Graphenbasiertes Änderungsmanagement von BIM-Modellen

Sebastian Esser[1]

[1]Technische Universität München, Lehrstuhl für Computergestützte Modellierung und Simulation

Zusammenfassung

Ansätze der modellbasierten Zusammenarbeit zwischen allen Projektbeteiligten wird in BIM-Projekten jeglicher Größe zunehmend eingesetzt. Die heutige Praxis beruht dabei auf dem Austausch und der Koordination so genannter Fach- oder Domänenmodelle, die dateibasiert als kleinste Einheit über eine Projektplattform ausgetauscht werden. Dieser Ansatz folgt dem Prinzip sogenannter Informationscontainer, wie sie in ISO 19650 im Kontext der BIM Level 2 Implementierungen erläutert werden [3].
Aufgrund der Vielzahl an beteiligten Experten und Domänen müssen im Verlauf der Planung einzelne Modelle angepasst oder erweitert werden, was zu mehreren Versionen und damit neuen Dateien führt. Die heutige Praxis beruht in solchen Fällen auf dem erneuten Austausch des gesamten Modells und führt zu einem ineffizienten Vorgehen, da jede Revision einen Großteil der bereits zuvor übermittelter Informationen enthält, die die Empfänger bereits in deren fachbezogene Systeme integriert haben.

In dieser Veröffentlichung werden neuartige Ansätze vorgestellt, die die Grundlagen für eine objektbasierte Versionierung von BIM Modellen ermöglichen und die angewandte Aktualisierung an die Projektbeteiligten abstrahiert. Insbesondere bei geringfügigen Veränderungen wie der Anpassung einzelner Parameter eines Objektes führt der vorgestellte Ansatz zu erheblich geringeren Datenmengen, die übermittelt werden müssen. Zur Beschreibung einzelner Komponenten eines BIM Modells wird hierbei auf graphenbasierte Repräsentationen zurückgegriffen, die das komplexe Objektnetzwerk innerhalb eines Modells adäquat und verlustfrei abbilden können. Dabei werden neben der allgemeinen Problemstellung erste Ansätze erläutert, die die Ermittlung einer angewandten Änderung zwischen zwei Modellzuständen abstrahiert. Diese bilden wiederum die Grundlage für zukünftige patch-basierte Methoden, um sämtliche Kopien eines Modells konsistent und aktuell zu halten. Damit werden erste technische Grundlagen für BIM Level 3 Implementierungen gelegt, die zukünftig Einzug in CDE-Systeme halten werden.

Stichworte: Graphen, Datenaustausch, Common Data Environment, Industry Foundation Classes, BIM Level 3 Kollaboration

1 Motivation und Problemstellung

Mit dem zunehmenden Einsatz von Applikationen und Methoden des Building Information Modelings (BIM) erlangt die modellbasierte Zusammenarbeit immer größere Bedeutung. Dabei werden mithilfe zentraler Projektplattformen planerische Informationen unter Zuhilfenahme passender Datenstrukturen als Modelle ausgetauscht. Solche Modelle können dabei neben zwei- und dreidimensionalen Geometrien ebenso semantische Informationen über einzelne Bauteile und deren Beziehungen untereinander beinhalten.

Neben dem modellbasierten Planen stellt der Informationsaustausch zwischen den an einem Projekt beteiligten Personen einen wesentlichen Aspekt heutiger Bauvorhaben dar. Dabei werden Prozesse zum Informationsaustausch zumeist mithilfe gemeinsamer Projektplattformen (sog. Common Data Environments (CDE)) realisiert, die den normativen Vorgaben von ISO 19650 und PAS 1192 folgen [2, 3]. Alle Gewerke planen in dezentralen und für die jeweiligen Belange optimierten Planungswerkzeugen und tauschen zu definierten Zeitpunkten die jeweiligen Fachplanungen modellbasiert aus. Mithilfe von Applikationen zur Koordination mehrerer Einzelmodelle können Konflikte zwischen einzelnen Fachmodellen frühzeitig erkannt werden.

Durch die Einarbeitung von Änderungswünschen entstehen im Verlauf eines Projektes stetig neue Versionen einzelner Modelle. Zusätzlich werden über den Verlauf verschiedener Plaungsphasen und während dem Lebenszykluses eines Bauwerks weitere Informationen generiert, die ebenfalls Eingang in die jeweiligen Modelle finden. In der heutigen Praxis werden Aktualisierungen an Modellen in der Regel durch die erneute Übermittlung des gesamten, aktualisierten Datei umgesetzt, die zusammen mit begleitenden Dokumenten einen Informationscontainer bilden.

Ein solcher Container ist im wesentlichen eine Sammlung einzelner Dateien, die als monolithisches Paket übermittelt werden. Ein BIM Modell wird als einzelne Datei behandelt, die im Datenaustauschprozess nicht weiter in kleinere Einheiten zerlegt wird. Folglich bestehen die übermittelten Modellrevisionen zum überwiegenden Teil aus Informationen, die bereits in den vorangegangenen Versionen ausgetauscht und koordiniert wurden.

Die an einer Modellrevision vorgenommenen Änderungen sind in der Regel im Vergleich zur Gesamtdatenmenge eines Fachmodells klein. Daraus lassen sich Überlegungen zur Versionierung anknüpfen, um zukünftig nicht mehr das Modell an sich, sondern lediglich

einzelne modifizierte Objekte als kleinste Einheit zu betrachten. Technologien zur inkrementellen Aktualisierung einer definierten Datenmenge werden bereits in anderen Branchen seit langer Zeit erfolgreich eingesetzt. Diese Methoden können allerdings nicht direkt auf die häufig stark vernetzten und komplexen Objektstrukturen eines BIM-Modells angewendet werden.

Um diese Einschränkungen zu lösen, bieten sich Konzepte der Mengenlehre sowie der Graphentheorie an. Dabei handelt es sich um weit verbreitete Methoden und Ansätze, die bereits in zahlreichen Fragestellungen der Wissenschaft zur Anwendung kamen und damit eine stabile Grundlage für die anvisierten Funktionalitäten zur Modellversionierung bereitstellen.

1.1 Strukturierung der Veröffentlichung

Die vorliegende Veröffentlichung stellt Ansätze dar, wie unter Zuhilfenahme graphentheoretischer Konzepte eine objektbasierte Versionierung von BIM Modellen umgesetzt werden kann. Besonderer Fokus wird dabei auf die möglichst generische Abstraktion von gängigen Datenmodellen auf eine flexibles graphenbasierte Repräsentation gelegt, um das entwickelte Konzept auf alle gängigen Datenmodelle im BIM-Kontext anwenden zu können. Es muss neben einer möglichst flexiblen Anwendbarkeit auf verschiedenste Spezifikationen sichergestellt werden, dass sämtliche Informationen verlustfrei verarbeitet werden. Dafür wird ein rekursiver Ansatz zum Abgleich der Modellinformationen eingeführt und dessen Limitationen im Bereich der Geometrieversionierung dargestellt.

Als weit verbreitetes und etabliertes Datenmodell wird exemplarisch auf die *Industry Foundation Classes* (IFC) zurückgegriffen, das zunehmend auch in Infrastrukturprojekten Anwendung findet.

2 Kontext und verwandte Ansätze

Bereits seit längerer Zeit spielen Fragen zur Versionierung strukturierter Daten eine große Rolle in Industrie und Wissenschaft. Insbesondere im Bereich der Softwareentwicklung entstehen Applikationen und Programme zumeist durch die Zusammenarbeit mehrerer Entwickler, sodass hier schon länger verschiedene Systeme für die Versionierung von textbasierten Daten zum Einsatz kommen. Bekannt sind hier vor allem Applikationen wie *Subversion*, *Git* oder *Mercurial*. Dabei erfasst eine zentrale Datenbank sämtliche Änderungen, die an den überwachten Textdateien durchgeführt werden. Ein Nutzer kann dabei die gesamte Änderungshistorie in seine Entwicklungsumgebung laden und unabhängig neue Funktionalitäten implementieren und deren Auswirkungen testen. Sobald ein neuer Versionszustand in die Historie gespeichert werden soll, erfolgt ein so genannter `Commit`.

Der Inhalt eines Versionszustands wird vereinfacht gesagt durch das Vergleichen jeder Textzeile zwischen initialer und modifizierter Textdatei umgesetzt, die hinzugefügte, modifizierte und gelöschte Zeilen erkennt. Das Resultat dieser *Diff*-Operation wird in der Historie gespeichert und anschließend mit einer serverbasierten Kollaborationsplattform synchronisiert (`Push`-Operation). Nach der Serversynchronisation ist es anderen Nutzern des Projekts möglich, die Veränderungen durch die Ausführung des *pull*-Befehls in deren lokale Kopien zu integrieren. Sofern getätigte Modifikationen widerspruchsfrei in die lokal gespeicherten Versionen anderer Beteiligter eingefügt werden können, erfolgt die Integration eingehender Änderungen automatisch. Lediglich im Falle von Konflikten zwischen den Änderungen verschiedener Autoren muss der Nutzer selbst entscheiden, welche der konkurrierenden Versionen verwendet werden soll [1].

Wie beschrieben stützen sich gängige Versionierungssysteme in der Softwareentwicklung auf Algorithmen, die zwei Textdateien mithilfe zeilenbasierter Methoden vergleichen. Das Verschieben eines bestehenden Textblocks innerhalb einer Datei wird dabei nicht als eine *Verschieben-Operation*, sondern als Kombination aus *Löschen* und *Hinzufügen* abgebildet. Dies ist für das Anwendungsfeld der Softwareentwicklung eine verkraftbare, wenig einschränkende Erkenntnis.

2.1 Anwendung textbasierter Versionierung auf komplexe Objektnetzwerke

Wie eingangs beschrieben zeichnen sich BIM Modelle durch komplexe und hochgradig vernetzte Objektstrukturen aus. Als Objekt wird in diesem Kontext die Instanz einer Klasse verstanden. Mithilfe passender Assoziationen bilden mehrere Objekte schlussendlich einzelne Bauteile im ingenieurmäßigen Sinne ab, die zum Beispiel eine geometrische Form, Materialien und thermische Eigenschaften eines rellen Gegenstands reflektieren.

Für die Speicherung und den Austausch von BIM-Modellen wird in vielen Fällen auf textbasierte Strukturen zurückgegriffen, die von passenden Export-Schnittstellen befüllt werden können. Im Kontext des IFC-Datenmodells kommt hier einerseits das *STEP Physical File (SPF)* Format sowie die Serialisierung in die als *IFCXML* bekannte Darstellung zum Einsatz.

Wenngleich es sich dabei genauso wie in der Softwareentwicklung um Textdateien handelt, führt die Anwendung der eben dargestellten Textversionierungssystemen zu meist aussagelosen Ergebnissen. Bei beiden Formaten wird während der Serialisierung der Objektstrukturen in die textuelle Repräsentation eine Sortierung aller Objekte eingeführt werden, die eine gewisse (zeilenweise) Hierarchie einführt Diese Strukturierung besitzt verglichen mit der im Objektnetzwerk gespeicherten Information keine Bedeutung, sondern ist rein technisch bedingt.

2.2 Ansatz zur Berücksichtigung von Topologiebeziehungen

Um die dargestellten Limitationen zu überwinden, bietet es sich an, die Objektnetzwerke innerhalb eines BIM-Modells nicht mehr textbasiert auf Basis der Serialisierungsergebnisse, sondern auf Grundlage der einzelnen Objekte und deren Beziehungen untereinander

zu betrachten. Dabei gilt es ein System zu erarbeiten, welches die angewandte Änderung zwischen zwei Modellversionen unter Berücksichtigung der topologischen Objektbeziehungen abstrahiert und diese anschließend als `Patch` an andere am Projekt beteiligten Experten übermittelt. Das vorliegende Paper konzentriert sich auf die topologische Abbildung der Modellinhalte, die die Grundlage für zukünftige Patch-Strukturen erschafft.

Ansätze der Mengen- und Graphentheorie stellen etablierte Konzepte dar, um die Zusammensetzung komplexer Datenmengen zu modellieren und mit diesen mithilfe topologischer Analysen zu interagieren. So wird es möglich, Änderungen zwischen Modellrevisionen durch Analysen der Objektnetzwerke zu abstrahieren. Ebenso können ermittelte Änderungen durch die Formulierung von Graphersetzungsregeln übermittelt und auf Empfängerseite wiederum in die dortigen Graphen integriert werden. Um die Anwendbarkeit des Systems in heute verfügbare Softwaresysteme sicherzustellen, können alle Graphen jederzeit zurück in dateibasierte Repräsentationen gewandelt werden, um so die Datenmenge mit existierenden Importschnittstellen in BIM-Applikationen laden zu können.

Abbildung 1 fasst den erläuterten Ansatz visuell zusammen. Jede Domäne arbeitet wie bisher in den für sie passenden Softwareapplikationen und stellt anschließend ihre Fachmodelle zum Datenaustausch bereit. Diese BIM-Modelle werden anschließend als Graphenstruktur gespeichert. Sobald eine neue Version des Fachmodells erstellt wird, wird diese mit der vorangegangenen Version verglichen,um die angewandten Änderungen zu ermitteln. Die Änderung wird anschließend in Form eines Patches abstrahiert und ermöglicht es, Kopien des veralteten Modells durch Anwendung des Patches ebenfalls auf den neuen Modellzustand zu aktualisieren. Um Kompatibilität mit importierenden BIM-Applikationen zu gewährleisten, können die im Graphen gespeicherten Informationen auch wieder in ein serialisiertes Dateiformat (zum Beispiel *STEP Physical File* oder *XML*-basierte Strukturen) übersetzt werden.

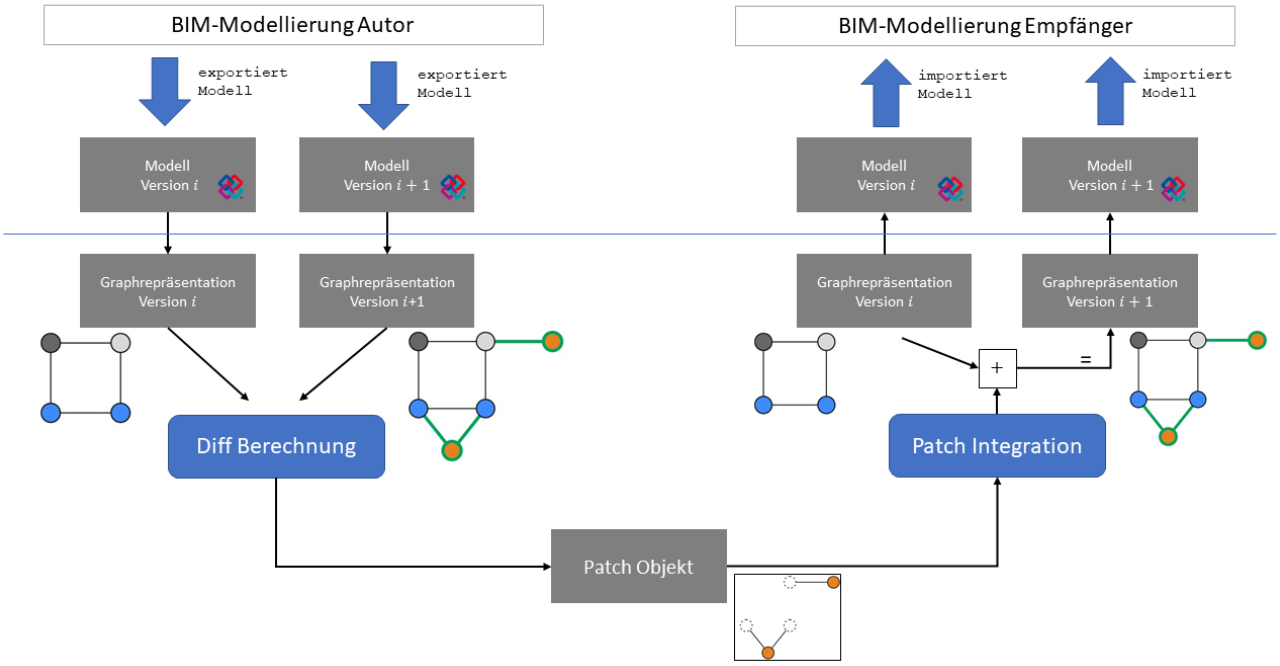

Abbildung 1: Systemarchitektur zur patchbasierten Modellaktualisierung verteilter Kopien von BIM-Modellen

Gesucht wird nun eine passende Darstellung für BIM Modelle, die sich möglichst gut mit existierenden und etablierten Datenstrukturen vereinbaren lässt und möglichst geringe Randbedingungen zur Übersetzung von Fachmodellen in die Graphenstruktur definiert. Die zentralen Konzepte werden in den nächsten Kapiteln exemplarisch anhand des IFC-Datenmodells erläutert. Die hierbei dargestellten Prinzipien können ebenfalls auf andere Datenstrukturen wie `PlanPro`, `RailML`, `LandXML` oder `CityGML` übertragen und angewandt werden.

3 Übersetzung von BIM Modellen in graphenbasierte Repräsentationen

Gebräuchliche Datenmodelle, die von heutigen BIM-Applikationen zum (herstellerneutralen) Datenaustausch unterstützt werden, folgen den Paradigmen der objektorientierten Programmierung [5]. Ein Datenmodell besteht aus mehreren *Klassen*, die die Eigenschaften und Methoden einzelner Objekte definieren. Die Begriffe *Instanz* und *Objekt* bezeichnen hingegen die tatsächliche

Befüllung einer Klasse mit konkreten Attributwerten. In einer Klassendefinition werden hierfür Attribute, deren Datentypen sowie Methoden festgelegt, die alle Instanzen dieser Klasse besitzen. Im Zusammenhang mit Datenmodellen zum Informationsaustausch wird ergänzend von so genannten *Entitäten* gesprochen, die ebenfalls dem Begriff der *Klasse* zuzuordnen sind, aber typischerweise keine eigenen Methoden, sondern ausschließlich Attribute definieren.

3.1 Eigenschaften und Randbedingungen zur graphenbasierten Modellrepräsentationen

Um die Anwendbarkeit für eine möglichst breite Palette von Schemaspezifikationen zu gewährleisten, wird ein generisches Metamodell für die graphenbasierte Darstellung von BIM-Modellen eingeführt, welches ein minimales Set an Anforderungen definiert, die von den zu versionierenden Datenspezifikationen (Formaten) einhalten werden muss.

Formal besteht ein Graph aus einer Knoten- und Kantenmenge, die Beziehungen zwischen einzelnen Elementen der Knotenmenge implementiert. Knoten und Kanten können - je nach gewähltem Graphensystem - zusätzliche Gewichte (d. h. Attribute in Form von Name-Wert-Paaren) tragen. Solche Graphen werden auch als *attribuierte Graphen* oder *Property Graphs* bezeichnet [4]. Außerdem werden jedem Knoten ein oder mehrere *Labels* zugeordnet, die helfen, eine bestimmte Menge von Knoten zu identifizieren und zu selektieren. Kanten können ungerichtet oder gerichtet sein [6].

Die Möglichkeit, einem Knoten Attribute als Schlüssel-Wert-Paare zuzuordnen, entspricht den objektorientierten Prinzipien, die in zahlreichen modernen Programmiersprachen zu finden sind. Darüber hinaus stellt das Konzept der Vererbung zwischen Klassen ein wesentliches Argument dar, welches im Folgenden für die Vergabe einzelner Labels an Knoten relevant wird.

Als Randbedingung für das graphbasierte Versionierungssystem wird definiert, dass jeder Knoten im Graphen genau ein Objekt aus dem Objektnetzwerk des BIM-Modells reflektiert. Alle Attribute einer Klasse werden an den Knoten als Name-Wert-Paar angehängt, während Assoziationen (teils auch als komplexe Attribute bezeichnet) zu einer anderen Klasseninstanz mit einer Kante im Graph modelliert werden.

3.2 Darstellung einzelner Klasseninstanzen in Knoten

Die Graphenrepräsentationen der BIM-Modelle verwenden drei verschiedene Knotentypen: *Primärknoten*, *Sekundärknoten* und *Verbindungsknoten*. Alle Knoten können flexibel durch Kanten verbunden werden, um gegenseitige Assoziationen und Referenzen abzubilden. Hierzu werden keine weiteren Einschränkungen definiert.

Primärknoten werden verwendet, um die Grundstruktur eines BIM-Modells im Graphen abzubilden. Die im BIM-Kontext eingesetzten Datenmodelle verfügen über eine abstrakte Basisklasse, die ein GUID-Attribut (globally unique identifier) definiert und diese Instanzen eindeutig identifizierbar macht. Mithilfe des Vererbungsprinzips erhalten alle Unterklassen einer solchen Basisklasse ebenfalls das GUID-Attribut. Die Strukturen, die sich aus allen Klasseninstanzen ergeben, die von der Basisklasse erben, wird häufig in einem Projektbrowser visualisiert und dient dem Nutzer zur Navigation innerhalb des Fachmodells.

Neben den *Primärknoten* werden *Sekundärknoten* eingeführt, die alle Klasseninstanzen reflektieren, die nicht von der Basisklasse erben und folglich kein GUID-Attribut besitzen. Im IFC-Datenmodell sind dies alle Klassendefinitionen, die im so genannten *Resource Layer* angesiedelt sind. Diese Klassen definieren Geometrien, Materialien oder modellieren sonstige Askepte wie Prozesse, Kosten oder Parameter (für Tragwerke, Bauphysik, Brandschutz, etc.). In der Schemadefinition ist zudem festgelegt, dass jede Klasseninstanz des *Resource Layers* stets mindestens eine Verknüpfung zu einer von der Basisklasse vererbten Klasseninstanz aufweisen muss. Umgekehrt bedeutet dieses Lemma, dass ein schema-konformes Fachmodell keine Instanzen des *Resource Layers* beinhalten darf, denen man nicht mindestens eine Klasseninstanz mit eindeutiger GUID zuweisen kann.

Die dritte Art von Knoten wird als *Verbindungsknoten* bezeichnet. Diese Knoten repräsentieren das Konzept der *objectified relationships*. Sie bieten die Möglichkeit, 1:n oder m:n Beziehungen zwischen Objekten zu modellieren und der jeweiligen Abhängigkeit weitere semantische Bedeutungen zuzuweisen. Ähnlich wie *Primärknoten* tragen *Verbindungsknoten* ein eindeutiges, durch die Schemaspezifikation festgelegtes GUID-Attribut.

Allen drei Knotentypen ist gemein, dass jeder Knoten neben Attributen der jeweiligen Klasseninstanz ein weiteres Attribut namens *EntityType* erhält. Dieses reflektiert den Namen der Klasse, die in der Schemaspezifikation des zugehörigen Datenmodells nachgeschlagen werden kann. Von besonderer Relevanz ist dieses Attribut, wenn die Informationsmenge aus dem Graphen wieder in eine ANSI-basierte Dateirepräsentation gewandelt werden soll.

3.3 Abbildung von Beziehungen und Assoziationen im Graphen

Eine Kante in einem Graphen stellt eine Beziehung zwischen zwei Elementen der Knotenmenge her und kann Attributen zugewiesen bekommen. Diese Attribute werden ähnlich zu den Attributsätzen der Knoten durch Name-Wert-Paare implementiert.

Zur Abbildung von BIM-Modellen werden Kanten verwendet, um Assoziationen zwischen zwei Klasseninstanzen abzubilden. Jede Kante erhält dabei das Attribut *relType*, das den Namen der Assoziation reflektiert.

