1 Overview

This document is a Technical Design document for the Metadata Modeller (MM) component of the eSciDoc infrastructure.
In general the Metadata Modeler is responsible for managing metadata schemas, validating metadata records and transforming metadata records from one schema to another.

In the first release only metadata schema management and validation of the eSciDoc internal metadata schema is realized.

Metadata for Content Items will be stored and processed as XML Documents. The structural schema (description of the metadata elements) can be specified with XML Schemas (XSD).

This document describes:

- technical concept for modelling genre-specific metadata schemas with XML schemas.
- technical Design of the retrieveMetadataSchema Service
- technical Design of the validateMetadataRecord Service

This document references design artefacts from
“W:\docs\04_Infrastructure\06_Technical_Designs\Metadata Modeler\”

For more information on XML schemas see:
http://www.w3schools.com/schema/default.asp
http://de.wikipedia.org/wiki/XML_Schema

1.1 Modelling genre-specific metadata schemas

The eSciDoc metadata set is specified from a functional point of view in the document “Systemspecification_eSciDoc_metadata_sets.doc”
This document defines several abstract containers (e.g. Identifier, Person) which are used in subsets. These subsets are genre specific.
To realize this approach it is necessary to have more than one XML schema, i.e. each defined subset will be realized with a specific XML schema.

To avoid redundancies between genre-specific XML schemas, all types will be defined in one global master schema and only references in the genre-specific ones.
So we will end up with one large master schema and several smaller genre-specific schemas, which only include those types which are relevant to the genre.

For example the master schema defines an element title like

```
<xs:element name="Title" type="xs:string" nillable="false"/>
```

The genre specific one will reference this title

```
<xs:element ref="eSciDoc:Title"/>
```

1.1.1 eSciDoc master schema

The master schema consists of there parts:

- Element definition (All metadata elements which can be potentially included in the genre-specific schemas must be defined here)
- Complex type definitions (All complex types used by the element definition should be defined here. In most cases these complex types can be matched with the container specified in the Systemspecification_eSciDoc_metadata_sets)

- Simple type definitions (Simple type definitions can be used to realizes list of possible values.

An example for such a master schema could look like the following XML schema document:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 rel. 3 sp1 (http://www.altova.com) by markus tietz (fiz karlsruhe) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.escidoc-
project.de/metadata/schema/0.1/" targetNamespace="http://www.escidoc-project.de/metadata/schema/0.1/"
elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xs:element name="Genre" type="Genres" nillable="false"/>
	<xs:element name="Creator" type="CreatorType" nillable="false"/>
	<xs:element name="Date" type="DateType" nillable="false"/>
	<xs:element name="Title" type="xs:string" nillable="false"/>
	<xs:element name="Identifier" type="IdentifierType" nillable="false"/>
	<xs:element name="Relation" type="RelationType" nillable="false"/>
	<xs:element name="Subject" type="xs:string" nillable="false"/>
	<xs:element name="PublicationStatus" type="PublicationStati" nillable="false"/>
	<xs:element name="Pages" type="xs:integer" nillable="false"/>


	<xs:complexType name="IdentifierType">
		<xs:simpleContent>
			<xs:extension base="xs:string">
				<xs:attribute name="type" type="IdTypes" use="required"/>
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
	<xs:complexType name="PersonType">
		<xs:sequence>
			<xs:sequence minOccurs="0" maxOccurs="unbounded">
				<xs:element name="Id" type="IdentifierType" nillable="false"/>
			</xs:sequence>
			<xs:element name="CompleteName" type="xs:string" nillable="false"/>
			<xs:sequence minOccurs="0" maxOccurs="unbounded">
				<xs:element name="GivenName" type="xs:string" nillable="false"/>
			</xs:sequence>
			<xs:element name="FamilyName" type="xs:string" minOccurs="0"/>
			<xs:element name="Title" type="xs:string" minOccurs="0"/>
			<xs:sequence minOccurs="0" maxOccurs="unbounded">
				<xs:element name="Pseudonym" type="xs:string"/>
			</xs:sequence>
			<xs:element name="Initials" type="xs:string" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="CreatorType">
		<xs:sequence>
			<xs:element name="Person" type="PersonType" minOccurs="0"/>
			<xs:element name="Role" type="CreatorRoles" nillable="false"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="RelationType">
		<xs:sequence>
			<xs:element name="Name" type="RelationTypes" nillable="false"/>
			<xs:element name="RelationIdentifier" type="RelationIdentifierType" minOccurs="0"/>
			<xs:element name="ShortDescription" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="RelationIdentifierType">
		<xs:simpleContent>
			<xs:extension base="xs:string">
				<xs:attribute name="type" type="RelationIdentifiers" use="required"/>
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
	<xs:complexType name="DateType">
		<xs:simpleContent>
			<xs:extension base="xs:date">
				<xs:attribute name="type" type="DateTypes" use="required"/>
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>


