schema model DOC	22
Model administration automation	23

Continued on next page

Composing the XML document schema model, Continued

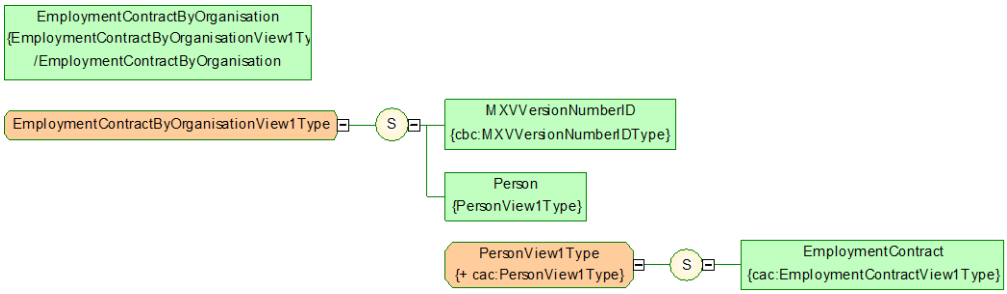
Document Schema model DOC

The DOC XML module is maintained and supplied by the organisation that maintains the UML semantic data model.

Each message type has its own DOC XML schema.

The DOC XML model is created from scratch, based on business requirements and in accordance with the UML semantic model:

The example DOC XML model diagram:



Note the reuse of CBC and CAC components.

This local model diagram does not depict the full extent of the document schema, because the library module components are not expanded. This is in line with the resulting XSD file, which makes references into the XML library XSD files only.

A graphical XML Schema editor is best suited to fully expand a XML structure that spans multiple XSD files. For details, see section 'Sample XML Document Schema'.

Continued on next page

Composing the XML document schema model, Continued

Model administration automation

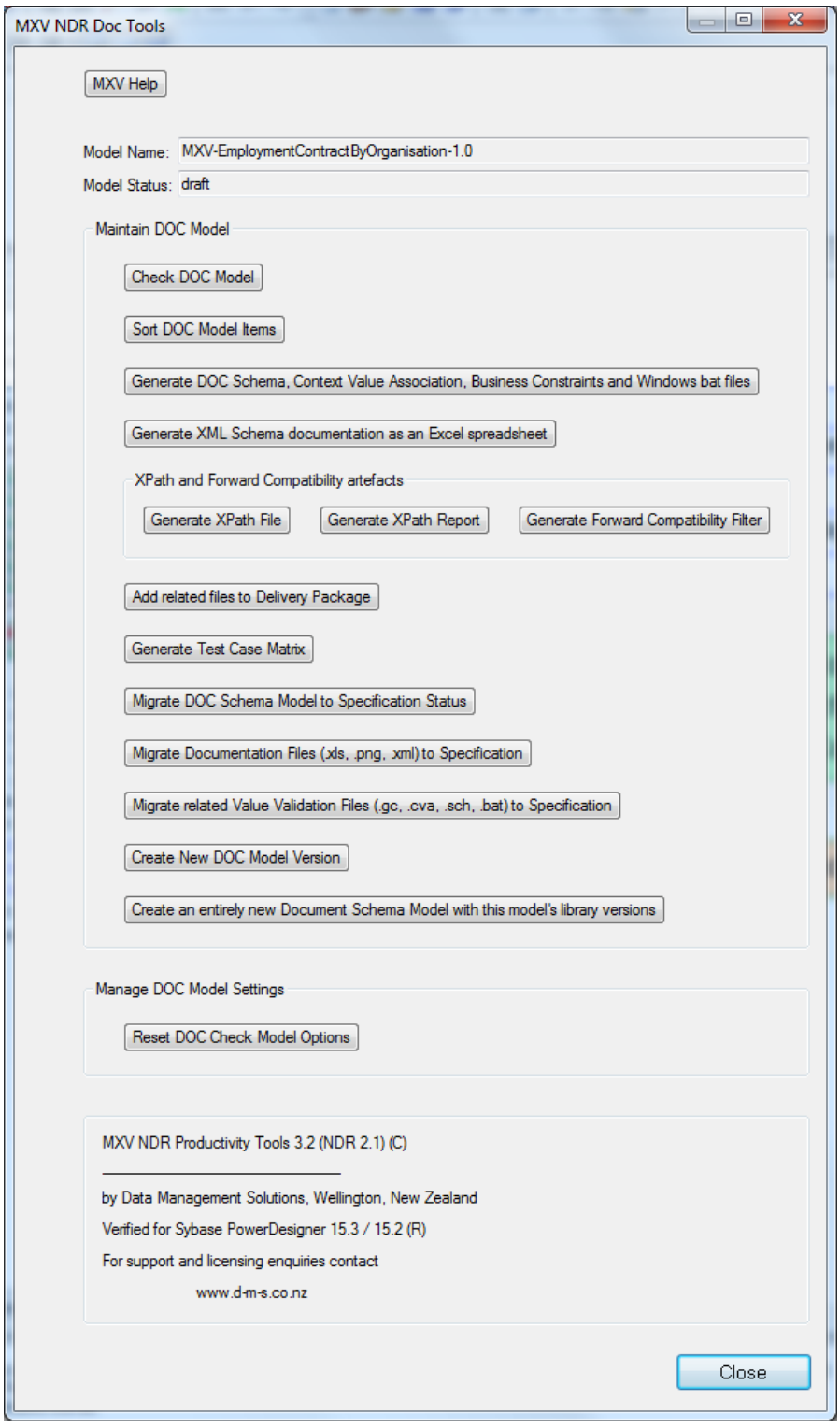
Data Management Solutions, www.d-m-s.co.nz has developed automation tools for as many of the model administration tasks as is feasible. Some of the software is being deployed by the New Zealand Ministry of Education since December 2007. Automated tasks for document schema models are:

- Checking Model
- Sorting Model Items
- Generate Document Schema
- Generate XML 2-pass Content Validation files:
  i.e. OASIS Genericode, Context Value Association, ISO Schematron skeletons
- Generate Schema documentation file (MS Excel format)
- Generate XPath XML file and XPath Report
- Generate Forward Compatibility Filter
- Adding related files to a standardised delivery package
- Generate Test Case Matrix
- Migrating draft model to specification status
- Migrating documentation files (Excel, diagrams, etc.) to specification status
- Migrating related XML 2-pass Content Validation files to specification status
- Creating a new document schema model version
- Creating an new document schema model with this model's library versions
- Cloning a document schema View
- Merging a document schema View a level up into CAC model

Continued on next page

Composing the XML document schema model, Continued

These MXV NDR DOC Tools are integrated into the PowerDesigner user interface:



These automation tools can be used in any industry, with only minor initial setup adjustments required. For more details, contact Data Management Solutions.

Validating an XML model

Background

Before XML models, both library and document schema models, can be used to generate correct and compliant XSD files, they must be valid and UBL/MXV NDR compliant.

In this section

The validation of XML models:

Topic	See Page
XML model validation	26

Continued on next page

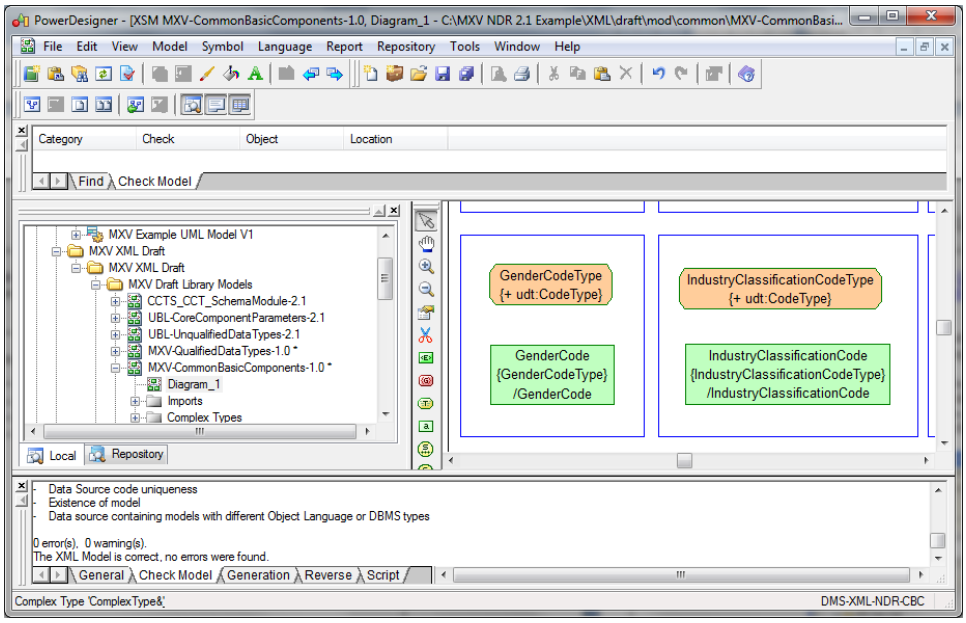
Validating an XML model, Continued

XML model validation

All XML models, such as QDT, CBC, CAC and DOC schema models, must comply with many UBL/MXV NDR rules if they are to generate correct and compliant XSD files.

The number and complexity of rules for a XML model to comply with quickly becomes difficult to manage manually.

To ease this burden on the model administrator, Data Management Solutions www.d-m-s.co.nz has to date developed over 220 automated XML model checks. These checks are executed at the push of a button, integrated into the PowerDesigner user interface.



For some errors and warnings, the user may select to fix this error automatically. The remaining violations require manual intervention.

These XML model checks can be used in any industry, with only minor initial setup adjustments required. For more details, contact Data Management Solutions.

Generating code from XML models

Background

Once the XML models are complete and validated, they may be used to generate various XML deliverables.

In this section

Generating deliverables from XML models:

Topic	See Page
Generating XSD files	28
Generating Genericode skeleton	29
Generating Context Value Association file skeleton	32
Generating Business Constraints file skeleton	33
Generating Forward Compatibility Filter	34

Continued on next page

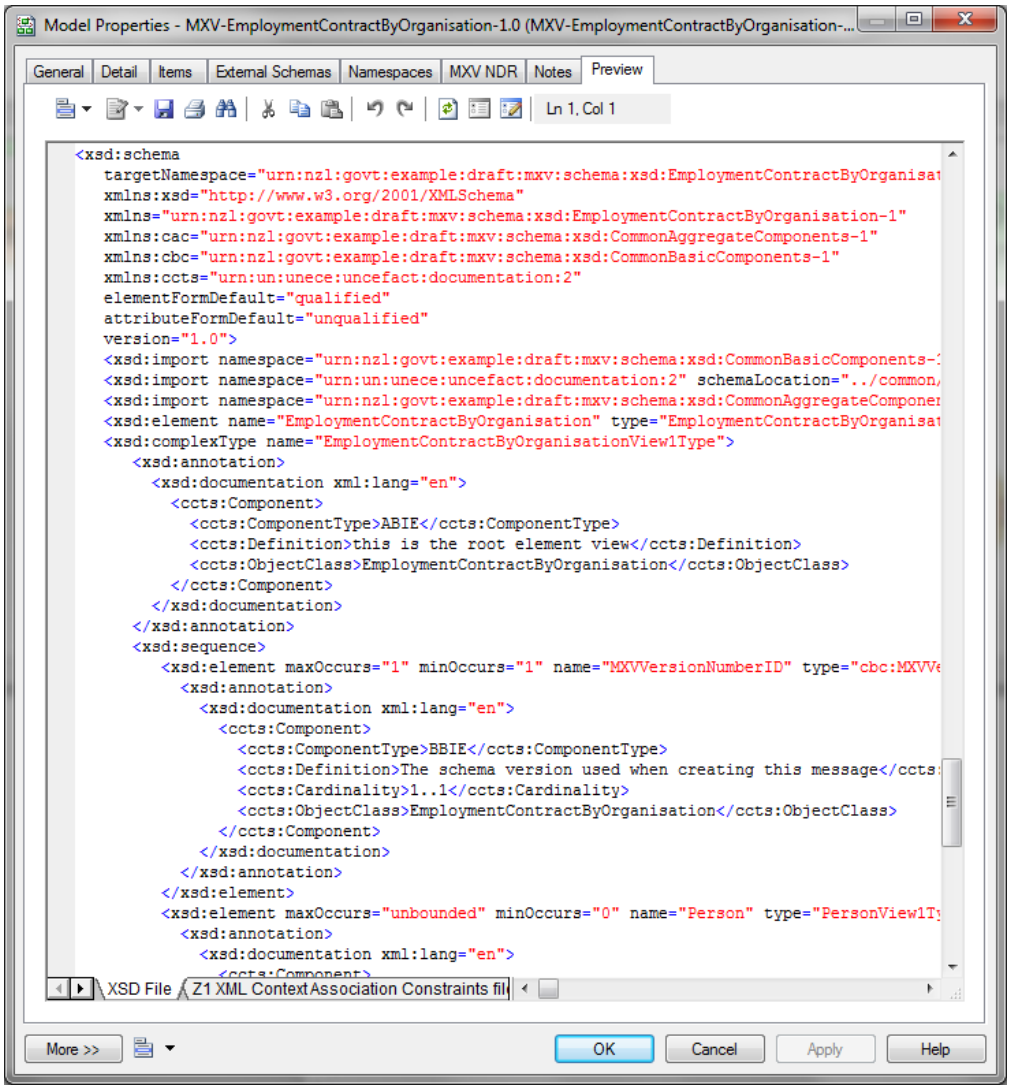
Generating code from XML models, Continued

Generating XSD files

The QDT, CBC, CAC and DOC models may be used to generate the XML Schema XSD files.

The PowerDesigner generator feature has been customised by Data Management Solutions to maximise the XSD files' UBL NDR look and feel, while minimising user input.

Generated XML Schema file (XSD):



These XML Schema generator customisations can be used in any industry, with only minor initial setup adjustments required. For more details, contact Data Management Solutions.

Continued on next page

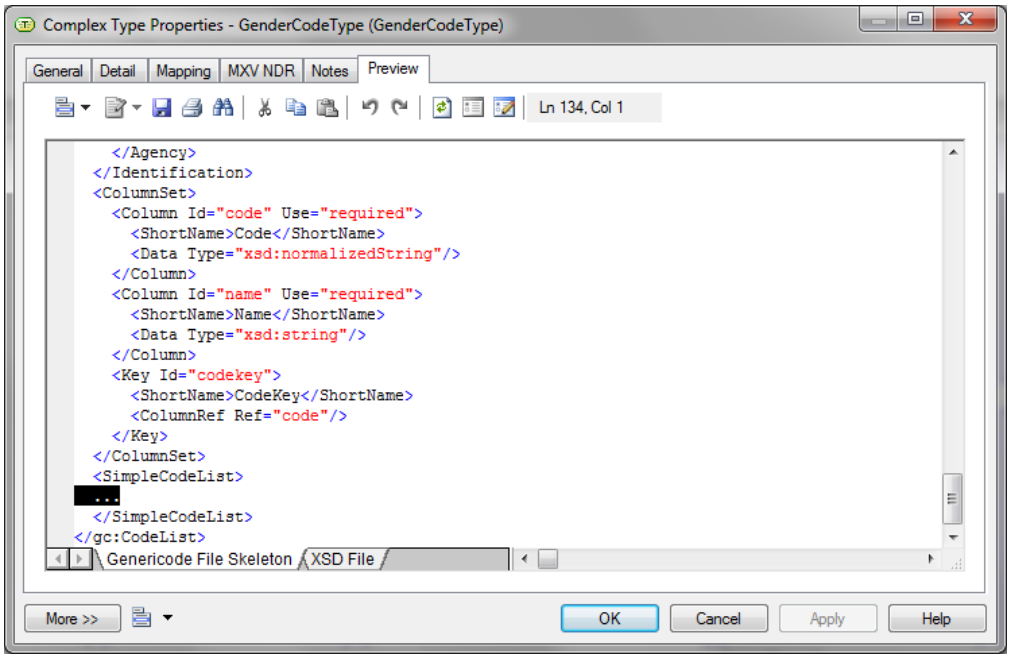
Generating code from XML models, Continued

Generating Genericode skeleton

Should an organisation wish to complement the UBL/MXV NDR methodology with the XML 2-pass content validation methodology for XML instance documents, then the QDT model may be used to generate a OASIS Genericode compliant skeleton file for each code defined in a QDT model.

The PowerDesigner generator feature has been customised by Data Management Solutions to create a Genericode compliant file.

Example of a generated Genericode skeleton file:



The administrator now only needs to fill in the '…' lines with the valid codes and names:

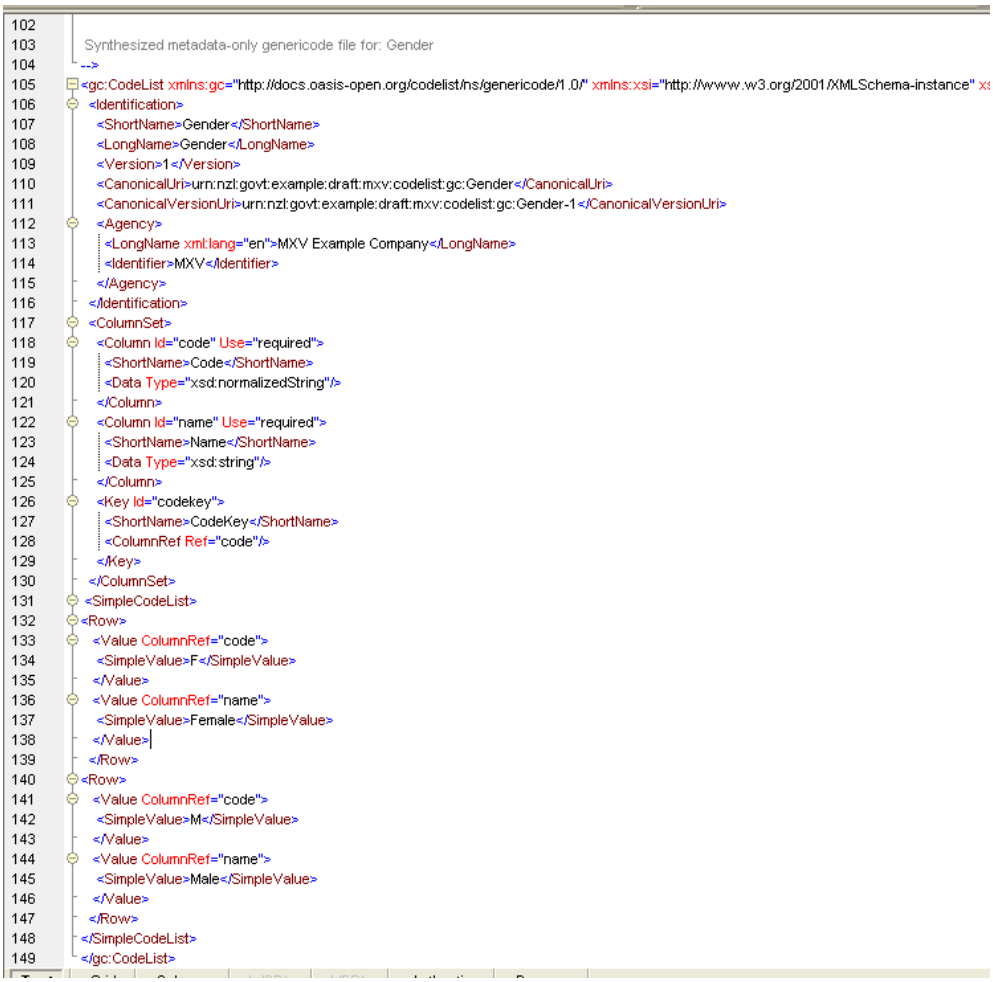
```
<Row>
        <Value ColumnRef="code">
                <SimpleValue>F</SimpleValue>
        </Value>
        <Value ColumnRef="name">
                <SimpleValue>Female</SimpleValue>
        </Value>
</Row>
<Row>
        <Value ColumnRef="code">
                <SimpleValue>M</SimpleValue>
        </Value>
        <Value ColumnRef="name">
                <SimpleValue>Male</SimpleValue>
        </Value>
</Row>
```

Alternatively, these may be imported from the preceding Genericode file version.

Continued on next page

Generating code from XML models, Continued

The resulting Genericode file is now complete, and may be rendered 'as-is' as XML file:



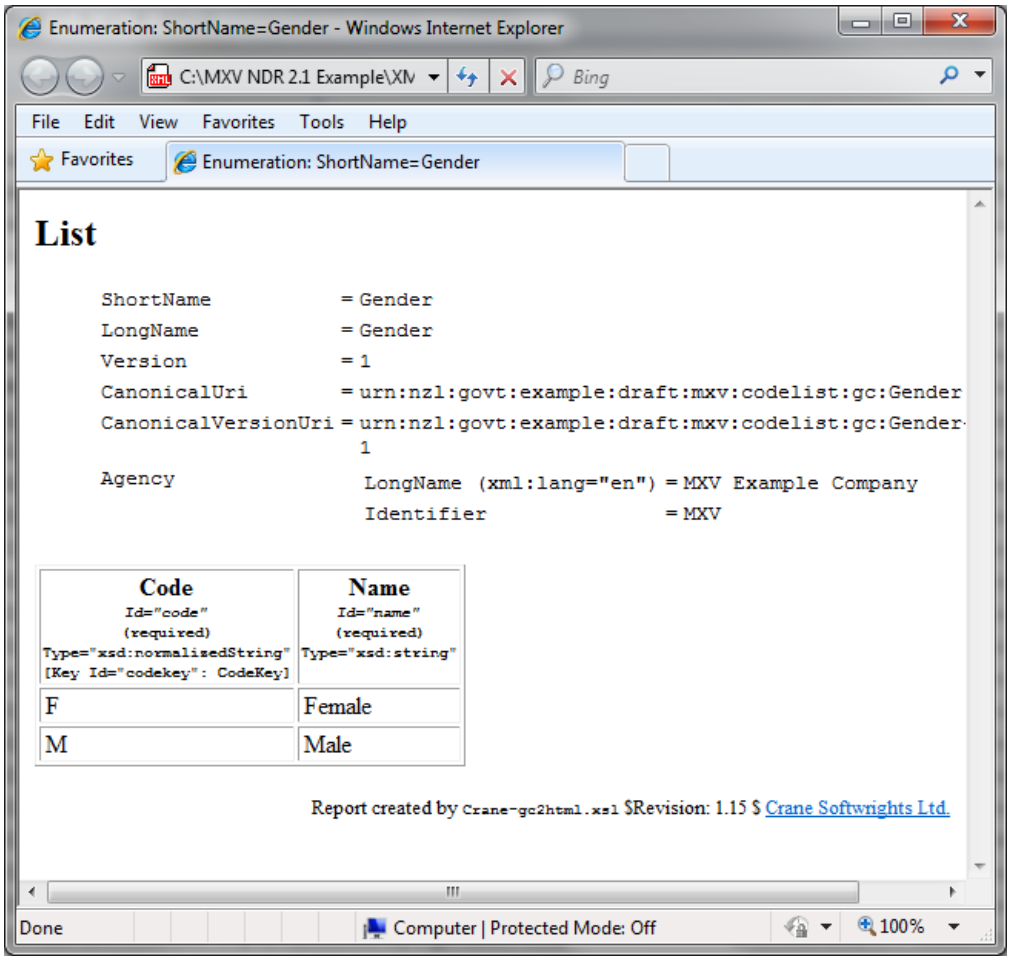
```
<!--
    Synthesized metadata-only genericode file for: Gender
-->
<gc:CodeList xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xs
  <Identification>
    <ShortName>Gender</ShortName>
    <LongName>Gender</LongName>
    <Version>1</Version>
    <CanonicalUri>urn:nzl:govt:example:draft:mxv:codelist:gc:Gender</CanonicalUri>
    <CanonicalVersionUri>urn:nzl:govt:example:draft:mxv:codelist:gc:Gender-1</CanonicalVersionUri>
    <Agency>
      <LongName xml:lang="en">MXV Example Company</LongName>
      <Identifier>MXV</Identifier>
    </Agency>
  </Identification>
  <ColumnSet>
    <Column Id="code" Use="required">
      <ShortName>Code</ShortName>
      <Data Type="xsd:normalizedString"/>
    </Column>
    <Column Id="name" Use="required">
      <ShortName>Name</ShortName>
      <Data Type="xsd:string"/>
    </Column>
    <Key Id="codekey">
      <ShortName>CodeKey</ShortName>
      <ColumnRef Ref="code"/>
    </Key>
  </ColumnSet>
  <SimpleCodeList>
  <Row>
    <Value ColumnRef="code">
      <SimpleValue>F</SimpleValue>
    </Value>
    <Value ColumnRef="name">
      <SimpleValue>Female</SimpleValue>
    </Value>
  </Row>
  <Row>
    <Value ColumnRef="code">
      <SimpleValue>M</SimpleValue>
    </Value>
    <Value ColumnRef="name">
      <SimpleValue>Male</SimpleValue>
    </Value>
  </Row>
  </SimpleCodeList>
</gc:CodeList>
```

… or …

Continued on next page

Generating code from XML models, Continued

Alternatively, the same Genericode file may be rendered in a user-friendly format using a stylesheet kindly provided by Crane Softwrights, http://cranesoftwrights.com/:



These Genericode generator customisations can be used in any industry, with only minor initial setup adjustments required. For more details, contact Data Management Solutions.

