Einbettung von prozeduralem Wissen in Produktdatenmodelle am Beispiel der Richtlinien für die Anlage von Landstraßen mithilfe der Programmiersprache IFCPL

Julian Amann[1]

[1]Lehrstuhl für Computergestützte Modellierung und Simulation · Ingenieurfakultät Bau Geo Umwelt · Leonhard Obermeyer Center · Technische Universität München · Arcisstraße 21 · 80333 München · julian.amann@tum.de

Zusammenfassung

In dieser Arbeit soll am Beispiel der Richtlinien für die Anlage von Landstraßen (kurz RAL) gezeigt werden, wie diese in ein IFC-gestütztes Produktdatenmodell eingebettet werden und so herstellerneutral in verschiedenen BIM-Softwareanwendungen (wie z. B. einer Trassierungssoftware) verwendet werden können. Dabei wird auf eine selbstentwickelte Programmiersprache (mit dem Namen IFCPL) und Interpreter zurückgegriffen, die es einem Domänenexperten erlaubt, Regeln der RAL sowohl möglichst flexibel als auch allgemeingültig zu beschreiben und abprüfen zu können.

Abstract

This work describes how guidelines such as the “Richtlinien für die Anlage von Landstraßen” (short RAL) can be integrated into an IFC based product model. This way guidelines can be exchanged in a vendor-neutral way between different BIM applications (e.g. Alignment Design Software). In order to achieve this goal a programming language (IFCPL) and a corresponding interpreter has been developed which enables a domain expert to define rules of the RAL in a general and flexible way and allows automatic checking of defined rules.

1 Einführung

1.1 Produktdatenmodelle für den Infrastrukturbau auf dem Vormarsch

Infrastrukturbauwerke spielen eine wichtige Rolle in unserem täglichen Leben. Verkehrswege wie Straßen oder Gleiswege, Brücken- oder Tunnelbauten bilden die Elemente unserer modernen Verkehrsinfrastruktur. Der Lebenszyklus eines Infrastrukturbauwerks startet mit den ersten Planungsaktivitäten und endet mit der Demontage, dem Abriss, der Entsorgung und Wiederverwertung. In der Nutzungszeit dazwischen sind Revisionsarbeiten, Wartungsarbeiten, Reinigungsarbeiten, Sanierungsarbeiten und viele andere Aktivtäten erforderlich.

Seit der Verfügbarkeit von Computern werden die einzelnen Lebenszyklusphasen eines Infrastrukturbauwerks durch Software unterstützt. Beispielsweise setzte bereits Leonard Obermeyer (Gründer der Unternehmensgruppe Obermeyer) Ende der 60er Jahre Computer ein,

um Statikberechnungen durchzuführen (WIKIPEDIA 2015). Inzwischen hat sich einiges getan und es wurden in Arbeiten wie dem ForBAU-Projekt (BORRMANN ET AL. 2009, GÜNTHNER, W. A., BORRMANN 2011) oder in der Dissertation (KAMINSKI 2010) untersucht, wie genau der gesamte Lebenszyklus eines Infrastrukturbauwerks durchgängig von der Planung bis zur Wartung mittels des Building Information Modelling (BIM) unterstützt werden kann und wo konkret Verbesserungsmöglichkeiten und –potentiale bestehen.

Um Infrastrukturbauwerke durch computergestützte Werkzeuge zu beschreiben und verwalten zu können, werden Produktdatenmodelle benötigt. Diese erfassen sowohl die Geometrie als auch die Semantik von Produkten und den Teilen, aus denen sie bestehen, in einem meist objektorientierten Datenmodell. Diese wurden entwickelt, um einen hochqualitativen Datenaustausch mit möglichst geringem Datenverlust und hoher Interoperabilität zwischen verschiedenen Akteuren und Softwareprodukten zu gewährleisten.

In den letzten Jahrzenten wurden zahlreiche Datenmodelle für den Infrastrukturbereich vorgeschlagen, manche wurden standardisiert, andere verließen niemals das Entwicklungsstadium. Derzeit erlebt insbesondere der Bereich Infrastrukturbau einen Boom von Produktdatenmodellen. Beispielsweise wird von dem Open Geospatial Consortium momentan das Produktdatenmodell InfraGML entwickelt, LandXML wird derzeit aktualisiert und weiterentwickelt und auch bei buildingSMART sind Projekte für den Straßen- (IFC-Road), Gleis- (IFC-Rail), Brücken- (IFC-Bidge) und Tunnelbau (IFC-Tunnel) geplant.