	<xs:simpleType name="IdTypes">
		<xs:restriction base="xs:string">
			<xs:enumeration value="URI">
				<!--Uniform Resource Identifier-->
```

```
                        </xs:enumeration>
                        <xs:enumeration value="PND">
                                <!--Personennormdatei, provided by the Deutsche Bibliothek-->
                        </xs:enumeration>
                        <xs:enumeration value="GKD">
                                <!--Gesamtkoerperschaftsdatei, provided by the Deutsche Bibliothek-->
                        </xs:enumeration>
                        <xs:enumeration value="ISSN">
                                <!--International Standard Serials Number-->
                        </xs:enumeration>
                        <xs:enumeration value="ISBN">
                                <!--International Standard Book Number-->
                        </xs:enumeration>
                        <xs:enumeration value="DOI">
                                <!--Digital Object Identifier-->
                        </xs:enumeration>
                        <xs:enumeration value="ZDB">
                                <!--Zeitschriftendatenbank-->
                        </xs:enumeration>
                        <xs:enumeration value="local">
                                <!--Identifier used locally-->
                        </xs:enumeration>
                </xs:restriction>
                <!--The Type of the identifier, e.g. PND for the  “Personen Normdatei der Deutschen
Bibliothek” or an URI pointing to an eSciDoc item. -->
        </xs:simpleType>
        <xs:simpleType name="CreatorRoles">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Author"/>
                        <xs:enumeration value="Artist"/>
                        <xs:enumeration value="Editor"/>
                        <xs:enumeration value="Publisher"/>
                        <xs:enumeration value="Series Editor"/>
                        <xs:enumeration value="Painter"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="RelationTypes">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Is Original Of"/>
                        <xs:enumeration value="Is Translation Of"/>
                        <xs:enumeration value="Is Version of"/>
                        <xs:enumeration value="Is Referenced By"/>
                        <xs:enumeration value="References"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="RelationIdentifiers">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="eSciDoc"/>
                        <xs:enumeration value="generic"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="DateTypes">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="created"/>
                        <xs:enumeration value="valid"/>
                        <xs:enumeration value="available"/>
                        <xs:enumeration value="published"/>
                        <xs:enumeration value="modified"/>
                        <xs:enumeration value="copyrighted"/>
                        <xs:enumeration value="submitted"/>
                        <xs:enumeration value="accepted"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="Genres">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Article"/>
                        <xs:enumeration value="InBook"/>
                        <xs:enumeration value="Book"/>
                        <xs:enumeration value="Proceedings"/>
                        <xs:enumeration value="Conference-Paper"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="PublicationStati">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="published"/>
                        <xs:enumeration value="in press"/>
                        <xs:enumeration value="accepted"/>
                        <xs:enumeration value="submitted"/>
```

```
                        <xs:enumeration value="unpublished"/>
                </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

1.1.2 eSciDoc genre-specific schemas

The genre-specific XML schemas define the elements, whether they are optional, their cardinality.
It is possible to redefine or extend types defined in the master schema. The list of possible genres can be narrowed to the single allowed value by the following snippet:

```
        <xs:redefine schemaLocation="eSciDoc.xsd">
                <xs:simpleType name="Genres">
                        <xs:restriction base="eSciDoc:Genres">
                                <xs:enumeration value="Article"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:redefine>
```

Only the genre-specific schemes will be used during validation and only they contain the root-element eSciDoc. An example for a article specific XML schema could look like the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 rel. 3 sp1 (http://www.altova.com) by markus tietz (fiz karlsruhe) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:eSciDoc="http://www.escidoc-
project.de/metadata/schema/0.1/" targetNamespace="http://www.escidoc-project.de/metadata/schema/0.1/"
elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:redefine schemaLocation="eSciDoc.xsd">
                <xs:simpleType name="Genres">
                        <xs:restriction base="eSciDoc:Genres">
                                <xs:enumeration value="Article"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:redefine>
        <xs:element name="eSciDoc">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="eSciDoc:Genre"/>
                                <xs:sequence maxOccurs="unbounded">
                                        <xs:element ref="eSciDoc:Creator"/>
                                </xs:sequence>
                                <xs:sequence maxOccurs="unbounded">
                                        <xs:element ref="eSciDoc:Date"/>
                                </xs:sequence>
                                <xs:element ref="eSciDoc:Title"/>
                                <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                        <xs:element ref="eSciDoc:Identifier"/>
                                </xs:sequence>
                                <xs:element ref="eSciDoc:PublicationStatus" minOccurs="0"/>
                                <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                        <xs:element ref="eSciDoc:Relation"/>
                                </xs:sequence>
                                <xs:element ref="eSciDoc:Subject" minOccurs="0"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

