INTERLIS Repository

Stefan Henrich, Rafael Schertenleib

Version 1.1.1, 2023-01-31

Inhalt

Fehler entdeckt? Diese Website wird via Github gepflegt.

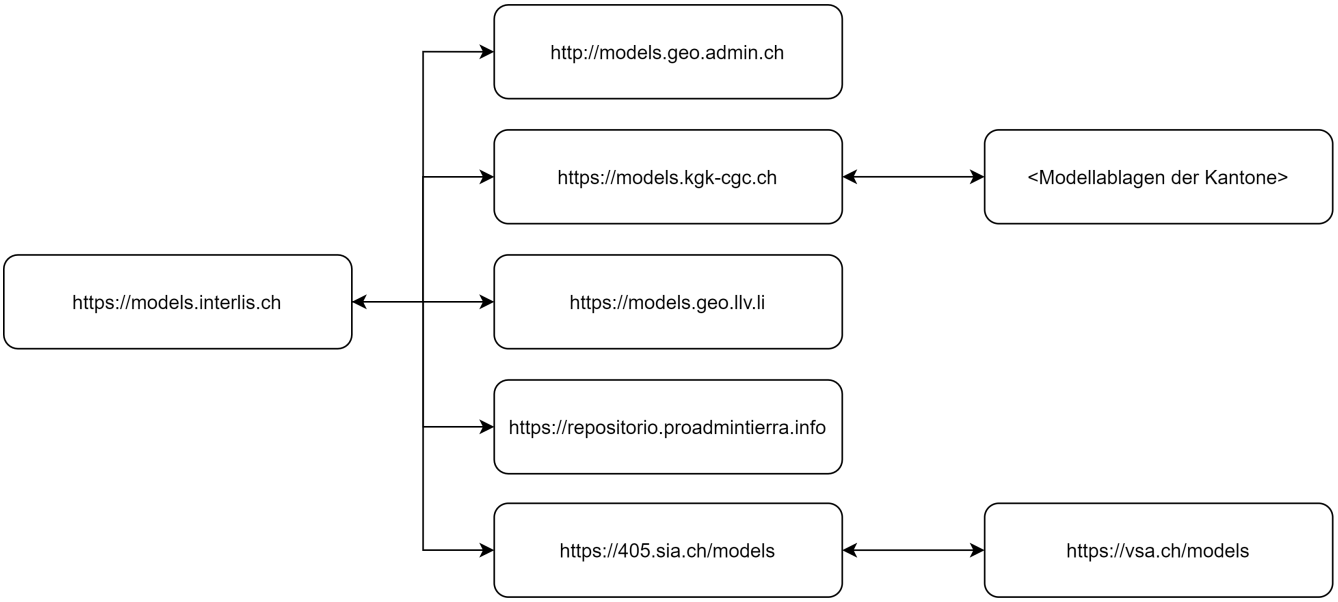
1. Einführung

1.1. Idee

Das Konzept des INTERLIS Repository sieht vor, INTERLIS Datenmodelle (.ili) und Transferdateien (.itf resp. .xtf) nutzbar zu machen, in dem diese über einen Index (Inhaltsverzeichnis) auffindbar sind. Dabei können die Modelldateien (INTERLIS1 und INTERLIS2) und die Transferdateien zusammen mit dem Inhaltsverzeichnis auf einem Webserver abgelegt sein und somit online bereitgestellt werden. Software-Werkzeuge, die diese Index auswerten, können anschliessend nach Bedarf automatisch INTERLIS-Modelle und Transferdateien herunterladen (Maschine-zu-Maschine, M2M). Der Vorteil für die Benutzer:innen einer entsprechenden Software liegt darin, dass sie sich nicht um die Modelldateien und deren Abhängigkeiten untereinander kümmern müssen. Die für einen Transferdatensatz (.itf resp. .xtf) benötigten Modelle werden automatisch heruntergeladen und in einem Zwischenspeicher (Cache) vorgehalten.

HINWEIS	Folgende Produkte unterstützen INTERLIS Repositories (Stand 2022): INTERLIS Compiler (ili2c), INTERLIS/UML Editor (umleditor), INTERLIS Validator (ilivalidator), INTERLIS Manager (ilimanager), iG/Check, infogrips INTERLIS Tools / INTERLIS Tools Pro, ili2db, FME Plug-In für INTERLIS und alle weiteren Werkzeuge, die auf obiger Software aufbauen.
HINWEIS	Beim Zwischenspeicher für das lokale Vorhalten der heruntergeladenen Modelldateien handelt es sich um ein Verzeichnis mit dem Namen `.ilicache`. Es befindet sich im jeweiligen Benutzerverzeichnis (z. B. unter Windows in `C:\Users\<Benutzername>\.ilicache`).

Die INTERLIS Repositories können untereinander vernetzt werden. So hat sich in der Schweiz eine föderalistisch geprägte, hierarchische Struktur von Modellablagen ergeben . An der Spitze steht das Repository unter http://models.interlis.ch. Die zwei wichtigsten Sub-Datenmodell-Ablagen sind dasjenige des Bundes (http://models.geo.admin.ch), welches alle verfügbaren INTERLIS-Modelle der Geobasisdatensätze des Bundesrechts enthält, und jenes von der KGK (Konferenz der kantonalen Geoinformations- und Katasterstellen) unter http://models.kgk-cgc.ch mit weiteren Subrepositories der einzelnen Kantone.



