OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

DATENMODELLE IN DER ABWASSERABLEITUNG

ANSATZ ZUR KONVERTIERUNG VON ISYBAU-XML-DATEN NACH IFC

MASTERARBEIT

AN DER

OSTBAYERISCHEN TECHNISCHEN HOCHSCHULE
REGENSBURG

FAKULTÄT BAUINGENIEURWESEN

VORGELEGT VON

MATTHIAS WIEDEMANN

MATRIKELNUMMER: 3204304

ABGABETERMIN: 30.11.2020

AUFGABENSTELLER: PROF. Dr.-ING M. OBERGRIESSER
ZWEITPRÜFER: PROF. Dr.-ING. T. EURINGER

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die zur Entstehung dieser Masterarbeit beigetragen haben.

Insbesondere möchte ich mich bei Herrn M. Eng. Friedrich Eder für sein Interesse an der Arbeit und die gute Betreuung bedanken.

Außerdem danken möchte ich Herrn Dipl.-Ing Bernhard Bock, der durch seine Unterstützung diese Arbeit möglich gemacht hat.

Herr Dipl.-Ing. Andreas Geiger vom Karlsruher Institut für Technologie (KIT) hat mich mit Antworten auf meine Fragen und das Bereitstellen einer aktualisierten Version des FZKViewers ebenfalls unterstützt. Dafür möchte ich mich herzlich bedanken.

Bedanken möchte ich mich auch bei der Firma tandler.com, insbesondere bei Herrn Dipl.-Inf. Gerald Angermair, für das Bereitstellen einer Lizenz für die Softwarelösung ++Systems.

Inhalt

1 Einleitung 1

2 Aufgabenstellung 2

2.1 Motivation 2

2.2 Zielsetzung 2

2.3 Lösungsansatz 2

3 Inhalt und Aufbau der auszutauschenden Daten 3

3.1 Datenarten 3

3.2 Objektarten 3

3.3 Objektgeometrie 5

4 Import des ISYBAU-XML-Austauschformates 7

4.1 Grundlagen zum XML-Format 7

4.2 Aufbau des ISYBAU-XML-Formates 8

4.3 Umsetzung des Datenimports 21

5 Konvertierung in das IFC-Austauschformat 30

5.1 Grundlagen zum IFC-Austauschformat 30

5.1.1 Aufbau des IFC-Schemas 30

5.1.2 Beziehungen 35

5.1.3 STEP Physical File 36

5.1.4 Darstellung von Geometrien 39

5.2 Umsetzung der Datenkonvertierung 43

5.2.1 Programmaufbau 43

5.2.2 Gewählte IFC Product Types 49

5.2.3 Geometrische Darstellungsmöglichkeiten 54

5.2.4 Verknüpfung von Semantik und Geometrie 68

5.3 Ergebnisbetrachtung 72

6 Fazit und Ausblick 75

7 Glossar 76

8 Abbildungsverzeichnis 79

9 Tabellenverzeichnis 83

10 Literaturverzeichnis 85

11 Verwendete Programme 87

1 Einleitung

Das Bauwesen erfährt zurzeit eine zunehmende Digitalisierung von Planungs- und Bauprozessen. Ein Begriff, der dabei zwingend fallen muss, ist Building Information Modeling (BIM). Es handelt sich dabei um eine kooperative Arbeitsmethode mit dem Ziel einer höheren Planungs-, Kosten- und Terminsicherheit. Wesentlicher Bestandteil von BIM ist das digitale Modell des Bauwerks, das über dessen gesamten Lebenszyklus von der Planung über die Ausführung bis zum Unterhalt und Rückbau mit Informationen ergänzt wird. Der gewünschte Effekt ist dabei eine größere Informationsdichte, auf welche die Projektbeteiligten über die Projektphasen hinweg zugreifen und somit einem Informationsverlust vorbeugen können.

Im Bereich der abwassertechnischen Anlagen, unter denen zum größten Teil Kanalsysteme verstanden werden können, ergeben sich für BIM viele Anwendungsmöglichkeiten. Dies liegt daran, dass im Laufe des Lebenszyklus eines Kanalsystems viele Maßnahmen wie Sanierungen oder Erweiterungen notwendig werden. Somit findet über einen langen Zeitraum hinweg ein Erfassen, Speichern und Austauschen von Informationen statt, das sich über ein digitales Modell lösen lässt. Und tatsächlich existiert in Deutschland seit 2006 das ISYBAU-XML-Format zum Austausch von Informationen zu abwassertechnischen Anlagen. Über die Jahre wurden eine große Anzahl von Softwareanwendungen zur Planung oder Berechnung von abwassertechnischen Anlagen entwickelt, die das ISYBAU-XML-Format als Standard zum Datenaustausch nutzten.

Mit der zunehmenden Popularität von BIM stellt sich demnach die Frage, wie Planung, Bau und Betrieb von abwassertechnischen Anlagen in BIM untergebracht werden können. Die erste Möglichkeit besteht in der Entwicklung und Erweiterung der Software zur Planung von abwassertechnischen Anlagen. Die zweite Möglichkeit ist die Nutzung der bestehenden Anwendungen in den beiden Bereichen Planung von Abwasseranlagen und BIM. Doch wie soll eine Kopplung dieser Softwareanwendungen erfolgen, wo diese doch keine gemeinsame Schnittstelle teilen? Dieses Problem kann durch Nutzung und Konvertierung von bestehenden standardisierten Austauschformaten gelöst werden. Auf Seiten der Abwasserplanung ist dies das ISYBAU-XML Format, das sich in Deutschland bei den Planungs- und Berechnungsprogrammen etabliert hat. Von BIM Seite eignet sich das Austauschformat der Industry Foundation Classes (IFC), welches als offenes Standardformat zum Austausch von Bauwerksinformationen genutzt wird. Die Entwicklung eines Konverters von ISYBAU nach IFC ergibt somit folgende Vorteile: Es handelt sich bei den Formaten um offene und weit verbreitete Austauschformate, die somit von einer großen Anzahl der Softwareanwendungen genutzt werden können. Zudem ist dies die kostengünstigere Variante im Vergleich zu einer Neuentwicklung von Software für den Anwendungsfall der abwassertechnischen Anlagen. Außerdem existiert bereits ein großer Bestand an ISYBAU-Daten, der so weiter genutzt werden kann.

2 Aufgabenstellung

2.1 Motivation

Diese Arbeit beschäftigt sich mit der Datenhaltung von Informationen zu abwassertechnischen Anlagen über das ISYBAU-XML-Format. Zudem wird das IFC-Format analysiert und dahingehend untersucht, ob eine Konvertierung der ISYBAU-Daten nach IFC möglich ist.

2.2 Zielsetzung

Das Ziel dieser Arbeit ist aufzuzeigen, dass sich IYBAU-XML-Daten in das IFC-Format konvertieren lassen. Dazu sollen theoretische Lösungsansätze ausgearbeitet werden, die eine Konvertierung von möglichst vielen ISYBAU-Daten unterstützt. Als Beweis für die These soll ein Prototyp eines Konverters entwickelt werden, der einen Teil der ISYBAU-Daten auf die theoretisch beschriebene Art in eine funktionierende IFC-Datei konvertieren kann. Die IFC-Datei soll von einem IFC-Viewer gelesen werden können und sowohl die geometrischen als auch semantischen Informationen der ISYBAU-Ausgangsdatei beinhalten.

2.3 Lösungsansatz

Die Lösung der Aufgabe lässt sich untergliedern in zwei Arbeitsschritte: Zunächst ist es notwendig, die ISYBAU-Daten für die Weiterverarbeitung einzulesen. Als zweiter Schritt erfolgt die Übertragung der ISYBAU-Daten in geeignete IFC-Elemente und die Ausgabe als Datei. Die Programmierung erfolgt dabei in C# in der Programmierumgebung Visual Studio von Microsoft. Ausschlaggebend für die Wahl der Programmiersprache war die vorhandene C# API IfcSharp-Library (https://github.com/IfcSharp/IfcSharpLibrary), mit der in C# IFC-Daten erzeugt werden können. Im theoretischen Teil werden vor allem die beiden Austauschformate bezüglich ihrer Schemata analysiert, um zum einen den Aufbau und Informationsgehalt von ISYBAU zu ermitteln und zum andern eine Möglichkeit zu finden, diesen über das IFC-Schema abzubilden.

3 Inhalt und Aufbau der auszutauschenden Daten

Als Hilfeleistung für Planung, Bau und Betrieb abwassertechnischer Anlagen gelten in Deutschland die Baufachlichen Richtlinien Abwasser (BFR-Abwasser).

Herausgeber ist das Bundesministerium des Innern, für Bau und Heimat, sowie das Bundesministerium der Verteidigung. Durch diese Richtlinien erfolgt eine Standardisierung der Planung von abwassertechnischen Anlagen in Liegenschaft des Bundes. Darüber hinaus definierten die BFR-Abwasser das hier behandelte Austauschformat ISYBAU-XML das zum digitalen Informationsaustausch von abwassertechnischer Anlagen herangezogen wird.

3.1 Datenarten

Die BFR-Abwasser unterscheiden zwischen zwei Datenarten. Diese sind Geometriedaten und abwassertechnische Fachdaten. Geometriedaten umfassen durch Vermessung ermittelte Lage und Höhendaten zur Abbildung von Liegenschaften. Die abwassertechnischen Fachdaten beinhalten alle für die Planung, den Bau und den Betrieb benötigten Daten. (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 53) Weiter untergliedert werden die abwassertechnischen Fachdaten in Stammdaten, bautechnische Zustandsdaten und hydraulische Zustandsdaten. Diese Daten können mithilfe der ISYBAU-XML-Schnittstelle ausgetauscht werden. Der bautechnische Bestand, wie zum Beispiel geometrische Lage oder Material einer abwassertechnischen Anlage, wird in den Stammdaten beschrieben. Da sich bautechnische Zustandsdaten und hydraulische Zustandsdaten auf die Stammdaten beziehen, müssen die Stammdaten entweder bereits vorhanden sein, oder beim Datenaustausch mitübergeben werden. (Bundesministerium des Inneren für, Bau und Heimat, 2019, pp. 53, 54) Die eingangs erwähnten Geometriedaten der Liegenschaft werden nicht mit der ISYBAU-XML-Datei übergeben.

3.2 Objektarten

Durch die BFR-Abwasser werden abwassertechnische Anlagen weiter untergliedert. (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 59) Grundsätzlich erfolgt dies durch die Unterteilung in die Objektarten

- Knoten
- Kante

Es existieren die drei Knotentypen „Schacht“, „Anschlusspunkt“ und „Bauwerk“. Damit werden sämtliche Schächte aber auch Sonderbauwerke wie beispielsweise Pumpwerke oder Becken, die Teil eines

Abwassernetzes sein können abgedeckt. Über Anschlusspunkte werden zum einen allgemeine Anschlüsse, beispielsweise einer Leitung an eine Haltung, aber auch Anschlüsse wie Hausanschlüsse oder ein Anschluss an einen Straßenablauf erfasst.

Die Objektart „Kante" wiederum wird unterteilt in die vier Kantentypen „Haltung", „Leitung", „Rinne" und „Gerinne". Eine Haltung beschreibt den „Abschnitt einer Abwasserleitung oder eines Abwasserkanals zwischen zwei angrenzenden Knotenpunkten" (Deutsches Institut für Normen e.V., 2014, p. 31). Ihre Aufgabe ist in der Regel das Sammeln und der Abtransport des Abwassers zu eine Abwasserbehandlungsanlage wie beispielsweise einer Kläranlage oder einer Entlastungsanlage. Die Leitung ist definiert als ein „meist erdverlegtes Rohr zur Ableitung von Abwasser von der Anfallstelle zum Abwasserkanal" (Deutsches Institut für Normen e.V., 2014, p. 30). Ein Beispiel für eine Leitung wäre also eine Hausanschlussleitung, die das häusliche Abwasser zu einer Haltung oder direkt zu einem Schacht ableitet. Die beiden Kantentypen „Rinne" und „Gerinne" sind weitere abwassertechnische Anlagen zum Abtransport von Abwasser. Rinnen können sowohl offen als auch abgedeckt ausgeführt werden. Ein Gerinne kann ein Fließgewässer, Entwässerungsgraben oder beispielsweise auch ein Messgerinne sein.

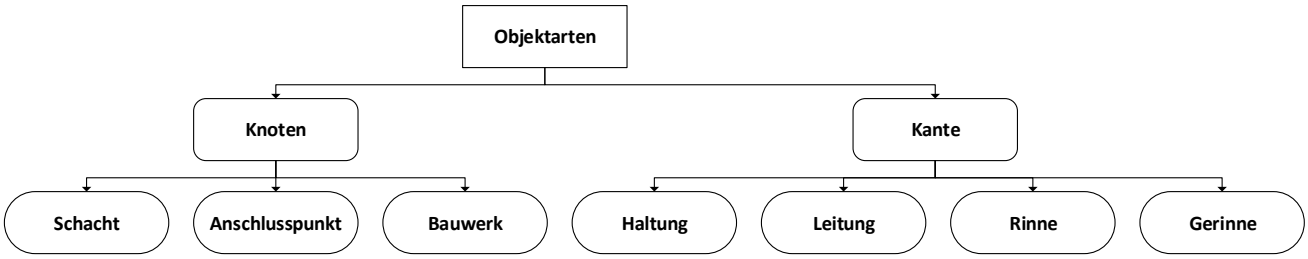


Abbildung 3-1: Aufgliederung der in ISYBAU abgebildeten Objektarten Knoten und Kante

Diese Unterteilung ist wichtig, da sie sich auch im Stammdatenbereich der ISYBAU-XML-Austauschdatei wiederfindet.

3.3 Objektgeometrie

Neben semantischen Daten zur Beschreibung von abwassertechnischen Anlagen in ihrer Substanz enthalten die Stammdaten auch geometrische Informationen. Dabei existieren die drei Geometrietypen Knoten, Kante und Polygon.

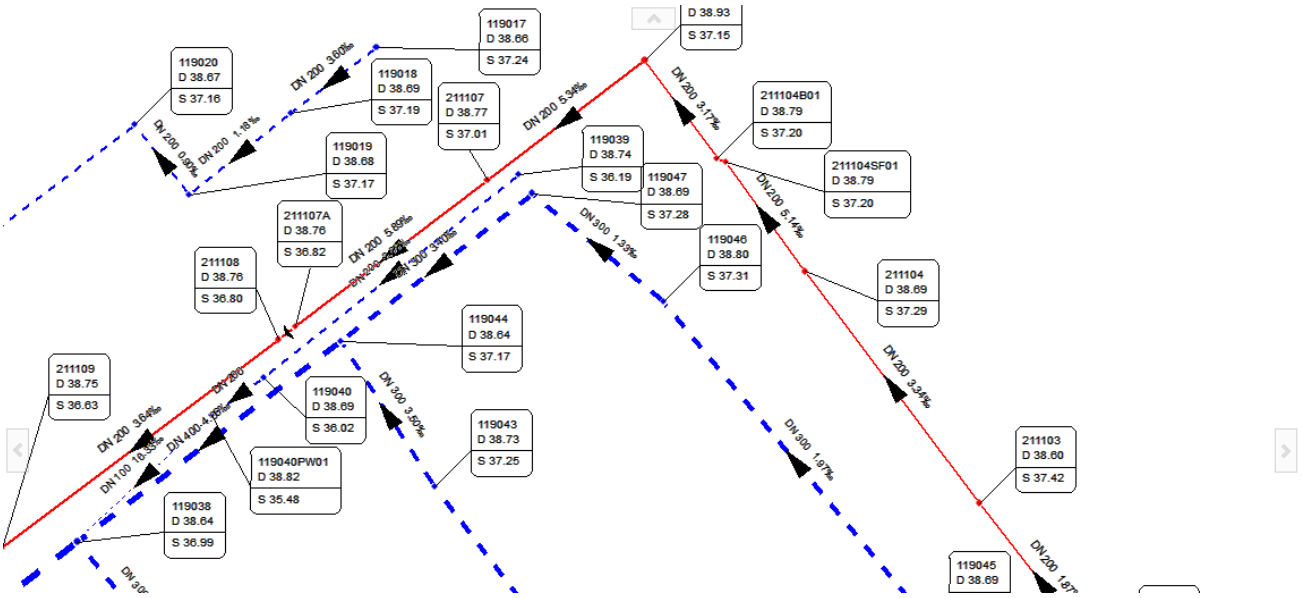
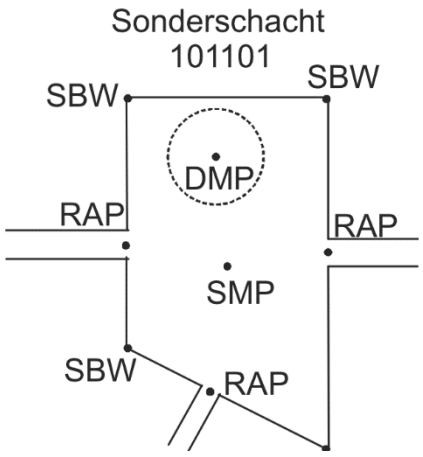


Abbildung 3-2: Darstellung eines Abwassernetzes als Knoten-Kantenmodell in ++Systems

Knoten sind Punkte im dreidimensionalen Raum und können beispielsweise markante Punkte eines Schachtbauwerkes wie den Schachtmittelpunkt abbilden. Eine Kante wird definiert durch zwei miteinander verbundene Punkte und beschreibt klassischerweise den Verlauf von Haltungen. Kanten sind jedoch beschränkt auf die geradlinige Verbindung von zwei Knoten. Um beispielsweise geknickte Leitungsverläufe darzustellen werden Polygone durch Zusammensetzten mehrerer Kanten gebildet. Geschlossene Polygonzüge können genutzt werden, um beispielsweise Bauwerkskonturen von Sonderbauwerken abzubilden. Bei Knoten ist „Punktattribut Abwasser“ ein wichtiges Attribut, dass die Funktion des Kontenpunktes genauer beschreibt. Beispielsweise erhält ein Knoten das Punktattribut Abwasser „SMP“ zur Deklaration als Schachtmittelpunkt. Ähnlich verhält es sich bei Polygonen mit dem Attribut „Polygonart“. Hier stehen die drei Polygonarten „innerer Polygonring eines Objektes (geschlossen)“, „äußerer Polygonring eines Objektes (geschlossen)“ und „Polylinie eines Objektes (offen)“ zur Auswahl. (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 820) Es ist wichtig



festzuhalten, dass Knotenobjekte wie Schächte oder Bauwerke geometrisch nicht nur über Knoten dargestellt werden müssen, sondern auch über Kanten beschrieben werden können.

Abbildung 3-3: Abbildung der Schachtgeometrie eines Sonderschachtes durch einen geschlossenen Polygonzug über die Punkte mit den Punktattributen „SBW“. Die Punkte „RAP“ markieren den Rohranschlusspunkt eines Zu- oder Ablaufs. „SMP“ (Schachtmittelpunkt) und „DMP“ (Deckelmittelpunkt) sind weitere Punkte zur Abbildung des Schachtes. (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 71)

4 Import des ISYBAU-XML-Austauschformates

4.1 Grundlagen zum XML-Format

Extensible Markup Language, kurz XML, ist ein textbasiertes Format zur Abbildung strukturierter Informationen. Entwickelt wurde das XML-Format vom World Wide Web Consortium (World Wide Web Consortium, 2019). Beim World Wide Web Consortium (W3C) handelt es sich um ein internationales Konsortium zur Entwicklung von Web-Standards (World Wide Web Consortium, 2020). Neben dem XML-Standard lassen sich beispielhaft die beiden Programmiersprachen HTML (Hypertext Markup Language) und CSS (Cascading Style Sheets), die zur Strukturierung und zum Layouten von Webseiten herangezogen werden, als weitere vom W3C entwickelte Technologien aufführen. (World Wide Web Consortium, 2019). Mittlerweile hat sich XML als weit verbreitetes Austauschformat etabliert, da damit strukturiert viele unterschiedliche Daten ausgetauscht werden können.

Für den einheitlichen Austausch besteht die Möglichkeit ein sogenanntes XML-Schema zu definieren. Hierfür stehen verschiedene Sprachen zur Beschreibung eines XML-Dokumentes zur Verfügung. Beispielsweise Document Type Definition (DTD) oder XML-Schema Definition (XSD). Das Schema beschreibt über eine Ansammlung von Elementen und Attributen den allgemeinen, baumförmigen Aufbau einer Austauschdatei. Es legt fest, welche Elemente und Attribute an welcher Stelle und wie häufig auftreten können oder gar müssen und welchen Datentypen sie entsprechen. Dadurch weiß ein Programm einerseits wie es Daten zu exportieren hat und andererseits in welcher Form Daten zum Import in einer Austauschdatei auftreten können. (World Wide Web Consortium, 2019) Auf den konkreten Aufbau der ISYBAU-XML-Austauschformates wird mit Beispielen im folgenden Kapitel eingegangen.

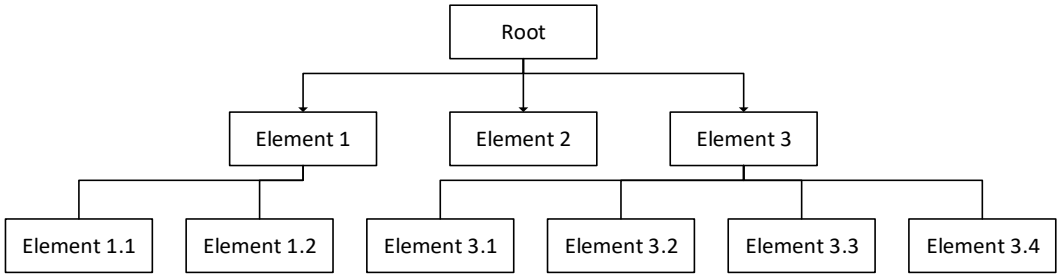


Abbildung 4-1: Baumförmiger Aufbau des XML-Schemas, ausgehend von einem Root-Element.

4.2 Aufbau des ISYBAU-XML-Formates

Über das ISYBAU-XML-Format kann ein standardisierter Austausch von Daten zu abwassertechnischen Anlagen, wie sie in den Baufachlichen Richtlinien Abwasser beschrieben sind, erfolgen. Ein solcher Austausch ist in verschiedenen Projektphasen zwischen unterschiedlichen Projektbeteiligen notwendig, beispielsweise bei der Erfassung von Bestandsdaten, der hydraulischen Berechnung von bestehenden oder neu geplanten Kanalnetzen oder der Bestandsdokumentation nach Durchführung einer Maßnahme. Hierfür muss in der jeweilige Softwarelösung eine ISYBAU-XML-Schnittstelle integriert sein. Das ISYBAU-Format selbst ist ebenfalls in den Baufachlichen Richtlinien Abwasser definiert und daher in Deutschland das Standardformat zum Austausch von abwassertechnischen Daten. (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 673)

Das ISYBAU-Schema ist im XSD-Format definiert. Dessen Aufbau wird im Folgenden genauer analysiert, da Inhalt und Aufbau von ISYBAU-XML die Grundlage für die weitere Bearbeitung der Daten hin zum IFC-Austauschformat bilden.

Untergliedert ist das ISYBAU-Schema nach Bundesministerium des Inneren für Bau und Heimat (2019, p. 677) in folgende sogenannte Datenbereiche:

- Metadaten
- Stammdaten
- Zustandsdaten
- Hydraulische Daten
- Betriebsdaten
- Präsentationsdaten

Diese Aufteilung erfolge aus Gründen der Übersichtlichkeit auch für die Schemadateien, was zur Folge hat, dass das Schema für das ISYBAU-Austauschformat nicht in einer, sondern in mehreren Dateien definiert wird. Zusätzlich führt das Schema auch sogenannte Referenzlisten. Diese beinhalten eine Liste von vorgegebenen Werten, die ein Element in der Austauschdatei annehmen kann. Auch die Referenzlisten werden in einer eigenen Schemadatei geführt, sodass das ISYBAU-XML-Schema in der 2017-07er Version über die sieben, in Abbildung 4-2 dargestellten .xsd-Dateien definiert wird.

Name	Änderungsdatum	Typ	Größe
1707-betriebsdaten.xsd	19.11.2018 11:26	XML Schema File	10 KB
1707-hydraulikdaten.xsd	19.11.2018 11:27	XML Schema File	89 KB
1707-metadaten.xsd	19.11.2018 11:27	XML Schema File	24 KB
1707-praesentationsdaten.xsd	19.11.2018 11:27	XML Schema File	8 KB
1707-referenzlisten.xsd	19.11.2018 11:27	XML Schema File	249 KB
1707-stammdaten.xsd	19.11.2018 11:27	XML Schema File	80 KB
1707-zustandsdaten.xsd	19.11.2018 11:27	XML Schema File	41 KB

Abbildung 4-2: .xsd-Dateien zur Beschreibung des ISYBAU-XML-Schemas

Auch in einer ISYBAU-Austauschdatei lässt sich diese Struktur wiederfinden. Die Informationen sind aufgeteilt in Datenkollektive. Dabei bilden die Metadaten den Kopf des ISYBAU-Austauschformates. Die Datenkollektive zu denen Stammdaten, Zustandsdaten, Hydraulikdaten, Betriebsdaten und Präsentationsdaten gehören, sind ein Unterelement der Metadaten. Die Referenzlisten tauchen als solche in der Austauschdatei nicht auf. Allerdings referenzieren bestimmte Elemente die in den Referenzlisten vorgegebenen Werte. Beim Austausch werden nicht wie bei der Schemadefinition mehrere Austauschdateien benötigt. Es können beispielsweise Stamm-, Hydraulik- und Betriebsdaten in einer Austauschdatei übermittelt werden.

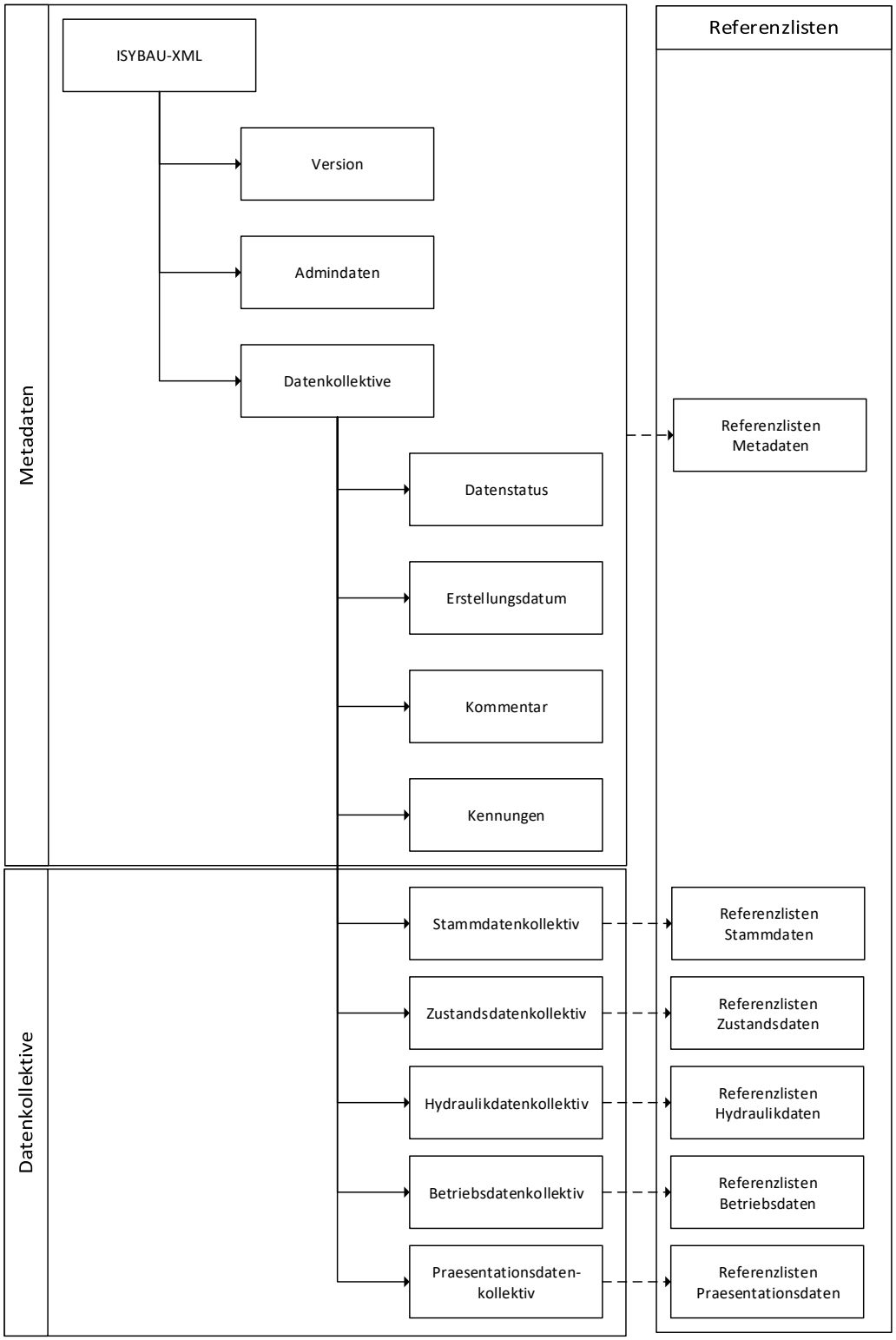


Abbildung 4-3: Strukturierung des ISYBAU-Austauschformates

Die Metadaten bilden den Kopf einer ISYBAU-Austauschdatei. Sie enthalten Informationen, die für die gesamte Datei gültig sind. Das oberste Element der Metadaten ist die `Identifikation.` Sie bildet das Root-Objekt für das gesamte Schema, was bedeutet, dass jedes im ISYBAU-XML-Austauschformat enthaltene Element ein Subelement von `Identifikation ist.` Die ersten drei Unterelemente von `Identifikation` sind `Version`, `Admindaten` und `Datenkollektive. Version` enthält als Wert die ISYBAU-XML-Version der Austauschdatei, beispielsweise `2017-07` für die aktuell neueste Version. Die `Admindaten` enthalten mehrere Subelemente, die allgemeine administrative Informationen enthalten, unter anderem zu Projektbeteiligten, zuständigen Behörden oder zur Liegenschaft. Der Knoten `Datenkollektive` enthält neben den Stammdaten-, Zustandsdaten-, Hydraulikdaten-, Betriebsdaten- und Präsentationsdatenkollektive auch allgemeine Informationen zu den Datenkollektiven.

Über die Stammdaten werden die abwassertechnischen Anlagen in ihrer Geometrie und Substanz beschrieben. (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 686) Das bedeutet, dass über Koordinaten die Lage und über weitere Informationen wie Durchmesser das Objekt in seiner Form dargelegt wird. Informationen beispielsweise zum Material geben des Weiteren Aufschluss über die Substanz der Anlagen. Zudem enthalten die Stammdaten Informationen zu Dokumenten und Sanierungsdaten. Geometrisch wird die in den BFR beschriebene Gliederung in Kanten- und Knotenmodelle umgesetzt. Das bedeutet im Weiteren die Untergliederung in Haltungen, Leitungen, Rinnen, Gerinne, Schächte, Anschlusspunkte und Bauwerke. In den Stammdaten wird jedes dieser Objekte als eigene abwassertechnische Anlage geführt und beschrieben.

