TECHNISCHE UNIVERSITÄT DRESDEN

**Fakultät Bauingenieurwesen** Institut für Bauinformatik

Diplomarbeit

# BIM-BASIERTE RETTUNGSWEGERMITTLUNG MITTELS GRAPHENTHEORIE

Carsten Kerbitz

Diplomarbeit

# BIM-basierte Rettungswegermittlung mittels Graphentheorie

Development of a BIM based
escape route determination method
utilizing graph theory

Carsten Kerbitz

Betreut durch:
Prof. Dr.-Ing. Raimar Scherer

sowie
M.Sc. Fangzheng Lin und Dipl.-Ing. Ngoc Trung Luu

Eingereicht am 12. April 2019

TECHNISCHE UNIVERSITÄT DRESDEN

**Fakultät Bauingenieurwesen** Institut für Bauinformatik

**Aufgabenstellung für die Diplomarbeit**

Name: cand. Ing. Carsten Kerbitz

Vertiefung: Konstruktiver Ingenieurbau

**Thema:** **BIM-basierte Rettungswegermittlung mittels Graphentheorie**
(Development of a BIM based escape route determination method utilizing graph theory)

**Zielsetzung:**

Die Rettungswegführung stellt bei Gebäudeentwürfen und Brandschutzplanungen eine zentrale Einflussgröße dar. Sowohl Architekten als auch Brandschutzfachplaner müssen nachweisen, dass maximale Rettungsweglängen nicht überschritten und erforderliche Mindestrettungswegbreiten eingehalten werden. Bisher werden hierfür üblicherweise zwei Methoden angewendet: Bei kleineren Objekten wird der Rettungswegnachweis mithilfe von Handrechnungen, basierend auf baurechtlichen Vorschriften, geführt, was mit einem zeitlichen Aufwand verbunden ist und letztlich Ergebnisse mit hohen Toleranzen liefert. Für große Objekte werden Personenstromsimulationen durchgeführt, die exaktere Aussagen über Erfordernisse an Rettungsweglängen und -breiten liefern, dafür aber auch deutlich kostenintensiver sind.
Durch den Einsatz von Building Information Modeling (BIM) ist es möglich, Geometriedaten und Nutzungsparameter direkt aus einem Gebäudemodell zu filtern und zu verarbeiten. Mit dieser Diplomarbeit soll untersucht werden, inwiefern die Nutzung dieser Daten möglich und für die Praxis sinnvoll ist, um Rettungswege automatisch erkennen und bemessen zu lassen. Es soll hierfür ein Softwareprototypentwickelt werden, welcher die Geometriedaten des BIM Modells verarbeitet und damit automatische Erkennung von Rettungswegen aus den beispielhaften Gebäudemodellen ermöglicht.
Im Ergebnis wird ein Nachweis ausgegeben, aus dem zu entnehmen ist, ob für jeden im Gebäude vorhandenen Aufenthaltsraum die maximal zulässige Rettungsweglänge nicht überschritten wird und die vorhandenen Rettungswegbreiten für den zu erwartenden Personenstrom ausreichend dimensioniert sind.
Diese Art der Rettungswegermittlung soll als Kontrolle für die Handrechnungen dienen (sofern es zuverlässig funktioniert, diese auch ersetzen) und als Vorkalkulation für die Machbarkeit von Personenstromsimulation eingesetzt werden.

**Arbeitsumfang:**

Während der Ausarbeitung sollen folgende Punkte bearbeitet werden:

1. Überblick und Bewertung des Standes der Forschung und Technik im Bereich der Rettungswegermittlung in Gebäuden.
2. Betrachtung der Möglichkeiten für den Einsatz von Building Information Modeling zum aktuellen Zeitpunkt für die Rettungswegermittlung.
3. Entwickeln einer BIM-basierten Methodik zur Ermittlung von Rettungswegen in Gebäuden mittels Graphentheorie
4. Erstellen eines Softwareprototypes für die Verifikation der entwickelten Methode.
5. Bewertung des entwickelten Konzepts an einem Beispiel.

Wiss. Betreuer TU Dresden: Dipl.-Ing. Ngoc Trung Luu

ausgehändigt am: 08.11.2018

einzureichen am: 08.03.2019

Prof. Dr.-Ing. Raimar Scherer
Verantwortlicher Hochschullehrer

# Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsinstitution vorgelegt und ist auch noch nicht veröffentlicht worden.

Dresden, 12. April 2019

# Kurzfassung

Jedes Gebäude muss sichere Rettungswege beinhalten, über welche es in einem Brandfall zügig verlassen werden kann. In den Landesbauordnungen sind Vorgaben zu maximalen Rettungsweglängen und der Rettungswegführung festgelegt, deren Einhaltung gegenwärtig i.d.R. mittels zeitaufwendiger Handrechnungen nachgewiesen wird. Die vorliegende Diplomarbeit diskutiert die Machbarkeit einer automatisierten Rettungswegermittlung unter Verwendung von Daten aus dem Building Information Modeling. Im Gegensatz zur klassischen 2D-Planung werden mit diesem Ansatz sowohl geometrische als auch semantische Informationen betrachtet, welche in einem Bauwerksmodell gespeichert werden. In dieser Arbeit wird ein Schema entwickelt, anhand dessen zunächst die relevanten Bauwerksdaten wie Wände, Öffnungen und Notausgänge ebenso wie Informationen zur Personenbelegung aus einem IFC-Bauwerksdatenmodell gefiltert werden können. Die möglichen Wegstrecken innerhalb des Gebäudes werden durch einen Graphen repräsentiert, der mithilfe eines graphentheoretischen Algorithmus auf kürzeste Pfade untersucht wird. Diese stellen die günstigsten Rettungswege in dem untersuchten Gebäude dar. Darüber hinaus wird der Personenstrom je Rettungsweg ermittelt, um eine eventuell notwendige Kapazitätsanalyse zu ermöglichen. Die entwickelte Methodik wird in einem Softwareprototypen implementiert, dessen Funktionalität anhand eines Beispielgebäudes validiert wird. Im Ergebnis wird mit dieser Diplomarbeit nachgewiesen, dass die Nutzung von BIM-Daten zur Rettungswegermittlung grundsätzlich möglich ist. Mit den aktuellen Möglichkeiten des IFC-Datenschemas kann eine vollumfängliche Automatisierung dieses Prozesses jedoch noch nicht erfolgen.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Abkürzungen

| | |
|---|---|
| **BIM** | Building Information Modeling |
| **Brep** | Boundary representation |
| **CAD** | Computer-aided design |
| **CSG** | Constructive solid geometry models |
| **HLS** | Heizung, Lüftung, Sanitär |
| **IFC** | Industry Foundation Classes |
| **JRE** | Java Runtime Environment |
| **KS** | Koordinatensystem |
| **STEP** | Standard for the exchange of product model data |
| **WCS** | World coordinate system |
| **XML** | Extensible Markup Language |
| **XSD** | XML-Schema |

# 1 Einleitung

Für ausnahmslos jedes Gebäude, in welchem sich bestimmungsgemäß Menschen aufhalten, müssen Rettungswege nachgewiesen werden. Da diese bei einem Brandszenario einen unmittelbaren Einfluss auf die Erfolgswahrscheinlichkeit einer Personenrettung und damit auf das Leben von Menschen haben, wird ihnen im Brandschutzwesen eine besondere Bedeutung zugeschrieben. Die Anordnung von Rettungswegen stellt daher für Gebäudeentwürfe eine zentrale Einflussgröße dar, weil der Gebäudegrundriss so angelegt sein muss, dass alle Anforderungen hinsichtlich einer sicheren Rettungswegführung erfüllt werden. Für eine kosteneffiziente Planung sollten Rettungswege bereits im Entwurfsstadium berücksichtigt werden, um nachträgliche Änderungen aufgrund fehlerhaft dimensionierter Rettungswege auszuschließen.

Aufgrund der besonderen Bedeutung für das Leben und die Gesundheit von Menschen werden die Regelungen zum Brandschutz in den Bauordnungen der einzelnen Bundesländer getroffen. In diesen sind maximale Rettungsweglängen, bestimmte Vorgaben bezüglich der Rettungswegführung und teilweise die Dimensionierung von Rettungswegbreiten festgelegt. Gegenwärtig weisen Brandschutzfachplaner mittels Handrechnungen anhand von zweidimensionalen Grundrissplänen nach, dass die vorgeschriebene maximale Rettungsweglänge nicht überschritten wird und die Rettungswegführung insgesamt den bauordnungsrechtlichen Vorschriften entspricht. Dies ist ein zum Teil zeitaufwändiges Verfahren, insbesondere dann, wenn sich Änderungen an dem betrachteten Gebäude ergeben und ein Nachweis erneut geführt werden muss.

Seit einigen Jahren hat sich das „digitale Bauen“ in Form des sogenannten *Building Information Modeling* (BIM) etabliert. Dabei werden geometrische und semantische Informationen zu einem Gebäude in digitalen Modellen hinterlegt, sodass eine ganzheitliche Betrachtung und Verarbeitung der Bauwerksdaten möglich ist. Aktuell ist die BIM-Methodik jedoch noch nicht abschließend entwickelt, sodass die Erweiterung der Möglichkeiten dieser Arbeitsweise derzeit vorangebracht wird. Insbesondere die Fortentwicklung offener Standards für den Datenaustausch steht dabei im Fokus wissenschaftlicher Forschungen.

Für den Datenaustausch verschiedener Fachdisziplinen werden unterschiedliche Modellansätze verfolgt. Ein Ansatz besteht darin, alle Bauwerksinformationen in einem einzigen *Bauwerksdatenmodell* zu hinterlegen. Gegenwärtig existiert mit den *Industry Foundation Classes* (IFC) ein offener Datenstandard, mithilfe dessen die Informationen ausgewählter Fachdisziplinen des Bauwesens gespeichert und plattformunabhängig ausgetauscht werden können. Das Brandschutzwesen ist bislang nur in sehr geringem Umfang in diesem Datenschema berücksichtigt. Dennoch bietet es das Potential, erforderliche Daten für eine automatisierte Rettungswegermittlung bereitzustellen. Damit könnten Rettungswegnachweise zukünftig deutlich schneller erstellt werden, was den Planungsablauf nicht nur in finanzieller, sondern auch in qualitativer Hinsicht optimieren würde.

Gegenstand dieser Arbeit ist eine Untersuchung, ob und inwiefern die bislang durch das Datenschema IFC unterstützen Daten für eine automatisierte Rettungswegermittlung genutzt werden können. Dabei sollen zum einen Geometriedaten aus dem Bauwerksmodell gefiltert werden, zum anderen aber auch Nutzungsparameter wie die Personenbelegung von Räumen. Um die möglichen Wegstrecken (und damit Rettungswege) innerhalb eines

Gebäudes zu ermitteln, wird auf Basis des Gebäudegrundrisses ein Graph erzeugt, welcher alle möglichen Laufwege repräsentiert. Unter Anwendung graphentheoretischer Algorithmen lassen sich die kürzesten Pfade innerhalb des Grundrisses ermitteln, welche dann die vorhandenen Rettungswege darstellen. Aus hinterlegten Nutzungsdaten sollen Personenströme pro Rettungsweg addiert werden, was eine Kapazitätsabschätzung ermöglicht. Ziel ist die Ausgabe eines vollständigen Nachweises, aus dem hervorgeht, ob die bauordnungsrechtlichen Vorschriften hinsichtlich der Rettungswege eingehalten werden. Um die entwickelte Methode zu validieren, wird ein Softwareprototyp erstellt, mit welchem die automatisierte Rettungswegermittlung anhand eines Beispielobjekts durchgeführt werden kann.

Kapitel 2 – *Stand der Forschung und Technik* – führt zunächst in die Thematik des Brandschutzwesens und des Building Information Modeling ein. Es werden die bauordnungsrechtlichen Vorschriften, welche die Rettungswegermittlung behandeln, sowie die Inhalte eines Rettungswegnachweises erläutert. Anschließend werden die BIM-Arbeitsweise sowie das IFC-Datenmodell vorgestellt und die Berührungspunkte zwischen BIM und der Rettungswegermittlung aufgezeigt.

In Kapitel 3 – *Graphentheorie* – erfolgt eine Einführung in das Gebiet der Graphentheorie. Dort werden insbesondere die einzelnen Elemente beschrieben und die Abstraktion eines allgemeinen Problems über einen Graphen erklärt. Dies bildet die Überleitung zum darauffolgenden Kapitel 4 – *Rettungswegermittlung auf Basis graphentheoretischer Ansätze*. Dort wird explizit auf die Überführung von Grundrissdaten in Graphen eingegangen, was mittels zweier unterschiedlicher Ansätze möglich ist. Diese werden ebenfalls diskutiert und ausgewertet, wobei eine der beiden Methoden letztlich in den Softwareprototypen implementiert wird. Zusätzlich erfolgt in diesem Kapitel eine Vorstellung unterschiedlicher graphentheoretischer Algorithmen, mit denen eine „kürzeste-Pfad-Ermittlung“ durchgeführt werden kann. Auch hier wird ein Algorithmus für die Anwendung in dem Softwareprototypen ausgewählt.

Kapitel 5 – *Datenstruktur IFC* – erläutert den Zusammenhang zwischen Bauteilen und Informationen, welche für die Rettungswegermittlung erforderlich sind, sowie deren Repräsentation in dem IFC-Datenschema. Hier wird insbesondere auf die einzelnen IFC-Klassen, ihre Vererbung und definierten Attribute eingegangen. Damit wird ein Überblick gegeben, an welchen Stellen sich in einer IFC-Struktur die erforderlichen Informationen befinden.

Die Implementierung der entwickelten Methodik in den Softwareprototypen wird in Kapitel 6 beschrieben. Zunächst wird hier erläutert, wie die Daten aus einer IFC-Datei ausgelesen und für eine programminterne Weiterverarbeitung aufbereitet werden. Dabei wird auf die jeweiligen Bauteile und deren spezifischen Anforderungen an die Transformation ihrer Daten eingegangen. Anschließend erfolgt die Beschreibung der Implementierung der in Kapitel 4 erläuterten Methodik zur Rettungswegfindung und schließlich der Aufbereitung und Ausgabe der ermittelten Ergebnisse. Abschließend werden die Funktionalität des entwickelten Softwareprototypen anhand eines Beispielgebäudes validiert sowie mögliche Schwierigkeiten und Optimierungspotenzial diskutiert.

# 2 Stand der Forschung und Technik

## 2.1 Stand der Forschung und Technik in der Rettungswegermittlung

### 2.1.1 Überblick

Durch die staatliche Fürsorgepflicht, Gesundheit, Leib und Leben der Bürgerinnen und Bürger zu schützen, ist der vorbeugende Brandschutz sehr umfangreich in den Landesbauordnungen gesetzlich verankert. Grundsätzlich müssen ohne Ausnahme alle Gebäude vier Schutzziele erfüllen, die in allen Landesbauordnungen der Bundesrepublik Deutschland verankert sind. Dort heißt es, dass „bauliche Anlagen [...] so anzuordnen, zu errichten, zu ändern und instand zu halten [sind], dass

1. der Entstehung eines Brandes und
2. der Ausbreitung von Feuer und Rauch vorgebeugt wird

und bei einem Brand

3. die Rettung von Menschen und Tieren sowie
4. wirksame Löscharbeiten möglich sind“ [5].

Konkretisiert werden diese Schutzziele durch Anforderungen an Baustoffe, Bauteile, Zugänge, Rettungswege und weitere Sicherheitseinrichtungen in den Bauordnungen. Das Bauordnungsrecht funktioniert dabei nach dem „Baukastenprinzip“. Durch Größe und Art der Nutzung gelten dem Gebäude entsprechende Brandschutzanforderungen, die aufeinander abgestimmt sind und bei deren Einhaltung ein funktionierender Brandschutz gewährleistet ist. Abweichungen von diesen Vorschriften sind grundsätzlich möglich, sofern diese, gemessen an den oben genannten Schutzzielen, keine Verringerung des Sicherheitsniveaus bedeuten. Gegebenenfalls sind geeignete Kompensationsmaßnahmen zu treffen, mit denen sich ein den Vorschriften gleichwertiges Schutzniveau erreichen lässt.

In den meisten Fällen ist die Anfertigung eines Brandschutzkonzeptes gefordert, in welchem ganzheitlich alle brandschutztechnischen Maßnahmen aufzuzeigen und die Einhaltung der Vorschriften nachzuweisen sind. Rettungswege stehen als zentrales Element im Fokus der modernen Brandschutzplanung. Mit ihnen wird gewährleistet, dass Personen bei einem Brandszenario die Möglichkeit haben, sich in Sicherheit zu bringen und körperliche Schäden von sich abzuwenden. Ebenso bieten sie der Feuerwehr die Möglichkeit, nicht mehr handlungsfähige Personen aus dem Gebäude zu retten und einen inneren Löschangriff durchzuführen.

Bei einem Brandereignis steht den Personen nur ein begrenzter Zeitraum zur eigenständigen Flucht zur Verfügung, da aufgrund der zunehmenden Rauchentwicklung die Selbstrettungsfähigkeit abnimmt. Dieser Zeitraum kann variieren und ist abhängig von Parametern wie der Raumhöhe, den Brandlasten oder Entrauchungseinrichtungen. Ziel muss es sein, die Räumungsdauer niedriger zu halten als die verfügbare Zeit, damit Personen möglichst ohne gesundheitliche Beeinträchtigungen das Gebäude verlassen können [19]. Die Räumungszeit beinhaltet die Detektionszeit (Zeit, bis ein Brand überhaupt bemerkt wird), die

Alarmierungszeit (Zeit, bis alle Personen informiert sind), die Reaktionszeit (Zeit, bis die Personen auf den Alarm reagieren) und die Fluchtzeit, wobei die Summe aus Reaktionszeit und Fluchtzeit die *Evakuierungszeit* ergibt [19].

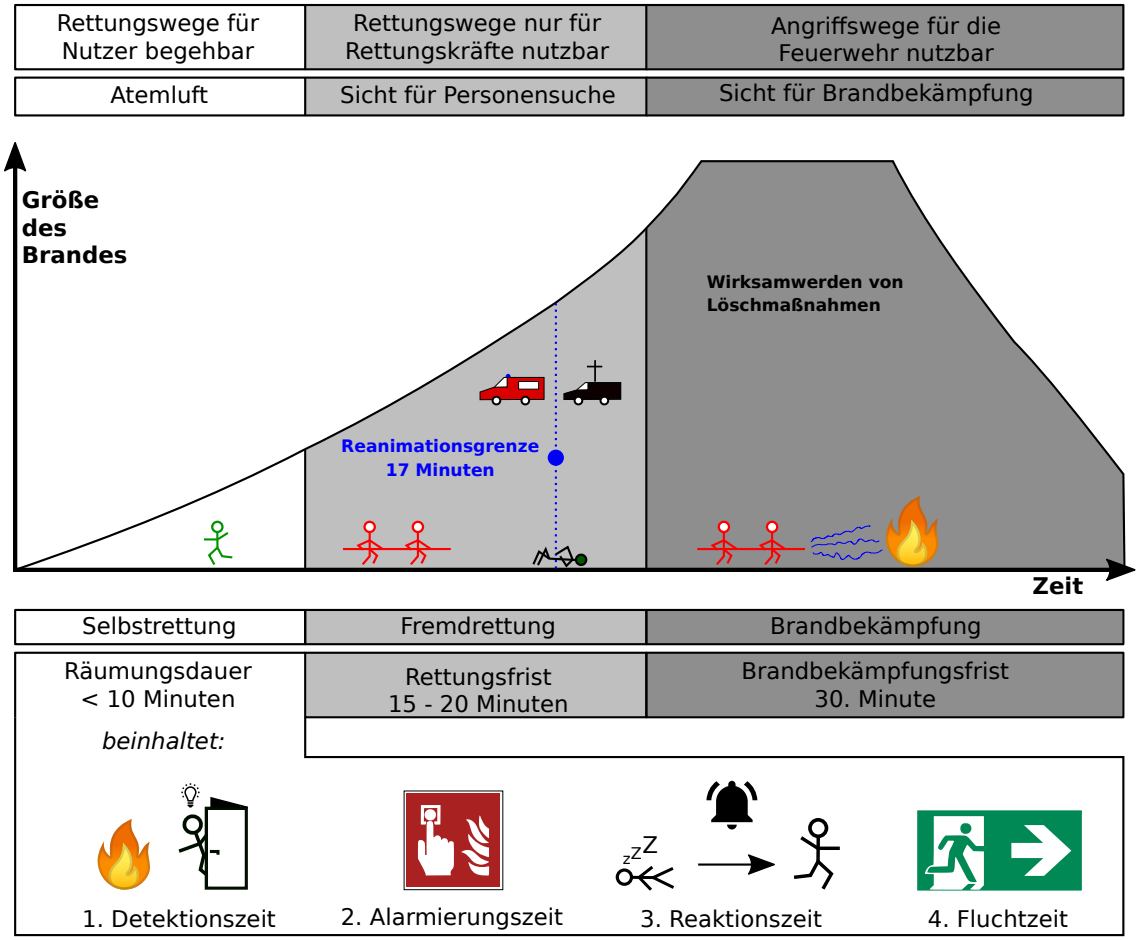



Abbildung 2.1: Verlauf von Brandereignissen [19], bearbeitet

Das Bauordnungsrecht berücksichtigt die gesamte Räumungsdauer insofern, als dass durch eine vorgeschriebene, auf Erfahrung basierte Rettungsweglänge[1] gewährleistet ist, dass aufgrund der begrenzten Wegstrecke die Personen genug Zeit haben, sich selbst in Sicherheit zu bringen. Die einzelnen Zeitfaktoren werden mit diesem Wert pauschal abgedeckt. Dies stellt die konventionelle Methode der Rettungswegermittlung dar, was heute im überwiegenden Teil der Planungen angewandt wird. Eine genauere Beschreibung ist in den Abschnitten 2.1.2 und 2.1.3 gegeben.

Seit etwa 20 Jahren haben sich neben diesem Verfahren auch Ingenieurmethoden etabliert, mit denen im Gegensatz dazu die Evakuierungszeit berechnet werden kann und somit individuellere Aussagen bezüglich der Rettungswegsituation in einem Gebäude machen lassen. Allerdings sind die Ingenieurmethoden deutlich zeitaufwendiger und dementsprechend teurer, weswegen sich ihr Einsatz gegenwärtig nur bei großen und komplexen Sonderbauten lohnt. Dort werden sie hauptsächlich eingesetzt, um Abweichungen von bauordnungsrechtlichen Vorschriften anhand der Einhaltung der vier Schutzziele zu begründen. In Abschnitt 2.1.4 werden diese Verfahren näher vorgestellt.

[1] Siehe dazu Seite 6

### 2.1.2 Begriffsbestimmungen

Die in dieser Arbeit behandelte Thematik basiert vorwiegend auf Grundlagen und Vorschriften des Bauordnungsrechts. Das Bauordnungsrecht ist Teil des öffentlichen Baurechts und wird föderal auf Landesebene umgesetzt. Somit existieren in Deutschland gegenwärtig 16 verschiedene Landesbauordnungen, deren grundsätzliche Strukturen und Gesetze auf der *Musterbauordnung* (vgl. [5]) basieren. Die Musterbauordnung ist kein eigenständiges Gesetz, sondern stellt lediglich die Grundlage für die einzelnen Landesbauordnungen dar, an der sich die Länder orientieren sollen.

Zur allgemeinen Beschreibung nicht landesspezifischer Sachverhalte wird in der Regel immer die Musterbauordnung herangezogen, da sie entsprechend allgemein auf alle Bundesländer angewendet werden kann. Insbesondere die den Brandschutz betreffenden Paragraphen unterscheiden sich in den einzelnen Landesbauordnungen nur an manchen Stellen; die meisten Vorschriften sind identisch aus der Musterbauordnung übernommen.

In dieser Arbeit wird ausschließlich auf die Musterbauordnung Bezug genommen, da die dort genannten und für diese Arbeit relevanten Vorschriften hinsichtlich der Rettungswege auch wortwörtlich in allen Landesbauordnungen vorhanden und dementsprechend auch in allen Bundesländern rechtsverbindlich sind. In diesen Vorschriften werden einige Begriffe genannt, denen eine tiefer gehende Definition zugrunde liegt. Diese Begriffe werden nachfolgend erläutert, um ein einheitliches Verständnis für bestimmte Sachverhalte zu schaffen.

#### Rettungsweg

Wege, die Personen zurücklegen müssen, um sich im Gefahrenfall in Sicherheit zu bringen, nennen sich *Fluchtwege*. Sie führen entweder ins Freie oder in einen sicheren Bereich. Der Begriff „Fluchtweg“ zielt auf die Flucht von Personen, das heißt auf die Selbstrettung, ab. Der Begriff „Rettungsweg“ definiert im engeren Sinne Wege, die von Rettungskräften begangen werden müssen, um anderen Personen im Gefahrenfall Hilfe leisten zu können. Hierbei handelt es sich also um Fremdrettung. Ferner gibt es noch die sogenannten „Angriffswege“, die der Feuerwehr zur Rettung von Personen und zum Löschangriff dienen [11].

Während in arbeitsrechtlichen Vorschriften (wie zum Beispiel den *Technischen Regeln für Arbeitsstätten* [10]) grundsätzlich von *Fluchtwegen* die Rede ist, verwendet das Bauordnungsrecht konsequent den Begriff *Rettungsweg*. Im Zusammenhang mit diesen Vorschriften sind mit beiden Begriffen Wege gemeint, die sowohl der Selbst- als auch der Fremdrettung sowie einem Löschangriff der Feuerwehr dienen. Ist bauordnungsrechtlich ein Rettungsweg gefordert, so muss dieser Weg zwingend sowohl der Flucht als auch der Rettung dienen können. Die Begriffe „Rettungsweg“ und „Rettungswegermittlung“ stehen in dieser Arbeit immer im Kontext des Bauordnungsrechts, weshalb auch hier immer bidirektionale Wege gemeint sind. In der Regel entsprechen alle Wege innerhalb eines Gebäudes dieser Definition von Rettungswegen, weswegen der Begriff in dieser Arbeit auch nicht weiter differenziert werden muss.

Bauordnungsrechtlich werden für jede Nutzungseinheit mit Aufenthaltsräumen zwei voneinander unabhängige Rettungswege ins Freie gefordert, die allerdings innerhalb des Geschosses über denselben notwendigen Flur (siehe Seite 7) führen dürfen [5]. Nicht zu ebener Erde liegende Nutzungseinheiten mit Aufenthaltsräumen müssen über eine *notwendige Treppe* erschlossen werden. Notwendige Treppen und Ausgänge ins Freie werden auch als bauliche Rettungswege bezeichnet.

Gemäß § 33 (3) der Musterbauordnung muss ein zweiter Rettungsweg nicht zwingend baulich sein, sondern auch über Rettungsgeräte der Feuerwehr (Leitern) führen. Dies gilt jedoch nur unter der Voraussetzung, dass die Feuerwehr hinsichtlich der Personenrettung keine Bedenken hat und für bestimmte Gebäudehöhen erforderliche Hubrettungsfahrzeuge besitzt. Auf einen zweiten Rettungsweg kann vollständig verzichtet werden, sofern „die Rettung über einen sicher erreichbaren Treppenraum möglich ist, in den Feuer und Rauch nicht eindringen können“ [5].

### Notwendiger Treppenraum und maximale Rettungsweglänge

Der Treppenraum ist ein Raum innerhalb eines Gebäudes, der eine Treppe beinhaltet, über die mehrere oder meist alle Geschosse des Gebäudes erschlossen werden können. Allgemein gebräuchlicher ist der Begriff „Treppenhaus“, der historisch darauf zurückzuführen ist, dass Treppen in alten Festungsanlagen häufig in separaten Türmen oder Gebäuden („Häusern“) untergebracht waren. Da heutzutage jedoch nahezu alle Treppen innerhalb des Hauptgebäudes nur in einem eigenen Raum liegen, spricht der Gesetzgeber konsequent von „Treppenräumen“ [5].

Bauordnungsrechtlich ist ein notwendiger Treppenraum ein gegenüber anderen Räumen und Fluren brandschutztechnisch wirksam abgetrennter Raum, in dem sich eine *notwendige Treppe* befindet. Die Abtrennung des Treppenraums erfolgt mittels feuerwiderstandsfähiger Umfassungsbauteile sowie rauchdichten und oft feuerhemmender Türen, weswegen ein Treppenraum als sicherer Bereich gilt.

Aus diesem Grund ist die maximale Rettungsweglänge in § 35 der Musterbauordnung festgelegt, also dem Paragraphen, der die notwendigen Treppenräume beschreibt. Dort heißt es, dass „von jeder Stelle eines Aufenthaltsraumes sowie eines Kellergeschosses [...] mindestens ein Ausgang in einen notwendigen Treppenraum oder ins Freie in höchstens 35 m Entfernung erreichbar sein [muss]“ [5]. Das bedeutet, dass die maximale Länge eines Rettungsweges nur bis zu der Tür zu einem notwendigen Treppenraum (oder einem Ausgang ins Freie) nachzuweisen ist, wenngleich ein Rettungsweg natürlich nicht am Treppenraum endet, sondern über diesen ins Freie fortgeführt werden muss. Diese Wegstrecke wird der nachzuweisenden Rettungsweglänge jedoch nicht hinzugerechnet. Die Beschränkung der maximalen Rettungsweglänge bezieht sich nur auf den *ersten* Rettungsweg. Der *zweite* Rettungsweg unterliegt keinen bauordnungsrechtlichen Längenbeschränkungen.

Dass ein Treppenraum zum *notwendigen* Treppenraum wird ist abhängig davon, ob er eine *notwendige* Treppe beinhaltet. Existiert in einem Gebäude bereits eine notwendige Treppe, sind weitere Treppen nicht notwendig, weshalb an diese sowie die sie beinhaltenden Räume keine brandschutztechnischen Anforderungen bestehen. Sofern allerdings diese weiteren Treppen als Rettungsweg erforderlich werden, sind sie *notwendige* Treppen; die sie umschließenden Räume dementsprechend *notwendige Treppenräume*.

### Rettungswegbreite

Mit dem Begriff „Rettungswegbreite“ wird die nutzbare Breite aller Elemente im Verlauf eines Rettungsweges bezeichnet, also die nutzbare Breite von notwendigen Fluren, notwendigen Treppen sowie das lichte Öffnungsmaß von Türen. Die Musterbauordnung schreibt hier keine konkreten Werte vor, sondern verlangt lediglich, dass sie „für den größten zu erwartenden Verkehr ausreichen“ [5].

### Notwendige Flure

Als *notwendige Flure* sind Flure auszubilden, die innerhalb einer Nutzungseinheit mit einer Grundfläche von mehr als 200 $m^2$ (Büro- und Verwaltungsgebäude 400 $m^2$) liegen [5]. Notwendige Flure haben brandschutztechnische Anforderungen, um als Teil von Rettungswegen sowohl eine Selbst- als auch eine Fremdrettung zu ermöglichen.

### Aufenthaltsraum

Obwohl „Aufenthaltsraum“ in der Musterbauordnung ein zentraler Begriff ist, wird er in dieser jedoch an keiner Stelle klar definiert. Etwas genauer werden hier die Verwaltungsvorschriften und Durchführungsverordnungen zu den jeweiligen Landesbauordnungen. So benennt beispielsweise die Verwaltungsvorschrift zur Sächsischen Bauordnung Wohn- und Schlafräume, Wohndielen, Wohn- und Kochküchen, Versammlungsräume, Arbeitsräume, Gasträume, Unterrichtsräume, Krankenräume, Warteräume, Geschäftsräume, Verkaufsräume und Werkstätten als Aufenthaltsräume [33]. In Fachkreisen ist unter anderem auch die Definition anerkannt, dass es sich bei einem Raum um einen Aufenthaltsraum handelt, wenn er mehr als zwei Stunden pro Tag genutzt wird. Im Zweifel ist allerdings immer eine sachkundige Feststellung durch geeignete Sachverständige zu erstellen.

### Nutzungseinheit

Auch der Begriff „Nutzungseinheit“ wird in der Musterbauordnung häufig gebraucht, da Nutzungseinheiten nicht zuletzt auch für den Rettungswegnachweis von entscheidender Bedeutung sind. Außer, dass die Grundfläche einer Nutzungseinheit die Brutto-Grundfläche ist, wird in der Musterbauordnung allerdings keine weitere Erklärung zu deren Bedeutung geliefert. Auch hier geben erst die Verwaltungsvorschriften der einzelnen Landesbauordnungen genauere Hinweise, wie zum Beispiel die Verwaltungsvorschrift zur Sächsischen Bauordnung. Darin heißt es, dass eine Nutzungseinheit als „eine in sich abgeschlossene Folge von Aufenthaltsräumen [gilt], die einer Person oder einem gemeinschaftlichen Personenkreis zur Benutzung zur Verfügung stehen“ [33]. Das können Wohnungen, Büros oder Praxen sein, ebenso wie Einraumwohnungen.

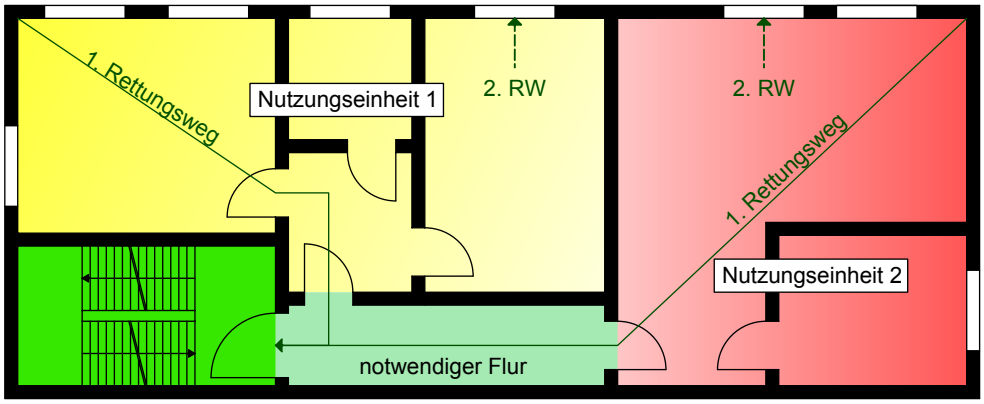



Abbildung 2.2: Zulässige Rettungswegführungen zweier Nutzungseinheiten

Hinsichtlich der Rettungswegermittlung ist zu beachten, dass für jede Nutzungseinheit ein erster Rettungsweg zu einem notwendigen Treppenraum oder einem Ausgang ins Freie

unabhängig von anderen Nutzungseinheiten nachgewiesen werden muss. Das bedeutet, dass ein erster Rettungsweg von einer Nutzungseinheit nicht durch eine andere Nutzungseinheit verlaufen darf, so wie in Abbildung 2.3.

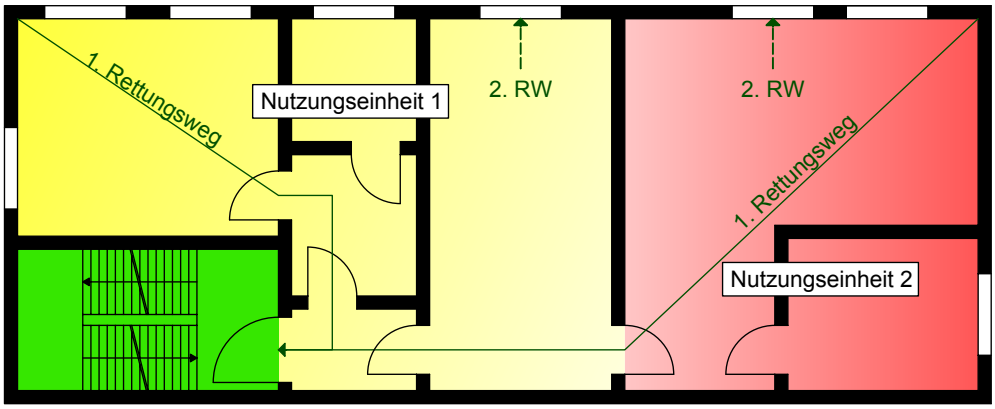



Abbildung 2.3: Unzulässige Rettungswegführungen zweier Nutzungseinheiten

Im Gegensatz dazu darf der zweite Rettungsweg über eine benachbarte Nutzungseinheit führen, sofern dauerhaft sichergestellt ist, dass der Zugang zu dieser Nutzungseinheit jederzeit möglich ist.

#### Sonderbau

Wohngebäude sowie kleinere Bürogebäude zählen in der Regel zu den Standardbauten, auf welche die Vorschriften der Bauordnungen immer anzuwenden sind. In § 2 (4) der Musterbauordnung werden jedoch eine Reihe an Tatbeständen aufgeführt, bei deren Erfüllung eine bauliche Anlage als „Sonderbau“ eingestuft wird. Beispiele hierfür sind Hochhäuser, Schulen, Krankenhäuser oder Versammlungsstätten.

An Sonderbauten können besondere Anforderungen gestellt, aber auch Erleichterungen hinsichtlich bauordnungsrechtlicher Vorschriften gewährt werden. Für einige Sonderbauten existieren bestimmte Sonderbauvorschriften, welche über den Vorschriften der Muster- bzw. Landesbauordnungen stehen. Liegt bei einer baulichen Anlage ein Sonderbaustatus vor, so muss die entsprechende Sonderbauvorschrift eingehalten werden, wenn sie vorhanden ist. In diesem Fall handelt es sich dann um einen *geregelten Sonderbau*. Sofern keine Sonderbauvorschriften vorliegen, handelt es sich um einen *ungeregelten Sonderbau*, was den Brandschutzfachplanern eine größere Freiheit bei der Auswahl von Maßnahmen ermöglicht, gleichzeitig aber auch ein hohes Maß an Entscheidungskompetenz abverlangt, da es für bestimmte Sachverhalte keine gesetzliche Grundlage gibt.

### 2.1.3 Rettungswegermittlung anhand bauordnungsrechtlicher Vorschriften

Die in den meisten Fällen angewandte Methode, Rettungswegläng en und -breiten zu ermitteln, ist die klassische geometrische Messung anhand von Plänen. Da heutzutage alle Pläne auch digital vorliegen, ist das Ausmessen von Strecken jeglicher Art in einem CAD-Programm recht einfach. Bei großen Objekten, in denen sehr viele Rettungswege nachzuweisen sind, kann dies jedoch einen großen zeitlichen Aufwand bedeuten. Dennoch ist es

die bislang einzige Möglichkeit, Rettungsweglängen und -breiten aus einem Plan oder einer Datei zu erhalten, um sie anhand der vorgeschriebenen Werte zu beurteilen.

Zunächst wird dabei der (vermeintlich) am weitesten von einem notwendigen Treppenraum oder einem Ausgang ins Freie entfernte Punkt innerhalb eines Aufenthaltsraumes ermittelt und die (vermeintlich) kürzeste Strecke zu den entsprechenden sicheren Bereichen gemessen. Ist diese kürzer oder gleich 35 Meter, so gelten die bauordnungsrechtlichen Vorschriften als erfüllt (siehe Abbildung 2.4). Es ist dabei ausreichend, den offensichtlich

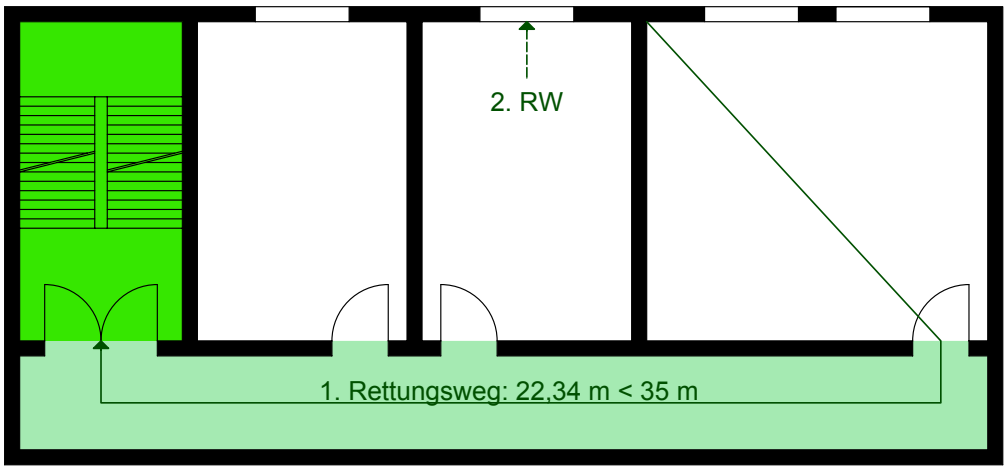



Abbildung 2.4: Gemessene Rettungsweglänge in einer Nutzungseinheit

längsten Rettungsweg aus einem Raum nachzuweisen, da die Rettungswege aus anderen Räumen, welche zu demselben notwendigen Treppenraum führen, augenscheinlich kürzer sind, so wie in Abbildung 2.4. In Tiefgaragen oder anderen baulichen Anlagen, in denen sehr große Räume vorhanden sind ist es auch üblich, von einem Treppenraum einen 35 Meter Radius zu ziehen und zu prüfen, ob alle Bereiche damit abgedeckt werden.

Rettungswegbreiten müssen in diesem Zuge ebenfalls nachgewiesen werden. Da die Musterbauordnung keine konkreten Maße vorgibt, liegt es im Ermessen der Entwurfsverfasser bzw. Fachplaner, die vorhandenen Breiten anhand der Personenzahl einzuschätzen. Es kann dabei helfen, Maße aus anderen Vorschriften oder Richtlinien heranzuziehen (z.B. Technische Regeln für Arbeitsstätten oder DIN 18065), wenngleich dadurch keine Rechtsverbindlichkeit oder Rechtssicherheit entsteht.

Sofern es sich bei dem betrachteten Gebäude um einen Sonderbau handelt, sind eventuell vorhandene Sonderbauvorschriften anzuwenden. In diesen sind häufig konkrete Angaben zu Rettungswegbreiten gemacht, wie zum Beispiel in der *Muster-Versammlungsstättenverordnung*. Dort ist festgelegt, dass die nutzbare Rettungswegbreite 1,20 m je 200 darauf angewiesene Personen betragen muss [6]. Die *Muster-Schulbaurichtlinie* schreibt 1,20 m ebenfalls als Mindestrettungswegbreite vor, allerdings mit dem Zusatz, dass Staffelungen nur in Schritten von 0,60 m zulässig sind. Dieser Wert beruht auf der Annahme, dass einer einzelnen Person 60 Zentimeter Breite zugeordnet werden und 100 Personen darüber in annehmbarer Zeit evakuiert werden können, ohne dass Panik ausbricht oder Rettungswege verstopfen. Darüber hinaus werden in der Muster-Schulbaurichtlinie noch Mindestvorgaben für Ausgänge aus Unterrichtsräumen (0,90 m) und notwendige Flure (1,50 m) gemacht [4]. Diese Werte sind immer nur für den jeweiligen Sonderbau verbindlich, weshalb zwingend

eine Einstufung nach § 2 (4)[2] der Musterbauordnung durch die Entwurfsverfasser oder die Brandschutzfachplaner erfolgen muss. Ferner können auch sie ebenfalls als Hilfestellung genutzt werden, um bei Standardgebäuden und bekannten Nutzerzahlen eine valide Einschätzung bezüglich der notwendigen Rettungswegbreite zu treffen.

Kommt es zu Abweichungen von bauordnungsrechtlichen Vorschriften, so hat der Brandschutzfachplaner eine Bewertung zu treffen, ob diese Abweichung im Rahmen tolerierbarer Grenzen liegt oder Kompensationsmaßnahmen zu ergreifen sind, um die definierten Schutzziele anderweitig zu erreichen. Eine vorhandene Rettungsweglänge von 35,5 m $>$ 35 m kann mit einem Hinweis auf die Geringfügigkeit der Überschreitung hingenommen werden, während ein um fünf Meter längerer Rettungsweg möglicherweise den Einsatz einer Brandmeldeanlage zur frühzeitigen Alarmierung und Evakuierung erfordert. Parameter, die solche Kompensationsmaßnahmen beeinflussen, sind neben geometrischen Verhältnissen auch Bestandsschutz, Ortskundigkeit und Selbstständigkeit der Personen und Sichtverhältnisse.

Derartige Abweichungen lassen sich mit verbalen Ausführungen der Brandschutzplaner im Brandschutzkonzept begründen und sind gängige Praxis. Bei sehr großen Gebäuden oder komplexen Strukturen stößt diese Art an ihre Grenzen, wenn nicht mehr nachvollziehbar ist, wie weit der Einfluss der Kompensationsmaßnahmen reicht und ob dies schlussendlich die Schutzziele ausreichend erfüllt. In diesem Falle stehen den Brandschutzfachplanern Ingenieurmethoden zur Verfügung, anhand derer sich rechnerisch bestimmte Situationen simulieren lassen, um damit Rückschlüsse auf die Qualität der Rettungswege zu ziehen. Sie werden im folgenden Abschnitt beschrieben.

### 2.1.4 Rechnerische Rettungswegermittlung

Die Einhaltung bauordnungsrechtlicher Vorschriften zu unterschiedlichen brandschutzrelevanten Komponenten (Abschottung, Alarmierung, Rauchfreihaltung) führt zu einem definierten Sicherheitsniveau, welches der Erfüllung der bauordnungsrechtlichen Schutzziele entspricht [20]. Zwar sind die Anforderungen an bestimmte Bauteile abhängig von der Gebäudehöhe, -art und -ausdehnung, eine individuelle Betrachtung der gebäudespezifischen Besonderheiten ist damit jedoch nicht vollends möglich. So könnte aufgrund der gegebenen Situation die Einhaltung bestimmter Vorschriften gar nicht nötig sein, während in anderen Fällen die Bauordnung die Situation nur unzureichend abdeckt (wie bei ungeregelten Sonderbauten).

Im Gegensatz zu den deskriptiven Argumenten, mit denen Sachverhalte anhand von Vorschriften, Erfahrungen und persönlichen Einschätzungen begründet werden, handelt es sich bei ingenieurmäßigen Verfahren um „handfeste“ rechnerische Nachweise. Mit diesen können die beeinflussenden Randbedingungen und Parameter deutlich genauer berücksichtigt werden, sodass die auf das Objekt zugeschnittene Lösung im Ergebnis einen sicheren und wirtschaftlichen Gebäudebetrieb ermöglicht [20].

Obgleich es sich bei den angewandten Rechenverfahren um fundierte, allgemein anerkannte Regeln der Technik handelt, sind eine verständliche Darstellung und eine Dokumentation für Dritte zwingend notwendig [20]. Sind Rechnungen und Modellannahmen für die den Brandschutz prüfende Stelle nicht nachvollziehbar, so wird einer Abweichung trotz aufwendiger Berechnungen nicht zugestimmt.

Gegenwärtig existieren zwei Ansätze, mit denen Personenströme berechnet werden können. Das sind zum einen *hydraulische Modelle*, nach denen die meisten Handrechenverfahren

[2] Sonderbautatbestände

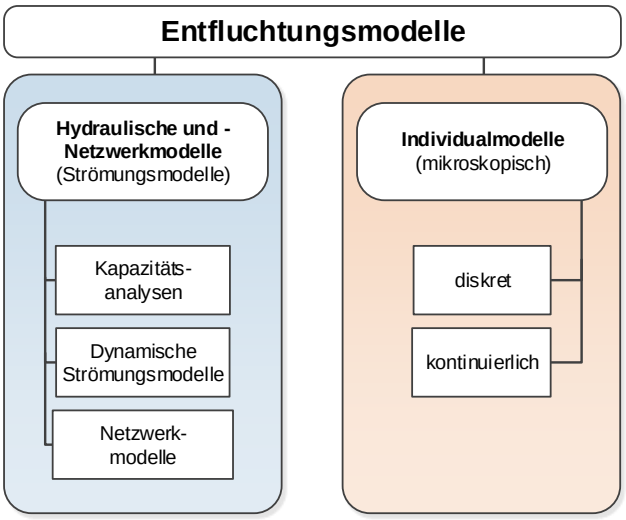



Abbildung 2.5: Entfluchtungsmodelle, nach [20]

funktionieren, sowie *Individualmodelle*, die nur mittels Computerrechnungen ausgewertet werden können [19]. Abbildung 2.5 zeigt eine Übersicht gegenwärtig existierender Modelle. Beiden Ansätzen gemein ist, dass sie im Ergebnis Evakuierungszeiten ausgeben, die es zur Evakuierung eines Gebäudes benötigt.

### Hydraulische Modelle

Bei dem hydraulischen Ansatz wird der Personenstrom wie ein fluides Medium angesehen, das durch ein modellhaftes Leitungssystem strömt [21]. Ein Rettungsweg besteht dabei aus verschiedenen Wegelementen (Ausgang, Treppe, Flur etc.), für die aus empirischen Untersuchungen Daten hinsichtlich des spezifischen Durchflusses in Abhängigkeit zur Personendichte vorliegen [20]. Die Personengruppe wird dabei als homogen und der Personenstrom als stationär betrachtet. Ergebnis der Berechnung ist die Zeit, bis die letzte Person dieser Personengruppe über den vorgegebenen Rettungsweg ein Ziel erreicht hat.

Die *Kapazitätsanalyse* geht dabei von statischen Parametern für die einzelnen Wegstrecken aus. Die Kapazität eines Wegelements steht in Abhängigkeit von seiner Breite und weiteren Einflussgrößen wie z.B. Stufenabmessungen, was die Berechnungsansätze zur Bestimmung der Fluchtzeit durch ein Wegelement liefert [20]. In *dynamischen Strömungsmodellen* können weitere Parameter berücksichtigt werden, insbesondere solche, die Personenbewegungen mit in die Berechnung einfließen lassen. [20]. Beispielhaft sei hier das Verfahren nach *Predtetschenski und Milinski* genannt, welches ein weit verbreitetes Verfahren bei Evakuierungsberechnungen darstellt. Als letzte Erweiterungsstufe gibt es die sogenannten *Netzwerkmodelle*. Dort werden die kritischen Wegelemente als Knoten eines Systems dargestellt, in dem Informationen zu der Art und Abmessungen dieser Elemente hinterlegt sind [20]. Zusätzlich lassen sich individuelle Parameter wie beispielsweise Mobilitätseinschränkungen oder Wahlmöglichkeiten bei Fluchtwegalternativen berücksichtigen. [17].

Prinzipiell sind die Ergebnisse aus Berechnungen nach dem hydraulischen Ansatz als optimistisch einzuschätzen, da die Einflussgrößen zumeist statisch sind und der Personenstrom

als einheitliches Fluid ohne individuelle Verhaltensweisen betrachtet wird [19]. Zudem muss immer ein Rettungsweg vorgegeben werden, um die Evakuierungsdauer berechnen zu können. Zwar existieren mit den dynamischen Strömungsmodellen und den Netzwerkmodellen Ansätze, die Berechnung zu optimieren, noch genauere Ergebnisse liefern jedoch die im nächsten Abschnitt vorgestellten *Individualmodelle*.

#### Individualmodelle

Bei *Individualmodellen* wird der Personenstrom nicht als ganzheitliche Masse betrachtet, sondern jede Person als einzelnes Individuum, das unterschiedliche Eigenschaften hat, eigene Entscheidungen treffen und andere Personen beeinflussen kann [20]. Vorzugebene Parameter hierfür sind die individuelle Mobilität und Charakterisierung bestimmter Verhaltensweisen (z.B. Fluchtwegeauswahl) [17]. Aus der Simulation dieser individuellen Bewegungsabläufe ergibt sich ein Zusammenhang von Dichte und Gehgeschwindigkeit, was die Räumungszeit wesentlich beeinflusst [17]. Da die Bewegungen individuell und kleinschrittig simuliert werden, ist dies nur mit computergestützten Berechnungen möglich [20].

Unter den Individualmodellen gibt es zwei unterschiedliche Typen, das *räumlich diskrete* sowie das *kontinuierliche* Modell. Bei ersterem werden die verfügbaren Laufflächen eines Gebäudes in ein feines Gitternetz diskretisiert, in welchem sich die Personen von Zelle zu Zelle fortbewegen können. Auf einem Element kann sich stets nur eine Person befinden. Die individuelle Bewegung ist abhängig von dem eigenen Ziel und der angestrebten Bewegung benachbarter Personen [17]. Durch die Gitterstruktur werden individuelle Mobilitätsparameter bei der Simulation jedoch nur eingeschränkt oder gar nicht berücksichtigt. Beim räumlich kontinuierlichen Modell entfällt die Rastereinteilung, wodurch sich die Personen innerhalb der begrenzenden Raumgeometrie vollständig frei bewegen können. Darüber hinaus ist die Einbeziehung von Körpermaßen möglich, wodurch sich insgesamt eine hohe Flexibilität ergibt und gegenüber den räumlich diskreten Modellen noch genauere Ergebnisse erzeugt werden. [17].

Aufgrund der Komplexität des Themengebiets können Evakuierungsberechnungen im Rahmen dieser Arbeit nicht berücksichtigt werden. Da sie eine wichtige Ergänzung zu den begrenzen Möglichkeiten der händischen Rettungswegermittlung sind, sollten sie hier jedoch nicht unerwähnt bleiben. Zudem könnte ihr Einsatz nicht zuletzt auch wegen der Möglichkeit, Daten des Building Information Modeling als Grundlage für die Modellbildung automatisch zu verarbeiten, in Zukunft zunehmen. Hierzu hat es bereits erfolgreiche Untersuchungen gegeben, Bauwerksdaten in Simulationsprogramme zu übertragen [16].

## 2.2 Stand der Forschung und Technik im Building Information Modeling

### 2.2.1 Konventionelle Planung

Die klassische Hochbauplanung basiert seit der Entstehung des Bauwesens auf zweidimensionalen Plänen, welche das Gebäude in Grundrissen, Schnitten und Ansichten darstellen, um den am Bau Beteiligten die Konstruktion zu visualisieren. Zur Erstellung dieser Pläne hat sich die Arbeit mit CAD-Programmen etabliert, wodurch eine einfache und zügige Arbeitsweise und die schnelle Verbreitung von Daten möglich ist. Auch die Verwendung von dreidimensionalen Darstellungen ist durch die Anwendung von CAD-Programmen günstig

geworden, wodurch Planern und bauausführenden Unternehmen erweiterte Möglichkeiten zur Verfügung stehen.

Der Austausch dieser Daten erfolgt in digitaler Form häufig als *.dxf* oder *.pdf* Datei, wobei es immer noch zu Schwierigkeiten kommen kann, insbesondere wenn verschiedene Planer mit unterschiedlichen Programmen arbeiten. Zudem liefern auch die dreidimensionalen Pläne nur geometrische Informationen über das Gebäude, jedoch keine semantischen. So müssen einzelne Fachplaner ihre eigenen Fachmodelle immer zusätzlich erstellen und die daraus gewonnenen Erkenntnisse (z.B. Öffnungen von Lüftungsleitungen) müssen wieder händisch in die ursprüngliche geometrische Zeichnung eingefügt werden. Der Entwurfsverfasser hat zudem darauf zu achten, dass die einzelnen Fachmodelle aufeinander abgestimmt sind und bei nachträglichen Änderungen diese auch in allen Modellen berücksichtigt werden, was nicht nur zeitaufwändig, sondern auch fehleranfällig ist.

### 2.2.2 Building Information Modeling (BIM)

Die Probleme, welche bei dem Datenaustausch bei konventionellen Planungen auftreten, können durch den Einsatz von *Building Information Modeling*, kurz *BIM* genannt, vermieden werden. BIM ist kein Programm, sondern der Überbegriff für eine Methode zur Bauwerksdatenmodellierung mithilfe von Software. Dabei geht es um mehr als die reine dreidimensionale Gebäudemodellierung. Bei der BIM-Methodik werden den einzelnen Bauwerkselementen Objekteigenschaften zugewiesen, die wiederum untereinander und mit externen Daten verknüpft werden können, was ermöglicht, den Planungsprozess, die Bauausführung und auch die Gebäudebewirtschaftung zu optimieren.

BIM versteht sich als ganzheitliche Arbeitsweise unter Verwendung aller Daten während des Baulebenszyklus [27]. Neben den geometrischen Daten gehören zu einem Gebäude alphanumerische Daten (z.B. Raumbuch, Bauteillisten, Leistungsverzeichnisse), Berechnungsdaten (z.B. Stabwerksmodelle, FEM), Prozessdaten (Bauablaufpläne), Kommunikationsdaten, Sensordaten und Metadaten [7]. Durch das Zusammenführen dieser Daten ist eine Betrachtung systemübergreifender Zusammenhänge möglich, was insbesondere bei Großprojekten aufgrund der Komplexität dieser Bauten kaum möglich ist.

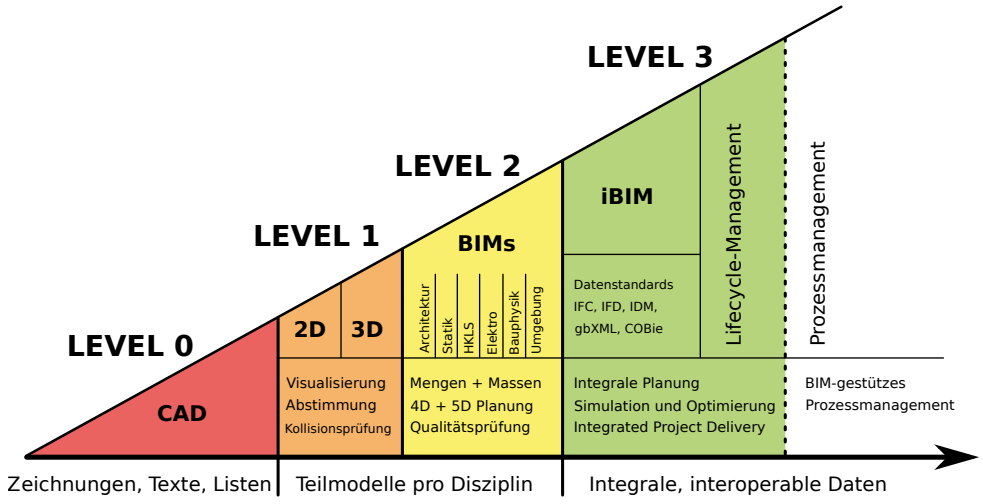



Abbildung 2.6: BIM-Level, nach [24]

In der Fachwelt ist häufig von den vier Stufen des BIM die Rede (siehe Abbildung 2.6). Damit wird der Umfang der BIM-Methodik ausgedrückt, wobei erst ab Level 2 eine „echte“ BIM-Arbeitsweise vorhanden ist. Level 0 beinhaltet zunächst nur zweidimensionale Geometriedarstellungen, weshalb es sich hier um die konventionelle Arbeitsweise handelt und daher als nulltes Level bezeichnet wird. In Level 1 werden zwei- und dreidimensionale Darstellungen erstellt, in denen bereits interdisziplinäre Inhalte abgebildet und überlagert werden können (z.B. sind fehlende Wanddurchbrüche für Lüftungsleitungen erkennbar). Im Gegensatz zu Level 2 fehlen in diesem Level semantische Informationen. Durch diese zeichnet sich das Building Information Modeling erst aus, da hier nicht mehr nur geometrische Informationen hinterlegt sind, sondern eben auch semantische. Dadurch lassen sich objektspezifische Eigenschaften der einzelnen Bauelemente darstellen, ermitteln und von den jeweiligen Fachdisziplinen weiterverarbeiten. Dieses Level entspricht dem *little-BIM* [24]. Für BIM-Level-3 muss dieses Level noch um den interdisziplinären Informationsaustausch der einzelnen, fachspezifischen und semantischen Teilmodelle erweitert werden. Hier stehen verschiedene Fachmodelle in Kommunikation miteinander und können ihre Informationen untereinander austauschen. Auch das Lifecycle-Management kann in diesem Level berücksichtigt werden. Ab diesem Umfang handelt es sich um *big-BIM* [24].

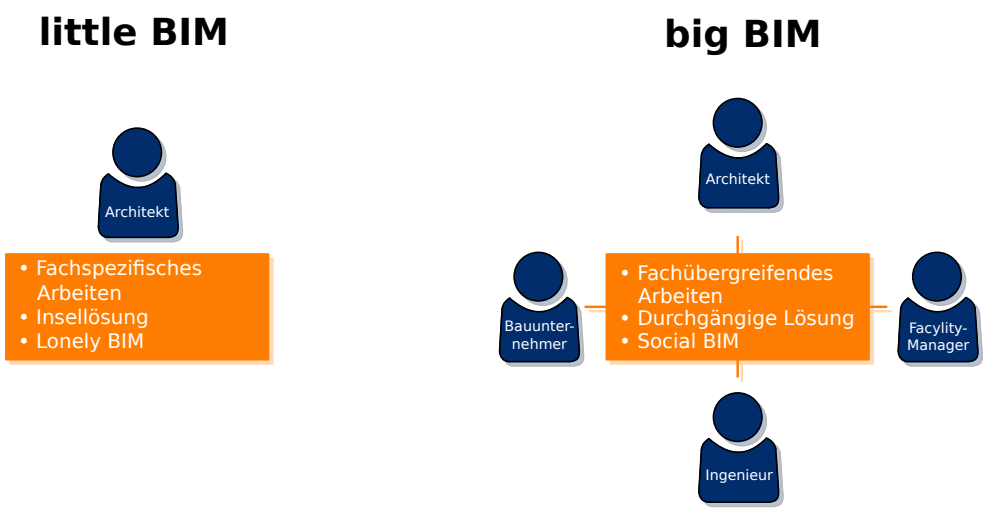



Abbildung 2.7: Little-BIM vs. big-BIM, nach [23]

Die Anwendung von Building Information Modeling hat sich zum gegenwärtigen Zeitpunkt etabliert und wird bei vielen Projekten im In- und Ausland in unterschiedlichem Umfang angewandt. Es ist derzeit aber noch nicht möglich, alle Bereiche des Bauwesens mit der BIM-Methodik zu erfassen, sodass die Anwendung von BIM unterschiedlich weit fortgeschritten ist.

### Open-BIM vs. closed-BIM

Damit der Datenaustausch im BIM erfolgreich funktioniert, müssen alle beteiligten Systeme mit allen Daten umgehen können. An dieser Stelle ist zwischen zwei Ansätzen des BIM-Prozesses zu unterscheiden, dem *closed-BIM* und dem *open-BIM*. Beim closed-BIM gibt es eine einheitliche Software, in der sämtliche Fachmodelle wie Gebäudeentwurf, Tragwerksplanung, HLS-Planung etc. erstellt werden können. Das führt zu einer unkomplizierten

Koordination der einzelnen Fachplanungen, da alles mit einem Programm bzw. einer Programmfamilie eines Softwareanbieters bearbeitet wird und dadurch die jeweiligen Fachmodelle optimal ineinander greifen. Dieser geschlossene Prozess geht jedoch mit dem Nachteil einher, dass ein Datenaustausch zu externen Projektpartnern, die nicht die gleiche Software verwenden, nur eingeschränkt oder gar nicht möglich ist. Der Im- und Export von Daten in andere Herstellerformate ist zwar vorgesehen, aufgrund fehlender Übergaberichtlinien jedoch mit Informationsverlusten und einem erhöhten Aufwand verbunden. Dies kann dazu führen, dass sich aus dem Einsatz von BIM gegenüber konventionellen Planungsmethoden keine Vorteile ergeben [25].

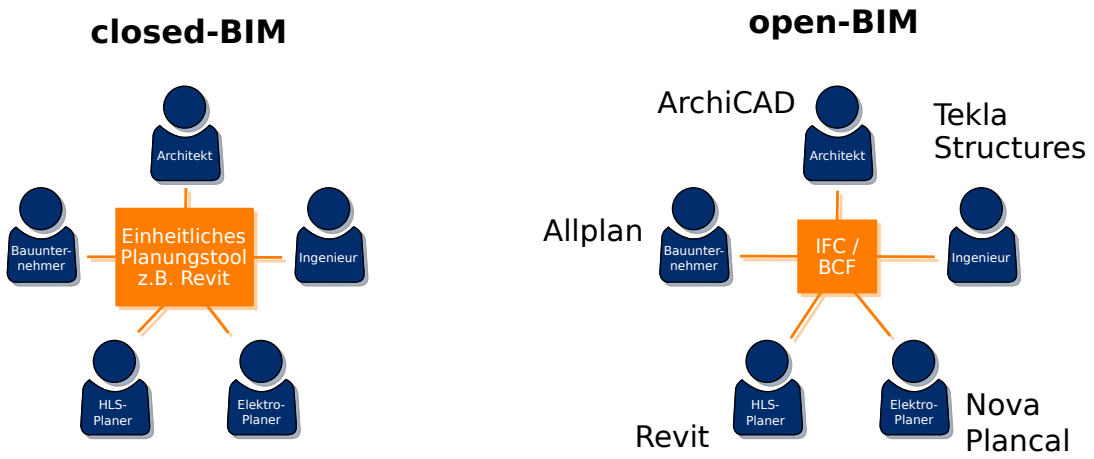



Abbildung 2.8: Closed-BIM vs. open-BIM, nach [22]

Im Gegenentwurf dazu steht open-BIM. Diese Methodik funktioniert software- und herstellerübergreifend und ermöglicht die Einbindung unterschiedlichster Planungstools und deren Informationsaustausch untereinander. Eine Umstellung auf spezielle Software, welche verschiedene Fachmodelle in einem Modell vereinigt, ist nicht notwendig, sodass alle Fachplanungen mit ihren gewohnten Programmen arbeiten können. Der Datenaustausch funktioniert über offene Standards, auf die alle Softwarehersteller Zugriff haben und somit die Schnittstellen zu anderen Anwendungen schaffen können. Hierin offenbart sich die größte Herausforderung des open-BIM, da das Schaffen einheitlicher Standards für Austauschformate eines enormen Entwicklungsaufwandes bedarf, der sich über einen langen Zeitraum erstreckt [25]. Wenngleich einige dieser Standards heute schon praxistauglich sind (z.B. IFC, siehe Abschnitt 2.2.3), besteht noch viel Forschungs- und Entwicklungsbedarf, um einen vollständig reibungslosen Datenaustausch zwischen allen Anwendungen zu erhalten.

### Kombinationen der BIM-Ansätze

Aus den Ansätzen von little-BIM, big-BIM, closed-BIM und open-BIM ergeben sich nun vier Kombinationsmöglichkeiten zur Verbindung der Parameter Durchgängigkeit und Offenheit: *Little-closed-BIM*, *little-open-BIM*, *big-closed-BIM* und *big-open-BIM*.

Das little-closed-BIM beschreibt den Arbeitsansatz, in dem ein Anwender lediglich mit seinem eigenen Fachmodell arbeitet und die Daten auch mit niemandem austauscht. So erstellt ein Architekt beispielsweise seinen digitalen Entwurf, teilt diese Daten aber nicht mit anderen Projektbeteiligten. Beim little-open-BIM erstellt ein Anwender zwar auch sein

eigenes Fachmodell, stellt dieses aber in einem offenen Austauschformat (z.B. IFC) anderen zur Verfügung [23].

Beim Ansatz des big-closed-BIM arbeiten verschiedene Planer (Architekt, Tragwerksplaner) mit der gleichen Software (z.B. Graphisoft Archicad), sodass der Datenaustausch im herstellereigenen, proprietären Format erfolgen kann. Die big-open-BIM-Arbeitsweise beschreibt die Zusammenarbeit verschiedener Planungsbeteiligten, die jeweils mit unterschiedlichen Softwarelösungen arbeiten und ihre Daten über offene Austauschformate miteinander teilen [23].

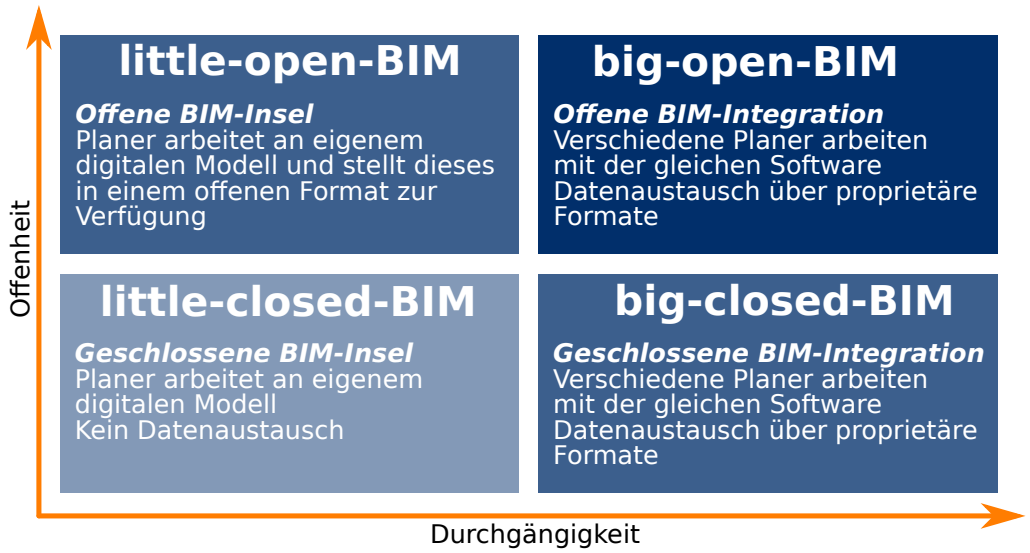



Abbildung 2.9: Kombinationen der BIM-Ansätze

Die Wahl des Arbeitseinsatzes hängt bei einem Projekt in erster Linie von der Größe des Projekts sowie den finanziellen und technischen Möglichkeiten der Projektbeteiligten ab. Mit den Vorteilen einer offenen, plattformunabhängigen Planung gehen auch Probleme einher, die einen reibungslosen Planungsablauf stören können. Daher ist die Fortentwicklung offener Schnittstellen im Building Information Modeling Gegenstand aktueller wissenschaftlicher Forschungen, wovon letztlich die Anwender profitieren. Aufgrund der prinzipiellen Vorteile gegenüber proprietären Lösungen ist davon auszugehen, dass die offenen BIM Arbeitsweisen in den kommenden Jahren an Bedeutung gewinnen.