Übersicht der öffentlichen Datenmodell-Ablagen (Stand 2022)

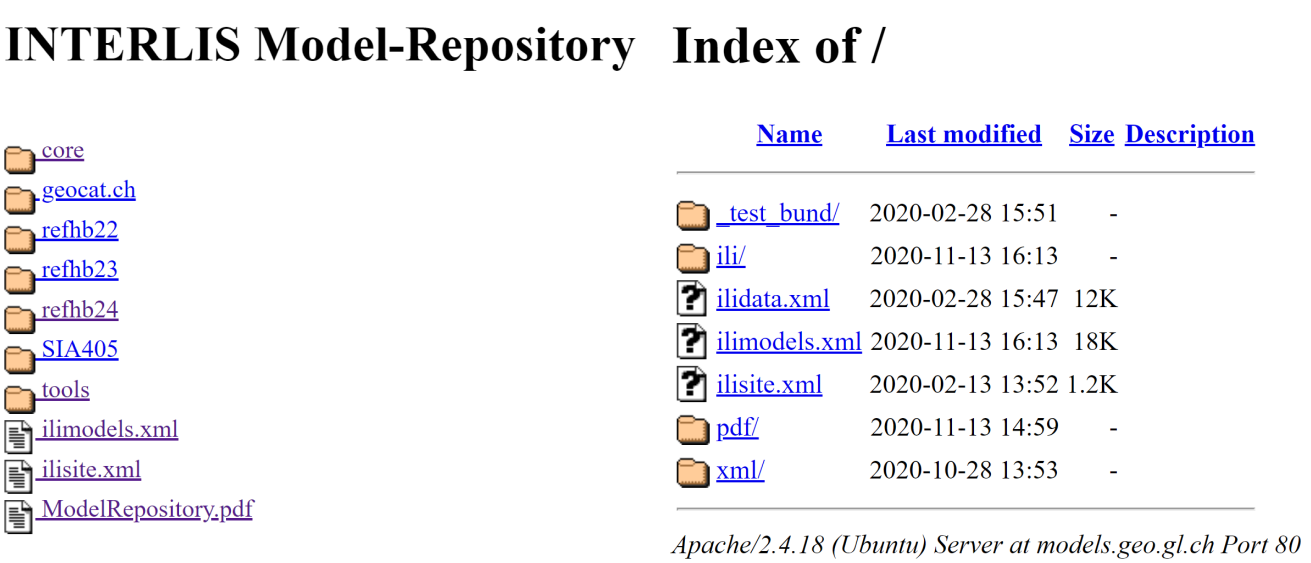
Im Zusammenhang mit INTERLIS Repositories sind folgende Konfigurationsdateien von Bedeutung:

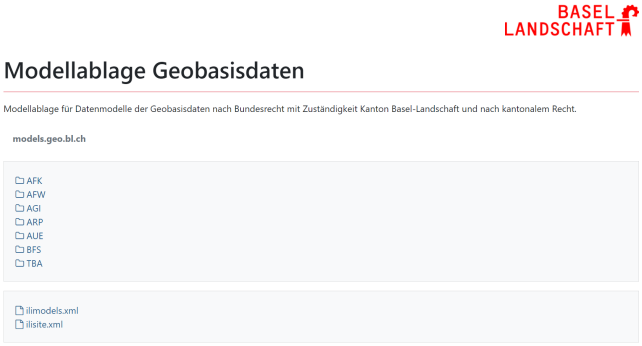
Datei	Inhalt
`ilimodels.xml`	Liste/Index aller im aktuellen Repository verfügbaren Datenmodelle mit den zugehörigen Metadaten.
`ilidata.xml`	Liste/Index aller im aktuellen Repository verfügbaren Transferdateien mit den zugehörigen Metadaten.
`ilisite.xml`	Beschreibung des aktuellen Repository; http(s)-Links auf übergeordnete und untergeordnete Repositories.

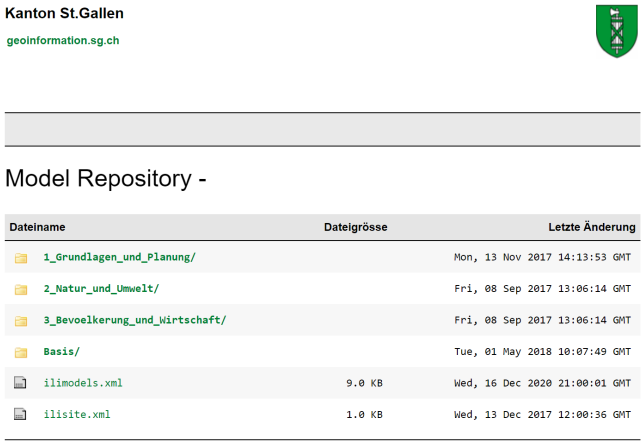
Jeder der oben genannten Konfigurationsdateien liegt - wie könnte es anders sein - ein eigenes INTERLIS Datenmodell zugrunde. Damit sind die Konfigurationsdateien nichts anderes als INTERLIS Transferdateien mit der Endung .xml (statt .xtf). Somit lassen sich die Konfigurationsdateien auch mittels Werkzeugen wie *ilivalidator* auf ihre Korrektheit prüfen.

1.2. Technische Bereitstellung

Hinter einem INTERLIS Repository steht keine Software, die installiert und betrieben werden muss. Vielmehr müssen die Konfigurationsdateien und die bereit zu stellenden INTERLIS-Modelle und Transferdateien über eine stabile URL erreichbar sein. Dies kann z. B. durch das Hochladen der Daten auf einer eigenen Website erfolgen oder aber auch auf einem öffentlichen Datenspeicher (Github Repository). Aus diesem Grund unterscheiden sich die bestehenden Repositories auch in ihrer Darstellung.