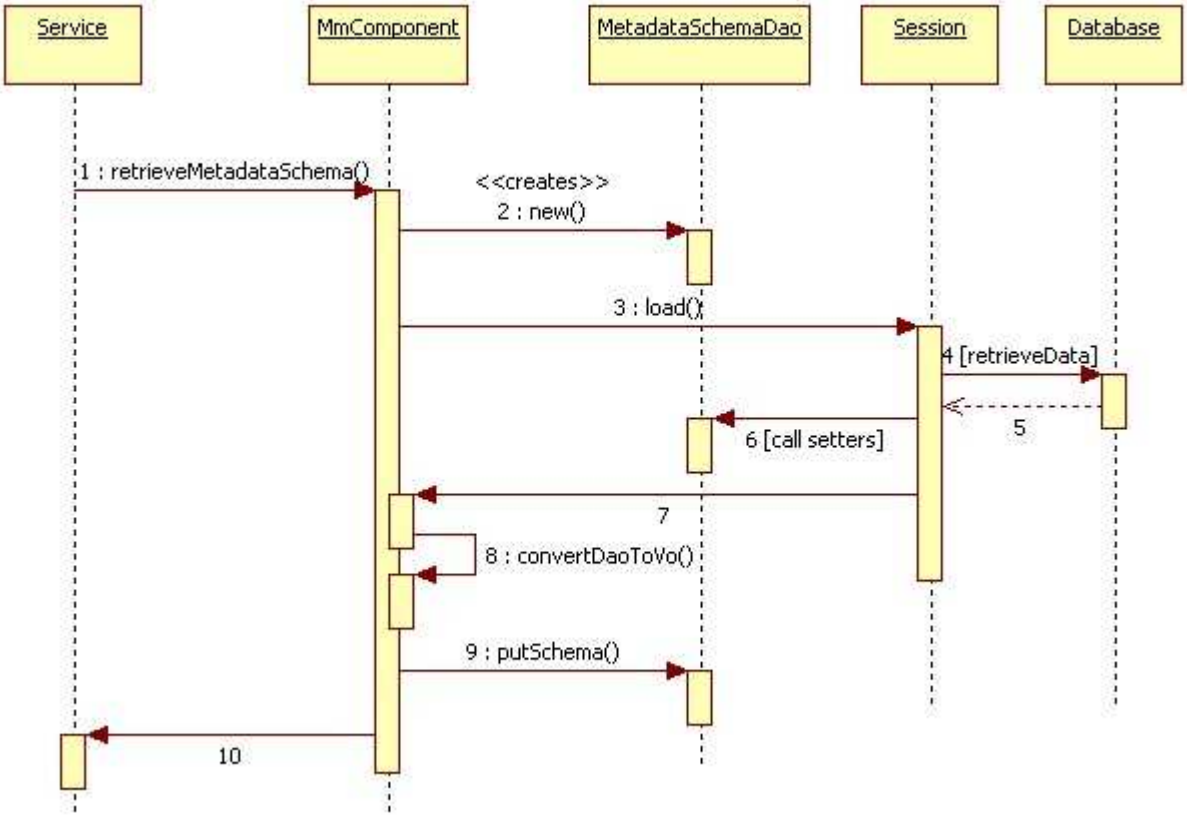
1.2 Metadata schema management

Although we will have in the first release only a fixed set of metadata schemas they will nevertheless be stored in a database to be prepared for the following releases in which (non-eSciDoc) metadata schemas can be updated on the fly.
The schemas which are much more often read than changed will be cached in the following releases.

The service

```
SchemaVo retrieveMetadataSchema(final String id, final String genre);
```

gets an identifier and a genre as input parameter. With the id and the genre it checks the database. If the schema is in the database it can be retrieve by the means of Hibernate.
The service is summarized in the following sequence diagram:



1.3 Metadata validation

Since metadata records are XML documents and metadata schemas are XML schemas standard techniques for validation of XML documents can be utilized. There are several open source implementations for XML parsing and validation available, e.g. Xerces2-J which is the most common one.

In general there are two different APIs to process XML documents: SAX and DOM.
SAX is a lexical, event-driven interface in which a document is read serially and its contents are reported as "callbacks" to various methods on a handler object of the user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed. It is better suited to situations in which certain types of information are always handled the same way, no matter where they occur in the document.

DOM is an interface-oriented API that allows for navigation of the entire document as if it were a tree of "Node" objects representing the document's contents. A DOM document can be created by a parser, or can be generated manually by users (with limitations). Data types in DOM Nodes are abstract; implementations provide their own programming language-specific bindings. DOM implementations tend to be memory intensive, as they generally require the entire document to be loaded into memory and constructed as a tree of objects before access is allowed.

For the service

```
SchemaValidationVo validateMetadataRecord(final String metadataRecord, final String schemaId,
final String genre);
```

It is only necessary to parse the document and not to navigate through it. Therefore a SAX Parser is more appropriate for this service, since SAX Parsers offer a better performance than DOM Parsers.

A specific SAX parser implementation can be wired into the MM component by spring, since all implementations stick to the general standard interface defined in the abstract class `javax.xml.parsers.SAXParser`

The following sequence diagram describes the service internally: