TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Bauingenieurwesen Institut für Bauinformatik

Modellierung und Verarbeitung der Bewehrungsdaten von Brückenmodellen in IFC

Projektarbeit

vorgelegt von

Yixuan Zeng

Verantwortlicher Hochschullehrer

Prof. Dr. Ing. Raimar J. Scherer

Betreuer

Dipl.-Ing. Al-Hakam Hamdan

Dresden, 23.12.2018

Aufgabenstellung

Thema:

Modellierung und Verarbeitung der Bewehrungsdaten von Brückenmodellen in IFC

Zielsetzung:

Im Bereich des Building Information Modelling (BIM) hat sich zur Beschreibung von Gebäudemodellen der offene Standard der Industry Foundation Classes (IFC) etabliert. Allerdings unterstützt IFC momentan noch nicht die Modellierung von Brückenbauwerken, weswegen deren Bauteile nur mittels Proxy-Elementen in einem BIM-Modell erstellt werden können. Ein wichtiger Aspekt ist hierbei, neben der Erstellung grundlegender Komponenten, wie dem Über- oder Unterbau einer Brücke, auch die Bewehrung, deren Modellierung in IFC ebenfalls noch nicht für Brücken ausgelegt ist. Hierbei existeiren für Bewehrungsstäbe und –matten bereits Entitätstypen in IFC, allerdings noch nicht für Spannstahlelemente.

Ziel der Projektarbeit ist die Entwicklung eines Konzeptes zur Modellierung von Bewehrungs-, sowie Spannstahl-Komponenten in proxybasierten Brückenmodellen. Darauf aufbauend sollen Methoden erarbeitet werden, die eine Selektion dieser Daten mittels Filteralgorithmen und eine anschließende Datenverarbeitung ermöglichen. Hierbei sollen insbesondere der Bewehrungsgrad der Bauteile sowie der Spanngliedverlauf automatisiert ermittelt werden.

Arbeitsumfang:

Die Arbeit umfasst die folgenden Schritte:

1. Untersuchung der verschiedenen Möglichkeiten Bewehrung und Spannstahlkomponenten in IFC zu modellieren.
2. Erarbeitung eines Konzeptes zur Implementierung von Elementen, die eine Repräsentation der verschiedenen Bewehrungstypen unterstützen.
3. Beispielhafte Umsetzung des erarbeiteten Konzeptes aus Schritt 2 durch Implementierung von Bewehrungsdaten in ein gestelltes IFC-Brückenmodell.
4. Entwicklung von Methoden, die die Bewehrungselemente aus dem Brückenmodell durch Filterung selektieren.
5. Entwicklung von Algorithmen, die die gefilterten Datensätze verarbeiten und dadurch tragwerksrelevante Informationen ermitteln, wie Bewehrungsgrad und Spanngliedverlauf.
6. Nachvollziehbare Dokumentation der Entwicklungsschritte und der Ergebnisse in einem schriftlichen Bericht.
7. Aufbereitung und öffentliche Präsentation der Ergebnisse.

Ehrenwörtliche Erklärung

Hiermit erkläre ich, dass ich die Projektarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht habe.

Die Projektarbeit habe ich bisher keiner anderen Institution in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher auch nicht veröffentlicht.

_____

Ort, Datum

_____

Unterschrift

Einleitung

Mit der Entwicklung und Anwendung von Building Information Modeling (BIM) werden die Planungs- und Ausführungsqualität, Kosten, Koordination, sowie der Zeitaufwand von Bauprojekten optimiert. Um die Kollaboration unterschiedlicher Fachplaner mit einem standardisierten Datenaustauschformat zu verbessern, ist ein offenes, herstellerunabhängiges, von buildingSMART auf der Datenmodellierungssprache EXPRESS entwickeltes Datenformat namens IFC, bzw. Industry Foundation Classes entwickelt worden.

Im IFC-Standard werden die Elemente des Gebäudemodells mittels äquivalenter Klassen erstellt, jedoch unterstützt IFC-Standard gegenwärtig noch nicht die Modellierung von Brückenbauwerken. Aus diesem Grund können deren Bauteile ausschließlich mittels Proxy-Elementen in einem BIM-Modell erstellt werden. Ein wichtiger Aspekt ist hierbei, neben der Erstellung grundlegender Komponenten, wie dem Über- oder Unterbau einer Brücke, auch die Bewehrung sowie die Spannglieder, deren Modellierung in IFC-Standard ebenfalls noch nicht für Brücken ausgelegt ist. Hierbei bezieht IFC-Standard für Bewehrungsstäbe und -matten, und Spannglieder bereits Entitätstypen ein, die aber keine Universalität für alle Bewehrungen und Spannglieder haben.

Im Projekt wird zum Ersten die grundlegenden Informationen von BIM, IFC und Bewehrung bzw. Spannglieder im Brückenbau recherchiert, und die verschiedenen Möglichkeiten der Modellierung der Bewehrung und Spannstahlkomponenten in IFC-Standard untersucht. In der nächsten Phase wurde ein in der BIM-Modellierungssoftware Revit erstelltes Brückenmodell der Pavel-Wonka Brücke in Pardubice, das im IFC-Modell mit Proxy-Elementen erstellt und exportiert wurde, zur Verfügung gestellt. Nach vorhanden Informationen werden im proxybasierten Brückenmodell Bewehrung und Spannglieder mittels BIM-Software Revit modelliert und ins IFC-Modell exportiert. Darauf aufbauend werden Methoden erarbeitet, die eine Selektion dieser Daten mittels Filteralgorithmen und eine anschließende Datenverarbeitung ermöglichen. Hierbei werden insbesondere der Bewehrungsgrad der Bauteile sowie der Spanngliedverlauf ermittelt und die Probleme und Schwerpunkte, die nicht gelöst werden können, analysiert, auswertet und theoretisch optimiert. Aus dem Projekt wird dann ein Blick relevanter Entwicklung in die Zukunft gerichtet.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AL Anwendungsebene

BIM Building Information Modeling

CAD Computer Aided Design

DL Domäne Ebene

ID Identifikator

IFC Industry Foundation Classes

NL Neutrale Ebene

NLa Algorithmische und Schlussfolgerungsfunktion

NLm Metafunktion der neutralen Ebene

NLs Semantische (Kern-) Funktion der neutralen Ebene

URL Uniform Resource Locator

1 Grundlagen

1.1 Building Information Modeling

Building Information Modeling (BIM) ist eine Methode, die zu der Optimierung der Planungs- und Ausführungsqualität, der Beschleunigung der Bauprojekten mit weniger Koordinierungsaufwand und Kosten dient. BIM kann sowohl im Bauwesen zur Bauplanung und Bauausführung als auch im Baubetrieb und -management angewendet werden. Mittels BIM-Software wird ein Bauwerk unter verschiedenen Aspekten digital modelliert und kombiniert. [1]

Das Konzept von BIM kann als Prozess definiert werden, der mit der Erzeugung einer 3D-Darstellung eines Objekts mit besonderen Eigenschaften beginnt, mit dem Aufbau von Beziehungen zwischen den verschiedenen Komponenten des Modells fortfährt und mit der Verwaltung und Angemessenheit des Modells an die Anforderungen der verschiedenen am Projekt beteiligten Instanzen endet.

BIM ist jedoch mehr als ein 3D-Modell. BIM bietet viele Funktionen, die zur Modellierung des Lebenszyklus (siehe Abbildung 1.1) eines Tragwerks erforderlich sind, und bietet die Grundlage für neue Entwurf-, Bauplanungs-, und Bauführungsfähigkeiten sowie für die Änderung der Rollen und Beziehungen zwischen einem Projektteam. BIM impliziert eine gesamte Prozesskette, die alle Phasen des Bauprozesses integriert. Während der Entwicklung dieser Prozesskette entsteht ein kontinuierlicher Fluss digitaler Informationen, der zu einer gemeinsamen virtuellen Plattform für alle Personen führt, die mit der Planung, dem Entwurf, dem Bau und dem Betrieb des Tragwerks in Verbindung stehen. Ziel dieses ständigen Informationsaustauschs ist einerseits eine bessere Zusammenarbeit zwischen den Projektbeteiligten und andererseits die Vermeidung der unnötigen Eingabe und Berechnung von Daten wie Flächen, Volumina und Materialeigenschaften.

Abbildung 1.1: BIM-Lebenszyklus

Im klassischen Bauprozess ist die Kommunikation und Koordinierung mit großem Aufwand verbunden (Abbildung 1.2). In jeder Phase (Bauplanung, Bauausführung und Baubetrieb usw.), hat jeder Beteiligter eigene Perspektiven, Pläne und Modelle. Eine Abgleichung bzw. Änderung zwischen verschiedenen Daten ist deshalb zeitaufwändig, ineffizient und sogar fehleranfällig.

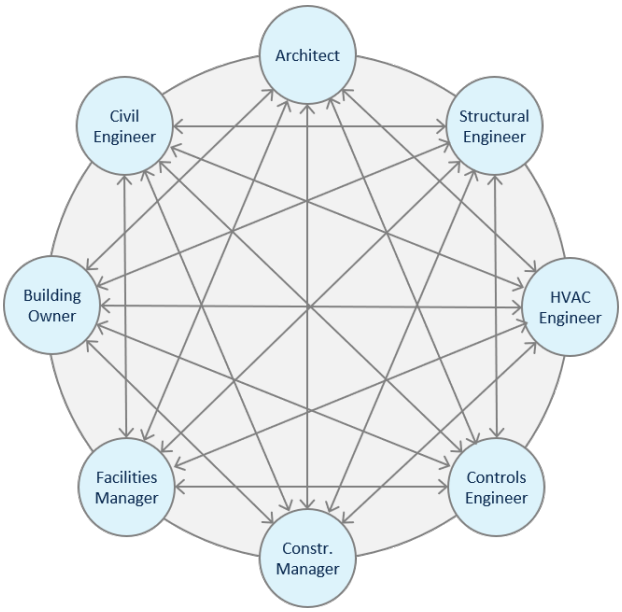


Abbildung 1.2: Klassische Modellübersicht [2]

Im Vergleich dazu, ist das wichtigste Konzept in BIM, dass jedes Element des Modells nicht mehr nur eine geometrische Darstellung einer physischen Entität ist, sondern ein virtuelles Objekt mit Parametern, das es zu definieren gilt. Diese Parameter können von geometrischen Informationen, räumlichen und mechanischen Eigenschaften bis zu Bautechnik und Kosten einbeziehen. Das Endergebnis eines BIM-Prozesses ist ein objektorientiertes Modell, das während des gesamten Lebenszyklus des Bauwerks einschließlich seines Abrisses von verschiedenen Fachleuten verwendet werden kann, und durch das seine Informationen und Wissen zwischen verschiedenen Arbeitsteams ausgetauscht werden kann.[3] Damit wird die Qualität der Daten verbessert und gleichzeitig Kosten und Zeitaufwand gespart. (siehe Abbildung 1.3)

1.2 Industry Foundation Classes

Die Interoperabilität von BIM bezieht sich auf die Fähigkeit der BIM-Systeme, Informationen zwischen den Projektteilnehmern auszutauschen und zu aktualisieren. Die Verwendung offener Schnittstellen ist im BIM-Prozess von grundlegender Bedeutung. Dieses offene Konzept, nämlich das Open BIM Konzept beruht auf dem offenen Austausch der Bauwerksdaten jedoch unabhängig von Softwares jeweiliger Beteiligten bzw. Arbeitsteams. Es ermöglicht eine fließende Kommunikation und Anpassung des Modells. Um das Problem des Datenaustausches im Bauprozesses zu lösen, ist die Einführung der Industry Foundation Classes (IFC) ein großer Schritt in der Entwicklung von Open BIM. IFC-Standard ist ein offenes, herstellerunabhängiges, auf der Datenmodellierungssprache EXPRESS entwickeltes Ausgabeformat von buildingSMART für einen

Datenaustausch im Bauwesen, um das Open BIM Konzept zu realisieren und die Kollaboration unterschiedlicher Fachplaner zu verbessern.[4] IFC-Standard beschreibt nicht nur die räumliche Geometrie sondern auch die relevanten Eigenschaften jedes im Modell einbezogenen Objektes. Alle Modelldaten können auf dieser gemeinsamen Datenbank zurückgehen und ständig synchronisiert werden, dadurch die Interoperabilität von der Planung und Ausführung bis in die Betriebsphase optimiert und Effizienz und Effektivität erhört werden sollen (Abbildung 1.3). IFC-Standard stellt auch eine Lösung der hohen Kostenprobleme dar, die die gemeinsame Nutzung der Daten behindern.

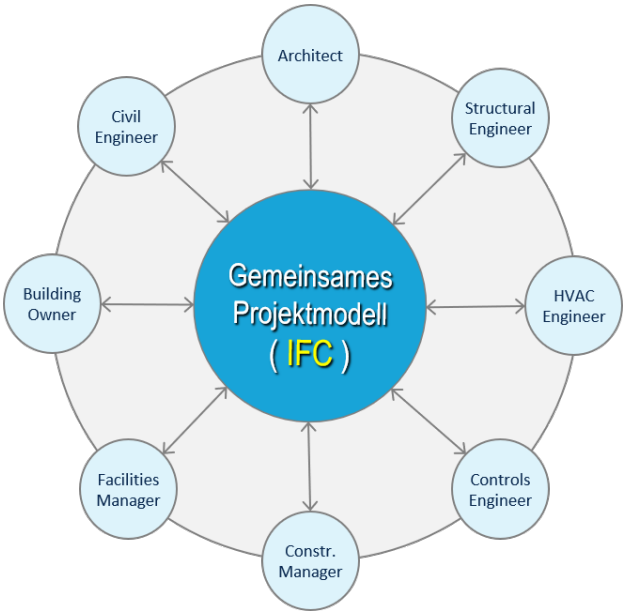


Abbildung 1.3: IFC-Projektmodell [2]

1.2.1 Datenstruktur

IFC-Standard ist in einer Menge von Schemata hierarchisch geordnet, die nach ihrem Inhalt zu einem Layer gehören. Der Inhalt jedes Schemas spezifiziert einen bestimmten Faktor, z.B. Geometrie, Zeit, Materialeigenschaft, Kosten, Bemessungen usw. Alle Schemata bilden sich in insgesamt vier Layers: Resource Layer, Core Layer und Core Extention Layer, Interoperability Layer, und Domain/Aplplication Layer.

- Resource Layer

Der unterste Layer, der alle individuellen Schemata der Ressourcendefinitionen enthält. Die Definitionen enthalten keine global eindeutige Kennung und sollen unabhängig von einer Definition auf einem höheren Layer verwenden werden. Zu diesem Layer gehören die Ressourcen von Material, Geometrie, Kosten, Topologie usw.

- Core Layer, Core Extension Layer

Der nächste Layer, der Kernel-Schemata und Extension-Schemata enthält und die allgemeine Entitätsdefinitionen enthält. Alle Entitäten werden im Kernel Layer definiert, oder tragen ein global eindeutige ID und optional Inhaber und Verlaufsinformationen. Dieser Layer bezieht Schemata von Kernel und Extension von Kontrolle, Produkt und Prozess ein.

- Interoperability Layer

Der nächste Layer, der Schemata einschließlich Entitätsdefinitionen enthält, die zu einem allgemeinen Produkt-, Prozess- oder Ressourcenspezialisierung spezifisch sind, verwendet in mehreren Disziplinen. Die Definitionen sind typisch für Inter-Domain Austausch und die gemeinsame Nutzung der Konstruktionsinformationen zu verwenden. In diesem Layer stehen alle Elemente, z.B. Shared Component Elements, Shared Management Elements, Shared Building Elements usw.

- Domain/Application Layer

Der oberste Layer, der Schemata der Entitätsdefinitionen enthält, die Spezialisierungen des Produktes, Prozesses, oder der Ressourcen spezifisch für eine bestimmte Disziplin sind. Die Definitionen sind typisch für Intra-Domain Austausch und die gemeinsame Nutzung der Informationen zu verwenden. Zum Layer gehören z.B. strukturelle Analyse, Baumanagement, Architektur usw.

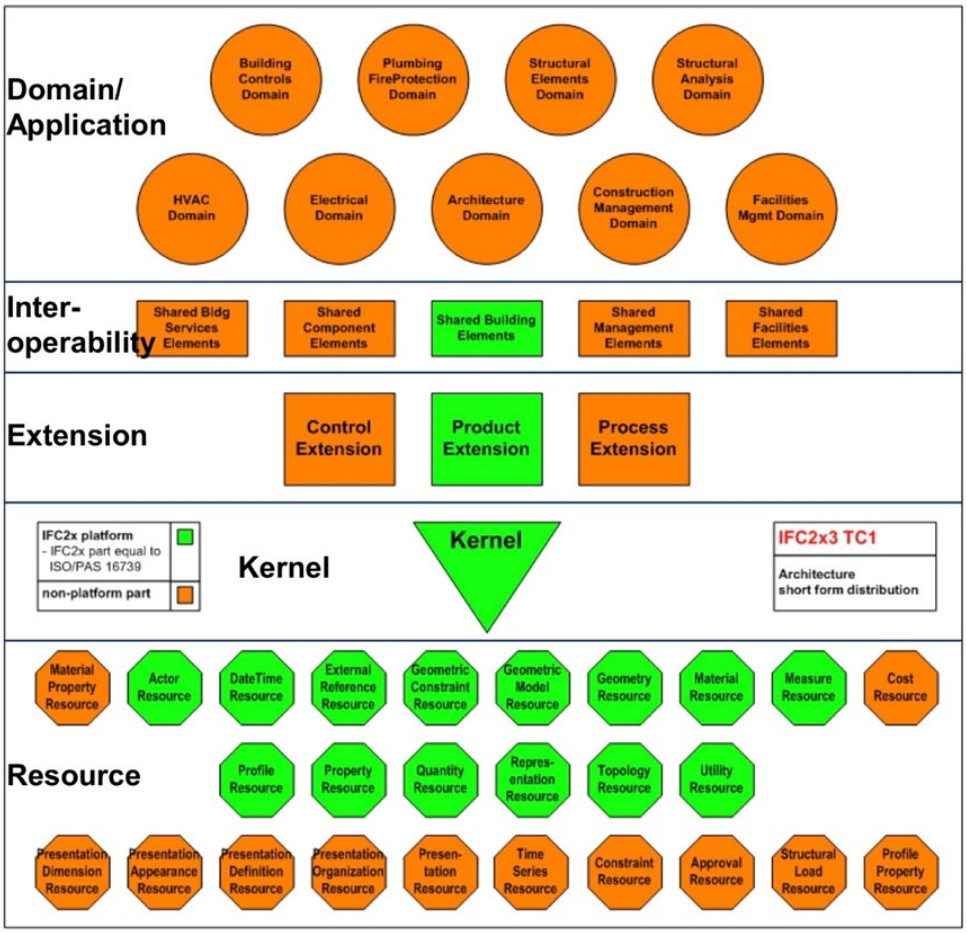


Abbildung 1.4: die in Layers geordneten Schemata von IFC [6]

1.2.2 Basis-Klassen

Schemata in IFC sind hierarchisch geordnet. Jede Klasse kann eine Klasse desselben oder eines tieferen Layers referenzieren, aber keine Klasse eines höheren Layers.

Referenzen sind nur innerhalb desselben Layers und von jedem Layer zu den Ressourcen-Layer erlaubt.

Bei der Hierarchie der Objektinstanzen sind alle identifizierbaren Objekte von der *IfcRoot*-Klasse abgeleitet. Jedes Objekt hat eine GlobalID, die sich während des gesamten Lebenszyklus niemals ändert. Die Hierarchie der Basis-Klassen und deren Subklassen bildet sich wie in Abbildung 1.5.

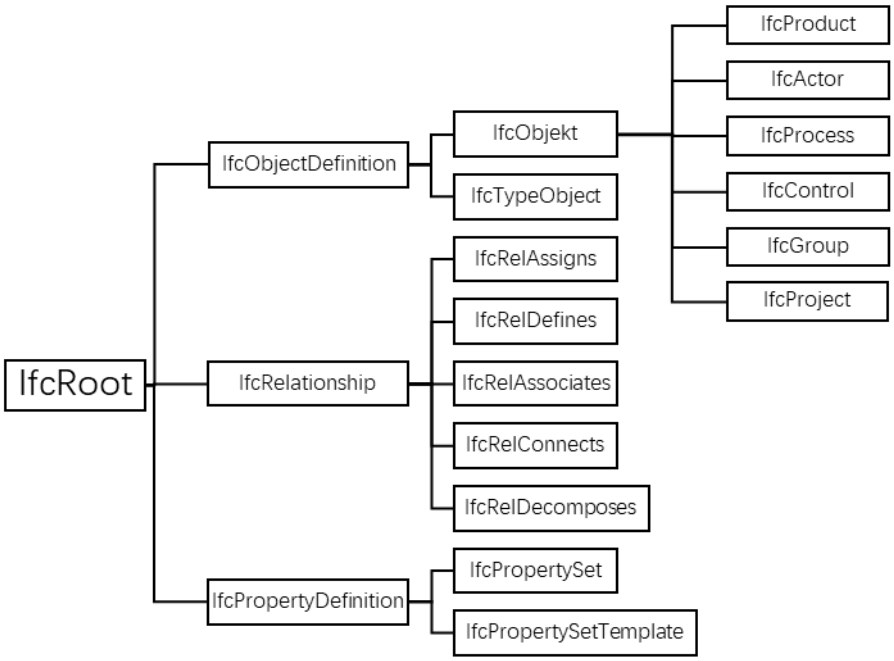


Abbildung 1.5: Basis-Klassen und deren Subklassen in IFC

1.2.3 Objekttypen

Das IFC-Schema beinhaltet sieben grundlegende Objekttypen: Produkte, Akteure, Prozesse, Steuerung, Gruppe, Projekt und Ressourcen.