2. ilimodels.xml — Das Modell für Datenmodelle

2.1. Bedeutung

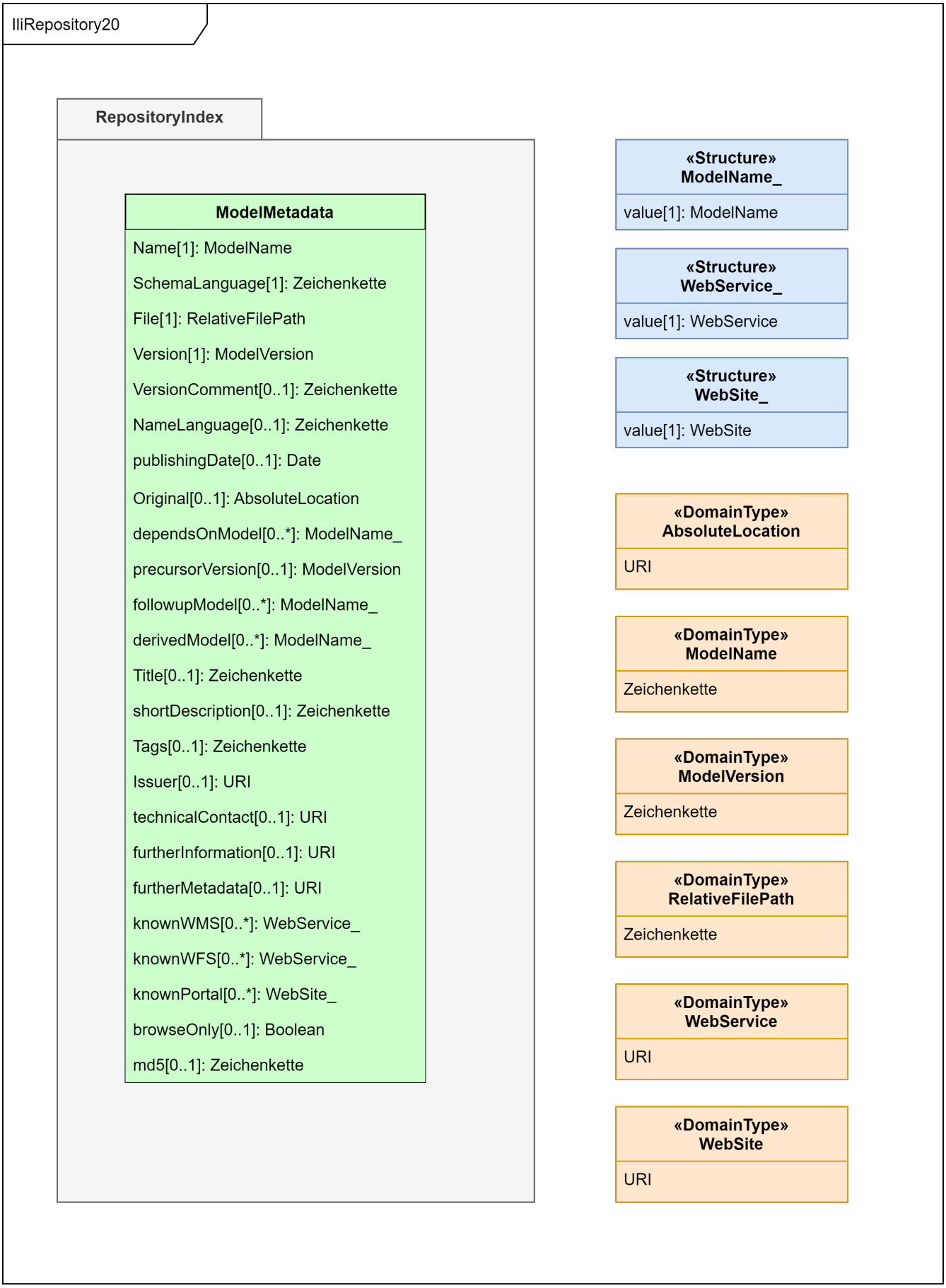
In der Datei `ilimodels.xml` werden alle INTERLIS-Modelle aufgeführt, die im Repository verfügbar sind. Die Datei liegt im Wurzelverzeichnis eines Repository. Neben dem Modellnamen und dem Pfad auf die effektive Modelldatei (.ili) werden weitere Metadaten geführt (z.B. Version, Publikationsdatum, Kontaktdaten).

HINWEIS

Ein Modell, das nicht in `ilimodels.xml` mit den zugehörigen Metadaten aufgeführt ist, existiert aus Sicht eines Tools, das sich dem Repository bedient, nicht. Es reicht also nicht, ein neues Modell nur im Repository-Verzeichnis abzulegen, sondern es muss auch in `ilimodels.xml` mit den korrekten Metadaten aufgelistet sein, damit es automatisch gefunden wird.

2.2. Aufbau

Den Daten in `ilimodels.xml` liegt das nachfolgende Datenmodell zugrunde:[1]