Das Zustandsdatenkollektiv enthält Inspektionsdaten zu Haltungen, Leitungen, Schächten und Bauwerken. Die Daten umfassen Informationen zu optischen Inspektionen, Dichtheitsprüfungen und Zustandsfilmen. Aufgrund der Inspektionsart wird in den Zustandsdaten differenziert zwischen Rohrleitungen, Knoten und Bauwerken. Bei Rohrleitungen, also Haltungen und Leitungen, erfolgt die Dokumentation entsprechend der Inspektionsrichtung horizontal, bei Knoten, beispielsweise einem Schacht, in vertikaler Richtung. Bauwerke erhalten eine Zustandsbeschreibung in textlicher Form. (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 718)

Die Hydraulikdaten „ermöglich[en] die vollständige Ablage hydrologischer/hydraulischer Zustandsdaten“ (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 739). Sie beinhalten Zielgrößen, objektbezogene hydraulische Stammdaten, Einzugsgebiete und Flächen, Belastungsdaten wie Niederschlagsreihen oder Trockenwetterabflüsse und Berechnungsergebnisse. (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 739)

Im Betriebsdatenkollektiv werden Beobachtungen zu Grundwasser und Boden dokumentiert. (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 768)

Präsentationsdaten beinhalten Informationen zur Plandarstellung, beispielsweise die Darstellung und Positionierung von Texten oder Symbolen. (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 773)

Für den Schritt des Datenimportes von ISYBAU-Daten ist es zunächst hilfreich den konkreten Aufbau und die einzelnen Funktionen der Schemaelemente genauer zu untersuchen. Da der Dateninhalt und die grobe Aufgliederung aus den vorangegangenen Abschnitten bekannt ist, erfolgt nun eine Analyse des XSD-Schemas anhand des ISYBAU-Schemas[1]. Im Folgenden wird das in der .xsd-Datei abgebildete XSD-Schema als „Schema“ und die ISYBAU Austauschdatei als „Instanz“ bezeichnet. Der Umfang der Erläuterungen beschränkt sich an dieser Stelle auf diejenigen Definitionen, die auch im ISYBAU-Schema Anwendung finden. Grundsätzlich stehen einem XSD-Schema weitaus mehr Möglichkeiten zur Verfügung, um den Aufbau einer XML-Datei zu beschreiben. Die folgenden Erläuterungen und Beispiele sind angelehnt an „XML Schema Teil 0: Einführung“ vom <edition W3.de> Projekt. Dabei handelt es sich um eine Übersetzung des Originaltextes vom W3C aus dem Englichen ins Deutsche. (<edition W3>, 2020)

Das Ziel des Schemas ist es zu definieren, welche Elemente in einer Instanz auftreten können. Dabei wird festgelegt an welcher Stelle, mit welcher Häufigkeit und in welcher Form die Elemente auftreten. Ein einfaches Beispiel ist das Element `Version`. `<xsd:element` zeigt an, dass es sich um ein Element handelt. In der Instanz entspricht ein Element einem Knoten in der XML-Datei. Der Name des Elementes, beziehungsweise des Knotens wird mit `name="Version"` festgelegt. Jedes Element muss einem Datentyp zugeordnet werden. In diesem Falls ist der Datentyp `gYearMonth`. Im Folgenden sind der Schemaausdruck und das Beispiel für die zugehörige Instanz aufgeführt. Als Beispieldateien werden die offiziellen von der BFR herausgegebenen Musterdateien verwendet[2]. Wie man sieht, wird der Knoten `Version` durch `<Version>` geöffnet. Anschließend folgt der Wert, der aufgrund des Datentyps `gYearMonth` aus einer Jahreszahl gefolgt von einem Monat bestehen muss. `</Version>` beendet die Syntax dieses Knotens.

1707-metadaten.xsd
`<xsd:element name="Version" type="xsd:gYearMonth"/>`
ISYBAU_XML-2017-Stammdaten.xml
`<Version>2017-07</Version>`

Tabelle 4-1: Beispiel zur Definition eines Elementes über einen `simpleType`

[1] Link zu den XSD-Schemadateien: https://www.bfr-abwasser.de/html/Materialien.1.40.html Zugriff am 02.03.2020
[2] Link zu den Beispieldatensätzen: https://www.bfr-abwasser.de/html/Materialien.1.39.html Zugriff am 31.03.2020

Der hier zugewiesene Datentyp `gYearMonth` gehört zu den `simpleTypes`. Die `simpleTypes` können als Standarddatentypen verstanden werden, die vom World Wide Web Consortium vorgegeben werden und in einem XML-Schema auftreten können. Im ISYBAU-Schema treten die folgenden `simpleTypes` auf:

`integer`	Numerisches Datenfeld, das eine mathematische ganze Zahl enthält. Plus- und Minuszeichen sind als Vorzeichen erlaubt.
`decimal`	Numerisches Datenfeld, das eine Dezimalzahl mit einer beliebigen endlichen Zahl der Ziffern 0 bis 9 vor und hinter dem Dezimaltrennzeichen zulässt. Als Dezimaltrennzeichen ist nur der Punkt zulässig. Die Vorzeichen Plus und Minus sind erlaubt. Exponentialschreibweisen werden nicht unterstützt.
`double`	Numerisches Datenfeld, das eine Gleitkommazahl in Dezimal- oder Exponentialschreibweise zulässt. Als Dezimaltrennzeichen ist nur der Punkt zulässig. Leerzeichen sind nicht zulässig.
`string`	Alphanumerisches Datenfeld, das eine beliebige Folge von Zeichen einschließlich Leerzeichen, die in einem XML 1.0-Dokument erlaubt sind, enthalten kann.
`token`	Alphanumerisches Datenfeld beliebiger Länge, das eine beliebige Folge von Zeichen einschließlich Leerzeichen, die in einem XML 1.0-Dokument erlaubt sind, enthalten kann.
`boolean`	Datenfeld, das einen logischen Booleschen Ausdruck enthält. Zulässig sind die Werte „1“, „0“, „true“, „false“.
`date`	Datenfeld, das eine Kalenderdatumsangabe enthält, die von ISO 8601 definiert wird. Zulässig ist die Angabe des Datums im Format JJJJ-MM-DD.
`gYearMonth`	Datenfeld, das eine Jahres- und Monatsangabe enthält, die von ISO 8601 definiert wird. Zulässig ist die Angabe im Format JJJJ-MM.
`gYear`	Datenfeld, das eine Jahresangabe enthält, die von ISO 8601 definiert wird. Zulässig ist die Angabe des Jahres im Format JJJJ.
`time`	Datenfeld, das eine Zeitangabe enthält, die von ISO 8601 definiert wird. Zulässig ist die Angabe der Uhrzeit im Format hh:mm:ss.

Tabelle 4-2: Definitionen laut BFR zu den im ISYBAU-Verwendeten simpleTypes (Bundesministerium des Inneren für, Bau und Heimat, 2019, pp. 676, 677)

Eine andere Art von Datentypen sind `complexTypes`. Sie sind keine vom W3C vorgegebenen Datentypen, sondern können vom Ersteller der XDS-Datei selbst definiert werden. Um ein Element einem `complexTypes` zuzuweisen gibt es zwei Möglichkeiten, die im folgenden Schemaauszug beide zur Anwendung kommen.

1707-stammdaten.xsd

```
<xsd:element name="Auftraege" minOccurs="0">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element name="Auftrag" type="StammAuftragType" maxOccurs="unbounded"/>
		</xsd:sequence>
	</xsd:complexType>
</xsd:element>
```

Tabelle 4-3: Beispiel zur Definition eines Elementes über einen `complexType`

Betrachten wir zunächst das Element `Auftrag`. Hier liegt eine der Zuweisung zu einer `simpleType`-ähnliche Syntax vor. Über den Befehl `type=` wird der Datentyp `StammAuftragType` zugewiesen. Dieser Datentyp ist in diesem Fall im selben XSD-Dokument definiert. Diese Art der Zuweisung zu einem komplexen Datentyp wird als „global" bezeichnet. Dadurch dass der Datentyp `StammAuftragType` global definiert ist, können auch andere Elemente neben `Auftrag` diesem Datentyp zugewiesen werden.

1707-stammdaten.xsd

```
<xsd:complexType name="StammAuftragType">
	<xsd:sequence>
		<xsd:element name="Auftragsbezeichnung">
			<xsd:simpleType>
				<xsd:restriction base="xsd:string">
					<xsd:maxLength value="60"/>
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:element>
		[. . .]
		<xsd:element name="Auftragsdatum" type="xsd:date" minOccurs="0"/>
		<xsd:element name="Auftragsart" type="AuftragsartSanierungType"/>
		[. . .]
	</xsd:sequence>
</xsd:complexType>
```

Tabelle 4-4: Definition des `complexTypes StammAuftragType`

Die Definition eines globalen `complexTypes` wird mit `<xsd:complexType>` und Festlegen des Datentypnamens eröffnet. Der darauf folgende Befehl `<xsd:sequence>` indiziert, dass der hier beschriebene Datentyp aus Subelementen besteht, die an dieser Stelle aufgelistet werden. Der oben angegebene Quelltext zum `StammAuftragType` ist aus Gründen der Übersichtlichkeit durch `[. . .]` auf drei Elemente gekürzt. Das erste Element `Auftragsbezeichnung` stellt eine Sonderform von `simpleTypes` dar, die im ISYBAU-Schema häufig zur Anwendung kommt. Hierbei wird ein `simpleType` abgewandelt oder eingeschränkt. In diesem Fall drückt `<xsd:restriction base="xsd:string">` aus, dass es sich um einen `simpleType` auf Grundlage eines `string` handelt. Als Einschränkung wird die maximale Länge des `strings` auf 60 Zeichen limitiert.

Das Element `Auftragsdatum` wird auf bekannte Art einem `simpleType` zugeordnet, während die `Auftragsart` vom `complexType AuftragsartSanierungType` ist. Dadurch, dass innerhalb eines `complexTypes` Elemente auftreten können, die wiederum von ein `complexType` sind, entsteht die tiefgehende Gliederung in der Instanz.

Nachdem wir nun anhand des Beispiels `Auftrag` aus Tabelle 4-3 eine Möglichkeit zur Definition von `complexTypes` kennen gelernt haben, betrachten wir in derselben Tabelle das Element `Auftraege.` Der zugehörige Datentyp wird nicht über `type=` zugewiesen, sondern unterhalb des Elementes definiert. Die Syntax gleicht der Syntax zur Definition eines globalen `complexTypes`, allerding wird dem Datentyp kein eigener Name zugewiesen. Als Resultat können keine anderen Elemente diesem lokalen Datentyp zugewiesen werden. Es ist daher vor allem sinnvoll einen globalen Datentyp zu definieren, wenn in einem Schema mehrere Elemente dieses Datentyps auftreten. Ein Datentyp, der einmalig von einem Element herangezogen wird, kann lokal definiert werden. Aus Gründen der Übersichtlichkeit eines Schemas kann es allerdings auch hilfreich sein, einmalig zugewiesene Datentypen global zu definieren.

In den oberen Beispielen traten bereits Angaben zur Häufigkeit eines Elementes auf. Diese sollen nun genauer erläutert werden. Um festzulegen, wie häufig ein Element, also ein Knoten in einer Instanz, auftreten darf, stehen einem Element die Attribute `minOccurs` und `maxOccurs` zur Verfügung. Der Wert von `minOccurs` legt fest wie oft der Konten mindestens in einer Instanz auftreten muss. Wird im ISYBAU-Schema die Häufigkeit eines Elements über `minOccurs` beschrieben, so nimmt dessen Wert meistens `„0“` an. Dies ist gleichbedeutend mit der Aussage, dass dieses Element optional ist. Es kann auftreten, muss aber nicht. Das maximale Auftreten eines Knotens in einer Instanz wird über `maxOccurs` festgelegt. Im ISYBAU-Schema nimmt dieser Wert überwiegend `„unbounded“` an. Damit können unendlich viele dieser Knoten in einer Instanz auftreten. Die Standartwerte für `minOccurs` und `maxOccurs` sind „1“. Werden zu diesen beiden Eigenschaften wie etwa beim Element `Version` in Tabelle 4-1 keine Angaben gemacht, so ist dies gleichzusetzten mit den Angaben `minOccurs=“1“` und `maxOccurs=“1“.` Damit diese Forderungen erfüllt sind, muss der Knoten `Version` genau einmal in der Instanz auftreten. Es lassen sich somit verschieden Kombinationen von Mindestangabe und maximaler Beschränkung durchführen, wobei lediglich beachtet werden muss, dass `maxOccurs` keinen Wert kleiner als `minOccurs` annehmen darf, da ansonsten ein logischer Fehler auftreten würde. (<edition W3>, 2020)

Es besteht auch die Möglichkeit, dass eine Auswahl aus zwei oder mehr Elementen getroffen werden muss. Das Beispiel des `complexType AbwassertechnischeAnlageType in` Tabelle 4-5 verdeutlicht, wann und warum eine solche Auswahl getroffen werden muss. Durch `<xsd:choice>` und den darauf folgenden Elementen `Kante` und `Knoten` steht im `AbwassertechnischeAnlageType` eine Auswahl zwischen diesen zur Verfügung. Dabei kann nur eines dieser Elemente in der Austauschdatei als Knoten auftreten. Eine Auswahl über `<xsd:choice>` tritt demzufolge immer dann in einem Schema auf, wenn an einer bestimmten Stelle theoretisch mehrere unterschiedliche Abzweigungen im XML-Schema möglich sind, in der konkreten Instanz jedoch nur ein Knoten vorkommen darf. Im Beispiel ist dies der Fall, da der Datentyp `AbwassertechnischeAnlageType` zwar die Optionen `Kante` und `Knoten` zur Verfügung stellen muss, die

tatsächliche abwassertechnische Anlage, die in einer ISYBAU-Datei beschrieben wird, allerdings nur entweder `Kante` oder `Knoten` sein kann.

1707-stammdaten.xsd

```
<xsd:complexType name="AbwassertechnischeAnlageType">
	<xsd:sequence>
		[. . .]
		<xsd:choice>
			<xsd:element name="Kante" type="StammKanteType" minOccurs="0"/>
			<xsd:element name="Knoten" type="StammKnotenType" minOccurs="0"/>
		</xsd:choice>
		[. . .]
	</xsd:sequence>
</xsd:complexType>
```

Tabelle 4-5: Implementierung eines choice-Elements am Beispiel von `AbwassertechnischeAnlageType`

Ein weiterer Punkt, der in den Beispielen auftritt und noch nicht behandelt wurde, sind Namespaces und Präfixe. Ein Namespace, deutsch auch Namensraum genannt, vereint Datentypen und Elemente. Alle Typen und sonstige verwendete Elemente einer Schemadatei müssen in einem Namespace definiert sein, damit sie verwendet werden können. Der Verweis auf die Namensräume erfolgt in den Schemadateien in der zweiten Zeile im Bereich `<xsd:schema/>`

1707-metadaten.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.bfr-abwasser.de"
xmlns:isy="http://www.bfr-abwasser.de"          targetNamespace="http://www.bfr-abwasser.de"
elementFormDefault="qualified">
```

Tabelle 4-6: Festlegen der Namespaces

Hier werden Namespaces Präfixe zugewiesen. Im Beispiel erhält der Namespace `"http://www.w3.org/2001/XMLSchema"` das Präfix `xsd` zugewiesen. Hinter diesem Namespace verbergen sich die von der W3C vorgegebenen Elemente und Datentypen, also zum Beispiel alle `simpleTypes`. Um in Schema auf diese zugreifen zu können, muss vor dem jeweiligen Element oder Datentyp das Präfix `xsd:` angegeben werden. Daraus folgt, dass all diese Elemente und Typs in einer eigenen Datei definiert worden sein müssen. Der Namespace wird bei der Verwendung von Microsoft Visual Studio erkannt, und das Programm greift automatisch auf die hinterlegte XSD-Datei „xsdschema.xsd“ zurück. Die Definition des `simpleType gYearMonth` sieht beispielsweise in der xsdschema.xsd-Datei wie folgt aus:

xsdschema.xsd

```
<xs:simpleType name="gYearMonth" id="gYearMonth">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#gYearMonth"/>
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse"  fixed="true"
      id="gYearMonth.whiteSpace"/>
  </xs:restriction>
</xs:simpleType>
```

Tabelle 4-7: Definition des `simpleTypes gYearMonth` in der xsdschema.xsd-Datei

Im Schema wird darüber hinaus der `targetNamespace` deklariert. Dieser legt den Namespace-Namen des vorliegenden Schemas fest. Das bedeutet, dass alle Elemente und Datentypen des ISYBAU-Schemas dem Namespace `http://www.bfr-abwasser.de` zugeordnet sind. Um auf diese auch zugreifen zu können muss dieser Namespace dennoch ein Präfix erhalten. Hierbei nutzt man die Möglichkeit, dass dieses Präfix auch leer sein kann. Wie man am Ausdruck `xmlns="http://www.bfr-abwasser.de"` sehen kann wird dazu nach `xmlns` keine Angabe zum Präfix gemacht. Veranschaulichen lässt sich dies anhand der beiden bekannten Elemente `Auftragsdatum` und `Auftragsart`. Sowohl vor dem Elementaufruf, als auch vor dem `simpleType date`, dem das Auftragsdatum zugeordnet wird, findet sich der Präfix `xsd:`, damit das Schema auf die jeweiligen Definitionen aus der xsdschema.xsd-Datei zugreifen kann. Vor dem Datentyp `AuftragsartSanierungType` findet sich kein Präfix, wodurch das Schema den Datentyp im vorliegenden ISYBAU-Schema sucht. Im folgenden Beispiel wurde der Namespace um das Präfix „`isybau`“ ergänzt, um die Auswirkung darstellen zu können

1707-metadaten.xsd

```
<xsd:element name="Auftragsdatum" type="xsd:date" minOccurs="0"/>
<xsd:element name="Auftragsart" type="AuftragsartSanierungType"/>
```

1707-metadaten.xsd modifiziert

```
<xsd:schema [. . .]" xmlns:isybau="http://www.bfr-abwasser.de" [. . .]
targetNamespace="http://www.bfr-abwasser.de" elementFormDefault="qualified">
[. . .]
<xsd:element name="Auftragsdatum" type="xsd:date" minOccurs="0"/>
<xsd:element name="Auftragsart" type="isybau:AuftragsartSanierungType"/>
```

Tabelle 4-8: Veranschaulichung der Funktionsweise von Präfix durch Ergänzung eines Präfixes „`isybau`“

Eine weitere Schemaeigenschaft, die im ISYBAU-Schema festgelegt wird, ist `elementFormDefault`, die auf `qualified` gesetzt ist. Diese Eigenschaft gibt an, dass in einer Instanz in den Knoten ebenfalls mit Präfixen gearbeitet werden muss. (<edition W3>, 2020) Im Falle von ISYBAU hat dies keinen Einfluss, da in der Instanz lediglich auf Objekte aus dem `http://www.bfr-abwasser.de`-Namespace zurückgegriffen wird, deren Präfix leer ist. Zur Veranschaulichung werden im Folgenden die Auswirkung auf eine Instanz des ISYBAU-Schemas durch die Änderung des Präfixes auf `isybau` dargestellt.

ISYBAU_XML-2017-Stammdaten_Sanierung_Abnahme.xml
<Auftragsdatum>2013-01-02</Auftragsdatum> <Auftragsart>2</Auftragsart>
ISYBAU_XML-2017-Stammdaten_Sanierung_Abnahme.xml modifiziert
<isybau:Auftragsdatum>2013-01-02</isybau:Auftragsdatum> <isybau:Auftragsart>2</isybau:Auftragsart>

Tabelle 4-9: Auswirkungen von `elementFormDefault="qualified"`, veranschaulicht durch die Ergänzung des Präfixes „isybau“

Wie eingangs erwähnt, ist das ISYBAU-Schema auf mehrere .xsd-Dateien aufgeteilt. Um dennoch auf die Elemente und Datentypen der anderen Schemadateien zugreifen zu können, werden diese in der 1707-metadaten.xsd-Datei über den Befehl `include` dem Schema hinzugefügt. Die `schemaLocation` gibt den Pfad an, in dem sich die XSD-Datei befindet. Eine Voraussetzung ist jedoch, dass die `targetNamespaces` der zu inkludierten Dateien mit denen der Schemadatei übereinstimmen. (<edition W3>, 2020)

1707-metadaten.xsd
<xsd:include schemaLocation="./1707-stammdaten.xsd"/> <xsd:include schemaLocation="./1707-zustandsdaten.xsd"/> <xsd:include schemaLocation="./1707-referenzlisten.xsd"/> <xsd:include schemaLocation="./1707-hydraulikdaten.xsd"/> <xsd:include schemaLocation="./1707-betriebsdaten.xsd"/> <xsd:include schemaLocation="./1707-praesentationsdaten.xsd"/>

Tabelle 4-10:Einbinden der Schemadateien in die 1707-metadaten.xsd-Datei

Die 1707-referenzlisten.xsd-Datei unterscheidet sich von den restlichen Schemadateien. In ihr werden keine Elemente oder `complexTypes` definiert. Stattdessen werden `simpleTypes`, in ähnlicher Weise wie es bereits bei `Auftragsbezeichnung` geschehen ist, deklariert. Diese Datentypen werden von den restlichen Schemata referenziert. Die Besonderheit dabei ist, dass die Elemente beziehungsweise Koten, die diesen Datentypen entsprechen, nicht beliebige Werte annehmen können, sondern in einer Liste aus vorgegebenen Werten enthalten sein müssen. Betrachten wir als Beispiel das Element `Objektart`. Dieses soll ausdrücken, ob es sich bei der vorliegenden abwassertechnischen Anlage um ein Kanten- oder Knotenobjekt handelt. Es sind daher lediglich zwei konkrete Werte möglich, die `Objektart` annehmen kann. Stellvertretend für die Werte `Kante` und `Knoten` werden hier `integer` verwendet. Die Einschränkungen bestehen darin, dass der `integer` aus maximal einer Ziffer bestehen darf, der Minimalwert beträgt „1“, der Maximalwert „2“. Als `enumeration` werden nun die beiden konkreten Werte aufgelistet, die der `simpleType ObjektartType` annehmen darf. Als Kommentar wird zudem angegeben, welcher der Werte für Kante oder Knoten steht. In der Instanz taucht allerdings nur der `integer` auf. Daher muss der Benutzer, beziehungsweise die Softwareanwendung, wissen, welche Angabe sich hinter dem Zahlenwert verbirgt. Neben dem `integer` werden im ISYBAU-Schema die `simpleTypes` der Referenzlisten auch von `strings` abgeleitet. Dabei wird lediglich die maximale Anzahl von Zeichen beschränkt.

1707-stammdaten.xsd

```
<xsd:element name="Objektart" type="ObjektartType"/>
```

1707-referenzlisten.xsd

```
<xsd:simpleType name="ObjektartType">
	<xsd:annotation>
		<xsd:documentation>G100 Objektdifferenzierung</xsd:documentation>
	</xsd:annotation>
	<xsd:restriction base="xsd:integer">
		<xsd:totalDigits value="1"/>
		<xsd:minInclusive value="1"/>
		<xsd:maxInclusive value="2"/>
		<xsd:enumeration value="1">
			<xsd:annotation>
				<xsd:documentation>Kante</xsd:documentation>
			</xsd:annotation>
		</xsd:enumeration>
		<xsd:enumeration value="2">
			<xsd:annotation>
				<xsd:documentation>Knoten</xsd:documentation>
			</xsd:annotation>
		</xsd:enumeration>
	</xsd:restriction>
</xsd:simpleType>
```

ISYBAU_XML-2017-Stammdaten.xml

```
<AbwassertechnischeAnlage>
	[. . .]
	<Objektart>2</Objektart>
	[. . .]
</AbwassertechnischeAnlage>
```

Tabelle 4-11: Beispiel zum Bezug eines Elements zu einer enumeration einer Referenzliste

Als letztes soll an dieser Stelle auf die Vergabe von Schlüsseln im ISYBAU-Schema eingegangen werden. Ein Schlüssel legt fest, dass ein Element eindeutig sein muss und dabei nicht null sein darf. Als Beispiel eignen sich die folgenden Definitionen zu den Datenkollektiven.

1707-metadaten.xsd

```
<xsd:element name="Datenkollektive" type="DatenkollektiveType">
	<xsd:key name="MKollektivKey">
		<xsd:selector xpath="isy:Kennungen/isy:Kollektiv"/>
		<xsd:field xpath="isy:Kennung"/>
	</xsd:key>
	<xsd:unique name="SKollektivKey">
		<xsd:selector xpath="isy:Stammdatenkollektiv"/>
		<xsd:field xpath="isy:Kennung"/>
	</xsd:unique>
	<xsd:keyref name="SKollektivRef" refer="MKollektivKey">
		<xsd:selector xpath="isy:Stammdatenkollektiv"/>
		<xsd:field xpath="isy:Kennung"/>
	</xsd:keyref>
[. . .]
</xsd:element>
```

ISYBAU_XML-2017-Stammdaten.xml

```
<Datenkollektive>
  <Datenstatus>1</Datenstatus>
  <Erstellungsdatum>2019-11-11</Erstellungsdatum>
  <Kommentar>ISYBAU XML-2017 Beispieldatensatz</Kommentar>
  <Kennungen>
    <Kollektiv>
      <Kennung>STA01</Kennung>
      <Kollektivart>1</Kollektivart>
      <Kollektiveigenschaft>
        <Stammdaten>
          <Stammdatentyp>1</Stammdatentyp>
          <Bautechnik>1</Bautechnik>
          <Geometrie>1</Geometrie>
          <Sanierung>0</Sanierung>
          <Umfeld>0</Umfeld>
        </Stammdaten>
      </Kollektiveigenschaft>
      <Regelwerk>6</Regelwerk>
      <Bearbeitungsstand>2019-04-01</Bearbeitungsstand>
      <Kommentar>ISYBAU XML-2017 Stammdaten ohne Sanierungsdaten</Kommentar>
    </Kollektiv>
  </Kennungen>
  <Stammdatenkollektiv>
    <Kennung>STA01</Kennung>
[. . .]
  </Stammdatenkollektiv>
</Datenkollektive>
```

Tabelle 4-12: Definition von Schlüsseln am Beispiel des Elementes `Datenkollektive`

Die Datenkollektive sind wie beschrieben ein Unterelement der Metadaten. Darüber hinaus wird in den Metadaten unter dem Pfad `Identifikation-Datenkollektive-Kennungen-Kollektiv` jedes Datenkollektiv vorab erfasst. Dabei wird jedem Datenkollektiv mit dem Element `Kennung` ein eindeutiger Schlüssel vergeben. Aufgerufen wird diese Zuweisung im Schema über `<xsd:key.` Anschließend erhält der Schlüssel einen Namen, in diesem Fall `"MKollektivKey".` Über `<xsd:selector xpath` wird durch das Schema navigiert, damit mit `<xsd:field xpath` das Feld, welches als Schlüsselelement definiert werden soll, ausgewählt werden kann. Anzumerken ist, dass das Schlüsselelement selbst nicht an dieser Stelle definiert wird. Im Beispiel erfolgt die Definition des Elementes `Kennung` als Subelement des `DatenkollektivTypes.` Schlüssel bezieht sich lediglich auf bereits definierte Elemente.

Dadurch, dass die Datenkollektive sowohl in den Metadaten als auch in den jeweiligen Datenkollektiven definiert werden, muss gewährleistet werden, dass sich über das Element `Kennung` ein Bezug zwischen Metadaten und Datenkollektiv herstellen lässt. Konkret gesagt, muss die `Kennung STA01` des Stammdatenkollektives in der Musterdatei ISYBAU_XML-2017-Stammdaten.xml auch im Bereich der Metadaten vorhanden sein. Damit diese Bedingung erfüllt wird, wird im Schema eine `<xsd:keyref` festgelegt. Diese erhält ebenfalls einen Namen und über `refer` wird angegeben, welcher Schlüssel referenziert werden soll. Im Beispiel bezieht sich der `keyref "SKollektivRef"` auf das definierte Schlüsselelement `"MKollektivKey".` Analog zum Schlüssel wird das Element über `selector` und `field` der `Kennung` im Bereich `Stammdatenkollektiv` zugewiesen.

Darüber hinaus muss gewährleistet werden, dass die Kennung des Stammdatenkollektivs ebenfalls eindeutig ist. Mit den getroffenen Definitionen zum Schlüssel und Bezug auf den Schlüssel ist noch nicht ausgeschlossen, dass mehrere Stammdatenkollektive die Kennung `STA01` tragen. Daher wird das Feld `Kennung` im `Stammdatenkollektiv` als `unique` definiert. Der Syntax gleicht dabei dem zur Definition eines Schlüssels und auch die Eigenschaften sind ähnlich. Ein Element, das als `unique` definiert ist, muss eindeutig sein, kann allerdings nicht wie der Schlüssel referenziert werden. (<edition W3>, 2020)

4.3 Umsetzung des Datenimports

Als erster Schritt zur Umsetzung der Konvertierung von ISYBAU-XML-Daten nach IFC müssen die Daten eingelesen werden. Hierfür wurde Microsoft Visual Studio als Programmierumgebung und C# als Programmiersprache gewählt. Ein Hauptgrund für die Verwendung von C# in dieser Arbeit ist die Möglichkeit zur Nutzung von bereits entwickelten C# Strukturen zur Umsetzung des Exports nach IFC.

Zum Datenimport wurde die XmlSerializer Klasse herangezogen. Diese ermöglicht in C# das Erstellen und Lesen von XML-Dateien auf Grundlage von im Projekt vorhanden Klassen. Die Klassen in C# erfüllen eine ähnliche Funktion wie die `ComplexTypes` im XSD-Schema. Über sie kann der Aufbau eines Objektes allgemein vordefiniert werden. Eine Klasse besteht dabei grundlegend aus Eigenschaften und Methoden. Die Eigenschaften sind vergleichbar mit den Elementen eines `ComplexTypes`. Methoden sind

Funktionen, die ausgeführt werden, wenn sie von einem Objekt einer Klasse aufgerufen werden. Um den Aufbau und die Funktion von Klassen zu veranschaulichen ist im Folgenden die Kasse „`Identifikation`“ im Vergleich zum `ComplexType Identifikation` dargestellt.

1707-metadaten.xsd

```
<xsd:element name="Identifikation">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element name="Version" type="xsd:gYearMonth"/>
			<xsd:element name="Admindaten" type="AdmindatenType">
			[. . .]
			<xsd:element name="Datenkollektive" type="DatenkollektiveType">
			[. . .]
			</xsd:element>
		</xsd:sequence>
	</xsd:complexType>
```

Klasse Identifikation