### Bauwerksmodelle

Um die Basis für eine herstellerunabhängige, kollaborative Arbeitsweise zu schaffen, wird seit einigen Jahren die Entwicklung zweier unterschiedlicher Modellansätze zum Datenaustausch erforscht und weiterentwickelt. Das sind zum einen die *integrierten Bauwerksinformationsmodelle* und zum anderen die sogenannten *Multimodelle*.

Die grundsätzliche Herausforderung bei der Schaffung von Datenschnittstellen ist die Vereinbarkeit vieler Systeme. Kann zwischen zwei Anwendungen eine direkte Schnittstelle mit vergleichsweise wenig Aufwand erzeugt werden, so müssen bei vielen Anwendungen alle miteinander interagieren können, was einen quadratischen Anstieg der benötigten Schnittstellen in Abhängigkeit der Anzahl an Anwendungen bedeutet [27].

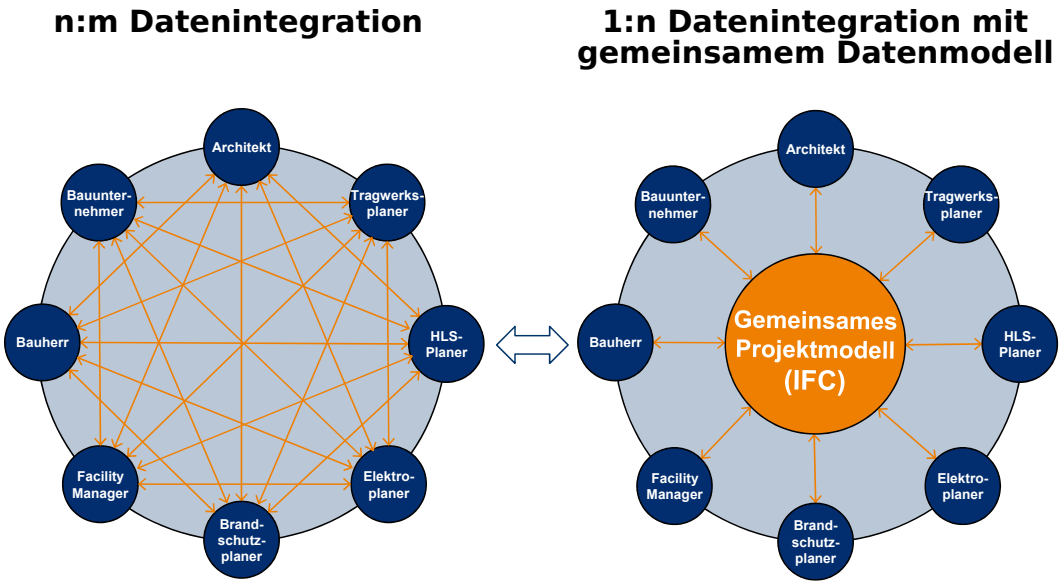



Abbildung 2.10: Datenintegration verschiedener Modellansätze, nach [28]

Aus diesem Grund verfolgt der Ansatz über *Bauwerksinformationsmodelle* das Ziel, möglichst viele Fachanwendungen über eine Schnittstelle, das heißt über ein einziges Austauschformat, abzubilden. Was zunächst nach einer proprietären Lösung klingt, ist jedoch eine Bestrebung nach einem umfassenden offenen Format, auf das die Softwareanwendungen aller Fachdisziplinen Zugriff haben und ihre fachspezifischen Daten dort einpflegen können. Hierfür hat sich der Datenstandard IFC (siehe dazu auch den folgenden Abschnitt 2.2.3) etabliert, der bereits das Datenabbild vieler Fachanwendungen unterstützt. Allerdings befinden sich manche dieser Anwendungen noch in der Entwicklungsphase, weitere sind bisher noch gar nicht implementiert. Die Erweiterung des IFC-Schemas ist jedoch nur langsam möglich, da bei der Berücksichtigung aller fachmodellspezifischer Besonderheiten das IFC-Schema äußerst komplex wird [27].

### 2.2.3 Industry Foundation Classes (IFC)

Hinter der Bezeichnung IFC steht ein internationales, offenes Format für die digitale Modellierung von Gebäuden und dem Datenaustausch im Bauwesen. Kernstück ist ein Bauwerksmodell, das über die Geometrie hinaus semantische Informationen über das Bauwerk bzw. die enthaltenen Objekte enthält und weitere Fachdisziplinen wie Tragwerks- und HLS-Planung, aber auch baubetriebliche Disziplinen wie Ablaufplanung und Kostenermittlung unterstützt. Die Entwicklung von IFC wird federführend von der Organisation *buildingSMART* vorangebracht, welche durch die Entwicklung und Einführung offener Standards die Veränderung der Bauwirtschaft vorantreibt [9].

IFC stellt sowohl ein *Informationsmodell* als auch ein *Dateiformat* dar. Als Informationsmodell wird ein Schema in EXPRESS- oder XSD[3]-Notation definiert, um eine Struktur für geometrische und semantische Informationen vorzugeben [8]. Dabei sind die darstellbaren Informationen für die einzelnen Modellelemente und die Relationen zwischen diesen durch das Schema festgelegt. Dadurch können Abhängigkeiten, Topologien und Vererbungen beschrieben werden, sodass verschiedene Anwendungen anhand dieser Datenstruktur die Informationen austauschen können.

[3] XML Schema Definition

Der Datenaustausch erfolgt über das IFC-Dateiformat, was eine ASCII-Textdatei darstellt. IFC-Dateien nutzen das STEP-physical-file Format nach ISO 10303-21 zur Darstellung der Inhalte. Dies schafft die offene, plattformunabhängige Basis zum kollaborativen Arbeiten verschiedener Anwender.

Die aktuellste Version des Standards ist IFC4, die gleichzeitig als ISO 16739 Norm veröffentlicht wird. Es ist eine Weiterentwicklung des IFC2x3 Standards, mit dem bereits eine umfassende Bauwerksmodellierung möglich war und für wissenschaftliche Forschungszwecke einen ausreichenden Informationsraum bereitstellte. Durch IFC4 sind die Möglichkeiten nochmals erweitert worden, sodass beispielsweise deutlich mehr Objekteigenschaften (sogenannte IFC-property-Sets) vergeben werden können oder eine verbesserte Unterstützung für parametrische Geometriedarstellung gegeben ist [1].

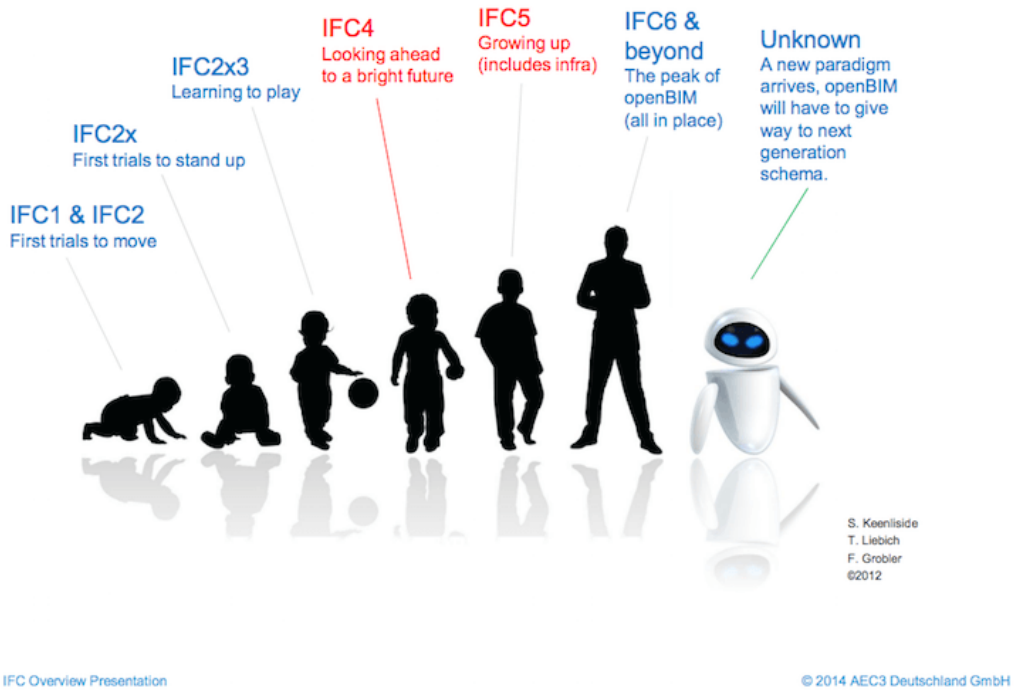



Abbildung 2.11: Frühere und zukünftige IFC-Entwicklung [1]

In obiger Abbildung ist zu erkennen, in welchem Stadium sich die IFC-Entwicklung derzeit befindet. Dabei wird deutlich, dass für einen umfassenden praxistauglichen Einsatz noch viel Entwicklungsarbeit nötig ist und es erst ab der Version IFC6 zu erwarten ist, dass Bauwerksmodelle umfassend in einer open-BIM Umgebung abgebildet werden können. Trotzdem wird IFC bereits in Planungs- und Ausführungsprozessen eingesetzt, sodass eine Evaluation dieses Datenschemas durch die Praxis laufend gegeben ist. Am weitesten verbreitet ist derzeit die Version IFC2x3 [8], weswegen der entwickelte Softwareprototyp anhand Daten dieser Version validiert wird. Darüber hinaus entsprechen die betrachten Klassenstrukturen und erforderlichen Attribute denen von IFC4, sodass hieraus kein Nachteil hinsichtlich des Informationsgehalts entsteht.

## 2.3 Berührungspunkte zwischen BIM und der Rettungswegermittlung

Die konventionelle Brandschutzplanung erfolgt durch geometrische Informationen, in der Regel in graphischer Form anhand zweidimensionaler Grundrisspläne. Neben der verbalen Beschreitung der Maßnahmen wird das Brandschutzkonzept durch Brandschutzpläne ergänzt, die neben dem Rettungswegverlauf auch Anforderungen an die Bauteile visualisieren. Diese Pläne werden auf Basis von übermittelten CAD-Daten erstellt und in ihre fachspezifische Form (quasi ein eigenes Fachmodell) überführt.

Eine automatisierte Rettungswegermittlung erfolgt auf diese Weise nicht. Es wäre möglich, die Geometriedaten aus den proprietären CAD-Dateien zu erhalten, wobei bestimmte semantische Informationen immer noch „manuell“ durch die Planer zu berücksichtigen sind. Für einen Algorithmus sind die einzelnen Linien einer CAD-Zeichnung zunächst nur Striche ohne weitere Bedeutungen; er weiß also nicht, ob es sich um eine Wand, eine Tür oder ein anderes Bauelement handelt. Um eine vollautomatische Rettungswegermittlung durchzuführen, sind semantische Informationen aber zwingend notwendig, damit durch den Algorithmus erkannt wird, welche Bereiche als Rettungsweg dienen können oder wo sich eine Tür zu einem notwendigen Treppenraum und damit dem „Zielpunkt“ eines Rettungsweges befindet.

Das IFC-Schema bietet die grundsätzliche Möglichkeit, semantische Informationen zum Bauwerk zu hinterlegen. Diese können beim Bearbeiten des Modells mit der Geometrie in Verbindung gebracht werden, was eine automatische Rettungswegermittlung grundsätzlich ermöglichen würde. Die Untersuchung der Machbarkeit und des Umfangs ist Gegenstand dieser Arbeit.

# 3 Graphentheorie

## 3.1 Allgemeines

Die Graphentheorie, grundsätzlich ein Teilgebiet der Mathematik, behandelt die Eigenschaften von Graphen und deren Beziehung zueinander. Dabei können die Problemstellungen, welche mittels Graphen gelöst werden sollen, aus ganz unterschiedlichen Anwendungsgebieten stammen. Viele Prozessabläufe des Alltags lassen sich durch Graphen modellieren und mittels Algorithmen optimieren bzw. umsetzen. Aus diesem Grunde ist die Graphentheorie auch eng mit der Informatik verbunden.

Historisch zurückzuführen sind die Anfänge der Graphentheorie auf Leonhard Euler (1707-1783), der zu Lebzeiten die Abstraktion von Sachverhalten in *Kanten* und *Knoten* entwickelte. Anlass dieser Forschung war die Lösung des bekannten „Königsberger Brückenproblems“, mit welchem Euler 1736 betraut worden war. Die Fragestellung war, ob es möglich sei, bei einem Stadtrundgang alle Brücken genau ein einziges Mal zu passieren und am Ende wieder zum Ausgangspunkt zu gelangen (siehe Abbildung 3.1).

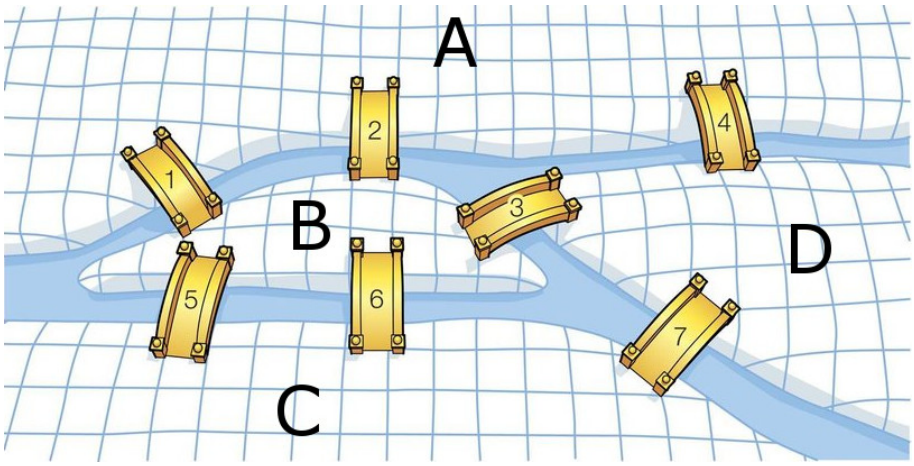



Abbildung 3.1: Königsberger Brücken im Stadtplan [30], bearbeitet

Prinzipiell wäre es möglich gewesen, durch Ermittlung sämtlicher möglichen Wege die Frage zu beantworten. Dies hätte jedoch, abgesehen von einem hohen Arbeitsaufwand, nur eine Lösung für genau dieses Problem ergeben. Eulers Anspruch war jedoch eine Methodik zu finden, mit welcher sich das Problem schneller löste und die obendrein noch allgemeingültig und damit auf andere Probleme anwendbar war.

Als ersten Ansatz modellierte Euler die Situation in einem Graphen. Dabei stellte er die Landschaftsstücke (A – D) als *Knoten* und die verbindenden Brücken (1 – 7) als *Kanten* dar (siehe Abbildung 3.2). Es ist zu erkennen, dass die geografischen Gegebenheiten in dem Modell nicht berücksichtigt sind. Das Modell ist auf das Wesentliche reduziert, nämlich die Darstellung der einzelnen Wege-Beziehungen untereinander. Auf dieser Basis stelle Euler eine These auf, die später von Carl Hierholzer im Jahre 1873 bewiesen wurde [15]: „Ein zusammenhängender Graph ist genau dann eulersch, wenn jede seiner Ecken geraden Grad hat“ [14]. Dies bedeutet, dass jeder Knoten eines Graphen eine gerade Anzahl an

angeschlossenen Kanten haben muss, damit jede Kante nur einmal „abgelaufen“ werden kann, um bei einem Durchlauf wieder am Ausgangspunkt anzukommen. Ein solcher Graph wird als *Eulerkreis* bezeichnet.

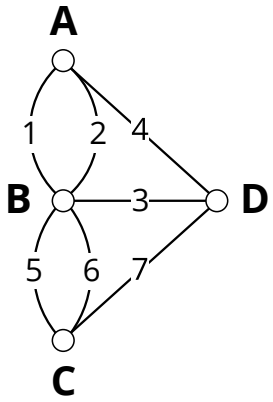



Abbildung 3.2: Königsberger Brücken als Graph

Mit dieser Erkenntnis konnte Euler die Antwort auf das „Königsberger Brückenproblem“ geben. Da $4 > 0$ der vorhandenen Knoten eine ungerade Anzahl an angeschlossenen Kanten haben, kann der Stadtrundgang nicht so erfolgen, dass jede der Brücken nur einmal passiert wird und man wieder zum Ausgangspunkt gelangt. Bei dem Königsberger Brückenproblem handelt es sich also nicht um einen Eulerkreis.

Es ist ebenfalls nicht möglich, jede der Brücken genau einmal zu passieren, wenn sich Start- und Zielpunkt unterscheiden. Auch hierfür ließ sich eine Gesetzmäßigkeit aufstellen, die als *Eulerweg* bezeichnet wird: „In einem zusammenhängenden Graphen existiert genau dann ein Eulerweg, wenn zwei Knoten ungeraden Grad haben“ [32]. Das bedeutet, dass ein Graph zwei Knoten mit einer ungeraden Anzahl an angeschlossenen Kanten haben muss (die übrigen Knoten müssen eine gerade Anzahl haben), damit alle Kanten nur ein einziges Mal durchlaufen werden. Dabei ist der Endknoten ungleich dem Ausgangsknoten. Ein bekanntes Beispiel hierfür ist das Zeichenspiel „das Haus vom Nikolaus“ (siehe Abbildung 3.3). Dort sind zwei Knoten mit jeweils 3 angeschlossenen Kanten vorhanden, die übrigen Knoten haben eine gerade Anzahl an angeschlossenen Kanten.

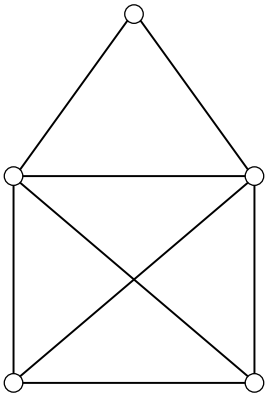


Abbildung 3.3: „Das Haus vom Nikolaus“

Wie sicherlich weitläufig bekannt ist, kann das „Haus vom Nikolaus“ ohne abzusetzen und ohne eine Kante doppelt zu zeichnen, in einem Zuge gezeichnet werden. Es ist jedoch un-

möglich, den Ausgangspunkt zu erreichen, ohne eine Kante ein weiteres Mal überzeichnen zu müssen. Dementsprechend handelt sich bei dem „Haus vom Nikolaus“ also um einen *Eulerweg*.

Seit Eulers Anfängen entwickelte sich in den letzten 200 Jahren die umfangreiche Disziplin der Graphentheorie, die heute Lösungen für verschiedenste Problemstellungen bietet. Ihnen allen gemein ist die Abstraktion des Anwendungsfalls auf einen Graphen, der aus Kanten und Knoten gebildet wird. In den nächsten Abschnitten werden die Grundlagen der Graphentheorie erläutert (Abschnitt 3.2). Bei der Vielzahl von Anwendungsgebieten wird sich auf das der kürzesten Pfade beschränkt, da andere Bereiche im Rahmen dieser Arbeit nur eine untergeordnete Bedeutung haben. Wegverbindungen sowie die Algorithmen zur Findung der kürzesten Strecke haben als zentrale Elemente für diese Arbeit eine weiterführende Relevanz.

## 3.2 Grundlagen der Graphentheorie

### 3.2.1 Elemente und grafische Darstellung

Auch 200 Jahre nach Euler bilden dessen Abstraktionen immer noch die Grundbausteine der Graphentheorie. Die zentrale Struktur ist dabei der *Graph*. Ein Graph repräsentiert eine Menge von Objekten und Verbindungen dieser Objekte untereinander. Die Objekte werden als *Knoten* (oder auch *Ecken*) bezeichnet, die Verbindungen zwischen ihnen als *Kanten* [14]. Wenn ein Knoten der Anfangs- oder Endpunkt einer Kante ist, so *indizieren* diese Elemente miteinander bzw. werden sie *indizent* genannt. Die beiden Knoten, die durch eine gemeinsame Kante miteinander verbunden sind, heißen *adjazent* bzw. *benachbart* [14].

Zunächst werden Graphen in zwei Arten unterschieden, und zwar in die

- *gerichteten Graphen* und die
- *ungerichteten Graphen*.

Bei gerichteten Graphen ist die Richtung, in welche die Kanten „abgelaufen“ werden dürfen, vorgegeben. Dies ist beispielsweise der Fall, wenn bei Routenplanungen Einbahnstraßen berücksichtigt werden müssen oder irreversible Zustandsänderungen zu modellieren sind [29]. Grafisch dargestellt wird die Richtung durch Pfeile (siehe Abbildung 3.4).

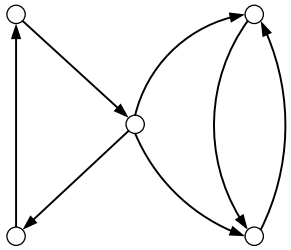


Abbildung 3.4: Gerichteter Graph

Die Kanten ungerichteter Graphen haben keine Orientierung, das heißt, sie dürfen in beiden Richtungen „abgelaufen“ werden. Beim Königsberger Brückenproblem wurden ungerichtete Graphen verwendet, da die Richtung des Stadtrundgangs unerheblich war, ebenso wie beim „Haus vom Nikolaus“. Die graphische Darstellung einer ungerichteten Kante erfolgt durch eine einfach Linie (siehe Abbildungen 3.2 und 3.3).

Sind in Graphen zwei Knoten durch mehrere Kanten miteinander verbunden (wie z.B. in Abbildung 3.4 die beiden Knoten auf der rechten Seite), so werden diese Kanten *Mehrfachkanten* genannt. Ein Graph, der Mehrfachkanten beinhaltet, heißt *Multigraph*. Sofern diese Situation nicht vorliegt, werden solche Graphen als *schlicht* bezeichnet [31].

Die grafische Anordnung der Elemente, also die Verteilung der *Knoten* und *Kanten* ist frei wählbar, d.h., dass derselbe Graph durch unterschiedliche Diagramme visualisiert werden kann (siehe Abbildung 3.5). Da es bei der weiteren Verwendung des Graphen lediglich auf die Beziehungen der Knoten und Kanten zueinander ankommt, ist die Positionierung der Elemente in einer Grafik unerheblich. Allerdings sollte nach Möglichkeit eine Darstellung gewählt werden, welche die gegebene Problemstellung möglichst anschaulich visualisiert.

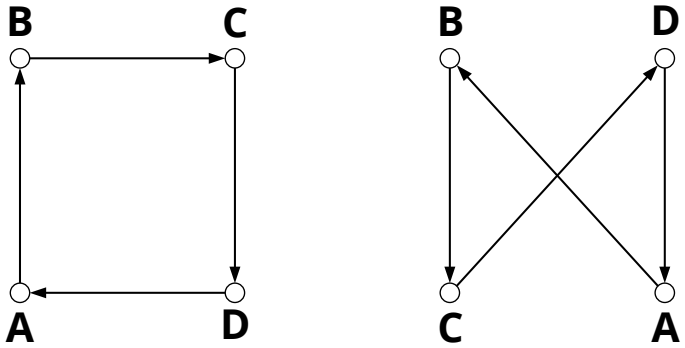



Abbildung 3.5: Unterschiedliche Diagramme zeigen denselben Graphen

Wie in den Abbildungen 3.2 und 3.4 ebenfalls zu erkennen ist, müssen die Kanten nicht zwangsweise gerade verlaufen, sondern können auch gekrümmt oder anderweitig geformt sein. Es empfiehlt sich jedoch der Übersicht halber, Kanten nach Möglichkeit gerade zu zeichnen und nur gekrümmt darzustellen, wenn Schnittpunkte mit anderen Kanten oder Knoten dadurch vermieden werden können [29].

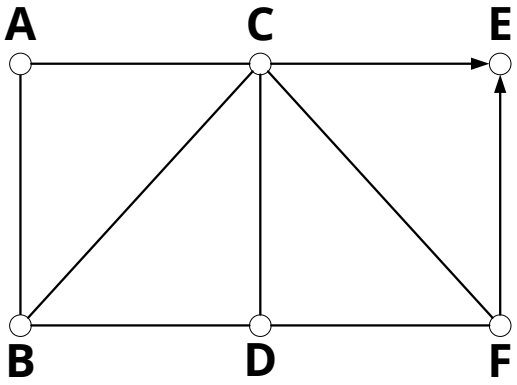
### 3.2.2 Mathematische Definition

Mathematisch ausgedrückt ist ein Graph „ein Paar $G = (V,E)$ disjunkter Mengen mit $E \subseteq [V]^2$; die Elemente von $E$ sind also 2-elementige Teilmengen von $V$. Die Elemente von $V$ nennt man die *Ecken* (oder *Knoten* des Graphen) $G$, die Elemente von $E$ seine *Kanten*“ [14]. In Bezug auf das Beispiel des „Königsberger Brückenproblems“ in Abbildung 3.1 wird der Graph $G = (V,E)$ mit $V = \{A,B,C,D\}$ und $E = \{\{A,B\},\{A,B\},\{A,D\},\{B,D\},\{B,C\},\{B,C\},\{C,D\}\}$ ausgedrückt. Der gerichtete Graph $G = (V,E)$ aus Abbildung 3.5 hat eine Knotenmenge $V = \{A,B,C,D\}$ und eine Kantenmenge $E = \{(AB),(BC),(CD),(DA)\}$. Die runden Klammern um die Kantenmengen zeigen, dass die Orientierungen der Kanten berücksichtigt werden müssen, während geschweifte Klammern um Kantenmengen die Unabhängigkeit der Kantenorientierung bedeuten.

Dass Graphen mathematisch beschrieben werden können bietet die Möglichkeit, mittels effizienter Algorithmen Lösungen für individuelle Problemstellungen zu finden. Dazu ist es nötig, die Beziehungen der Kanten und Knoten eines Graphen derartig aufzubereiten, dass sie maschinenlesbar werden. Hierfür gibt es verschiedene Datenstrukturen. Welche Datenstruktur notwendig oder günstig ist, hängt im Wesentlichen von der weiteren Verarbeitung bzw. den durchzuführenden Operationen ab [31]. Doch auch der Graph selbst kann gewisse

Merkmale mit sich bringen, welche die Anwendung bestimmter Datenstrukturen erfordern oder ausschließen.

Eine mögliche (mathematische) Darstellung eines Graphen gelingt mit der sogenannten *Adjazenzmatrix*. In einer solchen kann die direkte Nachbarschaft der Knoten zueinander abgebildet werden. Dabei wird der Graph $G$ in einer Matrix $A(G)$ mit der Dimension $n \times n$ dargestellt, wobei $n$ die Anzahl der im Graphen vorkommenden Knoten ist. Ein Eintrag $a_{ij}$ in der Matrix $A(G)$ ist immer dann gleich 1, wenn von einem Knoten $i$ zu einem Knoten $j$ eine Kante existiert. Falls dies nicht zutrifft, ist der Eintrag $a_{ij}$ gleich 0. Bei einer derartigen Matrix spricht man auch von einer *Booleschen Matrix*, da die Einträge entweder 0 oder 1 sein können [31]. Zunächst sind in einer solchen Adjazenzmatrix nur schlichte Graphen darstellbar, da eine Berücksichtigung von Mehrfachkanten nicht möglich ist.




$$A(G) = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

Abbildung 3.6: Graph mit zugehöriger Adjazenzmatrix

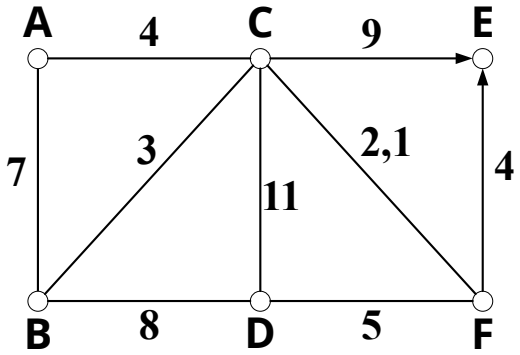
Abbildung 3.6 zeigt einen Graphen, der aus gerichteten und ungerichteten Kanten besteht, sowie die zugehörige Adjazenzmatrix $A(G)$. Die Beschriftung dieser Matrix außerhalb der Klammern dient hier nur der Übersicht. In den Zeilen werden die Start-Knoten angegeben, in den Spalten die jeweiligen Ziel-Knoten.In diesem Beispiel ist gut zu erkennen, dass es zwei gerichtete Verbindungen zum Knoten $E$ gibt, vom Knoten $E$ jedoch keine Verbindung zu anderen Knoten existiert, da die Einträge in der Zeile für Knoten $E$ überall 0 sind. Existieren in einem Graphen nur ungerichtete Verbindungen, so ist die Matrix symmetrisch [31].

Ein großer Nachteil der Abbildung eines Graphen in einer *booleschen Matrix* ist zum einen, dass keine Mehrfachkanten berücksichtigt werden können. Zum anderen gehen aus dieser Matrix keine weiteren Informationen über die Graphen hervor, außer die der Existenz von Kanten und deren Orientierung zwischen den verschiedenen Knoten. Für eine praxisnahe Anwendung ist es daher zunächst erforderlich, den Graphen zu modifizieren. Bei Wegeverbindungen beispielsweise, wie sie auch in dieser Arbeit behandelt werden, ist nicht nur die Existenz einer Verbindung von zwei Knoten von Bedeutung, sondern auch die Entfernung zwischen diesen. Eine Möglichkeit, dies in einem Graphen abzubilden ist, den Kanten Werte zuzuweisen. Ein derartiger Graph wird als *kantenbewertet* bezeichnet. Die Werte entsprechen dem, was abgebildet werden soll, beispielsweise einer Wegstrecke.

Als Beispiel dazu kann das Straßennetz eines Landes betrachtet werden. Einzelne Städte bilden die Knoten, die sie verbindenden Straßen die Kanten eines Graphen. Navigationssysteme und Routenplaner arbeiten mit genau diesen Graphen, um nun die kürzeste oder schnellste Route zwischen zwei Orten zu berechnen. Die Kanten sind dabei entsprechend

der Länge oder auch der Art der jeweiligen Straße gewichtet, sodass Algorithmen mit diesen Werten arbeiten können.

Der Graph $B(G)$ in Abbildung 3.7 weist die gleichen Verbindungen auf wie der Graph in Abbildung 3.6, hat jedoch zusätzlich bewertete Kanten. Bei einer Wegstrecke entsprächen die Werte der Entfernung zwischen den jeweiligen Knoten. Gegenüber der Booleschen Matrix wird in der zugehörigen Adjazenzmatrix nicht nur die Existenz einer Verbindung angegeben, sondern auch die Kantenbewertung der jeweiligen Verbindungen. Besteht keine direkte Kantenverbindung zwischen zwei Knoten, so bleibt der Eintrag 0.




$$B(G) = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} 0 & 7 & 4 & 0 & 0 & 0 \\ 7 & 0 & 3 & 8 & 0 & 0 \\ 4 & 3 & 0 & 11 & 9 & 2{,}1 \\ 0 & 8 & 11 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2{,}1 & 5 & 4 & 0 \end{pmatrix} \end{array}$$

Abbildung 3.7: Kantenbewerteter Graph mit zugehöriger Adjazenzmatrix

Je nach Größe des Graphen und dem Umfang der Weiterverarbeitung spielt auch die Komplexität und der Speicherbedarf der Abbildung eine große Rolle. Eine Adjazenzmatrix benötigt immer $n^2$ Speicherplätze, unabhängig davon, wie viele Kanten in dem Graphen existieren [31]. Mit *Adjazenzlisten* gibt es eine weitere Möglichkeit, Graphen abzubilden. Ein Vorteil der Adjazenzliste gegenüber einer Adjazenzmatrix ist, dass die Adjazenzliste unter bestimmten Bedingungen weniger Speicherplatz benötigt. In einer Adjazenzmatrix wird für jede potentielle Verbindung zwischen zwei Knoten ein Eintrag reserviert. Existiert keine Kante zwischen diesen Knoten, so ist der Eintrag 0. In einer Adjazenzliste werden alle Knoten aufgelistet und es wird für jeden angegeben, zu welchen anderen Knoten eine Verbindung besteht. Dies kann optional durch die Kantengewichtung erweitert werden.

Da mit Adjazenzmatrizen Zusammenhänge in einem Graphen anschaulicher vermittelt werden können, erfolgt die Beschreibung von Graphen zur Erläuterung der Methoden im folgenden Kapitel 4 mit Adjazenzmatrizen. In dem Softwareprototypen werden jedoch in weiterem Sinne Adjazenzlisten implementiert, da diese innerhalb der objektorientierten Struktur des Softwareprototypen besser handhabbar sind und die weiterführenden Verarbeitungen erleichtern (siehe Abschnitt 6.4.3).

# 4 Rettungswegermittlung auf Basis graphentheoretischer Ansätze

## 4.1 Graphentheorie in der Rettungswegermittlung

Wie in der Einleitung des vorhergehenden Kapitels bereits gesagt, können viele verschiedene, auf den ersten Blick auch nicht zusammenhängende Problemstellungen mittels Graphentheorie gelöst werden. In den meisten Fällen geht es um zeitliche, räumliche oder materielle Optimierungen sowie das strukturierte Verarbeiten von Zusammenhängen.

Auch in Bezug auf Rettungswegermittlungen kann die Graphentheorie genutzt werden, um die Kapazitäten und Wegstrecken zu berechnen. Ein Rettungsweg ist immer automatisch eine Wegstrecke, die abgelaufen werden muss, damit sich Personen in Sicherheit bringen können und gleichzeitig der Feuerwehr als Angriffsweg dient. Es ist offenkundig, dass die Länge eines Rettungsweges erheblichen Einfluss auf die Sicherheit hat, was nicht zuletzt in den Bauordnungen der einzelnen Bundesländer durch eine Begrenzung der maximalen Rettungsweglänge berücksichtigt wird. Insofern ist es naheliegend, möglichst kurze Wegstrecken als Rettungswege zu finden.

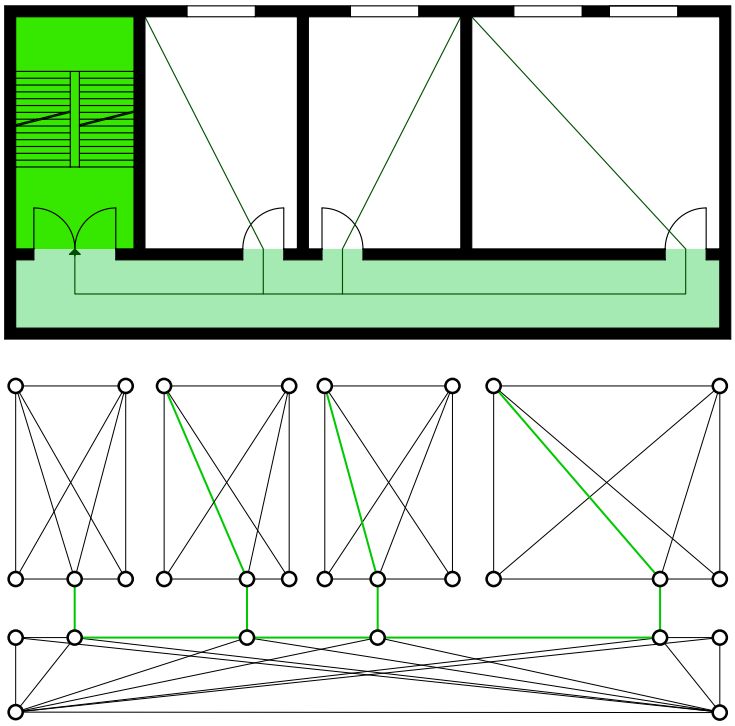


Abbildung 4.1: Rettungswege in einem Geschoss mit möglichem Graphen

Diese Arbeit behandelt die Wegstreckenbetrachtung innerhalb eines Gebäudes durch Abstraktion des Gebäudegrundrisses in einen Graphen. Als Knoten werden dann bestimmte Punkte in Räumen, Türen, Fluren und Treppenräumen definiert, die vorhandenen Weg-

strecken zwischen ihnen sind die Kanten. Diese Kanten werden mit der entsprechenden Distanz zwischen zwei verbundenen Knoten gewichtet. Mit graphentheoretischen Algorithmen lassen sich dann die kürzesten Pfade innerhalb des Graphen ermitteln, welche dann die günstigsten Rettungswege in dem Gebäude repräsentieren.

Ferner kann es auch zielführend sein, Kanten in Abhängigkeit der Fluchtzeit zu gewichten, da die Evakuierungsdauer nicht nur von der Wegstrecke abhängig ist, sondern noch weitere Parameter wie die Qualität des Rettungsweges, die Personenzahl und die Panik der Menschen beinhalten kann. Das Erforschen zeitabhängiger Evakuierungen ist jedoch Teilgebiet der Evakuierungssimulationen, welche in dieser Arbeit nicht behandelt wird. An dieser Stelle erfolgt lediglich der Verweis auf die Abschnitte 2.1.4 und 7.2, in denen die Einsatzmöglichkeiten der Evakuierungsberechnungen diskutiert werden.

## 4.2 Algorithmen zur Findung der kürzesten Wegstrecke

Die triviale Lösung zur Findung einer kürzesten Wegstrecke zwischen zwei Orten ist, alle möglichen Wegstrecken zu ermitteln und dann die kürzeste davon auszuwählen. Bei sehr kleinen Strukturen mag dies noch eine möglich Art der Ermittlung sein, aber bei komplexeren Strukturen führt dies zu einem exorbitanten Rechenaufwand, da eine Reihe von Strecken, welche offensichtlich nicht die kürzesten sind, trotzdem mit untersucht werden. Um dieser Problematik entgegenzuwirken, gibt es Algorithmen, die effiziente Suchstrategien anwenden, um einen kürzesten Pfad zu finden. Diese Algorithmen nehmen in den meisten Fällen bei der Suche eines kürzesten Pfades an, dass dieser jeweils aus kürzesten Teilpfaden besteht. Für den mathematischen Beweis dieser Annahme wird auf die Literatur verwiesen [12]. Nachfolgend werden einige bekannte kürzeste-Pfad-Algorithmen vorgestellt.

### 4.2.1 Dijkstra-Algorithmus

Ein bewährter Algorithmus zur Lösung des kürzesten-Pfad-Problems ist der sogenannte *Dijkstra-Algorithmus*. Dieser berechnet für einen gegebenen Startknoten $s$ den kürzesten Pfad zu allen nachfolgenden Knoten in einem kantenbewerteten Graph. Die Bewertung der Kanten muss dabei stets positiv sein. Der Prinzipielle Ablauf wird nachfolgend anhand eines Minimalbeispiels erläutert.

V={S,A,B,C,D,E}
W={}
U={S,A,B,C,D,E}

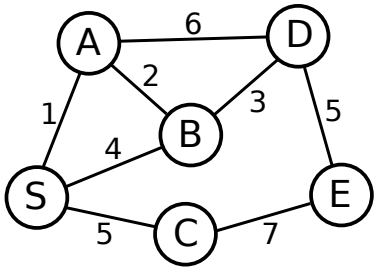



Abbildung 4.2: Beispielgraph

Abbildung 4.2 zeigt einen ungerichteten, kantenbewerteten Graphen mit sechs Knoten und acht Kanten. Es soll der kürzeste Pfad vom Knoten $S$ zu allen anderen Knoten bestimmt werden. Die Angabe eines Zielknotens ist beim Dijkstra-Algorithmus nicht erforderlich. Sie kann aber dennoch getätigt werden, um die Untersuchung abzubrechen, sobald dieser erreicht worden ist. Die Notation in diesem Beispiel bezieht sich auf die in Abbildung 4.3.

Mit der Variablen $s$ wird immer der Startknoten bezeichnet, mit der Variablen $i$ immer ein gerade besuchter und betrachteter Knoten. Die Variable $j$ kann mehrfach vorkommen und bezieht sich auf alle benachbarten Knoten des Knotens $i$.

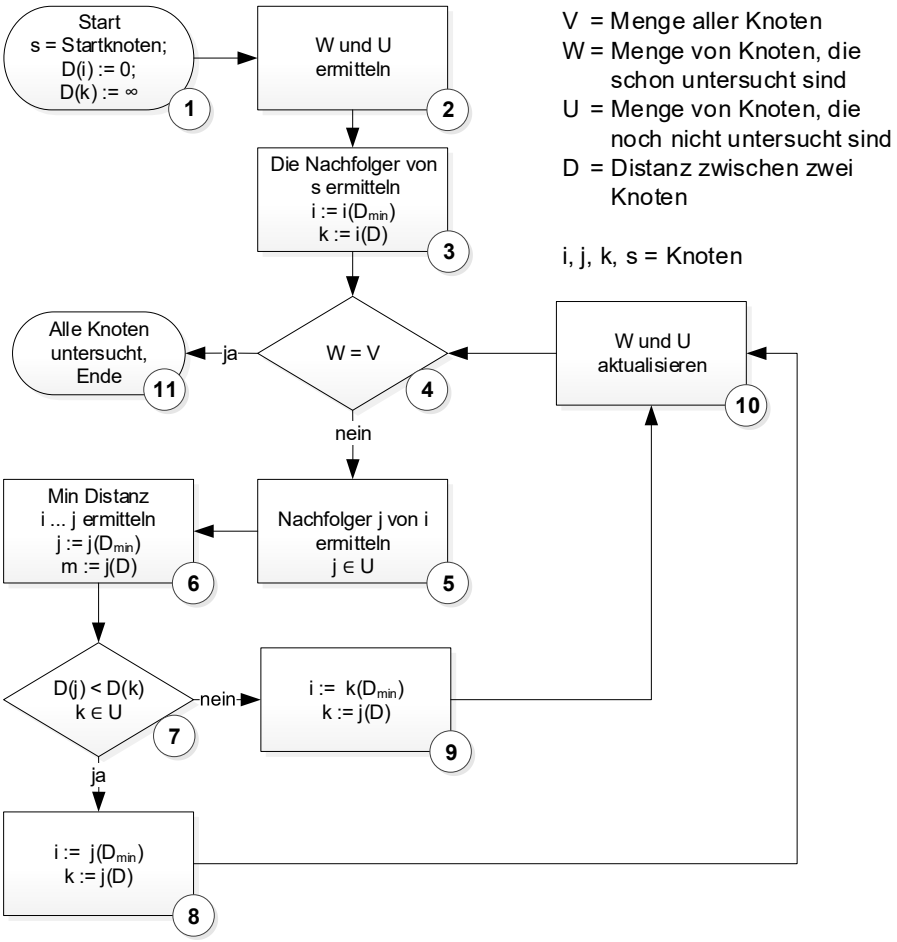



Abbildung 4.3: Schematischer Ablauf des Dijkstra-Algorithmus

Im Beispiel werden die verschiedenen Knoten $j1$, $j2$ usw. bezeichnet. Die Variable $k$ bezieht sich auf alle Knoten, die bereits bekannt (aber noch nicht besucht) sind und keine direkten Nachfolger des Knotens $i$ sind. Auch diese werden als $k1$, $k2$ usw. bezeichnet. Im Beispiel wird die Gesamtdistanz zum Startknoten direkt an den entsprechenden Knoten (z.B. $j1 = 1$) angetragen. Insgesamt ist diese Notation mathematisch nicht ganz korrekt, da sich eine Variable nur auf einen Wert beziehen kann, aus Gründen der Übersichtlichkeit und Verständlichkeit ist diese Abweichung hier jedoch bewusst gewählt worden.

Zu Beginn werden alle möglichen Verbindungen vom Startknoten zu den benachbarten Knoten ($A = j1$, $B = j2$, $C = j3$) ermittelt und der Knoten mit der kürzesten Strecke ausgewählt (A mit $j1 = 1$). Ein kürzerer Pfad von diesem zum Startknoten kann in dem Graphen nicht mehr vorkommen, sodass der Knoten A als „besucht“ gilt. Ein einmal besuchter Knoten wird bei dem Dijkstra-Algorithmus grundsätzlich nie erneut besucht, da zu diesem der kürzeste Pfad bereits gefunden wurde. Nun ist A der aktuell betrachtete Knoten ($i$) und es werden von diesem aus die Verbindungen zu den benachbarten Knoten B und C ($j$) geprüft. Parallel dazu sind B und C bekannte Knoten ($k$), die mit dem aktuell

betrachteten Knoten $i$ nicht in Verbindung stehen. Somit sind für B bereits zwei Pfade bekannt, wobei letztlich der kürzere relevant ist.

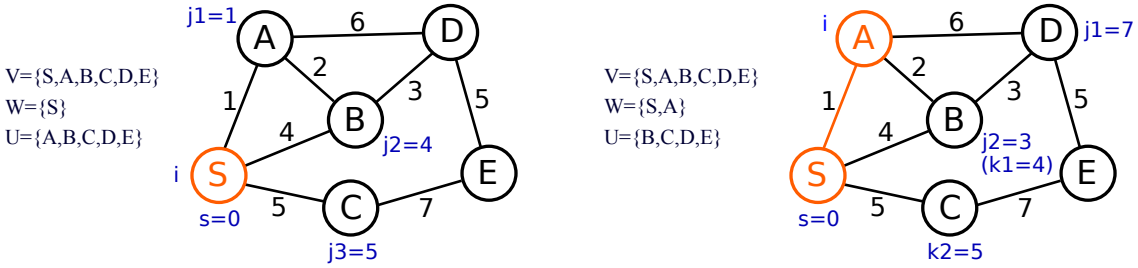



Abbildung 4.4: Beispielgraph: Ausgangszustand und erster besuchter Knoten

Da die Verbindung von Knoten A zu Knoten B an dieser Stelle die kürzeste ist ($j2 = 3$), wird nun auch Knoten B der Menge der bekannten Knoten hinzugefügt und die Ermittlung von diesem aus fortgesetzt. Hier gibt es nur einen noch nicht besuchten Nachbarknoten (D), mit der Distanz $j1 = 6$ zum Startknoten. Es existiert mit dem Knoten C allerdings noch ein weiterer, bekannter Knoten im Graphen mit einer gegenwärtig kürzeren Distanz zum Startknoten ($k1 = 5$). Dementsprechend wird nun der Knoten C als besucht markiert.

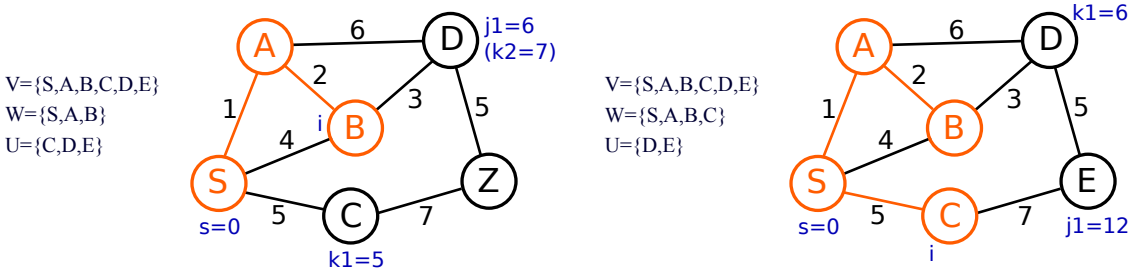



Abbildung 4.5: Beispielgraph: Drei bzw. vier besuchte Knoten

Von diesem existiert auch nur eine weitere benachbarte Verbindung zu dem unbesuchten Knoten E ($j1 = 12$), wobei nun die Verbindung zwischen den Knoten B und D ($k1 = 6$) den insgesamt kürzesten Pfad aufweist und daher der Knoten D an dieser Stelle als besucht gilt. Nun ist E der einzig unbesuchte Knoten im Graphen; der vom aktuell betrachteten Knoten D ein direkter Nachbar ist ($j1 = 11$). Bekannt ist außerdem die Verbindung zu E über den Knoten C ($k1 = 12$).

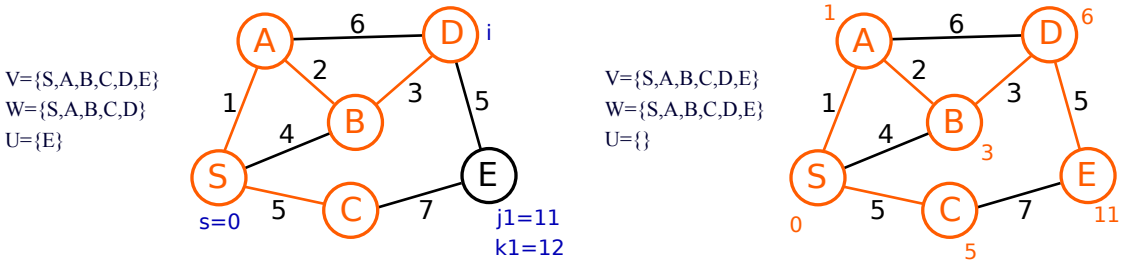



Abbildung 4.6: Beispielgraph: Fünf besuchte Knoten und Endzustand

Weil die Verbindung über Knoten D zu E den kürzeren Pfad darstellt, erhält der Knoten E den Wert aus $j1$ und wird mit in die Menge der besuchten Knoten aufgenommen. Da nun alle Knoten besucht und damit jeweils der kürzeste Pfad von jedem einzelnen zum

Startknoten ermittelt worden ist, ist die Untersuchung an dieser Stelle abgeschlossen. Abbildung 4.6 zeigt auf der rechten Seite den untersuchten Graphen mit allen Distanzen der einzelnen Knoten zum Startknoten S (orangene Zahlen) sowie die dabei abgelaufenen Pfade (orangene Kanten).

Diese Vorgehensweise ist unabhängig von der Größe und Komplexität eines Graphen und lässt sich auf beliebige Graphen mit positiven Kantengewichten anwenden. Der Dijkstra-Algorithmus findet dabei stets die optimale Lösung, d.h. der gefundene kürzeste Pfad ist auch tatsächlich der kürzeste existierende Pfad in einem Graphen [12]. Damit erweist er sich als geeigneter Algorithmus für die Rettungswegermittlung.

### 4.2.2 A*-Algorithmus

Der A*-Algorithmus gilt als Weiterentwicklung des Dijkstra-Algorithmus, der grundsätzliche Ablauf beider Algorithmen ist gleich. Allerdings wird beim A*-Algorithmus die kürzeste-Pfad-Suche um eine Schätzfunktion (Heuristik) erweitert, was zur schnelleren Ergebnisfindung beiträgt. Dabei wird zusätzlich zu den Kantengewichten die kürzeste Entfernung zwischen Start- und Zielknoten geschätzt. Dadurch werden nicht alle Knoten abgesucht, sondern nur diejenigen, die am wahrscheinlichsten Teil des kürzesten Pfades sind.

Die Heuristik darf dabei nicht überschätzt werden, sodass der tatsächliche kürzeste Pfad immer länger sein muss als der geschätzte. Bei Wegstreckenberechnungen stellt dabei die Luftlinie zwischen zwei Knoten eine günstige Heuristik dar. Aufgrund der Schätzfunktion ist neben der Angabe eines Startknotens die eines Zielknotens unbedingt erforderlich. Wie der Dijkstra-Algorithmus auch findet der A*-Algorithmus immer die optimale Lösung.

### 4.2.3 Bellman-Ford-Algorithmus

Der Bellman-Ford-Algorithmus unterscheidet sich vom Dijkstra- bzw. dem A*-Algorithmus. Mit ihm lassen sich auch Graphen mit negativen Kantengewichten berechnen, es dürfen sich allerdings keine negativ gewichteten Zyklen in dem Graphen befinden, die vom Startknoten aus erreichbar sind (was aber durch den Algorithmus erkannt und ausgegeben wird) [12]. Im Gegensatz zum Dijkstra-Algorithmus werden Knoten mehrmals besucht, da sich der Bellman-Ford-Algorithmus iterativ der Lösung nähert. Die Knoten erhalten wie beim Dijkstra-Algorithmus auch ihre Kosten vom zurückliegenden Pfad aus den einzelnen Kanten, durch mehrmaliges Ablaufen der Knoten und Kanten werden die Kosten jedoch laufend angepasst, bis keine Optimierung mehr möglich ist. Dann ist der Algorithmus beendet und der kürzeste Pfad gefunden.

### 4.2.4 Algorithmus von Floyd und Warshall

Mit dem Algorithmus von Floyd und Warshall lassen sich ebenfalls kürzeste Pfade innerhalb von Graphen mit negativem Kantengewicht ermitteln. Auch bei diesem dürfen keine negativ gewichteten Zyklen vorhanden sein [12]. Zunächst werden alle Distanzen von direkten Knotenverbindungen ermittelt, Distanzen nicht direkt verbundener Knoten bleiben vorerst unbekannt. Danach wird anhand der Verbindungen zu den Nachbarknoten geprüft, ob es für eine bekannte Distanz einen kürzeren Pfad über die benachbarten Knoten gibt. Während des Durchlaufs kann eine Distanz mehrfach verändert werden. Als Ergebnis wird der kürzeste Pfad von jedem Knoten zu einem anderen Knoten ausgegeben, sofern diese über Zwischenknoten erreichbar sind. Im Gegensatz zu den anderen Algorithmen muss

beim Algorithmus von Floyd und Warshall kein Start- oder Zielknoten angegeben werden. Bei diesem werden alle möglichen Pfade von allen Knoten zueinander ermittelt.

### 4.2.5 Auswahl des geeignetsten „Kürzester-Pfad-Algorithmus“

Der A*-Algorithmus ist in Bezug auf die Qualität der Ergebnisse gut geeignet, da er die optimale Lösung findet. Er arbeitet zudem sehr effizient, da aufgrund der verwendeten Heuristik nicht alle Knoten besucht werden müssen. Dieser Vorteil ist für die Lösungsfindung innerhalb von Räumen jedoch ungeeignet, da ein Endknoten zum Zeitpunkt der Rettungswegermittlung noch nicht bekannt ist. Da der A*-Algorithmus zwingend einen Start- und einen Endknoten vorgegeben bekommen muss, um die Heuristik anzuwenden, kann er an der Stelle somit nicht eingesetzt werden. Für Verbindungen, bei denen Start- und Endpunkt bekannt sind (z.B. Flure) kann er problemlos verwendet werden.

Der Algorithmus von Floyd und Warshall hat, wie auch der Bellman-Ford-Algorithmus, seinen Vorteil darin, dass er mit negativen Kantengewichten umgehen kann, sofern keine negativen Zyklen im Graph vorhanden sind. Darüber hinaus findet der Algorithmus von Floyd und Warshall sämtliche kürzesten Pfade, ohne dass ein Startknoten angegeben werden muss. Da der Bellman-Ford-Algorithmus und der Algorithmus von Floyd und Warshall eine längere Laufzeit haben als der Dijkstra-Algorithmus [12] und das Berücksichtigen negativer Kantengewichtung im Rahmen der Rettungswegermittlung nicht notwendig ist, werden beide in dieser Arbeit nicht eingesetzt.

Der Dijkstra-Algorithmus findet immer die optimale Lösung, sodass er hinsichtlich Genauigkeit für die Rettungswegermittlung gut geeignet ist. Da beim Start eines Algorithmus noch nicht bekannt ist, welcher Knoten am weitesten vom Startknoten entfernt ist (siehe dazu Abschnitt 4.3, bedarf es bei der Rettungswegermittlung eines Algorithmus, der diesen Knoten findet. Der Dijkstra-Algorithmus leistet dieses, was der signifikante Vorteil gegenüber dem A*-Algorithmus ist. Zudem weist der Dijkstra-Algorithmus auch eine geringe Laufzeit auf, weshalb er im Softwareprototypen dieser Arbeit Anwendung findet.

## 4.3 Datenmodellansätze für die Rettungswegermittlung

Nach Musterbauordnung muss „von jeder Stelle eines Aufenthaltsraumes sowie eines Kellergeschosses [...] mindestens ein Ausgang in einen notwendigen Treppenraum oder ins Freie in höchsten 35 m Entfernung erreichbar sein“ [5]. Dies bedeutet, dass die Wegstrecke aus einem Raum immer von dem am weitesten entfernt gelegenen Punkt in diesem Raum gemessen werden muss. Es ist also zunächst dieser Punkt zu ermitteln, bevor dann von diesem Punkt aus die kürzeste Wegstrecke aus dem Raum heraus zu einem notwendigen Treppenraum oder dem Ausgang ins Freie bestimmt werden kann.

Durch die Anwendung des Dijkstra-Algorithmus (siehe Abschnitt 4.2.1) werden die kürzesten Pfade zu allen Knoten in einem Graphen gefunden. Bei einem betrachteten Raum müssen also nur alle möglichen Knoten mittels Dijkstra-Algorithmus gefunden und dann derjenige mit der größten Entfernung zum Startpunkt ausgewählt werden.

Jeder Raum wird aus dem Zusammenschluss von mehreren Wänden, einem Boden und einer Decke gebildet. Da die Rettungswegermittlung zweidimensional und geschossweise erfolgt, genügt hierfür die Betrachtung der Wände. Sie bilden die Raumgeometrie im Grundriss. Zunächst ist also aus der Raumgeometrie ein Graph zu erstellen, der die wesentlichen Knoten und Verbindungen abbildet. Die Daten hierfür werden aus der entspre-

chenden IFC-Datei gefiltert, dazu mehr in den Kapiteln 5 und 6. An dieser Stelle wird angenommen, dass die Koordinaten von Wänden und Türen in einem zweidimensionalen Koordinatensystem bereits vorliegen. Im darauffolgenden Schritt werden mithilfe des Dijkstra-Algorithmus die kürzesten Pfade von der Tür (dem Ausgangspunkt eines Raumes) zu allen weiteren Punkten/Knoten ermittelt, sodass die Stelle bekannt wird, die sich am weitesten von einem Rettungsweg entfernt befindet.

Im Folgenden werden zwei unterschiedliche Ansätze für Algorithmen erläutert, mit denen der Graph und damit die Adjazenzmatrix für einen Raum erstellt werden kann. Anschließend werden die Vor- und Nachteile dieser Vorgehensweisen gegeneinander abgewogen, sodass unter dieser Analyse die favorisierte Methode in dem Softwareprototypen Anwendung findet.

### 4.3.1 Graph aus Eckpunkten

#### Allgemeines

Ein Ansatz zur Findung der für die Kürzester-Pfad-Berechnung notwendigen Knoten ist, die Eckpunkte der Wände eines Raums als solche Knoten zu nutzen. Ein rechteckiger Raum beispielsweise bestünde damit aus fünf Knoten; einer repräsentiert die Tür und vier für die Ecken. Die zugehörige Adjazenzmatrix hätte 25 Einträge. Die Lösung in diesem Falle ist offenkundig, da von jeder Ecke aus eine direkte Verbindung (Kante) zur Tür besteht, was direkt den jeweiligen Wegstrecken entspricht. Den entsprechenden Graphen aus diesem Grundriss zu erzeugen, wäre daher sehr einfach.

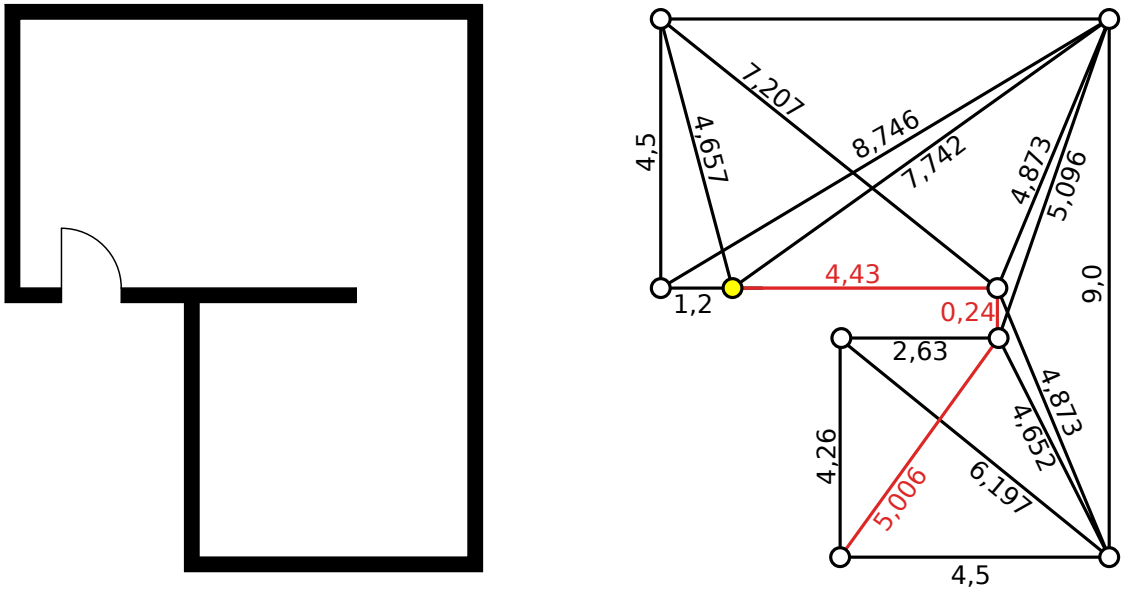



Abbildung 4.7: Raum mit zugehörigem Graphen aus Eckpunkten

Es können aber auch größere, vieleckige Grundrisse für einen Raum existieren. In diesen bestehen dann nicht mehr zwischen allen Punkten direkte Verbindungen, sodass zunächst andere Punkte passiert werden müssen, um von einem Startpunkt aus das Ziel zu erreichen. Ob zwischen zwei Punkten (Knoten) eine direkte Verbindung existiert, ober sich z.B. eine Wand zwischen ihnen befindet, muss durch den Algorithmus bei der Erzeugung des Graphen ermittelt werden.

### Ablauf des Verfahrens

Zur Verarbeitung in der Anwendung werden Wände als einzelne Strecken mit Anfangs- und Endkoordinaten ($x$,$y$) betrachtet, welche den Raum umschließen. Aus den äußeren Wänden setzt sich ein Polygon zusammen, dessen Ecken gleichzeitig aus dem Endknoten einer Wand sowie dem Startknoten einer benachbarten Wand bestehen. Diese Ecken sind Teil der Knoten des dem Raum zugehörigen Graphen. Wichtig zu beachten ist, dass ausschließlich die äußeren umschließenden Wände als Teil des Polygons erkannt werden, weil im weiteren Ablauf geprüft werden muss, ob bestimmte Punkte innerhalb oder außerhalb des Polygons liegen. Da es frei im Raum stehende Wände geben kann, die kein Teil der raumabschließenden Wände sind, müssen diese separat betrachtet werden, weil es sonst zu fehlerhaften Auswertungen bei der Prüfung, ob ein Punkt innerhalb oder außerhalb des Polygons liegt, kommt. Für die interne Verarbeitung ist es günstig, die geometrischen

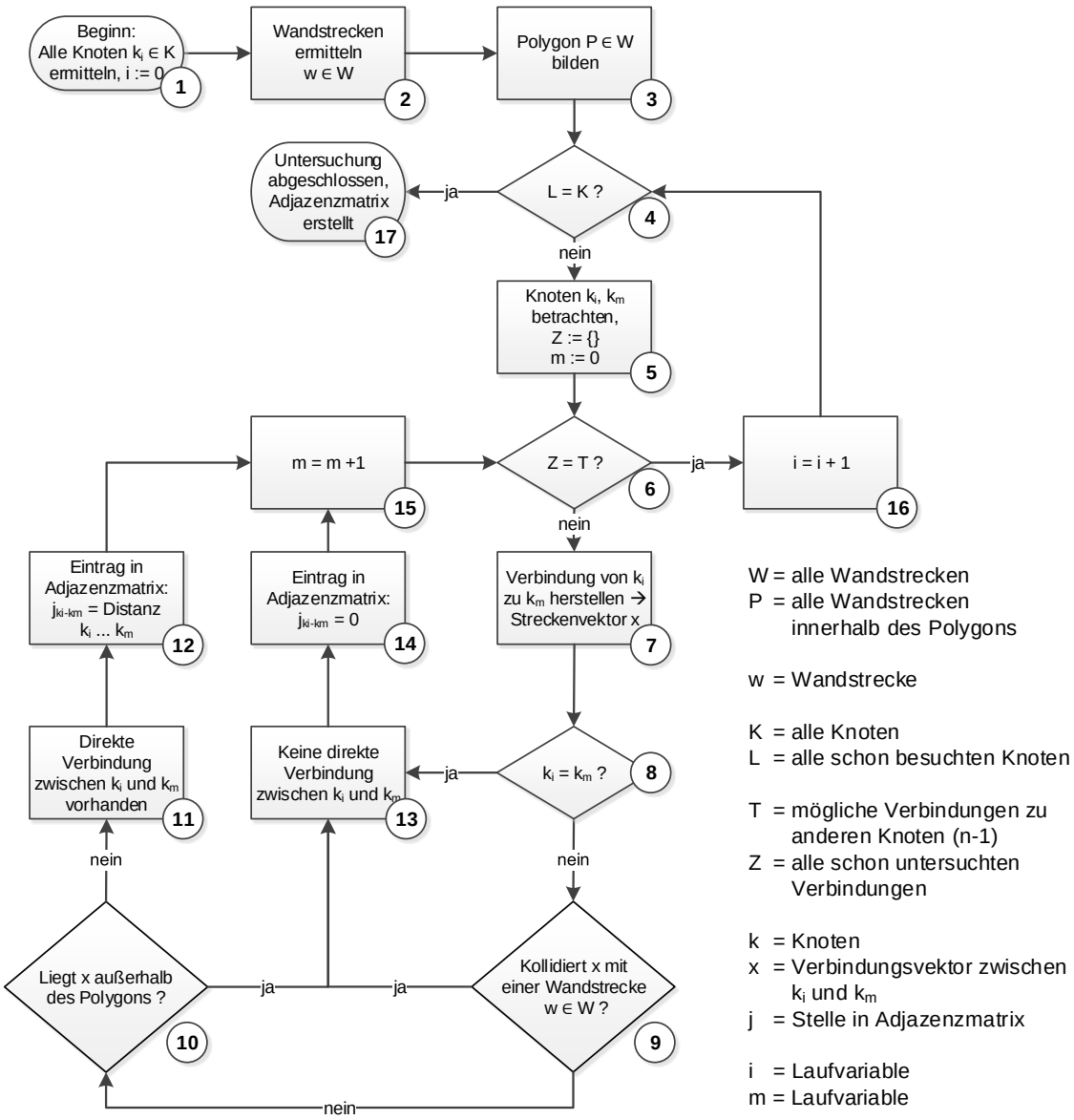



Abbildung 4.8: Schematischer Ablauf der Ermittlung des Graphen aus Eckpunkten

Elemente (Wände, Verbindungen) mit Vektoren zu beschreiben. In diesem sind Start- sowie Endpunkt (und damit auch entsprechend die Streckenlänge) enthalten, was für spätere Berechnungen notwendig ist.

Nachdem alle vorhandenen Knoten, Wandstrecken und das Raumpolygon bestimmt sind, werden nun die Knoten nacheinander einzeln betrachtet und direkte Verbindungen zu ihnen gesucht; eine graphische Übersicht liefert Abbildung 4.8. Dafür wird ein Knoten $k$ betrachtet und ein Vektor $x$ von diesem Knoten zu einem benachbarten Knoten $n$ erstellt (Nr. 7 in Abb. 4.8).

An dieser Stelle wird zuerst geprüft, ob eine hindernisfreie, direkte Laufstrecke zwischen den Knoten $k$ und $n$ existiert (Nr. 9 in Abb. 4.8), d.h., ob der Vektor $x$ kein Wandelement $w$ schneidet. Sofern keines der Wandelemente geschnitten wird, gibt es eine freie Strecke zwischen den Knoten. Die mathematische Berechnung der Schnittpunktberechnung ist schemenhaft in Abbildung 4.9 dargestellt; an dieser Stelle wird nicht weiter darauf eingegangen. Es sei lediglich gesagt, dass zwei Vektoren, die nicht echt parallel zueinander stehen, sich an irgendeinem Punkt immer schneiden. Liegt der Schnittpunkt jedoch außerhalb der betrachteten Strecke (also nicht auf der Strecke $x$ zwischen den Knoten $k$ und $n$), ist die untersuchte Verbindung hindernisfrei (siehe Nr. 6 in Abb. 4.9) und es kann die Prüfung der Lage erfolgen.