Die Produkte *IfcProdukt* stellen materielle Objekte dar, die ein Teil eines Projektes sind. Eine spezielle Subklasse davon ist *IfcProxy*, die eine Art von Container für umschließende Objekten sind, die durch zugeordnete Eigenschaften definiert werden und eine geometrische Repräsentation und Platzierung im Raum aufweisen können oder nicht. [6] *IfcProxy* ist für nicht im IFC-Datenmodell definierte Objekte verwendbar damit das Modell beliebig erweitert werden. Die Akteure *IfcActor* stellen die im Projekt aktiven Personen bzw. Organisationen dar und die Gruppe *IfcGroup* ist eine frei wählbare Gruppierung von Objekten. Die Steuerung *IfcControl* beschreibt Konzepte, die Produkten bzw. Prozessen steuern bzw. beschränken können. Die Ressourcen *IfcRessource* beinhalten die Informationen um die Kosten, den Zeitaufwand zu beschreiben. Als eine Beschreibung die die Aktivitäten wie bspw. Ausschreibung, Bauausführung, verbinden die Prozesse *IfcProcess* Objekte, auf denen der Prozess arbeitet als Input oder Output. Zu *IfcProzess* können Ressourcen zugeordnet werden. Und das Projekt *IfcProject* ist die Durchführung einiger Planungs-, Konstruktions-, bzw. Wartungsaktivitäten, die zu einem Produkt

führen. Das Projekt legt den Kontext für den Austausch oder die Weitergabe von Informationen fest und kann ein Bauprojekt darstellen, muss dies jedoch nicht.

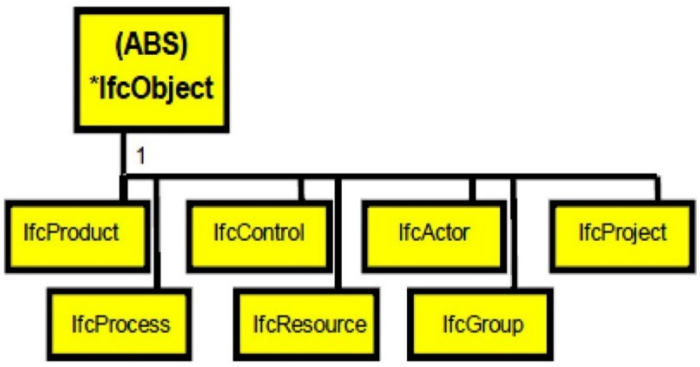


Abbildung 1.6: Objekttypen in IFC [6]

1.2.4 Beziehungen

Das IFC-Schema enthält fünf grundlegende Beziehungstypen: *IfcRelAssigns, IfcRelDefines, IfcRelAssociates, IfcRelConnects, IfcRelDecomposes*.

IfcRelAssigns ist die allgemeine Form einer Objektbeziehung in IFC. Sie ermöglicht einem Klient-Objekt die Dienste anderer Objekte anzuwenden. Sie erlaubt Beziehungen zwischen allen sieben Objekttypen zu erzeugen, die Subklassen von *IfcObject* sind. *IfcRelDefines* verwendet eine Typdefinition oder eine Eigenschaftssatzdefinition um die Eigenschaften der Objektinstanz zu definieren. Es ist eine spezifische vorkommende Beziehung mit implizierten Abhängigkeiten. *IfcRelAssociates* erlaubt Beziehungen zwischen Objekten und externen Dokumenten, einer oder mehrerer Klassifikationen, bzw. Bibliotheken. *IfcRelConnects* definiert eine Verbindung zwischen Objekten, die physikalisch oder logisch sein können. Als allgemeine Konnektivität impliziert dies keine Einschränkungen, jedoch definieren die Subtypen der Beziehung die anwendbaren Objekttypen für *IFcRelConnects* und die Semantik der jeweiligen Konnektivität. *IfcRelDecomposes* definiert eine „Teil-von-Hierarchie“ bzw. Aufbau. Durch Verschachtelung zerlegt sie die Objekte derselben Klasse und durch Aggregation die Objekte unterschiedlicher Klassen.

1.2.5 Geometriekonzept

Das Geometriekonzept in IFC-Standard bezieht eine Vielzahl von Geometrie-Klassen für bspw. Volumenmodelle, Oberflächenmodelle, Kurven, Profile usw. und Topologie-Klassen für Hüllen, 2-mannigfltige Oberflächen, Schleifen, Knoten usw. ein. Die Geometrie kann in IFC-Standard in mehreren Formen dargestellt werden, und zwar über ihre geometrische, topologische, oder eine vordefinierte Repräsentation, bzw. seit *IFC2x2* mögliche zusätzliche grafische Präsentation.

IfcGeometricRepresentationItem stellt ein geometrisches Repräsentationselement dar, das die Bedeutung hat, eine geometrische Position, Orientierung bwz. beides zu haben. Diese Bedeutung ist dargestellt, da sie ein kartesischer Punkt oder eine Richtung ist,

oder sie einen kartesischen Punkt oder eine Richtung in- bzw. direkt referenziert. Ein indirektes Referenzieren auf einen kartesischen Punkt oder eine kartesische Richtung bedeutet, dass ein gegebenes geometrisches Element durch ein bzw. mehrere dazwischen liegende geometrische- oder topologische Elemente den kartesischen Punkt oder die kartesische Richtung referenziert. Ein dreidimensionaler kartesicher Punkt könnte geometrisch durch einen Subtyp *IfcPolyline* präsentiert. *IfcPolyline* stellt eine begrenzte Kurve von n - 1 linearen Segmenten dar, die durch eine Liste von n Punkten definiert wird. Die kartesischen Punkte werden auch durch *IfcCartesianPoint* ausgedrückt. [6]

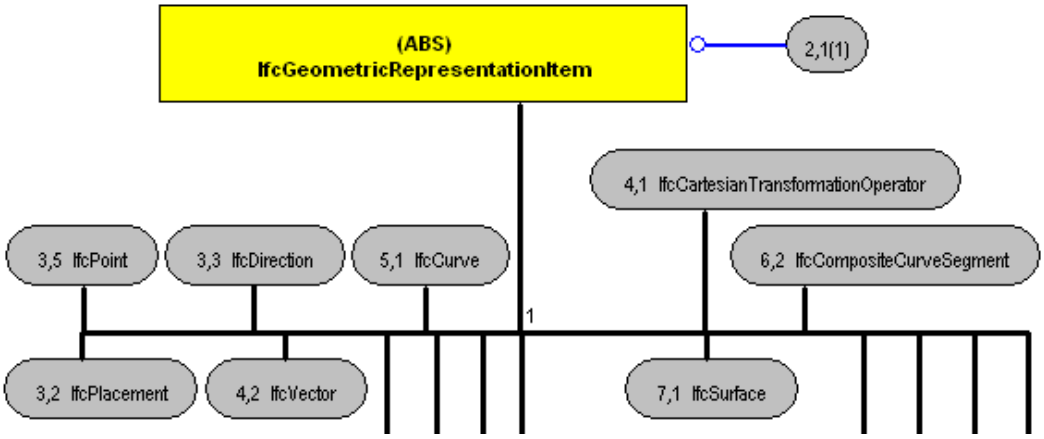


Abbildung 1.7: Express- G Spezifikation IfcGeometricRepresentationItem [6]

IfcTopologicalRepresentationItem beschreibt ein topologisches Repräsentationselement, das der Supertyp für alle topologischen Repräsentationselemente in der Geometrieressource ist, bspw. *IfcConnectedFaceSet, IfcEdge, IfcFace, IfcFaceBound, IfcPath, IfcVertex,* bzw. *IfcLoop*. Die topologische Repräsentation wird nicht direkt genutzt, sondern über die geometrische Repräsentation definiert. Ein dreidimensionaler kartesicher Punkt könnte in diesem Fall durch *IfcPolyloop* unter *IfcFace* (siehe Abbildung 1.8 und Abbildung 1.9 2,2) ausgedrückt werden. Eine Poly-Loop ist eine Schleife mit geraden Kanten, die einen planaren Bereich im Raum begrenzen. Die ist eine Schleife von Genus 1, in der die Schleife durch eine geordnete koplanare Sammlung von Punkten dargestellt wird, die die Scheitelpunkte der Schleife bilden.[6] Die Schleife besteht aus geraden Liniensegmenten, die einen Punkt in der Sammlung mit dem nachfolgenden Punkt in der Sammlung verbinden. Das Abschlusssegment erstreckt sich vom letzten bis zum ersten Punkt in der Sammlung.

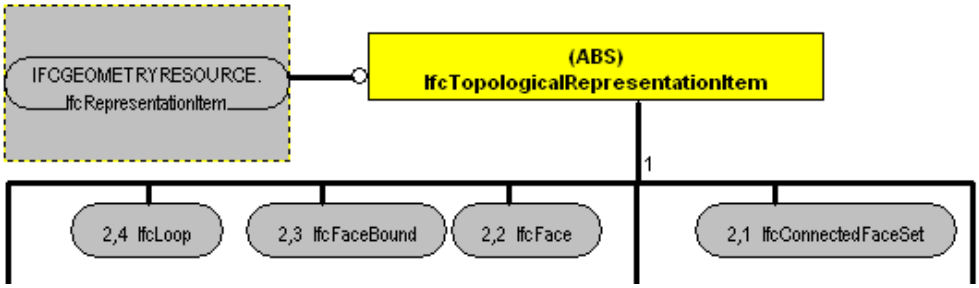


Abbildung 1.8: Express-G Spezifikation *IfcTopologicalRepresentationItem* [6]

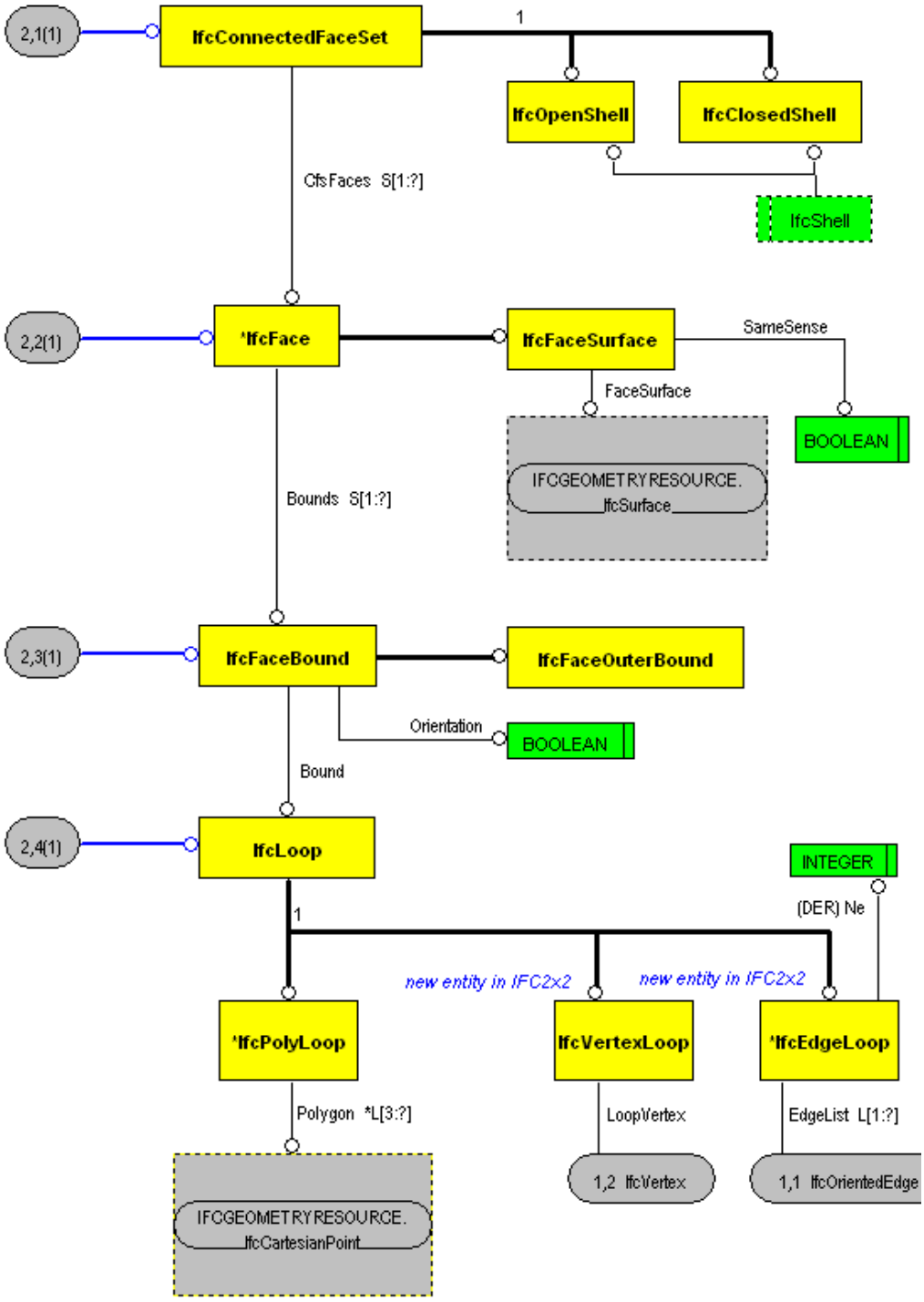


Abbildung 1.9: Express-G Spezifikation von *IfcFace* bis *IfcPolyLoop* [6]

IfcMappedItem verwendet einer vorhandenen Repräsentation als Repräsentationselement in einer zweiten Repräsentation. Das zugeordnete Element ermöglicht die Definition einer Repräsentation unter Verwendung anderer Repräsentationen.

IfcStyledItem ist eine Zuweisung eines Stils zur Präsentation eines geometrischen Darstellungselements, wie es in einer Repräsentation verwendet wird. Das *IfcStyledItem* enthält Informationen zum Präsentationsstil für Produkte, entweder explizit für ein *IfcGeometricRepresentationItem*, das Teil einer einem Produkt zugewiesenen

IfcShapeRepresentation ist, oder durch Zuweisen von Präsentationsinformationen zu *IfcMaterial*, das einem Produkt als andere Darstellung zugeordnet ist.

Wie sich die Geometrie eines Objektes präsentiert, kommt auf die zugrundeliegenden Datenstruktur der Ausgangssoftware an. Die Software muss jedoch für den Export alle benötigten Geometriedarstellungsansätze des IFC-Formates unterstützen.

1.2.6 Bewehrungsrelevante Information

- *IfcReinforcingElement*

Im Schema *IFCSTRUCTURALELEMENTSDOMAIN*, das die Möglichkeit bietet, verschiedene Arten von Bauelementen und Bauelementteilen darzustellen, existiert die abstrakte Klasse *IfcBuildingElementComponent*, die die in Bauelementen enthaltenen Items repräsentiert. Unter dieser abstrakten Klasse gehört die Klasse *IfcReinforcingElement*, die die in Beton eingebettete Stäbe, Drähte, Litzen und andere schlanke Elemente beschreibt, sodass die Bewehrung und der Beton im IFC-Datenmodell gegen Kräfte zusammenwirken. Die geometrische Repräsentation von *IfcReinforcingElement* wird von *IfcProductDefinitionShape* angegeben, wodurch mehrere geometrische Repräsentationen möglich sind. *IfcReinforcingElement* ist der Supertyp für *IfcReinforcingBar*, *IfcReinforcingMesh*, *IfcTendon* und *IfcTendonAnchor*.

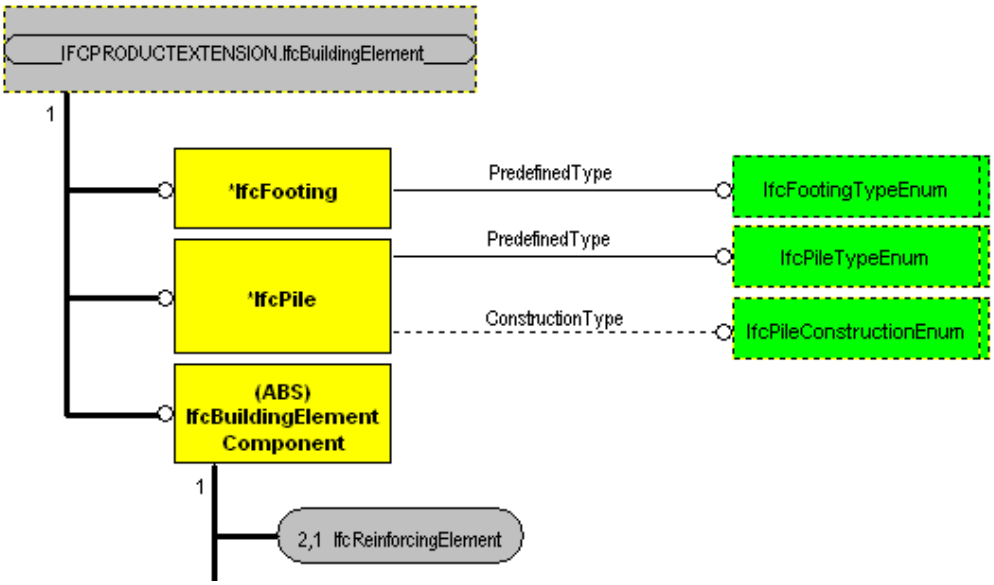


Abbildung 1.10: Express-G Spezifikation von *IfcReinforcingElement* [6]

- *IfcReinforcingBar*

Der Subtyp *IfcReinforcingBar* stellt einen Stahlstab dar, gewöhnlich mit einer vorteilhaften hergestellten Verformung, der in Beton- und Mauerwerkkonstruktion verwendet wird, um zusätzliche Festigkeit bereitzustellt. Ein einzelnes Exemplar dieser Klasse kann einen oder mehrere tatsächliche Bewehrungsstäbe präsentieren, zum Beispiel eine Reihe von Bewehrungsstäben. Die geometrische Repräsentation von *IfcReinforcingBar* wird von *IfcProductDefinitionShape* angegeben, wodurch mehrere geometrische Repräsentationen möglich sind. Die benötigten Attribute sind in Tabelle 1.1 dargestellt.

Tabelle 1.1: Attributdefinitionen von *IfcReinforcingBar*

Attribut	Definition
NominalDiameter	Der Nenndurchmesser definiert die Querschnittsgröße der Bewehrungsstange.
CrossSectionArea	Die effektive Querschnittsfläche eines Stahlstabs.
BarLength	Die Gesamtlänge des Stahlstabs. Die Gesamtlänge der gebogenen Stäbe wird nach lokalen Standards mit Korrekturen für die Biegungen berechnet.
BarRole	Die Rolle, der Zweck oder die Verwendung des Stahlstabs, d.h. Die Art der Lasten und Belastungen, die sie tragen soll.
BarSurface	Darstellung, ob die Stahlstaboberfläche glatt oder strukturiert ist.

EXPRESS specification:

```
ENTITY IfcReinforcingBar
   SUBTYPE OF (IfcReinforcingElement);
      NominalDiameter                : IfcPositiveLengthMeasure;
      CrossSectionArea               : IfcAreaMeasure;
      BarLength                      : OPTIONAL IfcPositiveLengthMeasure;
      BarRole                        : IfcReinforcingBarRoleEnum;
      BarSurface                     : OPTIONAL IfcReinforcingBarSurfaceEnum;
```

Abbildung 1.11: EXPRESS-Spezifikation von *IfcReinforcingBar* [6]

- *IfcTendon*

Als die einzige Klasse in IFC-Standard, die ein Spannglied darstellen kann, beschreibt *IfcTendon* beschreibt ein Stahlelement wie ein Draht, ein Kabel, ein Stab, eine Stange oder eine Litze, dem Beton eine Vorspannung einzuleiten, wenn das Element gespannt wird. Ebenfalls wird die geometrische Repräsentation von *IfcTendon* von *IfcProductDefinitionShape* angegeben, wodurch mehrere geometrische Repräsentationen möglich sind. Die Attribute sind in Tabelle 1.2 angezeigt. Bemerkenswerterweise befindet sich davon bezüglich der Information über Spanngliedoberfläche ausschließlich des Reibungskoeffizienten, jedoch kein geometrisches Attribut von Oberfläche im Fall von Spannstahlstab anzuzeigen, ob die Oberfläche glatt bzw. gerippt ist.

EXPRESS specification:

```
ENTITY IfcTendon
   SUBTYPE OF (IfcReinforcingElement);
      PredefinedType                 : IfcTendonTypeEnum;
      NominalDiameter                : IfcPositiveLengthMeasure;
      CrossSectionArea               : IfcAreaMeasure;
      TensionForce                   : OPTIONAL IfcForceMeasure;
      PreStress                      : OPTIONAL IfcPressureMeasure;
      FrictionCoefficient            : OPTIONAL IfcNormalisedRatioMeasure;
      AnchorageSlip                  : OPTIONAL IfcPositiveLengthMeasure;
      MinCurvatureRadius             : OPTIONAL IfcPositiveLengthMeasure;
```

Abbildung 1.12: EXPRESS-Spezifikation von *IfcTendon* [6]

Tabelle 1.2: Attributdefinitionen von *IfcReinforcingBar*

Attribut	Definition
PredefinedType	Vordefinierte generische Typen für Spannglied.
NominalDiameter	Nenndurchmesser，der die Querschnittsgröße des Spannglieds definiert
CrossSectionArea	Die effektive Querschnittsfläche des Spanngliedes.
TensionForce	Die maximal zulässige Zugkraft, die auf das Spannglied ausgeübt werden kann.
PreStress	Die auf das Spannglied aufzubringende Vorspannung.
FrictionCoefficient	Der Reibungskoeffizient zwischen Spannglied und Beton, während das Spannglied nicht verbunden ist.
AnchorageSlip	Die Verformung eines Ankers oder Schlupf von Spanngliedern beim Lösen der Vorspannvorrichtung.
MinCurvatureRadius	Der kleinste Krümmungsradius, berechnet auf der gesamten effektiven Länge des Spanngliedes, wo die Spannungseigenschaften noch gültig sind.

- *IfcTendonAnchor*

IfcTendonAnchor beschreibt einen Anker, der als die Endverbindung für Spannglied in vorgespanntem oder nachgespanntem Beton dient. Die geometrische Repräsentation von *IfcTendonAnchor* wird ebenfalls von *IfcProductDefinitionShape* angegeben.