```
public class Identifikation
{
	public string Version;
	public AdmindatenType Admindaten;
	public DatenkollektiveType Datenkollektive;
}
```

Tabelle 4-13: Exemplarischer Aufbau einer Klasse in C# nach dem Vorbild des `complexTypes Identifikation` aus dem ISYBAU-Schema

Die Klasse „`Identifikation`“ wird über den Befehl `public class Identifikation` erstellt. Der Begriff `public` definiert dabei, dass die Klasse öffentlich ist und von anderen Klassen darauf zugegriffen werden kann. Den Eigenschaften `Version`, `Admindaten` und `Datenkollektive` ist die jeweilige Klasse vorangestellt. Die Klassen `AdmindatenType` und `DatenkollektiveType` müssen zudem an einer anderen Stelle definiert sein, damit das Programm die Eigenschaften diesen Klassen zuordnen kann. Für den `simpleType gYearMonth` gibt es in diesem Fall keine vorgefertigte Klasse, die dem Format entsprechen würde. Daher wird die Eigenschaft `Version` hier als Element der Standardklasse `string` festgelegt.
Auf Grundlage solcher Klassen kann mithilfe einer Methode der XmlSerializer Klasse eine XML-Datei eingelesen werden. Dafür müssen jedoch zunächst alle Klassen entsprechend dem XSD-Schema erstellt werden. Hierzu bietet sich das XML Schema Definition-Tool (Xsd.exe) an. Dieses generiert aus einem XSD-Schema selbstständig die entsprechenden Klassen. (Microsoft Corporation, 2020) Als Eingangsdatei dient die 1707-metadaten.xsd-Datei. Da über den `include`-Befehl im Schema die XSD-Dateien für die Datenkollektive eingebunden sind, werden sowohl Klassen für die Metadaten als auch für alle restlichen enthaltenen Datentypen erstellt. Diese werden in eine .cs-Datei geschrieben, die in Visual Studio in ein Projekt eingebunden werden kann. Im konkreten Beispiel sieht die automatisch generierte Klasse `Identifikation` wie folgt aus:

Aus dem XSD-Schema automatisch generierte Klasse Identifikation

```
/// <remarks/>
[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.8.3928.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlTypeAttribute(AnonymousType = true, Namespace =
"http://www.bfr-abwasser.de")]
[System.Xml.Serialization.XmlRootAttribute(Namespace = "http://www.bfr-abwasser.de",
IsNullable = false)]

public partial class Identifikation
{

    private string versionField;

    private AdmindatenType admindatenField;

    private DatenkollektiveType datenkollektiveField;

    /// <remarks/>
    [System.Xml.Serialization.XmlElementAttribute(DataType = "gYearMonth")]
    public string Version
    {
        get
        {
            return this.versionField;
        }
        set
        {
            this.versionField = value;
        }
    }

    /// <remarks/>
    public AdmindatenType Admindaten
    {
        get
        {
            return this.admindatenField;
        }
        set
        {
            this.admindatenField = value;
        }
    }

    /// <remarks/>
    public DatenkollektiveType Datenkollektive
    {
        get
        {
            return this.datenkollektiveField;
        }
```

```
        set
        {
            this.datenkollektiveField = value;
        }
    }
}
```

Tabelle 4-14: Auszug aus den durch die Xsd.exe aus den ISYBAU XSD-Dateien automatisch erzeugten Klassen. Der hier abgebildete Auszug enthält die definition der Klasse `Identifikation`.

Der Ausschnitt lässt erkennen, dass bei der automatischen Generierung der Klassen deren Ausführung deutlich umfangreicher ausfällt, als dies im Beispiel in Tabelle 4-13 geschehen ist. Im Folgenden soll auf einige dieser Punkte eingegangen werden, um zu verstehen, auf welche Weise die Klassen erstellt wurden und welche Informationen enthalten sind. Die erste Zeile enthält die Information, dass dieser Quellcode von einem Programm namens **„xsd“** in der Version `4.8.3928.0` erzeugt wurde und bezieht sich somit auf das XML Schema Definition-Tool. In der nächsten Zeile wird festgelegt, dass die hier dargestellte Klasse vom Serializer serialisiert werden kann. Dies ist wie im hier vorliegenden Falle wichtig, um über den XmlSerializer eine ISYBAU-XML-Datei einlesen zu können. Der Aufruf `System.Diagnostics.DebuggerStepThroughAttribute()` hat zur Folge, dass beim debuggen des Codes in Visual Studio die Klasse im Schnelldurchlauf abgefertigt wird, ohne dabei die Einzelschritte innerhalb der Klasse abzubilden. Informationen zu den Datentypen sind über `System.Xml.Serialization.XmlTypeAttribute` definiert. Am Beispiel der Klasse `Identifikation` sind die beiden Eigenschaften `AnonymousType` und `Namespace` enthalten. Der `AnonymousType` spiegelt die Definitionsart des Datentyps in der Schemadatei wider. Da es sich bei `Identifikation` um einen lokal definierten Datentyp handelt, wird `AnonymousType` auf `true` gesetzt. Bei allen global definierten Datentypen ist hierfür kein Wert vorhanden. Im Falle des ISYBAU Schemas sind alle automatisch erzeugten Klassen entsprechend der XSD-Datei dem `Namespace http://www.bfr-abwasser.de` zugewiesen.

Die `Identifikation` wird an dieser Stelle durch `System.Xml.Serialization.XmlRootAttribute` als Root-Element festgelegt. Es wird dabei zum einen erneut der Namespace zugewiesen, zum andern über `IsNullable=false` die Anzahl dieses Elements die in der Austauschdatei auftreten muss auf größer Null gesetzt.

Die Klassen sind definiert als `public partial class`. Die Festlegung als `public` ist notwendig, um auch von anderen Klassen auf diese zugreifen zu können. Eine Klasse die darüber hinaus als `partial` definiert ist, kann an einer anderen Stelle im Programm erweitert werden. In Hinblick auf die Übersichtlichkeit des Programms erweist sich dies als günstig, da zur Umsetzung der Datenkonvertierung die bestehenden Klassen mit Eigenschaften und Methoden in einer eigenen Projektdatei ergänzt werden können.

Aus den Elementen eines Datentyps im Schema wurden automatisch Eigenschaften in den Klassen generiert. Dabei wurde jeweils eine entsprechende Eigenschaft als „private“ und eine als „public“ definiert. Wie am Beispiel des Elementes `Version` im Datentyp `Identifikation` nachvollzogen werden kann, erfolgt

dabei eine einheitliche Benennung: Die `private` Eigenschaft `versionField` wird klein geschrieben und erhält den Anhang „`Field`“, wohingegen die `public` Eigenschaft den Namen aus der Schemadatei übernommen hat. Die Aufteilung auf `private` und `public` hat folgenden Effekt: Auf die `private` Eigenschaft `versionField` kann nur innerhalb der Klasse `Identifikation` zugegriffen werden. Somit kann beispielsweise eine Methode in einer anderen Klasse weder den Wert von `versionField` abrufen noch ändern. Um dies zu tun kann jedoch auf die Eigenschaft `Version` zugegriffen werden, da diese `public` ist. Sie enthält die Aufrufe `get` und `set`. Über `get` wird der Wert von `versionField` abgerufen und kann von der externen Methode genutzt werden, wohingegen durch `set` der Wert von `versionField` von außen über die Eigenschaft `Version` geändert werden kann. In der Programmierung verwendet man diese Aufteilung zur Festlegung von Zugriffsrechten, um zu vermeiden, dass ungewollt Werte abgerufen oder geändert werden können. In diesem Falle wurde die Aufteilung beim automatischen Generieren der Klassen mit angelegt.

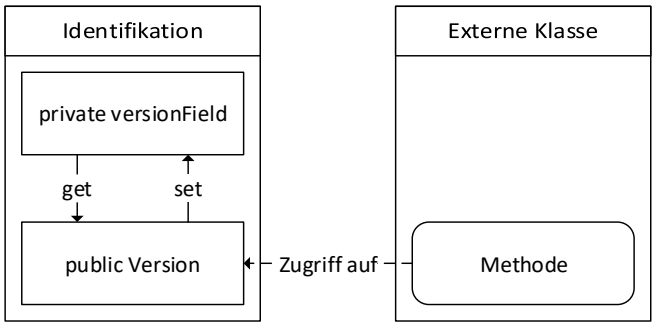


Abbildung 4-4: Zugriffsrechte von externe Klassen auf die Eigenschaften der automatisch generierten ISYBAU-Klassen. Die Methode der externen Klasse hat nur zugriff auf die `public` Eigenschaft `Version`. Über `get` und `set` interagiert diese mit `privateversionField`.

Ebenfalls automatisch erfolgt die Zuweisung der Datentypen. Diese entsprechen den jeweiligen Klassen im C#-Programm. Dabei sind, wie weiter oben bereits erwähnt, nicht alle `simpleTypes` aus dem XML-Schema als Standardklassen vorhanden. Am Beispiel der `Version` ist zu sehen, dass diese Eigenschaft nicht als `gYearMonth` übernommen werden konnten. Stattdessen wird sowohl die `private` als auch die zugehörige `public` Eigenschaft als `string` definiert. Vorgeschaltet zur `public` Eigenschaft wird jedoch über `System.Xml.Serialization.XmlElementAttribute(DataType="gYearMonth")` für den Serializer festgelegt, dass der zu erwartende Datentyp dieses Elementes `gYearMonth` entspricht (siehe Tabelle 4-14). Auf selbem Weg kann auch die Häufigkeit eines Elementes definiert werden.

Eine Variante der automatischen Generierung von Aufzählungen auf Grundlage der Referenzlisten-Schemadatei sieht man am Beispiel der Referenzliste „G100 Objektdifferenzierung“. In diesem Fall wird in C# keine Klasse für den Datentype `ObjektartType`, sondern eine Enumeration (kurz enum) erzeugt. Diese gibt feste Werte vor, die ein Objekt annehmen kann. Im Beispiel kann der Wert `Item1` oder `Item2` angenommen werden. Dies ist davon abhängig ob der Serializer entweder den Wert `"1"` oder `"2"` in der ISYBAU-Datei auffindet. Definiert wird diese Zuordnung über den Ausdruck `System.Xml.Serialization.XmlEnumAttribute`. Nicht mit übernommen werden die Anmerkungen zu den möglichen Werten, in diesem Fall „`Kante`“ und „`Knoten`“ aus der Referenzliste.

Automatsch generierte enum ObjektartType

```
public enum ObjektartType {

    /// <remarks/>
    [System.Xml.Serialization.XmlEnumAttribute("1")]
    Item1,

    /// <remarks/>
    [System.Xml.Serialization.XmlEnumAttribute("2")]
    Item2,
}
```

1707-referenzlisten.xsd

```
<xsd:simpleType name="ObjektartType">
	<xsd:annotation>
		<xsd:documentation>G100 Objektdifferenzierung</xsd:documentation>
	</xsd:annotation>
	<xsd:restriction base="xsd:integer">
		<xsd:totalDigits value="1"/>
		<xsd:minInclusive value="1"/>
		<xsd:maxInclusive value="2"/>
		<xsd:enumeration value="1">
			<xsd:annotation>
				<xsd:documentation>Kante</xsd:documentation>
			</xsd:annotation>
		</xsd:enumeration>
		<xsd:enumeration value="2">
			<xsd:annotation>
				<xsd:documentation>Knoten</xsd:documentation>
			</xsd:annotation>
		</xsd:enumeration>
	</xsd:restriction>
</xsd:simpleType>
```

Tabelle 4-15: Beispiel für eine automatisch generierte enumeration aus einer ISYBAU Referenzliste mit vorgegebenen integer Werten.

Beim Beispiel in Tabelle 4-15 ist `integer` die Basis zur Definition des `simpleTypes ObjektartType`. Daher erfolgt die automatische Umsetzung der Referenzliste wie oben beschrieben. Allerdings treten auch Referenzlisten auf, die auf einem anderen Datentyp basieren. Dabei gestaltet sich die `enum` auf eine etwas andere Weise. Wie am Beispiel der auf dem `simpleType string` basierenden Referenzliste „U104 Inspektionsrichtung“ in Tabelle 4-16 zu erkennen ist, erfolgt automatisch die Übernahme der konkreten

Werte `"O" und "U"`. Dadurch wird die Zuweisung über `XMLEnumAttribute` überfällig. Beim späteren Arbeiten mit Werten aus einer serialisierten ISYBAU-Datei ist es notwendig zu wissen, auf welche Art und Weise die automatische `enum` generiert wurde.

Automatisch generierte enum InspektionsrichtungType

```
public enum InspektionsrichtungType {

    /// <remarks/>
    O,

    /// <remarks/>
    U,
}
```

1707-referenzlisten.xsd

```
<xsd:simpleType name="InspektionsrichtungType">
	<xsd:annotation>
		<xsd:documentation>U104 Inspektionsrichtung</xsd:documentation>
	</xsd:annotation>
	<xsd:restriction base="xsd:string">
		<xsd:maxLength value="1"/>
		<xsd:enumeration value="O">
			<xsd:annotation>
				<xsd:documentation>von oben, in
Fliessrichtung</xsd:documentation>
			</xsd:annotation>
		</xsd:enumeration>
		<xsd:enumeration value="U">
			<xsd:annotation>
				<xsd:documentation>von unten, gegen
Fliessrichtung</xsd:documentation>
			</xsd:annotation>
		</xsd:enumeration>
	</xsd:restriction>

</xsd:simpleType>
```

Tabelle 4-16: Beispiel für eine automatisch generierte enumeration aus einer ISYBAU Referenzliste mit vorgegebenen string Werten.

Zu guter Letzt soll an dieser Stelle darauf eingegangen werden, wie das XML Schema Definition-Tool mit dem `<xsd:choice>`-Ausdruck umgeht. Wir erinnern uns, dass dieser dazu genutzt wird, um im Schema eine Auswahl zwischen mehreren Elementen zu ermöglichen. Beispielsweise kann eine abwassertechnische Anlage entweder ein Element des Typs `StammKanteType` oder `StammKnotenType` enthalten (siehe Tabelle 4-5). Bei der automatischen Erstellung der Klassen werden dabei in der Klasse `AbwassertechnischeAnlageType` die Eigenschaften `itemField` und `Item` angelegt. Diese sind als Eigenschaft der Klasse `object` instanziiert worden und fungieren als Platzhalter. Die Klasse `object` ermöglicht eine Zuweisung der Eigenschaften `itemField` und `Item` zu einer beliebigen anderen definierten Klasse. Zur korrekten Umsetzung erhält der `Serializer` über `XmlElementAttribute` die Anweisung das

`public object Item` entweder als `StammKanteType` oder als `StammKnotenType` zu serialisieren. Abhängig ist dies davon, ob der vorliegende Wert `"Kante"` oder `"Knoten"` annimmt.

Automatisch generierte Klasse AbwassertechnischeAnlageType

```
public partial class AbwassertechnischeAnlageType {
    [. . .]
    private object itemField;
    [. . .]
    /// <remarks/>
    [System.Xml.Serialization.XmlElementAttribute("Kante", typeof(StammKanteType))]
    [System.Xml.Serialization.XmlElementAttribute("Knoten", typeof(StammKnotenType))]
    public object Item {
        get {
            return this.itemField;
        }
        set {
            this.itemField = value;
        }
    }
    [. . .]
}
```

1707-stammdaten.xsd

```
<xsd:complexType name="AbwassertechnischeAnlageType">
      <xsd:sequence>
            [. . .]
            <xsd:choice>
                  <xsd:element name="Kante" type="StammKanteType" minOccurs="0"/>
                  <xsd:element name="Knoten" type="StammKnotenType" minOccurs="0"/>
            </xsd:choice>
            [. . .]
      </xsd:sequence>
</xsd:complexType>
```

Tabelle 4-17: Beispiel für eine automatisch generierte Klasse mit einem choice Element im ISYBAU Schema.

Die so automatisch generierten Klassen stehen dem Programm zu Deserialisierung zur Verfügung. Bei der Deserialisierung werden Objekte aus einer XML-Datei heraus neu erstellt. Grundlage für diese Objekte sind die vorab generierten Klassen aus dem XML-Schema. Somit werden die Informationen aus der XML-Datei gelesen und stehen zum Abruf oder zur Bearbeitung zur Verfügung. Das Gegenstück zur Deserialisierung stellt die Serialisierung dar, bei der Objekte in eine XML-Datei geschrieben werden. (Microsoft Corporation, 2020). Ausgeführt wird in diesem Fall die Deserialisierung der ISYBAU-XML-Datei durch die im Namespace `System.Xml.Serialization` vorhandene `XmlSerializer`-Klasse. Bei der Deserialisierung der ISYBAU-XML-Datei wird ein Objekt der Klasse `Identifikation` erzeugt. Somit lässt sich über die Eigenschaften dieses root Objektes auf die Baumstruktur des ISYBAU-Schemas zugreifen.

Im Beispiel ist zu erkennen, wie vom Objekt mit der Bezeichnung `Austauschdatei` über die Eigenschaften der Klasse `Identifikation` und deren Subklassen auf die Objektbezeichnung der ersten abwassertechnischen Anlage im ersten Stammdatenkollektiv zugegriffen werden kann:

```
Austauschdatei.Datenkollektive.Stammdatenkollektiv[0].AbwassertechnischeAnlage[0].Objektbezeichnung
```

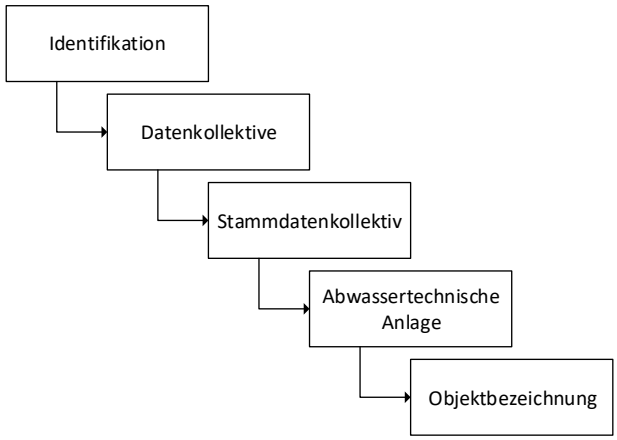


Abbildung 4-5: Navigation durch die ISYBAU Baumstruktur anhand des Identifikation Objektes „`Austauschdatei`“

Es ist anzumerken, dass eine Deserialisierung von doppelten Arrays in der Form `Array[][]` von der `XmlSerializer`-Klasse nicht unterstütz wird. Ein solches Array liegt in der Klasse `HydSpeicherbauwerkType` als Eigenschaft `Drosselkennlinie[][]` vor. Diese wird definiert im Hydraulikdatenkollektiv. Da in dieser Arbeit nur mit den Stammdaten gearbeitet wird, nimmt diese Klasse keinen Einfluss auf den Konvertierungsansatz nach IFC und ist manuell abgeändert worden, um die Deserialisierung durchführen zu können.

5 Konvertierung in das IFC-Austauschformat

Das Ziel dieser Arbeit ist es, einen Ansatz zur Konvertierung von ISYBAU-Daten nach IFC zu entwickeln. Auf die Merkmale und den Datenimport des ISYBAU-Schemas wurde in den vorherigen Kapiteln genauer eingegangen. In diesem Abschnitt soll das Zielformat IFC analysiert werden.

5.1 Grundlagen zum IFC-Austauschformat

Das IFC (Industry Foundation Classes) Format ist ein standardisiertes Datenmodell zum Austausch von Bauwerksmodellen. Anwendung findet IFC vor allem in Zusammenhang mit Building Information Modeling (BIM). Dabei handelt es sich um eine modellorientierte Arbeitsweise im Bauwesen, bei der das digitale Modell des zu erstellenden Bauwerkes eine zentrale Rolle spielt. Dieses Modell wird von der Planung, über den Bau bis zum Betreiben und dem Rückbau der baulichen Anlage erweitert und hat das Ziel, dem Verlust von Informationen vorzubeugen. Über IFC lassen sich diese Modellinformationen zwischen unterschiedlichen Projektbeteiligten und Softwareanwendungen austauschen. Die Folgenden Beschreibungen beziehen sich auf die IFC-Version 4.1 vom Juni 2018

5.1.1 Aufbau des IFC-Schemas

Grundlegend stehen im IFC-Schema Klassen zur Verfügung, welche als Entities bezeichnet werden. Diese können zur Beschreibung des Bauwerksmodells herangezogen werden. Neben den Entities werden Datentypen, sogenannte „Types“ definiert. In Analogie zu den in Kapitel 4.2 erläuterten Bestandteilen des ISYBAU-XML-Schemas, können die Entities mit `complexTypes` und die Types mit den `simpleTypes` verglichen werden. Entities enthalten Attribute, welche konkrete Werte der Types annehmen können. Die Types können dabei aber auch aus Enumeration oder Choice Elementen bestehen.

Darüber hinaus enthält das IFC-Schema Property Sets, Quantity Sets, Functions und Rules. Ziel der BIM-Methode, und somit auch des IFC-Austauschformates, ist es die Bauteilgeometrien mit zusätzlichen semantischen Informationen zu verknüpfen und somit Objekte ganzheitlich zu beschreiben. Die geometrische Beschreibung wird dabei über Entities und Types abgedeckt. Zusätzliche Eigenschaften können über die Property Sets und Quantity Sets zugeordnet werden. Dabei handelt es sich um im IFC-Schema vorgefertigte Gruppen von Eigenschaften, die einer Entity zugewiesen werden können. Am Beispiel der Entity `IfcWall` wäre dies das Property Set `Pset_WallCommon` (Abbildung 5-1). Dieses enthält zum Beispiel die Eigenschaft `FireRating`, mit der der Feuerwiederstand einer Wand beschrieben werden kann. Das IFC-Schema stellt eine Vielzahl dieser vorgefertigten Property Sets zur Verfügung. Jedoch ist es darüber hinaus möglich eigene Property Sets zu definieren. Hierfür steht die Entity `IfcPropertySet` zur Verfügung. In dieser lassen sich über das Attribut `HasProperties` benutzerdefinierte Properties bündeln.

Ähnlich verhält es sich mit den Quantity Sets. Diese beschreiben messbare Eigenschaften eines Objektes wie Länge, Fläche oder Volumen. Auch hier besteht die Möglichkeit neben den vordefinierten Quantity Sets über die Entity `IfcQuantitySet` eigene Quantity Sets zu erzeugen. Ein weiterer Bestandteil des IFC-Schemas sind die Functions. Diese sind Operatoren, die unter anderem Überprüfungen durchführen. Beispielsweise überprüft die Function `IfcUniquePropertyName`, ob die Namen von definierten Properties eindeutig sind. (buildingSMART International Limited, 2020) Im IFC-Schema Stand Version 4.1 sind lediglich zwei Rules als eigenständige Elemente definiert: `IfcSingleProjectInstance` legt fest, dass maximal eine Instanz der Entity `IfcProject` in einer Austauschdatei vorhanden sein darf, (buildingSMART International Limited, 2020) während die Rule `IfcRepresentationContextSameWCS` festlegt, dass bei mehreren Instanzen von `IfcGeometricRepresentationContext` alle auf demselben Koordinatensystem beruhen müssen. (buildingSMART International Limited, 2020)

PsetName	Properties

Pset_WallCommon

Template	PropertyName	Value	Reference
Single Value	Reference	IfcIdentifier	
Single Value	AcousticRating	IfcLabel	
Single Value	FireRating	IfcLabel	
Single Value	Combustible	IfcBoolean	
Single Value	SurfaceSpreadOfFlame	IfcLabel	
Single Value	ThermalTransmittance	IfcThermalTransmittanceMeasure	
Single Value	IsExternal	IfcBoolean	
Single Value	ExtendToStructure	IfcBoolean	
Single Value	LoadBearing	IfcBoolean	
Single Value	Compartmentation	IfcBoolean	
Enumerated Value	Status	IfcLabel	

Abbildung 5-1 Property Set `Pset_WallCommon` und die darin enthaltenen Eigenschaften (buildingSMART International Limited, 2020)

Definiert wird das IFC-Schema standartgemäß in der Datenmodellierungssprache EXPRESS. Vergleichbar ist dies mit der Definition des ISYBAU-XML-Schemas über XSD. Tatsächlich existiert auch eine XSD-Definition des IFC-Schemas, da ein Austausch als XML-Datei möglich ist. Auf die möglichen Austauschdatenformate der IFC-Daten wird im Kapitel 5.1.3 genauer eingegangen. Da es für die Beschreibung des IFC-Schemas besser geeignet ist, wird in dieser Arbeit in Beispielen auf das EXPRESS Schema zurückgegriffen. In der folgenden Tabelle ist jedoch zum Vergleich ein Auszug aus dem XSD- und EXPRESS-Schema abgebildet. Definiert wird in beiden Fällen die Entity `IfcWall`. Im oberen Bereich befindet sich die Definition über XSD und im unteren als EXPRESS. Im IFC-Schema spielt die Vererbung von Attributen eine wichtige Rolle. Dabei erbt eine Entity, die ein Subtype einer anderen Entity ist, die Attribute der übergeordneten Entity. Im Beispiel ist `IfcWall` ein Subtype von `IfcBuildingElement`. Im EXPRESS-Schema wird dies deutlich durch den Ausdruck `SUBTYPE OF (IfcBuildingElement)` definiert, während im XSD-Schema dies indirekt dadurch definiert wird, dass der `complexType IfcWall` eine

`extension` zur `base` `IfcBuildingElement` ist. Zur Konsequenz hat beides, dass `IfcWall` die Attribute von `IfcBuildingElement` erbt. Das EXPRESS-Schema führt darüber hinaus im Bereich `SUPERTYPE OF` auf, dass `IfcWall` selbst der übergeordnete Supertype für die Entities `IfcWallElementCase` und `IfcWallStandardCase` ist. Unter dem Bereich `SUBTYPE OF` werden in EXPRESS die eigenen Attribute von `IfcWall` definiert. In diesem Fall ist es das Attribut `PredefinedType`. Dieses ist optional und vom Typ `IfcWallTypeEnum`. Im Bereich `WHERE` sind Regeln aufgestellt, die erfüllt sein müssen, damit eine Instanz der Entity `IfcWall` korrekt instanziiert ist.

IFC4x1.xsd

```
<xs:element name="IfcWall" type="ifc:IfcWall" substitutionGroup="ifc:IfcBuildingElement"
nillable="true"/>
<xs:complexType name="IfcWall">
  <xs:complexContent>
    <xs:extension base="ifc:IfcBuildingElement">
      <xs:attribute name="PredefinedType" type="ifc:IfcWallTypeEnum" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

IFC4x1.exp

```
ENTITY IfcWall
 SUPERTYPE OF (ONEOF
    (IfcWallElementedCase
    ,IfcWallStandardCase))
 SUBTYPE OF (IfcBuildingElement);
       PredefinedType : OPTIONAL IfcWallTypeEnum;
 WHERE
       CorrectPredefinedType : NOT(EXISTS(PredefinedType)) OR
 (PredefinedType <> IfcWallTypeEnum.USERDEFINED) OR
 ((PredefinedType = IfcWallTypeEnum.USERDEFINED) AND EXISTS (SELF\IfcObject.ObjectType));
       CorrectTypeAssigned : (SIZEOF(IsTypedBy) = 0) OR
  ('IFC4X1.IFCWALLTYPE' IN TYPEOF(SELF\IfcObject.IsTypedBy[1].RelatingType));
END_ENTITY;
```

Tabelle 5-1 Vergleich der Definition einer IFC-Entity im XSD und EXPRESS-Schema

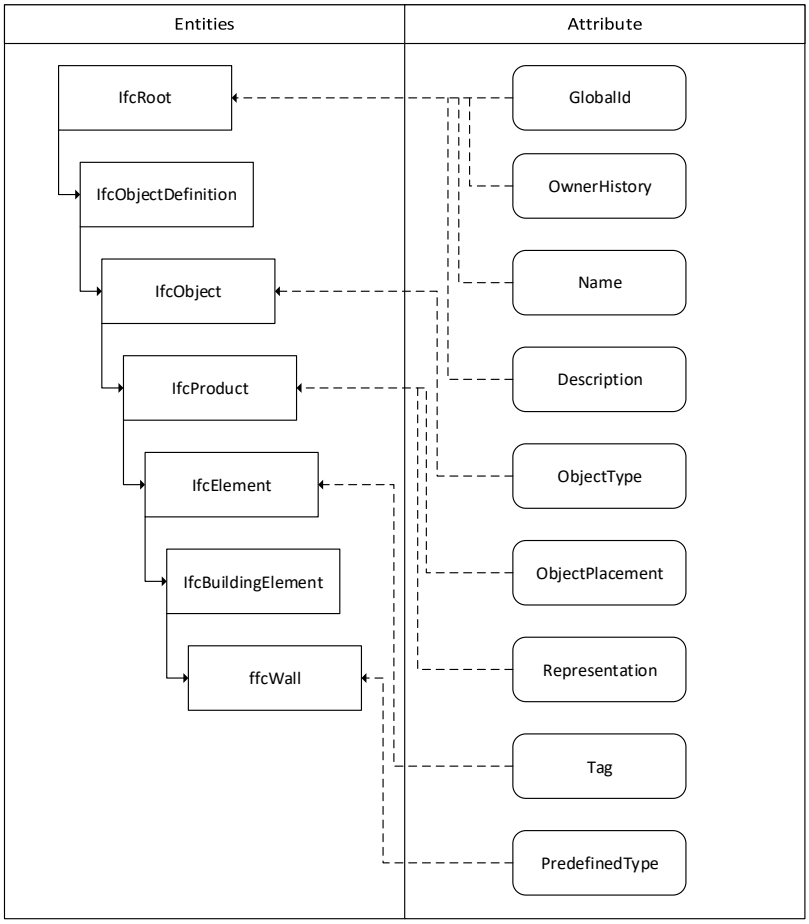


Abbildung 5-2 Vererbungshistorie der Entity `IfcWall`

Die Elemente Entities, Types, Property Sets, Quantity Sets, Functions und Rules des IFC-Schemas werden gegliedert in unterschiedliche Schichten (engl. *Layers*). (Borrmann , et al., 2015, p. 88) Dabei gilt, dass „Elemente der weiter oben liegenden Schichten auf Elemente der weiter unten liegenden Schichten verweisen dürfen, aber nicht umgekehrt“ (Borrmann , et al., 2015, p. 88). Es existieren die vier Layer *Core Layer*, *Shared Element Layer* oder auch *Interoperability Layer*, *Domain Layer* und *Resource Layer* (Borrmann , et al., 2015, p. 88).

Der Core Layer bildet die Grundstruktur des IFC-Schemas. Ausgangspunkt für alle weiteren Entities in *Shared Elements/Interoperability Layer* und *Domain Layer* ist die Entity `IfcRoot` (buildingSMART International Limited, 2020).