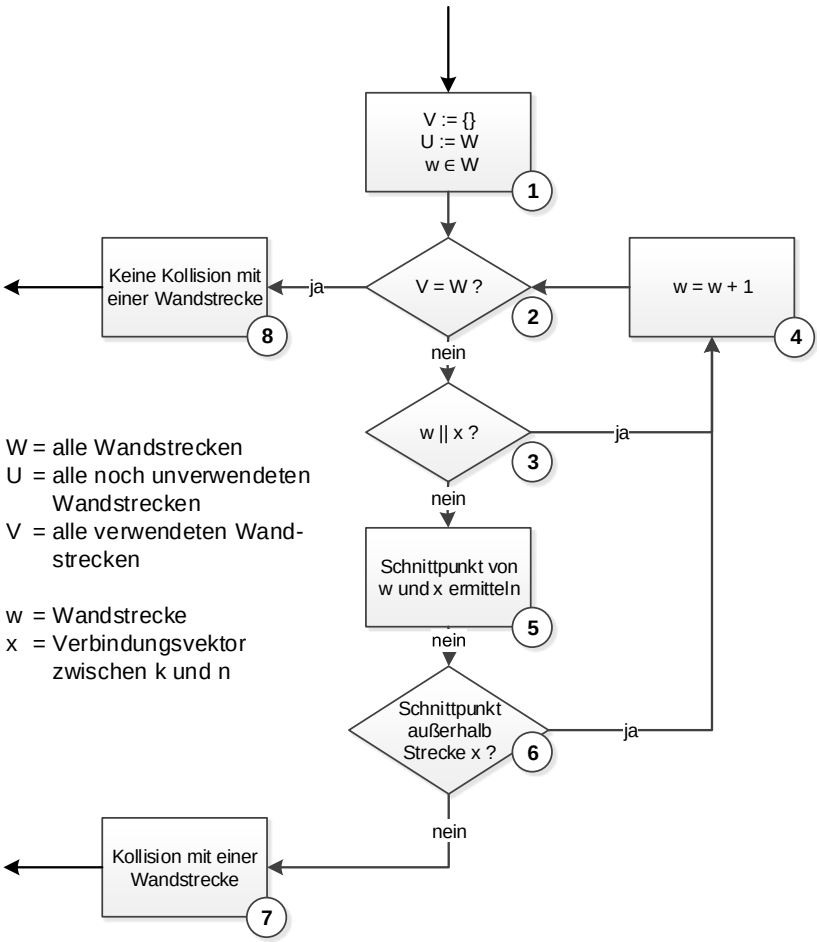



Abbildung 4.9: Prüfung auf Schnittpunkte zwischen potenziellem Laufweg und Wänden

Je nach Raumgeometrie kann es passieren, dass eine mögliche Verbindung theoretisch hindernisfrei ist (also von keiner Wand geschnitten wird), praktisch aber nicht existiert, weil die Verbindung außerhalb des Raums liegt, wie z.B. die Verbindung *TÜR - 5* in Abbildung 4.10. Daher muss noch eine Prüfung stattfinden, ob die Verbindungsstrecke innerhalb oder außerhalb des Raumpolygons liegt (siehe Nr. 10 in Abb. 4.8). Dies kann unter Verwendung des *Punkt-in-Polygon-Tests nach Jordan* geschehen.

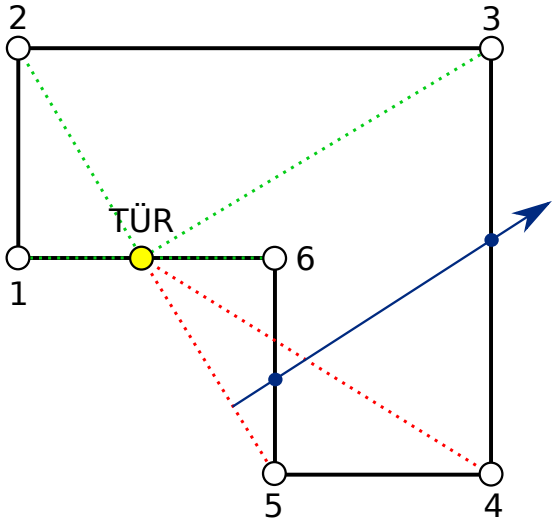



Abbildung 4.10: Prüfung der möglichen Verbindungen eines Knotens

Bei dem Punkt-in-Polygon-Test nach Jordan wird ermittelt, ob sich ein betrachteter Punkt innerhalb eines geschlossenen Polygonenzugs befindet oder nicht [2]. Dafür wird von diesem Punkt ein Strahl[4] in eine beliebige Richtung ausgerichtet und die Anzahl der Schnittpunkte mit dem Polygon gezählt. Wird das Polygon nicht oder in einer geraden Anzahl von dem Strahl geschnitten, so liegt der untersuchte Punkt außerhalb des Polygons, bei einer ungeraden Anzahl an Schnittpunkten innerhalb des Polygons [2]. Deshalb dürfen auch nur die Wände des Polygons zur Prüfung herangezogen werden, da innenliegende Wände ebenfalls einen Schnittpunkt erzeugen könnten und es somit zu einem verfälschten Ergebnis kommen würde.

Im Falle der Überprüfung, ob eine Verbindungsstrecke zwischen zwei Knoten innerhalb des Raums liegt, ist es ausreichend, einen beliebigen Punkt auf der Verbindungsstrecke auszuwählen und von diesem aus den Punkt-in-Polygon-Test durchzuführen. Natürlich kann es passieren, dass eine Strecke teilweise innerhalb und teilweise außerhalb des Polygons liegt (so wie die Strecke *TÜR - 4* in Abb. 4.10) und der Ausgangspunkt für den Strahl nun zufällig innerhalb oder außerhalb des Polygons liegt. Solche Konstellationen führen jedoch immer dazu, dass die potenzielle Verbindungsstrecke von einer Wand geschnitten wird und dadurch im bereits vorhergehenden Schritt diese Verbindung als nicht-existent in die Adjazenzmatrix eingetragen wird.

Sind beide Nachweise der Prüfung auf einen Schnittpunkt mit einem Wandelement und der Punkt-in-Polygon-Prüfung erbracht (Nr. 9 und 10 in Abb. 4.8), besteht die untersuchte Verbindung $x$ und die Distanz zwischen den beiden Knoten $k$ und $n$ wird an der entsprechenden Stelle als Kantenwert in die Adjazenzmatrix eingetragen (Nr. 12 in Abb. 4.8). Ist

[4] Halbgerade; beginnt an einem Punkt und erstreckt sich ins Unendliche.

eine der beiden Prüfungen negativ ausgefallen, so wird der Wert 0 in die Adjazenzmatrix eingetragen.

An dieser Stelle ist die Ermittlung für die erste potenzielle Verbindung abgeschlossen. In dem Verfahren wird nun der nächste Knoten $n$ ausgewählt (Nr. 15 in Abb. 4.8) und die Prüfung beginnt erneut. Sobald alle anderen Knoten besucht und somit potenzielle Verbindungen überprüft sind, werden vom nächsten Knoten $k$ aus wieder alle möglichen Verbindungen zu den benachbarten Knoten $n$ ermittelt, solange bis alle Knoten $k$ besucht sind und die Untersuchung beendet wird.

### Vor- und Nachteile

Ein großer Vorteil dieser Methodik besteht in der begrenzten Anzahl an Knoten und dem damit verbundenen geringen Rechenaufwand. Bei einer Knotenanzahl $k$ müssen $k * (k-1)$ Verbindungen untersucht werden, was mit modernen Computern problemlos und unmittelbar gelingt. Damit lassen sich auch größere Objekte zügig berechnen, sodass beispielsweise Architekten während der Planungsphase schnell ihre Rettungswegsituation überprüfen können, ohne lange Rechenzeiten dafür in Kauf nehmen zu müssen.

Ein weiterer Vorteil, den Graph aus den Eckknoten der Raumgeometrie abzubilden, besteht in der Korrektheit der Ergebnisse. Die Entfernung zwischen zwei Knoten entspricht auch (mit kleinen Einschränkungen, siehe nächster Absatz) der tatsächlichen Laufstrecke in einem Gebäude. Das realistische Ergebnis kann somit zu dem Vergleich mit den in den Bauordnungen vorgeschriebenen Werten herangezogen werden.

Natürlich bildet das hier vorgestellte Verfahren nicht exakt die Realität ab. Beispielsweise besteht in Abbildung 4.10 eine direkte Verbindung vom Knoten 6 zur Tür mit einer Distanz $d$. Die tatsächliche Laufstrecke wird in der Realität etwas länger sein, da sich die modellhafte Verbindung exakt auf der Wandstrecke befindet, die Laufstrecke aufgrund der Wanddicke und der Breite einer Person „neben“ der Verbindung liegt. Dieser Fehler lässt sich jedoch korrigieren. Eine Möglichkeit zur Korrektur besteht durch ein angepasstes Modell, in dem Hilfsknoten eingeführt werden, die sich an den vorhandenen Wandelementen orientieren und die resultierenden Abstände aus Personenbreite und Wanddicke einhalten. Ebenso könnten auch Korrekturfaktoren für die Streckenlänge ermittelt werden, was einer weiterführenden, empirischen Untersuchung bedürfte.

Die Anwendung des *Punkt-in-Polygon-Test nach Jordan* birgt die Problematik, dass ein Strahl genau auf einer Verbindungskante liegen kann. In diesem Falle herrscht ein indefiniter Zustand, da der Strahl unendlich viele Schnittpunkte hat und keine Aussage getroffen werden kann, ob sich ein Punkt innerhalb oder außerhalb des Polygons befindet. Lösungsansätze sind, den Strahl in anderen Winkeln auszurichten oder mittels infinitesimaler Verschiebungen das Überdecken von Strahl und Verbindungskante zu vermeiden [2]. Dies ist im Verfahren zu implementieren.

Der größte Nachteil der Abbildung des Graphen aus Eckknoten besteht darin, dass nur Knoten aus Eckpunkten für den Graphen ermittelt werden. Damit wird gleichzeitig unterstellt, dass der am weitesten von einem Ausgang entfernte Punkt immer in einer Raumecke liegt. Dies ist im überwiegenden Teil bestehender und zukünftig geplanter Raumgeometrien auch der Fall.[5] Dennoch kann es vorkommen, dass sich ein solcher Punkt auch mittig auf einer Geraden befinden kann, und zwar dann, wenn sich im Raum frei platzierte Wände befinden, so wie in Abbildung 4.11.

[5] An dieser Stelle werden nur Raumgeometrien mit geraden Wandelementen betrachtet.

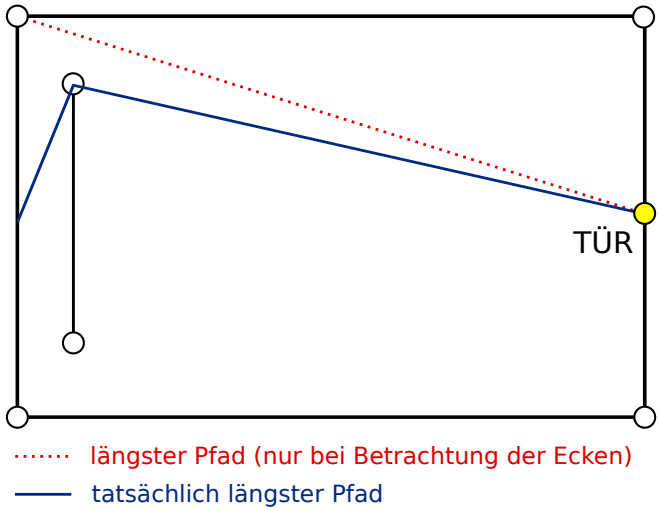



Abbildung 4.11: Vermeintlicher und tatsächlich längster Pfad

Der Fehler, der hieraus resultiert, ist vergleichsweise groß und muss dementsprechend vermieden werden. Korrekturfaktoren sind hier nicht anwendbar, da der Fehler je nach Raumgeometrie unterschiedlich groß ausfallen kann und ein statischer Faktor der erforderlichen Genauigkeit nicht entsprechen kann. Eine Lösungsmöglichkeit hierfür wäre, auf den Wandelementen Hilfsknoten einzuführen, was jedoch ebenfalls keine allgemeingültige Lösung sein kann, da der entsprechende Punkt auch mitten in einem Raum liegen kann, wenn zwei Ausgänge vorhanden sind (und auch aufgrund des Personenstroms benötigt werden). Es ist somit eine andere Möglichkeit zu finden, solche Punkte ermitteln zu können.

### 4.3.2 Graph aus Rasterknoten

#### Allgemeines

Eine zweite Möglichkeit, den Graphen eines Grundrisses aus einem Knotenraster zu bilden, gelingt über eine Rasterung. Dabei werden über den Grundriss eines Raumes Knoten in einem rechteckigen Raster mit einem vorgegebenen Abstand verteilt. Im Gegensatz zu der Graph-aus-Ecken-Methode werden nicht alle Verbindungen von allen Knoten zueinander gesucht, sondern nur die Verbindungen zu den direkten „Raster-Nachbarn“ eines betrachteten Knotens.

Die Anzahl der zu untersuchenden Knoten ist bei dieser Methode deutlich höher als bei der Graphenbildung aus Eckknoten. Besteht beispielsweise ein Graph für einen rechteckigen Raum mit den Abmessungen 3 m × 5 m aus insgesamt fünf Knoten, beinhaltet er nach Rastermethode 126 Knoten bei einem Rasterabstand von 0,4 m. Die zugehörige Adjazenzmatrix enthält entsprechend dem Quadrat 15876 Einträge, da sie ja alle Beziehungen der Knoten zueinander darstellen muss. Aus der Tatsache, dass ausschließlich die Verbindungen zu den direkten Nachbarknoten eines jeden Knoten untersucht werden, ergibt sich, dass der überwiegende Teil der Einträge in der Adjazenzmatrix 0 ist.

Mittels der Rastermethode werden alle Punkte eines Raumes mit der Genauigkeit des Rasterabstandes erfasst und ausgewertet. Somit wird in jedem Fall der Punkt gefunden, der von einem Ausgang am weitesten entfernt liegt, auch wenn er mitten im Raum liegt.

Durch den Einsatz des Dijkstra-Algorithmus wird der kürzeste Pfad von diesem Punkt zu einem Ausgang durch das Raster ermittelt.

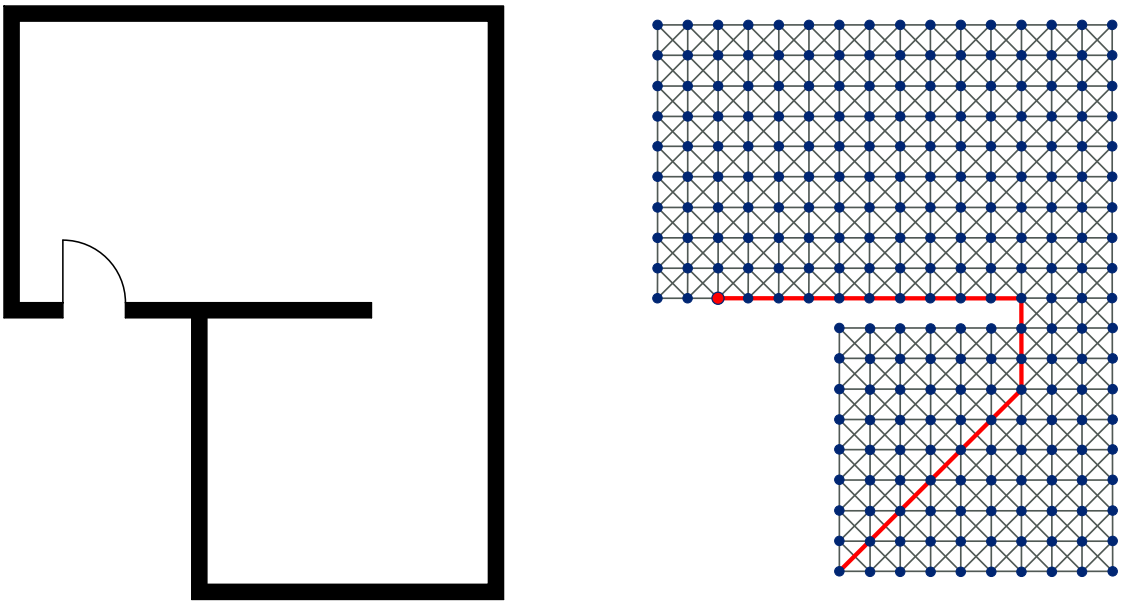


Abbildung 4.12: Raum mit zugehörigem Graphen nach Rasterung

### Ablauf des Verfahrens

Für die Erstellung der Adjazenzmatrix werden zunächst alle Wandelemente des betrachteten Raumes mit ihren Eckkoordinaten $(x,y)$ ermittelt. Die Strecken zwischen den Eckpunkten werden als *Verbindungsvektoren* abgespeichert, um diese bei den folgenden Berechnungen leicht verarbeiten zu können. Die Wandelemente, die den Raum äußerlich umschließen, bilden das Raumpolygon $P$, anhand dessen die Zugehörigkeit der Rasterpunkte zum Raum geprüft wird. Ein vereinfachtes Schema des Verfahrensablaufes ist in Abbildung 4.13 dargestellt.

Nachdem die Koordinaten der Wandelemente bekannt sind, können unter all diesen die minimalen und maximalen Koordinaten gesucht werden, um die benötigte Rasterfläche zu erhalten („Bounding-Box“). Unabhängig von der Geometrie des Grundrisses wird die „Bounding-Box“ immer rechteckig, sodass einige Punkte möglicherweise außerhalb dieser Box liegen, weshalb später der *Punkt-in-Polygon-Test nach Jordan* durchgeführt werden muss. Aus programminternen Gründen sollte die „Bounding-Box“ an allen vier Seiten um eine Reihe an Rasterpunkten gegenüber den minimalen und maximalen Wandkoordinaten erweitert werden, weil im späteren Verlauf des Verfahrens Nachbarknoten abgefragt werden und sich Fehler sowie Probleme im Randbereich dadurch frühzeitig vermeiden lassen.

Das Raster selbst wird mit einem vorgegebenen Knotenabstand innerhalb der „Bounding-Box“ ausgelegt. Dieser Abstand sollte sinnvoll gewählt werden, da er zu weit einen größeren Fehler und zu eng einen enormen Rechen- und Zwischenspeicheraufwand generiert. Verschiedene Testdurchläufe haben ergeben, dass ein Rasterabstand von 0,5 m eine gute Rechenperformance bei akzeptabler Toleranz bietet.[6]

[6] Siehe dazu Abschnitt 6.5.2.

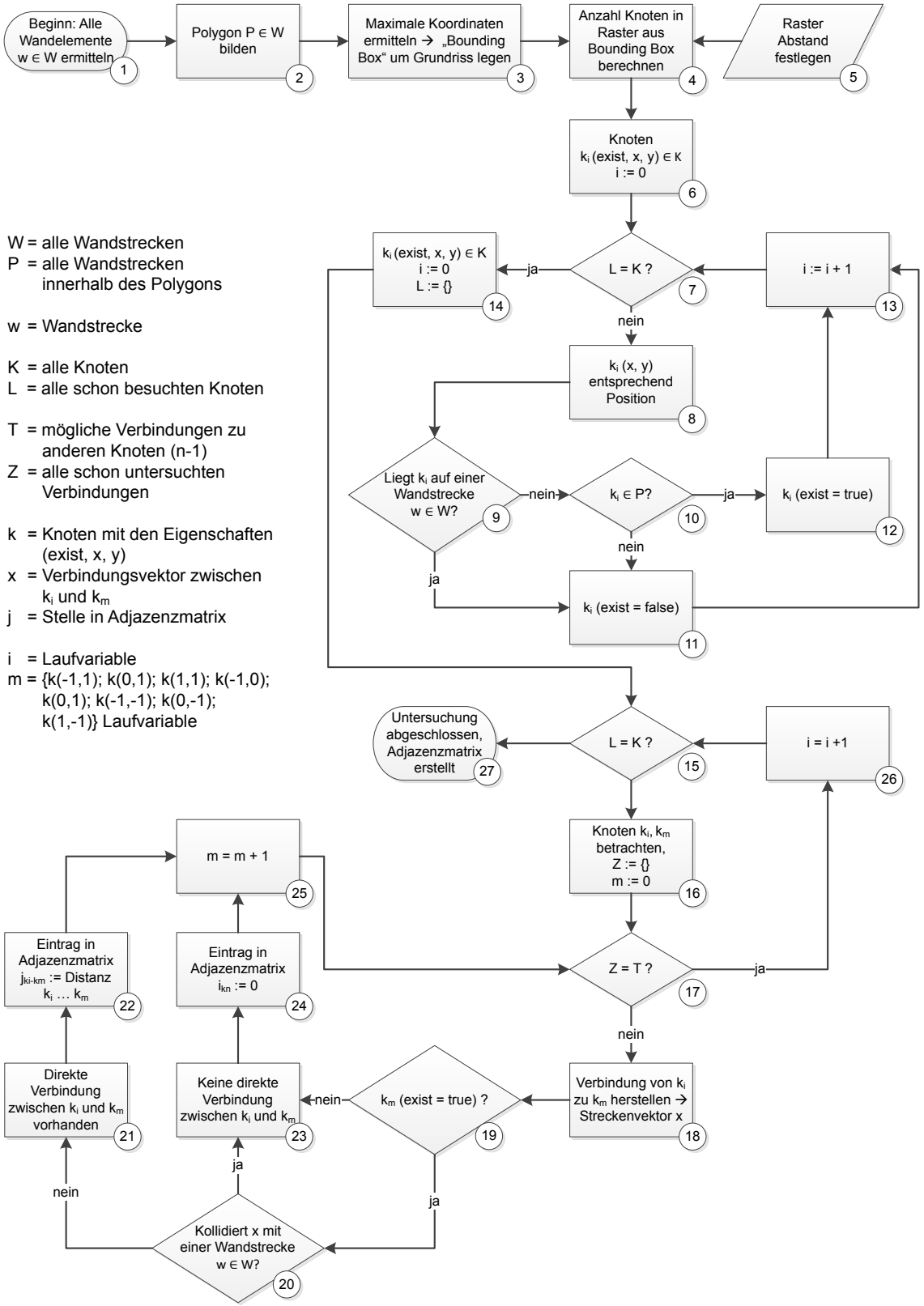



Abbildung 4.13: Schematischer Ablauf der Ermittlung des Graphen aus einem Raster

Die erzeugten Knoten $k_i$ werden intern als Objekte betrachtet, welche die Attribute $exist$, $x$ und $y$ haben (Nr. 6 in Abb. 4.13). Diese Eigenschaften werden in den folgenden Schritten 8 bis 11 zugewiesen. Die Koordinaten ($x$,$y$) im zweidimensionalen Grundriss eines Knotens ist durch den immer gleichen Rasterabstand bekannt und werden dem Knoten sofort als Attribut zugeordnet. Mit den Koordinaten kann zunächst überprüft werden, ob der Knoten Teil eine Wandelementes $w$ ist. Wenn dies der Fall ist, erhält der Knoten $k_i$ das Attribut $exist = false$. Damit kann im weiteren Ablauf schnell geprüft werden, ob zu diesem Knoten eine Verbindung hergestellt werden kann. Mit diesem Aussortieren wird auch das Problem bei dem im nächsten Schritt angewandten *Punkt-in-Polygon-Test nach Jordan* umgangen, bei dem ein indefiniter Zustand des Knotens eintritt, indem er auf der Wand liegt und keine klare Aussage getroffen werden kann, ob sich der Knoten innerhalb oder außerhalb des Polygons befindet. Anders als bei der *Graph-aus-Ecken-Methode* könnte zudem ein Knoten genau in einer Wandstrecke liegen und fälschlicherweise eine Verbindung „durch die Wand hindurch“ zwischen verschiedenen Knoten gefunden werden.

An dieser Stelle ist die Untersuchung der Existenz eines Knotens allerdings noch nicht abgeschlossen, da noch der *Punkt-in-Polygon-Test nach Jordan* für den betrachteten Knoten zu erfolgen hat (Nr. 10 in Abb. 4.13). Bei nicht-rechteckigen Raumgrundrissen wird es Knoten geben, die durch die „Bounding-Box“ grundsätzlich vorhanden sind, aber außerhalb des Raumpolygons liegen (siehe Abbildung 4.14). Daher muss noch eine In-Polygon-Prüfung erfolgen. Liegt der Knoten $k_i$ innerhalb des Polygons, so erhält er abschließend die Eigenschaft $exist = true$, andernfalls die Eigenschaft $exist = false$. Auf die nähere Erläuterung des *Punkt-in-Polygon-Test nach Jordan* wird hier nicht näher eingegangen, da sie bereits in Abschnitt 4.3.1 erfolgte.

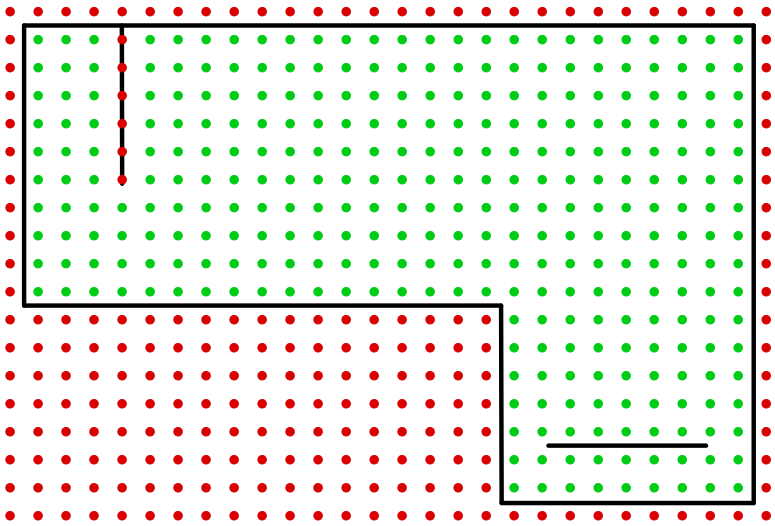


Abbildung 4.14: Raumgrundriss mit Rasterung

Ist die In-Polygon-Untersuchung für alle Knoten $k_i$ abgeschlossen, so haben alle ihre Attribute ($exist$,$s$,$y$) erhalten, womit die Untersuchung der Verbindungen beginnen kann. Der Verfahrensablauf hierfür entspricht im Wesentlichen dem Ablauf der Graph-aus-Eckpunkten-Methode aus Abschnitt 4.3.1. Doch im Gegensatz zu dieser Methode werden nun nicht alle denkbaren Verbindungen zu allen Knoten untersucht, sondern nur die Verbindungen zu den jeweiligen acht direkt benachbarten Knoten $k_m$ eines Knoten $k_i$. Ebenso werden nur Verbindungen von Knoten $k_i$, die selbst existieren (also $exist = true$).

Die Prüfung, ob eine Verbindung innerhalb oder außerhalb des Raumpolygons liegt, kann entfallen, da durch das Attribut *exist* schon bekannt ist, ob der benachbarte Knoten entweder Teil einer Wandstrecke oder außerhalb des Polygons liegt. Ist dies der Fall, wird an der entsprechenden Stelle der Adjazenzmatrix für diese Verbindung der Wert 0 eingetragen, da sie nicht vorhanden ist (siehe Nr. 19, 23, 24 in Abb. 4.13).

Wie in Abbildung 4.14 zu sehen ist, kann der Fall auftreten, dass eine Wand innerhalb eines Raumes zwischen den Rasterknoten liegt, sodass alle umliegenden Knoten $exist = true$ sind, aber keine direkte Verbindung zueinander haben. Daher ist die Prüfung auf Schnittpunkte mit den Strecken der Wandelemente durchzuführen, wie schon in Abschnitt 4.3.1 erklärt und in Abbildung 4.9 dargestellt. Gibt es keinen Schnittpunkt mit einer Wand zwischen den Knoten $k_i$ und $k_m$ existiert die Verbindung und wird als Kantengewichtung mit der entsprechenden Distanz $d$ in die Adjazenzmatrix eingetragen (Nr. 22 in Abb. 4.13).

Die Berechnung der Distanz $d$ zwischen zwei Knoten kann entfallen, da sie aufgrund des vorgegebenen Rasterabstandes $a$ bekannt ist. Zu horizontal oder vertikal benachbarten Knoten beträgt sie $a$, zu den diagonal benachbarten Knoten $\sqrt{2a^2}$. Sind alle potentiellen Verbindungen von allen Knoten $k_i$ untersucht, ist die Adjazenzmatrix vollständig und die Untersuchung kann abgeschlossen werden.

### Vor- und Nachteile

Der große Vorteil gegenüber der Graph-aus-Eckpunkten-Methode ist, dass hier auch Knoten gefunden werden können, die nicht Eckpunkte von Wänden sind und daher immer der Punkt innerhalb eines Raumes gefunden wird, der am weitesten von einem Ausgang entfernt liegt, auch wenn dieser frei im Raum liegt. Die Ungenauigkeit durch das Raster liegt in einem akzeptablen Bereich, sofern das Raster eine bestimmte Dichte nicht unterschreitet. Wie bereits erwähnt, ist den Versuchen nach ein Abstand von 0,5 m in Bezug auf Rechengeschwindigkeit, Speicherbedarf und Genauigkeit ein gutes Maß.[7]

Der Rechenaufwand ist jedoch ein grundsätzlicher Nachteil bei der Rasterung, weil sehr viele Knoten eingeführt und untersucht werden müssen, um ein brauchbares Ergebnis zu erhalten. Bei sehr großen Gebäuden kann dies durchaus einige Zeit in Anspruch nehmen, sofern der zu untersuchende Bereich nicht vorab begrenzt wird. Gerade unter dem Gesichtspunkt, dass der größte Teil der Knoten ohnehin nicht benötigt wird, ist dieser Algorithmus prinzipiell eher ineffizient.

Der größte Nachteil der Raster-Methode ist jedoch die Ungenauigkeit der Ergebnisse, die aus der indirekten Wegführung hervorgeht. Durch die Rasterung sind Laufwege ausschließlich horizontal, vertikal oder in einem 45°-Winkel möglich. Je nach Raumgeometrie kann ein Laufweg zwischen zwei (weiter entfernten) Knoten in einem anderen Winkel jedoch günstiger sein, was von der Rastermethode nicht berücksichtigt wird (Abbildung 4.15).

Die Differenz zwischen der ermittelten und der tatsächlichen Rettungswegläge hängt stark von der Raumgeometrie ab; je länger der Rettungsweg wird und damit potentielle Richtungswechsel zunehmen, desto größer wird die Abweichung. Da hier kein konstanter Wert vorliegt, ist eine pauschale Abschätzung dieses Fehlers nicht möglich. Ferner ist zu berücksichtigen, dass es bei der überwiegenden Anzahl an Gebäuden, bei welchen diese Art der Rettungswegermittlung zum Einsatz kommt, nur wenige bis keine Räume gibt, in denen Wände frei im Raum stehen oder mehrere Türen so angeordnet sind, dass sich der Knoten mit dem längsten Abstand zu einem Ausgang frei im Raum befindet. Da dies die einzigen

[7] Siehe Abschnitt 6.5.2.

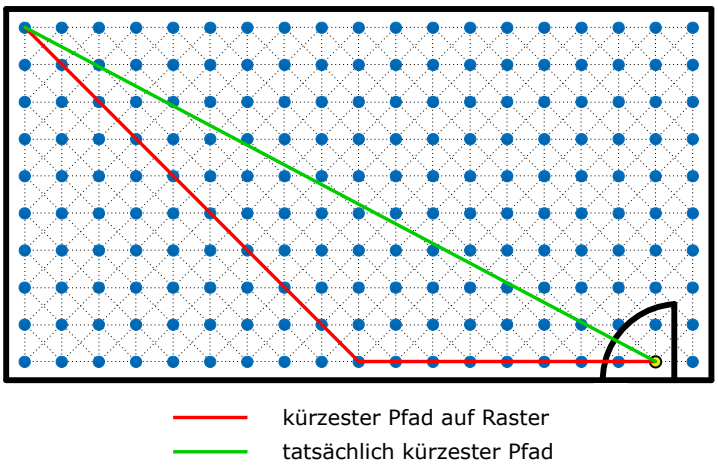



Abbildung 4.15: Raumgrundriss mit Rasterung

Fälle sind, in denen die Rastermethode ihre Vorteile gegenüber der Graph-aus-Eckpunkten-Methode ausspielt, würde bei sehr vielen Anwendungen ein methodenbedingter Fehler in Kauf genommen werden, der sich durch Anwendung eines anderen Algorithmus vermeiden ließe. Gleichwohl sind natürlich alle Geometrie-Szenarien zu berücksichtigen, damit das Programm vielfältig einsetzbar ist. Eine Optimierung der Raster-Methode hinsichtlich des benannten Fehlers ist nur damit möglich, die Findung von Verbindungen auch über mehrere Rasterpunkte hinweg zuzulassen.

## 4.4 Ergebnisse und weiteres Vorgehen

Aufgrund der prinzipbedingten Nachteile wird der Ansatz des „Graph aus Eckpunkten“ in dieser Arbeit nicht weiter verfolgt. Da nicht alle möglichen Fälle berücksichtigt werden (siehe Abbildung 4.11), ist diese Methode für den Einsatz in einer Rettungswegermittlung ungeeignet. Mit der „Graph aus Rasterknoten-Methode“ hingegen werden unabhängig von der Raumgeometrie immer die Knoten ermittelt, die auch tatsächlich am weitesten von einem bestimmten Startpunkt entfernt liegen. Aufgrund dessen ist sie die bevorzugte Methode und wird daher in dem Softwareprototypen implementiert (siehe Kapitel 6).

Für einen praxistauglichen Einsatz muss die Rastermethode hinsichtlich ihrer Genauigkeit jedoch noch optimiert werden. Möglich wäre eine Kombination aus der Graph-aus-Eckpunkten- und der Rastermethode. Dabei werden sowohl Rasterknoten erzeugt, als auch Knoten an den Eckpunkten von Elementen. Es werden dann Verbindungen wie bei dem Graph aus Eckpunkten mit der Erweiterung ermittelt, dass auch von den Rasterknoten mögliche Verbindungen zu den Eckknoten einbezogen werden (allerdings dann keine Verbindungen der Rasterknoten untereinander, um den Rechenaufwand zu reduzieren). Die kürzesten Pfade werden dann von jedem einzelnen Rasterknoten über die Eckknoten zum Zielpunkt ermittelt. Dadurch werden genauere Ergebnisse erzielt, da die Laufrichtung nicht mehr vom Raster vorgegeben wird, sondern frei ist. Allerdings werden durch die Eckknoten zusätzliche Knoten[8] eingeführt, was im Datenschema des Softwareprototypen (siehe Kapitel 6) ergänzend berücksichtigt werden müsste.