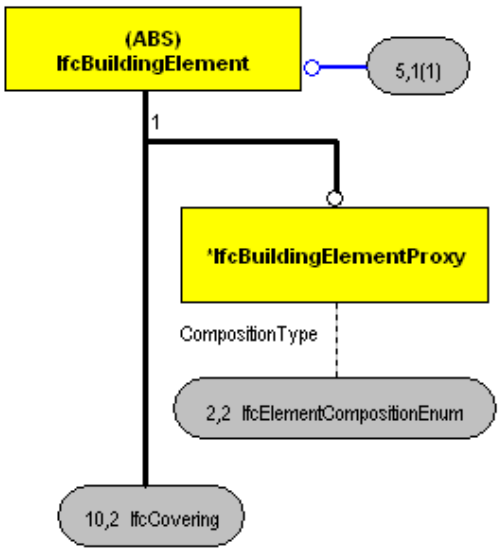


Abbildung 1.13: Express-G Spezifikation von *IfcBuildingElementProxy* [6]

- *IfcBuildingElementProxy*

IfcBuildingElementProxy (Abbildung 1.13), ein Subtyp von *IfcBuildingElement*, gehört dem Schema IFCPRODUCTEXTENSION, das das Konzept eines Produkts weiter spezialisiert. *IfcBuildingElementProxy* ist eine Proxy-Definition, die die gleiche Funktionalität

wie ein *IfcBuildingElement* bereitstellt, jedoch ohne eine vordefinierte Bedeutung des speziellen Typs des Bauelements zu haben, d.h. *IfcBuildingElementProxy* ist für nicht im IFC-Standard definierte Bauelement verwendbar, damit das Element sowie Modell beliebig erweitert werden. Die geometrische Repräsentation wird von *IfcProductDefinitionShape* und *IfcLocalPlacement* angegeben. Die Platzierung und Geometrie von einem Proxy-Element können durch kartesisches Koordinatensystem direkt geometrisch oder topologisch repräsentiert, jedoch wird die topologische Repräsentation mittles geometrischen IFC-Klassen realisiert. Bewehrung und Spannglieder, die nicht in IFC-Standard vordefiniert wird, können in diesem Fall durch *IfcBuildingElementProxy* in einem IFC-Modell erstellt. Die Entitäten beziehen Informationen von GlobalId, Name, Beschreibung usw. ein, die ebenfalls in Modellierung der Bewehrung bzw. Spannglieder gültig sind.

1.3 Bewehrung in Brückenbau

Brückenbau, als das ein oder mehrere Verkehrswege über andere Verkehrswegen und Hindernisse führende Bauwerk, sortiert sich nach dem Verwendungszweck in Straßen-, Eisenbahn-, Fußgängerbrücken usw. Eine Brücke kann je nach dem Bedarf des Verwendungszwecks, der Linienführung im Lage- bzw. Höhenplan, der Topographie des Geländes, der Ästhetik in der Umgebung usw. auf der Konstruktion von Platten-, Balken-, Bogen-, Hängebrücke usw. hergestellt werden. Daher werden die Baustoffe der Brücke wie bspw. Holz, Stein, Beton, bzw. Stahl, sowie die Kombination dazwischen festgestellt. Davon bezeichnet die Bewehrung meistens aus Stahl die Verstärkung von Betonbauteilen zur Erhöhung der Tragfähigkeit.

1.3.1 Bewehrung in Stahlbetonbrücke

In einem Bauwerk weist der Betonteil hervorragend die Druckfestigkeit auf, kann der allerdings geringe Zugkraft abtragen. Daher wird die Bewehrung aus Stahl wegen der ausgezeichneten Zugfestigkeit, zum Beton gut passenden thermalen Verformung und gutes Verbundverhältnis mit Beton weitverbreitet verwendet. Hauptsächlich ist die Bewehrung im Balken einer Stahlbetonbrücke als Längsbewehrung und Bügelbewehrung angeordnet.

Längsbewehrung ist die Bewehrung mit Betonstabstählen, die in Richtung der größten Bauteilabmessung angelegt sind, um das Biegemoment und/oder die Längskraft und Torsion abzutragen. Zur Aufnahme von Schubspannungen muss neben der reinen Biegezugbewehrung in einem Betonquerschnitt Schubbewehrung in Form von Bügeln angeordnet werden. Eine grundsätzliche Ordnung ist in Abbildung 1.14 dargestellt.

Laut EC2-1-1/NA ist der Betonstahl nach DIN 488 bzw. allg. bauaufsichtlicher Zulassung zu verwenden, der sich nach der Lieferform, der Duktilität, der Oberflächengestaltung bzw. dem Herstellverfahren unterschiedet.

Die Lieferform des Betonstahls umfasst Betonstabstahl, Betonstahlmatten, Betonstahl im Ring, Gitterträger und Bewehrungsdraht. Betonstabstahl B500 wird als gerader Stab geliefert, und sind vom Durchmessern 6, 8, 10, 12, 14, 16, 20, 25, 28, 32, 40 mm, und Längen von 12 bzw. 14 m bis 18 m genormt.

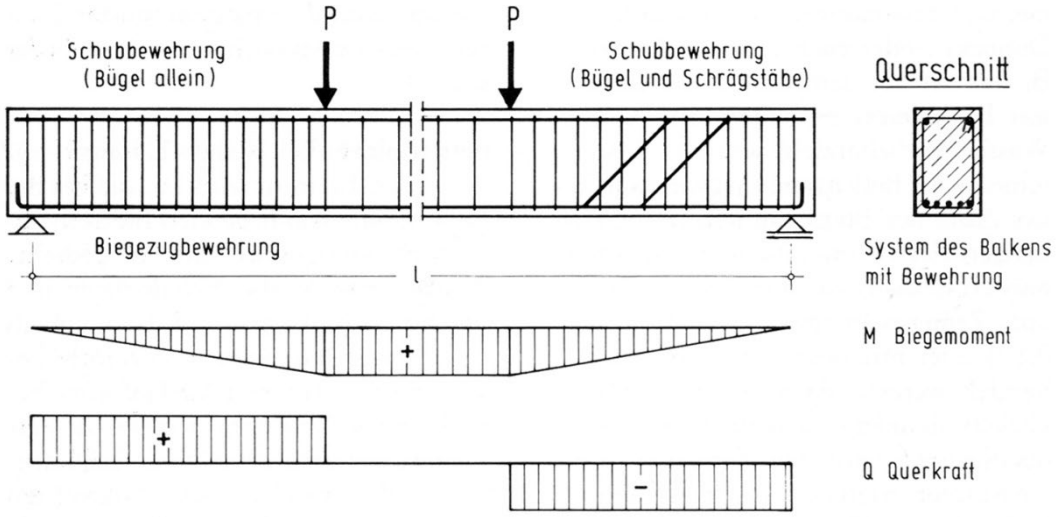


Abbildung 1.14: Anordnung der Längsbewehrung und Bügelbewehrung in Balken auf zwei Stützen [10]

Bezüglich der Oberflächengestaltung hat Betonstabstahl und -stahlmatte eine gerippte und Bewehrungsdraht jedoch eine glatte oder profilierte Oberfläche. Zur Kennzeichnung bezeichnet sich der Betonstahl B500A durch drei Rippenreihen (siehe Abbildung 1.15 a) und B500B durch zwei oder vier Rippenrehen (siehe Abbildung 1.15 b).

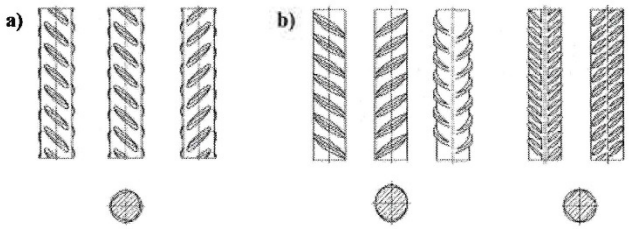


Abbildung 1.15: Kennzeichnung von Betonstahl: a) B500A b) B500B [11]

1.3.2 Spannglieder in Spannbetonbrücke

Bei vorgespannten Balken werden Druckkräfte und Biegemoment durch Anspannen der Spannglieder im Bauteil eingeleitet. Die aus äußerer Belastung auftretenden Zugspannungen und Biegemomenten werden dadurch ganz bzw. größtenteils abgenommen, sodass die Spannungen die Zugfestigkeit des Betons noch unterschreiten und der Balken an einer sicheren Seite anliegt. Bei gleicher Ausnutzung der zulässigen Betonspannungen können bei Spannbetonbalken ein kleinerer Querschnitt sozusagen ein schlankerer Balken verwendet werden.

Zu unterscheiden ist nach Spannstahl und Spannglied bzw. Spannverfahren. Hauptsätzlich besteht ein Spannglied aus Spannstahl, einem Hüllrohr, Ankern und ggf. Koppelstellen. Die Lieferformen des Spannstahls beziehen nach EC2-1-1, 3.3 Stäbe, Drähte bzw. Litzen ein. Ein Spannstab besitzt einen Durchmesser von 15 bis 36 mm mit einer glatten, gerippten bzw. mit Gewinderippen Oberfläche. Angesichts der Drähte beträgt der Durchmesser 5 bis 16 mm mit einer glatten, profilierten bzw. gerippten Oberfläche.

Spannstahl hat eine deutlich höhere Festigkeit im Vergleich zum Betonstahl, die sich auf 1030 bis 1770 N/mm² beläuft.

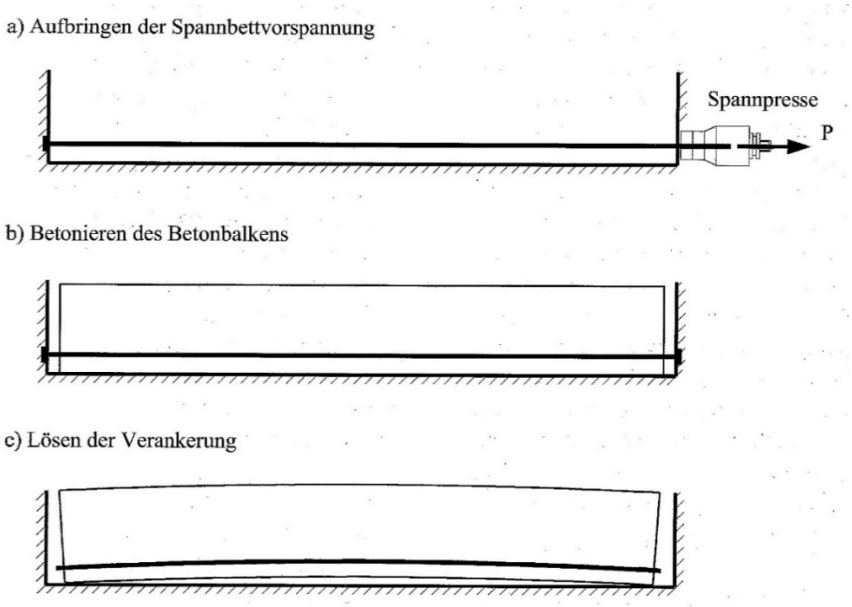


Abbildung 1.16: Prinzip der Vorspannung mit sofortigem Verbund [8]

Ein Spannglied bildet sich auch unterschiedlich nach dem Spannverfahren. Im Verfahren sofortigen Verbundes (siehe Abbildung 1.16) werden die Spannstähle erst gegen das Spannbett gespannt, danach wird der Balken darum betoniert. Die Verankerung wird nach Erhärten des Betons gelöst. D.h. in diesem Fall beziehen die Spannglieder kein Hüllrohr und keine Anker. Durch den Verbund zwischen Spannstählen und Beton ist die Spannkraft im Balken aufgebracht. Bei der Vorspannung mit nachträglichem Verbund (siehe Abbildung 1.17) wird der Balken mit Hüllrohr drin hergestellt, dann wird Spannstahl im Hüllrohr nach langer Richtung verschieben. Nach dem Abbinden des Betons wird der Spannstahl an den Ankern gespannt und die Hüllrohr mit Zementmörtel verpresst, dadurch ein Verbund zwischen Spannstahl und Beton erzeugt wird, während im Fall von Vorspannung ohne Verbund das Hüllrohr mit ausschließlich Korrosionsfett ausgefüllt und der Verbund ausschließlich durch Verankerung hergestellt wird. In diesen zwei Fällen bestehen die Spannglieder aus Spannstahl, einem Hüllrohr und Ankern.

Das Biegemoment eines Zweifeldträger zeigt sich in Abbildung 1.18 an. Zur Abminderung dieses Biegemoments ordnet sich das Spannglied an Zugspannungen abtragender Seite in der Form einer Kurve an, die sich im Feld im oberen und an Stützen im unteren Teil des Balkens ausrichtet. Der Spannglied- und Vorspannungsverlauf sowie der Momentverlauf infolge Vorspannung sind in Abbildung 1.19 angezeigt. Bei der Ermittlung der Größe der Vorspannungskraft sind Spanngliedfläche, Vorspannungszeitlauf, Pressenkraft bzw. Verankerung, Kurz- und Langzeitrelaxation des Spannstahls, Reibungsverluste, Verankerungsschlupf, elastische Verformungen des Bauteils usw. zu berücksichtigen.

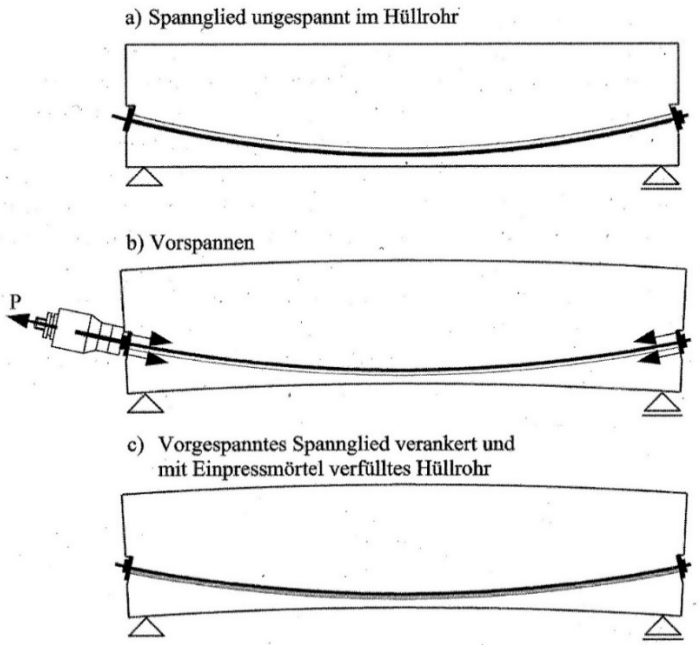


Abbildung 1.17: Prinzip der Vorspannung mit nachträglichem Verbund [8]

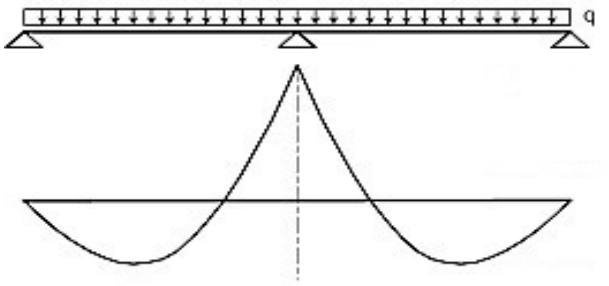


Abbildung 1.18: Biegemoment Zweifeld-Durchlaufträger [12]

Außer den oben gewähnten Faktoren sind für vorgespannte Balken Angaben über Spannstahlsorte, Anzahl, Typ, Lage der Spannglieder, Betondeckung usw. evtl. in einem Projekt bzw. einer Arbeit vonnöten.

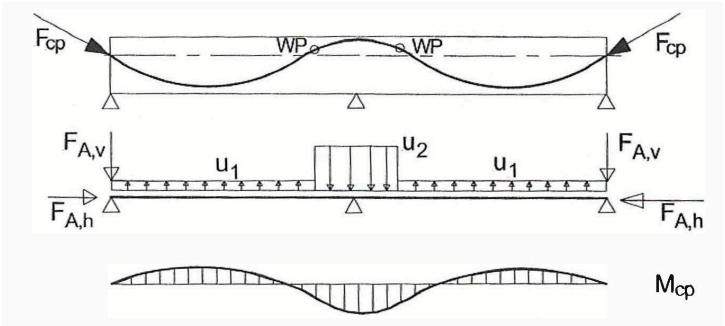


Abbildung 1.19: Spanngliedverlauf mit zupassender Scherkraft und dem Biegemoment auf Zweifeldträger [11]

1.3.3 Externe Vorspannung

Eine alternative Vorspannungsführung im Brückenbau ist die externe Vorspannung, die durch Dischinger patentierte und mit der Aue-Brücke 1934 in Sachsen erstmal realisiert wurde. „Ein externes Spannglied ist ein nachträglich vorgespanntes Spannglied, das außerhalb des Betonquerschnittes, aber innerhalb der Umhüllenden des Betontragwerkes liegt. Das Spannglied ist nur durch Ankerelemente und Umlenkelemente mit dem Betonüberbau verbunden.“[14] Diese Art der Vorspannung ist nicht nur eigenständig einsetzbar sondern auch in Kombination mit interner Vorspannung sowie Stahlbeton geeignet. Trotzdem werden die externe Vorspannung mit relativ höheren Kosten bei der Herstellung, hat die jedoch geringeren Wartungsaufwand und eine höhere Flexibilität bei einer eventuellen Verstärkungsanforderung. Daher wird die Bauweise ebenfalls bei einem Instandsetzen und Verstärken alten Brückenbaus häufig angewendet.

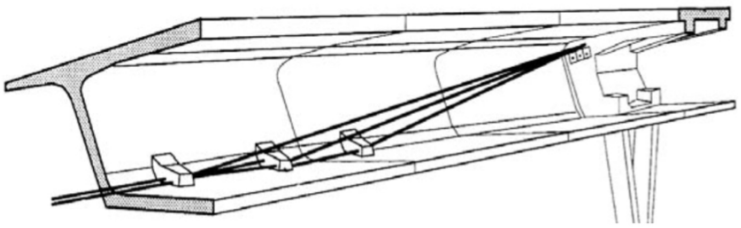


Abbildung 1.20: Externe Vorspannung bei der Segmentbauweise [15]

Abbildung 1.21: Talbrücke Berbke, Hohlkasten mit externen Längsspanngliedern [16]

2 Modellierung

2.1 Informationen zum Brückenmodell

Die Modellierungsbrücke, die Pavel Wonka Brücke (Abbildung 2.1), befindet sich in Pardubice, Tschechien wurde in den Jahren 1956 bis 1959 erbaut. Die Brücke besteht aus Spannbeton mit einer Länge von 172,5 m. Die dreifeldrige Brücke besitzt einer Hauptspannweite von 70 m und zwei Nebenspannweiten von 50 m an beiden Seiten.

Abbildung 2.1: Überblick der Pavel Wonka Brücke in Pardubice [17]

Nach dem vorhandenen Modell setzt sich der Balken aus drei drei-zelligen Hohlkästen ohne Druckstreben unter Kragplatten zusammen. Der Querschnitt im Feld ist in Abbildung 2.2 abgebildet, ein Teil von Details siehe in Abbildung 2.3. Mehr Einheiten sowie die Ansicht und Draufsicht siehe Anhang.

Außer den Längsbewehrungen in Stegen weist der Balken neben den Stegen die externen Spannstahllitzen auf. Nach den vorhandenen Daten sind jedoch die Informationen der Bewehrung und deren Durchmesser, Material, Bezeichnung, Länge sowie Längsbewehrungs- bzw. Spannliedverlauf, Spannkraft, Ausrichtung der Bügelbewehrungen usw. unklar.

Zur Ermittlung wurde das vorhandene Brückenmodell mittels der Software Revit durch das Institut für Bauinformatik, TU-Dresden erbaut (siehe Abbildung 2.4). Das Modell bezieht ausschließlich die notwendigen Grundteile einer Brücke einschließlich des Balkens, der Pfeiler und der Widerlager ein, die in Revit als allgemeines Modell mit dessen für Bewehrungen nicht geeignetem Basisbauteil erstellt wurden. Keine Querträger an den

Widerlagern, Bewehrungen oder Nebeneinrichtungen wie bspw. Kappen, Geländer, Fahrbahnbeläge usw. werden im Modell einbezogen. Die Querschnittform des Brückenbalkens wurde in der Modellierung als ein drei-zelliger Hohlkasten statt drei vereinfacht und das Balkenmodell werden in mehreren Segmenten eingeteilt, die als „Span“ bezeichnet werden. Hierbei enthalten die Eigenschaften jedes Segments die Abhängigkeit der Platzierung und die Länge bzw. das Volumen als Abmessungen. Andere Eigenschaften davon sind in Revit nicht vorhanden.

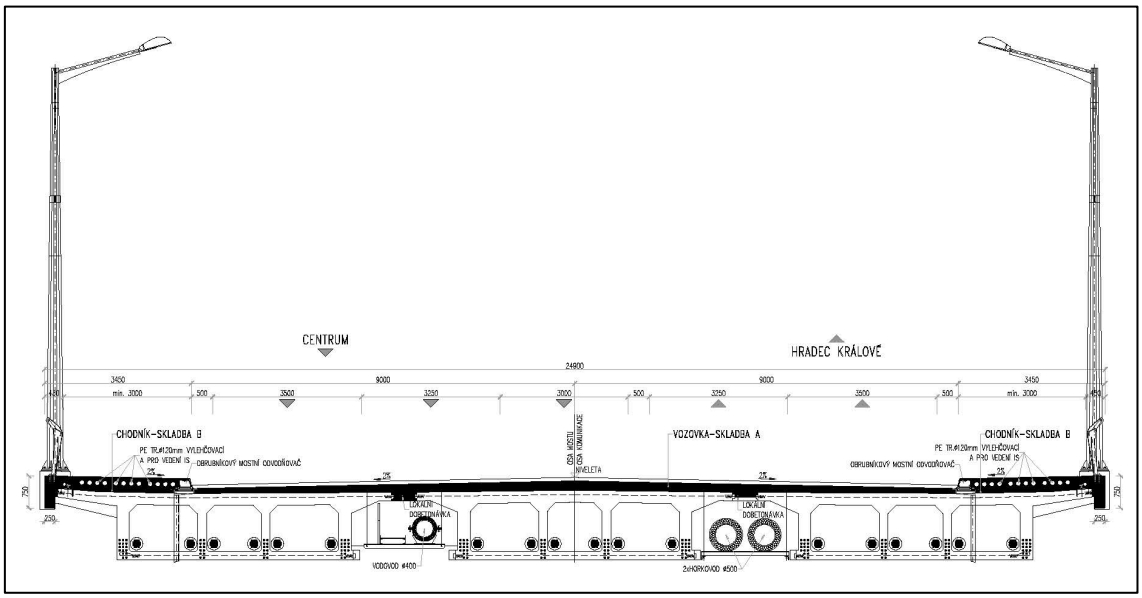


Abbildung 2.2: Überblick des Querschnitts im Feld (Quelle: Institut für Bauinformatik)

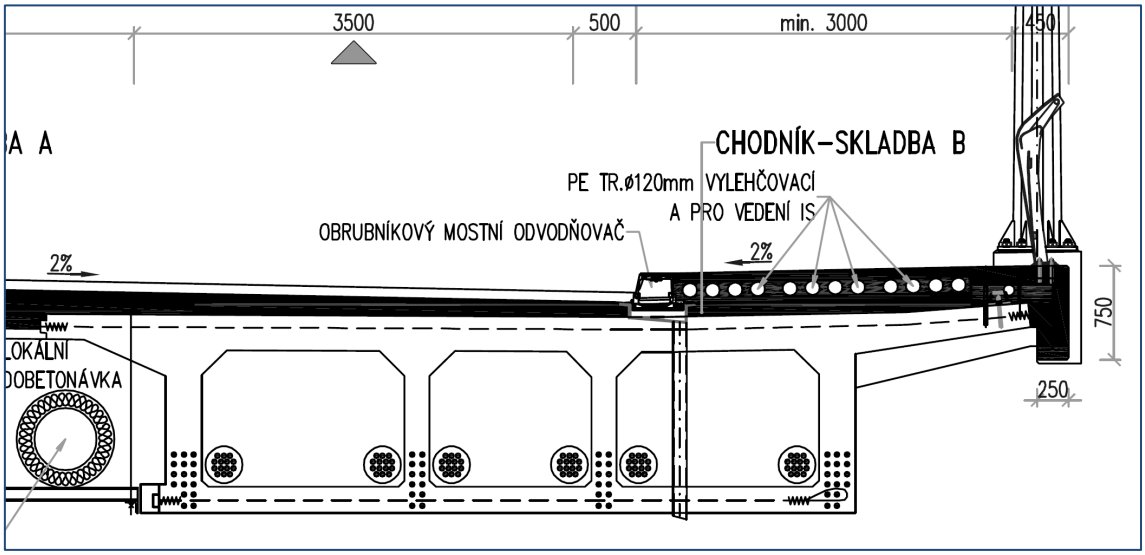


Abbildung 2.3: detaillierte Sicht des Querschnitts im Feld (Quelle: Institut für Bauinformatik)

Das davon exportierte IFC-Datenmodell steht ebenfalls dabei zu Verfügung. Alle Komponente des Modells sind in IFC-Datenmodell als *IfcBuildingElementProxy* erstellt, wo ausschließlich die Informationen über Elementname und -platzierung verfügbar sind während die anderen fehlen.