Im *Shared Element/Interoperability Layer* werden Subklassen der Entities des *Core Layers*, sowie weitere Elemente wie Types oder Property Sets definiert.

Der Domain Layer enthält wiederum Subklassen der Entities des *Shared Element/Interoperability Layers*. Das bedeutet, dass alle Entities dieser drei Layer Subklassen der `IfcRoot` Entity sind und deren Attribute *GlobalId, OwnerHistory, Name* und *Description* besitzen, wobei die letzten drei Attribute optional sind.

Bei den Entities des *Resource Layers* ist dies nicht der Fall. Diese leiten sich nicht von `IfcRoot` ab und besitzen daher keine eigene `GlobalId`. Als Folge dessen können diese Entities „nicht als eigenständige Objekte in einem IFC-Modell existieren, sondern müssen von einem Objekt referenziert werden, das eine Subklasse von `IfcRoot` instanziiert“ (Borrmann , et al., 2015, p. 90)

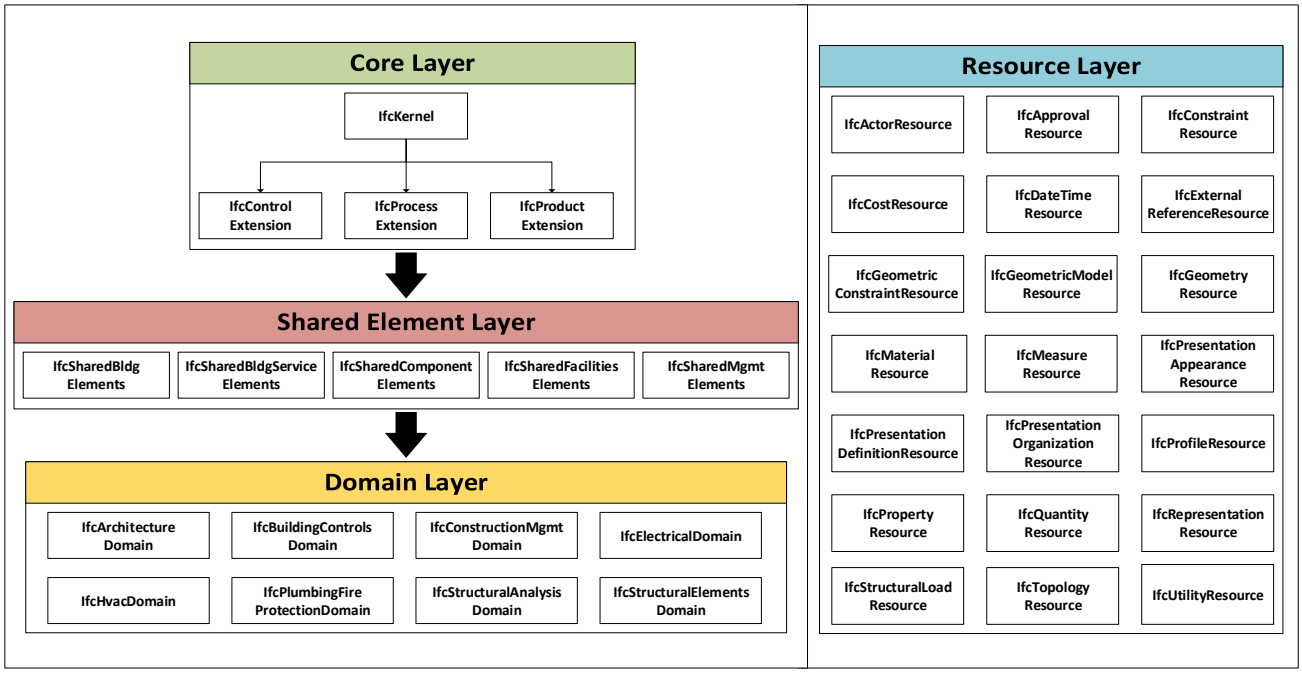


Abbildung 5-3: Layer Aufteilung des IFC-Schemas

Anhand eines simplen Beispiels soll dies veranschaulicht werden: Die Entity `IfcWall` wird definiert im *Shared Element/Interoperability Layer* und ist damit eine Subklasse von `IfcRoot`. Im IFC-Schema wird über `IfcWall` eine Wand definiert, deren geometrische Repräsentation über das Attribut `Representation` definiert wird. Der erwartete Type für dieses Attribut ist `IfcProductRepresentation`, welcher eine Entity des *Resource Layers* darstellt.

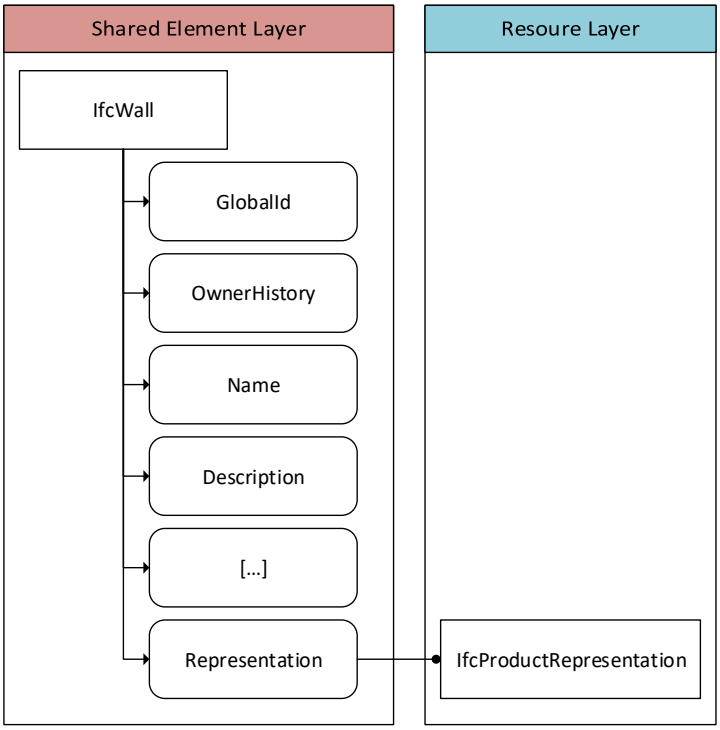


Abbildung 5-4 Referenzieren der Entity `IfcProductRepresentation` durch das Attribut `Representation` der Entity `IfcWall`

5.1.2 Beziehungen

Das IFC-Schema bietet die Möglichkeit, Objekte miteinander in Bezug zu setzen. Dies geschieht jedoch nicht durch eine direkte Beziehung, sondern über ein zwischengeschaltetes Beziehungsobjekt. Diese „Beziehungsobjekte sind immer Instanzen einer Subklasse von `IfcRelationship`“ (Borrmann , et al., 2015, p. 93)

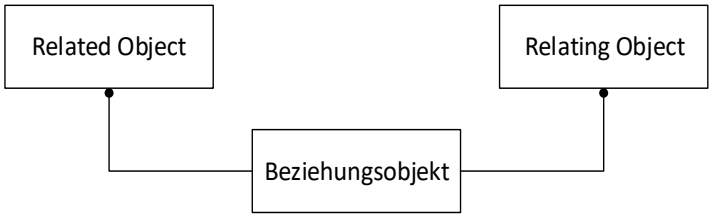


Abbildung 5-5 Schematische Darstellung einer Beziehung zwischen zwei Objekten über ein Beziehungsobjekt in IFC

Das *Related Object* ist dabei das Objekt, auf das sich das *Relating Object* bezieht. Abhängig vom Beziehungsobjekt kann es dabei mehrere oder bloß ein *Related Object* oder *Relating Object* geben. Das Zuweisen eines Materials zu einem Element soll die Beziehungen in IFC verdeutlichen. Das benötigte

Beziehungsobjekt ist vom Typ `IfcRelAssociatesMaterial`. Die Entity `IfcRelAssociatesMaterial` enthält die beiden Attribute *RelatedObjects* und *RelatingMaterial*. Das *Related Object* kann eine Instanz eines Subtypes von `IfcPropertyDefinition` oder `IfcObjectDefinition` sein. In diesem Beispiel ist eine Wand vom Typ `IfcWall` gewählt. Das Attribut *RelatedObjects* der Entity `IfcRelAssociatesMaterial` lässt mehrere *Related Objects* zu, jedoch nur ein *RelatingMaterial* vom Typ *IfcMaterialDefinition*, *IfcMaterialUsageDefinition* oder *IfcMaterialList*. Auf diese Weise kann einer Wand eine Materialeigenschaft zugewiesen werden.

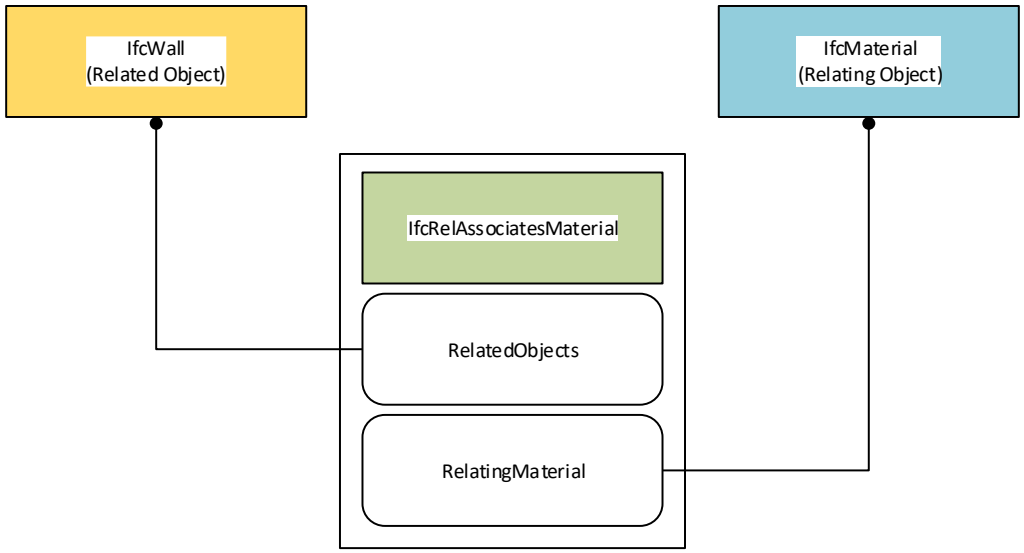


Abbildung 5-6 Zuweisen einer Materialeigenschaft zu einer Wand

5.1.3 STEP Physical File

Es stehen unterschiedliche Datenformat für eine konkrete IFC-Austauschdatei zur Verfügung. Zum einen das aus ISYBAU bekannte XML-Format. Am verbreitetsten ist jedoch der Austausch von IFC-Daten über das STEP Physical File (SPF). Daneben existieren weitere Möglichkeiten, wie zum Beispiel ein datenbankorientierter Austausch über SQLite. SPF bietet im Vergleich zu den anderen Formaten die geringste Dateigröße (buildingSMART International Limited, 2020). Deshalb, und aufgrund der weiten Verbreitung, wird in dieser Arbeit auf Beispiele anhand eines Auszuges aus der entsprechenden SPF-Datei eingegangen. Der Aufbau eines SPF-Formates soll nun anhand des Beispiels wall-with-opening-and-window.ifc[3] erläutert werden.

[3] Link zur Beispieldatei: https://standards.buildingsmart.org/IFC/RELEASE/IFC4_1/FINAL/HTML/link/wall-with-opening-and-window.htm Zugriff am 07.10.2020

wall-with-opening-and-window.ifc

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((''),'2;1');
FILE_NAME('','2019-03-20T15:56:46',(''),(''),'BuildingSmart IfcKit by
Constructivity','IfcDoc 12.0.0.0','');
FILE_SCHEMA(('IFC4x1'));
ENDSEC;


DATA;

#2= IFCOWNERHISTORY(#7,#8,$,.NOTDEFINED.,$,$,$,1323724715);

#28= IFCPRODUCTDEFINITIONSHAPE($,$,(#11,#12))

#38= IFCWALLSTANDARDCASE('3ZYW59sxj8lei475l7EhLU',#2,'Wall for Test Example','Description
of Wall',$,#70,#28,$,$);

#70= IFCLOCALPLACEMENT(#75,#76);
#71= IFCRELASSOCIATESMATERIAL('3DQ2_rihzBm8nLF98euhbs',#2,$,$,(#38),#96);


#96= IFCMATERIALLAYERSETUSAGE(#97,.AXIS2.,.POSITIVE.,-150.,$);
#97= IFCMATERIALLAYERSET((#98),$,$);
#98= IFCMATERIALLAYER(#99,300.,$,$,$,$,$);
#99= IFCMATERIAL('Name of the material used for the wall',$,$);
ENDSEC;



END-ISO-10303-21;
```

Tabelle 5-2: Auszug aus dem offiziellen Beispiel wall-with-opening-and-window.fic von buildingSmart.

In der ersten Zeile wird auf den ISO-Standard `ISO-10303-21` verwiesen, auf dem das SPF Format basiert. (Borrmann , et al., 2015, p. 117) Darauf folgt der Header, der allgemeine Informationen wie Beschreibung, Name, oder Autor enthalten kann. Außerdem im Header enthalten ist der Hinweis auf die verwendete Version des IFC-Schemas, in diesem Fall `IFC4x1`. `ENDSEC;` beendet den Header, woraufhin der eigentliche Datenbereich beginnt.

Im Datenbereich eines STEP Physical Files erhält jedes Element eine Nummer, dem ein `#`-Zeichen vorangestellt ist. Diese Nummern müssen nicht der tatsächlichen Zeilennummer in der Austauschdatei entsprechen und können auch durcheinander stehen. Wichtig ist jedoch, dass zur eindeutigen Identifizierung jedes Elements eine Nummer nur einmal vergeben sein darf.

Zentrales Element im oben dargestellten Auszug ist `IFCWALLSTANDARDCASE (#38)`. Die Entity `IFCWALLSTANDARDCASE` ist ein Subtype von `IfcWall` und besitzt folgende Attribute:

Nr.	Attribut	Type
1	GlobalId	IfcGloballyUniqueId
2	OwnerHistory	IfcOwnerHistory
3	Name	IfcLabel
4	Description	IfcText
5	ObjectType	IfcLabel
6	ObjectPlacement	IfcObjectPlacement
7	Representation	IfcProductRepresentation
8	Tag	IfcIdentifier
9	PredefinedType	IfcWallTypeEnum

Tabelle 5-3: Attribute von `IfcWallStandardCase`

In der Austauschdatei werden diese Attribute im eingeklammerten Bereich nach `IFCWALLSTANDARDCASE` instanziiert.

1	2	3	4	5	6	7	8	9
'3ZYW59sxj8lei475l7EhLU'	#2	'Wall for Test Example'	'Description of Wall'	$	#70	#28	$	$

Tabelle 5-4: Aufgliederung der konkreten Attributwerte einer Instanz von `IfcWallStandardCase` aus #38 des Beispiels in

Das erste Attribut *GlobalId* besteht aus einer 22-stelligen Zeichenfolge und bildet in der Austauschdatei eine eindeutige Identifizierung des abgebildeten Elements. Dieses Attribut ist ein Pflichtfeld für alle Daten, deren Entity die Subklassen von `IfcRoot` ist. Das zweite Attribut *OwnerHistory* verweist auf `#2` in der Austauschdatei. `#2` ist eine Instanz von `IfcOwnerHistory`, die wiederum in Ihren Attributen auf `#7` und `#8` verweist. Aus Gründen der Übersichtlichkeit sind diese im oben abgebildeten Auszug aus der Austauschdatei nicht mehr mit abgebildet. Die Attribute *Name* und *Description* sind vom Typ `IfcText` beziehungsweise `IfcLabel`. Diese geben in der Austauschdatei eine Zeichenfolge wieder, die in der Austauschdatei in Anführungszeichen geschrieben wird. Im Beispiel ist der *Name* mit 'Wall for Test Example' und die *Description* als allgemeiner Ausdruck 'Description of Wall' festgelegt. Ein „`$`"-Zeichen wie bei den Attributen 5,8 und 9 bedeutet, dass zu diesen Attributen keine Werte vergeben wurden. Dies ist nur bei optionalen Attributen möglich. Die Attribute Nummer 6 *ObjectPlacement* und 7 *Representation* verweisen auf die entsprechenden Elemente `#70` und `#28`.

Das Element #71 ist ein Beziehungsobjekt, das zur Zuweisung von Materialeigenschaften, wie im vorangegangenen Kapitel beschrieben, verwendet wird. Das Related Object ist in diesem Fall das Wandobjekt `#38`. Das Relating Material wird definiert in `#96`. Es ist möglich, mehreren Objekten die Materialeigenschaften zuzuweisen. Dazu werden mehrere Related Objects in runden Klammern aufgelistet.

Der Datenbereich endet ebenfalls mit dem Ausdruck `ENDSEC;`. `END-ISO-10303-21;` markiert das Ende der gesamten Austauschdatei

5.1.4 Darstellung von Geometrien

Kernelement der Arbeitsweise BIM ist das Bauwerksmodell, welches Geometrie und Sachinformationen miteinander vereint. In diesem Kapitel werden die unterschiedlichen geometrischen Darstellungsmöglichkeiten analysiert, die IFC zum Austausch des Modells bereitstellt. Definiert werden die Geometrien im Resource Layer als Subelemente der Entity `IfcRepresentationItem`. Somit sind alle Formen von Geometrien, die mit IFC ausgetauscht werden können, nicht von `IfcRoot` abgeleitet und besitzen keine eigene *GlobalId*. Dies hat zur Folge, dass sie nur in Bezug auf ein anderes Element mit *GlobalId* ausgetauscht werden können.

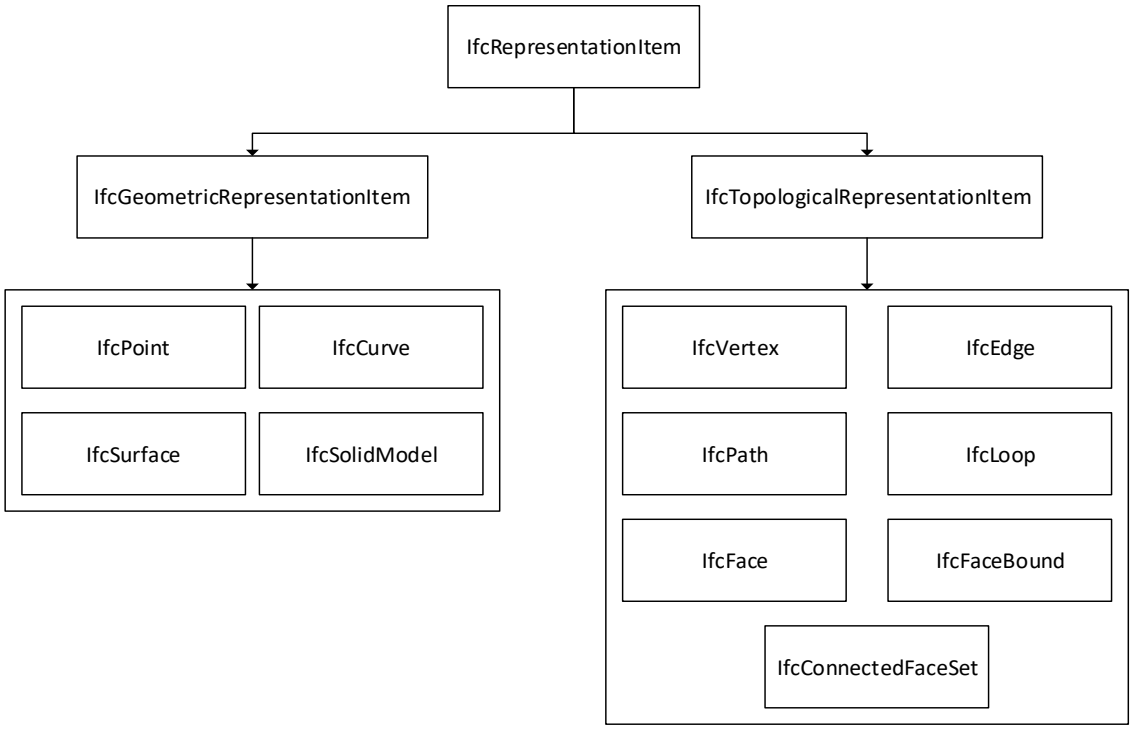


Abbildung 5-7: `IfcRepresentationItem` und seine wichtigsten Subtypes, die zur geometrischen Darstellung genutzt werden können

Die Entities zur geometrischen Darstellung lassen sich grob aufgliedern in Subtypes von `IfcGeometricRepresentationItem` und `IfcTopologicalRepresentationItem`.

Die Wichtigsten Subtypes von `IfcGeometricRepresentationItem` sind die vier Entities `IfcPoint`, `IfcCurve`, `IfcSurface` und `IfcSolidModel`. Diese stellen wiederum Subtypes zur Verfügung, die auf verschiedene Art

und Weise zur Darstellung von Punkten, Linien, Flächen und Volumenkörpern genutzt werden können. Für die Abbildung von Bauteilen sind die Darstellungsmöglichkeiten von Volumenkörpern von größerem Interesse. In IFC stehen dafür die drei Darstellungsverfahren CSG, Brep und Sweep zur Verfügung.

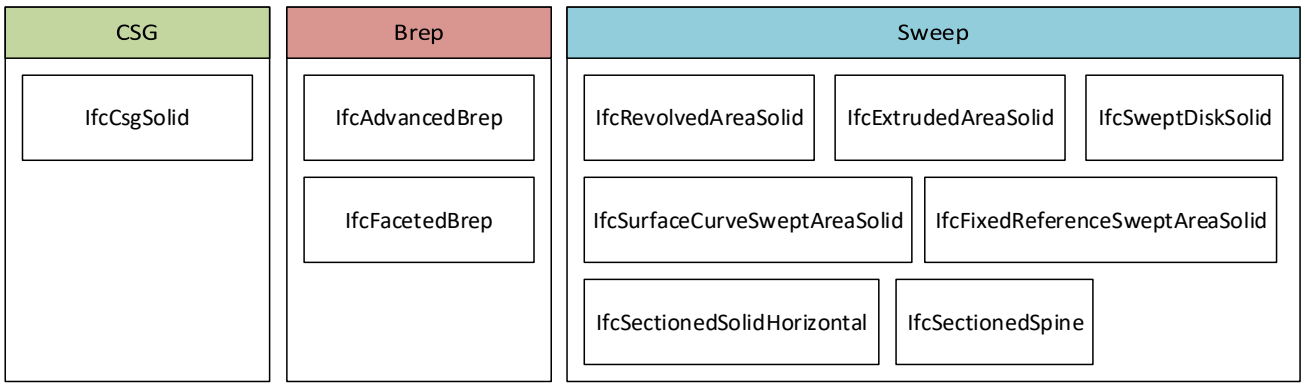


Abbildung 5-8: Übersicht über die vorhandenen Entities zur Darstellung von Volumenkörpern

Die Darstellungsweise Constructive Solid Geometrie (CSG) bedient sich der simplen geometrischen Grundfiguren Würfel, Pyramide, Kegel, Zylinder und Kugel, die in Kombination einen komplexeren Geometrischen Körper abbilden können.

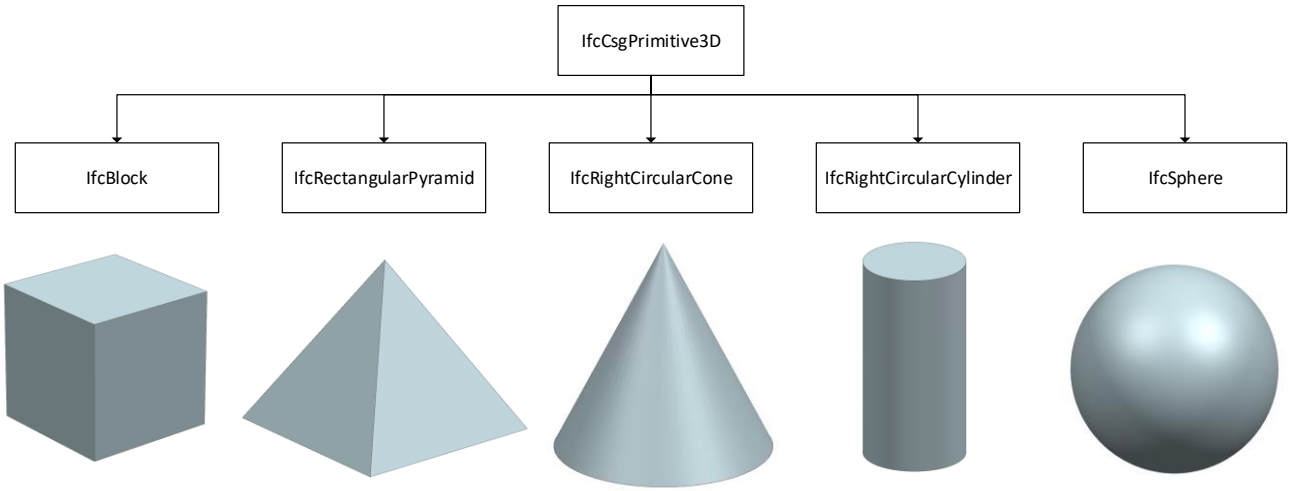


Abbildung 5-9: `IfcCsgPrimitive3D` und seine Subtypes zur Abbildung der geometrischen Grundfiguren Würfel, Pyramide, Kegel, Zylinder und Kugel

Zur Kombination der Grundfiguren stehen die drei booleschen Operatoren Union, Difference und Intersection zur Verfügung. Union ermöglicht die Verbindung zweier Volumenkörper zu einem gemeinsamen Körper. Durch die Anwendung von Difference kann hingegen von einem Volumenkörper ein anderer Volumenkörper abgezogen werden. Der dritte Operator Intersection erzeugt den Schnittkörper mehrerer Volumenkörper.

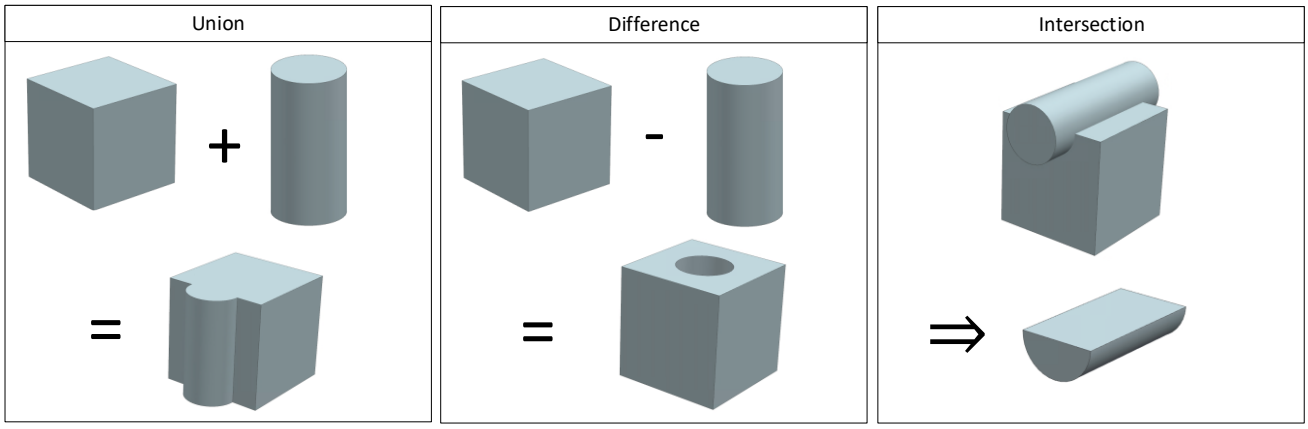


Abbildung 5-10: Exemplarische Abbildung der booleschen Operatoren Union, Difference und Intersection

Unter Boundary Representation (Brep) versteht sich die Abbildung eines Volumenkörpers über seine eingrenzenden Flächen. Diese Flächen wiederum werden definiert durch Randlinien, die durch das Verbinden von Punkten erzeugt werden.

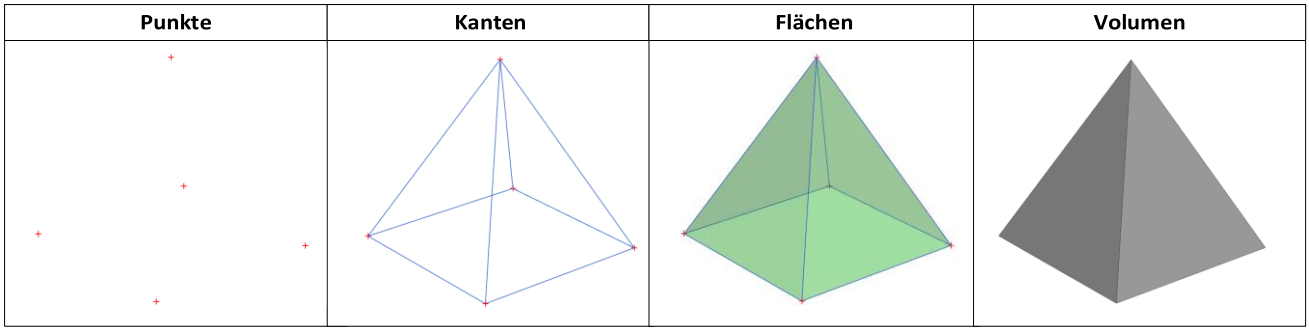


Abbildung 5-11: Die Schritte zur Darstellung eines Volumenkörpers mit Boundary Representation. Die Eckpunkte werden zunächst durch Linien verbunden, die sich wiederum zu Randlinien einer Fläche zusammensetzen. Mehrere Flächen bilden die Körperoberfläche und ermöglichen so die Definition eines Volumenkörpers.

Die dritte Variante zur Darstellung von Volumenkörpern ist das Sweep-Verfahren. Dieses umfasst neben dem so genannten Sweep auch die Extrusion und die Rotation. Grundlage für diese drei Darstellungen im Sweep-Verfahren ist ein geschlossenes Profil. Bei der Extrusion wird dieses Profil entlang eines Vektors in eine bestimmte Tiefe extrudiert. Die Profilfläche bildet dabei den Deckel und Boden des Volumenkörpers. Ein Sweep ist ähnlich der Extrusion, jedoch findet hier die Profilverschiebung nicht entlang eines Vektors, sondern entlang eines Pfades statt. Dadurch sind gekrümmte Verläufe möglich. Im IFC-Schema stehen unterschiedliche Entities zur Abbildung von Sweeps zur Verfügung, die jeweils eine besondere Art von Sweep erzeugen. Beispielsweise ist es mit den Entities `IfcSectionedSpine` und `IfcSectionedSolidHorizontal` möglich das Profil im Verlauf des Sweeps zu variieren. Bei der dritten Geometriedarstellung, der Rotation, wird das Profil um eine Rotationsachse in einem Rotationswinkel rotiert.

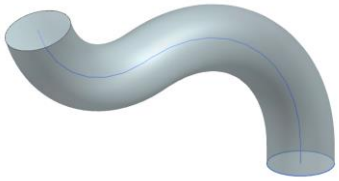
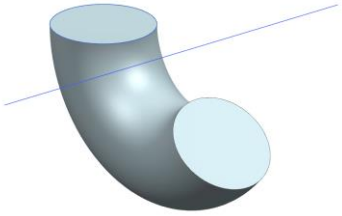
	Entities	**Komponenten**	**Beispiel**
Extrusion	IfcExtrudeAreaSolid	Profil Vektor Extrusionstiefe	
Sweep	IfcFixedReferenceSweptAreaSolid IfcSweptDiskSolid IfcSurfaceCurveSweptAreaSolid IfcSectionedSolidHorizontal IfcSectionedSpine	Profil(e) Pfad	
Rotation	IfcRevolveAreaSolid	Profil Rotationsachse Rotationswinkel	

Tabelle 5-5: Eine Übersicht über die im Sweep-Verfahren enthaltenen Darstellungsweisen Extrusion, Sweep und Rotation

Neben `IfcGeometricRepresentationItems` ermöglichen `IfcTopologicalRepresentationItems` die Darstellung von Objekten. Die Subklassen von `IfcTopologicalRepresentationItem` befassen sich dabei mit der Darstellung von Oberflächen. Dies geschieht ähnlich der Darstellungsweise der Boundary Representation über Punkte, Kanten, Randlinien und Flächen. Der Unterschied zur Brep ist, dass kein Volumenkörper abgebildet werden muss, sondern die Flächen auch eine offene Oberfläche beschreiben können.

5.2 Umsetzung der Datenkonvertierung

Auf Grundlage der wie im Kapitel 4.3 deserialisierten ISYBAU-Daten soll die Konvertierung nach IFC erfolgen. Dabei soll möglichst die bereits vorhandene Strukturierung der ISYBAU-Klassen genutzt und erweitert werden. Zum Erstellen der IFC-Objekte und zum Schreiben der Austauschdateien wird die IfcSharp-Library (https://github.com/IfcSharp/IfcSharpLibrary) verwendet. Diese stellt das IFC-Schema als Klassen in C# zur Verfügung und erzeugt aus den instanziierten Klassen Austauschdateien, unter anderem als STEP Physical File. Daher liegt das Hauptaugenmerk in den folgenden Kapiteln auf dem Mapping der ISYBAU-Daten zu geeigneten IFC-Klassen.

5.2.1 Programmaufbau

Der Quellcode des Programms strukturiert sich aus Gründen der Übersichtlichkeit in einzelne Dateien. Die automatisch aus dem Schema generierten ISYBAU-Klassen liegen in der isybau_classes.cs-Datei. Der Inhalt dieser Datei dient als Grundlage für den Import der ISYBAU-Daten. Dieser läuft über die isybau_import.cs-Datei ab, die zum größten Teil aus dem Deserialisierungsvorgang besteht. Die Konvertierung ist in der isybau_to_ifc.cs-Datei programmiert. Der Konvertierungsvorgang bedient sich einiger Hilfsklassen, die in der utility_classes.cs-Datei definiert sind. Gesteuert wird das gesamte Programm über die Main Methode in der Program.cs-Datei. Diese startet das Programm, und ruft die jeweilige Methode zum ISYBAU-Import und zur Konvertierung nach IFC auf.

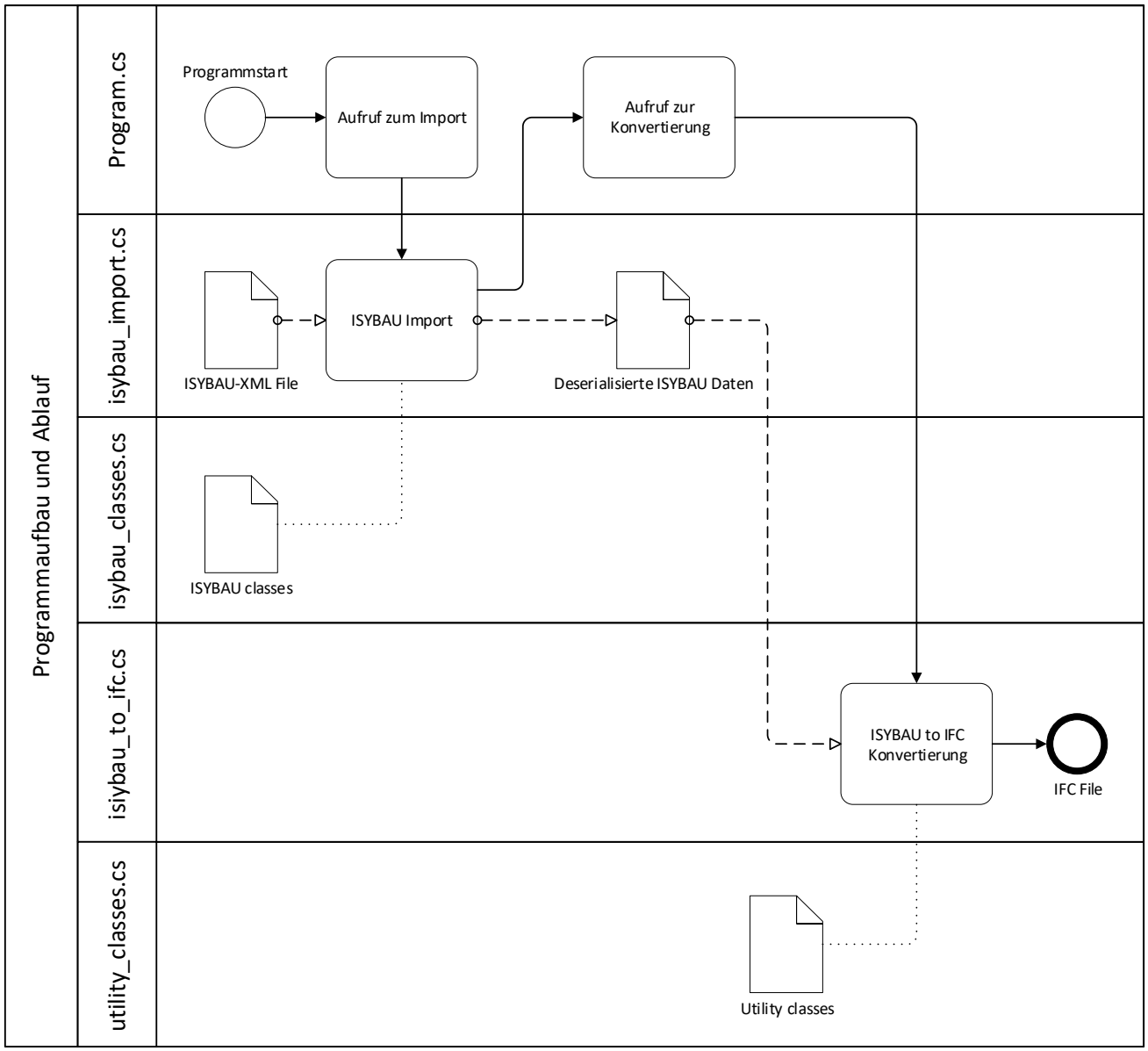


Abbildung 5-12: Übersicht über Programmaufbau und Ablauf des Konverters.

Durch die Deserialisierung liegen die ISYBAU-Informationen als Objekt `Austauschdatei` der Klasse `Identifikation` vor. Zur Konvertierung werden die Eigenschaften dieses Objekts durchlaufen und an entsprechender Stelle einem IFC-Objekt zugeordnet. Das Durchlaufen der Eigenschaften entspricht dabei dem Durchlaufen der Baumstruktur der eingelesenen ISYBAU-Datei. Dazu werden die bestehenden ISYBAU-Klassen um Eigenschaften und Methoden erweitert, die zur Konvertierung notwendig sind. Da die automatisch generierten ISYBAU-Klassen in der isybau_classes.cs Datei als `partial class` definiert sind, ist eine Erweiterung in der isybau_to_ifc.cs-Datei möglich. Beim Durchlaufen der Baumstruktur wird am jeweiligen Knoten die Methode `ToIfcInstance` aufgerufen. Diese erzeugt IFC-Objekte oder ruft

weitere Methoden des aktuellen Knotens auf. Am Ende der Methode wird die Methode `ToIfcInstance` des nächsten Knotens aufgerufen. Anhand der Abbildung 5-13 ist dieser Ablauf veranschaulicht dargestellt. Diese Vorgehensweise hat den Vorteil, dass das Programm beliebig erweitert werden kann indem die Methode `ToIfcInstance` am entsprechenden Knoten ergänzt wird.

isybau_to_ifc.cs

```
public partial class AbwassertechnischeAnlageType
{
    public void ToIfcInstance(List<object> parents)
    {
        parents.Add(this);

        if (this.Item is StammKanteType)
        {
            //Kante z.B. Haltung
            var kante = (StammKanteType)Item;
            List<object> elementparents = new List<object>(parents);
            kante.ToIfcInstance(elementparents);
        }
        else if (this.Item is StammKnotenType)
        {
            //Knoten z.B. Schacht
            var knoten = (StammKnotenType)Item;
            List<object> elementparents = new List<object>(parents);
            knoten.ToIfcInstance(elementparents);
        }
    }
}
```

Tabelle 5-6: Auszug aus der partial class `AbwassertechnischeAnlageType` mit Definition der Methode `ToIfcInstance`

Um den Ablauf zu verdeutlichen ist an dieser Stelle die Erweiterung der ISYBAU-Klasse `AbwassertechnischeAnlageType` um die Methode `ToIfcInstance` abgebildet. Der Methode wird die `parents`-Liste mit den Elternelementen übergeben, die in diesem Fall die konkreten Objekte `Identifikation`, `Datenkollektive` und `Stammdaten` enthält. Dies ist notwendig, da an manchen Stellen im Programm auf Eigenschaften von Elternelementen zugegriffen werden muss. Eine Navigation von einem Element der Klasse `AbwassertechnischeAnlageType` zurück zum Identifikationselement ist jedoch nicht mehr möglich. Daher wird das Elternelement in einer Liste geschrieben, welche beim Aufruf der Methode `ToIfcInstance` an das Kindelement übergeben wird. Dieser Schritt geschieht durch den Ausdruck `List<object> elementparents = new List<object>(parents);` vor dem Aufrufen der `ToIfcInstance` Methode des Kanten- oder Knotenobjekts. Somit verfügt jedes Element über eine eigne Liste, über die auf ein beliebiges vorheriges Elternelement zurückgegriffen werden kann.

Durch die Methode `ToIfcInstance` der Klasse `AbwassertechnischeAnlageType` erfolgt keine Konvertierung von ISYBAU-Daten in ein IFC-Objekt. Die Methode prüft lediglich ob es sich bei der abwassertechnischen Anlage um ein Kanten- oder ein Knotenobjekt handelt und ruft die

dementsprechende `ToIfcInstance` auf. Da diese ISYBAU-Eigenschaft im XSD-Schema als `choice` definiert ist (Siehe auch Seite 16) wird sie beim Deserialisieren der Eigenschaft `Item` vom `AbwassertechnischeAnlageType` zugeschrieben. Das Programm prüft daher an dieser Stelle den Type von `Item`. Da `Item` allgemein als `object` definiert ist und somit nicht wie ein Objekt der Klasse `StammKanteType` oder `StammKnotenType` behandelt werden kann, wird zunächst eine neue `Kante` oder ein neuer `Knoten` definiert, dem die Werte von `Item` zugeordnet werden. Anschließend wird die entsprechende Methode `ToIfcInstance` der Kante oder des Knotens aufgerufen.

isybau_to_ifc.cs

```
ifc.Model model = new Model();

ifc.Repository.CurrentModel = model;

new ifc.EntityComment("local coordinate system");
originField = new ifc.CartesianPoint(x: 0, y: 0, z: 0, EndOfLineComment: "origin");
dirXField = new ifc.Direction(x: 1, y: 0, z: 0, EndOfLineComment: "X-axis");
dirYField = new ifc.Direction(x: 0, y: 1, z: 0, EndOfLineComment: "Y-axis");
dirZField = new ifc.Direction(x: 0, y: 0, z: 1, EndOfLineComment: "Z-axis");

localCoordsField = new ifc.Axis2Placement3D(Location: Origin, Axis: DirZ, RefDirection:
DirX);

model.ToStepFile();
```

ISYBAU to IFC.ifc

```
/* local coordinate system                                                    */
#1=IFCCARTESIANPOINT((0.,0.,0.));/* origin */
#2=IFCDIRECTION((1.,0.,0.));/* X-axis */
#3=IFCDIRECTION((0.,1.,0.));/* Y-axis */
#4=IFCDIRECTION((0.,0.,1.));/* Z-axis */
#5=IFCAXIS2PLACEMENT3D(#1,#4,#2);
```

Tabelle 5-7: Erzeugen von IFC-Objekten mit Hilfe der IfcSharp-Library

Obenstehendes Beispiel soll veranschaulichen, wie mithilfe der IfcSharp-Library (https://github.com/IfcSharp/IfcSharpLibrary) IFC-Objekte erstellt werden können. Grundsätzlich kann diese über die bestehenden Quelldateien oder über eine aus den zur Verfügung stehenden Quellen erzeugte DLL-Datei eingebunden werden. In beiden Fällen werden die Entities des IFC-Schemas als C# Klassen zur Verfügung gestellt. Als ersten Schritt muss ein neues Model erzeugt werden und dem `Repository` als `CurrentModel` zugeordnet werden. In diesem Auszug wird in der Klasse `Identifikation` in der Methode `ToIfcInstance` der Ursprung des Koordinatensystems, sowie die X-, Y- und Z-Achse definiert. Dazu kann jeweils eine neue Instanz der benötigten Klassen `CartesianPoint` und `Direction` über den Befehl `new` erzeugt werden. In den runden Klammern nach dem Aufruf der Klasse werden die Werte für die Eigenschaften vergeben. Die IfcSharp-Library bietet neben den im IFC-Schema definierten Eigenschaften die Möglichkeit IFC-Objekte zu kommentieren. Dies geschieht über die optionale

Eigenschaft `EndOfLineComment` Außerdem kann über die Klasse `EntityComment` eine Kommentarzeile angelegt werden, wodurch die Austauschdatei übersichtlicher gestaltet werden kann. `originField`, `dirXField`, `dirYField` und `dirZField` sind `private` Eigenschaften, die in der `partial class Identifikation` ergänzt wurden und denen die IFC-Objekte zugeordnet werden. Zum Festlegen des Koordinatensystems über `localCoordsField` werden diesem die oben definierten Werte zugeordnet. Durch Aufrufen der Methode `ToStepFile()` werden die erzeugten IFC-Objekte automatisch in die Physical STEP File Austauschdatei „ISYBAU to IFC.ifc“ geschrieben. Die IFC Version entspricht dabei der der verwendeten IfcSharp-Library. Im Falle dieser Arbeit ist es die IFC-Version 4.1.

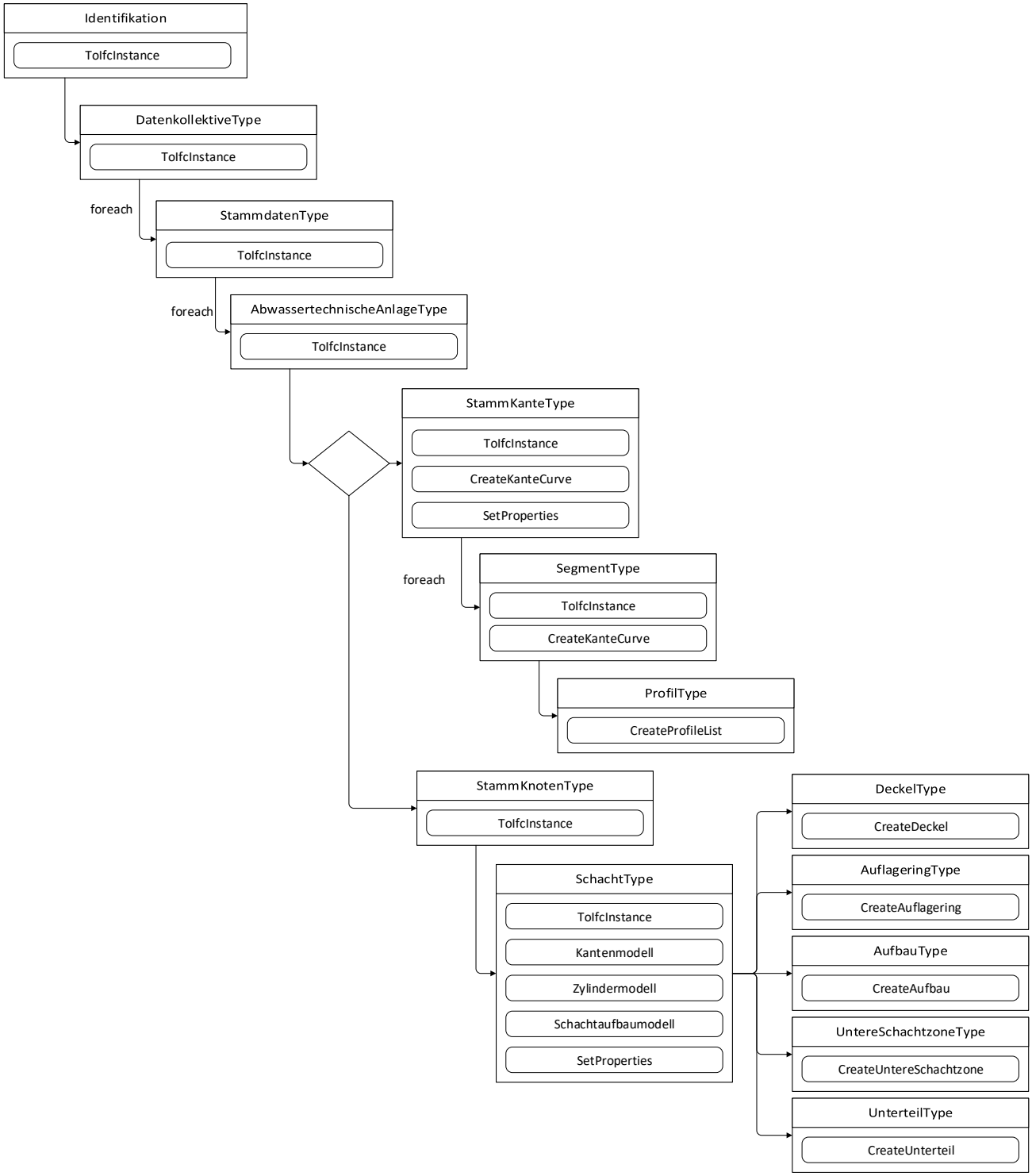


Abbildung 5-13: Programmablauf in der isybau_to_ifc.cs Datei zum Erzeugen der IFC-Austauschdatei über die Methoden `ToIfcInstance` auf Grundlage der eingelesenen ISYBAU-Inforamtionen

5.2.2 Gewählte IFC Product Types

Grundlegende Fragestellung diese Arbeit ist es, ob eine Konvertierung von ISYBAU-XML-Daten nach IFC möglich ist. Eine der Voraussetzungen hierfür ist, dass das IFC-Schema ausreichend Entities zur Verfügung stellt, um die ISYBAU-Daten in IFC abbilden zu können. Die abwassertechnischen Anlagen, die mit ISYBAU ausgetauscht werden, lassen sich grob in Knoten und Kantenobjekte gliedern. Dabei repräsentieren Kanten Haltungen, Leitungen, Rinnen und Gerinne, während Knoten Schächte, Anschlusspunkte und Bauwerke darstellen. Grundsätzlich eignen sich die Subklassen von `IfcDistributionFlowElement` für die Definition von abwassertechnischen Anlagen.

5.2.2.1 Kantenobjekte

Zur Abbildung der ISYBAU-Kantenobjekte eignet sich die Entity `IfcPipeSegment`. Diese steht im IFC-Schema zur Verfügung, um Rohrobjekte austauschen zu können. (buildingSMART International Limited, 2020) In der folgenden Tabelle sind die Attribute von `IfcPipeSegment` aufgelistet.

Nr.	Attribut	Type
1	GlobalId	IfcGloballyUniqueId
2	OwnerHistory	IfcOwnerHistory
3	Name	IfcLabel
4	Description	IfcText
5	ObjectType	IfcLabel
6	ObjectPlacement	IfcObjectPlacement
7	Representation	IfcProductRepresentation
8	Tag	IfcIdentifier
9	PredefinedType	IfcPipeSegmentTypeEnum

Tabelle 5-8: Attribute von `IfcPipeSegment`

Alle Entities, die Subklassen von `IfcElement` sind, erben die Attribute #1 bis #8 von `IfcElement`. Das hat eine einheitliche Vorgehensweise zur Definition von Bauobjekten zur Folge. Ein Großteil diese Entities besitzt zudem das Attribut #9 *PredefinedType*. Der Type dieses Attributes ist eine `TypeEnum`. Dies ist eine vorgefertigte Liste aus Werten, ähnlich der Referenzlisten des ISYBAU-Schemas, die auf das jeweilige zu beschreibende Bauobjekt angepasst ist. `IfcPipeSegmentTypeEnum` besitzt folgende Werte:

- CULVERT
- FLEXIBLESEGMENT
- RIGIDSEGMENT
- GUTTER
- SPOOL

- USERDEFINED
- NOTDEFINED

Das Programm wählt an dieser Stelle den Wert **USERDEFINED**. Diese geschieht, damit die tatsächlichen ISYBAU-Bezeichnungen Haltung, Leitung, Rinne und Gerinne in das IFC-Austauschformat konvertiert werden können. Am Beispiel einer Haltung soll dies veranschaulicht werden. Hierbei wird zunächst der *ObjectType* (Attribut #5) als **'Haltung'** definiert. Außerdem erhält das Attribut #3 *Name* die Objektbezeichnung der jeweiligen abwassertechnischen Anlage, in diesem Fall **'211099'**. Auf die Attribute #6 und #7 *ObjectPlacement* und *Representation* wird im folgenden Kapitel zur geometrischen Darstellung genauer eingegangen. Darüber hinaus ist anzumerken, dass die *GlobalId* bei der Instanziierung der Austauschdatei automatisch durch die IfcSharp-Library (https://github.com/IfcSharp/IfcSharpLibrary) erzeugt wird. Die benutzerdefinierte Beschreibung des Types erfolgt über die Entity `IfcPipeSegmentType`. Deren Attribute sind in der Tabelle 5-10 abgebildet. Die wichtigsten Attribute sind hier die Attribute #3 *Name*, #6 *HasPropertySets*, #9 *ElementType* und #10 *PredefinedType*. Die Attribute #3 und #9 erhalten den Wert **'Haltung'**. Über das Attribut #6 *HasPropertySets* werden die semantischen Informationen der ISYBAU-Austauschdatei über `IfcPropertySets` übergeben. Das Attribut #10 *ElementType* entspricht dem Attribut #9 von `IfcPipeSegment` und erhält ebenfalls den Wert **USERDEFINED**. Die Zuordnung von `IfcPipeSegmentType` zum `IfcPipeSegment` erfolgt über das Beziehungsobjekt `IfcRelDefinesByType`. *RelatedObject* ist dabei das `IfcPipeSegment` (#104), der *RelatingType* `IfcPipeSegmentType` (#107). Untenstehendes Beispiel ist schematisch in Abbildung 5-14 dargestellt und kann so für die Kantentypen Leitung, Rinne und Gerinne übernommen werden.

ISYBAU to IFC.ifc

```
/* IfcPipeSegment der Haltung                                          */

#104=ifcPipeSegment('3RSl0lh_99Pws_5VJjeivR',$,'211099',$,'Haltung',#103,#124,$,.USERDEF
INED.);

/* Definition der property sets                                        */

#105=ifcPropertySingleValue('ISYBAU_Profilhoehe',$,IFCLABEL('200'),$);

#106=ifcPropertySet('0R$9mELlj7Mh_Ie19oy3eh',$,'ISYBAU_Eigenschaften',$,(#105));

/* Type Definition                                                     */

#107=ifcPipeSegmentType('0ebioLIc50yuGTqGXAYhV_',$,'Haltung',$,$,(#106),$,$,'Haltung',.
USERDEFINED.);