Continued on next page

Generating code from XML models, Continued

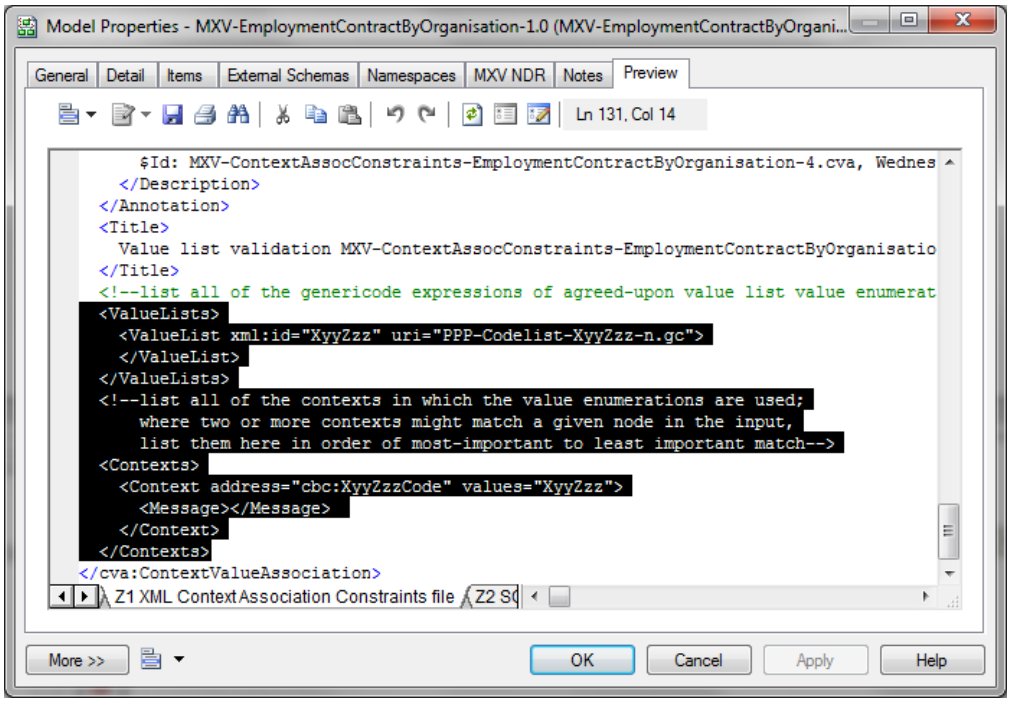
Generating Context Value Association file skeleton

Should an organisation wish to complement the UBL/MXV NDR methodology with the XML 2-pass content validation methodology for XML instance documents, then a DOC model may be used to generate an OASIS Context Value Association file skeleton.

The Context Value Association file defines which Genericode file is applied in which XPath context within a XML document schema.

The PowerDesigner generator feature has been customised by Data Management Solutions to create an OASIS Context Value Association compliant skeleton file.

Example of a generated Context Value Association skeleton file:



The administrator now only needs to fill in the required Genericode file references and contexts.
Alternatively, these may be automatically imported from the preceding Context Value Association file version.

These Context Value Association generator customisations can be used in any industry, with only minor initial setup adjustments required. For more details, contact Data Management Solutions.

Continued on next page

Generating code from XML models, Continued

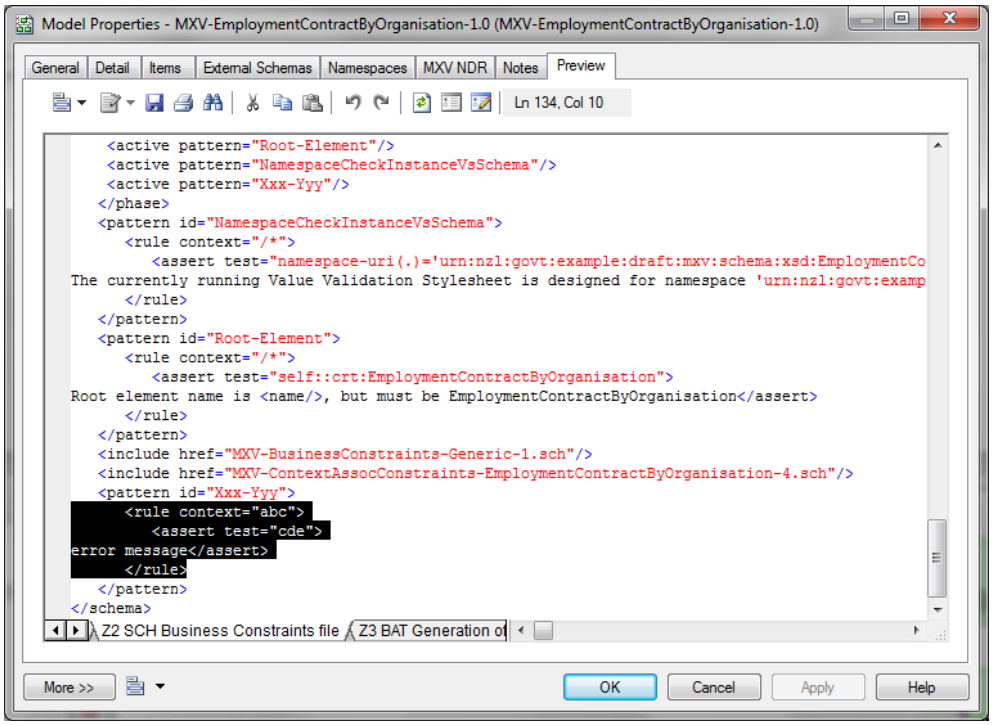
Generating Business Constraints file skeleton

Should an organisation wish to complement the UBL/MXV NDR methodology with the XML 2-pass content validation methodology for XML instance documents, then a DOC model may be used to generate an Business Constraints file skeleton.

The Business Constraints file defines more complex business rules using the ISO Schematron language.

The PowerDesigner generator feature has been customised by Data Management Solutions to create a Business Constraints skeleton file suitable for 2-pass validation.

Generated Business Constraints skeleton file:



The administrator now only needs to fill in the required patterns, rules and error messages.
Alternatively, these may be automatically imported from the preceding Business Constraints file version.

These Business Constraints generator customisations can be used in any industry, with only minor initial setup adjustments required. For more details, contact Data Management Solutions.