Neben einfachen Assoziationen können Listen von Assoziationen auftreten. Im Graphen werden alle in einer Liste enthaltenen Assoziationen durch eine einzelne Kante dargestellt. Zusätzlich erhalten diese Kanten neben dem *relType* Attribut ein weiteres Attribut *itemNo*, das die Position der Referenz innerhalb der gruppierenden Liste speichert.

3.4 Anwendung

Abbildung 2 wendet die zuvor dargestellten Prinzipien auf ein einfaches Datenmodell an, welches sich aus den Klassen (Entitäten) *Point*, *Line* und *ShapeElement* zusammensetzt.

Die Klasse *Line* referenziert mit ihren Attributen `StartPoint` und `EndPoint` jeweils auf Klasseninstanzen der *Point*-Klasse. Gleiches Prinzip wird für die Assoziation zwischen einem *ShapeElement* und einer *Line*-Instanz angewendet.

Jede Klasseninstanz wird im Graphen durch einen individuellen Knoten abgebildet. Als Festlegung zur Klassifizierung von Primär- und Sekundärknoten wird festgelegt, dass Instanzen der Klasse *ShapeElement* als Primärknoten, alle anderen Instanzen als Sekundärknoten behandelt werden. Sämtliche Attribute mit einfachem Datentyp werden jeweils direkt an den Knoteninstanzen als Name-Wert-Paare abgebildet. Ergänzend erhält jeder Knoten das Attribut *entityType*, welches den Namen der zugrundeliegenden Klasse des jeweiligen Datenmodells reflektiert. Assoziationen werden durch ein Attribut namens *relType* an der Kante abgebildet. Die Kante zeigt dabei von der Ursprungsklasse in Richtung der Assoziation. Manche Datenmodelle wie das IFC-Schema definieren neben Assoziationen auch inverse Verweise. Diese können in Form einer weiteren Kante gegenläufig zu den dargestellten Assoziationen ebenfalls abgebildet werden.

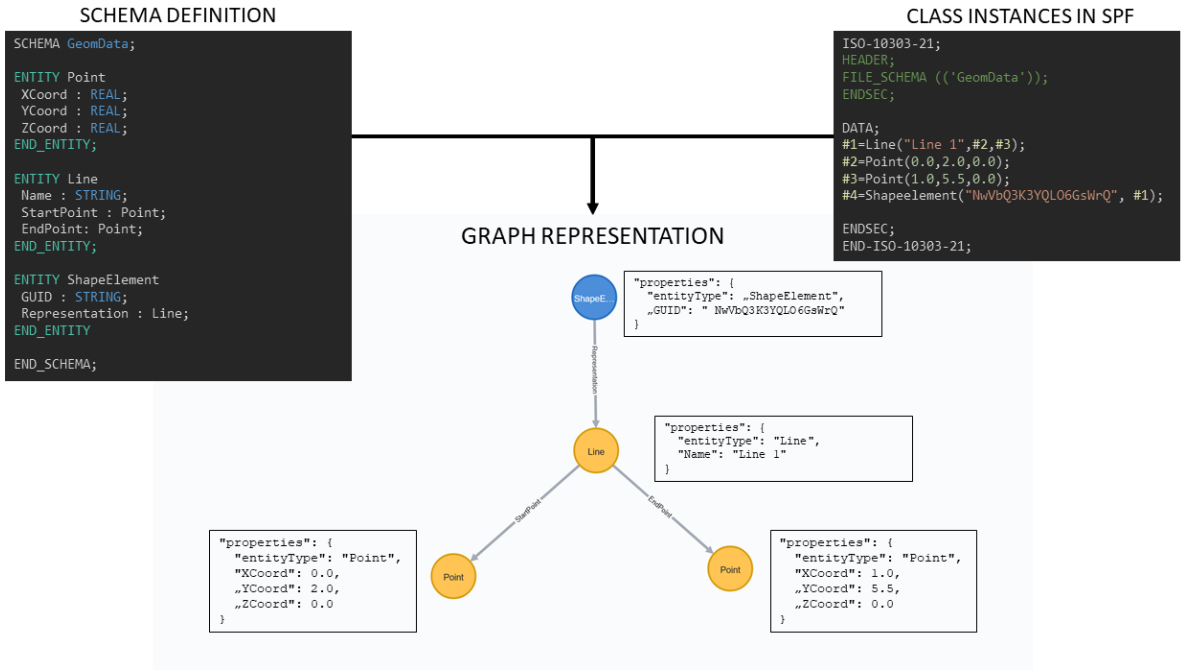

Abbildung 2: Zusammenhang zwischen Schemadefinition, Instanzdatenmenge und resultierender Graphenrepräsentation

4 Ermittlung angewendeter Modellveränderungen

Durch die zuvor eingeführte Darstellung des Objektnetzwerks innerhalb eines BIM Modells als Graphenstruktur wird es nun möglich, zwei gegebene Versionen topologisch auf deren Unterschiede hin zu untersuchen. Innerhalb der Topologiedarstellung existiert so keinerlei unbeabsichtigte Reihenfolge zwischen einzelnen Objekten mehr, die durch den Serialisierungsprozess in die textbasierte Repräsentation beim Exportvorgang eingeführt wurde.

Durch die getroffene Unterscheidung zwischen Primär- und Sekundärknoten erfolgt die Analyse zweier Modelle (oder genauer: deren beiden Graphen) in zwei Schritten:

1. Abgleich der Primärstruktur aus *Primär-* und *Verbindungsknoten*
2. Untersuchung der Sekundärstrukturen als Substruktur unter als äquivalent erkannten Primärknoten

Das Ziel besteht nun in der Ermittlung von Änderungen zwischen beiden Modellzuständen auf der generischen Graphdarstellung. Welche konkrete Klasseninstanz ein einzelner Knoten dabei reflektiert oder welches Datenmodell dem Graphen zugrundeliegt, soll dabei irrelevant sein.

4.1 Vergleich der Primärstruktur

Da *Primär-* und *Verbindungsknoten* über ein GUID-Attribut verfügen, das diesen Knoten ein eindeutiges Merkmal zuweist, handelt es sich gemäß mathematischer Definition um eine Menge mit eindeutig unterscheidbaren Elementen. Somit ist es möglich, mithilfe von Adjazenz-Untersuchungen zu ermitteln, ob einzelne Knoten im initialen und im aktualisierten Modell vorhanden sind und ob sie nach wie vor die gleichen Beziehungen zu anderen Knoten besitzen. Dies lässt sich über die Erstellung von Adjazenzmatrizen und deren Vergleich einfach ermitteln. Um diese Matrizen vergleichen zu können, muss allerdings sichergestellt werden, dass die Matrixeinträge für beide Modellversionen in gleicher Reihenfolge befüllt werden. Hierzu bietet sich entweder eine Sortierung nach den GUID-Attributwerten oder die Bildung von Hashsummen über die gesamten Knotenattribute an.

4.2 Vergleich der Sekundärstrukturen

Als Resultat des des vorangegangenen Schritts geht eine Teilmenge von Primärknoten hervor, die sowohl in der initialen als auch in der aktualisierten Version des BIM-Modells vorliegen. Diese bilden die Grundlage für den Vergleich der Sekundärstrukturen, die sich jeweils unter einem Primärknoten aufspannen. Anschaulich lassen sich die Sekundärstrukturen im Kontext des IFC-Datenmodells als Ressourcen verstehen, die die Positionierung, die geometrische Form oder sonstige domänenspezifischen Daten abbilden.

Im Gegensatz zu den Primärstrukturen können innerhalb der Menge von Sekundärknoten einzelne Elemente mehrfach auftauchen. Ein häufiger Fall sind beispielsweise Instanzen von kartesischen Punkten, die an verschiedenen Stellen des BIM-Modelles modelliert werden, aber stets die gleiche Koordinatenwerte repräsentieren. Mathematisch wird dieses Szenario nun als Multimenge verstanden und erweitert den Mengenbegriff um den Fakt, dass einzelne Elemente mehrfach in der Menge inkludiert sein können. Aus diesem Grund ist das beschriebene Vorgehen für den Abgleich zweier Primärstrukturen mit Adjazenzmatrizen nicht mehr anwendbar.

Weiterhin besteht die Zielsetzung im Identifizieren hinzugefügter und gelöschter Knoten sowie dem Erkennen von Modifikationen der Attribute bestehender Knoten. Weiter sollen Veränderungen in den modellierten Assoziationen erkannt werden, was eine semantische Analyse des Kantenattributs `relType` erforderlich macht.

Wie in 2 dargestellt, hat jeder Primärknoten ausgehende Kanten, die auf Sekundärknoten zeigen und demnach eine Assoziation zwischen den reflektierten Klasseninstanzen abbilden.

Gesucht werden nun Knoten aus den initialen und aktualisierten Graphen, die direkte Nachbarn zu den bereits als identisch identifizierten Elternknoten sind und über eine Kante mit identischen Kantenattributen verfügen. Zwei Knoten werden als identisch gehandhabt, wenn sie das gleiche Kantengewicht (in Form des *relType* Attributes) aufweisen. Nachbarknoten, die nur im initialen, aber nicht im aktualisierten Graphen vorhanden sind, stellen das Resultat einer Entfernen-Operation dar. Die Existenz eines Nachbarknotens im aktualisierten Modell, der nicht im Initialmodell als direkter Nachbar zum aktuellen Knoten zu finden ist, bedeutet im Umkehrschluss, dass eine Hinzufügen-Operation angewendet wurde.

Nach der Ermittlung aller Knotenpaare, die die gleichen Kanten zu den jeweiligen Elternknoten teilen, werden die Paare einem Vergleich ihrer Attributwerte unterzogen. Damit wird ermittelt, ob die Werte bestehender Attribute verändert wurden oder ob Attribute hinzugefügt beziehungsweise entfernt wurden.