UML-Diagramm des Datenmodells IliRepository20

Klasse ModelMetadata

Attributname	Bedeutung
Name[1]	Datenmodelname, wie angegeben in der .ili-Datei nach `MODEL` (z. B. "KGKCGC_FPDS2_V1_1").
SchemaLanguage[1]	Verwendete Schemasprache des Datenmodelles (gültige Werte: *ili1, ili2_2, ili2_3, ili2_4*).
File[1]	Relativer Pfad der .ili-Datei, worin das angegebene Datenmodell enthalten ist (in Bezug auf das Wurzelverzeichnis des Repositorys, wo auch die Datei ilimodels.xml liegt).
Version[1]	Version des Datenmodelles wie angegeben im Modell nach `VERSION` (z. B. "2020-04-17").
VersionComment[0..1]	Kommentar zu dieser Datenmodellversion; entspricht [Explanation] zur Version, welche im Datenmodell mit `//` eingeleitet und abgeschlossen wird (siehe auch Syntaxregel *ModelDef* im RefHB).
NameLanguage[0..1]	Natürliche Sprache, in der das Modell verfasst ist (z. B. "de"), wie angegeben im Datenmodell nach dem Datenmodellnamen.
publishingDate[0..1]	Entspricht dem Änderungsdatum der .ili-Datei.
Original[0..1]	URL der Original-Publikation des Datenmodelles durch die zuständige Stelle. Kann die URL eines Repository oder direkt der .ili-Datei sein.
dependsOnModel[0..*]	Auflistung aller im Datenmodell mittels `IMPORTS` und `TRANSLATION OF` referenzierten Datenmodelle, so dass ein Tool alle erforderlichen Dateien erst herunterladen kann, bevor es sie lesen / kompilieren muss.
precursorVersion[0..1]	Version des Vorgängers des Datenmodelles (z. B. "2020-04-01"); muss im gleichen Repository vorliegen.
followupModel[0..*]	Name des Nachfolgemodelles.
derivedModel[0..*]	Namen von Datenmodellen, welche keinen streng technischen (`IMPORTS`, `TRANSLATION OF`), wohl aber einen losen Bezug zum Datenmodell haben (z. B. ein Publikationsmodell).
Title[0..1]	Bezeichnung des Datenmodelles in natürlicher Sprache.
shortDescription[0..1]	Kurze Beschreibung oder Zusammenfassung des Datenmodelles in natürlicher Sprache.
Tags[0..1]	kommagetrennte Liste mit Schlagworten zum Datenmodell.
Issuer[0..1]	URL der Herausgeberin des Datenmodelles (z. B. `https://www.swisstopo.ch`).
technicalContact[0..1]	E-Mail Link für technische Fragen zum Datenmodell (z .B. `mailto:info@kgk-cgc.ch`).
furtherInformation[0..1]	URL zu weiterführender Dokumention für das Datenmodell.
furtherMetadata[0..1]	URL zu einem maschinenlesbaren Metadatensatz zum Datenmodell.

knownWMS[0..*]	URL zu WMS-Diensten, welche Daten in diesem Datenmodell bereitstellen.[2]
knownWFS[0..*]	URL zu WFS-Diensten, welche Daten in diesem Datenmodell bereitstellen.[2]
knownPortal[0..*]	URL zu Portalen, über die ein Nutzer Daten nach diesem Datenmodell betrachen oder herunterladen kann.
browseOnly[0..1]	`true` : Das Datenmodell (.ili-Datei) wird ignoriert. `false` (default): Das Datenmodell kann gefunden werden.
md5[0..1]	MD5-Prüfsumme der unter *File* angegebenen .ili-Datei.

HINWEIS

In einer .ili-Datei können grundsätzlich mehrere Datenmodelle (`MODEL xyz = ⋯`) aufgeführt sein. Dies ist z. B. häufig bei den Bundesmodellen der Fall, wo in derselben Datei je ein Modell für den Bezugsrahmen LV03 und eines für LV95 enthalten sind (Beispiel). Um beide Datenmodelle über die Repository-Funktionalität zugänglich zu machen, wird für jedes der Modelle ein Eintrag in `ilimodels.xml` benötigt. Beide Einträge verweisen jedoch auf dieselbe .ili-Datei.

2.3. Beispiel eines Eintrags in ilimodels.xml

```
<IliRepository20.RepositoryIndex.ModelMetadata TID="100503">
    <Name>GeometryCHLV95_V2</Name>
    <SchemaLanguage>ili2_4</SchemaLanguage>
    <File>CH/CHBase_Part1_GEOMETRY_V2.ili</File>
    <Version>2021-01-06</Version>
    <publishingDate>2021-01-06</publishingDate>
    <dependsOnModel>
        <IliRepository20.ModelName_>
            <value>Units</value>
        </IliRepository20.ModelName_>
        <IliRepository20.ModelName_>
            <value>CoordSys</value>
        </IliRepository20.ModelName_>
        <IliRepository20.ModelName_>
            <value>Geometry_V2</value>
        </IliRepository20.ModelName_>
    </dependsOnModel>
    <Issuer>https://models.geo.admin.ch/CH/</Issuer>
    <technicalContact>mailto:models@geo.admin.ch</technicalContact>
    <furtherInformation>https://www.geo.admin.ch/de/geoinformation-
schweiz/geobasisdaten/geodata-models.html</furtherInformation>
    <md5>62eb9dcfbf09d5c183cbbaca52e8b821</md5>
</IliRepository20.RepositoryIndex.ModelMetadata>
```