#108=ifcRelDefinesByType('0FbaPM9010fheeO$ydS1VC',$,$,$,(#104),#107);
```

Tabelle 5-9: Instanziierung von `IfcPipeSegment` und zugehöriger Type Definition

Nr.	Attribut	Type
1	GlobalId	IfcGloballyUniqueId
2	OwnerHistory	IfcOwnerHistory
3	Name	IfcLabel
4	Description	IfcText
5	ApplicableOccurrence	IfcLabel
6	HasPropertySets	IfcPropertySetDefinition
7	RepresentationMaps	IfcRepresentationMap
8	Tag	IfcIdentifier
9	ElementType	IfcLabel
10	PredefinedType	IfcPipeSegmentTypeEnum

Tabelle 5-10: Attribute von `IfcPipeSegmentType`

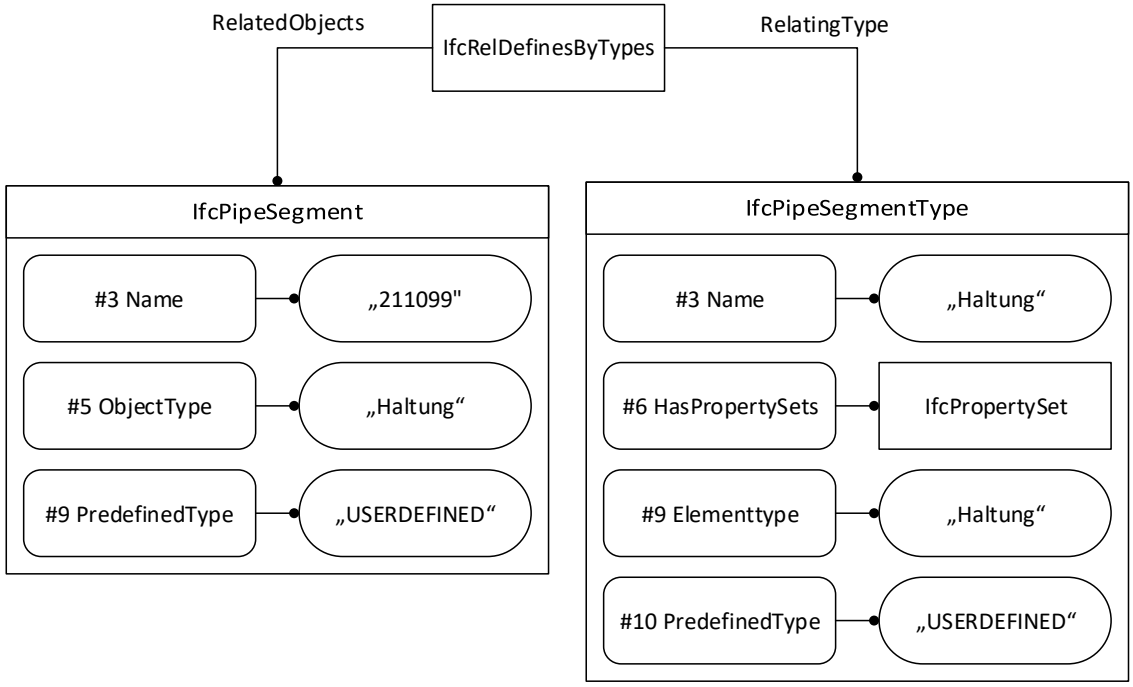


Abbildung 5-14: Schematische Darstellung der Type Zuordnung am Beispiel des Kantenobjekts Haltung

Wie bereits im oberen Absatz erwähnt, findet an dieser Stelle auch die Übertragung der semantischen ISYBAU-Informationen in die IFC-Austauschdatei statt. Genutzt wird dafür das Attribut #6 *HasPropertySets* von `IfcPipeSegmentType`, dem ein benutzerdefiniertes `IfcPropertySet` zugeordnet wird. Das `IfcPropertySet` (#106) erhält über Attribut #3 *Name* den Wert **`'ISYBAU_Eigenschaften'`**. Attribut #5 *HasProperties* enthält eine Liste von `IfcProperties`, die dem `IfcPropertySet` angehören. Im Beispiel wird die ISYBAU-Information Profilhoehe durch ein `IfcPropertySingleValue` (#105)

ausgedrückt. Genutzt wird dazu einmal das Attribut #1 *Name* zur Benennung der Property und #3 *NominalValue* zur Zuordnung des Wertes Profilhoehe aus der ISYBAU-Datei. Analog hierzu kann mit allen weiteren ISYBAU-Informationen verfahren werden. Auf diese Weise kann jeder abwassertechnischen Anlage ein eigenes `IfcPropertySet` eingerichtet werden, dessen `IfcProperties` dem jeweiligen Informationsgehalt im ISYBAU-Schema entspricht. Es ist anzumerken, dass das Programm im Zuge dieser Arbeit nur einzelne ISYBAU-Informationen berücksichtigt, da die Machbarkeit eine Konvertierung von ISYBAU-Daten nach IFC damit ausreichen nachgewiesen ist.

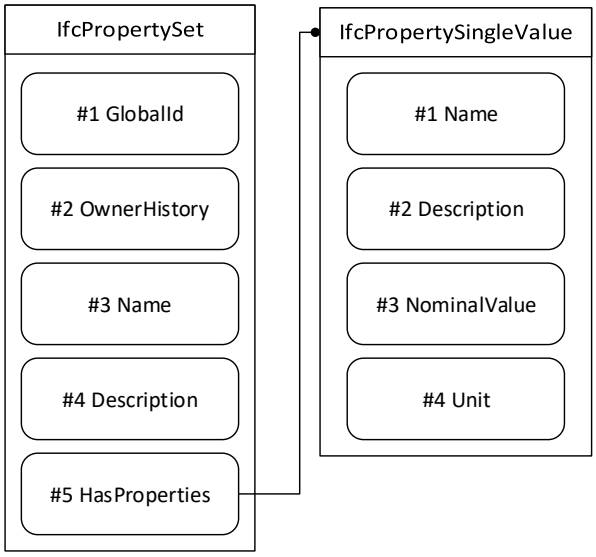


Abbildung 5-15: Schematische Darstellung vom Aufbau von `IfcPropertySet`

5.2.2.2 Knotenobjekte

Beim Mapping der ISYBAU-Knotenobjekte Schacht, Anschlusspunkt und Bauwerk wird analog zu den Kantenobjekten verfahren. Es wird jeweils eine Entity gesucht, die am besten das abzubildende ISYBAU-Objekt in IFC repräsentiert. Beschrieben wird der Type über die entsprechende Type Entity (siehe `IfcPipeSegment` und `IfcPipeSegmentType`) und `IfcPropertySet.` Für Schächte eignet sich die IFC Entity `IfcDistributionChamberElement` und die zugehörige Type Entity `IfcDistributionChamberElementType`. Beim ISYBAU-Objekt Anschlusspunkt ergibt sich die Problematik, dass es sich hierbei um ein virtuelles Objekt und nicht um ein reales Bauteil handelt. Eine geeignete IFC-Entity ist `IfcDistributionPort`. `IfcDistributionPort` besitzt jedoch anders als zum Beispiel `IfcPipeSegment` keine zugehörige Type Entity wie `IfcDistributionPortType`. Daher muss die Beschreibung des Types und die Zuordnung der Property Sets anderweitig erfolgen.

Nr.	Attribut	Type
1	GlobalId	IfcGloballyUniqueId
2	OwnerHistory	IfcOwnerHistory
3	Name	IfcLabel
4	Description	IfcText
5	ObjectType	IfcLabel
6	ObjectPlacement	IfcObjectPlacement
7	Representation	IfcProductRepresentation
8	FlowDirection	IfcFlowDirectionEnum
9	PredefinedType	IfcDistributionPortTypeEnum
10	SystemType	IfcDistributionSystemEnum

Tabelle 5-11: Attribute von `IfcDistributionPort`

Es bietet sich an, dem Attribut #5 *ObjectType* den Wert „Anschlusspunkt“ zu vergeben. Anders als bei den vorherigen Beispielen sollte jedoch das Attribut #9 nicht auf „USERDEFINED“ gesetzt werden. Stattdessen gibt die Enumeration den Wert „PIPE“ vor, der für das hier abzubildende ISYBAU-Objekt geeignet ist. Zudem besitzt `IfcDistributionPort` das Attribut #10 *SystemType* zur Beschreibung des Verteilersystems. Für den Fall einer abwassertechnischen Anlage eignet sich der Wert „WASTEWATER“. Somit ist der Type des Anschlusspunktes über IFC ausreichend abgebildet. Die Zuordnung von `IfcPropertySet` kann jedoch hierbei nicht über ein direktes Attribut erfolgen, sondern muss über eine Beziehung gelöst werden. Das benötigte Beziehungsobjekt ist `IfcRelDefinesByProperties`. Dieses stellt die Verbindung zwischen `IfcDistributionPort` und dem entsprechendem `IfcPropertySet` her (vergleiche Abbildung 5-16). Die Definition von `IfcPropertySet` erfolgt analog zum oben beschriebenen Ablauf.

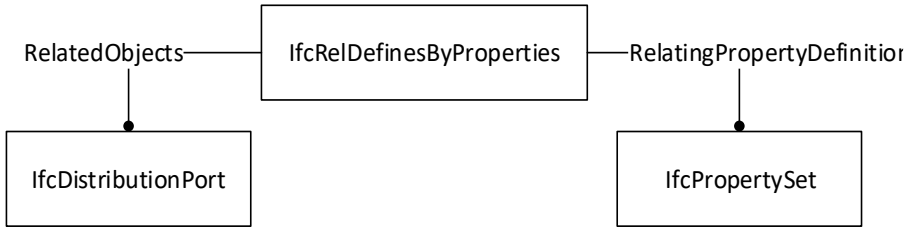


Abbildung 5-16: Herstellen einer Beziehung zwischen `IfcDistributionPort` und `IfcPropertySet` über das Beziehungsobjekt `IfcRelDefinesByProperties`

Beim dritten ISYBAU Knotenobjekt Bauwerk stellt sich das Problem, dass damit eine Vielzahl von unterschiedlichen Bauwerken beschrieben wird. Daher ist es nicht ausreichend eine allgemeingültige IFC Entity zu finden. Stattdessen muss für jeden Bauwerk Typ eine geeignete Entity gefunden werden. Die Tabelle 5-12 enthält eine Liste der ISYBAU-Bauwerke mit Zuordnung geeigneten IFC Entities. Zur Beschreibung des Types und der Zuordnung zu Property Sets existiert für jeden dieser Entities ein zugehöriger Type Entity.

ISYBAU-Bauwerk-Type	IFC Entity
Pumpwerk	IfcDistributionCamberElement
Becken	IfcFlowStorageDevice
Behandlungsanlage	IfcFlowTreatmentDevice
Klaeranlage	IfcFlowTreatmentDevice
Auslaufbauwerk	IfcFlowTerminal
Pumpe	IfcFlowMovingDevice
Wehr_Ueberlauf	IfcFlowController
Drossel	IfcFlowController
Schieber	IfcFlowController
Rechen	IfcFlowTreatmentDevice
Sieb	IfcFlowTreatmentDevice
Versickerungsanlage	IfcFlowTerminal
Zisterne	IfcFlowStorageDevice
Einlaufbauwerk	IfcFlowTerminal

Tabelle 5-12: Vorschlag geeigneter IFC Entities zur Abbildung von ISYBAU-Bauwerken

Als Zwischenfazit ist anzumerken, dass auf diese Weise sich die semantischen Informationen aus dem ISYBAU-Schema verlustfrei nach IFC übertragen lassen. Ausschlaggebend hierfür ist die Möglichkeit benutzerdefinierte Property Sets zu definieren, die den Informationsgehalt einer ISYBAU-Datei abdecken können. Das IFC-Schema stellt, wie in diesem Kapitel dargelegt, ausreichend Entities zur Verfügung, um alle ISYBAU-Kanten und -Knotenobjekte abzubilden. Als nächster Schritt stellt sich die Frage, ob das IFC-Schema geeignet ist neben den semantischen Informationen auch die Geometrien der ISYBAU-Objekte darzustellen.

5.2.3 Geometrische Darstellungsmöglichkeiten

Um die Geometrien der abwassertechnischen Anlagen in IFC darstellen zu können, muss zunächst analysiert werden, wie die geometrischen Informationen in ISYBAU vorliegen. Dabei ist wiederum eine Unterscheidung zwischen Kanten- und Knotenobjekten zu treffen. Kantenobjekte können entweder aus einer Kante oder aus einem aus mehreren Kanten zusammengesetzten Polygonzug bestehen. Eine Kante wird definiert durch einen Start und einen Endpunkt, beziehungsweise im Falle von gekrümmten Kanten aus einem Start, Mittel und Endpunkt. Das Ziel der Darstellungsweise in IFC soll jedoch kein Kantenmodell, sondern ein Volumenmodell sein. Dazu müssen in die geometrische Darstellung die Profilinformationen mit einfließen, um aus Kante und Profil einen Volumenkörper zu erzeugen. Aufgrund dieser Ausgangslage ist es sinnvoll, eine Darstellungsweise nach dem Sweep-Verfahren zu wählen.

Das klassische ISYBAU-Knotenobjekt ist ein Schacht. Als Geometrieinformationen besitzt dieser in der Regel die beiden Punkte Schachtmittelpunkt (SMP) und Deckelmittelpunkt (DMP). Der Schachtmittelpunkt liegt in der Mitte der Schachtsohle, der Deckelmittelpunkt an der Oberkante der Schachtabdeckung. Ergänzt werden diese geometrischen Informationen, ähnlich wie bei den Kantenobjekten, durch semantische Informationen. Ein Schachtobjekt untergliedert sich im ISYBAU-Schema in die einzelnen Schachtbauteile Deckel, Auflagering, Aufbau, untere Schachtzone und Unterteil. (siehe Abbildung 5-17) Über deren Eigenschaften wie Höhe und Breite lassen sich zusätzliche Informationen für die geometrische Darstellung in IFC als Volumenkörper erhalten. Als mögliche Darstellungsweise bietet sich auch hier das Sweep-Verfahren an, wobei aus den ISYBAU-Informationen eine Schachtmittelachse (in Abbildung 5-17 rot dargestellt) abgeleitet werden muss, die als Sweep-Kurve dient.

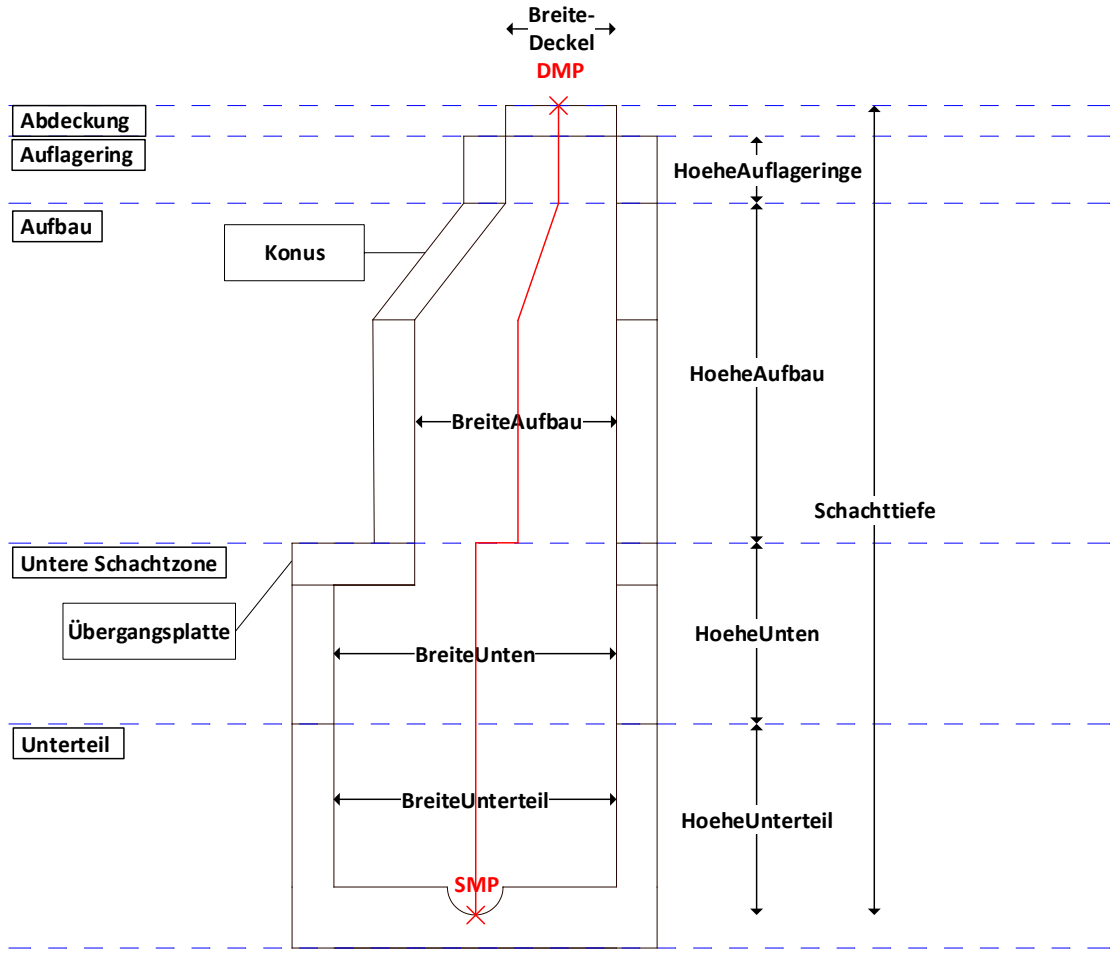


Abbildung 5-17: Schachtaufbau in Anlehnung an DIN EN 13508-2 Bild 1 (Deutsches Institut für Normen e.V., 2011, p. 12) mit Angaben der Schachtinformationen in ISYBAU

5.2.3.1 Übersicht der Darstellungsmöglichkeiten als Volumenmodell

Wie im vorangegangenen Kapitel ausgearbeitet, eignet sich das Darstellungsverfahren der Sweeps am besten, sowohl für die Kanten als auch Knotenobjekte. Daher werden an dieser Stelle die verschiedenen Sweep-Elemente im IFC-Schema analysiert und auf ihre Tauglichkeit zur Darstellung von abwassertechnischen Anlagen überprüft. Dafür ergeben sich gewisse Kriterien, die erfüllt sein müssen. Um diese zu verstehen, sollte zunächst ein genauerer Blick auf das Sweep-Verfahren geworfen werden. Die Darstellungsweise setzt sich im Wesentlichen aus den drei Komponenten Extrusionspfad, Extrusionsprofil und Profilpositionierung zusammen. Entlang des Extrusionspfades wird das Profil extrudiert, sodass ein Volumenkörper entsteht. Die Definition des Profils erfolgt dabei in der x-y-Ebene eines lokalen Koordinatensystems.

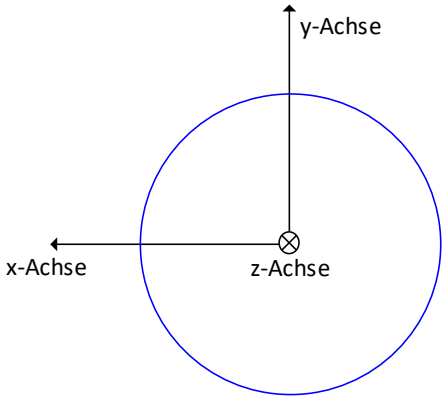


Abbildung 5-18: Definition eines Profils (blau) in der x-y-Ebene. Die z-Achse zeigt in die Ebene hinein.

Unterschiedliche Verfahren haben dabei unterschiedliche Richtlinien für die Form, Anzahl und Position der Profile. Die Profilform kann entweder frei definierbar sein, oder eine feste geometrische Form annehmen. Außerdem ermöglichen einige Sweep-Verfahren eine Variation der Profilform entlang der Extrusion.

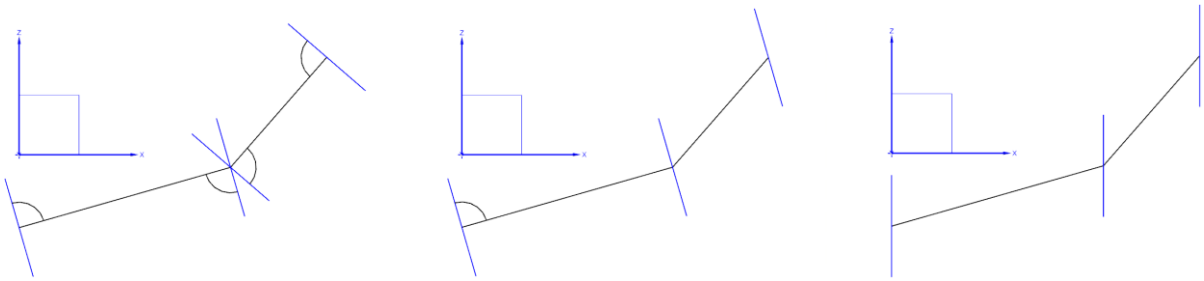


Abbildung 5-19: Unterschiedliche Verfahren zum Ausrichten des lokale Koordinatenssystems der Proftle: Senkrecht zum Pfad (links), alle Profile orientieren sich an einer Ausrichtung (mitte) oder feste Ausrichtung am globalen Koordinatensystem (rechts)

Es ergeben sich aus dem hier vorliegenden Anwendungsfall drei maßgebliche Kriterien:

1. Der Sweep muss entlang eines gekrümmten Pfades möglich sein, um Rohrleitungen mit Knicken und auch Schächte mit Konen darstellen zu können.
2. Die Profile sollten entlang des Sweeps variieren dürfen, um einen Rohrverlauf mit Muffen oder einen Schachtaufbau mit Konus abbilden zu können.
3. Die Position und Ausrichtung der Profile sollten frei wählbar sein, oder dem Anspruch einer Rohrleitung oder eines Schachtes entsprechen. Das heißt im Falle eines Rohres, dass das Profil sich im rechten Winkel zum Pfad ausrichtet und im Falle eines Schachtes immer in der Horizontalen liegt.

Das Ziel der Bewertung ist es, eine Darstellungsweise zu finden, die möglichst alle denkbaren Anforderungen an die Repräsentation als Volumenmodell erfüllt.

Die Darstellungsweise der Extrusion über die IFC Entity `IfcExtrudeAreaSolid` ist nicht geeignet für die Darstellung von Kantenobjekten, da die Extrusion entlang eines Vektors erfolgt. Somit sind keine Knicke möglich. Mithilfe der Subklasse von `IfcExtrudeAreaSolid` `IfcExtrudeAreaSolidTapered` ist zwar die Variation des Profils zwischen Start und Endprofil möglich, (buildingSMART International Limited, 2020) dies ist für diesen Anwendungsfall jedoch nicht ausreichend. Dasselbe gilt für die Darstellung von Knotenobjekten.

`IfcSweptDiskSolid` ist eine Darstellungsweise, bei der ein Kreisprofil entlang eines Radius extrudiert wird. Das Kreisprofil wird definiert durch einen Radius und kann optional durch einen inneren Radius ergänzt werden, sodass als Ergebnis ein Ringprofil entsteht, welches beim Extrudieren ein Rohr ergibt. Die Start- und Endposition der Extrusion kann über Stationierungen auf der Kurve festgelegt werden. Die Ausrichtung der Profile erfolgt dabei immer zentral entlang der Kurve, sodass diese Darstellungsweise gut geeignet ist, um rohrförmige Objekte abzubilden. (buildingSMART International Limited, 2020) Der Nachteil besteht darin, dass keine Änderung des Profils entlang des Sweeps möglich ist. Außerdem ist eine Abbildung von nicht kreisförmigen Profilen, wie zum Beispiel Ei-Profilen oder offene Gerinnen, nicht möglich. Die Kanten im ISYBAU-Schema repräsentieren die Sohle eines Abwassersystems, währen die Kurve im `IfcSweptDiskSolid` im Mittelpunkt des Profils liegt. Für eine korrekte Darstellung müsste somit die Kante in der Höhe um den halbe Innendurchmesser nach oben verschoben werden. Bei der Darstellung von Knotenobjekten wie beispielsweise Schächten, ist es ebenfalls problematisch, dass keine Variation des Profils entlang des Sweeps möglich ist. Nicht-kreisförmige Schachtprofile können nicht über `IfcSweptDiskSolid` dargestellt werden. Zudem wäre es im Falle von Übergangsplatten im Schachtaufbau notwendig, das innere Kreisprofil exzentrisch definieren zu können. Somit ist `IfcSweptDiskSolid`, auch wenn grundsätzlich für die Darstellung von Rohren gut anwendbar, für den Anwendungsfall der Konvertierung von ISYBAU-Daten nach IFC nicht geeignet.

Die Darstellungsweise über `IfcSectionedSpine` hat den Vorteil, dass zwei oder mehr Profile frei definiert werden können, die entlang einer Kurve vom Type `IfcCompositeCurve` extrudiert werden. Dadurch lassen sich alle Profilformen abbilden. Zudem ist eine Änderung des Profils innerhalb eines Sweeps möglich. Sowohl die Positionierung als auch die Ausrichtung muss ebenfalls frei definiert werden, wobei die Positionierung nicht anhand einer Stationierung entlang der Kurve, sondern durch einen Koordinatenpunkt erfolgt. (buildingSMART International Limited, 2020) Somit sind alle drei festgelegten Kriterien erfüllt und die Darstellungsweise `IfcSectionedSpine` sowohl für Kanten als auch Knotenobjekte geeignet. Ein Nachteil liegt jedoch im Aufwand bei der Angabe der Positionierung und Ausrichtung der Profile bei der Abbildung von Kantenobjekten. Start und Ende sind von Stationierungen auf der Kurve in Koordinatenpunkte umzurechnen. Außerdem muss die Ausrichtung für jeden dieser Punkte so bestimmt werden, dass das Profil senkrecht auf der Kurve steht. Bei Knotenobjekten wie Schächten verringert sich dieser Aufwand, da sich die Positionen aufgrund der gegebenen Punkte „SMP“ und „DMP“ einfacher ermitteln lassen und die Ausrichtung der Profile immer in der horizontalen Ebene liegen.

Einen anderen Ansatz zur Festlegung der Profilausrichtung verfolgt die Darstellungsweise `IfcFixedReferenceSweptAreaSolid`. Die Positionierung erfolgt wie bei `IfcSweptDiskSolid` über eine Start- und Endpositionierung entlang der Kurve. Das Profil ist frei definierbar, kann jedoch im Verlauf nicht variieren. Ausgerichtet wird das Profil durch die Angabe einer Referenzrichtung. Die lokale z-Achse des Koordinatensystems entspricht der Tangente an der Kurve. Über die angegebene Referenzrichtung wird die Richtung der x-Achse abgeleitet und das Koordinatensystem entsprechend um die y-Achse ergänzt. (buildingSMART International Limited, 2020) Bei Rohrsystemen besteht der Vorteil, dass sich das Profil im Verlauf der Extrusion zwar um die x-Achse neigen und um die y-Achse drehen, jedoch nicht um die z-Achse verdrehen kann. Daraus lässt sich ableiten, dass als Referenzrichtung, die in der x-z-Ebene liegt, immer die globale x-Achse genutzt werden kann. Einzige Ausnahme ist der Fall, dass die lokale z-Achse auf der globalen x-Achse liegt, da somit keine x-z-Ebene ermittelt werden kann. Bei Knotenobjekten kann dieser Fall nicht auftreten, da die lokale und globale z-Achse immer übereinstimmen.

Einen kombinierten Ansatz aus `IfcSectionedSpine` und `IfcFixedReferenceSweptAreaSolid` bietet die Darstellungsweise `IfcSectionedSolidHorizontal`. Es können mehrere Profile frei definiert werden, die im Verlauf variieren können. Die Positionierung erfolgt über Stationierungen entlang der Kurve. Die Ausrichtung der Profile erfolgt nicht frei wie bei `IfcSectionedSpine`. Die y-Achse zeigt entweder in globale z-Richtung oder orientiert sich senkrecht an der Kurventangente, was der Anforderung Nummer 3. gerecht wird. Zudem ist eine Variation der Profile entlang der Extrusion möglich, wodurch alle der drei Kriterien erfüllt werden. (buildingSMART International Limited, 2020) Nachteilig ist jedoch, dass die Entity erst mit der aktuellsten IFC-Version 4.1 eingeführt wurde. Somit haben zum Zeitpunkt dieser Arbeit viele Programme diese Darstellungsweise noch nicht implementiert, was eine Validierung des geometrischen Ergebnisses erschwert.

Als letzte Art der Sweep-Verfahren verfolgt die Entity `IfcRevolveAreaSolid` einen komplett anderen Weg zur geometrischen Abbildung, bei dem ein Profil um eine Achse rotiert wird. (buildingSMART

International Limited, 2020) Das Profil, das nunmehr nicht dem Querschnitt, sondern dem Längsschnitt eines rohrförmigen Objekts gleicht, kann frei definiert werden. Durch die Rotation und die Festlegung einer Rotationsachse ist eine Darstellung von gekrümmten Rohrsystemen nicht möglich. Allerdings ist diese Darstellungsweise eine gute Möglichkeit, um beispielsweise gerade Rohrstücke mit komplexen Längsschnitten abzubilden. Bei den üblichen Sweep-Verfahren wäre dies nur über die Variation von vielen Profilen entlang der Extrusion möglich. Die Rotation ermöglicht eine Abbildung mithilfe eines einzelnen Profils.

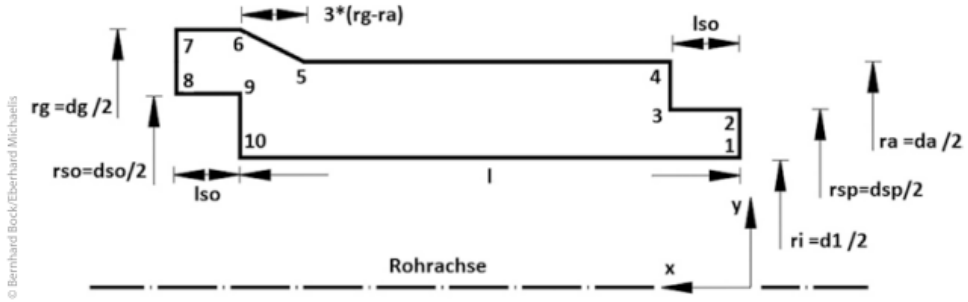


Abbildung 5-20: Exemplarische Aufbau eines Profils zur Erzeugung eines Rohrsegmentes über das Rotationsverfahren (Bock & Michaelis, 2019, p. 39)

IFC Entity	Vorteile	Nachteile
IfcExtrudeAreaSolid; IfcExtrudeAreaSolidTapered	+ Profilform frei definierbar + Profiländerung möglich	- nur geradlinige Extrusion entlang eines Vektors möglich
IfcSweptDiskSolid; IfcSweptDiskSolidPolygonal	+ Schnelle und einfache Möglichkeit, um Rohre darzustellen + Automatische Ausrichtung der Profile an der Kurve	- keine Profiländerung - nur kreisförmige Profile - Zentrale Positionierung des Profils an der Kurve
IfcSectionedSpine	+ Gekrümmte Extrusion möglich + Profile in Form und Position frei definierbar + Profiländerung möglich	- Aufwendige Bestimmung der Position und Ausrichtung der Profile für jede einzelne Stationierung
IfcFixedReferenceSwept-AreaSolid	+ Gekrümmte Extrusion möglich + Profilform frei definierbar + Automatische Bestimmung der z-Achse anhand der Kurventangente	- Keine Profiländerung

IfcSectionedSolidHorizontal	+ Profile in Form und Position frei definierbar + Ausrichtung der Profile an globaler z-Achse oder an Kurve + Profiländerung möglich	- Zurzeit in vielen Softwareanwendungen noch nicht implementiert
IfcRevolveAreaSolid	+ Gute Möglichkeit, um gerade Rohrsegmente mit komplexem Längsschnitt darzustellen	- Nur gerade Rohrsegmente

Tabelle 5-13: Übersicht über die verschiedenen Darstellungsweisen des Sweep-Verfahrens und Bewertung hinsichtlich der Nutzung als Darstellungsweise von abwassertechnischen Anlagen

5.2.3.2 Durchführung der geometrischen Darstellung

Aufgrund der Analyse und Beurteilung der möglichen Darstellungsweisen als Sweep in IFC, fällt die Wahl zu Abbildung von Kantenobjekten auf die Entity `IfcSectionedSolidHorizontal`. Diese wurde mit dem Ziel eingeführt, linienförmige Infrastrukturbauwerke wie Brücken darzustellen. Ein Blick auf die Attribute verrät genaueres über die Funktionsweise dieser Entity.

Nr.	Attribut	Type
1	Directrix	IfcCurve
2	CrossSections	IfcProfileDef
3	CrossSectionPosition	IfcDistanceExpression
4	FixedAxisVertical	IfcBoolean

Tabelle 5-14: Attribute von `IfcSectionedSolidHorizontal`

Attribut #1 enthält die Kurve, entlang der extrudiert werden soll. Aus den vorhandenen ISYBAU-Informationen muss nun eine geeignete Kurve in IFC erzeugt werden. In der ISYBAU-Datei werden die Kurven durch Start und Endpunkte definiert. Im Fall von Kreissegmenten liegt zusätzlich ein Mittelpunkt vor. `IfcIndexedPolyCurve` ist eine IFC Entity mit der Kurven und Polykurven auf eine sehr ähnliche Weise beschrieben werden (siehe auch Abbildung 5-21: Zusammensetzung von `IfcIndexedPolyCurve`). Zum einen liegt dabei eine Liste von Punkten vor und zum anderen eine Liste von Segmentinformationen. Diese enthalten Indizes, die jeweils auf Punkte der Punktliste verweisen und somit definieren, welche Punkte Teil einer Kante sind. Analog zu den Kantendefinitionen des ISYBAU-Schemas bestehen gerade Kantensegmente aus zwei Punkten. Kreissegmente werden durch drei Punkte beschrieben, wobei der dritte Punkt nicht den Kreismittelpunkt, sondern einen weiteren Punkt auf der Kurve darstellt.

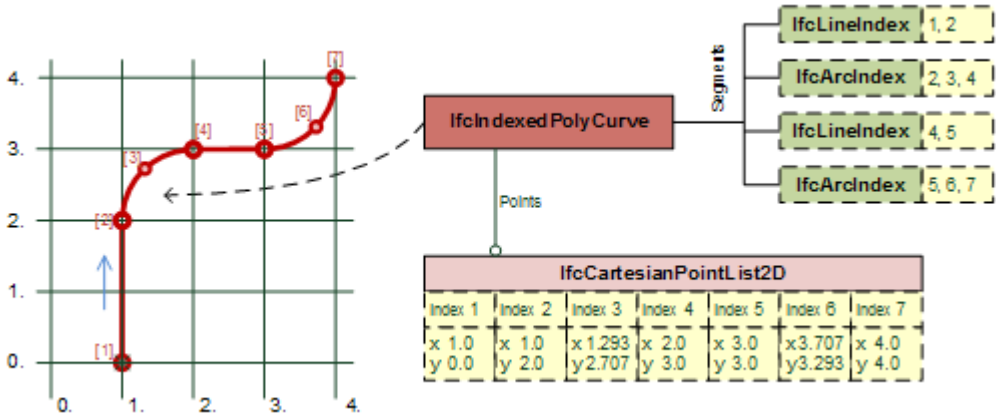


Abbildung 5-21: Zusammensetzung von `IfcIndexedPolyCurve` (buildingSMART International Limited, 2020))

Es bedarf somit einer Berechnung dieses dritten Punktes P3 über die bekannten Punkte S (Start), M (Mitte) und E (Ende). Als ersten Schritt wird der Hilfspunkt H mithilfe der Formel $\overrightarrow{OH} = \frac{1}{2} * (\overrightarrow{OS} + \overrightarrow{OE})$ bestimmt. Der Vektor $\overrightarrow{MH} = \overrightarrow{OH} - \overrightarrow{OM}$ zwischen den Punkten M und H bildet ausgehend vom Punkt M die Richtung, in der der Punkt P3 auf dem Kreissegment liegt. Der Abstand zum Punkt M entspricht dem Radius $r = |SM|$. Über Addition des Produktes aus Einheitsvektors $\overrightarrow{MH}^0$ und Radius zum Punkt M erhält man den Punkt P3:

$$\overrightarrow{OP3} = \overrightarrow{OM} + \overrightarrow{MH}^0 * r$$

Dieser liegt auf dem Kreissegment auf halber Strecke zwischen den Punkten S und E.

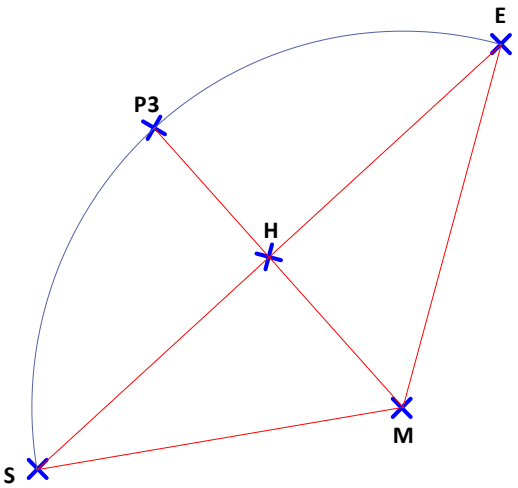


Abbildung 5-22: Ermittlung des Punktes P3 auf einem Kreissegment mithilfe der gegebenen Punkte Start, Mitte und Ende

Als nächster Schritt sind die zu extrudierenden Profile zu definieren. Diese werden über das Attribut #2 *CrossSections* in einer Liste von mindestens zwei Profilelementen der `IfcSectionedSolidHorizontal` zugewiesen. Die IFC Entity `IfcArbitraryProfileDefWithVoids` ermöglicht über die Festlegung einer äußeren Begrenzungskurve und mehrerer innerer Öffnungen eine flexible Profilform. In Abbildung 5-23 ist exemplarisch ein kreisförmiges Rohrprofil dargestellt. Dieses besteht aus einem äußeren und einem inneren Kreis. Bei Kreisprofilen entspricht die ISYBAU-Information Profilhoehe der Nennweite (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 691), woraus sich der innere Kreisradius r ermitteln lässt. Der äußere Radius kann entweder dem ISYBAU-Aussendurchmesser entnommen werden, oder durch „das Durchmesser/Wanddicken-Verhältnis (en: standard dimension ratio; SDR)“ (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 691) ermittelt werden. Da der Pfad, entlang dem extrudiert wird, der Sohle eines Kantenobjektes und nicht der Achse entspricht, ist es notwendig den Profilmittelpunkt so zu wählen, dass der Ursprung des lokalen Koordinatensystems, in dem das Profil definiert wird, sich in der Sohle des Profils befindet. Im konkreten Beispiel bedeutet dies, dass der Kreismittelpunkt vom Ursprung aus um den Innenradius in y-Richtung verschoben werden muss. Nicht-kreisförmige Profile, wie beispielsweise Ei-Profile, lassen sich über die Profilbreite, Profilhoehe und die genaue Zuordnung zur Profilart über die entsprechende Referenzliste Profilart konstruieren. Im Falle von Sonderprofilen liegt der Profilquerschnitt in einer Koordinatenliste vor, woraus sich eine IFC-Kurve ableiten lässt.

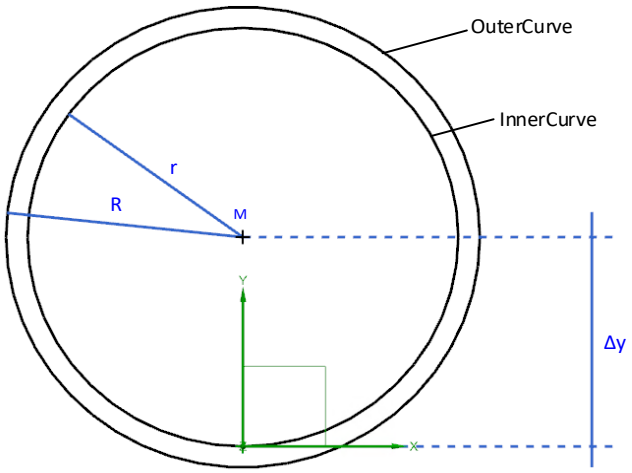


Abbildung 5-23: Kreisförmiger Rohrquerschnitt, beschrieben durch eine äußere und innere Kurve.

Nach der Konvertierung der Kanten- und Profilgeometrie folgt die Stationierung der Profile entlang der Extrusionskurve. Dabei liegen Start- und Endstationierung nicht immer am Kantenstart beziehungsweise ende, da die Kantenlänge nicht zwangsläufig der Rohrlänge entspricht. Im Falle von Haltungen zwischen Schächten ist es die Regel, dass das Kantenobjekt Haltung von Schachtmittelpunkt bis Schachtmittelpunkt verläuft. Das Rohr beginnt jedoch erst am Anschlusspunkt zum Schacht. Daher ermittelt das Programm

bereits bei der Konvertierung der Kantengeometrie in eine `IfcIndexedPolyCurve` über das Punktattribut Rohranschlusspunkt (RAP) der Kantenpunkte, ab welcher Länge entlang der Kurve das Rohr beginnt und endet. Auf diese Weise lassen sich Kantenobjekte als Gesamtobjekt abbilden.

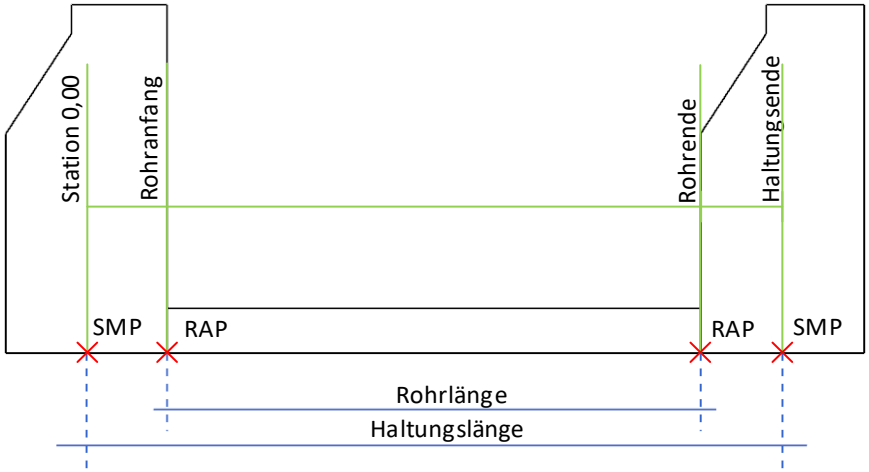


Abbildung 5-24: Unterschied zwischen Haltungs- und Rohrlänge

Die Ausrichtung der Profile erfolgt über das Attribut #4 *FixedAxisVertical*. Hierbei handelt es sich um einen booleschen Ausdruck, der die Werte true oder false annehmen kann. Somit sind zwei Arten von Ausrichtung vorgeschrieben, welche das Programm nicht selbständig über Koordinatenachsen definieren muss. Die erste Variante beim Wert `true` sieht vor, dass sich die y-Achse des Profils in positive globale z-Richtung orientiert. Man kann also vereinfacht behaupten, die Profile stehen vertikal. Die Drehung erfolgt immer entlang des Extrusionspfades, sodass eine Darstellung wie links in Abbildung 5-25 entsteht. Variante zwei, beim Wert `false`, richtet die lokale y-Achse des Profils senkrecht zum Extrusionspfad aus, während die lokale z-Achse tangential zum Extrusionspfad verläuft. In beiden Fällen zeigt die lokale x-Achse in Pfadrichtung betrachtet nach rechts.

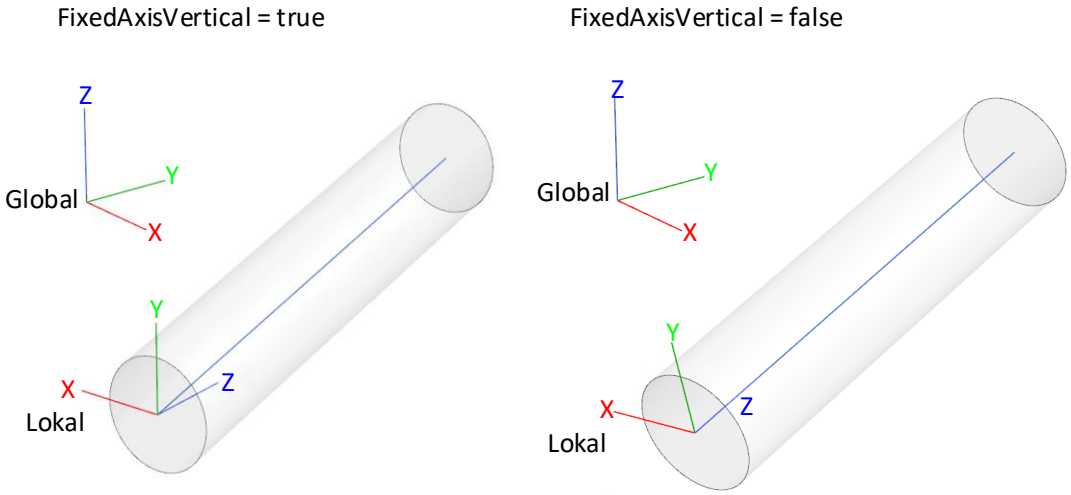


Abbildung 5-25: Abbildung als `IfcSectionedSolidHorizontal`, einmal mit FixedAxisVertical = true (links) und FixedAxisVertical=false (rechts)

Zur Abbildung von Rohrobjekten ist somit die Darstellung mit `FixedAxisVertical` = `false` geeignet, da Rohre in der Regel eine senkrechten Rohrabschluss in Bezug auf die Rohrachse aufweisen.

Darüber hinaus bietet ISYBAU XML-2017 die Möglichkeit Kantenobjekte in einzelne Segmente zu unterteilen. Durch diesen Schritt erhöht sich der Detaillierungsgrad, da Kantenobjekte nun mehr nicht als ein monolithisches Objekt, sondern als ein aus mehreren Bauteilen zusammengesetztes Bauwerk behandelt werden können. Am Beispiel einer Haltung setzt sich diese aus mehreren Rohrsegmenten zusammen, die miteinander verbunden das übergeordnete Bauwerk der Haltung ergeben. Die geometrische Unterteilung von Kanten in Segmente in IFC gestaltet sich weitestgehend unkompliziert, da die Position der Segmente in ISYBAU über Stationierungsangaben definiert ist, die so in Attribut #3 *CrossSectionPosition* von `IfcSectionedSolidHorizontal` übernommen werden können.

An dieser Stelle folgt ein Abschnitt, der über die Konvertierung von ISYBAU-Daten nach IFC hinaus geht, da zusätzliche Informationen für eine erweiterte geometrische Darstellung genutzt werden. Aufgrund der Aufteilung von Kantenobjekten in einzelne Segmente entstand die Idee, diese Rohrsegmente detailliert mit der Geometrie ihrer Muffen darzustellen. Hierzu bietet es sich an zusätzliche geometrische Informationen, zum Beispiel aus Produktdatenblättern von Rohrherstellern, in das Programm einzuspeisen, die die ISYBAU-Informationen ergänzen. Daraus können weitere Rohrprofile erzeugt werden, die entsprechend der Muffengeometrie entlang der Segmente stationiert werden.

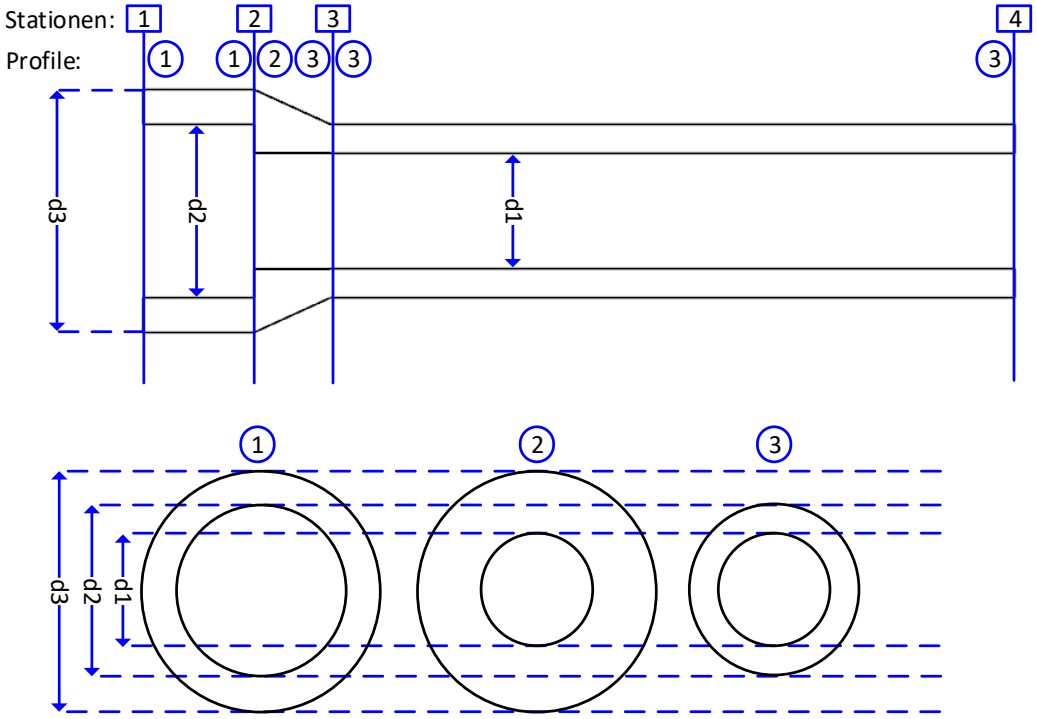


Abbildung 5-26: Abbildung eines Rohrsegments mit Muffengeometrie durch Stationierung variierender Rohrprofile

Die Abbildung 5-26 veranschaulicht wie eine solche Darstellung umgesetzt wird. Es gibt entlang des Rohrsegmentes Stationierungen, an denen eine Profiländerung vorliegt. Diese erfolgt durch eine Änderung der Durchmesser d1, d2 und d3. Im Bereich von Station 1 bis 2 liegt kontinuierlich der Rohrquerschnitt 1 vor. An Station 2 erfolgt ein Sprung im Querschnitt von Profil 1 auf Profil 2. Auch solche plötzlichen Profiländerungen ohne Übergangsbereich könne über `IfcSectionedSolidHorizontal` abgebildet werden. Station 2 entspricht dabei gleichzeitig dem Segmentstart nach ISYBAU, da die Muffe zwischen Station 1 und 2 über das ISYBAU-Segment hinausragt. Von Station 2 auf 3 findet ein kontinuierlicher Profilübergang von Profil 2 auf Profil 3 statt. Im letzten Bereich bis Station 4 liegt durchgehend der Profilquerschnitt 3 vor. Das Resultat ist in Abbildung 5-27 dargestellt. Eine solche erweiterte Darstellung kann zu Visualisierungszwecken genutzt werden. Außerdem können herstellerspezifische Rohrgeometrien in das Modell mit einfließen.

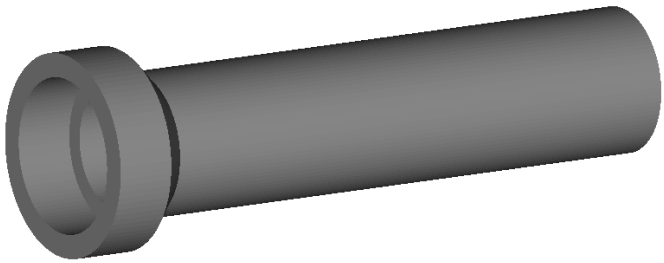


Abbildung 5-27: Darstellung eines Rohrsegmentes mit Muffe über `IfcSectionedSolidHorizontal` in FZKViewer

Somit ist die Darstellung von Kantenobjekten über `IfcSectionedSolidHorizontal` abgeschlossen. Anschließend wird nun die Darstellung von Knotenobjekten behandelt. Hierfür wurde die IFC Entity `IfcSectionedSpine` als geeignet befunden, da sie eine freie Profildefinition und eine Variation der Profile ermöglicht. Anders als bei Kantenobjekten ist bei Koten jedoch die Ausrichtung der Profile simpler, da diese in der globalen horizontalen Ebene liegen. Zudem muss die Positionierung nicht über Stationierung erfolgen, sondern kann über Koordinatenpunkte ermittelt werden. Ein Blick auf die Attribute von `IfcSectionedSpine` zeigt, dass die Entity die Anforderungen erfüllt. Die folgende Beschreibung zur Darstellung bezieht sich auf das Knotenobjekt Schacht, da sich für die restlichen Knotenobjekte Anschlusspunkt und Bauwerk über die ISYBAU-Informationen nicht genügend Informationen sammeln lassen um diese sinnvoll als Volumenmodell darzustellen. Was sich jedoch anbieten würde, wäre eine symbolische Darstellung über primitive geometrische Objekte.

Nr.	Attribut	Type
1	SpineCurve	IfcCompositeCurve
2	CrossSections	IfcProfileDef
3	CrossSectionPositions	IfcAxis2Placemet3D

Abbildung 5-28: Attribute von `IfcSectionedSpine`

Die *SpineCurve*, die den Extrusionspfad definiert, ist im Type beschränkt auf `IfcCompositeCurve`, welche sich jedoch wiederum aus Segmenten zusammen setzt, die durch eine beliebige Kurve vom Type `IfcBoundedCurve` beschrieben werden. Um die *SpineCurve* zu erzeugen, werden die ISYBAU-Geometrieinformationen Schachtmittelpunkt und Deckelmittelpunkt genutzt und mit den Höhenangaben zum Schachtaufbau ergänzt. Somit lassen sich markante Punkte im Schachtaufbau ermitteln, die über `IfcPolyline` miteinander verbunden werden (siehe Abbildung 5-29) und die Segmente der `IfcCompositeCurve` bilden (siehe Abbildung 5-30).

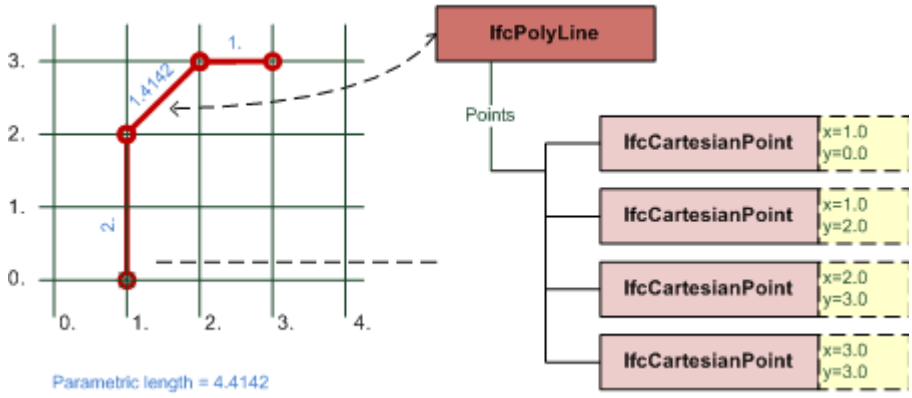


Abbildung 5-29: Zusammensetzung von `IfcPolyline` (buildingSMART International Limited, 2020)

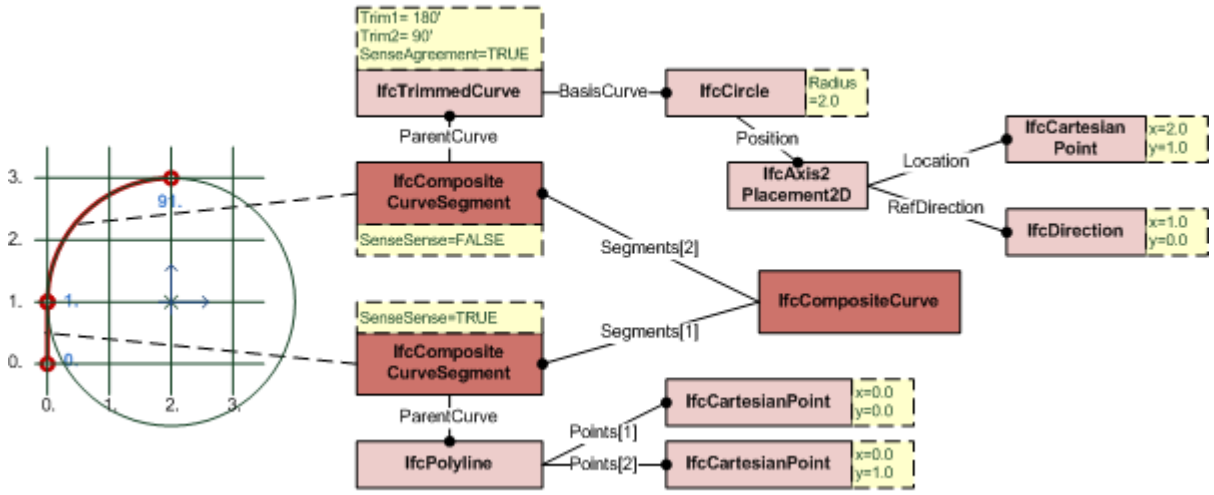


Abbildung 5-30: Zusammensetzung von `IfcCompositeCurve` (buildingSMART International Limited, 2020)

Ein Nachteil liegt darin, dass in der Praxis die Höhenangaben zum Schachtaufbau häufig nicht in den ISYBAU-Daten enthalten sind. Dies gilt ebenso für die Schachtbreiten, die zum Abbilden der Profile benötigt werden. Grundsätzlich erfolgt die Profildefinition auf dieselbe Art und Weise wie bei den Rohrprofilen. Lediglich eine Exzentrizität, wie sie aufgrund der Sohle bei Rohren auftritt, braucht nicht beachtet werden. Das ISYBAU-Schema sieht jedoch, anders als bei Kantenprofilen, keine Wandstärke für Schachtobjekte vor. Die Positionierung und Ausrichtung der Profile erfolgt anhand des Attributes #3 *CrossSectionPosition* über den Type `IfcAxis2Placement3D`. Definiert wird hierbei ein Ursprungspunkt, die Richtung der z-Achse und eine Richtung zur Bestimmung der x-Achse. In diesem Koordinatensystem wird das Profil in der x-y-Ebene abgebildet. Da die Orientierung der Koordinatenachsen des lokalen Koordinatensystems dem des globalen entsprechen, können die beiden Richtungsangaben frei gelassen werden. Das IFC-Schema sieht in diesem Fall vor, dass die Koordinatenachsen sich am globalen System ausrichten. (buildingSMART International Limited, 2020) Die Ursprungspunkte entsprechen den Punkten

im Schachtaufbau, die bereits für die Definition der *SpineCurve* ermittelt wurden, und können übernommen werden.

Auf diese Weise lassen sich Schächte über `IfcSectionedSpine` geometrisch abbilden. Allerdings reichten die ISYBAU-Informationen nicht aus, um ein genaues Abbild eines Schachtes zu definieren. Zum einen fehlen die Wandstärken, sodass nur der Schachthohlraum bestimmt ist. Zum anderen wird ein Konus oder eine Übergangsplatte in der Geometrie nicht weiter beschrieben. ISYBAU liefert lediglich die Information, ob in entsprechendem Schacht ein solches Bauteil verbaut ist. Außerdem liegen, wie oben erwähnt, die Höhen und Breiten der einzelnen Schachtbauteile häufig nicht vor. Daher erzeugt das Programm aus den ISYBAU-Informationen geometrische Darstellungen mit unterschiedlichem Detailierungsgrad, um generell eine Abbildung zu ermöglichen. Die erste orientiert sich dabei an der klassischen ISYBAU-Darstellung als Knotenmodell. Hierbei werden Schachtsohle und Schachtdeckel als Kreise abgebildet. Die zweite Variante stellt den Schacht als Volumenmodell dar, indem der Kreis der Schachtsohle bis auf die Deckelhöhe extrudiert wird. Die dritte Darstellungsweise nutzt ähnlich der erweiterten Darstellung von Rohrsegmenten zusätzliche Informationen zu den Schachtbauteilen. Bei dieser erweiterten geometrischen Darstellung wird der Schacht zudem in seine einzelnen Schachtbauteile unterteilt. Konkret bedeutet dies, dass Schachtunterteil, untere Schachtzone, Schachtaufbau, Ausgleichsringe und Abdeckung eigenständige Geometrien aufweisen und eine Kombination aus diesen Bauteilen den Schacht als Ganzes abbildet.

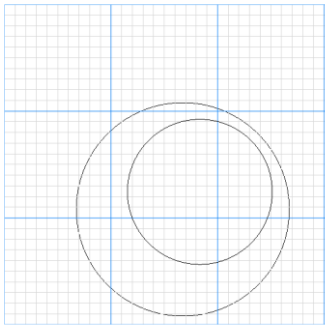
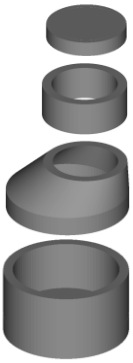
Knotenmodell	Zylindermodell	Erweiterte Darstellung
		

Abbildung 5-31: Die drei unterschiedlichen Darstellungsweisen eines Schachts als Kantenmodell in der Draufsicht (links), Zylindermodell (mitte) und erweiterte Darstellung (rechts).

5.2.4 Verknüpfung von Semantik und Geometrie

In den vorangegangenen Kapiteln wird unter anderem die Konvertierung der semantischen, sowie der geometrischen ISYBAU-Informationen beschrieben. Das Ergebnis in der IFC-Datei besteht aus der Kombination dieser beiden Informationsteile. Dabei liefert das Programm aufgrund der Schwierigkeiten

mit dem geometrischen Informationsgehalt drei unterschiedliche Repräsentationen. Zum einen ein Knoten-Kantenmodell, das als direkte Konvertierung der ISYBAU-Darstellung gesehen werden kann. Zum anderen eine Sweep-Darstellung der Knoten und Kanten anhand eines Profils. Hierzu werden die Profile kreisförmig aus der Breite des Schachtunterteils beziehungsweise aus der Profilhöhe erzeugt, da diese Informationen in der Praxis mit großer Wahrscheinlichkeit vorliegen. Diese beiden Darstellungen lassen sich somit für alle ISYBAU-Dateien durchführen. Die erweiterte Darstellung setzt einen gewissen Informationsgehalt voraus, um durch ergänzende Informationen ein detaillierteres Abbild zu schaffen. Zu diesen Informationen gehört zum einen die Unterteilung der Kantenobjekte in einzelne Segmente und die Beschreibung der einzelnen Schachtbauteile über Höhe und Breite.

Betrachtet wird nun die Verknüpfung der Geometrie mit den semantischen Informationen. Diese kann anhand von Abbildung 5-32 nachvollzogen werden. Jede der gewählten Entities aus Kapitel 5.2.2 zur semantischen Abbildung der abwassertechnischen Anlagen besitzt als Subklasse von `IfcProduct` das Attribut #7 *Representation*, dessen Type `IfcProductRepresentation` ist. Über Attribut #3 *Representations* von `IfcProductDefinitionShape`, einem Subtype von `IfcProductRepresentation`, können mehrere geometrische Repräsentationen in Form von `IfcShapeRepresentations` zugeordnet werden. Die `IfcShapeRepresentation` enthält letztendlich die `IfcRepresentationItems` wie `IfcSectionedSolidHorizontal` oder `IfcSectionedSpine`, die die geometriechen Informationen enthalten. Für eine bessere Darstellung im Viewer soll es allerdings möglich sein, das gesamte Modell in Richtung Ursprung zu transformieren. Daher wird die Entity `IfcMappedItem,` die eine Transformierung über Attribut #2 *MappingTarget* ermöglicht, zwischengeschaltet. Diese Transformation vom Type `IfcCartesianTransformationOperator3D` transformiert über Attribut #3 *localOrigin* das `IfcMappedItem` um x-, y- und z-Koordinate. Für diese Werte können manuell zum Beispiel die Koordinaten einer abwassertechnischen Anlage gewählt werden, sodass diese in den globalen Ursprung transformiert wird. Es ist wichtig, dass für alle `IfcMappedItems` derselbe Transformationswert genutzt wird, damit alle Geometrien im selben Maße transformiert werden. Die Geometrien, wie `IfcSectionedSolidHorizontal` oder `IfcSectionedSpine,` werden zunächst über Attribut #4 *Items* einer weiteren `IfcShapeRepresentation` zugeordnet, welche wiederum über Attribut #2 *MappedRepresentation* einer `IfcRepresentationMap` zugeordnet wird. `IfcMappedItem` vereint über Attribut #1 *MappingSource* die geometrische Darstellung der `IfcRepresentationMap` mit der Transformation. Zu erwähnen ist außerdem, dass über Attribut #6 *ObjectPlacement* des `IfcProduct` der globale Koordinatenursprung angegeben wird, da sich alle vom Programm behandelten Koordinatenangaben auf diesen Ursprung beziehen.

Abbildung 5-32: Zuordnung der geometrischen Darstellung zu einem Bauteil

Bei der Darstellung als Knoten-Kantenmodell werden die Kurven, die zur Darstellung genutzt werden, direkt über Attribut #4 *Items* von `IfcShapeRepresentation` zugeordnet. Ebenso verhält es sich mit den Entities `IfcSectionedSolidHorizontal` und `IfcSectionedSpine` bei der Darstellung als Zylindermodell. Bei der erweiterten Darstellungsweise ergibt sich die Besonderheit, dass Knotenobjekte, wie zum Beispiel Schächte, und Kantenobjekte wie Haltungen in weitere Bauteile aufgegliedert werden. Diese Bauteile erhalten die geometrische Darstellung, was zur Folge hat, dass die übergeordneten Knoten- und Kantenobjekte keine eigene geometrische Darstellung besitzen. Allerdings wird über das Bezugsobjekt `IfcRelAggregates` eine Beziehung zwischen den Bauteilen zum übergeordneten Objekt hergestellt. Am Beispiel einer Haltung sieht dies so aus, dass die Haltung das *RelatingObject* ist, während die einzelnen Rohrsegmente die *RelatedObjects* bilden. Dadurch ergibt sich, wie in der Browser Toolbar des FZKViewers gut zu erkennen ist, die Aufteilung der beiden Schachtelemente mit den Objektbezeichnungen 211099 und 211100 in die Bestandteile Unterteil, Aufbau, Auflagering und Deckel, sowie die Unterteilung der Haltung 211099 in die Segmente 1 bis 9.

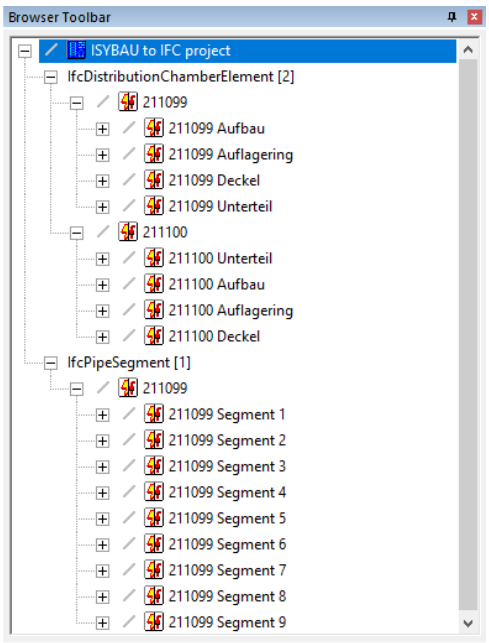


Abbildung 5-33: Unterteilung von Knoten- und Kantenobjekten bei der erweiterten Darstellung.

5.3 Ergebnisbetrachtung

Um das Ergebnis der Arbeit zu betrachten wird der entwickelte Prototyp des Konverters an einer simplen ISBAU-Datei getestet und der Konvertierungsvorgang bewertet. Nach dem Starten der Anwendung wird der Benutzter über einen Dialog zum Öffnen der ISYBAU-Datei aufgefordert. Als nächsten Schritt erfolgt die Deserialisierung der Daten anhand der automatisch aus dem ISYBAU-Schema generierten C#-Klassen. Sollten die Daten diesem Schema nicht entsprechen, bricht das Programm den Vorgang ab. Ansonsten wird über die Konsole die Zeile „`ISYBAU-Datei eingelesen`“ ausgegeben.