Abbildung 2.4: vorhandenes Brückenmodell in Revit-3D-Sicht

2.2 Modellierung in Revit

2.2.1 Modellierungssoftware Revit von Autodesk

Revit ist eine BIM-Software von Autodesk zur Modellierung für Bauingenieur, Architekten, Gebäudetechniker sowie Bauunternehmer. Sie bietet ein umfangreiches Werkzeugset, das effiziente Konstruktionsprozesse in einer BIM-Umgebung oder bei der Arbeit mit anderen Konstruktionsdisziplinen mit Autodesk unterstützt. Mit der Software können Benutzer ein Tragwerk bzw. eine Struktur sowie ihre Komponenten in 3D entwerfen und ebenfalls in 2D verarbeiten. Mit den auf BIM basierenden Werkzeugen von Revit können die Planung und Überwachung in verschiedenen Phasen des Lebenszyklus ausgeführt werden.

Die Software weist ein Werkzeugset zur Modellierung der Bewehrungen auf, das sich mit Versionsaktualisierung der Software ständig erweitert. In Revit 2013 wurden Einheiten für Bewehrungsfläche in den Funktionen addiert und in Revit Structure Bewehrungsmatten hinzugefügt. Seit 2014 kann die Bewehrungsüberdeckung für eine Basisabhängigkeit eines ausgewählten Bewehrungselements gesetzt werden. In Revit 2015 wurde die Platzierung der Bewehrungselemente in beliebigen 2D-Ansichten bzw. beliebigen Draufsichten, Ansichten und Schnittansichten ermöglicht. In Revit 2017 wurde die Funktion „Variable Bewehrungsverteilung“ verbessert und zwar kann eine Bewehrungsgruppe entlang geneigter Fläche veränderlich erstellt werden, sodass die Bewehrungen in einer komplexer Betonform automatisch einpassen können. Diese Funktion ist seit Revit 2018 auch bei gekrümmter Fläche geeignet. Außerdem sind die Modellierung und Detaillierung der Freiformbewehrung und die Platzierung von Bewehrung in Freiform-Betonobjekten große Entwicklung in Revit 2018. Die Erneuung von Revit 2019 ist die Kombination der in Revit 2017 und 2018 entwickelten Funktionen, d.h. die Freiformbewehrungen als Bewehrungsgruppe entlang einer Fläche auszurichten. [18]

2.2.2 Modellierung der Spannglieder

Da das vorhandene Brückenmodell in Revit erstellt wurde, wird deshalb die Modellierung ebenfalls in Revit ausgeführt. Es wurde Revit 2018 aufgrund weniger Datengröße, mehr Stabilität und besserer Kompatibilität aufgesetzt. Jedoch ist der Unterschied im Projekt benötigter Funktionen zwischen den Versionen 2018 und 2019 geringfügig.

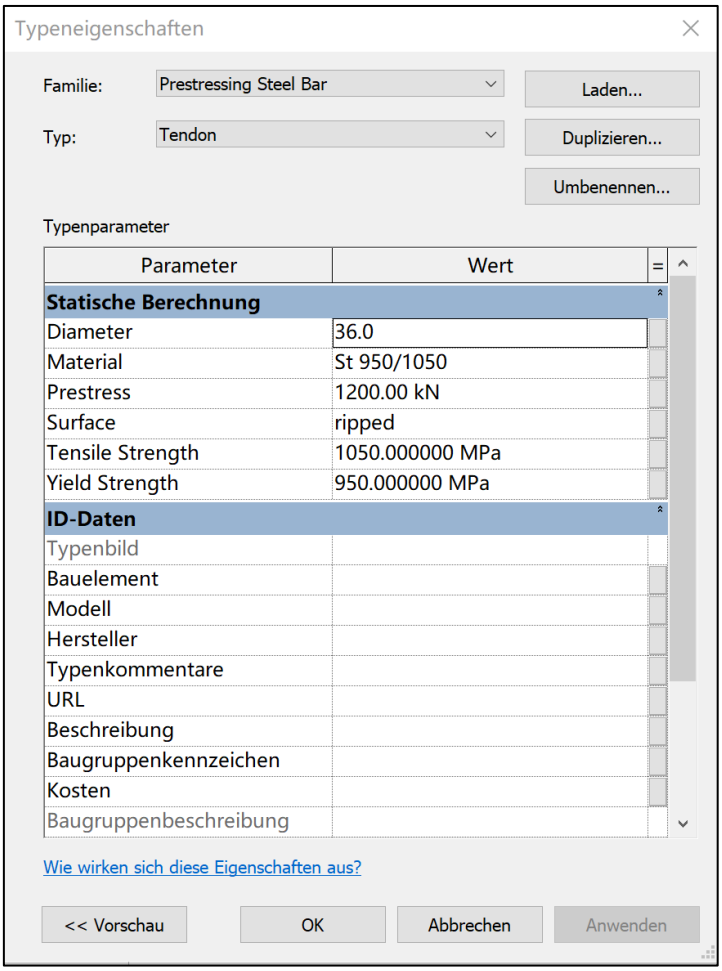


Abbildung 2.5: Parameter der Spannglieder

Nach den vorhandenen Daten ist die detaillierte Bewehrungsinformation unklar. Allerding liegt der Längsbewehrungsgrad laut den AutoCAD-Zeichnungen zwischen 1,6% bis 0,8%, der sich bereits größer herausstellt, als ein zulässiger Bereich des Mindestbewehrungsgrads. Aller Voraussicht nach dient die Längsbewehrung im Steg als Spannglied mit sofortigem Verbund. Die externen Spannlitzen werden daher ausschließlich als eine Maßnahme zum Verstärken betrachtet. Zur Ermittlung werden die Längsbewehrungen im Steg als die einzigen Spannglieder modelliert und die externen Litzen einschließlich der dazu gehörten Anker und Querträger vernachlässigt.

Revit besitzt keine spezifische Funktion zur Modellierung eines Spannglieds, deshalb werden die Spannglieder als eine neue Familie erstellt, die auf allgemeinem Modell als ihr Vorlagendatei basiert. Alle Elemente in Revit gehören zu Familien. Manche Familien, z. B. Wände, sind Bestandteile der Modellumgebung. Andere Familien, z. B. spezifische

Türen, müssen aus externen Bibliotheken in das Modell geladen werden. In Revit können Elemente ausschließlich unter Familien erstellt werden. [18]

Im Fall des Projektes werden die Spannglieder als eine neue Familie modelliert. Die Referenzebenen und -linien des Spanngliedverlaufs werden nach der Änderung des Balkenquerschnitts und dem Bedarf der Betondeckung erstellt. Ein Durchmesser von 36 mm und Spannstahl St 950/1050 mit gerippter Oberfläche werden in diesem Fall für den Spannstab ausgewählt. Die Vorspannung lautet 1200 kN. Alle relevanten Parameter inkl. Zugfestigkeit bzw. Streckgrenze sind in Abbildung 2.5 abgebildet. Das durch Revit ausgerechnete Volumen eines Bauelements steht jeweils in seinen eigenen Eigenschaften als einer der Abmessungen zu Verfügung.

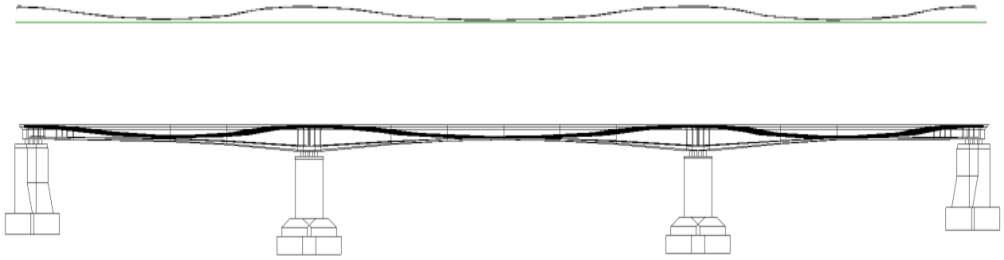


Abbildung 2.6: Spanngliedverlauf in Familienverarbeitungs- bzw. Brückenmodellsicht

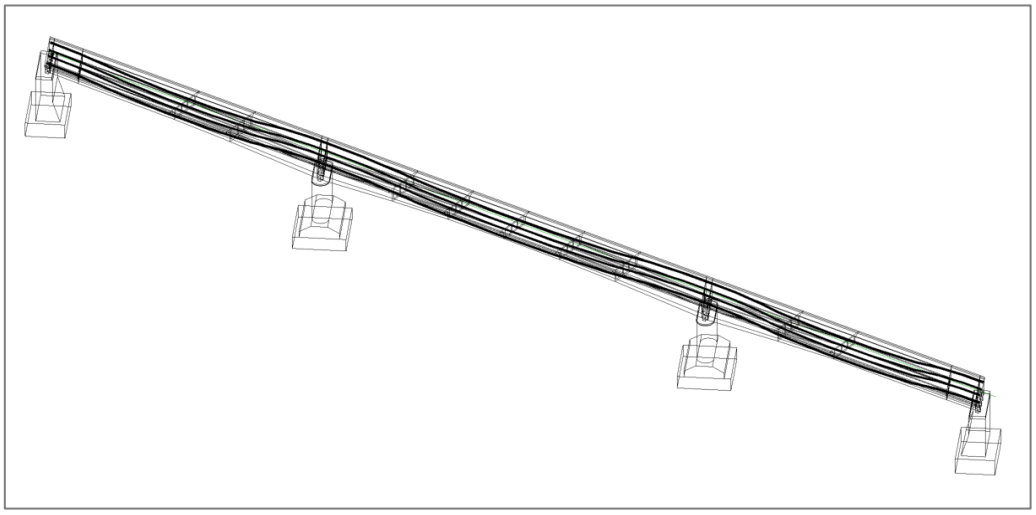


Abbildung 2.7: 3D-Drahtmodell mit Spanngliedern

Zur Abminderung des aus Einwirkungen auftretenden Biegemoments ordnen sich die Spannglieder an Zugspannungen abtragender Seite im Balkenmodell in der Form einer Kurve an, die sich im Feld im oberen und an den Stützen im unteren Teil des Balkens ausrichtet. Spanngliedverlauf siehe Abbildung 2.6. Das 3D-Drahtmodell wird in Abbildung 2.7 gezeigt.

Nebenher wurden die Spannglieder auch durch BIM/CAD-Programm Allplan teilweise modelliert, da Allplan die Funktion von Tendon-Modellierung aufweist. Allerdings war anschließend ein Exportversuch nicht erfolgreich.

2.2.3 Modellierung der Bügelbewehrung

Das Balkenmodell wurde auf einem allgemeinem Modell aufgebaut, in dem Bewehrung durch Revit-Bewehrungsfunktion jedoch nicht eingefügt werden darf. Das Modell muss deshalb in ein Modell in Ingenieurbauweise konvertiert werden. Alternative kann das Basisbauteil des allgemeinen Modells für Bewehrung geeignet eingestellt werden. In Anbetracht der Vielzahl der Verarbeitungen wird die Bewehrung ausschließlich in einem Segment erstellt und aufgrund der Vereinheitlichung des Aussehens und Datenexports wird das Segment als ein allgemeines Modell mit bewehrungsgeeignetem Basisbauteil eingestellt. Wegen weniger Krümmung der Unterfläche wird daher Segment „Span 1.3“ ausgewählt.

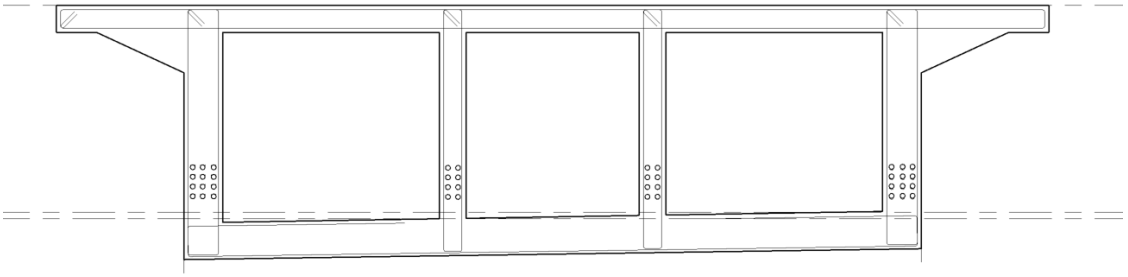


Abbildung 2.8: Ausrichtung der Bügelbewehrung

Die Bügelbewehrung wird in Bewehrungsgruppe mit maximalem Abstand von 100 mm und Betonüberdeckung von 30 mm als Basisabhängigkeit im Modell erstellt und die Ausrichtung und Form sind in Abbildung 2.8 abgebildet. Die Anzahl der Bügelbewehrung jeder Gruppe beläuft sich auf 103. Andere Parameter können in Abbildung 2.9 abgelesen werden.

Da das Segment Span 1.3 eine geneigte und weinig gekrümmte Unterfläche aufweist, wird die Bewehrung im Steg mittels der Funktion „variabler Bewehrungsgruppe“, die variablen Längen einzelner Stäbe im Basisbauteil zulässt, und die Bewehrung im Bodenplatten durch Abhängigkeitseinstellung anpassend modelliert. Allerdings können sich entlang der Unterfläche die Längen der Stegbewehrung nicht automatisch anpassen. Die Bodenplattenbewehrung kann zwar in Revit 2018 durch Abhängigkeitseinstellung nicht mit anpassender Platzierung erstellt werden, was in anschließend richtig wieder installierter Revit 2019 jedoch ermöglicht wird. Hierbei entsteht trotzdem eine kleine Abweichung der Platzierung der Bodenplattenbewehrung über eingestellte Basisabhängigkeit. Vermutlich eignet sich Revit nicht zügig für die Erkennung des manuell erstellten Hohlkastenmodells. Die Bewehrungsgruppe passt sich wahrscheinlich schwer an einem Tragwerk in Revit an, das komplexe Geometrie von Höhlen und Betonkörper besitzt, da gegenwärtig Revit sowieso keine spezifische Funktion aufweist, um einen Hohlkasten zu modellieren.

2.3 Export in IFC

Revit bietet einen vollständig zertifizierten IFC-Import und -Export an, die auf BuildingSMART®-IFC-Datenaustauschstandards basieren. Für den Export unterstützt Revit die

Standards von *IFC4*, *IFC2x3* und *IFC2x2*, wobei im Projekt das bewehrte Brückenmodell in *IFC2x3* exportiert wird. Revit exportiert Bauelemente in ein IFC-Datenmodell basierend auf Kategorien, zu denen die Bauelemente gehören. Z.B. ein Balken wird aus Revit in das IFC-Element *IfcBeam* exportiert, da der Balken ein Element zur Kategorie „Beam“ gehört.

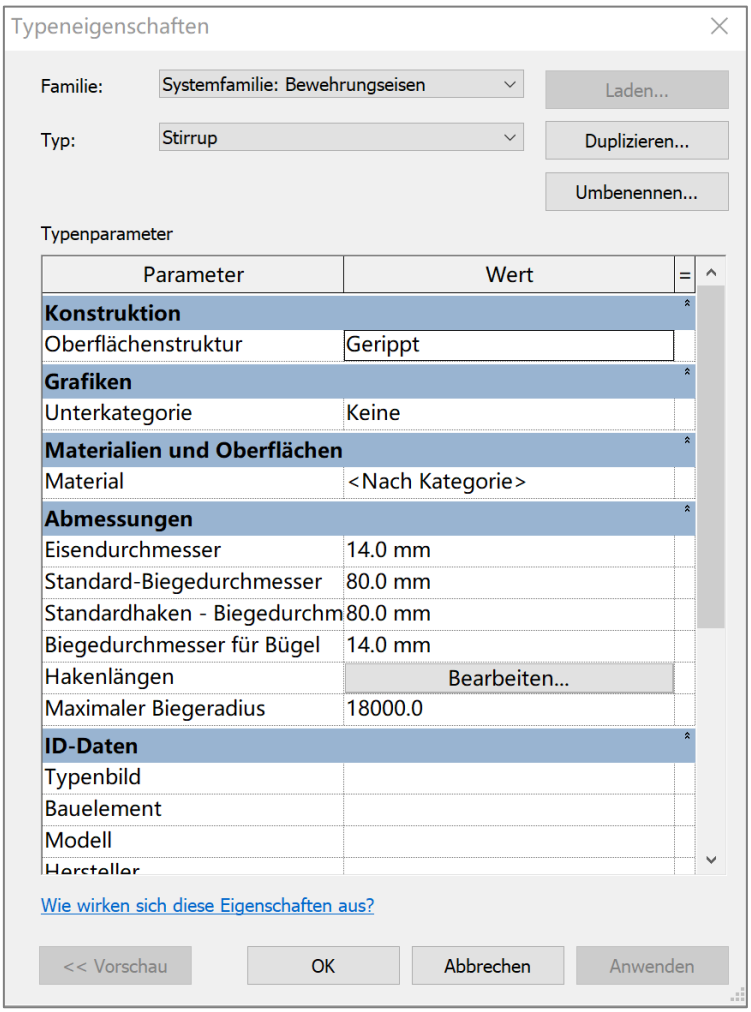


Abbildung 2.9: Parameter der Bügelbewehrung

Nach dem Export werden das auf allgemeinem Modell aufgebauten Brückenmodell und Spannglieder im IFC-Datenmodell durch *IfcBuildingElementProxy* erstellt. Das auf allgemeine aufgebauten Modell besitzt Allerdings weniger Tragwerkseigenschaften. In Abbildung 2.10 werden die Spannglieder im IFC-Datenmodell mittels BIM Vision angezeigt. Hierbei werden die Balkensegmente Span 3.3 und Span 3.1 in der Ansicht versteckt. Die Geometrie des Brückenmodells und der Spannglieder sind topologisch repräsentiert. Die zuvor eingegebenen Parameter der Spannglieder, die in Abbildung 2.5 angezeigt sind, sind jedoch nach dem Export verloren. Das durch Revit ausgerechnete Volumen wird ebenfalls nicht exportiert.

Merkwürdigerweise wurden die Spannglieder durch BIM/CAD-Programm Allplan teilweise modelliert, und ein Exportversuch wurde demnach gemacht, der schiefgegangen ist, möglichweise aufgrund der problematischen Installierung von Allplan wegen Softwarekonflikts ähnlich wie bei der Installierung von Revit 2019 am Anfang des Projektes.

Es könnte auch sein, dass das aus Revit exportierte IFC-Datenmodell basierend auf dem für Bewehrung nicht geeignetem Modell in Allplan problematisch importiert wurde, und demnach gegen darin modellierte Spannglieder ein Konfliktfehler entstünde. Weitere Ermittlung des genauen Grunds wird während des Projektes nicht durchgeführt.

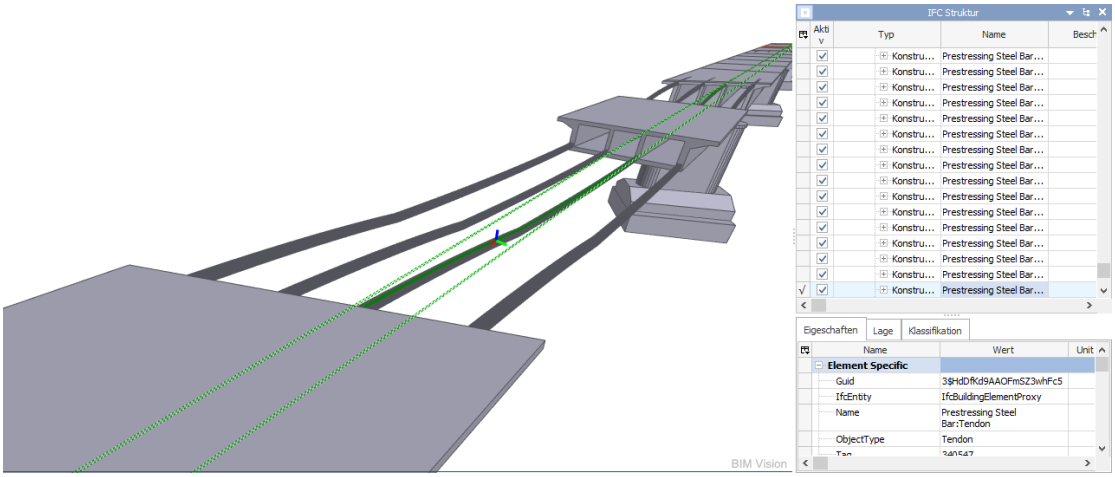


Abbildung 2.10: Spannglieder in 3D-Sicht mittels BIM Vision

Die Modellierung der Bügelbewehrung in Revit wurde im Balkensegment Span 1.3 durchgeführt und wird ins IFC-Element *IfcReinforcingBar* exportiert. Alle Parameter in Abbildung 2.9 sind mittels BIM Vision ablesbar, während das in Revit gezeigte Elementvolumen in das IFC-Datenmodell nicht als einer der Parameter exportiert wird. Vermutlich hat Revit einen eigenen Algorithmus, um das Volumen des jeweiligen Bauelements auszurechnen, jedoch wird das Ergebnis nicht ins IFC-Modell exportiert. Das Volumen ist aber durch einen selbstentwickelten Algorithmus berechenbar. In Abbildung 2.11 ist die Bügelbewehrung im Span 1.3 (Element Span 1.3 versteckt) dargestellt.