3. ilisite.xml — Die Vernetzung beginnt

3.1. Bedeutung

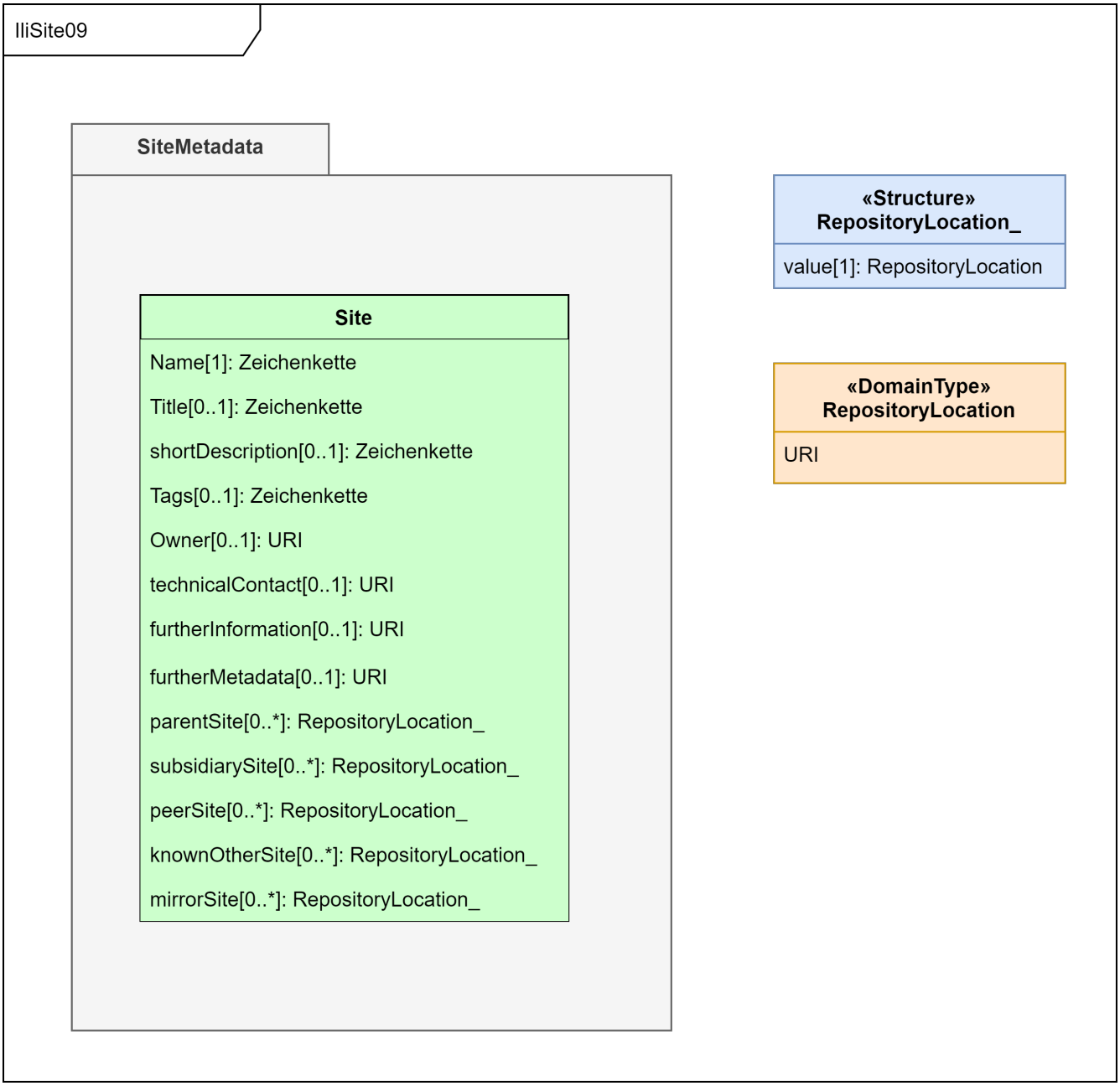
Mittels der Datei `ilisite.xml` wird ein Repository identifiziert. Die Datei liegt im Wurzelverzeichnis eines Repositorys und beinhaltet dessen Metadaten (z.B. Name, Betreiberin, Kontaktdaten). Zudem können darin auch URL zu übergeordneten und untergeordneten Repositories angegeben werden. Werkzeuge, die nach Modellen oder Transferdateien suchen,[3] können sich dadurch von Repository zu Repository hangeln, um ihre Suche auszudehnen. Die Suche läuft nach dem Breadth-First-Verfahren (Wikipedia) ab.

[300] | *docs/img/Breadth-First-Search-Algorithm_finalFrame.gif*

Funktionsweise Breadth-First-Search-Algorithmus

3.2. Aufbau

Den Daten in `ilisite.xml` liegt das nachfolgende Datenmodell zugrunde:



UML-Diagramm des Datenmodells IliSite09

Das Modell besteht nur aus einer einzigen Klasse, *Site*, welche 13 Attribute enthält.

Klasse Site	
Attributname	**Bedeutung**
Name[1]	Name der Seite (z. B. qualifizierter Hostname wie `models.kgk-cgc.ch`).
Title[0..1]	Titel der Seite (z. B. "Model-Repository der Kantone und der KGK-CGC").
shortDescription[0..1]	Kurze Beschreibung zum Zweck des Repositorys.
Tags[0..1]	kommagetrennte Liste mit Stichworten, die mit diesem Repository in Verbindung gebracht werden sollen.
Owner[0..1]	URL der zuständigen Stelle (z. B. `https://www.kgk-cgc.ch`).

technicalContact[0..1]	E-Mail Link für technische Fragen zum Repository (z .B. `mailto:info@kgk-cgc.ch`).
furtherInformation[0..1]	URL zu weiterführender Dokumention für dieses Repository.
furtherMetadata[0..1]	URL zu einem maschinenlesbaren Metadatensatz zu diesem Repository.
parentSite[0..*]	URL zu weiteren übergeordneten Repositories.
subsidiarySite[0..*]	URL zu weiteren untergeordneten Repositories.
peerSite[0..*]	URL zu weiteren Repositories (auf gleicher Organisationsebene).
knownOtherSite[0..*]	URL zu anderen bekannten Repositories, wird bei Standardmodellsuche ignoriert.
mirrorSite[0..*]	URL zu gespiegelten Repositories, falls diese Seite offline ist.

HINWEIS

Wie in der Tabelle erkennbar, sind die Attribute *parentSite* und *subsidiarySite* nicht obligatorisch: Gibt man weder übergeordnete noch untergeordnete Repositories an, so funktioniert das Repository komplett autark. Das ist z. B. für Testzwecke, Entwicklungsumgebungen oder vertrauliche Datenmodelle geeignet, die nicht von ausserhalb einer Organisation gefunden werden sollen. Um ein solches autarkes Repository verwenden zu können, muss man dieses in den Software-Werkzeugen explizit als Modellquelle angeben.