```
ISYBAU to IFC Konverter
Copyright(c) 2020 Matthias Wiedemann
this software use IfcSharp (see https://github.com/IfcSharp)
ISYBAU-Datei eingelesen
```

Abbildung 5-34: ISYBAU-Datei eingelesen

Anschließend erfolgt die folgende Abfrage:

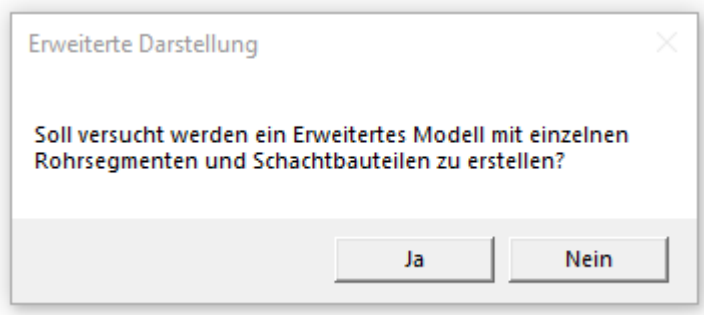


Abbildung 5-35: Abfrage zur Darstellungsweise

Wählt der Benutzer „Ja“, erstellt der Konverter ein erweitertes Modell wie in Kapitel 0 beschrieben. Sollten nicht ausreichend Informationen hierfür in den ISYBAU-Daten enthalten sein, wird für das entsprechende Objekt keine Geometrie erstellt. Die Konsole gibt zudem aus, welches Objekt betroffen ist.