[8] Zusätzliche Knoten zu den Rasterknoten, denen durch das Datenschema des Softwareprototypen bereits bestimmte Attribute zugewiesen werden, die auf der Rasterung basieren.

Als alternative Optimierung könnte ein Verfahren gewählt werden, bei dem zunächst die kürzesten Pfade nach der Rastermethode ermittelt und die Pfade anschließend optimiert werden. Dabei werden nur noch die Knoten, an denen ein Richtungswechsel des Pfades vorliegt, berücksichtigt und direkte Verbindungen zwischen diesen Knoten erzeugt, sofern sie möglich sind. Dadurch wird ebenfalls die Richtungsvorgabe durch das Raster aufgehoben und es lassen sich die genaueren, direkten Wegstrecken ermitteln. Hierbei würden keine zusätzlichen Knoten benötigt werden, sodass alle Knoten im Verlauf der Wegstrecke auch Teilmenge der Rasterknoten sind. Dies würde keine Anpassung der Datenstruktur des Softwareprototypen erfordern.

Da eine Umsetzung dieser Optimierungen in dem Softwareprototypen den Umfang dieser Arbeit übersteigen würde, wird in dem Softwareprototypen lediglich die unoptimierte Rastermethode implementiert. Zur Veranschaulichung der prinzipiellen Machbarkeit ist dies ausreichend, sodass die Optimierung hinsichtlich der Ergebnisse dieser Arbeit verzichtbar ist.

# 5 Datenstruktur IFC

## 5.1 Erforderliche Elemente für den Softwareprototypen

Um die Grundlage für den Algorithmus aus Kapitel 4 zur Rettungswegermittlung zu schaffen, ist es notwendig, die entsprechenden geometrischen Informationen zu bestimmten Elementen aus einem Bauwerksmodell zu erhalten. Diese bilden den Grundriss, für den die Rettungswege nachzuweisen sind. Die in dieser Arbeit betrachteten geometrischen Elemente sind

- Wände,
- Stützen und
- Türen bzw. Türöffnungen.

Darüber hinaus gilt es auch, einige semantische Informationen zu erhalten. Für die reinen Grundrisse wäre eine *BIM-basierte* Rettungswegermittlung nicht nötig, da zweidimensionale Geometriedaten auch aus Dateien konventioneller Planungen erhältlich sind. Die *BIM-basierte* Rettungswegermittlung zeichnet sich dadurch aus, dass auch die semantischen Informationen (was ist ein Rettungsweg? Welche Räume existieren im Gebäude und wie viele Personen halten sich darin auf?) verarbeitet und mit einbezogen werden können. Dementsprechend müssen zusätzlich

- Rauminformationen (Geometrie und Belegung) sowie
- Informationen zu Rettungswegen (Treppenräume, Ausgänge ins Freie)

aus dem Bauwerksmodell gefiltert werden. Bei der konventionellen Planung sind Wände, Stützen, Einrichtungsgegenstände, Fenster und Türen lediglich zusammengesetzte Linien sowie Räume eine Ansammlung vieler zusammengesetzter Linien. Für das CAD-System sind diese Bauteile entsprechend nur Striche, eine Unterscheidung erfolgt allein durch einen kognitiven Entscheidungsprozess des Betrachters. Bei einer BIM-orientierten Bauwerksmodellierung werden die verschiedenen Bauteile auch als solche behandelt, sodass auch ein Computer in der Lage ist, zwischen diesen zu differenzieren.

Im IFC-Schema werden verschiedene Bauteile durch eigene Klassen repräsentiert, sodass eine grundsätzliche Unterscheidung von Bauwerkselementen anhand dieser Klassen erfolgen kann. Die Zuordnung von Bauteilen zu einer IFC-Klasse erfolgt beim Export der Datei aus dem CAD-Programm durch einen sogenannten „IFC-Übersetzter“. Theoretisch ist diese Zuordnung frei wählbar, sodass auch unsinnige Verknüpfungen (z.B. eine Decke als IfcWall) möglich sind; im Sinne eines vollständigen und für andere nachvollziehbaren Exports sollte dies aber unterlassen werden. In der Regel ist die korrekte Zuordnung in CAD-Programmen bereits konfiguriert, sodass die IFC-Datei automatisch generiert wird.

Aufgrund verschiedener Umstände kann es jedoch zu unterschiedlichen, wenn auch gleichermaßen korrekten Zuordnungen kommen. Zur Repräsentation von beispielsweise Wänden existieren im IFC-Schema zwei Klassen, die *IfcWall* und die *IfcWallStandardCase*, wobei letztere von erstgenannter Attribute erbt. Der Unterschied zwischen diesen beiden Klassen besteht darin, dass IfcWallStandardCase nur für Wände mit einer gleichbleibenden Dicke genutzt werden kann, während IfcWall auch komplexere Geometrien zulässt [18]. Gleichwohl ist es aber auch möglich, eine gerade Wand mit IfcWall zu repräsentieren, während

der umgekehrte Fall nicht vorgesehen ist. In der IFC-Datei des Beispielobjekts dieser Arbeit werden Wände beiden Entitäten zugeordnet, wobei eine genaue Regel nicht festgestellt werden konnte.

Alle Beispielgebäudemodelle, anhand derer der Softwareprototyp im Rahmen dieser Arbeit getestet wurde, sind mit dem Programm „ArchiCAD 21“ von der Firma Graphisoft erstellt und die IFC2x3-Datei [9] mit dessen „allgemeinen Übersetzer“ erzeugt worden. Die nachfolgende Tabelle gibt einen Überblick über die von diesem IFC-Export gewählten Zuordnungen.

Tabelle 5.1: Zuordnung von Bauteilen zu IFC-Klassen

| **Bauteil** | **IFC-Klasse** |
|---|---|
| Wände | IfcWall |
| | IfcWallStandardCase |
| Stützen | IfcColumn |
| Räume | IfcSpace |
| Öffnungen | IfcOpeningElement |
| Türen | IfcDoor |

Türen und Öffnungen sind unterschiedliche Elemente, da mit der Klasse IfcDoor das eigentliche Bauteil „Tür“ beschrieben wird, während IfcOpeningElement die Aussparung in einer Wand beschreibt. Räume sind im eigentlichen Sinne keine Bauteile, sondern ergeben sich aus dem Zusammenschluss mehrerer anderer Bauteile. Sie müssen jedoch im CAD-Programm von Hand gesetzt werden, wobei sich die Raumgeometrie automatisch ermittelt, weitere Eigenschaften dann aber manuell zuweisbar sind. Dieser entsprechende Raum wird durch die Klasse IfcSpace repräsentiert.

## 5.2 Beschreibung der IFC-Klassen

### 5.2.1 Allgemeines

Gemäß dem IFC-Schema stehen die entsprechenden Bauteilklassen in Beziehung zu anderen Klassen oder haben bestimmte vorgeschriebene oder optionale Attribute. Quellcode 5.1 zeigt beispielhaft die Einträge für eine Wand, repräsentiert durch die Klasse IfcWall, sowie ein dazu referenziertes Objekt der Klasse IfcProductDefinitionShape in einer IFC-Datei.

Quellcode 5.1: Auszug aus IFC-Datei

```
#159536=IFCWALL('0AR_2E_2L45w5bQ_EiTlY0',#12,'Wand-180',$,$,
   #159392,#159531,'0A6FE08E-F825-4417-A165-6BE3AC76F880');
#159531=IFCPRODUCTDEFINITIONSHAPE($,$,(#159514,#159520,#159528));
```

Der Aufbau dieser Zeilen ist durch das IFC-Schema vorgegeben und beinhaltet die jeweiligen Attribute. Diese sind durch Kommata getrennt, Dollarzeichen bedeuten leere Felder. In der Klasse *IfcWall* steht beispielsweise der erste Eintrag an der Stelle für die Element-ID, „Wand-180“ an der für den Elementnamen. Es ist zu erkennen, dass hier keine Informationen bezüglich der Wandgeometrie oder der Position hinterlegt sind. Diese werden durch Referenzen (gekennzeichnet durch die Rautezeichen) zu anderen Objekten hergestellt. So

[9] Da IFC2x3 die gegenwärtig verbreiteter ist als IFC4, wurde diese Version gewählt. Siehe dazu auch Abschnitt 2.2.3.

verweist #159531 hier auf das Objekt der Klasse *IfcProductDefinitionShape* darunter, welches wiederum auf weitere Objekte referenziert, aus denen schließlich die Geometrie der Wand hervorgeht. Mit der Referenzierung werden Redundanzen vermieden, die bei der Verarbeitung der Daten zu Problemen führen könnten. Als Beispiel könnte eine Wand mehrere Türöffnungen haben. Jede Öffnung bezieht sich dabei auf dieselbe Wand, sodass für diese Öffnungen eine Referenz zu dieser Wand hergestellt wird. Ohne die Referenzierung müsste für jede Öffnung ein neues Objekt „Wand“ erzeugt werden, was nicht mehr der Realität entspräche, da ja nur eine Wand existiert. Zudem kommt es spätestens dann zu Problemen, wenn an der Wand etwas verändert wird, da sich die Änderungen nur auf die „originale Wand“ auswirken und die weiteren Objekte dann manuell angepasst werden müssten.

Die Vorgaben zum Aufbau der einzelnen Zeilen für die jeweiligen Klassen werden in EXPRESS- oder XSD-Notation festgelegt. Quellcode A.1 in Anhang A.1 zeigt solche Festlegungen in EXPRESS-Notation für die Klasse IfcWall. Nicht alle der jeweiligen Eigenschaften bzw. Attribute sowie Referenzen werden dabei für jede Klasse neu definiert, sondern durch Vererbung von übergeordneten, abstrakten Klassen übernommen.

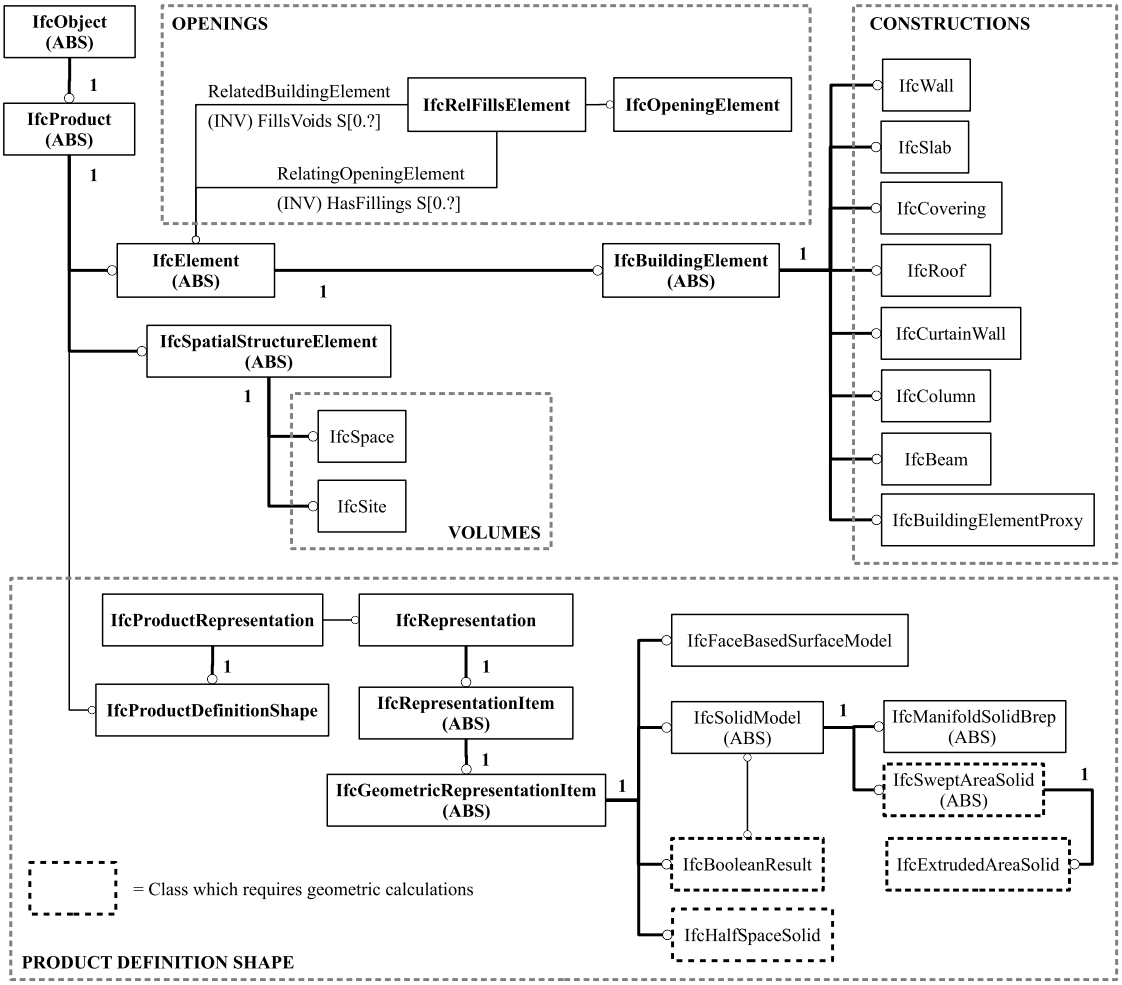



Abbildung 5.1: Unvollständiger Ausschnitt des IFC-Schemas
auf Basis der graphischen EXPRESS-G-Notation [26]

Abbildung 5.1 zeigt eine Übersicht des IFC-Schemas mit einigen wichtigen Klassen in der EXPRESS-G-Notation. Die dünneren Verbindungslinien zeigen dabei die Referenzen

zwischen verschiedenen Klassen, die dicker gezeichneten Linien stehen für die Vererbung, wobei jeweils die Klasse, an welcher sich der Kreis befindet, von der anderen Klasse erbt. So ist hier zu erkennen, dass z.B. sowohl die IfcWall- als auch die IfcColumn-Klasse Eigenschaften der übergeordneten Klasse IfcBuildingElement erben, die wiederum Attribute der IfcElement-Klasse erbt. Umgekehrt kommen alle Eigenschaften, die zur Klasse IfcElement gehören, auch in den Klassen IfcWall und IfcColumn vor. Nachfolgend werden die für diese Arbeit relevanten Bauteilklassen und deren referenzierte Klassen beschrieben.

### 5.2.2 Klassenstruktur der (Bau-)Elemente

Alle in einem IFC-Schema vorkommenden unabhängigen Klassen erben Attribute von der Klasse IfcRoot. Sie ist die höchste Instanz in der IFC-Vererbungshierarchie. So werden die Attribute *Global ID* und *Owner History* sowie die optionalen Attribute *Name* und *Description* durch diese Klasse definiert. Unabhängige Klassen sind in IFC i.d.R. solche, in der physische Elemente definiert werden (so z.B. in IfcWall), während beschreibende Klassen wie IfcBoundingBox abhängig sind, da sie nur in Verbindung mit unabhängigen Klassen vorkommen. Somit hat jedes Bauteil in einer IFC-Datei eine globale ID, anhand derer es eindeutig identifiziert werden kann.

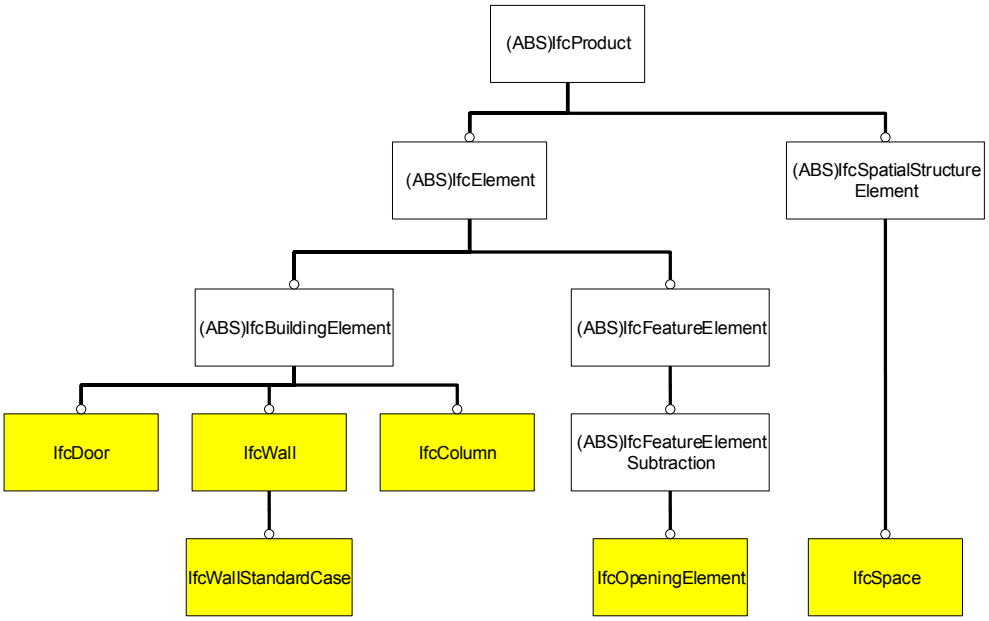



Abbildung 5.2: Vererbungshierarchie der Klassen aller benötigten (Bau-)Elemente in EXPRESS-G

Bauteile, von denen geometrische Informationen im Rahmen dieser Arbeit für die Rettungswegermittlung benötigt werden, sind Wände und Stützen. Zudem müssen Wandöffnungen bekannt sein, da Rettungswege zwangsläufig auch durch Türen verlaufen. Entsprechend Tabelle 5.1 werden diese Bauteile durch die Klassen IfcOpeningElement, IfcColumn und IfcWall bzw. IfcWallStandardCase repräsentiert. Diesen Klassen ist gemein, dass sie alle von der abstrakten Klasse IfcElement Eigenschaften erben. Diese wiederum erbt Eigenschaften von der abstrakten Klasse IfcProduct, in welcher Referenzen zu geometrischen Beschreibungen von Elementen festgelegt werden. Dementsprechend besitzt jede Unterklasse von IfcProduct eine Beschreibung der Form und eine Ortsangabe.

Darüber hinaus wird noch die Geometrie des Grundrisses eines jeden Raums benötigt, um eine Zuordnung der Rettungswege geordnet nach Räumen zu ermöglichen. Räume werden gemäß Tabelle 5.1 in der Klasse IfcSpace spezifiziert, welche eine Unterklasse der abstrakten Klasse IfcSpatailStructureElement ist. Diese ist ebenfalls eine Unterklasse von IfcProduct, sodass die geometrische Beschreibung von IfcSpace denselben Hintergrund hat wie die der IfcBuildingElements.

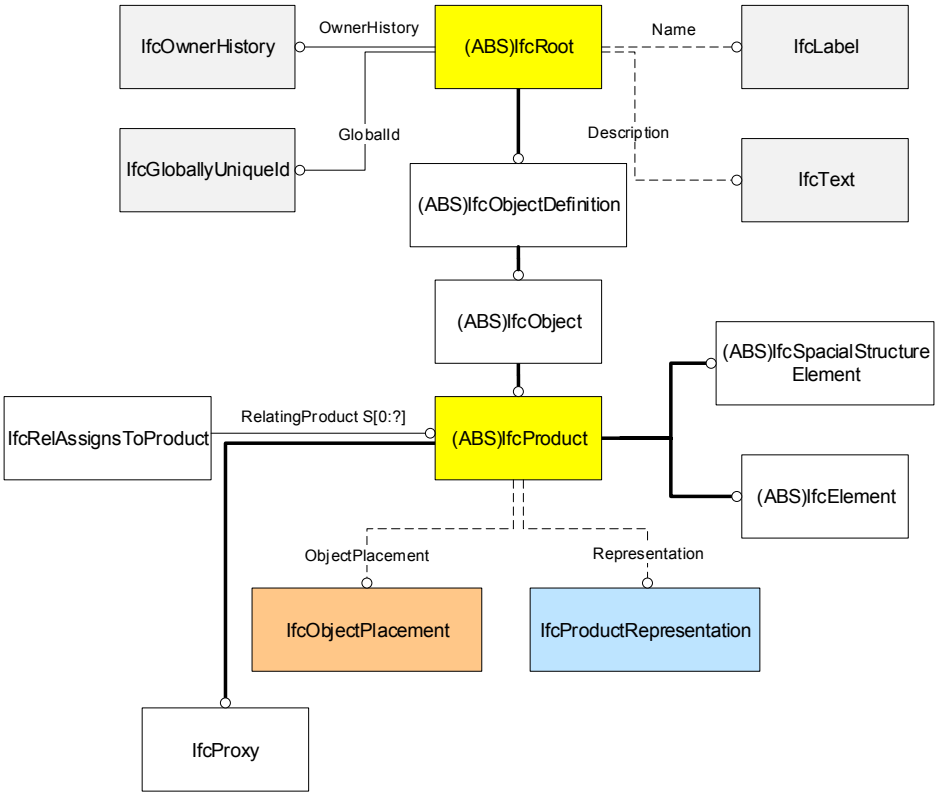



Abbildung 5.3: EXPRESS-G Darstellung der abstrakten Klassen IfcRoot und IfcProduct (unvollständige Übersicht)

Die grundsätzliche Bauteilgeometrie von Objekten aller Unterklassen von IfcProduct wird über eine Referenz zu einem Objekt der Klasse IfcProductRepresentation oder einer Unterklasse dieser beschrieben, die Bauteilplatzierung über eine Referenz zu einem Objekt der Klasse IfcObjectPlacement oder einer Unterklasse dieser. Zu Objekten der Unterklassen von IfcProductRepresentation und IfcObjectPlacement kann auch direkt von dem ausgehenden Objekt (hier ein Bauteil) hin referenziert werden, da die Objekte der Unterklassen von ihren Oberklassen erben und dementsprechend zuweisungskompatibel sind.

### 5.2.3 Bauteilgeometrie

Die geometrische Form eines jeden Unterobjekts der Klasse IfcProduct wird durch eine Instanz der Oberklasse IfcProductRepresentation oder einer ihrer Unterklassen (z.B. IfcProductDefinitionShape) näher bestimmt. Allerdings enthält sie noch keine Attribute zu Bauteilgeometrien; diese werden erst durch weitere Referenzen zu anderen Klassen festgelegt. Dadurch kann eine objektspezifische geometrische Beschreibung erfolgen, da bspw.

ein gekrümmtes Bauteil eine mathematisch komplexere Definition benötigt als eine quaderförmige Stütze. Abbildung 5.4 zeigt die entsprechenden Klassen und Referenzen inklusive ihrer Vererbungshierarchie, um die Geometriedarstellung eines Bauteils, einer Öffnung oder eines Raums zu erhalten.

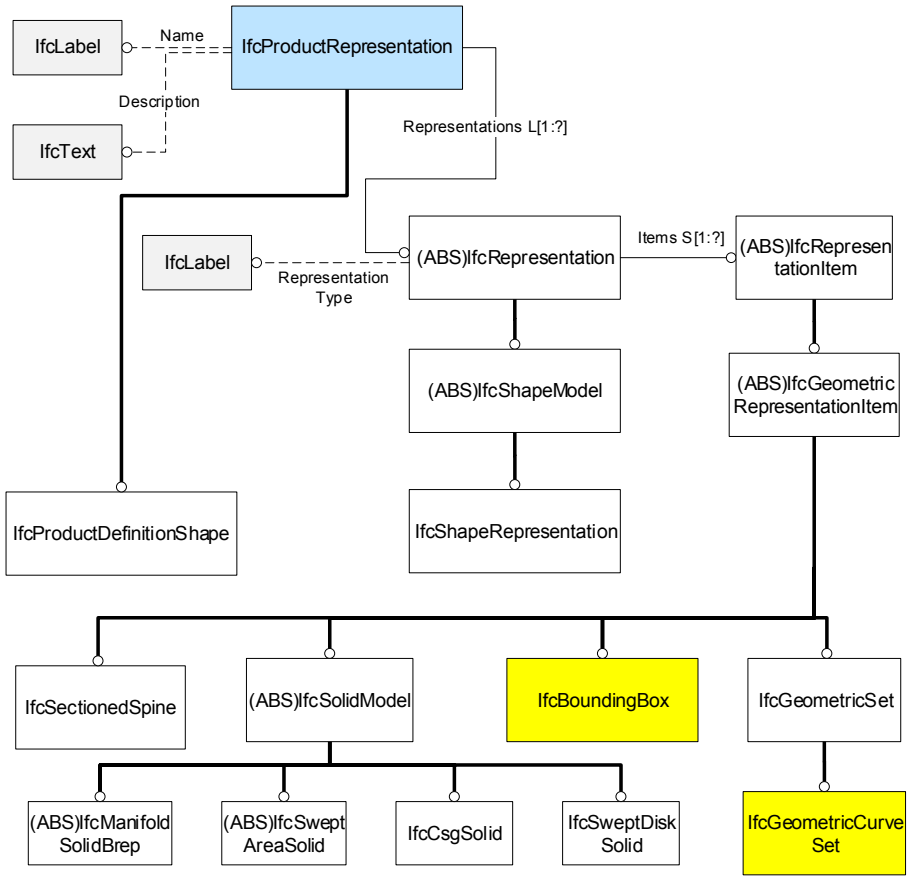



Abbildung 5.4: EXPRESS-G Darstellung der zur Bauteilgeometrie beschreibenden relevanten Klassen (unvollständige Übersicht)

Die Klasse IfcProductRepresentation schreibt, neben einem optionalen Namen und einer verbalen Beschreibung, zunächst nur vor, dass ein zu dieser Klasse referenzierendes Objekt mindestens ein oder mehrere weiterführende topologische oder geometrische Darstellungen einer Unterklasse von IfcRepresentation haben muss. Objekte, deren Form (engl. „shape“) beschrieben wird, referenzieren i.d.R. direkt zu IfcProductDefinitionShape, die sich gegenüber IfcProductRepresentation dadurch erweitert, dass zu dieser Klasse bestimmte Referenzen geführt werden, die formbestimmende Komponenten enthalten [18].

Durch die abstrakte Klasse IfcRepresentation wird vorgeschrieben, dass die referenzierende Instanz mindestens ein oder mehrere Objekte einer Unterklasse von IfcRepresentationItem hat, in welchen die Darstellungsräume definiert werden. Es wird zwangsweise zu einer Unterklasse von IfcRepresentation referenziert; sofern eine geometrische Beschreibung erforderlich ist, erfolgt die Referenz zu einem Objekt der Klasse IfcShapeRepresentation.

Die abstrakte Klasse IfcRepresentationItem ist die Oberklasse mehrerer Unterklassen, welche Darstellungsräume für topologische und eben auch geometrische Beschreibungen liefern. So ist die abstrakte Klasse IfcGeometricRepresentationItem die Oberklasse für 16

weitere Unterklassen, mit denen die geometrische Position und/oder die Orientierung von Objekten beschrieben werden kann. Abbildung 5.4 zeigt nur eine Auswahl dieser Klassen.

Da es sich bei allen Elementen aus Tabelle 5.1 um solche handelt, die eine geometrische Formbeschreibung benötigen, wird von diesen in der IFC-Beispieldatei direkt aufgrund der Zuweisungskompatibilität zu IfcProductDefinitionShape, von dort unmittelbar zu IfcShapeRepresentation und dann entsprechend zu IfcBoundingBox oder IfcGeometricCurveSet referenziert. Insbesondere diese letzte Zuordnung ist jedoch abhängig vom gewählten IFC-Übersetzer, sodass bei einem anderen CAD-Programm die Geometriebeschreibung eines identischen Gebäudemodells durch andere Klassen erfolgen kann. Im Rahmen dieser Arbeit wird sich allerdings auf IfcBoundingBox und IfcGeometricCurveSet beschränkt, da mit diesen einfach zu arbeiten ist und sie einen ausreichenden Informationsraum für den Anwendungsfall der Rettungswegermittlung bieten.

### IfcBoundingBox

Mit der Klasse IfcBoundingBox wird ein dreidimensionaler Körper durch seine maximale Ausdehnung in alle drei Richtungen beschrieben. Es ergibt sich dadurch immer ein Quader, der die äußeren Dimensionen des Bauteils angibt.

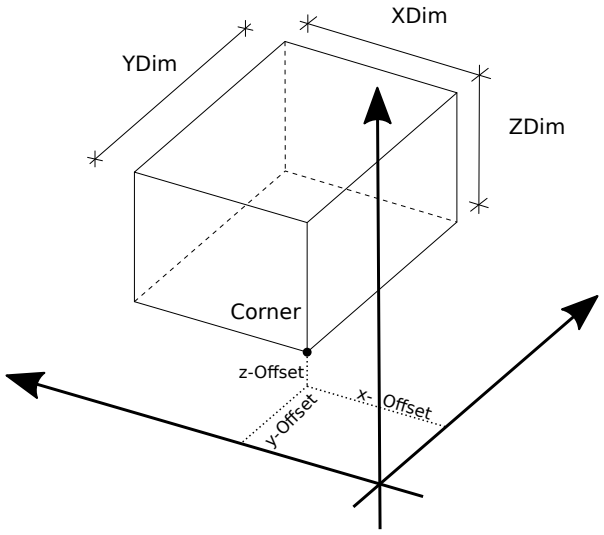



Abbildung 5.5: Geometrie der „IfcBoundingBox“, nach [18], bearbeitet

Gemäß Spezifikation wird in der IfcBoundingBox-Klasse für den „Corner“ eine Referenz zur Klasse IfcCartesianPoint und für die Abmessungen „XDim“, „YDim“ und „ZDim“ jeweils ein Attribut vom Typ IfcPositiveLengthMeasure, was eine Fließkommazahl darstellt, vorgeschrieben. An dieser Stelle ist die Bauteilgeometrie damit erstmalig mit konkreten Werten beschrieben.

Jedes Bauteil hat ein eigenes, lokales Koordinatensystem (vgl. auch Abschnitt 5.2.4). In IfcCartesianPoint wird der örtliche Versatz des ausgehenden Eckpunkts der Bounding-Box („Corner“) zum Ursprung des lokalen Koordinatensystems des Bauteils angegeben. Dabei enthält die Klasse IfcCartesianPoint drei Attribute, ebenfalls vom Typ IfcPositiveLengthMeasure, womit der Versatz in x-, y- und z-Richtung als Fließkommazahl angegeben wird. Die Bounding-Box wird parallel zu den Achsen des lokalen Koordinatensystems aufgespannt, ausgehend von dem Eckpunkt „Corner“.

Die „Bounding-Box“ wird für jedes Bauteil erzeugt[10], was nicht immer sinnvoll ist, da beispielsweise bei gekrümmten Bauteilen trotzdem ein Quader angegeben wird. Für quaderförmige Bauteile wie eine Wand, eine rechteckige Stütze oder eine Wandöffnung ist die Bounding-Box als Geometrie beschreibende Klasse ausreichend genau und wird im Softwareprototypen (siehe Kapitel 6) als Basis für den zweidimensionalen Raumgrundriss gefiltert.

### IfcGeometricCurveSet und IfcPolyline

Räume sind Körper, die durch einen Grundriss und eine Ausdehnung in die Höhe ein Volumen enthalten. Eine Beschreibung über die Bounding-Box ist zwar naheliegend und findet auch statt, ist aber nicht in allen Fällen günstig, da sie bei einem nicht-rechteckigen Grundriss die Raumgeometrie nicht ausreichend genau abbildet. Zur Grundrissdarstellung erfolgt daher neben der Referenz zur Bounding-Box eine Referenz zur Klasse IfcGeometricCurveSet, welche auf eine Liste von mindestens ein oder mehrerer zwei- oder dreidimensionaler Punkte (IfcPoint) und Kurven (IfcCurve) verweist [18]. Eine Unterklasse von IfcCurve ist IfcPolyline, die durch eine Liste mindestens zweier Punkte vom Typ IfcCartesianPoint einen Polygonzug erzeugt. In IfcCartesianPoint sind nur x- und y-Koordinaten (kein z) angegeben, da an dieser Stelle nur der Grundriss abgebildet wird. Die Koordinaten beziehen sich auf das lokale Koordinatensystem.

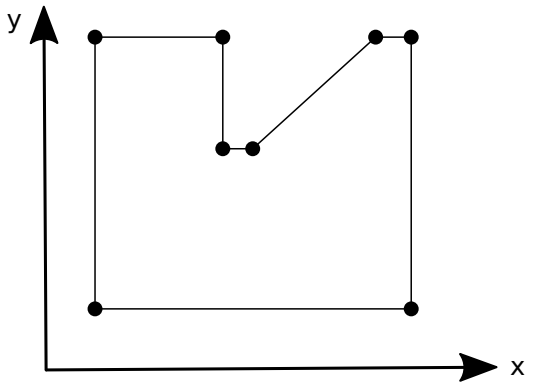



Abbildung 5.6: Geometrie einer „IfcPolyline“

Für einen geschlossenen Polygonzug ähnlich dem in Abbildung 5.6, wie er bei einem Raum erforderlich ist, muss die letzte Angabe in der Liste wieder auf den ersten Punkt referenzieren. Damit liegt der Raumgrundriss vor und kann für die Rettungswegermittlung weiter genutzt werden. Es sei noch darauf hingewiesen, dass in der IfcPolyline lediglich die Schnittpunkte von Raumwänden berücksichtigt werden, nicht jedoch Punkte an im Raum freistehenden Wänden oder Aussparungen.

### Weitere Klassen zur Geometriebeschreibung

Neben den Referenzen zu IfcBoundingBox und IfcGeometricCurveSet bestehen auch Referenzen zu weiteren geometrischen Darstellungen, da von der IfcRepresentation bzw. IfcShapeRepresentation zu einer Liste, d.h. einem oder mehreren IfcRepresentationItems referenziert wird und IfcBoundingBox bzw. IfcGeometricCurveSet nur eine grobe Beschreibung

[10] Nur in der mit ArchiCAD erzeugten IFC-Beispieldatei. In einer mit Autodesk Revit erzeugten Datei wurde die BoundingBox nicht angegeben.

der Geometrie erlauben. Auch ist es teilweise vom IFC-Übersetzer abhängig, nach welchen Klassen Objekte zur geometrischen Beschreibung erstellt werden. In der Klasse IfcShapeRepresentation sind neben der BoundingBox und dem GeometricCurveSet weitere Typen zur Formbeschreibung als vordefinierte Typen enthalten [18]. Nachfolgend wird eine Auswahl dieser kurz beschrieben.

Neben den groben Geometriebeschreibungen wie BoundingBox wird in der Liste von IfcRepresentationItems auch eine Brep-Darstellung angegeben, wodurch die exakte Geometrie eines Elements oder Bauteils wiedergegeben wird. Ausgehend von der Klasse IfcFacetBrep wird zu sehr vielen weiteren Objekten verschiedener Klassen referenziert, um die Bauteilgeometrie zu reproduzieren. Für den in dieser Arbeit entwickelten Softwareprototypen ist die Filterung der Brep-Darstellung jedoch zu komplex und hätte gemessen an dem Programmieraufwand keinen nennenswerten wissenschaftlichen Mehrwert gebracht.