3.3. Beispiel für eine Datei ilisite.xml

```
<IliSite09.SiteMetadata BID="b0">
    <IliSite09.SiteMetadata.Site TID="1">
        <Name>models.interlis.ch</Name>
        <Title>Allgemeine technische INTERLIS-Modelle</Title>
        <Owner>http://www.interlis.ch</Owner>
        <technicalContact>mailto:info@interlis.ch</technicalContact>
        <subsidiarySite>
            <IliSite09.RepositoryLocation_>
                <value>http://models.geo.admin.ch</value>
            </IliSite09.RepositoryLocation_>
            <IliSite09.RepositoryLocation_>
                <value>http://models.kkgeo.ch</value>
            </IliSite09.RepositoryLocation_>
            <IliSite09.RepositoryLocation_>
                <value>http://models.geo.llv.li</value>
            </IliSite09.RepositoryLocation_>
            <IliSite09.RepositoryLocation_>
                <value>https://repositorio.proadmintierra.info</value>
            </IliSite09.RepositoryLocation_>
        </subsidiarySite>
    </IliSite09.SiteMetadata.Site>
```

```
</IliSite09.SiteMetadata>
```

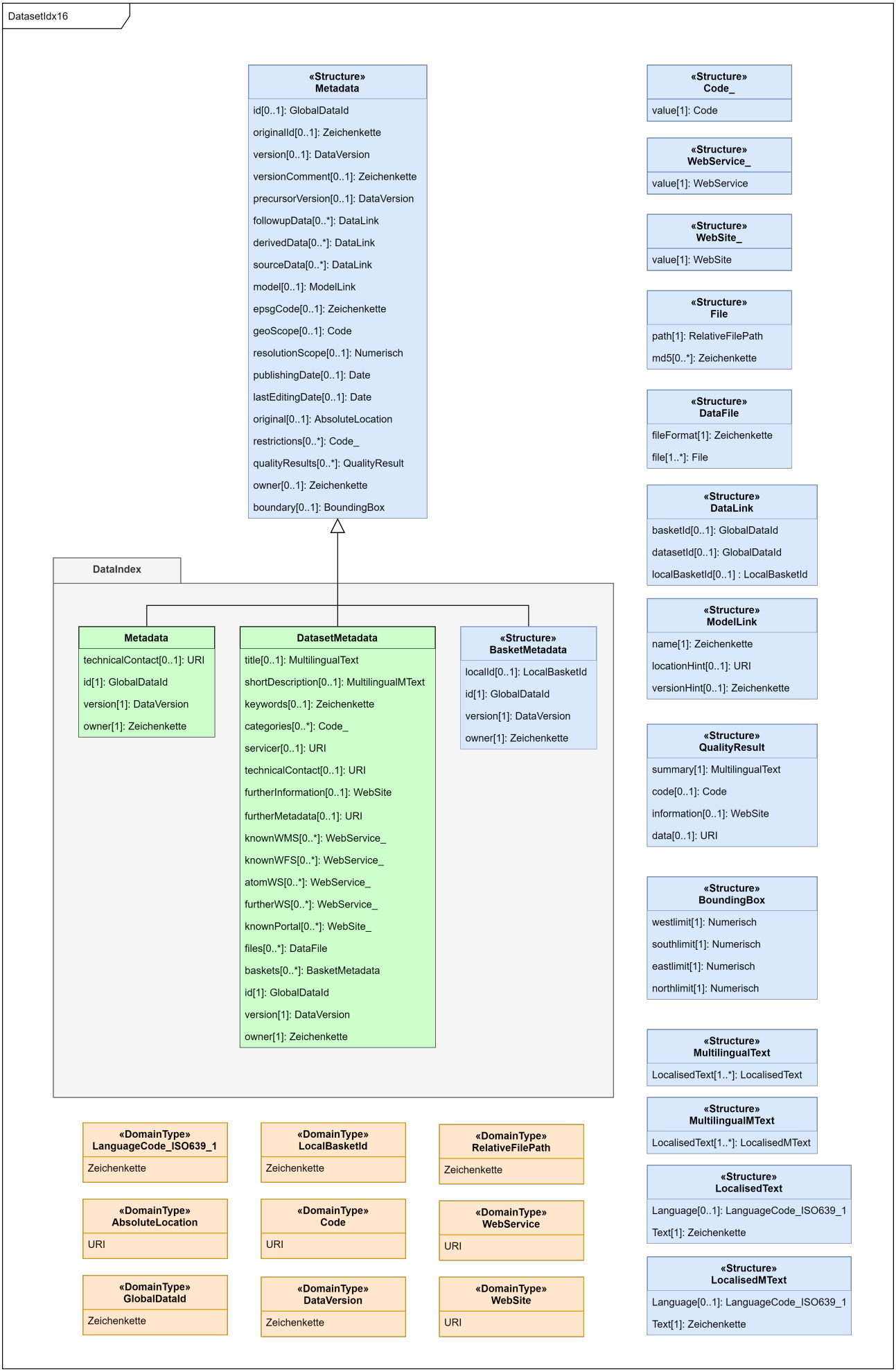
4. ilidata.xml — Das Modell für Transferdateien

4.1. Bedeutung

In der Datei `ilidata.xml` werden alle Transferdateien aufgeführt, die in einem Daten-Repository verfügbar sind. Sie liegt im Wurzelverzeichnis eines Repositorys. Es können eine Vielzahl von Metadaten erfasst werden.

4.2. Aufbau

Den Daten in `ilidata.xml` liegt das nachfolgende Datenmodell zugrunde:



UML-Diagramm des Datenmodells DatasetIdx16

Klasse DatasetMetadata	
Attributname	**Bedeutung**
(erbt alle Attribute aus der Struktur «Metadata», siehe unten)	
title[0..1]	Titel des Datensatzes (kann mehrsprachig sein).
shortDescription[0..1]	Kurze Beschreibung oder Zusammenfassung des Datensatzes (kann mehrsprachig sein).
keywords[0..1]	Kommaseparierte Liste mit Schlüsselbegriffen zum Datensatz.
categories[0..*]	*(Verwendungszweck noch unklar)*
servicer[0..1]	URL zum Bereitsteller des Datensatzes.
technicalContact[0..1]	E-Mail Link für technische Fragen zum Datensatz (z .B. `mailto:info@kgk-cgc.ch`).
furtherInformation[0..1]	URL zu einem weiterführenden Dokument oder einer Website mit zusätzlichen Informationen.
furtherMetadata[0..1]	URL zu einem maschinenlesbaren Daten zu diesem Datensatz.
knownWMS[0..*]	URL zu WMS-Diensten, welche den Datensatz bereitstellen.
knownWFS[0..*]	URL zu WFS-Diensten, welche den Datensatz bereitstellen.
atomWS[0..*]	URL zu ATOM-Diensten, welche den Datensatz bereitstellen.
furtherWS[0..*]	URL zu jeder anderen Art von Web-Diensten, welche diesen Datensatz bereitstellen oder prozessieren.
knownPortal[0..*]	URL zu Portalen, über die ein Nutzer diesen Datensatz betrachen oder herunterladen kann.
files[0..*]	Relative Dateipfade (pro vorhandener Datei) zu diesem Datensatz, inklusive Dateiformat und MD5 Prüfsumme.
id[1]	*geerbtes Attribut, hier obligatorisch (via CONSTRAINT).*
version[1]	*geerbtes Attribut, hier obligatorisch (via CONSTRAINT).*
owner[1]	*geerbtes Attribut, hier obligatorisch (via CONSTRAINT).*

Klasse Metadata	
Attributname	**Bedeutung**
(erbt alle Attribute aus der Struktur «Metadata», siehe unten)	
technicalContact[0..1]	E-Mail Link für technische Belange
id[1]	*geerbtes Attribut, hier obligatorisch (via CONSTRAINT).*
version[1]	*geerbtes Attribut, hier obligatorisch (via CONSTRAINT).*

owner[1]	*geerbtes Attribut, hier obligatorisch (via CONSTRAINT).*

Die wichtigsten Strukturen

Struktur «Metadata»	
Attributname	**Bedeutung**
id[0..1]	Stabiler und eindeutiger Datensatz-Identifikator (z. B. "ch.bfe.kernkraftwerke.1"). Wird in der Dokumentation von Software-Werkzeugen auch als `datasetId` bezeichnet.
originalId[0..1]	Datensatz-Identifikator, so, wie ihn der Eigentümer der Daten verwendet.
version[0..1]	Version des Datensatzes.
versionComment[0..1]	Kommentar zu dieser Version des Datensatzes (z. B. "Entwurf").
precursorVersion[0..1]	Version des Vorgängers des Datensatzes – muss im gleichen Repository vorliegen.
followupData[0..*]	Angaben zum Nachfolger des Datensatzes, die auch in einem anderen Repository vorliegen können.
derivedData[0..*]	Angaben zu abgeleiteten Daten des Datensatzes, die auch in einem anderen Repository vorliegen können.
sourceData[0..*]	Falls es sich beim Datensatz um abgeleitete Daten handelt, so können hier die Quelldatensätze angegeben werden, die auch in einem anderen Repository vorliegen können.
model[0..1]	INTERLIS-Modell oder Schema, welches zum Datensatz gehört. Qualifizierter TOPIC-Name, wenn es sich um INTERLIS Baskets handelt (Name, Repository, Versionshinweis).
epsgCode[0..1]	Bezugssystem dieses Datensatzes als EPSG Code; undefiniert, falls es sich um Daten ohne Geometrie handelt (z. B. Kataloge) oder um Daten, welche in mehreren Koordinatenreferenzsystemen (CRS) vorliegen.
geoScope[0..1]	*(Verwendungszweck noch unklar)*
resolutionScope[0..1]	Referenzmassstab des Datensatzes (Massstabszahl, also z. B. "50000" bei einem Massstab von 1:50'000).
publishingDate[0..1]	Publikationsdatum dieser Version des Datensatzes.
lastEditingDate[0..1]	Nachführungsdatum dieser Version des Datensatzes.
original[0..1]	URL der Original-Publikation dieses Datensatzes durch die zuständige Stelle (Link auf Daten-Repository oder Datei).
restrictions[0..*]	URL auf Nutzungsbestimmungen.

qualityResults[0..*]	Zusammenfassung und Verlinkung auf Dokumente (Logs) zur Validierung des Datensatzes (kann mehrsprachig sein).
owner[0..1]	URL der zuständigen Stelle.
boundary[0..1]	BBOX des Datensatzes.

Struktur «BasketMetadata»	
Attributname	**Bedeutung**
(erbt alle Attribute aus der Struktur «Metadata», siehe oben)	
localId[0..1]	Lokaler Identifikator für diesen Datenbehälter; ist obligatorisch, falls *id* fehlt (via CONSTRAINT)
id[1]	*geerbtes Attribut; hier obligatorisch, falls* localId *fehlt (via CONSTRAINT).*
version[1]	*geerbtes Attribut, hier obligatorisch (via CONSTRAINT).*
owner[1]	*geerbtes Attribut, hier obligatorisch (via CONSTRAINT).*

Struktur «DataFile»	
Attributname	**Bedeutung**
fileFormat[1]	MIME type des Datensatzes; für INTERLIS 2.3 lautet er `application/interlis+xml;version=2.3`
file[1..*1]	Relativer Pfad sowie MD5 Prüfsumme des Datensatzes .