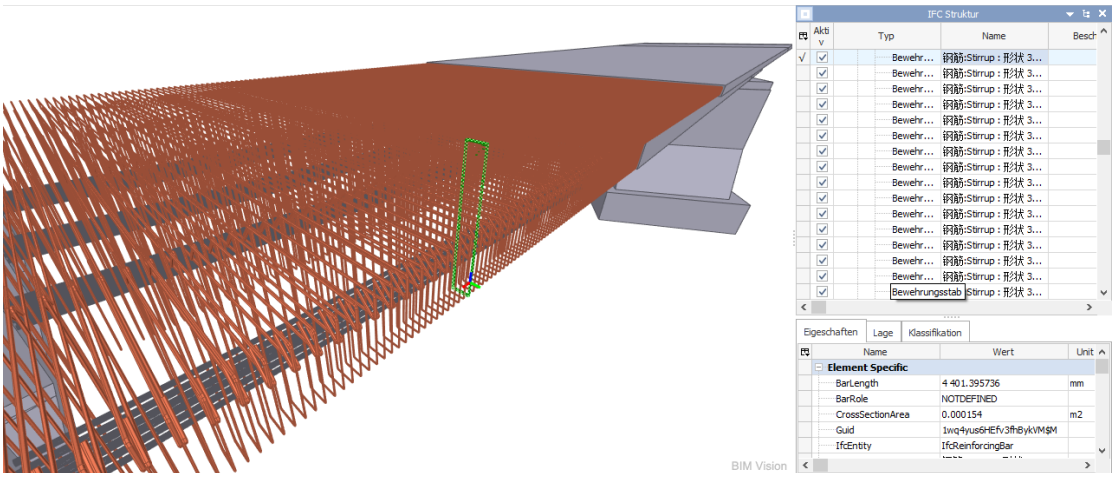


Abbildung 2.11: Bügelbewehrung in 3D-Sicht mittels BIM Vision

3 Filterung der notwendigen Informationen

3.1 Generisches Filterframework und BIMfit

3.1.1 Einführung

Durch die Anwendung von BIM verbessern sich die Integration und die Koordination zwischen beteiligten Fachdisziplinen. Der IFC-Standard von Open BIM bietet ein offenes Format für den Informationsaustausch und -einheit zwischen verschiedene Fachdisziplinen an. Gleichzeitig führt das zu einer hohen Komplexität und Größe interdisziplinärer Informationsräume des Bauwerksmodells sowie den heterogenen und vielfältigen Informationsanforderungen dadurch. Jedoch entspricht dem Informationsbedarf eines Beteilige normalweise nur eine Teilmenge Daten aus dem vollständigen Modell. Eine Verarbeite für Datenentnahme und dadurch eine vereinfachte Sicht auf das Modell sind deshalb in jeder einzigen Anwendung erforderlich. Allerdings ist es zeitaufwändig und mühsam für jeden Beteiligten die benötigten spezifischen Informationen aus dem großen Modell zu entnehmen.

Auf diesem Grund vollzieht sich die Entwicklung der Filtermethoden, wovon ein mehrheitlicher Teil sich auf Filterung der benötigen Daten im IFC-Modell konzentriert. Der Zweck des Filterns von Datenmodells ist um die zahlreiche Verarbeitung der Entnahme von Anforderungsdaten abzunehmen, damit die unnötigen Informationen und Details ausgespart werden. Das Filtern dient für einem spezifischen Informationsbedarf, z.B. die Ermittlung der Gradiente aller Dächer, der Anzahl der Türen alles Zimmer, des Bewehrungsgrads eines Tragwerks, bzw. der Montageplanung im Bauprozess von einem Bauwerk aus Fertigbeton. Mit dem anwendungsspezifischen Teilmodell bzw. der geteilten Menge der Daten, die durch bestimmten Filter selektiert und reduziert werden, wird die Interpretierung in der jeweiligen Anwendung leichtert und der spezielle Bedarf jeder Beteiligten im Bauprojekt zeit- und mühesparend erfüllt.

3.1.2 Modellsichten

Die hohe Komplexität und Größe zuzüglich der Heterogenität verschiedener Fachdisziplinen führt zu einer vielfältig strukturellen Gesamtmodellsicht mit der Kompliziertheit und Verflochtenheit. Abgesonderte Modellsichten können durch die Filterung aus einer selektierten, reduzierten und transformierten Menge Daten abgebildet werden. Diese können bestimmte kontextabhängige Anforderungen und spezifische Informationsbedürfnisse eines Endbenutzers beschreiben. Die unterschiedlichen Modelltypen bei bspw. einer Gebäudeanalyse können sich in statischen Modellsichten, dynamischen Modellsichten, Multimodellsichten bzw. Systemmodellsichten einordnen.

Eine statische Modellsicht drückt die Informationsreduktion eines Gebäudemodells auf der Grundlage einer Subschemaspezifikation aus, die den Informationsbedarf für einen spezifischen Informationsaustausch eines übergeordnete Geschäftsprozesses definiert.[19] Bspw. zeigt Abbildung 3.1 (b) ein eigenständiges Teilmodell aus einer

Gebäudeanalyse an, deren Informationsgehalt im Vorfeld des eigentlichen Filterprozesses definiert wird.

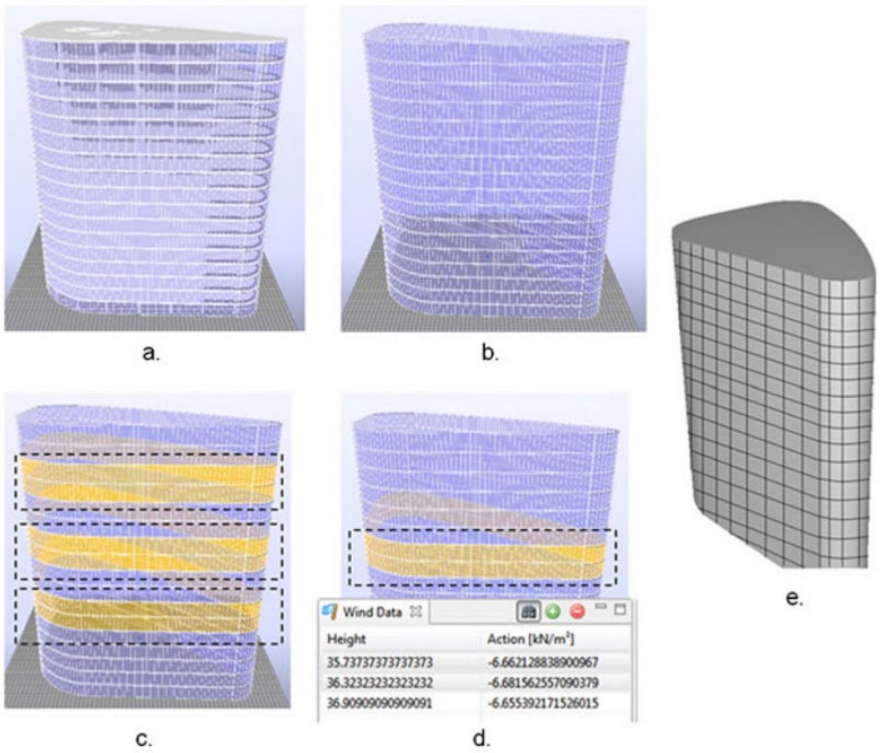


Abbildung 3.1: Die unterschiedlichen Modellsichttypen, wie sie bspw. an der Gebäudeanalyse auftreten: (a) vollständiges Bauwerksmodell, (b) statische, (c) dynamische, (d) Multimodell-, (e) Systemmodellsicht [19]

Eine dynamische Modellsicht beschreibt den konkreten aufgabenspezifischen Informationsbedarf eines Modells, bspw. Abfrage des Bewehrungsvolums in einem Stahlbetonbauwerk bzw. aller Bauelemente aus einem bestimmten Material. Der breite bzw. vielfältige Anwendungsbereich ist deshalb ein Vorteil der dynamischen Modellsicht. Ansonsten reicht diese Modellsicht von der Unterstützung der Visualisierungen bis zu umfassenden baubetrieblichen Modellanalysen. Als Beispiel wird eine dynamische Modellsicht in Abbildung 3.1 (c) dargestellt.

Multimodellsichten sind sowohl statische als auch dynamische Modellsichten, die angesichts fachspezifischer Datenmodelle aufgebaut werden. Diese sind geeignet für die Ermittlung spezifischer Bauelemente unter nbestimmten Bedingungen. Bspw. zeigt Abbildung 3.1 (d) alle Fassadenelemente oberhalb des 5. OG an, die eine besonders hohe Windsogbelastung aufweisen.

Systemmodellsichten beschreiben auch die statische bzw. dynamische Modellsichten, die jedoch einer systemspezifischen Teilmenge eines Bauwerksmodells oder Multimodells zugeordnet sind. In Abbildung 3.1 (e) zeichnet sich bspw. ein Gebäudehüllensystem. Die Sichten einbeziehen die in einem System funktionalen Randbedingungen mit und repräsentieren mehr Informationen als der einzelnen Elemente davon.

3.1.3 Filtern auf verschiedenen Informationsstrukturebenen

Das Erstellen der Modellansichten für eine große Menge von Anforderungssituationen erfordert den kombinierten Einsatz von Methoden zum Filtern von Modelldaten auf verschiedenen Informationsstrukturebenen, die sich in Schema-. Klassen-, Objekt- und Systemebenen einordnen können.

Das Filtern auf Schemaebene zeichnet sich die Reduzierung einer Modellinstanz um eine bestimmte Menge der Klassen und damit um alle instanziierten Objekte dieser Klassen. Die Modellreduktion korrespondiert mit einem vordefinierten Subschema. Diese Filterart entspricht den „standardisierten“ Domänemodellsichten. Das Filtern auf Klassenebene definiert die Modellreduktion und Objektselektion, die auf die Typbeschränkungen basieren. Das Filtern auf Objektebene selektiert Objekte durch die Auswertung der spezifischen Eigenschaften der individuellen Objekte. Dadurch können die Multimodellsichten erzeugt werden. Das Filtern auf Systemebene, das selbstverständlich zu den Systemmodellsichten ausführt, ist der Begriff von Objektselektion und Rekonfiguration, die auf algorithmischen bzw. wissensbasierten Funktionen aufgebaut werden.

3.1.4 Deskriptive und präskriptive Filtermethoden

Als zuvor erwähnt, der Zweck des Filterns ist die Selektion und Reduktion von einem Ausgangsmodell zu einer Teilmodell, das ein spezifischer Informationsbedarf erfüllt und damit die Verarbeitung vereinfacht werden kann. Der Prozess kann prinzipiell auf de- und präskriptives Filtern erfolgen. Ein deskriptiver Filterprozess verwendet ein formales Schema bzw. eine formale Sprache, wie zu einem Zeitpunkt der Modellinhalt teilweise abgenommen, bzw. ein Schema zu seinem Subschema reduziert werden. Dieser vordefinierte Ansatz entspricht dem Filtern auf Schemaebene. Ein präskriptiver Filterprozess verwendet eine Menge von Basisoperationen, die zur Laufzeit zu einem logischen Ansatz zusammengesetzt und durchgeführt werden. Diese Art von Prozess korrespondiert mit dem Filtern auf Klassen-, Objekt-, und Systemebene.

Im Vergleich zu deskriptiven- weisen präskriptive Methoden im Allgemeinen eine höhere Flexibilität und Mächtigkeit auf. Sie sind zudem leichter erweiterbar und erlauben eine schnelle Anpassung an veränderte Anforderungssituationen.

3.1.5 Generisches Filterframework

Das generische Filterframework bildet sich auf der bausteinförmigen Filterfunktionalität heraus. Prinzipiell sind die komplizierten Filteraufgaben in die konfigurierbaren, flexibel kombinierbaren und wiederverwendbaren Filtermodule aufzuteilen. Das Framework bezieht die Filterarten auf verschiedenen Informationsstrukturebenen mit dem präskriptiven Filterprozess ein, einschließlich des Filterns auf Klassen-, Objekt- bzw. Systemebene.

Das generische Framework besteht aus mehrere generischen Filterfunktionen, durch die ein komplexer Filterprozess durchgeführt und erweitere Filteroperationen konzipiert werden kann, um das spezifische Bedürfnis zu erfüllen. Die Funktionen, die auf Basisfunktionen zurückgreifen, setzen die Filterarten auf konzeptioneller Ebene um. Die

Basisfunktionen unterteilen die spezifischen komplexen Aufgaben auf die an jeden bestimmten Anwendungsbereich zugeordneten Unterfunktionen, damit sich der Filterprozess komplexer Aufgaben stufenweise anordnet. Die Funktionen befinden sich nach dem Anwendungsbereich auf neutraler, Domain-, und Anwendungsebene.

Die neutrale Ebene bezieht die domänen- und anwendungsunabhängigen Basisfunktionen, die Filterfunktionen in den unterschiedlichen Ebenen realisieren. Die Basisfunktionen können in Meta-, semantische bzw. Kern-, und algorithmische und Schlussfolgerungsfunktionen strukturiert werden. Die Metafunktionen basieren auf den zugrundeliegenden Modellierungskonzepten, wie bspw. EXPRESS, und bilden die Modellsemantik auf die Modellierungskonzepte ab. Die semantische bzw. Kernfunktionen basieren auf einem spezifischen Datenschema und nutzen die explizite Semantik in einem Datenmodell, wie bspw. *IFC2x3* bzw. *IFC4*. Die Kernfunktionen bestehen aus einer endlichen Menge von Metafunktionen. Die algorithmischen Funktionen und Schlussfolgerungsfunktionen basieren ebenfalls auf einem spezifischen Datenschema, jedoch nutzen sie die implizite Semantik eines Modells, dadurch das funktionale Mapping zwischen Konzepten unterschiedlicher Schemata erlaubt wird. Abbildung 3.2 zeigt den hierarchischen Aufbau der drei Ebenen und Basisfunktionen der neutralen Ebene.

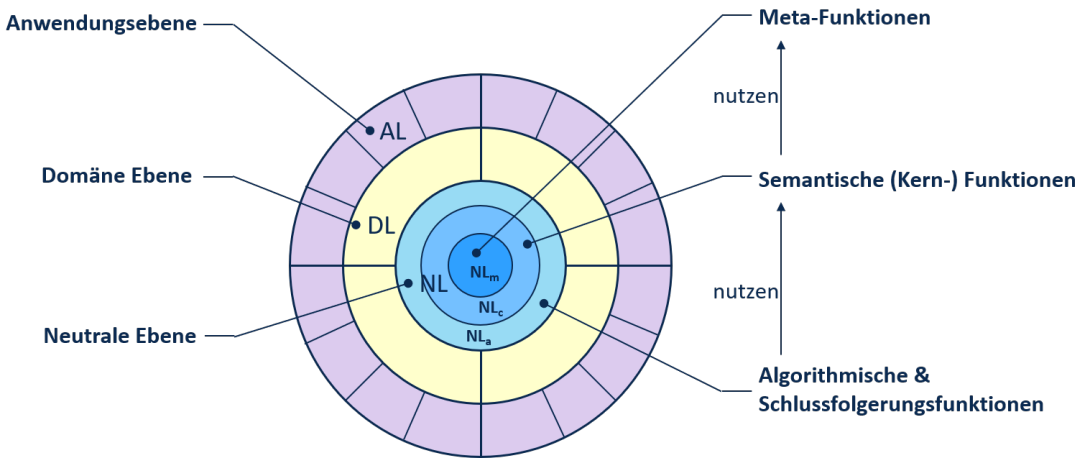


Abbildung 3.2: Hierarchische Darstellung des generischen Filterframeworks auf drei Ebenen und der Funktionstypen der neutralen Ebene [21]

3.1.6 BIMfit

Zur Realisierung des generischen Ansatzes der modularen Basisfunktionen und Unterstützung des Filterns auf Klassen-, Objekt-, und Systemebene wurde am Institut für Bauinformatik der TU Dresden die BIM Filtertoolbox BIMfit als Referenzimplementierung entwickelt. Die Java-Bibliothek BIMfit basiert auf dem Konzept des generischen Filterframeworks und implementiert eine Grundmenge von Basisfunktionen der neutralen Ebene, einschließlich den Metafunktionen und darauf aufgebaute Kern- und Schlussfolgerungsfunktionen. BIMfit umfasst insgesamt 95 Funktionen, die sich in 31 Meta-, 53 Kern- und 12 Schlussfolgerungsfunktionen unterteilt, und entsprechend ihrer Typen in drei individuellen Klassen wahrgenommen werden.[19] Die bereitgestellten Funktionen können entweder selbstständig, oder in Kombination miteinander für eine komplexe Filteroperation appliziert werden, wie bspw. bedingte Objektselektion, Export valider Teilmodelle

auf Basis einer Selektionsmenge, semantische Modellprüfung Ableitung impliziter Modellinformationen, usw.

Die Metafunktionen ermöglichen den Zugriff auf durch das EXPRESS-Schema definierte Datenmodelle, z.B. IFC. Über die Klasse „StepDataModel“, die als JAVA-Repräsentation des STEP-Modells wirkt, werden Metafunktionen von BIMfit aufgerufen. Dazu gehört wie bspw. die Funktion `getEntitiesWithAttributeValue`, die zur Ermittlung aller Objekte eines bestimmten Typs mit einem bestimmten Attributwert dient.

Die Kernfunktion werden auf den Metafunktionen aufgebaut und nutzen die explizit in einem Modell definierte Semantik. Die Kernfunktionen werden über die Klasse „*IfcDataModel*“ aufgerufen, die die IFC-JAVA-Repräsentation ist. Alle Kernfunktionen können eigenständig verwendet werden. Dazu gehört bspw. die Funktion `getAdjacentSpaces` zur Ermittlung der benachbarten Räume eines gegebenen Raumes auf der Grundlage vorhandener Raumbegrenzungen.

Algorithmische und Schlussfolgerungsfunktionen sind nur implizit vorhanden und bestehen ebenfalls aus Basisfunktionen. Aufgerufen werden die algorithmischen und Schlussfolgerungsfunktionen über die Klasse *IfcGeomDataImpl*. Die Funktionen kombinieren Kernfunktionen mit arithmetischen Operationen, und sollten damit geometrische bzw. topologische Informationen beschaffen, jedoch können derzeit nur geometrische Aufgaben erfüllen. Bspw. dient die Funktion `getAbsolutePoint` zur Transformation der lokalen Koordinaten eines dreidimensionalen Punktes eines *IfcProduct* Objekts bezüglich der Werte des globalen Koordinatensystems.

3.2 Filteralgorithmen und Datenverarbeitung der Modellierung

3.2.1 Filteranforderungen des Projektes

Im Prozess zuvor des Projektes wurden die Spannglieder und Bügelbewehrungen im Brückenmodell bereits mit der Software Revit modelliert, und ins Format IFC exportiert. Aufgrund keiner Modellierungsfunktion von Spanngliedern in Revit wurde der Spannstäbe als eine neue Familie unter der Vorlage allgemeines Modell erstellt, und wird daher in IFC als Proxy-Elemente erkannt. Das gegebene Balkenmodell wurde als kein ingenieurbauweises Tragwerk sondern allgemeines Modell in Revit erstellt. Es wird demnach auch als Proxy-Elemente in IFC abgelesen. Hierbei wird die Balkensegmente Span 3.3 und Span 3.1 in der Ansicht versteckt. Die Eigenschaft des Balkensegments Span 1.3 wurde geändert, dass der Basisbauteil für Bewehrungen geeignet eingestellt wurde, damit sich die Bügelbewehrungen problemlos im Span 1.3 anpassen kann. Nach dem Export wurde die Bewehrungen als *IfcReinforcingBar* im IFC-Datenmodell erstellt.

Das Projekt erfordert die benötigten Informationen der modellierten Bewehrungs-, sowie Spannstahl-Komponenten durch eine Selektion dieser Daten mittels Filteralgorithmen. Hierbei wird die Daten mittels der am Institut für Bauinformatik entwickelte BIM Filter Toolbox (BIMfit) in JAVA-Programmierungssprache selektiert.

Vor allem sollten alle Informationen der Spannglieder und Normalbewehrungen ausgefiltert werden. Bei den Informationen der Spannglieder wird der Spannliedverlauf

erfordert und bei den gesamten Bewehrungen der Bewehrungsgrad. Ansonsten sollte das Alignment der Spannglieder ermittelt werden. Zuzüglich der benötigten Daten sollte es auch möglich sein, alle anderen Informationen bspw. Bewehrungsquerschnitt, -material, usw. bei Interesse mittels dem Filteralgorithmen zu selektieren.

3.2.2 Im Projekt benötigte Funktionen

- `getEntitiesOf`

Die Funktion `getEntitiesOf` dient zur Suchung aller Entitäten der spezifischen Klassentypen, und auch die Subklassen. Im Projekt werden die Entitäten *IfcReinforcingBar* dadurch gesucht, um alle Bügelbewehrungen auszufiltern.

Tabelle 3.1: Spezifikation der Metafunktion `getEntitiesOf`

Input	`T[] getEntitiesOf(Class<? extends T> type)`
Parameter	`T` – Subklasse von der Entität `type` – der Klassentyp der gesuchten Entitäten
Output	Alle Entitäten der spezifischen Typen
Throw	`jsdai.lang.SdaiException`

- `getEntitiesWithAttributeValue`

Durch die Funktion `getEntitiesWithAttributeValue` werden alle Entitäten gesucht, die im spezifizierten Attribut einen gegebenen Wert aufweisen. Die Funktion ist unabhängig von der Entitätsklasse, jedoch nur für die gegebene Klasse. Die Funktion sucht nicht in einer Menge von Werten. Im Projekt werden die Entitäten von den Spanngliedern mit dem Attribut Value „Tendon“ ausgesucht, damit eine Liste aller Spanngliedern erstellt werden.