Continued on next page

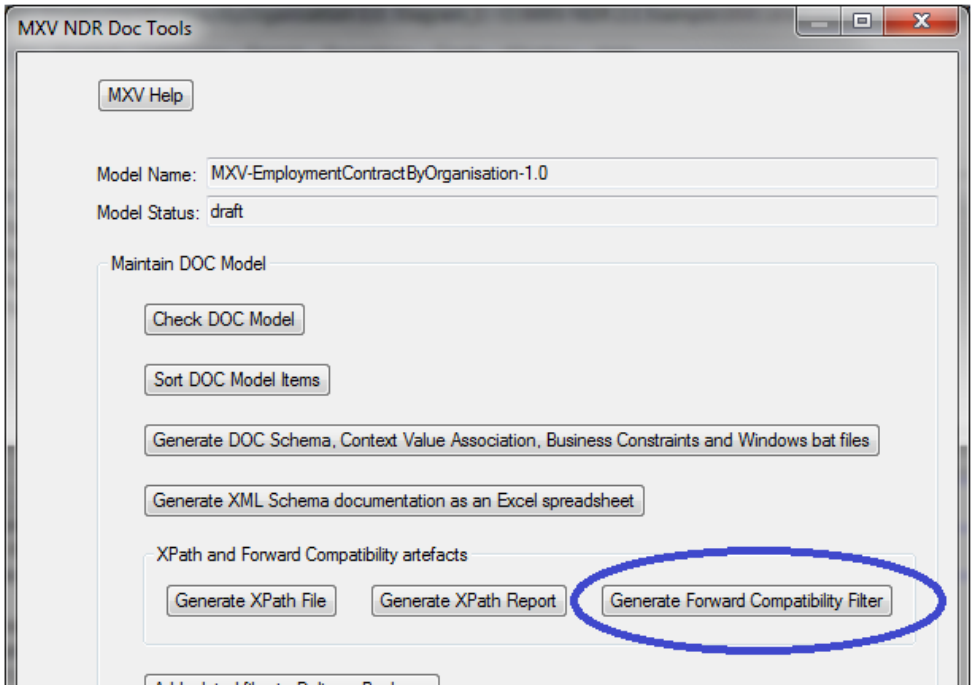
Generating code from XML models, Continued

Generating Forward Compatibility Filter

Should an organisation expect to have multiple minor versions of a document schema in concurrent use by their exchange partners, backward and forward compatible processing of these minor versions may be required.

Backward compatibility of prior minor versions is built into MXV by NDR design and versioning rules.
Forward compatibility with newer minor versions however requires special attention.

An optional plug-in generates a filter XSLT stylesheet which provides forward compatibility for instances based on newer minor versions than installed at the processing organisation.



Forward compatibility is achieved by pruning branches and elements from the instance.
The filter is producing a copy of the original instance, and eliminates all branches and elements not known to the currently implemented minor version schema.
If the filtered instance is schema valid, the filtered instance may be processed as if it was based on the same minor version that is currently installed.
Otherwise, a filtered and still invalid instance indicates issues beyond the scope of forward compatibility.

The filters are derived from similar work for UBL by Crane Softwrights.

Note that filtering does require a special processing model (included in the generated MXV Windows Command files)

Sample XML Schemas and outputs

Background

The sample schemas described in this paper are available both as a diagram and as XSD files.

In this section

Sample Schemas:

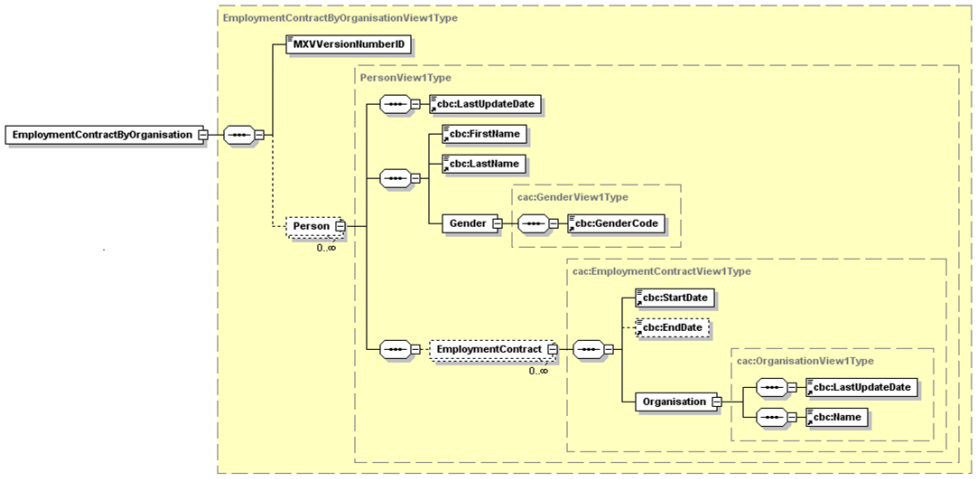
Topic	See Page
Sample XML Document Schema	36
Schema Documentation	36
Sample XSD files	36
Sample XPath report	37
Sample XPath XML file	37

Continued on next page

Sample XML Schemas and outputs, Continued

Sample XML Document Schema

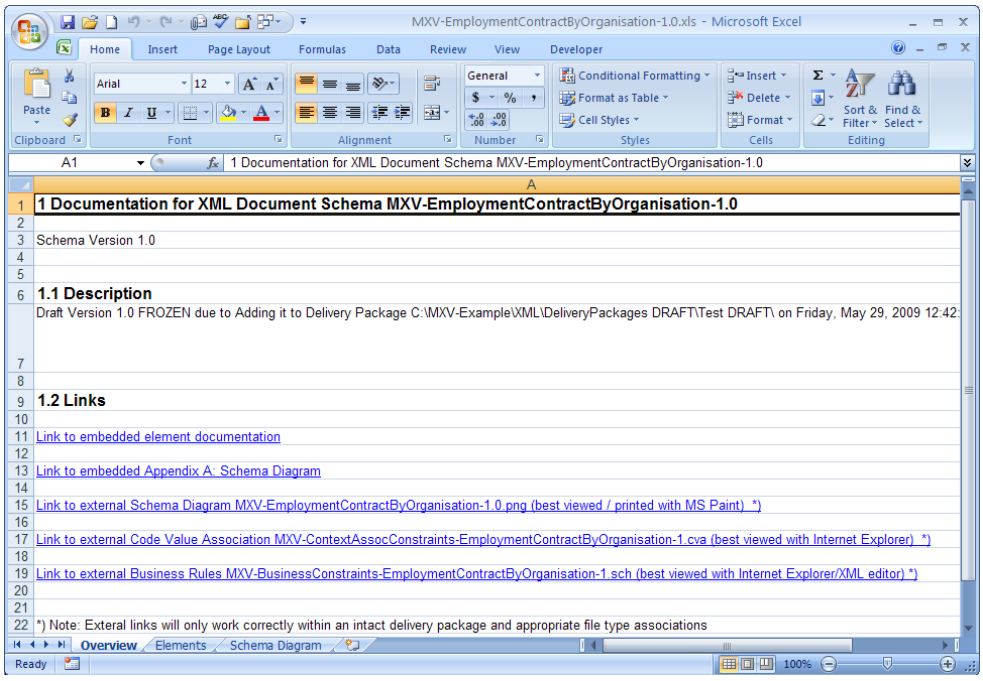
The sample XML schema generated from the model in section Composing the XML document schema model:



Note the re-use references labelled cac: and cbc: into different XML library modules.

Schema Documentation

A documentation file may be generated for each document Schema, using an MS Excel format:



Sample XSD files

For a zipped sample delivery package of the above XML Schema files, please contact Data Management Solutions, www.d-m-s.co.nz. Alternatively, a trial copy of these models and tools is also available for download.

Continued on next page

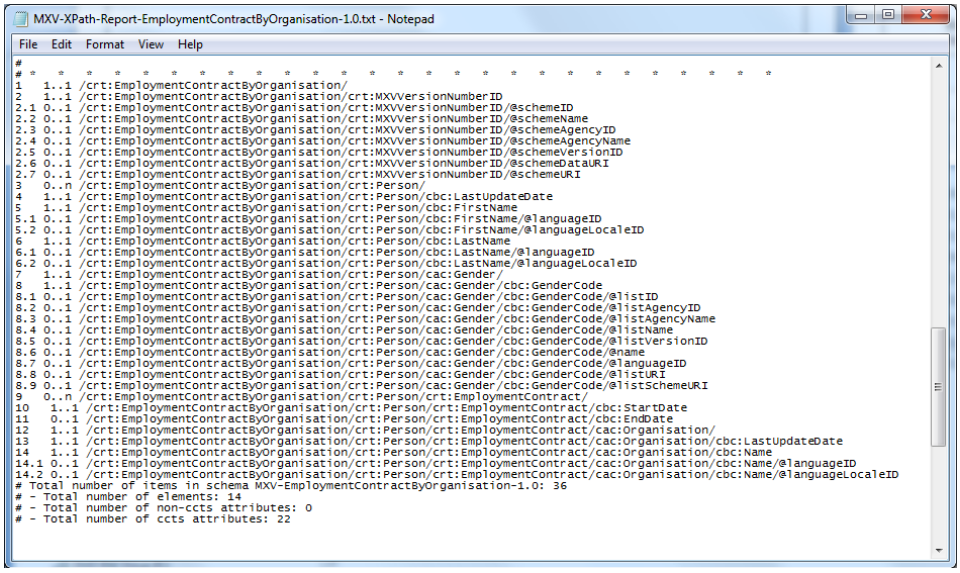
Sample XML Schemas and outputs, Continued

Sample XPath report

An XPath text file may be generated which enumerates all valid absolute XPath addresses for this document schema.

The generated file adheres to the same format used by OASIS for UBL XPath Reports.

The XPath Report file is a useful reference when coding XPath addresses for this document schema.

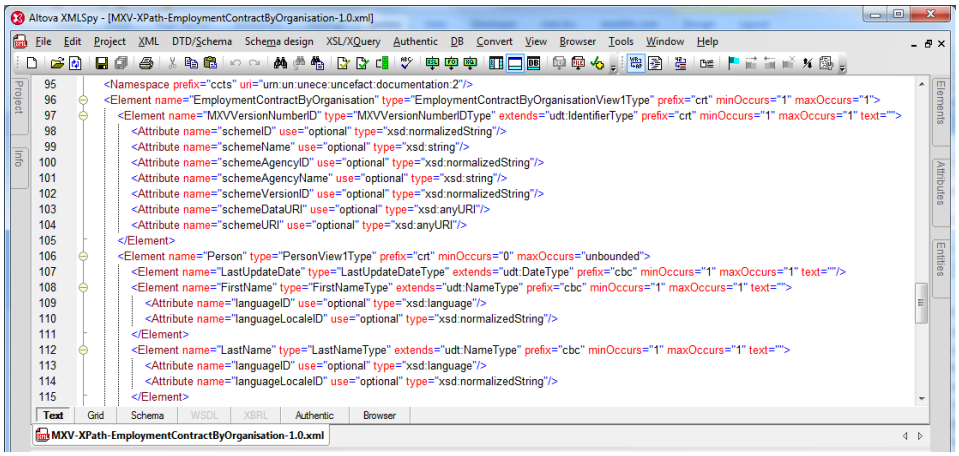


Sample XPath XML file

An XPath XML file may be generated which defines the information found in an XML instance of this XML document schema.

The generated XPath file complies with and validates against the OASIS XPath document model (XPath.xsd).

The XPath XML file is useful when machines are to process the XPath addresses of an instance document.



UML and XML model integration benefits

Background

The model-driven approach, combined with UBL/MXV NDR design, supported by the Sybase PowerDesigner tool suite, supplemented by Data Management Solutions' meta model extensions, model checks and automation tools provides an efficient integration of UML and XML models.

In this section

Benefits of the UML and XML integration:

Topic	See Page
Centralised management	39
Tool enforced validation and consistency	39
Cascading class and attribute definitions	39
Automation of repeating tasks	40
Impact analysis	40

Continued on next page

UML and XML model integration benefits, Continued

Centralised management

The storage of hand-coded XML Schemas is usually scattered across many LAN drives, creating a challenge to maintain them coherently. Even if they are all stored in one location, having to scan the XML Schema files to find element XML tag names is cumbersome.

The model-driven approach centralises all UML and XML models and their versions in one Sybase PowerDesigner model repository, enabling centralised maintenance and management not only of the models, but also of the generated XSD and other files.

Tool enforced validation and consistency

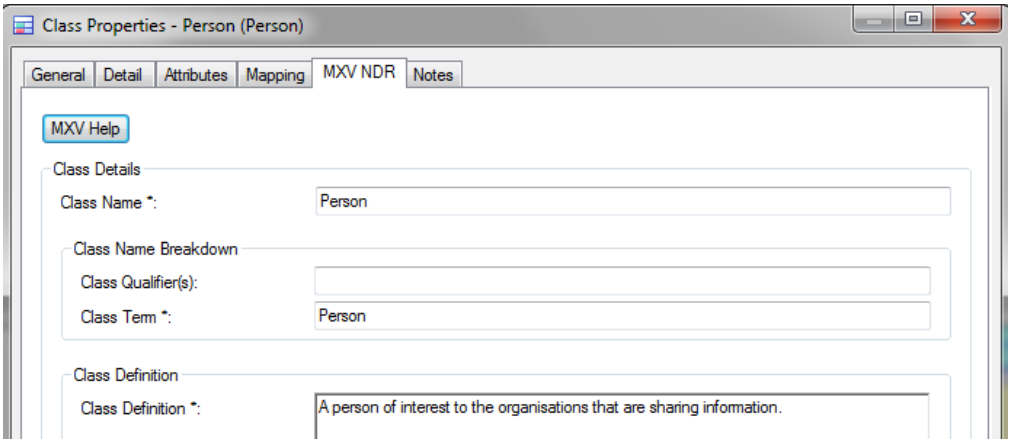
The many UBL/MXV NDR rules are difficult to enforce manually. The modelling tool in conjunction with the described customisations allows efficient end-to-end validations, delivering consistent XML Schema and XML 2-pass content validation deliverables.

All generated artefacts share the same look and feel, and where feasible re-use the same XML components. A single XML tag name always has the same definition and meaning regardless of what XML DOC schema it is used in.

Cascading class and attribute definitions

The definitions of classes and attributes in the UML model cascade down into the XML models, where they may be appended, but not changed, for more message specific contexts. The stem of the definition however matches.

Example: The class definition for 'Person' in the UML model …



… matches the element definition 'Person' in the generated DOC XSD file:

```
<xsd:element maxOccurs="unbounded" minOccurs="1" name="Person" type="cac:PersonView1Type">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      <ccts:Component>
        <ccts:ComponentType>ASBIE</ccts:ComponentType>
        <ccts:Definition>A person of interest to the organisations that are sharing information.</ccts:Definition>
        <ccts:Cardinality>1..unbounded</ccts:Cardinality>
        <ccts:ObjectClass>Example</ccts:ObjectClass>
        <ccts:AssociatedObjectClass>cac:PersonView1</ccts:AssociatedObjectClass>
      </ccts:Component>
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

Continued on next page

UML and XML model integration benefits, Continued

Automation of repeating tasks

The tight integration of UML and XML models permits the programmatic automation of relatively complex, error-prone and onerous tasks.

For example, the transitioning of UML objects into XML library module components allows automatically populating large parts of the XML models where possible.

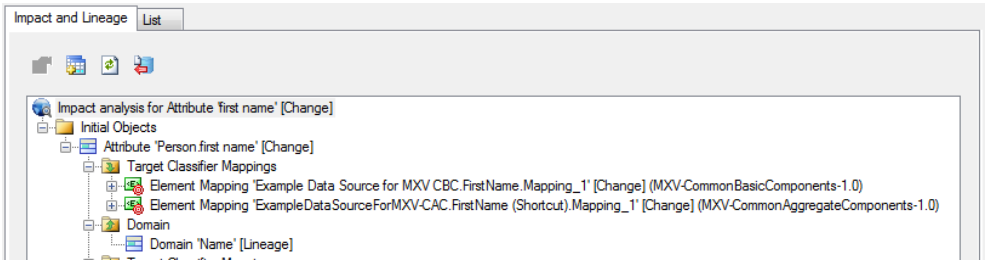
A seamless integration of XML models and XML 2-pass content validation related files automates the tedious and error-prone copying of content validation files when migrating them to specification status or creating delivery packages.

Such automation not only drastically reduces the administrator's turn-around time, but also the number of mistakes and change iterations.

Impact analysis

The end-to-end linking of UML and XML objects enables impact analysis and usage reports across model versions:

Example:
The attribute 'Person.first name' is implemented by XML element FirstName in a CBC XML model:



As soon as there are many versions of each model, efficient impact analysis is the linchpin for sustainable change management.

Unsustainable change management on the other hand is characterised by an inability to do impact analysis, increasing the risk of unexpected impacts, such as invalidating existing implementations, or cutting off an easy forward-migration path for existing implementations.

Appendix A: Glossary

Semantic model	A semantic data model defines the meaning of data unambiguously in form of a model. In this case study, the semantic model ESDM is a UML model containing domains, class diagrams, classes, attributes, associations and their definitions. The semantic model defines what data items and structures are permitted in compliant data exchanges, and what meaning the data items carry. Also referred to as canonical model, ontology model or vocabulary model
OASIS	Organisation for Advancement of Structured Information Standards. For details, see http://www.oasis-open.org/
OASIS Universal Business Language (UBL)	An OASIS Standard for electronic business. For details, see http://www.oasis-open.org/specs/index.php - ublv2.0[3]
OASIS Universal Business Language Naming and Design Rules (UBL NDR)	A section of the OASIS UBL standard pertaining to a normative set of XML schema design rules and naming conventions for the creation of business based XML schemas. For details, see http://www.oasis-open.org/specs/index.php - ublv2.0
Model-driven XML Vocabulary Naming and Design Rules (MXV NDR)	A variant version of UBL NDR customised for the model-driven design of prescriptive XML Schemas. MXV NDR is the industry-neutral name for the sum of UBL rules and their variations. MXV NDR has been adopted by the New Zealand Education Sector under the name of ESL NDR[4] to accommodate a non-UBL semantic model. The rules of MXV NDR and ESL NDR are identical.

Continued on next page

[3] As of July 2011, OASIS has not yet published the UBL NDR 2.1 documentation. At this stage MXV emulates UBL NDR 2.1

[4] ESL NDR: Education Sector Language Naming and Design Rules

Appendix A: Glossary, *Continued*

UN/CEFACT	The United Nations Centre for Trade facilitation and Electronic Business UN/CEFACT facilitate the development of e-business standards that can cross all international boundaries and help increase interoperability and lower transaction costs.
UML	Unified Modelling Language, a modelling notation suitable for object-oriented data modelling and other modelling purposes.
XML 2-pass content validation	A technique of validating the data content of an XML instance document after it successfully passed its structural/lexical schema validation. Code lists as well as complex business rules may be defined and validated platform independently. The 2-pass validation uses the following components: - OASIS Genericode - OASIS Context Value Association - An ISO Schematron/XSLT implementation available from: http://www.cranesoftwrights.com/resources/ubl/index.htm#cva2sch
Genericode	An OASIS standardised XML format for code lists. For details see http://www.genericode.org/ http://www.oasis-open.org/committees/codelist
Context Value Association	An OASIS standardised XML vocabulary using addresses to specify hierarchical document contexts and their associated constraints. For details see http://www.oasis-open.org/committees/codelist
ISO/IEC 11179 Part 5	The International Organisation for Standardization maintains this standard for naming data elements (besides many other standards). This standard forms an integral part of UBL and MXV NDR