4.3. Beispiel für zwei Datensätze in ilidata.xml

```
<DatasetIdx16.DataIndex.DatasetMetadata TID="7ea67d69-49a6-49e8-b024-21a7780cbed5">
    <id>ch.gl.transportation.wanderwege.hpm_type.1</id>
    <title>
        <DatasetIdx16.MultilingualText>
            <LocalisedText>
                <DatasetIdx16.LocalisedText>
                    <Language>de</Language>
                    <Text>HPM-Type</Text>
                </DatasetIdx16.LocalisedText>
            </LocalisedText>
        </DatasetIdx16.MultilingualText>
    </title>
    <shortDescription>
        <DatasetIdx16.MultilingualMText>
            <LocalisedText>
                <DatasetIdx16.LocalisedMText>
                    <Language>de</Language>
```

```
                    <Text>
                        HPM-Type: kantonale Ergänzung um Typ "GL_Landesfussweg" und
"GL_Landesfussweg_route"
                    </Text>
                </DatasetIdx16.LocalisedMText>
            </LocalisedText>
        </DatasetIdx16.MultilingualMText>
    </shortDescription>
    <version>2018-09-14</version>
    <owner>mailto:geoinformation@gl.ch</owner>
    <files>
        <DatasetIdx16.DataFile>
            <fileFormat>application/interlis+xml;version=2.3</fileFormat>
            <file>
                <DatasetIdx16.File>
                    <path>xml/GL_Hpm_Catalogues_V1_1.xml</path>
                    <md5>eef3eaa5894dddf635cf5e214d031f7b</md5>
                </DatasetIdx16.File>
            </file>
        </DatasetIdx16.DataFile>
    </files>
    <baskets>
        <DatasetIdx16.DataIndex.BasketMetadata>
            <id>gl_hpm_type</id>
            <!-- BID of basket in XTF -->
            <version>2018-09-14</version>
            <model>
                <DatasetIdx16.ModelLink>
                    <name>hpm_network_V1.hpm_catalogues</name>
                    <!-- qualified TOPIC name -->
                </DatasetIdx16.ModelLink>
            </model>
            <owner>mailto:geoinformation@gl.ch</owner>
        </DatasetIdx16.DataIndex.BasketMetadata>
    </baskets>
</DatasetIdx16.DataIndex.DatasetMetadata>
```

5. Werkzeuge

5.1. Grundlagen

Die Pflege von Daten- und Modell-Repositories ist entscheidend für das Funktionieren derjenigen INTERLIS-Werkzeuge, die sich darauf abstützen. Bei einer umfangreichen Anzahl von Modellen und Datensätzen ist eine manuelle Nachführung von *ilimodels.xml* und *ilidata.xml* jedoch zeitaufwändig und fehleranfällig. Aus diesem Grund wurde ein spezielles Software-Werkzeug entwickelt, um die Pflege der Index-Dateien zu automatisieren.[4] Zudem gibt es die eine oder andere Option in den herkömmlichen INTERLIS Werkzeugen, welche den Umgang mit Repositories erleichtert.

5.2. Das Werkzeug ilimanager

Die Software *ilimanager* [5] bietet Funktionalität an, um die Arbeit mit Index-Dateien zu automatisieren. Es stehen folgende Funktionen zur Verfügung:

Pflege von ilimodels.xml

- Erstellen von ilimodels.xml, wobei ein lokales Verzeichnis angegeben wird, das nach Modelldateien (.ili) durchsucht wird.
- Aktualisieren von ilimodels.xml aufgrund von Veränderungen im angegebenen Verzeichnis.

Pflege von ilidata.xml

- Erstellen von ilidata.xml, wobei ein lokales Verzeichnis angegeben wird, das nach Transferdateien (.xtf resp. .itf) durchsucht wird.
- Aktualisieren von ilidata.xml aufgrund von Veränderungen im angegebenen Verzeichnis.

Repository-Funktionen

- Klonen eines bestehenden Repositorys.

Beispiele und die Dokumentation des gesamten Funktionsumfangs von *ilimanager* sind in der Dokumentation auf github beschrieben.[6]

5.3. Integritätsprüfung eines Model Repository

Der INTERLIS Compiler (ili2c) bietet die Option an, um ein Model Repository und alle darin indexierten Modelle zu prüfen. Die Option lautet `--check-repo-ilis` und sie wird folgendermassen aufgerufen:

```
java -jar ili2c.jar --check-repo-ilis <url>
```

HINWEIS	Der Kanton Solothurn betreibt basierend auf dieser Funktionalität einen Online-Dienst namens Repo Checker, der regelmässig alle Repositories prüft, die online verfügbar sind (mit https://models.interlis.ch als Einstiegspunkt).

5.4. Integritätsprüfung eines Data Repository

Der INTERLIS Validator (ilivalidator) bietet die Option an, um ein Data Repository und alle darin indexierten Datensätze zu prüfen. Die Option lautet `--check-repo-data` und sie wird folgendermassen aufgerufen:

```
java -jar ilivaldator.jar --check-repo-data <url>
```

[1] Das Vorgängermodell zu `IliRepository20` heisst `IliRepository09`. Dieses unterscheidet sich hauptsächlich darin, dass das Attribut *SchemaLanguage* beschränkt ist auf die Werte *ili1*, *ili2_2* und *ili2_3*. Spätestens wenn also Modelle in der INTERLIS Version

2.4 im Repository verfügbar gemacht werden sollen, muss `ilimodels.xml` auf `IliRepository20` basieren.

[2] Falls ein `&`-Zeichen in der URL enthalten ist, muss es durch die Zeichenfolge `&` ersetzt werden.

[3] siehe z. B. die Online-Suche von ilimodels.ch

[4] Die Datei *ilisite.xml* ist im Gegensatz zu den anderen beiden Index-Dateien sehr statisch; einmal eingerichtet und mit Verweisen auf über- und untergeordnete Repositories versehen, werden daran nur in Ausnahmefällen Änderungen vorgenommen. Es besteht zurzeit kein Software-Werkzeug, welches ilisite.xml automatisch.

[5] Die Software befindet zurzeit in einer Beta-Phase und ist noch nicht offiziell verfügbar. Weitere Details zum Entwicklungsstand können über das github-Repository von ilimanager in Erfahrung gebracht werden.

[6] Am Zweiten INTERLIS Anwender:innen-Treffen im Herbst 2022 fand ein Workshop zum Thema INTERLIS Repositorys statt. Die Unterlagen des Workshops sind im INTERLIS-Forum verfügbar.