Tabelle 3.2: Spezifikation der Metafunktion `getEntitiesWithAttributeValue`

Input	`jsdai.lang.EEntity[] getEntitiesWithAttributeValue(Class<? extends jsdai.lang.EEntity> clazz, String atrributeName, Object value`
Parameter	`claszz` – Klasse der gesuchten Entitäten `attributeName` – das spezifische Attribut `value` – der referenzierte Value zu suchen
Output	`Entities`, die der spezifischer Value als Referenz hat
Throw	`jsdai.lang.SdaiException` – wenn ein Problem auftretet

- `getValueOfAttribute`

Durch die Funktion `getValueOfAttribute` wird einen bestimmten Attribut Value einer Entität gesucht. Bspw. im Projekt werden die Values „Diameter“, „CrossSetionArea“ aus Entitäten *IfcReinforcingBar* gesucht, um den Bewehrungsgrad auszurechnen.

Tabelle 3.3: Spezifikation der Metafunktion `getValueOfAttribute`

Input	`Object getValueOfAttribute(jsdai.lang.EEntity entity, String attributeName, boolean deepSearch)`
Parameter	`entity` – die Instanz einer Expressklasse `attributeName` – das spezifische Attribut `deepSearch` – sucht in Subelementen mit diesem Attribut, d.h. es wird auch der Wert in den Referenzen gesucht (Achtung: zeitaufwändig)
Output	Der Value (Referenz) des Attributs
Throw	`jsdai.lang.SdaiException` – wenn ein Problem auftretet

- `getPolylinePoints`

Diese algorithmische Funktion gehört zur Klasse *IfcGeomDataImpl*. Durch die Funktion können die Punkte einer Polyline zurückgeliefert.

Tabelle 3.4: Spezifikation der Funktion `getPolylinePoints`

Input	`public jsdai.SIfc2x3.EIfccartesianpoint[] getPolylinePoints(jsdai.SIfc2x3.EIfcpolyline polyline)`
Parameter	`polyline` – die Polylinie
Output	Array d. Punkte der Polylinie
Throw	`NotConformToImplementationGuide` – Wenn der Inhalt nicht dem IFC-Implementierungsführung entspricht.

3.2.3 Filtern der allgemeinen Daten der Bewehrungen und Spannglieder

Das gegebene Brückenmodell ist als allgemeines Modell in Revit erstellt und der Balkenteil besteht aus etlichen Segmenten. Grundsätzlich können die Bewehrungen in Revit ausschließlich in einem Tragwerkmodell, das in Revit erkannt werden kann, durch die Ingenieurbau-Funktion Bewehrung erbaut. Als allgemeines Modell muss die Eigenschaft des Basisbauteils für die Bewehrungen anpassend eingestellt werden. Aufgrund der Vielzahl der Verarbeitung der Änderungen wurde ausschließlich der Balkensegment

Span 1.3 als das Schlüsselelement verarbeitet, d.h. das Erbauen der Bügelbewehrung wurde im Balkensegment Span 1.3 durchgeführt.

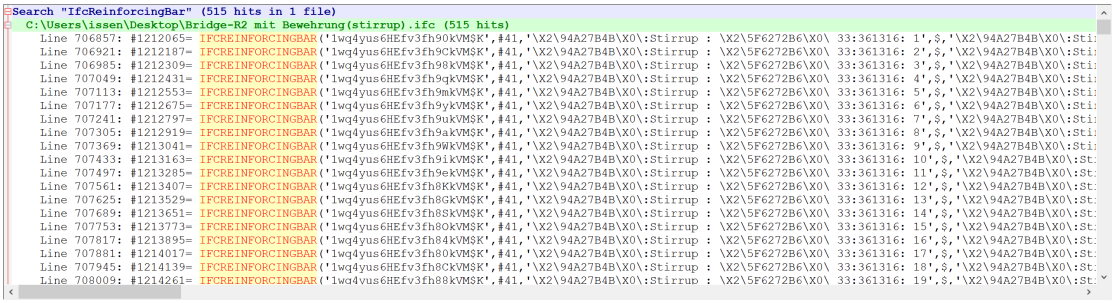


Abbildung 3.3: Gesuchte Bewehrungsinformationen durch Schlüsselwort „*IfcReinforcingBar*“ im IFC-Datenmodell mittels Notepad++

```
12 public class Filter {
13     private StepDataModel model;
14     public void createModel(String ifcfile) {
15         StepParser parser = new StepParser("C:\\Users\\issen\\eclipse-workspace\
   \Projektarbeit");
16         try {
17             model = parser.parse(ifcfile);
18         } catch (ParsingException e) {
19             System.out.println("Loading File failed");
20             e.printStackTrace();
21         }
22     }
23
24     public void filterReinforcement() {
25         try {
26             EIfcreinforcingbar[] barList = model.getEntitiesOf(EIfcreinforcingbar.class);
27             for (int i=0; i<barList.length; i++) {
28                 System.out.println(barList[i]);
29             }
30         } catch (SdaiException e) {
31             System.out.println("Filtering ReinforcingElements failed");
32             e.printStackTrace();
33         }
34     }
35
36     public void filterTendon() {
37         try {
38             EEntity[] tendonList =
   model.getEntitiesWithAttributeValue(EIfcbuildingelementproxy.class, "ObjectType", "Tendon");
39             for (int i=0; i<tendonList.length; i++) {
40                 System.out.println(tendonList[i]);
41             }
42         } catch (SdaiException e) {
43             System.out.println("Filtering TendonElements failed");
44             e.printStackTrace();
45         }
46     }
47 }
```

Abbildung 3.4: `filterReinforcement` und `filterTendon`

Alle Bügelbewehrungen bilden sich als *IfcReinforcingBar* im IFC-Datenmodell aus, wie in Abbildung 3.3 gezeigt, die benötigten Daten mittels Notepad++ gesucht.

Mithilfe der Funktion `getEntitiesOf` in BIMfit können die allgemeinen Informationen der Bügelbewehrungen selektiert. Die Funktion `getEntitiesOf` dient zur Suchung aller Entitäten der spezifischen Klassentypen, und auch der Subklassen. In diesem Fall werden die Entitäten *IfcReinforcingBar* dadurch gesucht, um alle Bügelbewehrungen auszufiltern. Der Codeausschnitt von JAVA-Code des Filterns stellt sich in Abbildung 3.4 错误!未找到引用源。 dar. Das Filtern-Programm wird durch Methode `main` durchgeführt, Einzelheiten siehe im Anhang.

Im Vergleich zu Bügelbewehrungen ist das Filtern der Spannglieder relativ komplizierter, da nicht nur die Spannglieder sondern auch das ganze vorhandene Brückenmodell im IFC-Datenmodell als Proxy-Elemente erstellt wurden. Durch die BIMfit-Funktion `getEntitiesWithAttributeValue` werden alle Entitäten von den Spannggliedern gesucht, die im spezifizierten Attribut einen gegebenen Wert aufweisen. Die Funktion ist unabhängig von der Entitätsklasse, jedoch nur für die gegebene Klasse. Die Daten der Spannglieder werden durch „Tendon“ im Form *IfcLabel* im IFC-Datenmodell angemarkert (siehe Abbildung 3.5).

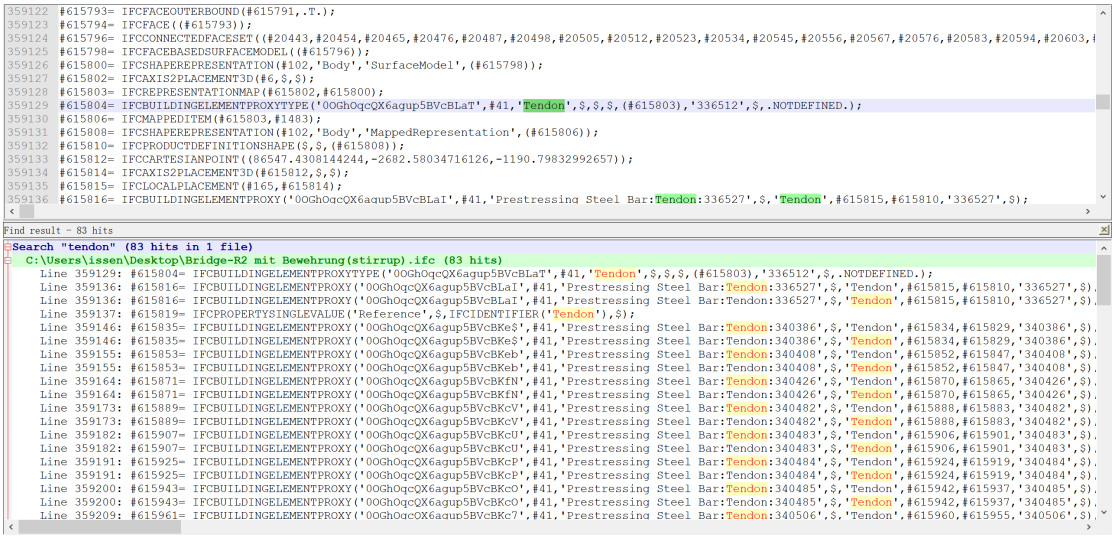


Abbildung 3.5: Gesuchte Daten der Spannglieder durch Schlüsselwort „Tendon“ im IFC-Datenmodell mittels Notepad++

Diese Daten beziehen die Attribute bspw. GlobalId, Name, ObjectType, usw. ein. Mittels dieser Funktion der BIMfit werden in JAVA-Sprache die Entitäten von den Spannggliedern mit dem Attribut Value „Tendon“ ausgesucht, damit eine Liste aller Daten der Spannglieder erstellt werden. Die Algorithmen stellen sich im Anhang aus.

Merkwürdigerweise werden die meiste Parameter wie z.B. der Querschnitt, der Durchmesser, bzw. die Lange eines Stahlstabs in allgemeinen Daten der Bügelbewehrungen als entsprechende Attribute beinhaltet, jedoch werden die in Revit neu gebauter Familie eingegebene Parameter nicht im IFC-Datenmodell erfolgreich exportiert.

3.2.4 Ermittlung und Konzept des Bewehrungsgrads

Der Bewehrungsgrad wird in *IFC4* als ReinforcementVolumeRation unter Pset_ConcreteElementGeneral gezeichnet. Es beschreibt das geforderte Verhältnis der effektiven Masse der Bewehrung im Verhältnis zur effektiven Masse des Betons für dieses Element. Allerdings in *IFC2x3* besteht keine entsprechende Definition. Andererseits kommt dieser Parameter nicht in Revit vor, aufgrund dessen keine passende Information im IFC-Datenmodell exportiert wird. Daher sollte ein Algorithmus für den Bewehrungsgrad entwickelt und implementiert werden.

```
10 public class Filter {
11     private StepDataModel model;
12     public void createModel(String ifcfile) {
13         StepParser parser = new StepParser("C:\\Users\\issen\\eclipse-workspace\
   \Projektarbeit");
14         try {
15             model = parser.parse(ifcfile);
16         } catch (ParsingException e) {
17             System.out.println("Loading File failed");
18             e.printStackTrace();
19         }
20     }
21
22
23     public void filterReinforcementValue() {
24         try {
25             EIfcreinforcingbar[] barList1 = model.getEntitiesOf(EIfcreinforcingbar.class);
26             double sumVolume = 0;
27             for (int i=0; i<barList1.length; i++) {
28                 double crossSection = (double) model.getValueOfAttribute
   (barList1[i],"CrossSectionArea",false);
29                 double length = (double) model.getValueOfAttribute(barList1[i], "BarLength",
   false);
30                 double volume = crossSection * length/1000;
31
32                 sumVolume = sumVolume + volume;
33             }
34             System.out.println(sumVolume);
35         } catch (SdaiException e) {
36             System.out.println("Filtering ReinforcingElementValue failed");
37             e.printStackTrace();
38         }
39     }
40 }
```

Abbildung 3.6: Algorithmus des Bewehrungsvolumen

Wie eher erwähnt, die Ermittlung der Querbewehrungsgrad wird ausschließlich im Segment Span 1.3 durchgeführt. Grundsätzlich wird der geometrische Bewehrungsgrad durch den Quotienten aus Bewehrungsvolumen und Stahlbetonvolumen beschrieben. Um das Bewehrungsvolumen auszurechnen muss erst zwei Parameter, die Länge und Querfläche der Bewehrungen, die als Attribute im IFC-Datenmodell einbezogen sind, aus den Daten selektiert und ausgefiltert werden. Durch die Funktion `getValueOfAttribute` werden die Attribut Values, Länge und Querfläche unter der Entität *IfcReinforcingBar* gesucht (Codeausschnitt siehe in Abbildung 3.6, Einzelheiten siehe Anhang). Das Ergebnis 0,51 m³ kommt dadurch heraus (siehe Abbildung 3.7)

```
Problems  Javadoc  Declaration  Console  Coverage
<terminated> main (1) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (2018年12月5日 下午11:17:20)
[StepParser - Wed Dec 05 23:17:20 CET 2018]  Build JSDAI repository C:\Users\issen\eclipse-workspace\Projektarbeit...
[StepParser - Wed Dec 05 23:17:27 CET 2018]  Parsing file finished succesfully!
0.5103754025245573
```

Abbildung 3.7: Ergebnis des Bewehrungsvolumens

Angesichts des Volumens des Segments Span 1.3 (siehe in 错误!未找到引用源。) stellt ebenfalls keine direkte Information zur Verfügung. Ursprünglich im Revit-Modell wurde das Volumen bereits durch die Software automatisch ausgerechnet und als Abmessung in Elementeigenschaften protokolliert. Jedoch konnte diese Information nicht im IFC-Datenmodell exportiert werden. Das Konzept zur Beschaffung des Volumens muss auch entworfen werden.

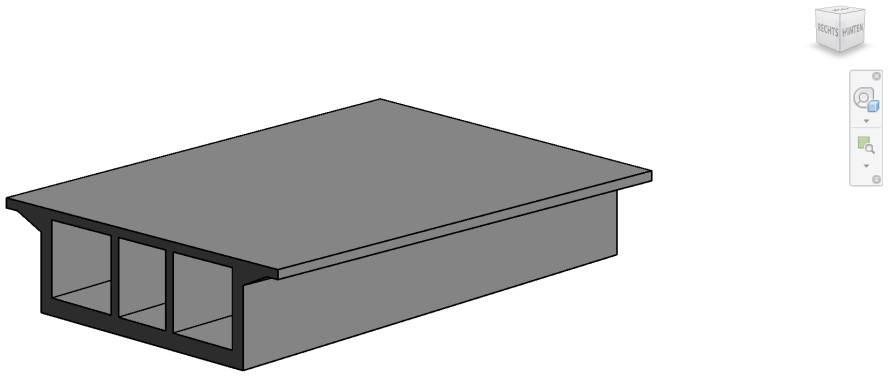


Abbildung 3.8: Span 1.3 in Revit 3D-Sicht

Um das Volumen des Segments zu beschaffen sollte erst die Querschnittgröße festgestellt werden. Der Querschnitt des Hohlkastens zeigt sich wie in 错误!未找到引用源。 an. Grundsätzlich ist die Querschnittgröße gleich die Differenz der ganzen Querfläche und Hohlfläche. Die Unterkante des Querschnittes neigt sich, sodass die Außenkante trapezförmig abgebildet ist. Darum ist die Querschnittgröße gleich die Differenz der Außentrapezgröße und Hohlfläche plus die schattige Fläche:

$$A = A_{Auß} - A_{Hohl} + A_S \qquad (3.1)$$

Für den Fall in 错误!未找到引用源。 ist die Hohlfläche gleich:

$$A_{Hohl} = \frac{1}{2}(h_{11} + h_{12}) \cdot l_1 + \frac{1}{2}(h_{21} + h_{22}) \cdot l_2 + \frac{1}{2}(h_{31} + h_{32}) \cdot l_3 \qquad (3.2)$$

Oder abgekürzt

$$A_{Hohl} = \sum_{i=1}^{3} \frac{1}{2}(h_{i1} + h_{i2}) \cdot l_{\mathrm{i}} \qquad (3.3)$$

Die Größe einer Höhe h_{i1} bzw. h_{i2} hängt jedoch von der Steigung der Unterkante nach y-Richtung im Modell ab. Geht man deshalb davon aus, dass die Höhe h_{ij} mittels der Steigung der Unterkannte k_y gleich

$$h_{ij} = h + k_y y \qquad (3.4)$$

dann ergibt sich mit

$$l_{\mathrm{i}} = y_{i2} - y_{i1} \qquad (3.5)$$

die Hohlfläche

$$A_{Hohl} = \sum_{i=1}^{3} \frac{1}{2}[(h + k_y y_{i1}) + (h + k_y y_{i2})] \cdot (y_{i2} - y_{i1}) \qquad (3.6)$$

nämlich

$$A_{Hohl} = \sum_{i=1}^{3} h\,(y_{i2} - y_{i1}) + \frac{1}{2}\,k_y\,({y_{i2}}^2 - {y_{i1}}^2) \qquad (3.7)$$

Bezüglich der Außentrapezgröße erhielt man

$$A_{Auß} = \frac{1}{2}(h_{01} + h_{02}) \cdot l \qquad (3.8)$$

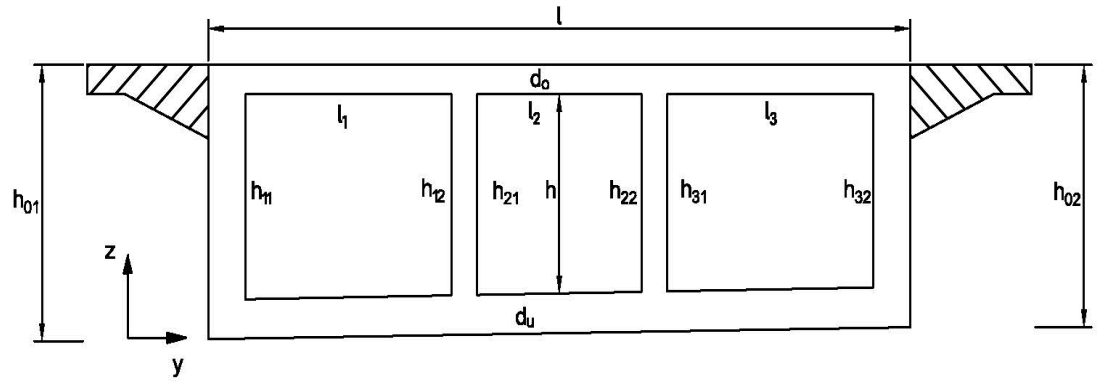


Abbildung 3.9: Querschnitt Span 1.3

Ebenfalls wie Gleichung (3.7) mit

$$h_{0j} = h + d_o + d_u + k_y y \qquad (3.9)$$

und

$$l = y_{02} - y_{01} \qquad (3.10)$$

$$d_o = z_{oo} - z_{ou} \qquad (3.11)$$

$$d_u = z_{uo} - z_{uu} \qquad (3.12)$$

lässt sich die Außentrapezgröße auf die Form

$$A_{Auß} = [h + (z_{oo} - z_{ou}) + (z_{uo} - z_{uu})]\,(y_{02} - y_{01}) + \frac{1}{2}\,k_y\,({y_{02}}^2 - {y_{01}}^2) \qquad (3.13)$$

Aus Gleichung (3.1), Gleichung (3.7) und Gleichung (3.13) erhielt man die Querschnittgröße A auf die Form

$$A = [h + (z_{oo} - z_{ou}) + (z_{uo} - z_{uu})]\,(y_{02} - y_{01}) + \frac{1}{2}\,k_y\,({y_{02}}^2 - {y_{01}}^2) - \left[\sum_{i=1}^{3} h\,(y_{i2} - y_{i1}) + \frac{1}{2}\,k_y\,({y_{i2}}^2 - {y_{i1}}^2)\right] + A_S \qquad (3.14)$$

Wie in der Längsansicht in Abbildung 3.10 gezeigt, die Gradiente der Unterebene führt zur Änderung des Querschnittes. Die Schlüsselhöhe h ändert sich nach der Änderung der Unterebene, während der andere Teil des Querschnittes konstant bleibt. Insbes. ist die in 错误!未找到引用源。 schattige Fläche in den ganzen Balken eine Konstante. Die Unterebene, die in der Ansicht eine Linie zeigt, stellt eigentlich eine Kurve dar, die einer grade Linie annähert. Mit einem vereinfachten Algorithmus nennt man diese Unterkante eine schräge grade Linie, dann ergibt sich die Höhe

$$h = h(x) = h_0 + k_x x \ (0 \le x \le L) \qquad (3.15)$$

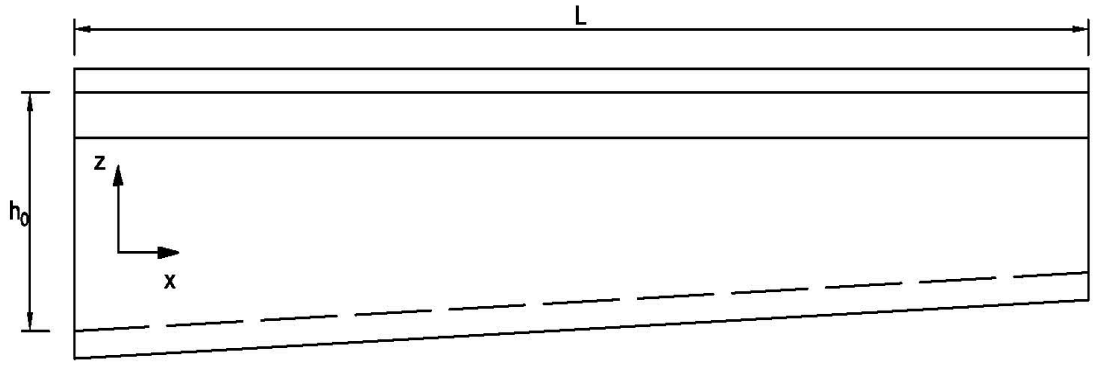


Abbildung 3.10: Ansicht Span 1.3 nach Längsrichtung

Hierbei ist h_0 die Höhe h im Anfangsquerschnitt. Ausgehend von Gleichung (3.14) erhielt man die Querschnittgröße, die sich nach Längsrichtung ändert:

$$\begin{aligned} A = {} & [h(x) + (z_{oo} - z_{ou}) + (z_{uo} - z_{uu})]\,(y_{02} - y_{01}) \\ & + \frac{1}{2}\,k_y\,({y_{02}}^2 - {y_{01}}^2) \\ & - \left[\sum_{i=1}^{3} h(x)\,(y_{i2} - y_{i1}) + \frac{1}{2}\,k_y\,({y_{i2}}^2 - {y_{i1}}^2)\right] + A_S \end{aligned} \qquad (3.16)$$

Das Volumen von Span 1.3 ist das Integral von Geleichung (3.16). Die Lösung lautet dann

$$\begin{aligned} V = \int_0^L \Bigg(& \big((h_0 + k_x x) + (z_{oo} - z_{ou}) + (z_{uo} - z_{uu})\big)\,(y_{02} - y_{01}) \\ & + \frac{1}{2}\,k_y\,({y_{02}}^2 - {y_{01}}^2) \\ & - \left(\sum_{i=1}^{3} (h_0 + k_x x)\,(y_{i2} - y_{i1}) + \frac{1}{2}\,k_y\,({y_{i2}}^2 - {y_{i1}}^2)\right) \\ & + A_S \Bigg)\, dx \end{aligned} \qquad (3.17)$$

Tabelle 3.5: benötige Informationen

benötige Koordinaten	z_{oo}, z_{ou}, z_{uo}, z_{uu}, y_{01}, y_{02}, y_{i1}, y_{i2}
benötige Konstanten	L, h_0, k_x, k_y, A_S

Das Ergebnis kommt daher auf die Koordinaten der Schlüsselpunkten an, die in Tabelle 3.5 angezeigt werden. Alle Konstanten davon, L, h_0, k_x, k_y bzw. A_S, können auch durch die Koordinaten und einfache ebene Geometrie ausgerechnet werden. Alle Koordinaten werden als *IfcCartesianPoint* ausgedrückt, die die Subklassen dem Schema von *IfcGeometryResource* gehört, jedoch in dem Fall des Projektes topologisch repräsentiert werden. Daher sind die benötigten Koordinaten durch BIMfit nicht beschaffbar, da die Schlüsselpunkte nicht automatisch durch einen Algorithmus mittels der Funktionen von BIMfit erkannt, unterschieden und gefiltert werden können.