Alle Knotenpaare, die auf Grundlage passender Assoziationen als gleich identifiziert wurden, bilden nun den erneuten Input für die erläuterte Prozedur. So wird rekursiv die gesamte Substruktur aus Sekundärknoten analysiert. Das Rekursionslimit definiert sich einerseits durch Knoten, die keine weiteren Kindsknoten besitzen (Blattknoten) und andererseits, wenn ein Knotenpaar referenziert wird, deren Substruktur bereits erfolgreich abgeglichen wurde.

Alle erkannten Änderungen werden passend in ein Diff-Objekt gespeichert, welches die Grundlage für die Patch-Formulierung darstellt.

5 Ergebnisse und Diskussion

Der vorgestellte Ansatz der topologiebasierten Versionierung von BIM-Modellen ermöglicht es, die Limitationen textbasierter Methoden für BIM-Modelle zu überwinden. Dabei wurde mit der Definition eines generischen Graph-Metamodells ein Ansatz getroffen, der durch zahlreiche Datenspezifikationen unterstützt werden können. Die Definition der drei Knotentypen Primär, Sekundär, Verbindungsknoten sowie der Reflektion von Klassennamen im *EntityType* Attribut und der Abbildung von Assoziatonen mithilfe des Kantenattributs *relType* ermöglichen eine Darstellung der Instanzdaten, die einerseits flexibel einsetzbar ist und andererseits keinerlei Informationen der Ursprungsmodelle auslässt.

Als kritisch zu bewerten ist allerdings der rekursive Ansatz zum Abgleich der Sekundärstrukturen, die sich ausgehend von Primärknoten aufspannen. Wenngleich das erläuterte Verfahren die erwarteteten Ergebnisse liefert, so ist dieses Vorgehen insbesondere bei dem Abgleich von Bauteilgeometrien ineffizient. Es gilt daher, in der Zukunft effektivere Methoden einzusetzen, mithilfe derer zwei geometrische Repräsentationen performanter mit einander abgeglichen werden können. In Frage kommen hier beispielsweise Konzepte des *Well Known Texts* oder der Einsatz von Octrees, die eine voxelbasierte Darstellung von Geometrie ermöglichen. Es ist daher zu erörtern, ob die Geometrieversionierung tatsächlich auf dem bisherigen Ansatz der 1:1 Abbildung eines Objektnetzwerks im Graphen der bestmögliche Ansatz ist oder ob eine Hinzunahme weiterer etablierter Konzepte die Versionierung verbessern kann.

6 Ausblick

Die in dieser Veröffentlichung erläuterten Ansätze stellen erste Konzepte dar, wie BIM-basierte Zusammenarbeit zukünftig von einer dateibasierten, hin zu einer objektbasierten Kommunikation erweitert werden kann. Besonderes Augenmerk wurde dabei auf die Erhaltung der weit verbreiteten und fortschrittlichen Datenmodelle entwickelt, die bereits heute in vielen BIM-Applikationen unterstützt werden. Durch den Einsatz graphenbasierter Konzepte und passender technischer Systeme können angewandte Änderungen zwischen zwei Modellversionen adäquat erkannt und formalisiert werden. Damit wird es zukünftig möglich, verteilte Kopien eines Modells mithilfe von Aktualisierungspatches stets auf dem aktuellen Stand zu halten, ohne die gesamte, aktualisierte Modellversion erneut an Projektpartner verteilen zu müssen.

Literatur

[1] John D. Blischak, Emily R. Davenport und Greg Wilson. „A Quick Introduction to Version Control with Git and GitHub“. In: *PLoS Computational Biology* 12.1 (2016), S. 1–18. ISSN: 15537358. DOI: 10.1371/journal.pcbi.1004668.

[2] BSi. *PAS 1192-2: 2013: Specification for information management for the capital/delivery phase of construction projects using building information modelling*. 2013.

[3] CEN. *DIN EN ISO 19650-2:2019: Organisation und Digitalisierung von Informationen zu Bauwerken und Ingenieurleistungen, einschließlich Bauwerksinformationsmodellierung (BIM) - Informationsmanagement mit BIM - Teil 1: Begriffe und Grundsätze*. 2019. DOI: 10.31030/3030494.

[4] Jan Hidders. „A Graph-based Update Language for Object-Oriented Data Models“. Diss. EIdhoven University of Technology, 2001. DOI: 10.6100/IR551259.

[5] ISO. *ISO 10303-11:2004: Industrial automation systems and integration - Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure (ISO 10303-11:1994)*. 2004. URL: https://www.iso.org/standard/38047.html.

[6] Ian Robinson, Jim Webber und Emil Eifrem. *Graph Databases*. Hrsg. von Marie Beaugureau. O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2015. ISBN: 9781491930892. DOI: 10.1016/b978-0-12-407192-6.00003-0.