Die wirtschaftliche Motivation für diese Entwicklungen liegt klar auf der Hand: Internationale Standards erweitern den Markt für die Softwarehersteller und sorgen für mehr Wettbewerb. Gleiches gilt auch für die Ausschreibung und Vergabe von Planungs- und Bauvorhaben.

1.2 Das Produktdatenmodell IFC-Alignment

Im folgendem soll kurz das IFC-Alignment-Produktdatenmodell (Amann et al. 2014b) vorgestellt werden, das als Grundlage für diese Arbeit dient. Im Prinzip handelt es sich dabei um eine Erweiterung von IFC 4, die mit dem Erscheinen einer der nächsten IFC-Versionen offizieller Bestandteil der IFC werden soll.

IFC-Alignment dient zur Beschreibung einer Straßen- bzw. Bahntrassierung. Das Modell basiert auf der etablierten Herangehensweise, die einen Entwurf in Lageplan und Höhenplan (Gradiente) vorsieht. Das zugrundeliegende konzeptionelle Modell (Amann et al. 2014b) wurde in Zusammenarbeit mit dem OpenGIS Consortium (OGC) umgesetzt. Basierend auf diesem Modell wurde IFC-Alignment entwickelt. Im Rahmen von OGC soll zukünftig ein Standard namens InfraGML entworfen werden, der auf der Geography Markup Language (GML) beruht. Dabei dient ebenfalls das von buildingSMART und OGC gemeinsam entwickelte konzeptionelle Modell als Grundlage. Der Entwurf von OGC sieht allerdings neben der reinen Trassierung auch noch weitere Anwendungsfälle vor, beispielsweise die Landvermessung oder Liegenschaftsverwaltung, und wird im Unterschied zu IFC-Alignment für die Abbildung des Bestands in GIS-Systemen verwendet werden. Durch die Zusammenarbeit von buildingSMART und OGC, die das gemeinsame konzeptionelle Modell auf Basis von UML entwickelt haben, sollen die Harmonisierung zwischen beiden Standardisierungsvorhaben gesichert und eine vielversprechende Verbindung zwischen der BIM- und GIS-Welt geschaffen werden.

Im Folgenden wird das Datenmodell IFC-Alignment genauer betrachtet. Abbildung 1 zeigt einen Überblick über die neu eingeführten Entitäten in das IFC4-Schema.