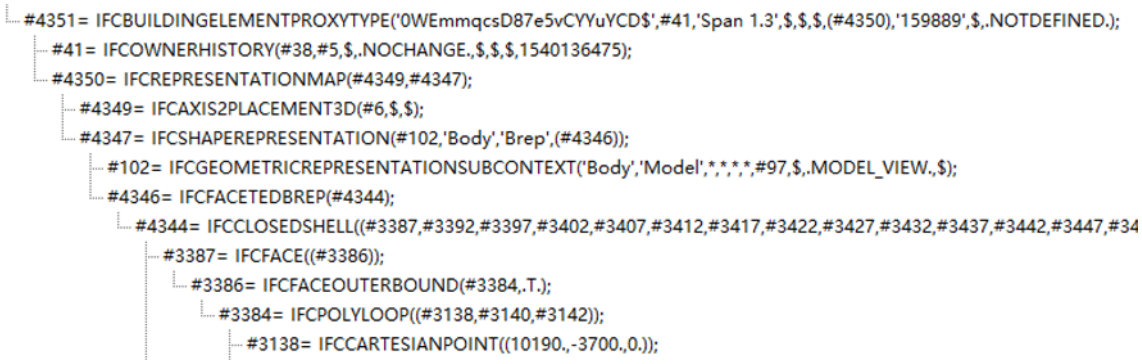


Abbildung 3.11: topologisch definierte Punkte von Span 1.3

In Revit-Software enthielt man schon das Volumen von Span 1.3, das gleich 49,197 m³ ist, damit ergebt sich der Querbewehrungsgrad: 1.04%.

Angesichts des Längsbewehrungsgrad wird das Volumen der Spannglieder und das Volumen des ganzen Balkens gebraucht. Allerdings wurden die kurvenförmigen Spannglieder auch durch *IfcBuildingProxy* im IFC-Datenmodell erstellt, worin die Länge- und der Querschnittinformation nicht exportiert wurden. Obwohl ein Algorithmus durch das Integral und die Schlüsselkoordinaten des Modells entwickelt werden könnte, ist die erforderte Information wegen topologischer Repräsentation durch BIMfit auch nicht filterbar. Laut der Information in Revit ist das Volumen gleich 6.92 m³.

Das Volumen des ganzen Balkens drückt sich ähnlich wie vom Segment Span 1.3 aus, jedoch bildet sich die Unterkannte des Balkens in Längsansicht eine Kettenlinie ab. Die Funktion eine Kettenlinie lautet

$$y = a \cdot \cos h \frac{x}{a} \qquad (3.18)$$

Oder einfach

$$y = \frac{a\left(e^{\frac{x}{a}} + e^{-\frac{x}{a}}\right)}{2} \qquad (3.19)$$

Unterschiedlich von Gleichung (3.15) ergibt sich daher die Höhe in jedem Halbbalken mit der gemeinsamen Länge des Balkens

$$h = h(x) = h_0 + \frac{a\left(e^{\frac{x}{a}} + e^{-\frac{x}{a}}\right)}{2} \; (0 \le x \le \frac{1}{2} L_{gem}) \qquad (3.20)$$

Hierbei ist a eine Konstante mit den betroffenen Parametern, die Länge einer ganzen Kettenlinie L_{Kette} und die Länge des Balkens L_{gem}

$$\frac{L_{Kette}}{a} = \sinh\frac{\frac{1}{2}L_{gem}}{a} \qquad (3.21)$$

Dann lässt sich das Volumengleichung auf die Form

$$\begin{aligned} V = 2 \cdot \int_0^{\frac{1}{2}L_{gem}} \Bigg(& \Bigg(\Bigg(\Bigg(h_0 + \frac{a\left(e^{\frac{x}{a}} + e^{-\frac{x}{a}}\right)}{2} \Bigg) + (z_{oo} - z_{ou}) + (z_{uo} \\ & - z_{uu}) \Bigg) (y_{02} - y_{01}) + \frac{1}{2}\, k_y \, ({y_{02}}^2 - {y_{01}}^2) \\ & - \Bigg(\sum_{i=1}^{3} \Bigg(h_0 + \frac{a\left(e^{\frac{x}{a}} + e^{-\frac{x}{a}}\right)}{2} \Bigg) (y_{i2} - y_{i1}) \\ & + \frac{1}{2}\, k_y \, ({y_{i2}}^2 - {y_{i1}}^2) \Bigg) + A_S \Bigg) dx \end{aligned} \qquad (3.22)$$

Tabelle 3.6: benötige Informationen

benötige Koordinaten	$z_{oo}, z_{ou}, z_{uo}, z_{uu}, y_{01}, y_{02}, y_{i1}, y_{i2}$
benötige Konstanten	$L_{gem}, h_0, a, k_y, A_S$

Die Gleichung ergibt sich ausschließlich einen Näherungswert, da ein kleiner Teil vom Balken, der auf den Stützen steht, eine flache Unterebene hat. Das Volumen ergebt sich 837 m³ nach den Daten in Revit. Der Längsbewehrungsgrad ist daher gleich 0,83%.

3.2.5 Ermittlung des Spanngliedverlaufs und -alignmentes

Um der Spanngliedverlauf und -alignment zu ermitteln, sollten grundsätzlich deren Geometrie durch mehrere Schlüsselpunkte mittels Class-Graphics in der JAVA-Sprache angezeichnet werden. Die Koordinaten der benötigten Punkte werden durch

IfcCartesianPoint im IFC-Datenmodell erstellt. *IfcCartesianPoint* beschreibt einen Punkt, der durch seine Koordinaten in einem zwei- oder dreidimensionalen rechteckigen kartesischen Koordinatensystem oder in einem zweidimensionalen Parameterraum definiert wird. Die Entität ist in einem zwei- oder dreidimensionalen Raum definiert. *IfcCartesianPoint* ist der Subtyp von *IfcPoint*, der unter *IfcGeometricRepresentationItem* gehört.

Die Form eines Objektes kann durch mehrere geometrischen Repräsentationen definiert werden. Alle Repräsentations-Elemente, die für eine bestimmte Repräsentation genutzt werden, können über ihre geometrische, ihre topologische bzw. eine vordefinierte Repräsentation beschrieben werden.

IfcGeometricRepresentationItem stellt ein geometrisches Repräsentationselement dar, das die zusätzliche Bedeutung hat, eine geometrische Position, Orientierung bwz. beides zu haben. Diese zusätzliche Bedeutung ist dargestellt, da sie ein kartesischer Punkt oder eine Richtung ist, oder sie einen kartesischen Punkt oder eine Richtung in- bzw. direkt referenziert. Ein indirektes Referenzieren auf einen kartesischen Punkt oder eine kartesische Richtung bedeutet, dass ein gegebenes geometrisches Element durch ein bzw. mehrere dazwischen liegende geometrische- oder topologische Elemente den kartesischen Punkt oder die kartesische Richtung referenziert.

Abbildung 3.12: topologisch definierte Punkte der Spannglieder

IfcTopologicalRepresentationItem beschreibt ein topologische Repräsentationselement, das der Supertyp für alle topologischen Repräsentationselemente in der Geometrieressource, bspw. *IfcConnectedFaceSet, IfcEdge, IfcFace, IfcFaceBound, IfcPath, IfcVertex,* bzw. *IfcLoop*. Die topologische Repräsentation wird nicht dierekt genutzt, sondern über die geometrische Repräsentation definiert. D.h., obwohl *IfcCatesianPoint* unter

IfcGeometricRepresentationItem gehört, können die Punkte doch für eine topologische Repräsentation definiert werden.

Im Fall des Projektes wird die Platzierung der Spannglieder im IFC-Datenmodell topologisch repräsentiert (siehe Abbildung 3.12). Jedoch besitzt die geometrische Repräsentation keine Definitionen kartesischer Punkte (siehe Abbildung 3.13). Die Punkte sind in *IfcPolyloop* unter *IfcFace* definiert. Eine Poly-Loop ist eine Schleife mit geraden Kanten, die einen planaren Bereich im Raum begrenzen. Die ist eine Schleife von Genus 1, in der die Schleife durch eine geordnete koplanare Sammlung von Punkten dargestellt wird, die die Scheitelpunkte der Schleife bilden.[6] Die Schleife besteht aus geraden Liniensegmenten, die einen Punkt in der Sammlung mit dem nachfolgenden Punkt in der Sammlung verbinden. Das Abschlusssegment erstreckt sich vom letzten bis zum ersten Punkt in der Sammlung. In diesem Fall werden die Koordinaten die Schlüsselpunkten eines Spanngliedes unmöglich mithilfe BIMfit auszufiltern, da die topologischen Punkte nicht direkt das Objekt geometrisch repräsentiert und BIMfit keine passenden Funktionen besitzt. D.h., der Spanngliedverlauf und dessen Alignment sind ausschließlich durch BIMfit leider NICHT BESCHAFFBAR.

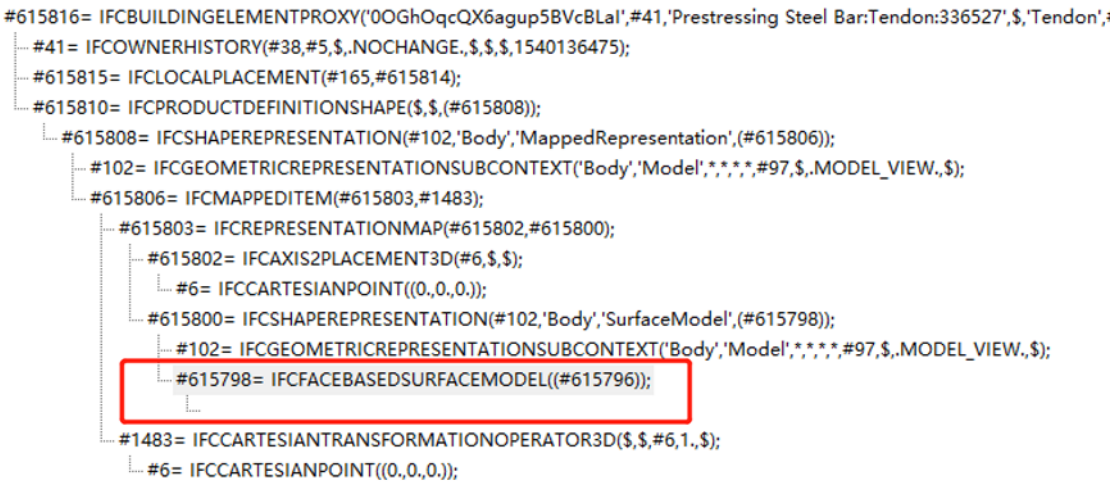


Abbildung 3.13: keine betroffenen Punkte unter geometrischer Repräsentation

Die Ermittlung könnte durchgeführt werden, wenn die Koordinaten durch geometrische Repräsentation definiert würden. Der Subtyp *IfcPolyline* stellt eine begrenzte Kurve von n - 1 linearen Segmenten dar, die durch eine Liste von n Punkten definiert wird. Die kartesischen Punkte werden auch durch *IfcCartesianPoint* ausgedrückt. In diesem Fall könnte die benötigen Koordinaten mittels der Funktionen `getValueOfAttribute` bzw. `getPolylinePoints` in BIMfit selektiert und ausgefiltert werden, dann durch einen entworfenen Algorithmus mithilfe der Class-Graphics in der JAVA-Sprache angezeichnet werden.

4 Zusammenfassung

4.1 Information über Bewehrung und Spannglieder in IFC-Standard

Bezüglich der Bewehrung gehören in IFC alle relevante Klasse zur Superklasse *IfcReinforcingElement*, die die in Beton eingebettete Stäbe, Drähte, Litzen und andere schlanke Elemente beschreibt. Die geometrische Repräsentation von *IfcReinforcingElement* und ihren Subklassen wird von *IfcProductDefinitionShape* angegeben, wodurch mehrere geometrische Repräsentationen möglich sind.

Der Subtyp *IfcReinforcingBar* stellt einen Stahlstab dar, gewöhnlich mit einer vorteilhaften hergestellten Verformung, der in Beton- und Mauerwerkkonstruktion verwendet wird, um zusätzliche Festigkeit bereitzustellt. Ein einzelnes Exemplar dieser Klasse kann einen oder mehrere tatsächliche Bewehrungsstäbe darstellen. Als die einzige Klasse in IFC, die einen Spannglied darstellen können, der Subtyp *IfcTendon* beschreibt ein Stahlelement wie ein Draht, ein Kabel, ein Stab, eine Stange oder eine Litze, dem Beton eine Vorspannung einzuleiten, wenn das Element gespannt wird. Bemerkenswerterweise befindet sich davon bezüglich der Information über Spanngliedoberfläche ausschließlich der Reibungskoeffizient, jedoch kein geometrisches Attribut von Oberfläche im Fall von Spannstahlstab anzuzeigen, ob die Oberfläche glatt bzw. gerippt ist. Der Subtyp *IfcTendonAnchor* stellt den Anker als die Endverbindung für Spannglieder im Beton dar.

Außer vorgenannten Klassen kann ein Bewehrungs- bzw. Spanngliedelement durch *IfcBuildingElementProxy* in einem IFC-Modell erstellt werden. *IfcBuildingElementProxy*, ein Subtyp von *IfcBuildingElement*, spezialisiert das Konzept eines Produkts weiter. *IfcBuildingElementProxy* ist eine Proxy-Definition, die die gleiche Funktionalität wie ein *IfcBuildingElement* bereitstellt, jedoch ohne eine vordefinierte Bedeutung des speziellen Typs des Bauelements zu haben, d.h. *IfcBuildingElementProxy* ist für nicht im IFC-Datenmodell definierte Bauelement verwendbar, damit das Element sowie Modell beliebig erweitert werden. Die Platzierung und Geometrie von einem Proxy-Element können durch kartesisches Koordinatensystem direkt geometrisch oder topologisch repräsentiert, jedoch wird die topologische Repräsentation mittles geometrischen IFC-Klassen realisiert. Die Entitäten beziehen Informationen von GlobalId, Name, Beschreibung usw. ein, die ebenfalls in Modellierung der Bewehrung bzw. Spannglieder gültig sind.

Zusammenfassend unterstützt IFC-Standard gegenwärtig relativ ausreichende Bewehrungsinformation und die notwendige Spanngliedinformation. Aufgrund weniger Stärke der Bewehrung und weniger Bedarf der Spannglieder eines Gebäudes ist die vorhandenen Klasse in IFC-Standard für ein Gebäudemodell genügend, jedoch nicht genug für ein Brückenmodell. Sonstige Information der Bewehrung und Spannglieder eines Brückenmodells kann ausschließlich durch *IfcBuildingElementProxy* erstellt.

4.2 Ermittlung der Bewehrung in IFC-Modell

Die Modellierung der Bewehrung mittels auf BIM basierender Software Revit wird aufgrund der Vielzahl der Verarbeitung in ausschließlich einem Balkensegment durgeführt.

Um die Bewehrung durch die Bewehrungsfunktion in Revit im Brückenmodell einzufügen wird der Basisbauteil für Bewehrung geeignet eingestellt.

Anschließend wird die Bügelbewehrung als eine Bewehrungsgruppe im Balkensegment nach eingegebenen Parametern und Basisabhängigkeiten platziert und durch die Funktion automatisch ergänzt. Da das ausgewählte Segment eine geneigte und weinig gekrümmte Unterfläche aufweist, wird die Bewehrung im Steg und Bodenplatten durch damit korrespondierende Funktionen modelliert. Allerdings können sich entlang der Unterfläche die Bewehrung nicht automatisch anpasst. Die Bodenplattenbewehrung kann zwar in Revit 2018 dadurch nicht mit anpassender Platzierung erstellt werden, was jedoch in Revit 2019 ermöglicht wird. Bezüglich der Stegbewehrung ist die Erstellung in beiden Versionen nicht erfolgreich. Vermutlich eignet sich Revit nicht zügig für die Erkennung des manuell modellierten Hohlkastenmodell. Die Bewehrungsgruppe passt sich wahrscheinlich schwer an einem Tragwerk in Revit, das komplexe Geometrie von Höhlen und Betonkörper besitzt, da gegenwärtig Revit sowieso keine spezifische Funktion besitzt, um einen Hohlkasten zu modellieren.

Das in Revit erstellte Bewehrungsmodell wird ins IFC-Element *IfcReinforcingBar* exportiert, das alle Parameter einbezieht, die in Revit zu Eigenschaften der Bewehrung gehören. Allerdings wird das in Revit gezeigte Elementvolumen in das IFC-Datenmodell nicht als einer der Parameter exportiert. Vermutlich hat Revit ihr eigner Algorithmus, um das Volumen jeweiliges Bauelements auszurechnen, jedoch wird das Ergebnis nicht ins IFC-Modell exportiert. Alle benötigen Informationen einschließlich des Volumens können durch einen Algorithmus mittels BIMfit selektiert und ausgerechnet werden.

Die am Institut für Bauinformatik der TU Dresden entwickelte BIM Filtertoolbox BIMfit basiert auf dem Konzept des generischen Filterframeworks und implementiert eine Grundmenge, die entweder selbstständig, oder in Kombination miteinander für eine komplexe Filteroperation appliziert werden können.

Mithilfe der Funktion `getEntitiesOf` in BIMfit können die allgemeinen Informationen der Bügelbewehrungen selektiert. Die Funktion `getEntitiesOf` dient zur Suchung aller Entitäten der spezifischen Klassentypen, und auch der Subklassen. In diesem Fall werden die Entitäten *IfcReinforcingBar* dadurch gesucht, um alle Bügelbewehrungen auszufiltern. Der Java-Code dafür siehe in Kapitel 3.2.3.

Die Bügelbewehrung wird in einem Segment modelliert. Um den Querbewehrungsgrad zu ermitteln wird das Volumen der Bügelbewehrung und des Betonsegments benötigt. Daher wird erst die Länge und Querfläche der Bewehrungen, die als Attribute im IFC-Datenmodell einbezogen sind, aus den Daten selektiert und ausgefiltert. Durch die Funktion `getValueOfAttribute` werden die Attribut Values, Länge und Querfläche unter der Entität *IfcReinforcingBar* gesucht. Anschließend wird ein Algorithmus entwickelt, um das Volumen durch ausgefilterte Länge und Querfläche auszurechnen. Genauer Algorithmus siehe Kapitel 3.2.3. Das Ergebnis 0,51 m³ kommt dadurch heraus.

Angesichts des Volumens des Segments stellt keine direkte oder direkt relevante Information zur Verfügung. Das Konzept zur Beschaffung des Volumens muss durch kartesische Koordinaten benötigter Punkte entworfen werden. Die entworfene Gleichung dafür siehe Kapitel 3.2.4. Alle Koordinaten werden als *IfcCartesianPoint* im proxybasierten

IFC-Modell dargestellt, die die Subklassen dem Schema von *IfcGeometryResource* gehört, jedoch in dem Fall des Projektes topologisch repräsentiert werden. Daher sind die benötigten Koordinaten durch BIMfit nicht beschaffbar, da die benötigten Punkte nicht automatisch durch einen Algorithmus mittels der Funktionen von BIMfit erkannt, unterschieden und gefiltert werden können.

4.3 Ermittlung der Spannglieder in IFC-Modell

Zur Ermittlung der Spannglieder im IFC-Modell werden die Längsbewehrungen im Steg als Spannglieder modelliert. Revit besitzt keine spezifische Funktion zur Modellierung eines Spannglieds, deshalb wird die Spannglieder als eine neue Familie erstellt, die auf allgemeinem Modell als ihr Vorlagendatei basiert. Alle notwendigen Parameter werden währen Erbauens der neuen Familie auf der Eigenschaftsliste unter statischer Berechnung eingegeben. Zur Abminderung des aus Einwirkungen auftretenden Biegemoments ordnen sich die Spannglieder an Zugspannungen abtragender Seite im Balkenmodell in der Form einer Kurve an, die sich im Feld im oberen und an den Stützen im unteren Teil des Balkens ausrichtet.