```
keine ausreichenden Informationen für Objekt:211112 vorhanden
```

Abbildung 5-36: keine ausreichenden Informationen vorhanden

Für den Fall, dass keine erweiterte Darstellung erfolgen soll, wird die Darstellung als Knoten-Kantenmodell und als Zylindermodell erzeugt. Im Konvertierungsvorgang generiert das Programm mithilfe der IfcSharp-Library (https://github.com/IfcSharp/IfcSharpLibrary) IFC-Elemente und füllt diese mit den eingelesenen ISYBAU-Daten. Abschließend wird die IFC-Datei geschrieben, was ebenfalls

über die IfcSharp-Library geschieht. Nach abgeschlossener Konvertierung gibt die Konsole die Zeile „`Konvertierung abgeschlossen`“ aus und fordert den Benutzer auf eine beliebige Taste zu drücken, um das Programm zu beenden.

```
Konvertierung abgeschlossen
Beliebige Taste drücken
```

Abbildung 5-37: Konvertierung abgeschlossen

Von Interesse sind nun die Ergebnisdateien, die im Ordner der .exe-Datei liegen. Betrachtet wird zunächst das Ergebnis der einfachen Darstellungsweise über den FZKViewer. Wie in Abbildung 5-38 links zu erkennen ist, wird nach dem Einlesen der Austauschdatei der Schachtaufbau über Zylinder vereinfacht. Das Haltungsobjekt zwischen den Anschlusspunkten zu den Schächten wird nicht in einzelne Segmente unterteilt und als ein zylinderförmiges Objekt dargestellt. In derselben Austauschdatei mitenthalten ist die Darstellung als Knoten-Kantenmodell (siehe Abbildung rechts), die sich einzeln für die Objekte `IfcDistributionChamberElement` und `IfcPipeSegment` umschalten lässt. Hierbei erfolgt mit entsprechender Höhe und Durchmesser die Darstellung von Schachtsohle und Abdeckung. Die Haltung wird durch eine Kante repräsentiert, die von Schachtmittelpunkt zu Schachtmittelpunkt auf Höhe der Haltungssohle verläuft. Unabhängig von der Darstellungsweise erhalten Schacht und Haltung wie im Kapitel 5.2.2 erläutert Ihre Beschreibung über Types und Property Sets.

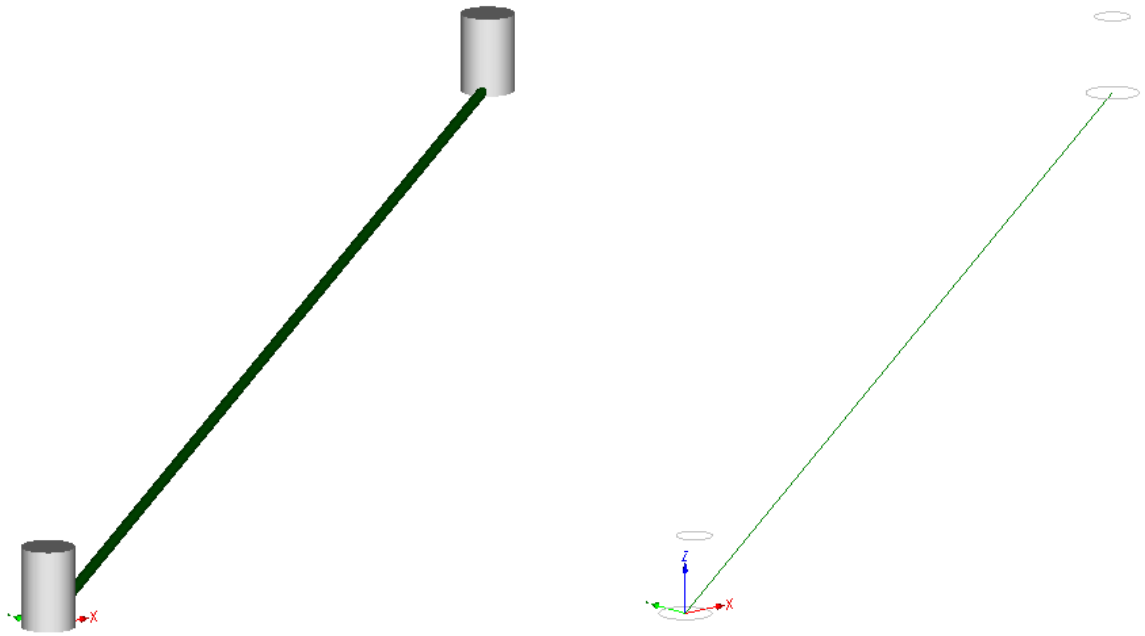


Abbildung 5-38: Darstellung der konvertierten ISYBAU-Daten im FZKViewer als Zylindermodell (links) und Knoten-Kantenmodell (rechts)

Die Austauschdatei zur erweiterten Darstellung enthält mehr Objekte als die vorher beschriebene Ergebnisdatei, da Schacht und Haltung in einzelne Bauteile untergliedert werden. Die Schächte bestehen aus Unterteil, Aufbau, Auflagering und Deckel, die Haltung aus den einzelnen Rohrsegmenten. Die Schachtelemente sind dabei Instanzen der Entity `IfcDistributionChamberElement`, da das IFC-Schema keine passenden Entities für den Schachtaufbau zur Verfügung stellt. Die Beschreibung der Elemente erfolgt analog zum einfachen Modell, wobei hierbei auch die Schachtelemente und Rohrsegmente Types und Property Sets erhalten.

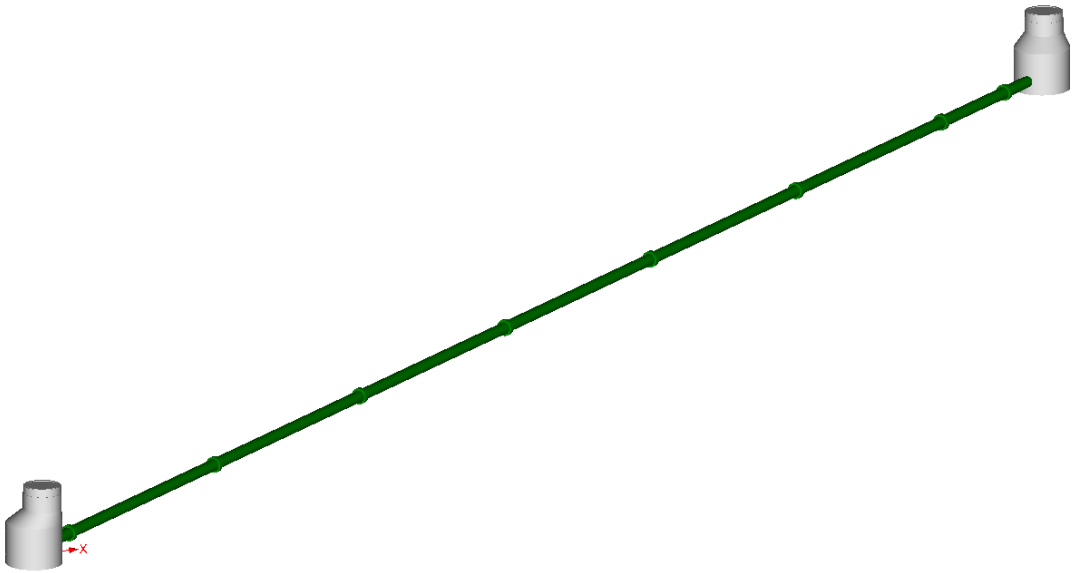


Abbildung 5-39: Erweiterte Darstellung der konvertierten ISYBAU-Daten im FZKViewer. Schacht und Haltung setzen sich dabei aus mehreren Bauteilen zusammen.

6 Fazit und Ausblick

Es konnte aufgezeigt werden, dass eine Konvertierung von ISYBAU-XML-Daten nach IFC möglich ist, da das IFC-Schema auseichende Strukturen hierfür zur Verfügung stellt. Es wurden dabei jedoch teilweise benutzerdefinierte Objekte wie Property Sets oder Types definiert. Dies ist verständlich, da das ISYBAU-Schema ein sehr spezielles und fachspezifisches System ist und IFC nicht exakt für alle ISYBAU-Elemente ein Element mit gleicher Bezeichnung und gleichen Eigenschaften enthält. Dies verschlechtert die Qualität der IFC-Datei jedoch nicht. Im Gegenteil ist es als positiv anzusehen, dass IFC die Möglichkeit bietet über Benutzerdefinitionen den Informationsgehalt der ISYBAU-Daten aufzunehmen. Was die geometrische Abbildung betrifft kann das Knoten-Kantenmodell der ISYBAU-XML-Datei vollständig abgebildet werden. IFC bietet darüber hinaus die Möglichkeit, Abwassernetze auch über Volumenmodelle darzustellen. Hierbei hat sich jedoch ergeben, dass von Seiten des ISYBAU-Schemas nur vereinfachte geometrische Darstellungen sinnvoll sind, da der Informationsgehalt nicht auf eine detaillierte Abbildung ausgelegt ist. Andererseits wurde dargelegt, dass von Seiten des IFC-Schemas eine solche detaillierte Abbildung möglich ist. Dabei ist zu erwähnen, dass die hier gewählte Darstellungsweise als `IfcSectionedSolidHorizontal` zum Zeitpunkt dieser Arbeit bei vielen IFC-Viewern noch nicht vollständig implementiert ist. Es ist jedoch davon auszugehen, dass dies mit der Zeit geschehen wird, da die Darstellungsweise für linienförmige Infrastrukturbauwerke wie Brücken oder Straßen ausgelegt ist und somit in Zukunft ein wichtiger Bestandteil von BIM im Infrastrukturbau werden wird. Sollte die Modellierung und Abbildung von Kanalnetzten in Zukunft an Bedeutung gewinnen, wovon auszugehen ist, bedarf es Normen, die Festlegungen zur Darstellungsweise in IFC treffen. Dabei könnten bereits bestehende Festlegungen aus dem ISYBAU-Schema, die in dieser Arbeit als Ausgangslage definiert wurden, als Vorlage dienen. Als einfaches Beispiel ist dabei zu nennen, dass das Kurvenobjekt zur Beschreibung einer Haltung auf Höhe der Sohle verlaufen muss und nicht auf Achshöhe, da dies in der praktischen Anwendung der Planung und Bestandsvermessung die sinnvollere Angabe ist. Das ISYBAU-XML-Format selbst ist dabei wie in dieser Arbeit gezeigt nicht zwangsläufig ein Auslaufmodell. Stattdessen kann auch in Zukunft in Verbindung mit einem Konverter die bestehende Softwarelandschaft auf Basis von ISYBAU weiter genutzt werden. Die Chance des Schemas besteht in einer Erweiterung im Bereich der geometrischen Darstellung, um dem Anspruch einer Abbildung als Volumenmodell gerecht zu werden. Da in dieser Arbeit hauptsächlich die Konvertierung der ISYBAU-Stammdaten behandelt wurde, wäre es als Ausblick in die Zukunft interessant, auf den Stammdaten aufbauend auch Hydraulik- und Zustandsdaten nach IFC zu konvertieren.

7 Glossar

Abwasserkanal	„meist erdverlegte Rohrleitung oder andere Vorrichtung zur Ableitung von Abwasser aus mehreren Quellen“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 32)
Anschluss	„die Stelle, an der eine Rohrleitung mit einer anderen Rohrleitung bzw. einem Einstiegschacht oder einem Kontrollschacht zusammengeführt wird“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 38)
Ausgleichsring	„Bauteil, verwendet als Ausgleichskonstruktion“ (Deutsches Institut für Normen e.V., 2014, p. 34); Ausgleichkonstruktion: „Teil eines Bauwerkes, unmittelbar unterhalb von der Schachtabdeckung, verwendet, um die Höhe der Rahmen an die Straßen- bzw. Geländeoberfläche anzugleichen“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 34)
BIM	Building Information Modelling: Modellorientierte, digitale und kooperative Arbeitsweise in der Planung, dem Bau und Betrieb von Bautechnischen Anlagen.
Brep	Boundary Representation: Darstellungsweise eines Volumenkörpers im IFC-Schema über äußere Umschließungsflächen.
CSG	Constructive Solid Geometrie: Darstellungsweise eines Volumenkörpers im IFC-Schema über die Kombination von einfachen geometrischen Grundkörpern.
DLL-Datei	Dynamic Link Library: „Eine dll ist eine Bibliothek, die Code und Daten enthält, die von mehr als einem Programm gleichzeitig verwendet werden können.“ (Microsoft Corporation, 2020)
Entity	Bezeichnung für eine Klasse im IFC-Schema.

Haltung	„Abschnitt einer Abwasserleitung oder eines Abwasserkanals zwischen zwei angrenzenden Knotenpunkten“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 31)
IFC	Industry Foundation Classes: Standardisiertes Austauschformat zum Austausch von Bauwerksmodellen.
ISYBAU	Integriertes DV-System Bauwesen: Datenverarbeitungssystem zum Speichern und Austauschen von Informationen zu abwassertechnischen Anlagen.
Konus	„Teil eines Einstieg- oder Kontrollschachtes, bei dem sich die Querschnittsfläche allmählich verändert“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 37)
Leitung	Gleichzusetzten mit Abwasserleitung: „Meist erdverlegtes Rohr zur Ableitung von Abwasser von der Anfallstelle zum Abwasserkanal“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 30)
Property Set	Eine Ansammlung von spezifischen Eigenschaften, die zur Beschreibung eines Objektes in IFC genutzt wird.
Rohr	„Bestandteil einer Abwasserleitung oder eines Abwasserkanals, werkseitig hergestellt als Einzelstück, vorgesehen zur Verbindung mit anderen Rohren“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 31)
Rohrleitung	„Hohlstruktur zum Transport von Abwasser zwischen Einsteigschächten oder anderen Bauwerken“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 31)
Schachtabdeckung	„Oberer Teile eines Einstieg- oder Kontrollschachts, bestehend aus Deckel und tragendem Rahmen“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 35)
Schachtaufbau	„<Teile eines Einstieg- oder Kontrollschachts> Bauteil, das die untere Schachtzone mit der Oberfläche verbindet, um Zugang zu

	ermöglichen“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 37)
Schachtunterteil	„Unterteil eines Einstieg- oder Kontrollschachtes, das gegebenenfalls eingeformte Gerinne mit Auftritt umfasst und das den direkten Anschluss an erdverlegte Abwasserleitungen und -kanäle ermöglicht“ [DIN EN 16323 (Deutsches Institut für Normen e.V., 2014, p. 34)]
STEP Physical File	SPF oder auch nur STEP: Datenformat zum Austausch von Informationen. Meistverwendetes Datenformat zum Austausch von IFC-Daten.
Sweep	Darstellungsweise eines Volumenkörpers im IFC-Schema durch die Verschiebung eines Profils entlang einer Richtung, eines Pfads oder durch Rotation um eine Achse.
Übergangsplatte	„<Teil eines Einstieg- oder Kontrollschachts> horizontales Bauteil zum Abschluss eines Schachtringes mit Öffnung zur Aufnahme eines Schachtaufbaus“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 37)
Untere Schachzone	„Arbeitsraum innerhalb eines Einstieg- oder Kontrollschachts über dem Gerinne“ [DIN EN 16323] (Deutsches Institut für Normen e.V., 2014, p. 35)
XML	Extensible Markup Language: Ein textbasiertes Austauschformat zur Abbildung strukturierter Informationen. (World Wide Web Consortium, 2019)
XSD	XML-Schema Definition: Sprache zur Definition des Aufbaus des XML-Austauschformates.

8 Abbildungsverzeichnis

Abbildung 3-1: Aufgliederung der in ISYBAU abgebildeten Objektarten Knoten und Kante 4

Abbildung 3-2: Darstellung eines Abwassernetzes als Knoten-Kantenmodell in ++Systems 5

Abbildung 3-3: Abbildung der Schachtgeometrie eines Sonderschachtes durch einen geschlossenen Polygonzug über die Punkte mit den Punktattributen „SBW“. Die Punkte „RAP“ markieren den Rohranschlusspunkt eines Zu- oder Ablaufs. „SMP“ (Schachtmittelpunkt) und „DMP“ (Deckelmittelpunkt) sind weitere Punkte zur Abbildung des Schachtes. (Bundesministerium des Inneren für, Bau und Heimat, 2019, p. 71) 6

Abbildung 4-1: Baumförmiger Aufbau des XML-Schemas, ausgehend von einem Root-Element 7

Abbildung 4-2: .xsd-Dateien zur Beschreibung des ISYBAU-XML-Schemas 9

Abbildung 4-3: Strukturierung des ISYBAU-Austauschformates 10

Abbildung 4-4: Zugriffsrechte von externe Klassen auf die Eigenschaften der automatisch generierten ISYBAU-Klassen. Die Methode der externen Klasse hat nur zugriff auf die `public` Eigenschaft `Version`. Über `get` und `set` interagiert diese mit `privateversionField.` 25

Abbildung 4-5: Navigation durch die ISYBAU Baumstruktur anhand des Identifikation Objektes „`Austauschdatei`“ 29

Abbildung 5-1 Property Set `Pset_WallCommon` und die darin enthaltenen Eigenschaften (buildingSMART International Limited, 2020) 31

Abbildung 5-2 Vererbungshistorie der Entity `IfcWall` 33

Abbildung 5-3: Layer Aufteilung des IFC-Schemas 34

Abbildung 5-4 Referenzieren der Entity `IfcProductRepresentation` durch das Attribut `Representation` der Entity `IfcWall` 35

Abbildung 5-5 Schematische Darstellung einer Beziehung zwischen zwei Objekten über ein Beziehungsobjekt in IFC 35

Abbildung 5-6 Zuweisen einer Materialeigenschaft zu einer Wand 36

Abbildung 5-7: `IfcRepresentationItem` und seine wichtigsten Subtypes, die zur geometrischen Darstellung genutzt werden können 39

Abbildung 5-8: Übersicht über die vorhandenen Entities zur Darstellung von Volumenkörpern 40

Abbildung 5-9: `IfcCsgPrimitive3D` und seine Subtypes zur Abbildung der geometrischen Grundfiguren Würfel, Pyramide, Kegel, Zylinder und Kugel 40

Abbildung 5-10: Exemplarische Abbildung der booleschen Operatoren Union, Difference und Intersection 41

Abbildung 5-11: Die Schritte zur Darstellung eines Volumenkörpers mit Boundary Representation. Die Eckpunkte werden zunächst durch Linien verbunden, die sich wiederum zu Randlinien einer Fläche zusammensetzen. Mehrere Flächen bilden die Körperoberfläche und ermöglichen so die Definition eines Volumenkörpers. 41

Abbildung 5-12: Übersicht über Programmaufbau und Ablauf des Konverters. 44

Abbildung 5-13: Programmablauf in der isybau_to_ifc.cs Datei zum Erzeugen der IFC-Austauschdatei über die Methoden `ToIfcInstance` auf Grundlage der eingelesenen ISYBAU-Inforamtionen 48

Abbildung 5-14: Schematische Darstellung der Type Zuordnung am Beispiel des Kantenobjekts Haltung 51

Abbildung 5-15: Schematische Darstellung vom Aufbau von `IfcPropertySet` 52

Abbildung 5-16: Herstellen einer Beziehung zwischen `IfcDistributionPort` und `IfcPropertySet` über das Beziehungsobjekt `IfcRelDefinesByProperties` 53

Abbildung 5-17: Schachtaufbau in Anlehnung an DIN EN 13508-2 Bild 1 (Deutsches Institut für Normen e.V., 2011, p. 12) mit Angaben der Schachtinformationen in ISYBAU 55

Abbildung 5-18: Definition eines Profils (blau) in der x-y-Ebene. Die z-Achse zeigt in die Ebene hinein. 56

Abbildung 5-19: Unterschiedliche Verfahren zum Ausrichten des lokale Koordinatenssystems der Proftle: Senkrecht zum Pfad (links), alle Profile orientieren sich an einer Ausrichtung (mitte) oder feste Ausrichtung am globalen Koordinatensystem (rechts) 56

Abbildung 5-20: Exemplarische Aufbau eines Profils zur Erzeugung eines Rohrsegmentes über das Rotationsverfahren (Bock & Michaelis, 2019, p. 39) 59

Abbildung 5-21: Zusammensetzung von `IfcIndexedPolyCurve` (buildingSMART International Limited, 2020)) 61

Abbildung 5-22: Ermittlung des Punktes P3 auf einem Kreissegment mithilfe der gegebenen Punkte Start, Mitte und Ende 61

Abbildung 5-23: Kreisförmiger Rohrquerschnitt, beschrieben durch eine äußere und innere Kurve. 62

Abbildung 5-24: Unterschied zwischen Haltungs- und Rohrlänge 63

Abbildung 5-25: Abbildung als `IfcSectionedSolidHorizontal`, einmal mit FixedAxisVertical = true (links) und FixedAxisVertical=false (rechts) 64

Abbildung 5-26: Abbildung eines Rohrsegments mit Muffengeometrie durch Stationierung variierender Rohrprofile 65

Abbildung 5-27: Darstellung eines Rohrsegmentes mit Muffe über `IfcSectionedSolidHorizontal` in FZKViewer 66

Abbildung 5-28: Attribute von `IfcSectionedSpine` 66

Abbildung 5-29: Zusammensetzung von `IfcPolyline` (buildingSMART International Limited, 2020) ... 67

Abbildung 5-30: Zusammensetzung von `IfcCompositeCurve` (buildingSMART International Limited, 2020) 67

Abbildung 5-31: Die drei unterschiedlichen Darstellungsweisen eines Schachts als Kantenmodell in der Draufsicht (links), Zylindermodell (mitte) und erweiterte Darstellung (rechts). 68

Abbildung 5-32: Zuordnung der geometrischen Darstellung zu einem Bauteil 70

Abbildung 5-33: Unterteilung von Knoten- und Kantenobjekten bei der erweiterten Darstellung. 71

Abbildung 5-34: ISYBAU-Datei eingelesen 72

Abbildung 5-35: Abfrage zur Darstellungsweise 72

Abbildung 5-36: keine ausreichenden Informationen vorhanden 72

Abbildung 5-37: Konvertierung abgeschlossen 73

Abbildung 5-38: Darstellung der konvertierten ISYBAU-Daten im FZKViewer als Zylindermodell (links) und Knoten- Kantenmodell (rechts) 73

Abbildung 5-39: Erweiterte Darstellung der konvertierten ISYBAU-Daten im FZKViewer. Schacht und Haltung setzen sich dabei aus mehreren Bauteilen zusammen..... 74

9 Tabellenverzeichnis

Tabelle 4-1: Beispiel zur Definition eines Elementes über einen `simpleType` 12

Tabelle 4-2: Definitionen laut BFR zu den im ISYBAU-Verwendeten simpleTypes (Bundesministerium des Inneren für, Bau und Heimat, 2019, pp. 676, 677) 13

Tabelle 4-3: Beispiel zur Definition eines Elementes über einen `complexType` 14

Tabelle 4-4: Definition des `complexTypes StammAuftragType` 14

Tabelle 4-5: Implementierung eines choice-Elements am Beispiel von `AbwassertechnischeAnlageType`. 16

Tabelle 4-6: Festlegen der Namespaces 16

Tabelle 4-7: Definition des `simpleTypes gYearMonth` in der xsdschema.xsd-Datei 17

Tabelle 4-8: Veranschaulichung der Funktionsweise von Präfix durch Ergänzung eines Präfixes „`isybau`“ 17

Tabelle 4-9: Auswirkungen von `elementFormDefault="qualified",` veranschaulicht durch die Ergänzung des Präfixes „`isybau`“ 18

Tabelle 4-10:Einbinden der Schemadateien in die 1707-metadaten.xsd-Datei 18

Tabelle 4-11: Beispiel zum Bezug eines Elements zu einer enumeration einer Referenzliste 19

Tabelle 4-12: Definition von Schlüsseln am Beispiel des Elementes `Datenkollektive` 20

Tabelle 4-13: Exemplarischer Aufbau einer Klasse in C# nach dem Vorbild des `complexTypes Identifikation` aus dem ISYBAU-Schema 22

Tabelle 4-14: Auszug aus den durch die Xsd.exe aus den ISYBAU XSD-Dateien automatisch erzeugten Klassen. Der hier abgebildete Auszug enthält die definition der Klasse `Identifikation`. 24

Tabelle 4-15: Beispiel für eine automatisch generierte enumeration aus einer ISYBAU Referenzliste mit vorgegebenen integer Werten. 26

Tabelle 4-16: Beispiel für eine automatisch generierte enumeration aus einer ISYBAU Referenzliste mit vorgegebenen string Werten. 27

Tabelle 4-17: Beispiel für eine automatisch generierte Klasse mit einem choice Element im ISYBAU Schema. 28

Tabelle 5-1 Vergleich der Definition einer IFC-Entity im XSD und EXPRESS-Schema 32

Tabelle 5-2: Auszug aus dem offiziellen Beispiel wall-with-opening-and-window.fic von buildingSmart.37

Tabelle 5-3: Attribute von `IfcWallStandardCase` 38

Tabelle 5-4: Aufgliederung der konkreten Attributwerte einer Instanz von `IfcWallStandardCase` aus #38 des Beispiels in Tabelle 5-2 38

Tabelle 5-5: Eine Übersicht über die im Sweep-Verfahren enthaltenen Darstellungsweisen Extrusion, Sweep und Rotation 42

Tabelle 5-6: Auszug aus der partial class `AbwassertechnischeAnlageType` mit Definition der Methode `ToIfcInstance` 45

Tabelle 5-7: Erzeugen von IFC-Objekten mit Hilfe der IfcSharp-Library 46

Tabelle 5-8: Attribute von `IfcPipeSegment` 49

Tabelle 5-9: Instanziierung von `IfcPipeSegment` und zugehöriger Type Definition 50

Tabelle 5-10: Attribute von `IfcPipeSegmentType` 51

Tabelle 5-11: Attribute von `IfcDistributionPort` 53

Tabelle 5-12: Vorschlag geeigneter IFC Entities zur Abbildung von ISYBAU-Bauwerken 54

Tabelle 5-13: Übersicht über die verschiedenen Darstellungsweisen des Sweep-Verfahrens und Bewertung hinsichtlich der Nutzung als Darstellungsweise von abwassertechnischen Anlagen 60

Tabelle 5-14: Attribute von `IfcSectionedSolidHorizontal` 60

10 Literaturverzeichnis

Deutsches Institut für Normen e.V., 2011. *Untersuchung und Beurteilung von Entwässerungssystemen außerhalb von Gebäuden - Teil2: Kodiersystem für die optische Inspektion.* Berlin: Beuth Verlag GmbH.

<edition W3>, 2020. *XML Schema Teil 0: Einführung.* [Online] Available at: http://www.edition-w3.de/TR/2001/REC-xmlschema-0-20010502/ [Zugriff am 03 Mai 2020].

Bock, B. & Michaelis, E., 2019. Building Information Modeling in der Abwasserableitung mit opneBIM. *Wasser und Abfall*, Mai.

Borrmann , A., König , M., Christian, K. & Beez, J., 2015. *Buildin Information Modeling - Technologische Grundlagen und industrielle Praxis.* Wiesbaden: Springer Vieweg.

buildingSMART International Limited, 2020. *IFC Formats - buildingSMART Technical.* [Online] Available at: https://technical.buildingsmart.org/standards/ifc/ifc-formats/ [Zugriff am 16 März 2020].

buildingSMART International Limited, 2020. *IFC4 Documentation.* [Online] Available at: https://standards.buildingsmart.org/IFC/RELEASE/IFC4_1/FINAL/HTML/ [Zugriff am 16 März 2020].

Bundesministerium des Inneren für, Bau und Heimat, 2019. *Baufachliche Richtlinien Abwasser,* Berlin: s.n.

Deutsches Institut für Normen e.V., 2014. *DIN EN 16323 Wörterbuch für Begriffe der Abwassertechnik.* Berlin: Beuth Verlag GmbH.

Microsoft Corporation, 2020. *Dynamic Link Library (dll) - Windows Client | Microsoft Docs.* [Online] Available at: https://docs.microsoft.com/de-de/troubleshoot/windows-client/deployment/dynamic-link-library [Zugriff am 29 November 2020].

Microsoft Corporation, 2020. *XML Schema Definition-Tool (Xsd.exe) | Microsoft Docs.* [Online] Available at: https://docs.microsoft.com/de-de/dotnet/standard/serialization/xml-schema-definition-tool-xsd-exe [Zugriff am 15 September 2020].

Microsoft Corporation, 2020. *XmlSerializer Klasse (System.Xml.Serialization) | Microsoft Docs.* [Online] Available at: https://docs.microsoft.com/de-

de/dotnet/api/system.xml.serialization.xmlserializer?view=netcore-3.1
[Zugriff am 15 September 2020].

World Wide Web Consortium, 2019. *HTML & CSS - W3C*. [Online] Available at: https://www.w3.org/standards/webdesign/htmlcss [Zugriff am 02 März 2020].

World Wide Web Consortium, 2019. *Schema - W3C*. [Online] Available at: https://www.w3.org/standards/xml/schema [Zugriff am 02 März 2020].

World Wide Web Consortium, 2019. *XML Essentials - W3C*. [Online] Available at: https://www.w3.org/standards/xml/core [Zugriff am 02 März 2020].

World Wide Web Consortium, 2020. *W3C Deutschland/Oesterreich » Über das W3C*. [Online] Available at: https://www.w3c.de/about/ [Zugriff am 02 März 2020].

11 Verwendete Programme

Bei der Erstellung dieser Arbeit wurden Folgende Programme verwendet:

++Systems
Version 12.00.00

FZKViewer
Build 1686

Microsoft Visio Plan 2 MSO
Version 16.0.12527.21230

Microsoft Visual Studio Community 2019
Version 16.4.5

Microsoft Word
Version 16.0.12527.21230

Siemens NX
Version 1855

Snipping Tool
Version 10.0.19041