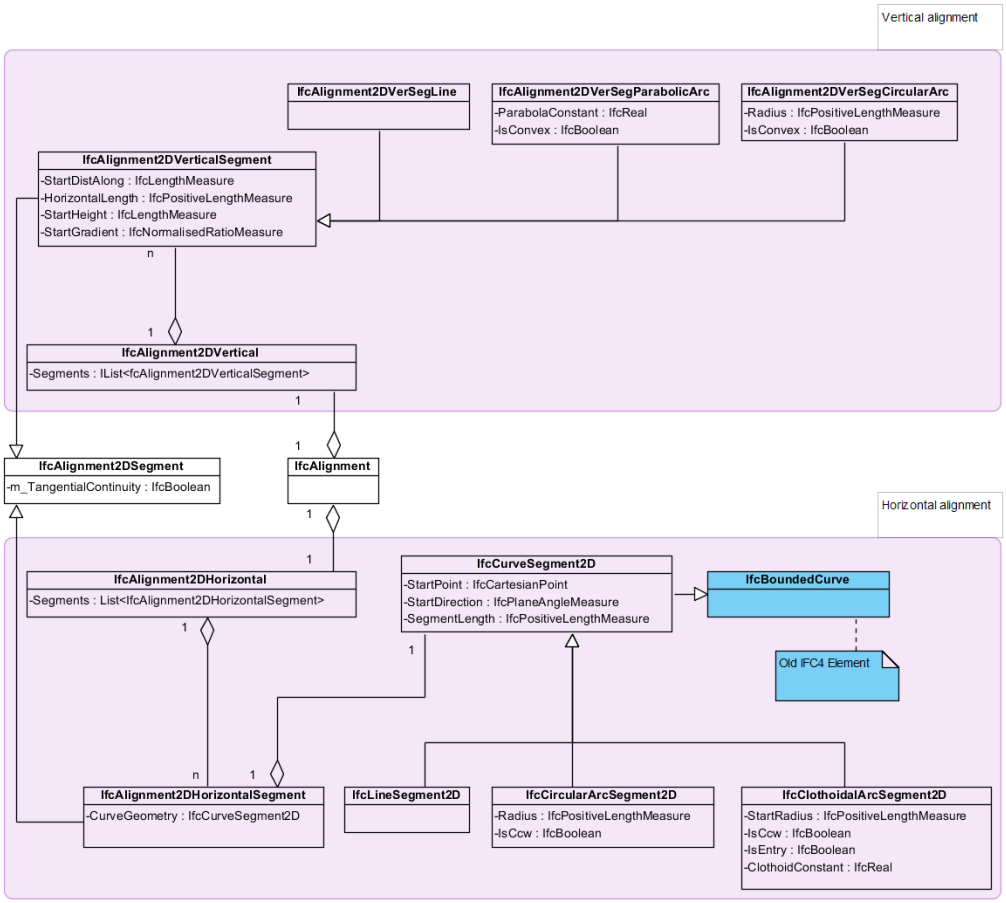

Abb. 1: IFC-Alignment als UML-Klassendiagram mit den wichtigsten Elementen

Als zentrales Element wurde die Klasse *IfcAlignment* eingeführt. Diese referenziert den Lageplan (*IfcAlignment2DHorizontal*) und den Höhenplan (*IfcAlignment2DVertical*). Der Lageplan selbst besteht aus den bekannten Trassierungselementen Gerade (*IfcLineSegment2D*), Kreisbogen (*IfcCircularArcSegment2D*) und Klothoide (*IfcClothoidalArcSegment2D*). Neben der Klothoide gibt es derzeit keine weiteren Übergangskurven im Standard, jedoch werden in den geplanten Erweiterungen IFC-Road bzw. IFC-Rail zukünftig weitere Übergangskurven wie z.B. der Blossbogen definiert werden. Gerade, Kreisbogen und Übergangskurve besitzen eine Untermenge von gleichen Datenattributen. Daher wurde eine gemeinsame Basisklasse namens *IfcCurveSegement2D* eingeführt, die gemeinsame Datenattribute wie Startposition/-richtung und Segmentlänge beinhaltet. Die Klasse *IfcAlignment2DHorizontal* selbst besteht aus einer geordneten Liste von *IfcAlignment2DHorizontalSegment*-Elementen,

die jeweils auf ein konkretes Trassierungselement verweisen (Gerade, Kreisbogen oder Klothoide).

1.3 Dynamisches statt statisches Wissen

Ein Produktdatenmodell spiegelt das Wissen über ein Produkt, wie z. B. eine Straße, wider. Derzeitige Produktdatenmodelle aus dem Umfeld des Infrastrukturbaus sind im Wesentlichen auf die statischen Aspekte (im Folgenden als plain data oder auch als einfache Daten bezeichnet) von Produktdaten ausgelegt. Beispielsweise werden in dem Produktdatenmodell IFC-Alignment zu einer Übergangkurve wie einer Klothoide u. a. geometrische Eigenschaften wie Startpunkt, Startrichtung, Länge, Startradius, Orientierung (gegen oder im Uhrzeigersinn) oder Klothoidenkonstante gespeichert. Daneben können auch weitere semantische Eigenschaften wie beispielsweise die Zugehörigkeit (Relationen) zu einer Trasse, einem Projekt und einem geographischen Referenzsystem gespeichert werden. Abbildung 2 zeigt die Daten, die bei einer Geraden (*IfcLineSegment2D*), einem Kreisbogen (*IfcCircularArcSegement2D*) und einer Klothoide (*IfcClothoidalArcSegment2D*) im IFC-Alignment-Standard ausgetauscht werden.

P6 IFC Alignment		Georeferencing	3512978,55	5516154,935						
Description	IfcCurveSegment2D type	StartPoint x	StartPoint y	StartDirection	SegmentLength	Radius	StartRadius	IsCcw	IsEntry	ClothoidConstant
KREIS1	IfcCircularArcSegment2D	59,4093	44,1393	5,246193178	30,48455731	15,00004037		WAHR		
KREIS1	IfcCircularArcSegment2D	84,9063	43,6077	0,995306222	31,41598417	15,0000179		WAHR		
KREIS1	IfcCircularArcSegment2D	73,1004	66,7512	3,089702668	32,34733551	14,99998973		WAHR		
A1	IfcLineSegment2D	-184,362	41,1065	5,070085217	80,86091947					
A1	IfcClothoidalArcSegment2D	-156,051	-34,6364	5,070078518	12,7657		NIL	WAHR	WAHR	19,56964486
A1	IfcCircularArcSegment2D	-150,757	-46,2242	5,282844626	39,35834523	30,00005917		WAHR		
A1	IfcClothoidalArcSegment2D	-116,31	-58,5789	0,31159524	13,3333		30	WAHR	FALSCH	19,999975
A1	IfcLineSegment2D	-104,387	-52,6753	0,533824083	205,2693204					
A1	IfcLineSegment2D	72,3224	51,7717	1,25114073	9,887040915					
A1	IfcCircularArcSegment2D	75,4293	61,1579	1,251138699	21,96498274	38,00006707		FALSCH		
A1	IfcCircularArcSegment2D	87,8144	78,9283	0,673113454	12,61254363	99,99994132		FALSCH		
A1	IfcLineSegment2D	98,145	86,1494	0,546986411	23,64781526					
KREIS2	IfcCircularArcSegment2D	79,8895	46,8989	4,14030379	18,3215068	9,000041403		FALSCH		
KREIS2	IfcCircularArcSegment2D	64,5746	47,1922	2,104590177	21,81802509	8,999945947		FALSCH		
KREIS2	IfcCircularArcSegment2D	75,1505	60,3155	5,963535915	16,40898422	8,999988579		FALSCH		
BAUSTR	IfcLineSegment2D	-154,153	-39,3048	6,204220206	10,64487074					
BAUSTR	IfcCircularArcSegment2D	-143,541	-40,1445	6,20421657	45,70679716	29,9999923		WAHR		
BAUSTR	IfcCircularArcSegment2D	-111,413	-14,0141	1,444595286	30,56086349	12,00003372		FALSCH		
BAUSTR	IfcLineSegment2D	-88,8024	-10,1042	5,181036406	4,749601914					
PROV2	IfcCircularArcSegment2D	63,0066	65,0031	5,093363	11,11893336	60,00002728		WAHR		
PROV2	IfcLineSegment2D	68,0708	55,1223	5,278679223	25,01952012					
PROV2	IfcCircularArcSegment2D	81,494	34,0084	5,278677807	17,28013318	50,00000941		WAHR		
PROV2	IfcLineSegment2D	93,0762	21,3004	5,624282501	33,5807428					
PROV2	IfcCircularArcSegment2D	119,627	0,74065	5,624280405	13,34448655	25,0000259		WAHR		
PROV2	IfcCircularArcSegment2D	131,814	-4,29735	6,158058656	14,42827434	23,49992549		FALSCH		

Abb. 2: Daten, die bei einer Geraden (*IfcLineSegment2D*), einem Kreisbogen (*IfcCircularArcSegement2D*) und einer Klothoide (*IfcClothoidalArcSegment2D*) im IFC Alignment Standard ausgetauscht werden

Konkrete Berechnungsvorschriften, im Folgenden als dynamische Eigenschaften (oder dynamic data oder intelligente Daten) bezeichnet, die aus gegebenen Daten neue Ergebnisse berechnen, fehlen im Datenmodell. Denkbar wäre hier z. B. die Möglichkeit einer dynamischen Positionseigenschaft, die zu einem gegebenen Stationierungswert (horizontal gemessene Abwicklungslänge einer Strecke eines Referenzpunkts) die Gauß-Krüger-Koordinaten im Lageplan berechnet (vgl. hierzu (Amann et al. 2014a)). Diese Berechnungsvorschrift ist für jeden Übergangsbogen unterschiedlich.

Die Einführung solcher Konzepte ist zwar in existierenden Standards sichtbar, aber noch nicht deutlich ausgeprägt. Über dieses Beispiel hinaus ergeben sich eine Reihe weiterer inte-

ressanter Anwendungsfälle. Beispielsweise kann man dieses Konzept nutzen, um Bauvorschriften zu erfassen (z. B. ein Geländer an einer Brücke muss eine bestimmte Mindesthöhe besitzen). Auch firmeninterne Regeln und Richtlinien lassen sich so abbilden.

In dieser Arbeit soll am Beispiel der Richtlinien für die Anlage von Landstraßen (kurz RAL) gezeigt werden, wie diese in ein IFC-gestütztes Produktdatenmodell eingebettet werden können und so herstellerneutral in verschiedenen Konstruktionssystemen verwendet werden. Dabei wird auf eine selbstentwickelte Programmiersprache (IFCPL) und auf einen selbstentwickelten Interpreter zurückgegriffen (siehe dazu auch (Amann et al. 2014a)), um Regeln der RAL sowohl möglichst flexibel als auch allgemeingültig zu beschreiben und abprüfen zu können.

2 Ein intelligentes Produktdatenmodell

2.1 Offene Standards erleichtern die Zusammenarbeit

Bei der Planung von komplexen Infrastrukturbauten wie Brücken, Tunnel oder Straßen sind viele verschiedene Experten involviert. Der Planungsprozess hat sich weg vom Zeichenbrett hin zur Planung mithilfe computergestützter Modellierungssoftware entwickelt. Bei der Konstruktion und Planung von Infrastrukturbauten entstehen viele Daten. Durch Datenaustausch können diese Informationen geteilt werden. Ein erhebliches Problem, das dabei entsteht, ist, dass die Daten oft nur in einem proprietären Datenformat vorliegen. Dies macht es schwierig bis unmöglich, Daten zwischen verschiedenen Programmen auszutauschen. Dadurch wird es oft nötig, bestimmte Modelle in mehreren Varianten zu erstellen, was zur Verletzung des DRY-Prinzips (keine Redundanz bzw. möglichst geringe Redundanz) führt. Dadurch kann es z. B. zu Inkonsistenzen kommen, was fatale Auswirkungen auf die Planung und Konstruktion haben kann. Doppelte Datenhaltung führt zu steigenden Kosten, die durch den Mehraufwand der doppelten Datenhaltung entstehen. Immer wenn es in einem Datenmodell eine Änderung gibt, muss diese im anderen Datenmodell nachgezogen werden. Neben der Konstruktion sind Datenmodelle von komplexen Infrastrukturbauten auch nach ihrer Erstellung von großem Interesse, beispielsweise für Revisions- und Wartungsarbeiten oder auch für den Rückbau von Bauwerken. Ein neutrales Produktdatenmodell ist besonders in der Planungsphase wünschenswert, da es den Planungsbeteiligten die Möglichkeit bietet, mit verschiedenen Softwarewerkzeugen zu arbeiten. Idealerweise bietet ein neutrales Produktdatenmodell Möglichkeiten, um flexibel auf Änderungen reagieren zu können. Sollte sich beispielsweise die Trassierung eine Straße ändern, müssen sich auch alle auf ihr befindlichen Brücken und Tunnelbauwerke anpassen bzw. alle geltenden Richtlinien gegengeprüft werden - der Berliner Flughafen lässt grüßen. Bisher gibt es für den Infrastruktursektor kein neutrales Datenformat bzw. Produktdatenmodell, das es erlaubt, solche Richtlinien zu speichern und diese einfach zwischen verschiedenen Programmen auszutauschen. Im Rahmen dieser Arbeit soll aufgezeigt werden, wie ein solches Produktdatenmodell aussehen könnte.

2.2 Die IFC-Programmiersprache (IFCPL)

Um prozedurales Wissen in austauschbare IFC-Modelle einzubetten, wurde die IFC-Programmiersprache (IFC programming language) entwickelt (Amann et al. 2014a). Diese wird in Kurzform auch als IFCPL bezeichnet. Diese erlaubt es, Programme (Algorithmen) zu verfassen und in einem herstellerneutralen Datenformat auszutauschen. Die definierten

Algorithmen können beispielsweise Verarbeitungs- oder Analyseprozesse beinhalten. Mögliche Anwendungen für die IFCPL reichen von der Definition von komplexen Funktionen für die Mengenermittlung bis zur Beschreibung des „Verhaltens“ von parametrischen Objekten (Lee et al. 2006). Da diese Programme einen Teil des Austauschprozesses darstellen, können diese wiederum bei der empfangenden Seite (Softwareapplikation) genutzt werden. Auf diese Weise können der Programmieraufwand auf der empfangenden Seite reduziert und Fehlinterpretationen der Daten verhindert werden. Betrachtet man aktuelle Softwareprodukte, die IFC-Dateien austauschen, sind Datenverluste und Fehlerinterpretationen leider häufig vorzufinden. Genauso werden häufig Standards nicht vollständig implementiert, sodass es zu zeitaufwändig für die Softwarehersteller ist, jede mögliche Kombination des Datenmodells zu unterstützen. Dies rechtfertigt den hier gewählten Ansatz.