Eine Geometriedarstellung, die bei dem Test eines IFC-Exports von Wänden mit Autodesk Revit 2016 gewählt wurde, ist die „SweptSolid"-Darstellung. Dort werden geschlossene, ebene Flächen erzeugt und extrudiert. Von der IfcPolyline unterscheidet sich SweptSolid dahingehend, dass hier auf vordefinierte Profile zur Beschreibung geometrischer Formen zurückgegriffen wird. Als Erweiterung dazu gibt es noch die „Advanced-SweptSolid"-Darstellung, bei der die Profile anhand einer Leitkurve geführt werden [18]. Weitere Darstellungstypen sind „Constructive solid geometry models" (CSG) und „Clipping", die beide allerdings auf erzeugte Brep- oder SweptSolid-Formen basieren und teilweise im Rahmen von IFC2x3 noch nicht berücksichtigt werden.

Darüber hinaus gibt es noch die geometrische Repräsentation über eine sogenannte „SectionedSpine". Dabei werden für ein Bauteil, das meist gegenüber seinem Querschnitt eine große Längsausdehnung hat (wie bspw. ein Lüftungskanal), an charakteristischen Stellen die Querschnitte definiert. Die Längsausdehnung (sie muss nicht zwangsweise gerade, sondern kann auch gekrümmt verlaufen) wird durch die sogenannte „spine" (engl. für Wirbelsäule) dargestellt. An den entsprechenden Stellen der spine befinden sich die Querschnitte, die insgesamt in ihrer Zusammensetzung die Form des Bauteils beschreiben.

### 5.2.4 Positionierung von Elementen

Wie im vorhergehenden Abschnitt bereits erwähnt wurde, wird die geometrische Repräsentation eines Objekts ausgehend von dessen eigenem lokalen Koordinatensystem beschrieben. Durch die Referenz auf eine Unterklasse von IfcObjectPlacement wird die Position und Ausrichtung dieses lokalen Koordinatensystems in Bezug zu einem anderen Koordinatensystem gesetzt. Darüber hinaus kann die Position des lokalen Koordinatensystems auch durch ein Raster angegeben werden, was in der IFC-Beispieldatei dieser Arbeit jedoch nicht vorkommt und hier daher nicht näher betrachtet wird.

Sofern der Bezug zu einem anderen Koordinatensystem (also kein Raster) hergestellt wird, erfolgt die Referenzierung direkt zu einem Objekt der Klasse IfcLocalPlacement. Von dieser aus wird der Versatz des eigenen Koordinatensystems entweder zu einem Koordinatensystem eines anderen Objekts oder zu dem globalen Koordinatensystem (WCS) angegeben. Die Klasse IfcLocalPlacement sieht hierfür gemäß EXPRESS-Spezifikation zwei Referenzen vor, obligatorisch ein „RelativePlacement" und optional ein „PlacementRelTo". Das RelativePlacement muss immer angegeben werden und beschreibt den Versatz zwischen eigenem und Bezugskoordinatensystem mit konkreten Werten über eine Unterklasse von IfcAxis2Placement. Wird die optionale Referenz „PlacementRelTo" nicht angegeben, handelt es sich bei diesem Versatz um den vom eigenen Koordinatensystem zum WCS.

Bei Angabe der PlacementRelTo-Referenz beschreibt das RelativePlacement ebenfalls den Versatz zwischen dem eigenen und einem Bezugskoordinatensystem, wobei es sich bei dem Bezugskoordinatensystem an dieser Stelle um das lokale Koordinatensystem eines anderen Objekts und nicht um das WCS handelt. Die PlacementRelTo-Referenz wiederum verweist auf das IfcObjectPlacement dieses anderen Objekts, um dessen Versatz zu einem weiteren Koordinatensystem zu ermitteln. Auch das andere Objekt muss nicht unbedingt direkt in Bezug zum WCS stehen, sodass es dazu kommen kann, dass es mehrerer Verweise bedarf, um den Bezug eines Objekts zum WCS zu erhalten (siehe dazu auch Abschnitt 6.4.2).

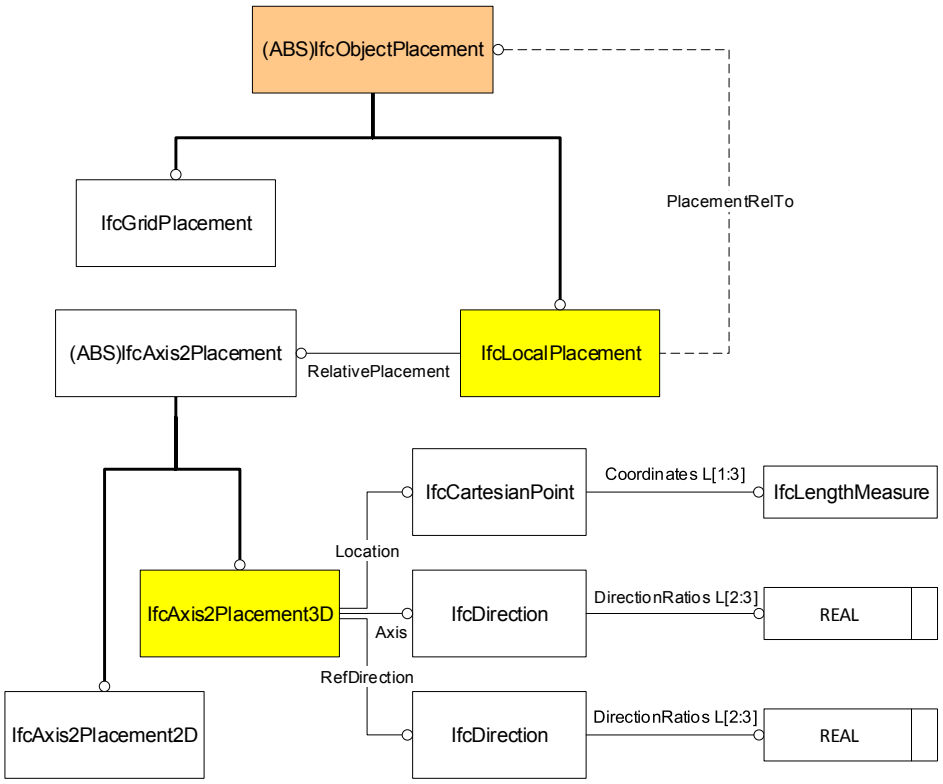



Abbildung 5.7: EXPRESS-G Darstellung der zur Bauteilplatzierung beschreibenden relevanten Klassen (unvollständige Übersicht)

In der Klasse IfcAxis2Placement3D, auf die bei dreidimensionalen Räumen von IfcLocalPlacement direkt referenziert wird, sind alle Attribute zur Beschreibung eines Versatzes spezifiziert. Mit dem Verweis auf IfcCartesianPoint wird der Versatz von dem referenzierten zum eigenen Koordinatensystem in x-, y- und z-Richtung parallel zu den entsprechenden Achsrichtungen des referenzierten Koordinatensystems abgebildet. Die Werte liegen über IfcLengthMeasure als Fließkommazahl vor.

Die „Axis“ beschreibt die Richtung bzw. Neigung der lokalen z-Achse des eigenen Koordinatensystems in Bezug zu dem referenzierten Koordinatensystem und wird über die Klasse IfcDirection repräsentiert. Diese hat zwei oder drei Attribute vom Typ REAL, welche die Richtung der Drehung (in diesem Fall die Neigung der z-Achse) als X:Y oder X:Y:Z Koordinaten angeben.

Der letzte Verweis ist die „RefDirection“, was die Drehung des eigenen Koordinatensystems in der Ebene (also um die z-Achse) in Bezug zum referenzierten Koordinatensystem angibt.

Auch hier erfolgt die Repräsentation über die Klasse IfcDirection, welche die Drehung über X:Y oder X:Y:Z Koordinaten angibt.

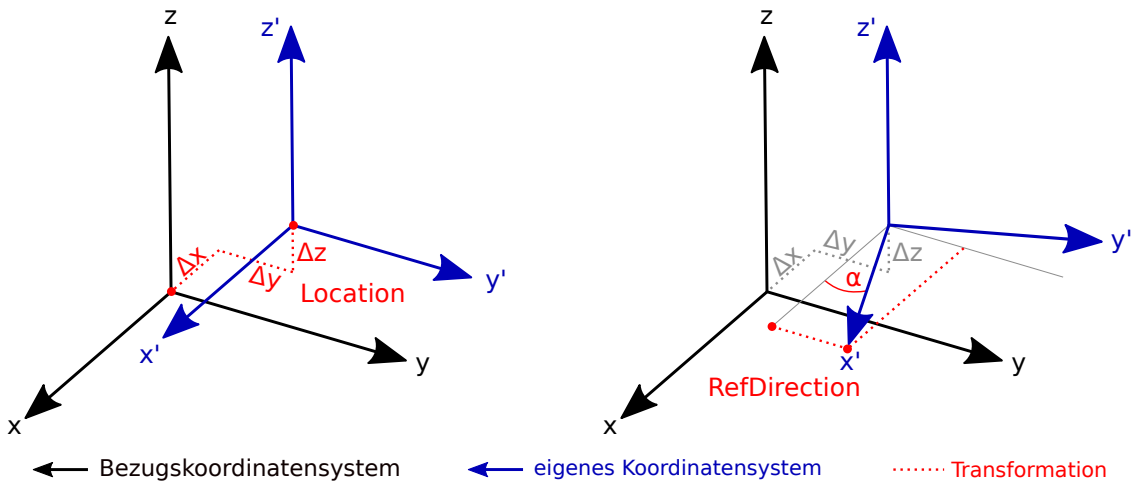



Abbildung 5.8: IfcAxis2Placement3D mit Location und RefDirection in der Ebene

Abbildung 5.8 zeigt den Zusammenhang zwischen der Location und der RefDirection in Bezug auf zwei Koordinatensysteme. Bei der RefDirection ist nur die Drehung in der Ebene (also um die z-Achse) dargestellt, da eine Neigung im Rahmen dieser Arbeit nicht betrachtet werden muss und auch in der IFC-Beispieldatei nicht vorkommt. Der Winkel $\alpha$ ergibt sich aus den Koordinaten, die in IfcDirection für die RefDirection angegeben werden.

Durch Angabe des ObjectPlacements ist es somit möglich, die Position und Richtung der lokalen Koordinatensysteme aller Objekte in Bezug zu einem globalen Koordinatensystem zu setzen. Da über die ProductRepresenation die Form der Bauteile gegeben ist, kann im Zusammenspiel mit dem ObjectPlacement das geometrische Gebäudemodell zusammengesetzt werden.

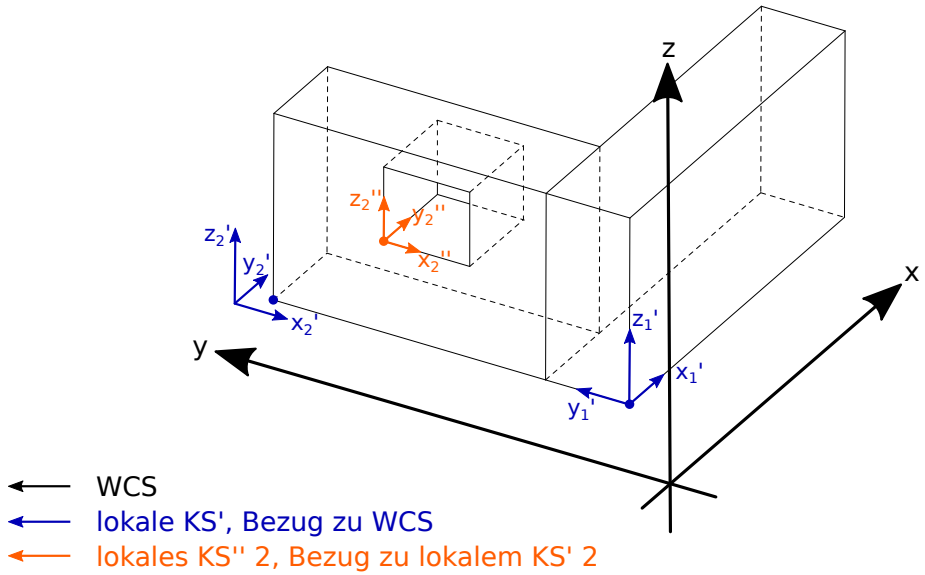



Abbildung 5.9: Zusammensetzung mehrerer Bauteile und ihre lokalen Koordinatensysteme

### 5.2.5 Bauteileigenschaften

Die IFC-Datenstruktur bietet die Möglichkeit, einem Bauteil/Objekt weitestgehend frei gewählte Eigenschaften zuzuweisen. Das können bestimmte Materialparameter wie Feuerwiderstandsfähigkeit, Wärmedurchlässigkeit oder Schalldichtigkeit sein, aber auch betriebswirtschaftliche Daten wie Kaufpreis oder Garantie.

Diese Eigenschaften für die entsprechenden Objekte werden durch die Klasse IfcPropertySet repräsentiert. Die IfcPropertySet-Klasse ist eine Unterklasse der abstrakten Klassen IfcPropertySetDefinition und IfcPropertyDefinition, wobei letztere eine Unterklasse von der bereits bekannten IfcRoot ist. Der Verweis von einem Objekt auf dessen PropertySets erfolgt jedoch nicht über eine direkte Referenz, sondern über eine Inverse Beziehung über die Klasse IfcRelDefinesByProperties. Das bedeutet, dass nur die Klasse IfcRelDefinesByProperties sowohl auf das Objekt als auch auf die IfcPropertySetDefinition referenziert, jedoch nicht umgekehrt.

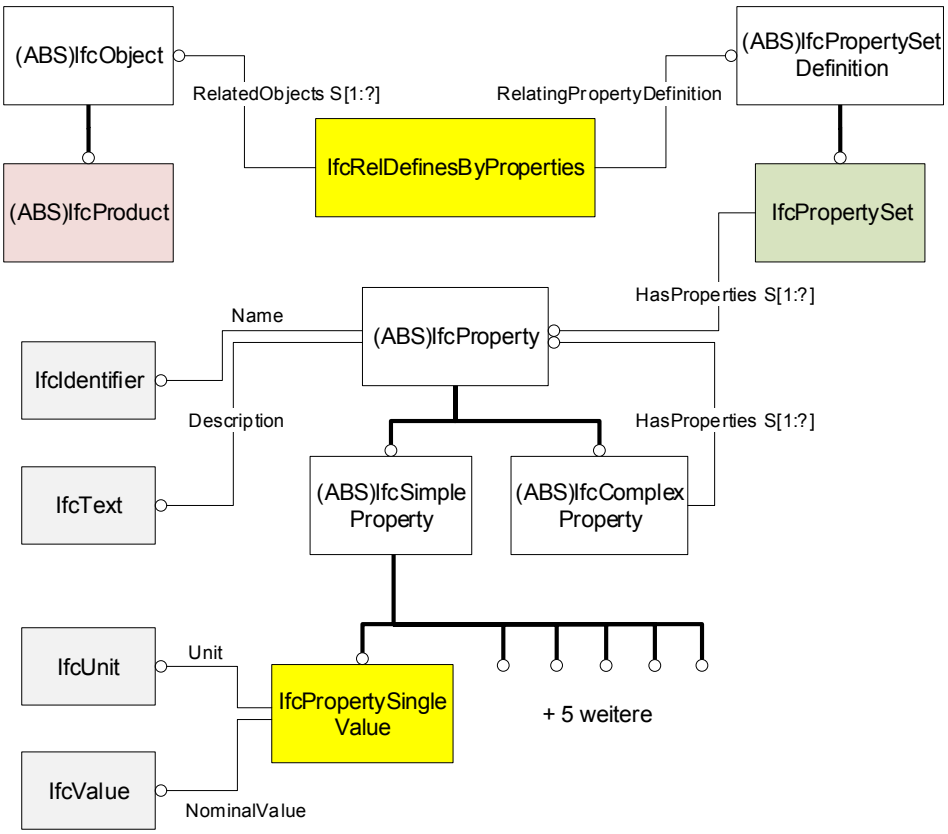



Abbildung 5.10: EXPRESS-G Darstellung der Eigenschaften zuweisenden Klassen (unvollständige Übersicht)

Die Klasse IfcPropertySet referenziert zu einem oder mehreren Unter-Objekten der abstrakten Klasse IfcProperty. Ein Bauteil/Objekt kann somit mehrere Eigenschaften haben. In IfcProperty werden bereits die Attribute *Name* und *Description* spezifiziert, wobei mit „Name“ die Bezeichnung der Objekteigenschaft, also z.B. „FireResistanceRating“ mit dem Datentyp String beschrieben wird.[11] Eine dieser Klassen, zu denen von IfcPropertySet aus

[11] Die Klasse IfcIdentifier ist immer vom Typ „String“.

hin referenziert wird ist die IfcPropertySingleValue. Neben ihr existieren fünf weitere Unterklassen vom Typen IfcProperty; sie ist aber die einzige, die in der IFC-Beispieldatei und damit im Rahmen dieser Arbeit relevant ist. Diese Klasse spezifiziert die dem Bauteil zugewiesene Eigenschaft nun über eine Einheit (IfcUnit) und über den entsprechenden Wert (IfcValue). Der Wert kann allerdings bereits in IfcUnit enthalten sein, sodass IfcValue ggf. nicht belegt sein muss, wie z.B. im nachstehenden Quellcode 5.2 gezeigt. Dort hat ein Bauteil die Eigenschaft, ein Notausgang zu sein (#19571, erkennbar am Namen „FireExit“), was über den Boolean-Wert „wahr“ (IFCBOOLEAN(.T.)) ausgedrückt ist. Die Position von IfcValue (letzte Position) bleibt dabei leer („$“).

Quellcode 5.2: Darstellung von IfcPropertySets in IFC-Datei

```
#19575=IFCPROPERTYSET('0ytT$T0FhJOVnX6MwYnChu',#12,
   'Pset_DoorCommon',$,(#19565,#19566,#19571));
#19565=IFCPROPERTYSINGLEVALUE('ThermalTransmittance',$,
   IFCTHERMALTRANSMITTANCEMEASURE(1.7),$);
#19566=IFCPROPERTYSINGLEVALUE('AcousticRating',$,IFCLABEL(''),$);
#19571=IFCPROPERTYSINGLEVALUE('FireExit',$,IFCBOOLEAN(.T.),$);

#19583=IFCPROPERTYSET('3img6R7uZkTS8qj2XbZJTP',#12,
   'Pset_FireRatingProperties',$,(#19581,#19582));
#19581=IFCPROPERTYSINGLEVALUE('FireResistanceRating',$,
   IFCLABEL('T30'),$);
```

Einem Bauteil/Objekt können mehrere PropertySets zugeordnet sein. Dabei werden die verschiedenen Eigenschaften und Parameter gemäß ihrer Bedeutung in Gruppen kategorisiert, die dann jeweils durch eine eigene Instanz von IfcPropertySet repräsentiert werden. In Quellcode 5.2 sind beispielsweise die Eigenschaften „ThermalTransmittance“, „AcousticRating“ und „FireExit“ referenzierte Instanzen zum IfcPropertySet „Pset_DoorCommon“, während der Parameter „FireResistanceRating“ zur Gruppe „Pset_FireRatingProperties“ gehört. Zu jeder einzelnen Instanz von IfcPropertySet wird ebenfalls eine eigene Instanz von IfcRelDefinesByProperties angelegt, von welcher aus entsprechend auf die IfcPropertySet-Instanz und auf das zugehörige Bauteil referenziert wird.

Es handelt sich bei all diesen Parametern und Eigenschaften jedoch nicht um durch das IFC-Schema eindeutig spezifizierte, sondern um individuell zu benennende und mit Werten zu belegende Attribute, deren Semantik sich nur aus übereinstimmenden Interpretationen des jeweiligen Attributnamens ergibt. Das bedeutet, dass dem Attribut, welches bspw. mit „Feuerwiderstandsfähigkeit“ benannt wurde und den zugehörigen Wert „F 90“ erhält, nur aufgrund allgemein anerkannter Interpretationen die Bedeutung zugeschrieben wird, die Widerstandsfähigkeit gegen Feuer mit dem genannten Wert auszudrücken. Es wäre auch möglich, die „Wärmedurchlässigkeit“ mit dem Wert „F 90“ anzugeben, was zwar gemäß der IfcPropertySingleValue-Definition korrekt wäre, semantisch jedoch keinen Sinn ergäbe. Im Gegensatz dazu ist eindeutig festgelegt, dass z.B. der Wert des Attributs „XDim“ in der Klasse IfcBoundingBox immer die Ausdehnung der Bounding Box in x-Richtung des lokalen Koordinatensystems angibt. Hierzu gibt es keinen Interpretationsspielraum, da die Bedeutung von „XDim“ durch das IFC-Schema bestimmt wird.

Durch bestimmte Namenskonventionen bei der IFC-Übersetzung kann der Interpretationsspielraum begrenzt werden, sodass die Eigenschaften von unterschiedlichen IFC-Übersetzern und Programmen gleichermaßen interpretiert und ausgewertet werden. Über die Definition der Einheit mit der Klasse IfcUnit kann zudem die Auswahl des Wertes eingegrenzt werden (z.B. dass die Einheit Boolean nicht „F 90“ sein kann), wobei die Zuordnung zur

Bezeichnung der Eigenschaft immer noch Interpretationssache bleibt. Dadurch besteht immer noch eine gewisse Datenunschärfe, weswegen die Verwendung und Verarbeitung von IfcPropertySets vorsichtig geschehen muss.

Für diese Arbeit ist es notwendig, die IfcPropertySingleValue-Angaben „FireExit“ (Datentyp Boolean) aus der IfcPropertySet-Gruppe „Pset_DoorCommon“ sowie die Angabe „OccupancyNumber“ (Datentyp Integer) aus der Gruppe „Pset_SpaceOccupancyRequirements“ zu filtern. *FireExit* wird in Zusammenhang mit Türen verwendet und gibt an, ob diese Tür ein Ausgang ins Freie oder zu einem Treppenraum ist. Türen, auf die das zutrifft, werden bei der Rettungswegermittlung als Startwert festgelegt, zu denen dann die Rettungswege ermittelt werden. Die *OccupancyNumber* gibt die Personenbelegung an und wird zusammen mit Räumen verwendet, sodass später der Personenstrom pro Rettungsweg angegeben werden kann.

### 5.2.6 Inverse Beziehungen der Elemente

Im vorherigen Abschnitt ist bereits auf inverse Beziehungen von Objekten hingewiesen worden. Es handelt sich dabei stets um Beziehungen von anderen Objekten, die eine bestimmte Bedeutung für ein Objekt/Bauteil haben, allerdings *zu* diesem Objekt hin referenziert werden. Das Bauteil selbst „kennt“ diese Beziehung daher nicht. In Quellcode 5.3 ist ein Ausschnitt aus der IFC-Beispieldatei gegeben, der eine solche inverse Beziehung zwischen einem IfcPropertySet und einer IfcWall zeigt.

Quellcode 5.3: Darstellung zweier inversen Beziehungen in einer IFC-Datei

```
#527=IFCWALL('2fvwWlWFb9z8sJVlZ0rF1P',#12,'Wand-023',$,$,#378,#522,
   'A9E7A82F-80F9-49F4-8D93-7EF8C0D4F059');
#570=IFCPROPERTYSET('2V0kVI4CZ4RyAMXrbYx2dq',#12,
   'Pset_ConcreteElementGeneral',$,(#566));
#575=IFCRELDEFINESBYPROPERTIES('0T43kkc28f5Bo4vtFy71ym',#12,$,$,
   (#527),#570);
```

Weder in IfcWall, noch in IfcPropertySet ist eine Referenz zu der jeweils anderen Klasse gegeben, sodass ein Zusammenhang zwischen der Wand und ihren Betoneigenschaften zunächst nicht gegeben ist. Dieser wird erst invers über die Klasse IfcRelDefinesByProperties hergestellt, da von dieser aus zu den beiden Klassen referenziert wird. Dass eine Klasse inverse Beziehungen hat, geht aus der IFC-Dokumentation [18] hervor (siehe dazu auch die EXPRESS-Darstellungen ausgewählter Klassen in Anhang A. Somit ist bei der Filterung von Daten zu beachten, dass nach einer weiteren, referenzierenden Klasse gesucht werden muss, wenn bestimmte Informationen zu einem Bauteil benötigt werden.

Um die relevanten Daten für die Rettungswegermittlung bereitstellen zu können, ist die Filterung einiger Klassen erforderlich, deren Zusammenhang über inverse Beziehungen hergestellt werden. Konkret betrifft dies bestimmte Informationen, die zu einer Tür (IfcDoor) gehören. So werden die unterschiedlichen Angaben, ob es sich um einen Notausgang (IfcPropertySet) handelt, welche Abmessungen die Öffnung hat (IfcOpeningElement) und in welcher Wand sich die Öffnung befindet, benötigt. Untereinander sind diese Klassen nicht referenziert, sodass ein Bezug nur über inverse Beziehungen möglich ist. Hierfür wird von drei weiteren Klassen, welche im IFC-Schema ausschließlich den Zweck der Verbindung von Elementen erfüllen, auf jeweils zwei der Klassen referenziert. Die für diesen Anwendungsfall erforderlichen „Hilfsklassen„ sind IfcRelDefinesByProperties, IfcRelFillsElement und IfcRelVoidsElement.

Die Klasse IfcRelDefinesByProperties ist in Abschnitt 5.2.5 bereits vorgestellt worden. Sie referenziert zu einem oder mehreren Unterobjekten von IfcObject (hier IfcDoor) und zu einem Unterobjekt von IfcPropertySetDefinition (hier IfcPropertySet). IfcRelFillsElement stellt die Beziehung zwischen einem oder mehreren Unterobjekten von IfcElement (hier IfcDoor) und einem Objekt von IfcOpeningElement her. Die Klasse IfcRelVoidsElement beinhaltet ein *RelatedOpeningElement* aus der Oberklasse IfcFeatureElementSubtraction (hier IfcOpeningElement) und ein *RelatingBuildingElement* der Oberklasse IfcElement (hier IfcWall).

Wie in Abbildung 5.11 zu sehen ist, stehen dadurch alle Klassen, welche für die Rettungswegermittlung relevante Informationen zu einer Türöffnung liefern, in Beziehung zueinander. Zur besseren Übersichtlichkeit ist in dieser Abbildung auf die Darstellung der Vererbungshierarchie verzichtet worden. Die dort gezeigten Referenzen direkt zu den entsprechenden Klassen sind aufgrund der Zuweisungskompatibilität möglich.

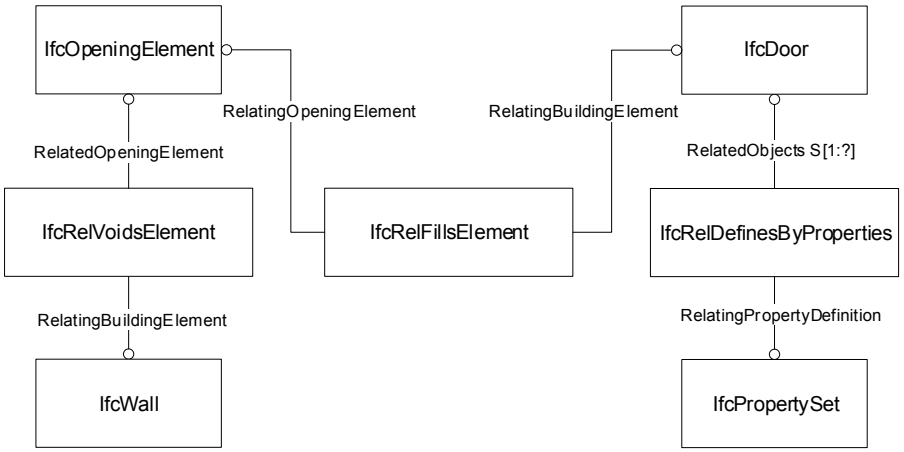



Abbildung 5.11: EXPRESS-G Darstellung der inversen Beziehungen einer Türöffnung (Übersicht ohne Vererbung)

Neben den inversen Beziehungen bezüglich Türen werden für den Softwareprototypen Informationen aus einer inversen Beziehung zwischen Räumen (IfcSpace) und der Personenbelegung je Raum (IfcPropertySet) benötigt. Auch diese Beziehung wird über die Klasse IfcRelDefinesByProperties hergestellt, da es sich bei IfcSpace um eine Unterklasse von IfcObject handelt.

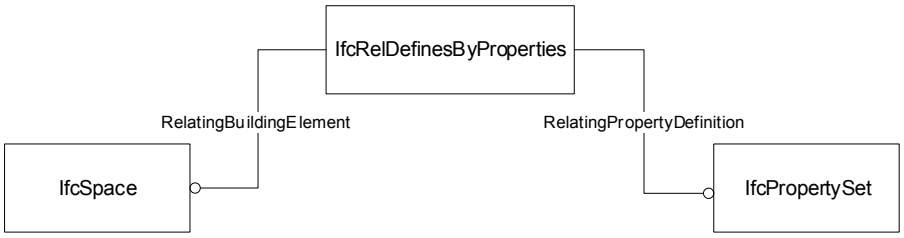



Abbildung 5.12: EXPRESS-G Darstellung der inversen Beziehungen eines Raums (Übersicht ohne Vererbung)

# 6 Implementierung in den Softwareprototypen

## 6.1 Leistungsumfang des Softwareprototypen

Der in Kapitel 4 entwickelte Algorithmus sowie die Filterung der in Kapitel 5 beschriebenen Objekte werden in einem Softwareprototypen implementiert, um die theoretische Methodik zu validieren. Mit diesem soll der Transfer von IFC-Daten in ein eigenes Modell, die Ermittlung von Rettungswegen sowie die anschließende Aufbereitung der Ergebnisse gelingen.

Die Zielsetzung dieser Arbeit ist ein BIM-gestützter Rettungswegnachweis, welcher die vorhandene maximale Rettungsweglänge jedes Aufenthaltsraums sowie die erforderliche Rettungswegbreite je Personenstrom ausgibt. In den Abschnitten 2.1.2f ist bereits darauf hingewiesen worden, dass in der Musterbauordnung keine konkreten Werte für Rettungswegbreiten vorgeschrieben werden, sondern nur in Sonderbauvorschriften. Allerdings übersteigt eine automatisierte Überprüfung des Sonderbaustatus den Umfang dieser Arbeit erheblich, weswegen ein Abgleich von Rettungswegbreiten mit bauordnungsrechtlichen Vorschriften in diesem Softwareprototypen nicht implementiert wird. Es erfolgt jedoch eine Zuordnung von Personenströmen zu Rettungswegen, sodass eine nachträgliche händische Bemessung ermöglicht wird. Somit werden von dem entwickelten Softwareprototypen Rettungsweglängen, die Zuordnung zu Rettungswegen (erster und zweiter) sowie Personenströme je Rettungsweg ermittelt. Die Ermittlung von Rettungswegen geschieht geschossweise, da gemäß Musterbauordnung „in jedem Geschoss mindestens zwei voneinander unabhängige Rettungswege“ [5] vorhanden sein müssen. Der Softwareprototyp soll in der Lage sein, Daten aus mehrgeschossigen Gebäuden zu filtern und die Geschosse separat zu betrachten.

Rettungswege müssen zu notwendigen Treppenräumen oder Ausgängen ins Freie geführt werden. Diese beiden Möglichkeiten sollen dementsprechend auch als Fluchtziele im Softwareprototypen dienen, zu denen die automatisch ermittelten Rettungswege führen. Damit sich eine Tür, die Ziel eines Rettungswegs ist, von den übrigen Türen des Gebäudes unterscheidet, muss diese entsprechend gekennzeichnet werden. Eine automatische Ermittlung dieser Türen wäre mittels der Klasse IfcRelSpaceBoundary realisierbar. Diese Klasse referenziert zu einer Instanz von IfcSpace sowie zu allen umfassenden Bauteilen, sodass ein logischer Zusammenhang zwischen dem Raum und seinen angrenzenden Bauteilen entsteht. Wird dieser Raum als Treppenraum definiert, können dann über IfcRelSpaceBoundary die zugehörigen Türen ermittelt und als Fluchtziele deklariert werden. Der IFC-Export aus CAD-Programmen wie Archicad funktioniert derzeit allerdings noch nicht zuverlässig [13][12], weshalb die Definition von Notausgängen in diesem Softwareprototypen manuell erfolgt. Dafür wird im CAD-Programm an den entsprechenden Türen die Eigenschaft „Notausgang“ positiv gesetzt[13], was in der IFC-Datei durch ein IfcPropertySet bzw. IfcPropertySingleValue „FireExit“ repräsentiert wird. Da die Definition über IfcPropertySets keine

[12] Bei dem Export des Beispielgebäudes aus Archicad 21 in eine IFC-Datei konnte ebenfalls keine Zuordnung zwischen Räumen und Umfassungsbauteilen erzeugt werden.

[13] Archicad 21 bietet eine Reihe an vordefinierten Eigenschaften für Türen an, die als IfcPropertySet exportiert werden.

semantisch eindeutige Lösung ist (siehe dazu Abschnitt 5.2.5), wird dieser Ansatz nur als Übergangslösung betrachtet.

Aufgrund des begrenzten Umfangs dieser Arbeit stellt die entwickelte Software kein vollwertiges Programm, sondern nur einen Prototypen dar. Mit diesem soll die prinzipielle Machbarkeit einer BIM-basierten Rettungswegermittlung nachgewiesen werden; es besteht kein Anspruch an ein praxistaugliches Programm. Dementsprechend werden bestimmte Elemente eines Bauwerks ausgeschlossen, welche für die Machbarkeitsstudie keine signifikante Bedeutung haben. So werden nur gerade Wände und rechteckige Stützen betrachtet, da runde oder gekrümmte Elemente einen erhöhten Arbeitsaufwand hinsichtlich der Verarbeitung bedeuten. Diagonal verlaufende Wände sind möglich, es werden jedoch keine Türen in diese Wände eingebaut. Dies brachte in Testdurchläufen Fehler hervor, die aus Rechenungenauigkeiten des Compilers resultieren und somit keine prinzipbedingten Fehler darstellen. Eine Behebung dieses Fehlers wäre im zeitlichen Rahmen dieser Arbeit nicht möglich gewesen, sodass Türen in diagonal verlaufenden Wänden nicht betrachtet werden. Des weiteren wird immer nur ein einzelnes, zusammenhängendes Gebäude erstellt, da die verwendete Java-Methode zur Erzeugung des Gebäudeaußenpolygons bei mehreren nicht ineinander liegenden Polygonen inkonsistente Ergebnisse liefert.