Nach dem Export werden das Brücken- bzw. Spanngliedmodell in IFC-Daten durch *IfcBuildingElementProxy* erstellt. Die Geometrie des Brücken- und der Spanngliedmodells ist topologisch repräsentiert. Die zuvor eingegebenen Parameter der Spannglieder sind jedoch nach dem Export verloren. Zum Versuch wird anschließend mehrere Parameter auf der Eigenschaftsliste unter ID-Daten in Revit eingegeben, von den 3 Parameter: Hersteller, Typenkommentare und URL erfolgreich in IFC exportiert sind.

Merkwürdig wurden die Spannglieder durch BIM/CAD-Programm Allplan teilweise modelliert, und ein weiter Exportversuch ist jedoch schiefgegangen, möglichweise aufgrund der problematischen Installierung von Allplan wegen Softwarekonflikts. Es könnte auch sein, dass das aus Revit exportierte IFC-Datenmodell basierend auf für Bewehrung nicht geeignetem Modell in Allplan problematisch importiert würde, und demnach gegen darin modellierte Spannglieder ein Konfliktfehler entstünde. Weitere Ermittlung des genauen Grunds wird während des Projektes nicht durchgeführt.

Zur Filterung aller relevanten Information der Spannglieder werden mittels BIMfit-Funktion `getEntitiesWithAttributeValue` alle Entitäten von den Spanngliedern gesucht, die im spezifizierten Attribut einen gegebenen Wert aufweisen. Die Funktion ist unabhängig von der Entitätsklasse, jedoch nur für die gegebene Klasse. Die Einheiten von Java-Code siehe Kapitel 3.2.3.

Angesichts des Längsbewehrungsgrad wird das Volumen der Spannglieder und das Volumen des ganzen Balkens gebraucht. Allerdings wurden die kurvenförmigen Spannglieder auch durch *IfcBuildingProxy* im IFC-Datenmodell erstellt, worin die Länge- und der Querschnittinformation nicht exportiert wurden. Im Fall des Projektes wird die Platzierung der Spannglieder im IFC-Datenmodell topologisch repräsentiert. Jedoch besitzt ihre geometrische Repräsentation keine Definitionen kartesischer Punkte. Die Punkte sind in *IfcPolyloop* unter *IfcFace* definiert. Eine Poly-Loop ist eine Schleife mit geraden Kanten, die einen planaren Bereich im Raum begrenzen. Obwohl ein Algorithmus durch das Integral und die Schlüsselkoordinaten des Modells entwickelt werden könnte, ist die

erforderte Information durch BIMfit auch nicht filterbar, da die topologischen Punkte nicht direkt das Objekt geometrisch repräsentiert und BIMfit keine passenden Funktionen besitzt. Ebenfalls sind daher der Spannliedverlauf und dessen Alignment ausschließlich durch BIMfit leider NICHT BESCHAFFBAR.

4.4 Mögliche Optimierung und Ausblick

Während des Projektes traf die Ermittlung auf einige Schwierigkeiten und Probleme in verschiedenen Phasen. In diesem Teil werden mehrere Ideen dafür diskutiert.

In der Modellierungsphase wird die Bügelbewehrungsgruppe in variable Geometrie nicht perfekt erstellt. Zur Optimierung der Anpassung kann die in Revit 2019 neu gestellte Kombinationsfunktion „Freiform-Bewehrung“ in „Bewehrungsgruppe“ benutzt. Freiformbewehrung folgt den unterschiedlichen Geometrien der Basisbauteilüberdeckungs-Referenzen. Die Schwierigkeit liegt ausschließlich daran, dass das Programm den manuell modellierten Hohlkasten erkennt oder nicht.

In der Exportphase sind alle eingegebenen Parameter das proxybasierte Spannliedmodell verloren. Wegen der Ähnlichkeit zwischen Stahlstäben und Spannstäben könnte das Modell auf Bewehrungsmodell als sein Vorlagendatei in Revit erstellt, um den Verlust notwendiger Daten zu vermeiden. Alle zusätzlichen Informationen der Spannglieder wie Vorspannkräfte können als neue Eigenschaften auf die Liste eingefügt werden. Jedoch in diesem Fall wird das Spannliedmodell in *IfcReinforcingBar* erstellt, nicht *IfcTendon*.

Im Projekt wird die Geometrie des aus Revit exportierten proxybasierten Modells topologisch repräsentiert, sodass die benötigten Koordinaten der Schlüsselpunkte mittels BIMfit nicht ausgefiltert werden können. Es hängt von den zugrundeliegenden Datenstruktur der Ausgangssoftware ab, wie sich die Geometrie eines Objektes repräsentiert. Die Software muss jedoch für den Export alle benötigten Geometriedarstellungsansätze des IFC-Formates unterstützen. Die Modellierung könnte daher durch ein Zusatzmodul ausgeführt werden, das Modellierung des Brückenelements unterstützt. Solche Module stehen im Autodesk Store zu Verfügung. Alternative könnte ein anderes Programm verwendet, wie bspw. die Software Tekla Structure, die eine offene, präzise und dynamische 3D-Umgebung für die Bauindustrie bietet und in der ein IFC-Datenmodell als Referenz importiert werden darf. Möglichweise könnte das dadurch erstellte Spannliedmodell in *IfcTendon* exportiert werden.

Die in Revit 2017, 2018 und 2019 neu entwickelten Bewehrungsfunktionen sind zwar nicht perfekt jedoch schon ein großer Schritt zur Bewehrungsmodellierung. Die würden in gewissem Maße in kommenden Versionen verbessert. In Anbetracht der notwendigen Rolle, die Brücken in Architektur und Bauwesen spielen, würde Revit bestimmt in das Bereich eingehen, sodass mehr relevante Funktionsmodule entwickelt würden. Bezüglich IFC hat BuildingSMART schon einige Kandidaten für die noch unveröffentlichte *IFC 2x4*-Version freigegeben, die vielfältigere Klassen für reichlichere Information einbeziehen könnte. Die Kompatibilität und Austauschbarkeit zwischen verschiedenen BIM-Softwares und IFC-Datenmodell werden im Laufe der Entwicklung allmählich optimiert.

5 Verzeichnisse

5.1 Abbildungsverzeichnis

5.2 Tabellenverzeichnis

5.3 Literaturverzeichnis

[1] Borrmann, André/ König, Markus/ Koch, Christian/ Jakob, Beetz: „Building Information Modelling. Technologische Grundlage und industrielle Praxis“, Springer Vieweg, Wiesbaden, 2015.

[2] Scherer, Raimar Joseph: „Die BIM Methode“, Vorlesung BIW4-70, TU-Dresden, Institut für Bauinformatik, 2015.

[3] Robinson, Clive: “Structural BIM. discussion, case studies and latest developments”, in: “The Structural Design of Tall and Special Buildings”, 16, Wiley Interscience, USL www.interscience.wiley.com, DOI: 10.1002/tal.417, UK, 2007, S. 519-533.

[4] Eastman, Chuck/ Teicholz, Paul/ Sacks, Rafael/ Liston, Kathleen: “BIM handbook. a guide to building information modeling for owners, managers, designers, engineers, and contractors”, Wiley, Hoboken/ NJ, 2008.

[5] Köbler, Kalle: „Untersuchung der IFC-gestützten Modellübertragung zur Ableitung von Modellierempfehlungen für Architekten“, TU-München, Lehrstuhl für Computergestützte Modellierung und Simulation, 2017.

[6] BuildingSMART. Model - Industry Foundation Classes (IFC), URL http://www.buildingsmart-tech.org/.

[7] Scherer, Raimar Joseph: „Systementwicklung. IFC-Objekt-Datenmodell II“, Vorlesung WP4-70, TU-Dresden, Institut für Bauinformatik.

[8] Mehlhorn, Gerhard/ Curbach, Manfred: „Handbuch Brücken. Entwerfen, Konstruieren, Berechnen, Bauen und Erhalten“, 3. Auflage, Springer Vieweg, Wiesbaden, 2014.

[9] Weidemann, Hans: „Brückenbau. Stahlbeton- und Spannbetonbrücken“, Weiner, Düsseldorf, 1982.

[10] Lamprecht, Heinz-Otto: „Beton-Lexikon“, Beton-Verl., Düsseldorf, 1990.

[11] Schneider, Klaus-Jürgen/ Goris, Alfons/ Albert, Andrej: „Bautabellen für Ingenieure. mit Berechnungshinweisen und Beispielen“, 22. Auflage, Bundesanzeiger Verlag, Köln, 2016.

[12] Werkle, Horst: „Nichtlineare Schnittgrößenermittlung“, Vortrag auf dem Seminar „DIN 1045-neu“, Fachhochschule Konstanz, 2014.

[13] Eibl, Josef: „Externe Vorspannung und Segmentbauweise“, Universität Karlsruhe (TH), Workshop „Externe und Verbundlose Vorspannung – Segmentbrücken“, Ernst, Berlin, 1998.

[14] Deutsches Institut für Normung e.V. (Hrsg.): „DIN-Fachbericht 102:2009-03. Betonbrücken“, Dt. Ausgabe, Beuth Verlag, Berlin, 2009, Kap. III, Abschnitt 2.

[15] Schaich, Mike/ Bleicher, Achim: „Konstruktiver Ingenieurbau II“, Vorlesungsskript, 2. Auflage, TU-Berlin, Fachgebiet Massivbau, Berlin, 2008.

[16] Krautwald, W.: „Extern vorgespannte Brücken. Erfahrungsbericht eines Bauausführenden“, Workshop „Externe Vorspannung und Segmentbauweise“, Karlsruhe, 1998.

[17] Kolísko, Jiří/ Vacek, Vítězslav/ Řeháček, Stanislav: „Repeated diagnostics of maintained prestressed bridge structure, interpretation of changes in relation to durability", in: „Key Engineering Materials", doi:10.4028/www.scientific.net/KEM.691.366, die Schweiz, 2016.

[18] „Revit 2018 Produkthilfe", Autodesk, URL http://help.autodesk.com/view/RVT/2018/DEU/

[19] Scherer, Raimar Joseph/ Schapke, Sven-Eric: „Informationssysteme im Bauwesen 1. Modelle, Methoden und Prozesse", Spring Vieweg, Berlin/ Heidelberg, 2014.

[20] Hamdan, Al-Hakam: „Anwendung von BIMfit und ifcQL an einem BIM-Modell", TU-Dresden, Institut für Bauinformatik, 2016.

[21] Scherer, Raimar Joseph: „Modell-basiertes Arbeiten. Filtern mit BIMfit", Übung 3 WP4-70, TU-Dresden, Institut für Bauinformatik.

[22] Wülfing, Alexander/ Windisch, Ronny/ Baumgärtel, Ken: „BIMfit - Ein modulares Softwarewerkzeug zur Abfrage und Filterung von Gebäudeinformationsmodellen", in: Hegemann, Felix/ Kropp, Christopher/ Rahm, Tobias/ Szczesny, Kamil (Hg.): „Forum Bauinformatk 2012", Europäischer Univ.vlg., Bochumer Univ.vlg., Bochum, 2012, S. 9-16.

[23] Wülfing, Alexander: „BIMFIlterToolbox – Tutorial V. 1.0", 2012.

6 Anlagen

Anhang 1: Ansicht der Pavel Wonka Brücke

Anhang 2: Querschnitt Pavel Wonka Brücke im Feld und auf den Stützen

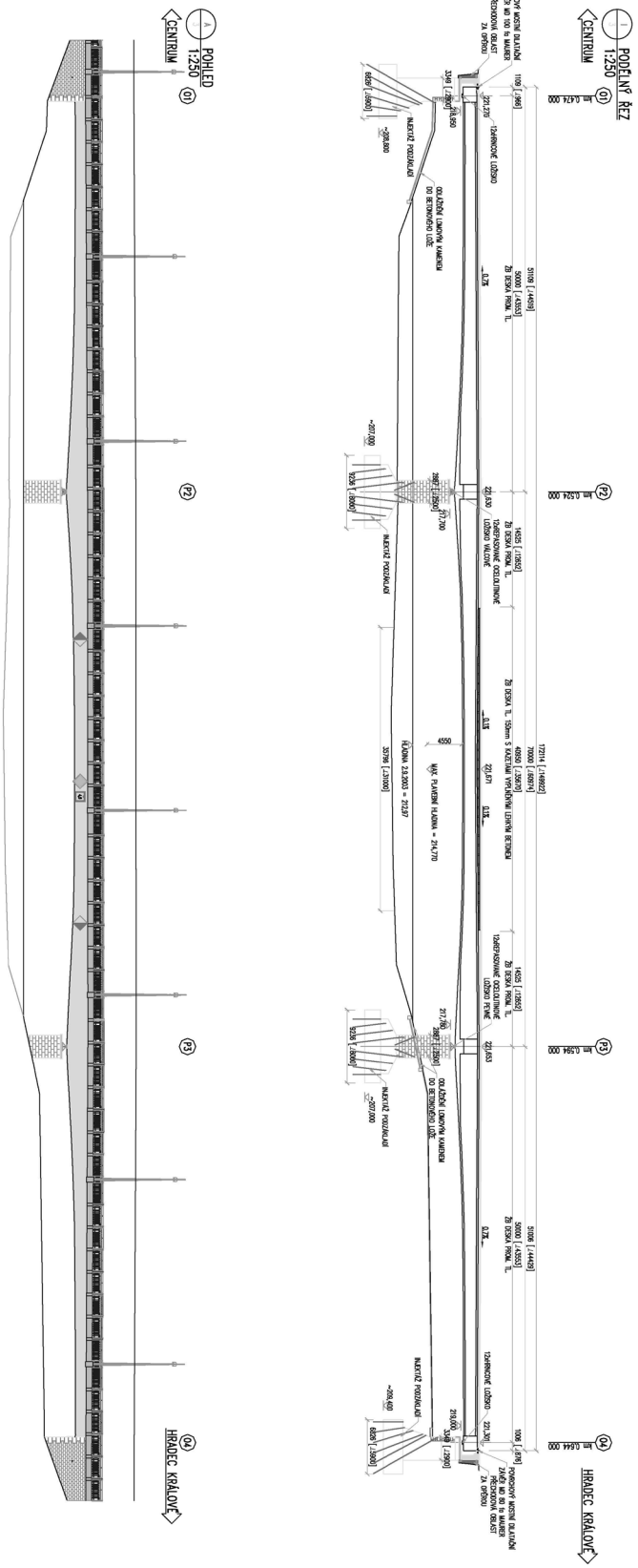
Anhang 3: Algorithmus von filterReinforcement und filterTendon

Anhang 4: main-Methode von filterReinforcement

Anhang 5: main-Methode von filterTendon

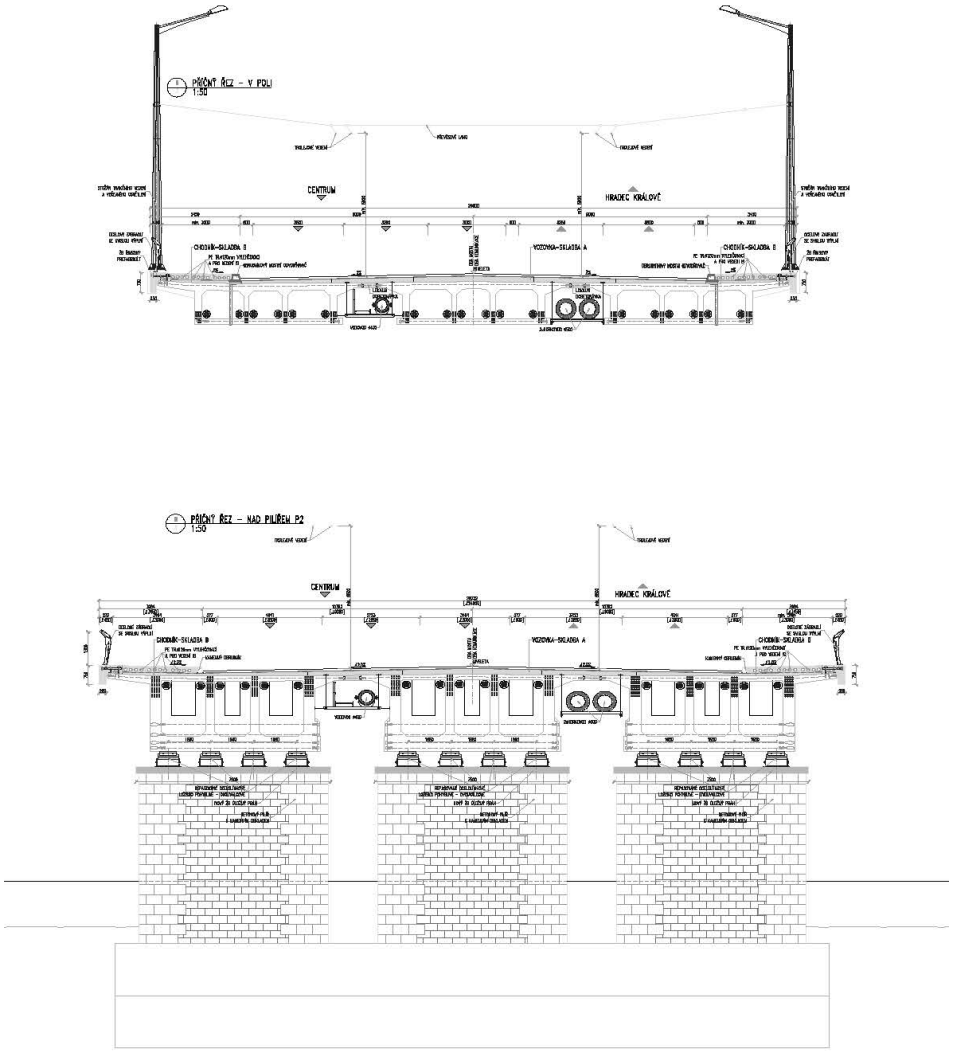
Anhang 6: Algorithmus von Volumen der Bewehrung

Anhang 7: main-Methode des Algorithmus von Volumen der Bewehrung



Anhang 1: Ansicht der Pavel Wonka Brücke

Quelle: Institut für Bauinformatik, TU-Dresden, im Rahmen des Projektes cyberBridge.



Anhang 2: Querschnitt Pavel Wonka Brücke im Feld und auf den Stützen

Quelle: Institut für Bauinformatik, TU-Dresden, im Rahmen des Projektes cyberBridge.

Filter.java

```
1 package filter;
2
3
4 import de.tudresden.bau.cib.exceptions.parser.ParsingException;
5 import de.tudresden.bau.cib.model.StepDataModel;
6 import de.tudresden.bau.cib.parser.StepParser;
7 import jsdai.SIfc2x3.EIfcbuildingelementproxy;
8 import jsdai.SIfc2x3.EIfcreinforcingbar;
9 import jsdai.lang.EEntity;
10 import jsdai.lang.SdaiException;
11
12 public class Filter {
13     private StepDataModel model;
14     public void createModel(String ifcfile) {
15         StepParser parser = new StepParser("C:\\Users\\issen\\eclipse-workspace\
   \Projektarbeit");
16         try {
17             model = parser.parse(ifcfile);
18         } catch (ParsingException e) {
19             System.out.println("Loading File failed");
20             e.printStackTrace();
21         }
22     }
23
24     public void filterReinforcement() {
25         try {
26             EIfcreinforcingbar[] barList = model.getEntitiesOf(EIfcreinforcingbar.class);
27             for (int i=0; i<barList.length; i++) {
28                 System.out.println(barList[i]);
29             }
30         } catch (SdaiException e) {
31             System.out.println("Filtering ReinforcingElements failed");
32             e.printStackTrace();
33         }
34     }
35
36     public void filterTendon() {
37         try {
38             EEntity[] tendonList =
   model.getEntitiesWithAttributeValue(EIfcbuildingelementproxy.class, "ObjectType", "Tendon");
39             for (int i=0; i<tendonList.length; i++) {
40                 System.out.println(tendonList[i]);
41             }
42         } catch (SdaiException e) {
43             System.out.println("Filtering TendonElements failed");
44             e.printStackTrace();
45         }
46     }
47 }
48
```

Anhang 3: Algorithmus von filterReinforcement und filterTendon

main.java

```java
package test.main;

import filter.Filter;

public class main {

    public static void main(String[]args) {
        Filter filter = new Filter();
        filter.createModel("C:\\Users\\issen\\eclipse-workspace\\Bridge.ifc");
        filter.filterReinforcement();
    }
}

```

Anhang 4: main-Methode von filterReinforcement

main.java

```java
package test.main;

import filter.Filter;

public class main {

    public static void main(String[]args) {
        Filter filter = new Filter();
        filter.createModel("C:\\Users\\issen\\eclipse-workspace\\Bridge.ifc");
        filter.filterTendon();
    }
}

```

Anhang 5: main-Methode von filterTendon

Filter.java

```
1 package filter;
2
3 import de.tudresden.bau.cib.exceptions.parser.ParsingException;
4 import de.tudresden.bau.cib.model.StepDataModel;
5 import de.tudresden.bau.cib.parser.StepParser;
6 import jsdai.SIfc2x3.EIfcreinforcingbar;
7 import jsdai.lang.SdaiException;
8
9
10 public class Filter {
11     private StepDataModel model;
12     public void createModel(String ifcfile) {
13         StepParser parser = new StepParser("C:\\Users\\issen\\eclipse-workspace\
   \Projektarbeit");
14         try {
15             model = parser.parse(ifcfile);
16         } catch (ParsingException e) {
17             System.out.println("Loading File failed");
18             e.printStackTrace();
19         }
20     }
21
22
23     public void filterReinforcementValue() {
24         try {
25             EIfcreinforcingbar[] barList1 = model.getEntitiesOf(EIfcreinforcingbar.class);
26             double sumVolume = 0;
27             for (int i=0; i<barList1.length; i++) {
28                 double crossSection = (double) model.getValueOfAttribute
   (barList1[i],"CrossSectionArea",false);
29                 double length = (double) model.getValueOfAttribute(barList1[i], "BarLength",
   false);
30                 double volume = crossSection * length/1000;
31
32                 sumVolume = sumVolume + volume;
33             }
34             System.out.println(sumVolume);
35         } catch (SdaiException e) {
36             System.out.println("Filtering ReinforcingElementValue failed");
37             e.printStackTrace();
38         }
39     }
40 }
41
42
```

Anhang 6: Algorithmus von Volumen der Bewehrung

main.java

```java
package test.main;

import filter.Filter;

public class main {

    public static void main(String[]args) {
        Filter filter = new Filter();
        filter.createModel("C:\\Users\\issen\\eclipse-workspace\\Bridge.ifc");
        filter.filterReinforcementValue();
    }
}

```

Anhang 7: main-Methode des Algorithmus von Volumen der Bewehrung