Das IFCPL (IFC Programming Language) Programm selbst ist nicht Teil des IFC-Schemas, sondern Teil einer IFC-Instanzdatei. Somit findet es sich nicht auf der Schemaebene, sondern auf der Instanzebene. Im Vergleich zu einem Ansatz auf Schemaniveau ermöglicht der instanzbasierte Ansatz eine wesentlich höhere Flexibilität in Bezug auf das Definieren von neuen IFCPL-Programmen, da eine Schemamodifikation einen langwierigen Standardisierungsprozess nach sich ziehen würde. Die Integration von IFCPL-Programmen in das IFC-Schema sowie die Ausführungsumgebung wurden in (Amann et al. 2014a) vorgestellt und sollen hier nicht wiederholt werden. Im Folgenden werden nur die neuen Features der IFCPL-Programmierumgebung beschrieben.

2.3 IFCPL und die Richtlinien für die Anlage von Landstraßen

Die Richtlinien für die Anlage von Landstraßen (RAL) sollen den Entwurf von sicheren und funktionsgerechten Landstraßen gewährleisten und standardisieren (FGSV 2013). Beispielsweise schreibt die RAL vor: „... die Länge von Geraden [soll] auf max L = 1.500 m begrenzt werden.“. Diese Vorschrift lässt sich einfach mittels IFCPL umsetzten:

```
using IFC4_P6_longform_draft5.exp;
func checkMaximumIfcLineSegment2D() {
        // input property set provides IfcAlignment named alignment
        for(IfcLineSegment2D line : alignment) {  // visit each IfcLineSegment2D
                if(line.SegmentLength > 1500) {
                        return false;
                }
        }
        return true;
}
```

Einige Besonderheiten dieses Programmes gibt es zu erklären:

1) Die `using` Anweisung bindet alle Entitäten des IFC4_P6_longform_draft5.exp EXPRESS-Schemas in die Programmumgebung ein. Damit können alle Entitäten samt ihrer Attribute des EXPRESS-Schemas direkt im IFCPL-Programm verwendet werden, ohne diese zu definieren. Dieses Feature bezeichne ich als Type-Injektion (Type-Injection), da man neue Typen (gemeint sind Datentypen) einfach über das entsprechende EXPRESS-Schema (im Beispiel *IFC4_P6_longform_draft5.exp*) bekannt machen kann, ohne diese explizit zu definieren. Bisher wurde die Type-Injektion im Rahmen dieser Arbeit nur prototypisch für ein EXPRESS-Schema implementiert. Analog könnte man aber auch bei einem XML Schema vorgehen und so z. B. OKSTRA (Objektkatalog für das Straßen- und Verkehrswesen) unterstützen.

Bei der bisher Implementierten Type-Injektion wird zunächst das EXPRESS-Schema geparst und anschließend jede Typ- oder Entity-Deklaration mit in die Symboltabelle für das IFCPL-Programm aufgenommen. Auf die Attribute einer Entität kann mittels des Punktoperators „.“ zugegriffen werden.

2) Die `for` Anweisung erlaubt es, über alle referenzierten Attribute einer Entität zu iterieren. Genauso wäre `for(IfcLine2DSegment2D line : product)` mit `product` vom Typ `IfcProduct` möglich. Alle referenzierten Entitäten werden über eine Breitensuche ausfindig gemacht.
3) Als Input-Property-Set wird ein `IfcAlignment` mit dem Namen *alignment* angenommen.
4) Das Ouput-Property-Set besteht aus einem booleschen Wert, der angibt, ob die Regel erfüllt oder verletzt wurde.

Ein anderes Beispiel für eine Richtlinie der RAL ist die Vorgabe von Kurvenmindestradien bei der Elementfolge Gerade – Klothoide – Kreisbogen. Tabelle 1 gibt hierzu einen kleinen Überblick (Diese gilt nur für bestimmte Straßenkategorien – siehe dazu auch (FGSV 2013)).

Tabelle 1: Kurvenmindestradien bei der Elementfolge Gerade – Klothoide – Kreisbogen

Länge L[m] der Geraden	**min R [m] des Kreisbogens**
L ≥ 300 m	min R > 400m
L < 300 m	min R > L

Diese Richtlinie lässt sich mit folgendem IFCPL-Programm ausdrücken:

```
func checkCurveMinRadius() {
        // consider always three successive segments
        for(i = 0; i <= alignment.Segments.Count-3; i++) {
                if(alignment.Segments[i]   is IfcLineSegment2D &&
                   alignment.Segments[i+2] is IfcClothoidalArcSegment2D) {
                        line = alignment.Segments[i];
                        arc = alignment.Segments[i+2];

                        if(line.SegmentLength >= 300.0) {
                                if(arc.Radius <= 400.0)
                                        return false;
                        }
                        else {
                                if(arc.Radius < line.SegmentLength)
                                        return false;
                        }
                }
        }

        return true;
}
```

Anmerkungen zum Programm:

1) Bei Listen wird immer implizit das Attribut `Count` angeboten, das die Anzahl der Elemente in der Liste angibt.

2) Der Indexoperator „`[]`“ kann verwendet werden, um auf die unterschiedlichen Elemente einer Liste zuzugreifen.
3) Die Funktion liefert den Wert `true`, wenn die Bedingungen der Tabelle 1 erfüllt sind.

3 Schlusswort

Es wurde exemplarisch gezeigt, wie sich Regeln der RAL mithilfe der IFCPL abbilden lassen und wie neben der Berechnung von beliebigen Übergangskurven (vgl. (Amann et al. 2014)) die IFCPL auch für den gezeigten Anwendungsfall eingesetzt werden kann. Ähnlich kann man auch noch weitere Regeln der RAL abbilden und herstellerneutral austauschen. Darüber hinaus könnte man auch projektspezifische oder firmeninterne Regeln abbilden. Das Prinzip ist auch auf andere Produktdatenmodelle einfach übertragbar (z. B. OKSTRA). Die Anwendungsfälle der IFPCL sind vielschichtig und lassen noch großen Raum für weitere Fallstudien. Die Anreicherung von Produktdatenmodellen mit intelligenten Daten wie der RAL stellt einen Mehrwert für die Nutzer solcher Produktdatenmodelle dar und kann die Attraktivität von offenen, herstellerneutralen Produktdatenmodellen steigern. Ebenfalls integriert sich dieser Ansatz wunderbar in einen Open-BIM-Ansatz.

Literatur

AMANN, J., FLURL, M., JUBIERRE, J. R., Borrmann, A. (2014a), An Approach to Describe Arbitrary Transition Curves in an IFC Based Alignment Product Data Model, In: Proc. of the 2014 International Conference on Computing in Civil and Building Engineering, Orlando, USA.

AMANN, J., BORRMANN, A., CHIPMAN, T., LEBEGUE, E., LIEBICH, T., SCARPONCINI, P. (2014b), P6 IFC Alignment – Conceptual Model, http://www.buildingsmart-tech.org/downloads/ifc/ifc5-extension-projects/ifc-alignment/ifcalignment-conceptualmodel-cs (abgerufen am 29.07.2015).

BORRMANN, A., JI Y., WU I-C., OBERGRIEßER M., RANK E., KLAUBERT C. & GÜNTHNER, W. (2009), ForBAU - The virtual construction site project, In: Proc. of the 24th CIB-W78 Conference on Managing IT in Construction. Istanbul, Türkei.

GÜNTHNER, W. A., BORRMANN, A. (EDS) (2011), Digitale Baustelle – innovativer Planen, effizienter Ausführen. Werkzeuge und Methoden für das Bauen im 21. Jahrhundert, Springer Verlag, Heidelberg.

FORSCHUNGSGESELLSCHAFT FÜR STRAßEN- UND VERKEHRSWESEN (FGSV) (2013), Richtlinien für die Anlage von Straßen.

KAMINSKI , I. (2010), Potenziale des Building Information Modeling im Infrastrukturprojekt - Neue Methoden für einen modellbasierten Arbeitsprozess im Schwerpunkt der Planung, Books on Demand; Auflage: 3 (13. August 2010).

LEE, G., SACKS, R. & EASTMAN, C. (2006), Specifying parametric building object behavior (BOB) for a building information modeling system. Automation in Construction, 15(6), pp.758–776.

WIKIPEDIA (2015), Leonhard Obermeyer, https://de.wikipedia.org/wiki/Leonhard_Obermeyer (abgerufen am 29.07.2015).