## 6.2 Auswahl der Programmiersprache

Die Umsetzung des Softwareprototypen erfolgt in der objektorientierten Programmiersprache Java, welche von der Firma Sun Microsystems, einem Tochterunternehmen von Oracle, entwickelt wurde. Anwendungen, die in Java programmiert sind, werden in der Java-Laufzeitumgebung (engl. Java Runtime Environment, JRE) ausgeführt, welche wiederum auf dem PC-Betriebssystem installiert ist. Das JRE liefert eine virtuelle Maschine und ermöglicht so ein betriebssystemunabhängiges Ausführen von Java-Anwendungen. Die virtuelle Maschine übernimmt die Speicherallocation, sodass dies nicht durch den Programmierer vorgenommen werden muss.

Die Vorteile von Java bestehen in der Einfachheit gegenüber anderen objektorientierten Programmiersprachen bei gleichzeitiger Leistungsfähigkeit. Zudem ist Java so strukturiert, dass unerwünschte Systemfehler durch die Architektur vermieden werden. Für die Arbeit mit dem IFC-Informationsmodell eignet sich eine objektorientierte Programmiersprache insofern gut, als dass das IFC-Schema ebenfalls objektorientiert aufgebaut ist und sich daher gut durch eine solche Programmiersprache verarbeiten lässt.

Um aus einer IFC-Datei das Bauwerksmodell auszulesen und als Java-Informationsmodell in eine Anwendung einzubinden, ist die Verwendung einer zusätzlichen Schnittstelle günstig. Ein reines Auslesen und Weiterverarbeiten der Testzeilen aus einer IFC-Datei wäre zwar machbar, aber aus objektorientierter Sicht nicht empfehlenswert. Die ausgelesenen Elemente müssten alle manuell in Objekte umgewandelt werden, was nicht nur einen enormen Arbeitsaufwand, sondern auch ein hohes Fehlerpotential mit sich bringt. Um diese Probleme zu vermeiden, wurden bereits entsprechende Schnittstellen entwickelt. In diesen ist das IFC-Schema als Klassenstruktur in einer Java-Bibliothek hinterlegt, sodass beim Einlesen der IFC-Datei aus den vorhandenen Elementen direkt Java-Objekte erzeugt werden. Die in diesem Softwareprototypen verwendete „IFC Java Toolbox“ wurde von der Firma Apstex Tauscher und Theiler GbR aus Weimar entwickelt und bietet neben den vorgenannten Funktionen auch eine gute Dokumentation zu ihren Klassen und Methoden.

## 6.3 Datenstruktur zum geschossweisen Filtern

Die in der Musterbauordnung vorgeschriebene maximale Rettungsweglänge von einem Aufenthaltsraum zu einem notwendigen Treppenraum oder einem Ausgang ins Freie bezieht sich ausschließlich auf horizontale Rettungswege; vertikale Rettungswege sind die Treppenräume selbst, die als sicherer Bereich gelten und daher keine Längenbeschränkung haben. Infolgedessen kann die Rettungswegermittlung in einem Gebäude immer geschossweise erfolgen, da für jeden Grundriss in jedem Geschoss die Rettungswege nachgewiesen werden müssen.

Dementsprechend müssen aus dem IFC-Gebäudemodell die Elemente eines Grundrisses (Wände, Türen) geschossweise ausgelesen und zur weiteren Verarbeitung aufbereitet werden. An dieser Stelle gibt es unterschiedliche Datenstrukturen, nach denen die Daten bereitgestellt werden und wie der Softwareprototyp mit diesen umzugehen hat. Nachfolgend werden zwei Ansätze diskutiert, nach denen die Daten für die Anwendung der Rettungswegermittlung strukturiert werden können.

### 6.3.1 Topologische Raumbetrachtung

In Kapitel 4 wurde die Rettungswegermittlung anhand eines einzelnen Raums vorgestellt. Der Rettungsweg führt dabei von dem am weitesten entfernt gelegenen Punkt bis zum Ausgang aus dem Raum. Bei Betrachtung des gesamten Geschosses muss der Rettungsweg von dieser Tür allerdings noch bis zur Tür ins Freie bzw. zur Tür eines notwendigen Treppenraums weiterführen. Bei der topologischen Raumbetrachtung werden die kürzesten Pfade für jeden Raum separat ermittelt und zu einer Gesamtstrecke bis zum notwendigen Treppenraum bzw. dem Ausgang ins Freie zusammengesetzt.

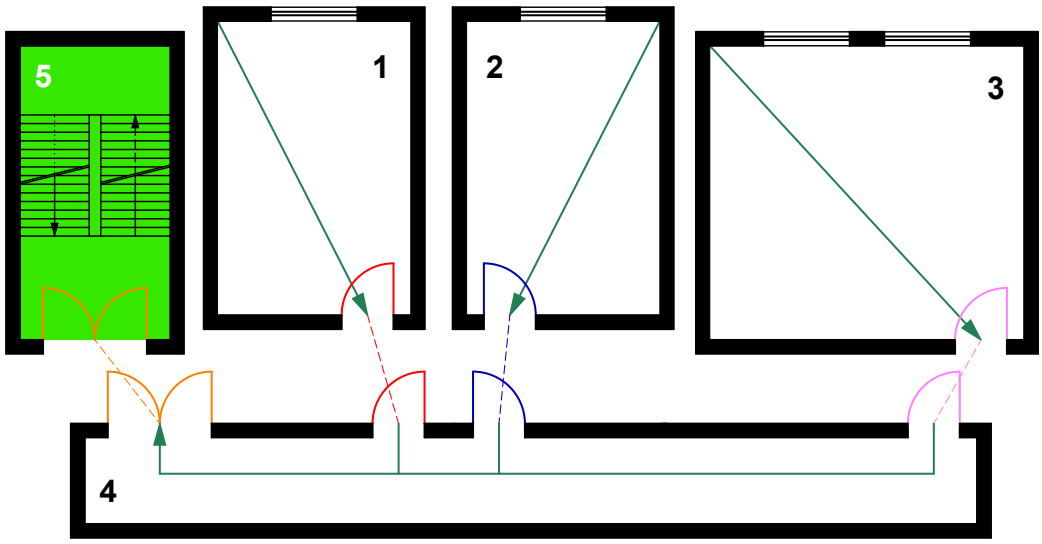



Abbildung 6.1: Topologische Raumbeziehungen

Wie in Abbildung 6.1 zu erkennen ist, müssen die Rettungswege, welche aus benachbarten Räumen (1 – 3) zu einem Raum (4) führen, über diesen zum notwendigen Treppenraum (5) fortgesetzt werden. Die geometrische Lage der verschiedenen Räume zueinander ist dabei nicht von Bedeutung, da die Verbindungen über die Türöffnungen hergestellt werden. Dies bedeutet aber auch, dass für jeden Raum die Türobjekte bekannt sein müssen, sodass

automatisiert geprüft werden kann, welche Räume durch dieselbe Tür miteinander verbunden sind. Nur dann können auch die entsprechenden Rettungswege der Räume korrekt fortgesetzt werden.

Hierbei offenbart sich allerdings auch die größte Herausforderung der topologischen Raumbetrachtung, da neben der Rettungswegermittlung selbst zusätzlich noch die Zuordnung algorithmisch stattfinden muss. Möglich wäre das unter Verwendung der in Abschnitt 6.1 bereits erläuterten Klasse IfcRelSpaceBoundary, welche die Beziehungen zwischen Räumen (IfcSpace) und den anliegenden Bauteilen herstellt.

Eine zweite Problemstellung dieses Ansatzes ergibt sich, sobald es in jedem Raum mehrere Türen gibt. In diesen Fällen „weiß“ der Algorithmus noch nicht, zu welcher Tür der Rettungsweg geführt werden muss, um in Summe den kürzesten Pfad zu ergeben. Außerdem muss die Option beachtet werden, dass die Strecke zwischen zwei Türen innerhalb desselben Raums Teil eines Rettungsweges aus einem anderen Raum sein kann.

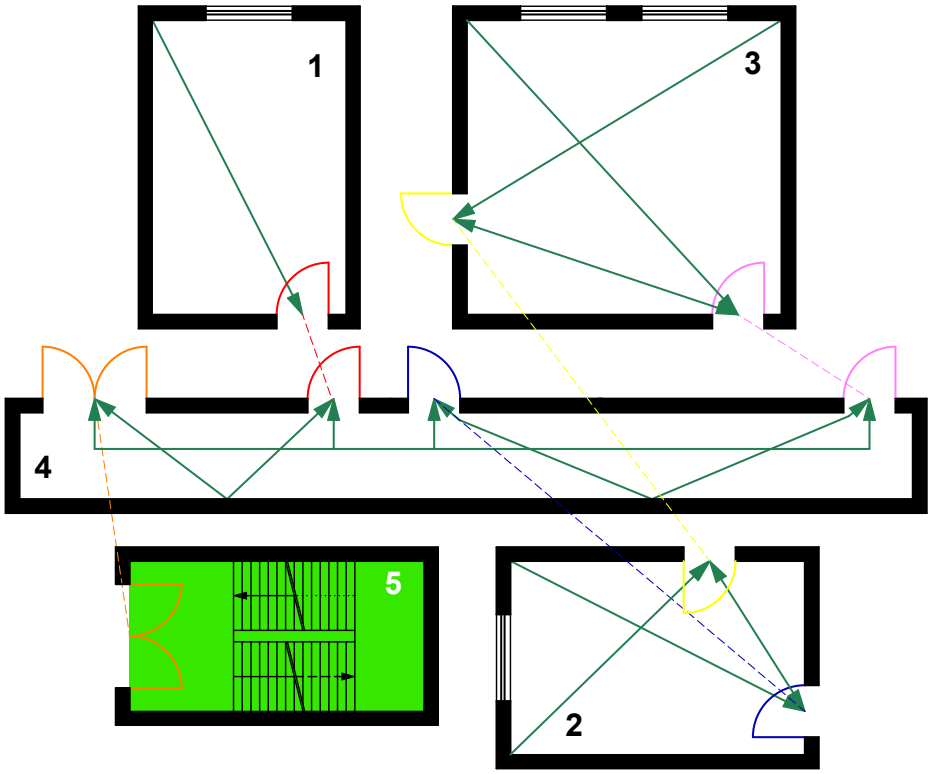



Abbildung 6.2: Komplexe topologische Raumbeziehungen

Während Abbildung 6.1 die topologischen Raumbeziehungen und Rettungswege sehr idealisiert darstellt, zeigt Abbildung 6.2 alle Strecken zwischen Türen innerhalb desselben Raums sowie die maximalen Entfernungen zu den jeweiligen Ausgängen der Räume. Für den Algorithmus ist die geometrische Lage der Räume unbedeutend, da nur wichtig ist, dass dieselben Türöffnungen der Räume miteinander verknüpft sind.

In diesem Minimalbeispiel ist zu erkennen, dass es sich bei den topologischen Raumbeziehungen um sehr komplexe Strukturen handelt, die einer aufwendigen Softwareimplementierung bedürfen. Da derzeit außerdem Exportschwierigkeiten in die IfcRelSpaceBoundary-Klasse (siehe Abschnitt 6.1) bestehen, ist zur Rettungswegermittlung ein alternativer Ansatz zu verfolgen.

### 6.3.2 Vollständige Geschossbetrachtung

Im Gegensatz zur topologischen Geschossbetrachtung wird das Geschoss als ein Raum mit vielen Wänden betrachtet. Die Türen werden dabei nicht berücksichtigt, wobei die Türöffnungen als Öffnungen in den Wänden entsprechend vorhanden sein müssen. Es wird dann, ausgehend vom Notausgang, zu jedem Knoten des Rasters der kürzeste Pfad ermittelt, womit das gesamte Geschoss abgedeckt ist. Beispielhaft sind in Abbildung 6.3 die Rettungswege zu den Räumen 1 – 3 dargestellt.

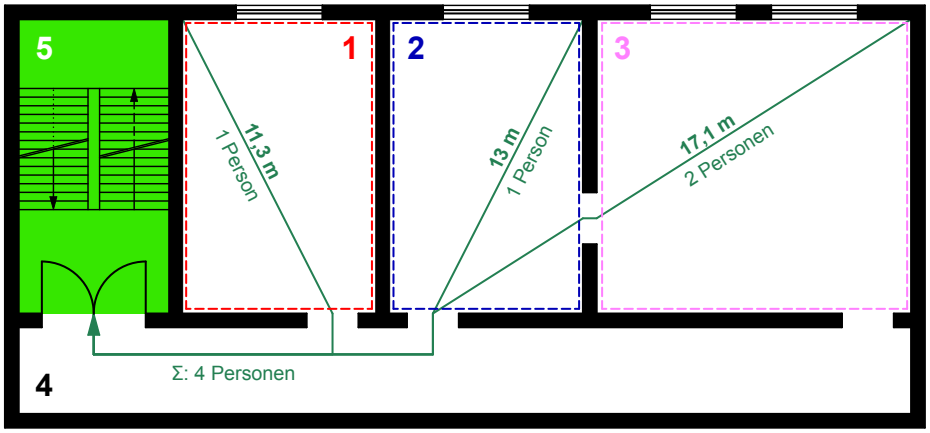



Abbildung 6.3: Rettungswege bei vollständiger Geschossbetrachtung

Prinzipiell kann der Pfad zu jedem Rasterknoten auch ausgegeben werden, wobei für den Rettungswegnachweis nur die maximale Rettungsweglänge ausschlaggebend ist. Ist diese nachgewiesen (17,1 m $<$ 35 m in Abb. 6.3), weist dies auch die anderen Rettungswege nach, da diese in jedem Fall kürzer sind (11,3 m $<$ 13 m $<$ 17,1 m).

Die Zuordnung der Rasterknoten zu den jeweiligen Räumen ist mithilfe der IfcSpace-Klasse möglich. Dort wird ein geschlossenes Polygon entsprechend der Raumfläche angegeben, was annäherungsweise in Abbildung 6.3 durch die gestrichelten Linien dargestellt wird. Mithilfe des *Punkt-in-Polygon-Test nach Jordan* (siehe Abschnitt 4.3.1) kann die Zugehörigkeit eines Knotens zu einem Raum überprüft werden. Durch die Verknüpfung der Räume mit den beinhalteten Knoten ist es möglich, die maximale Rettungsweglänge für jeden einzelnen Raum auszugeben. Darüber hinaus kann dann auch die Personenbelegung der Räume zu einem Personenstrom summiert werden, der sich für einen Notausgang ergibt.

Damit erfüllt der Ansatz der vollständigen Geschossbetrachtung alle Anforderungen, die im Rahmen der BIM-basierten Rettungswegermittlung an den Softwareprototypen gestellt werden. Da die vollständige Geschossbetrachtung außerdem gegenüber der topologischen Raumbetrachtung den geringeren Implementierungsaufwand bietet, wird sie als der geeignetere Ansatz angesehen und dementsprechend auch in dem Softwareprototypen angewendet.

## 6.4 Programmablauf des Softwareprototypen

### 6.4.1 Übersicht über den gesamten Programmablauf

Der Programmablauf des Softwareprototypen gliedert sich grundsätzlich in vier Teile, die in den nachfolgenden Abschnitten detaillierter erläutert werden. Einleitend wird an dieser Stelle ein zusammenfassender Überblick über den gesamten Programmablauf gegeben. Im ersten Schritt wird die IFC-Datei eingelesen und durch das Apstex Framework in die Java-Anwendung übertragen. Das Gebäudemodell liegt dann als Java-Objekt in der Struktur des IFC-Schemas vor. An dieser Stelle werden die benötigten Elemente gefiltert und für die Weiterverwendung aufbereitet. So müssen beispielsweise die Geometriedaten der Wände in Koordinaten umgewandelt und diese dann in das globale Koordinatensystem transformiert werden.

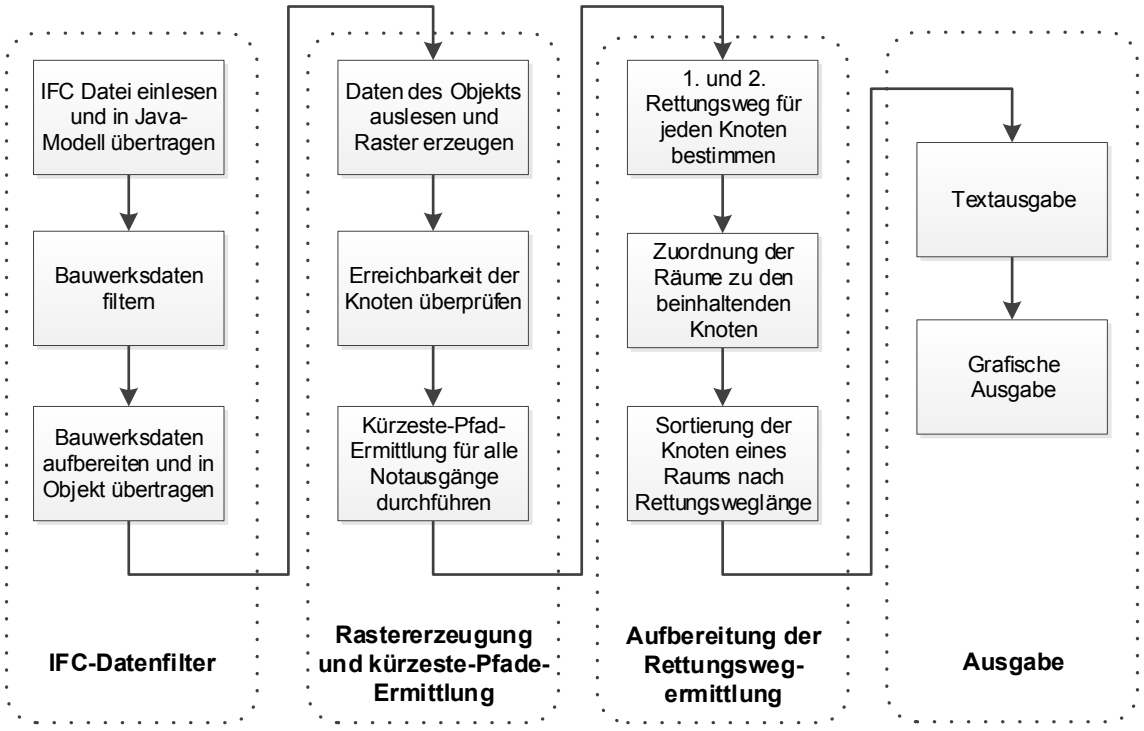



Abbildung 6.4: Übersicht über den Programmablauf des Softwareprototypen

Im darauffolgenden Schritt werden die Rasterknoten erzeugt und die potentiellen Wegstrecken zwischen den Knoten ermittelt. Dies geschieht nach dem in Abschnitt 4.3.2 vorgestellten Verfahren. Anschließend erfolgt die kürzeste-Pfade-Berechnung unter Verwendung des Dijkstra-Algorithmus (Abschnitt 4.2.1). Die jeweiligen kürzesten Pfade der einzelnen Knoten zu den Notausgängen werden in den Knoten selbst gespeichert, sodass das Raster als Objekt zur Weiterverarbeitung an den nächsten Programmteil übergeben werden kann.

Im dritten Teil werden die Daten aus der kürzesten-Pfad-Berechnung aufbereitet. Dies beinhaltet die Sortierung der möglichen Rettungswege anhand ihrer Länge, um den ersten und ggf. zweiten Rettungsweg zu bestimmen. Ebenfalls erfolgt an dieser Stelle die Zuordnung der Rasterknoten zu den Räumen, in denen sie liegen. Darauf basierend kann für jeden Raum der Knoten mit dem längsten ersten Rettungsweg ermittelt werden, welcher für den Rettungswegnachweis ausschlaggebend ist.

Der letzte Programmteil behandelt die Ausgabe der ermittelten Daten. In diesem werden die aussagekräftigen Daten aus den Knoten und Raumobjekten, in denen alle Informationen hinterlegt sind, zur Ausgabe aufbereitet. Die objektorientierte Struktur der ermittelten Daten ermöglicht es, verschiedene Informationen entsprechend der gewünschten Ausgabe zu filtern. Aus diesem Grund ist dieser Prozess als eigener Programmteil beschrieben. Die gefilterten Informationen und Werte werden in Textform sowie grafisch ausgegeben.

### 6.4.2 Struktur und Ablauf des IFC-Datenfilters

#### Objekte zur internen Datenübergabe

Um die erforderlichen Informationen für die Rettungswegermittlung zu erhalten, müssen diese aus dem IFC-Bauwerksmodell gefiltert werden.[14] Zur Vorbereitung des Einlesens der IFC-Daten werden zunächst einige Objekte erzeugt, in welchen die gefilterten Daten zur Übergabe an den nächsten Programmteil zwischengespeichert werden. Diese Objekte sind Instanzen von selbstdefinierten Klassen, die eine für diesen Anwendungsfall passende Datenstruktur vorgeben.

Die Informationen aller Bauteile, deren geometrische Repräsentation über die IfcBounding-Box ausgedrückt wird, werden in Instanzen der Klasse *PlanElement* gespeichert. Somit besteht für alle konstruktiven (Wände, Stützen) und subtraktiven Elemente (Türöffnungen mit zusätzlichen Angaben zu referenzierten Wänden und Notausgangsinformationen) dasselbe Grundgerüst, was optional um Informationen wie referenzierte Bauteile oder die Existenz von Notausgängen ergänzt werden kann. Abbildung B.1 in Anhang B.1 zeigt eine vollständige Übersicht aller Attribute und deren Datentypen sowie Objektmethoden der Klasse PlanElement.

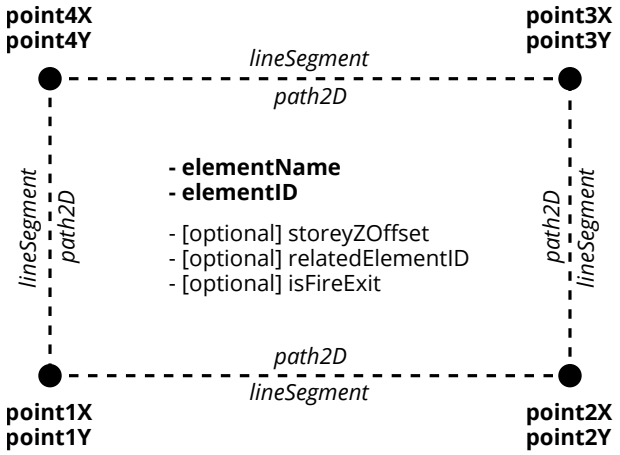



Abbildung 6.5: Veranschaulichung der von der Klasse PlanElement definierten Attribute

Die Abbildung 6.5 zeigt alle obligatorischen und optionalen Attribute von PlanElement in einer grafischen Veranschaulichung. Die vier Eckpunkte (insgesamt acht Fließkommazahl-Attribute) sind Koordinaten und geben den Grundriss eines Elements wieder. Die Pfade *path2D* und *lineSegment* zwischen diesen Punkten werden vom Konstruktor der Klasse PlanElement automatisch erzeugt. Bei den *lineSegments* handelt es sich um Instanzen der Klasse LineSegment, in welcher aus Koordinaten Vektoren erzeugt werden.[15] Path2D ist

[14] Zu den erforderlichen Bauteilen und deren zugehörigen IFC-Klassen siehe Abschnitt 5.1.
[15] Die Vektoren werden im späteren Programmverlauf für die Schnittpunktberechnung benötigt.

eine Klasse aus der Java-Bibliothek java.awt.geom und erzeugt ebenfalls zweidimensionale Pfade, welche sich zu einem geschlossenen Polygon verbinden lassen. Durch Anwendung von Objektmethoden kann mithilfe dieser Klasse der Punkt-in-Polygon-Test erfolgen. Der *elementName* gibt die Bezeichnung des jeweiligen Elements an, *elementID* die globale ID, welche dem Element vom IFC-Übersetzter zugeschrieben worden ist.

Zu diesen obligatorischen Angaben, die alle Elemente[16] haben müssen, können zur Berücksichtigung bauteilspezifischer Besonderheiten optionale Attribute in dem PlanElement gesetzt werden. Dazu gehört das *storeyZOffset* [17], was den Versatz des lokalen Koordinatensystems des Geschosses, in welchem das Element liegt, zum globalen Koordinatensystem WCS in z-Richtung angibt (Höhenlage). Über diese Angabe erfolgt im weiteren Programmablauf die Zuordnung zu den Geschossen. Die *relatedElementID* speichert bei IfcOpening-Elements die globale ID der zugehörigen Wand. Damit werden später die Wandöffnungen für Türen im Grundriss erzeugt. Das Boolean-Attribut *isFireExit* bezieht sich ebenfalls auf IfcOpeningElements und gibt an, ob es sich bei einer Tür um einen Notausgang handelt. Damit werden bei der Rettungswegermittlung die Ausgangsknoten für den kürzesten-Pfad-Algorithmus definiert.

Das gemeinsame Grundgerüst für diese Elemente ist deswegen möglich, da ihre geometrische Repräsentation über die BoundingBox gleich ist. Ebenso haben sie alle eine globale ID sowie eine Bezeichnung. Die optionalen Attribute müssen nicht mit Werten belegt, sondern können leer gelassen werden, was sich auf den weiteren Programmablauf nicht auswirkt. In der professionellen Softwareentwicklung würde *PlanElement* eine abstrakte Klasse darstellen; die optionalen Attribute für die unterschiedlichen Elemente[18] würden in weiteren, von PlanElement erbenden zusätzlichen Klassen, definiert werden. Dabei geht es im Wesentlichen um einen guten Programmierstil, da es eine flexiblere Nutzung dieser Klassen erlaubt. Für den Softwareprototypen im Rahmen dieser Arbeit ist die Definition in einer Klasse ausreichend.

Um die programminterne Handhabung der Vielzahl an PlanElement-Instanzen zu vereinfachen, werden diese in einem Objekt der Klasse *Floor* gespeichert (ebenfalls dargestellt in Abbildung B.1 in Anhang B.1).

Neben PlanElement gibt es mit *SingleRoom* eine weitere selbstdefinierte Klasse, welche die Datenstruktur für die Verarbeitung der Informationen zu Räumen definiert. Im Gegensatz zu den zuvor betrachteten Bauteilen wird die Geometrie von Räumen, hier im Besonderen die Grundfläche, durch ein Polygon mit einer variablen Anzahl an Koordinaten dargestellt. Dementsprechend muss die Klasse SingleRoom ein Attribut beschreiben, welches aus einer ArrayList[19] vom Typen „Point“ besteht. *Point* ist wiederum eine eigene Klasse, die zwei Fließkommazahlen (zur Repräsentation einer x- und y-Koordinate) beschreibt. Somit kann in einer Instanz von SingleRoom eine beliebige Anzahl an xy-Koordinaten gespeichert werden, welche zusammen ein Raumpolygon ergeben.

Ähnlich wie bei den Attributen aus PlanElement definiert SingleRoom die Attribute *roomId*, *roomName* und *roomLongName*. RoomId beschreibt die globale ID eines Raums, roomName die Raumnummer und roomLongName die Raumbezeichnung. Ebenfalls wird mit

[16] Wände, Stützen, Öffnungen.

[17] Das Attribut storeyZOffset wird an dieser Stelle als optional bezeichnet, da eine Angabe nicht durch den Konstruktor gefordert wird. Tatsächlich wird in dem Softwareprototypen für jedes Element das storeyZOffset-Attribut gesetzt.

[18] Wände, Stützen, Öffnungen.

[19] „Dynamisches“ Array, dessen Dimension nicht vorab angegeben werden muss, sondern sich variabel verändern kann.

dem Attribut *storeyZOffset* der Höhenversatz zwischen dem WCS und dem lokalen Koordinatensystem des Bezugsgeschosses angegeben. Darüber hinaus wird das Attribut *occupancy* definiert, womit die Personenbelegung des entsprechenden Raums beschrieben wird. Diese Attribute werden für die jeweiligen Instanzen während des IFC-Datenfilterprozesses gesetzt. Das Attribut *containedVertices* besteht aus einer ArrayList vom Typen *Vertex* und speichert damit alle Rasterknoten, welche innerhalb des Raumpolygons liegen. Das Raumpolygon wird allerdings erst im Programmteil „Aufbereitung der Rettungswegermittlung“ erzeugt, sodass die Zuordnung der Räume und das Setzen dieses Attributs auch erst in diesem Programmabschnitt erfolgt.

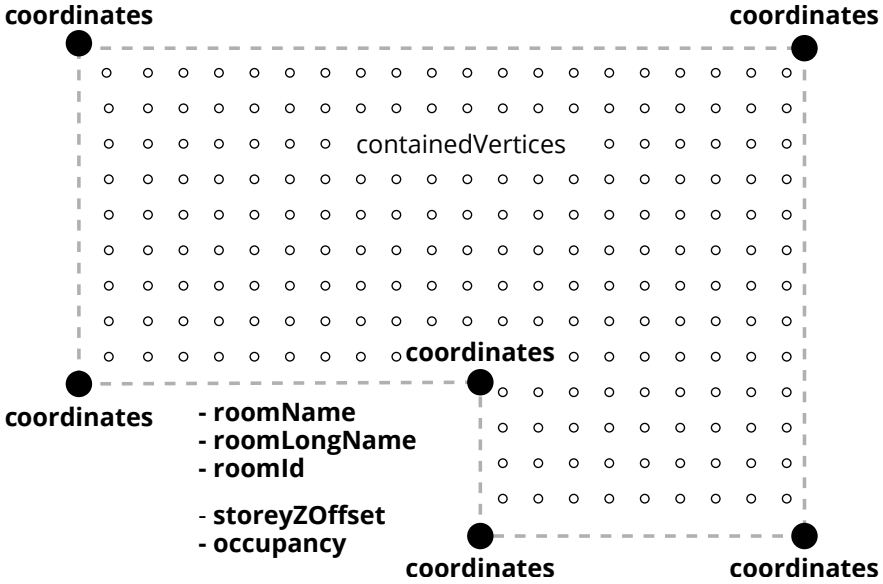



Abbildung 6.6: Veranschaulichung der von der Klasse *SingleRoom* definierten Attribute, Raumpolygon beispielhaft erzeugt aus sechs Koordinaten

Da der Konstruktor keine Argumente beim erzeugen einer Instanz von SingleRoom verlangt, sind alle Attribute prinzipiell optional. Sie werden während des Filterprozesses und bei der Verknüpfung der Rasterknoten mit den Räumen aber alle mit Werten belegt. Die vollständige Darstellung der Klassen SingleRoom, Point und Rooms mit den Datentypen aller Attribute sowie den Objektmethoden ist in Anhang B.2 gegeben. Zur Vereinfachung der programminternen Verarbeitung werden alle Instanzen in einer ArrayList zusammengefasst, welche wiederum ein Attribut zum Objekt einer Klasse *Rooms* ist.

Sowohl die Instanzen von PlanElement als auch die von SingleRoom werden in den späteren Programmteilen als semantische Elemente weiterverwendet.[20] Nach Aufbereitung der gefilterten Elemente (siehe Seite 82) stellen die PlanElement-Objekte alle Elemente dar, welche für die Rastererzeugung und Wegstreckenermittlung die geometrische Grundlage bilden. Die Instanzen von SingleRoom repräsentieren die Räume des Gebäudemodells in dem Softwareprototypen und stellen die vorgenannten Informationen bereit.

### Datenimport mittels Apstex-Framework

Mit dem Apstex-Framework wird einem Java-Softwareprojekt eine weitere Bibliothek hinzugefügt, welche sämtliche Klassen und Relationen des IFC-Schemas als Java-Struktur

[20] Da diese Elemente Informationen enthalten, die für den Programmablauf und die Rettungswegermittlung eine vordefinierte Bedeutung haben, kann hier von einer Semantik gesprochen werden.

bereitstellt. Ebenfalls wird ein Werkzeug bereitgestellt, mit welchem es möglich wird, eine IFC-Datei einzulesen und in diese Java-Struktur zu konvertieren. Hierfür muss in dem Softwareprojekt ein Objekt der Klasse *IfcModel* erzeugt werden, welchem die IFC-Daten über eine sogenannte „read-Methode“ hinzugefügt werden. Dieses Objekt beinhaltet das vollständige Bauwerksmodell aus der IFC-Datei in der Java-Struktur.

Instanzen von IfcModel stellen, wie anderen Instanzen der übrigen Klassen auch, weitere Objektmethoden bereit. Mit einer dieser Methoden können nun alle Objekte einer bestimmten Klasse aus dem IfcModel gefiltert und als „Collection“ dieses Typs [21] in einem separaten Objekt abgelegt werden. Dieses beinhaltet wieder Objektmethoden, mittels derer auf Attribute und referenzierte Elemente dieser Objekte zugegriffen werden kann.

Es müssen die Daten der entsprechenden Instanzen von IfcWall, IfcOpeningElement, IfcColumn sowie IfcSpace ermittelt und derartig aufbereitet werden, dass sie in den Klassen PlanElement und SingleRoom abgespeichert werden können. Diese Prozesse werden in den folgenden Abschnitten detaillierter beschrieben.

### Datenfilter für Wände und Stützen

Wände und Stützen liefern für diesen Softwareprototypen die gleichen Informationen aus der IFC-Datei und unterscheiden sich hier daher nur durch ihren Klassennamen. Da der Ablauf des Datenfilters für beide Elemente gleich ist, wird an dieser Stelle nur der für die Klasse IfcWall beschrieben. Der gleiche Algorithmus wird im Softwareprototypen analog auch auf die Elemente der Klasse IfcColumn angewendet.

Zunächst werden aus IfcModel alle Instanzen der Klasse IfcWall gefiltert [22] und in einem Collection-Objekt gespeichert. Durch die Collection kann nun iteriert werden, um von jedem einzelnen IfcWall-Element die entsprechenden Informationen zu erhalten. Die globale ID und die Bezeichnung sind gemäß der IFC-Klassenstruktur[23] direkt in dem jeweiligen Objekt enthalten und können über entsprechende Objektmethoden ausgegeben werden. Die Lage sowie die Bauteilgeometrie müssen mittels weiterer Iterationen über die referenzierten Elemente ermittelt werden.

Zuerst werden die Koordinaten der Grundrissgeometrie ermittelt, da diese im Anschluss auf Basis der Informationen zum Versatz der Koordinatensysteme in Koordinaten des globalen Koordinatensystems transformiert werden. Die Vorgehensweise orientiert sich dabei an dem Klassenschema aus Abbildung 5.4 (Seite 50). Für das betrachtete IfcWall-Objekt wird das IfcProductRepresentation-Objekt (in der IFC-Beispieldatei *IfcProductDefinitionShape*-Objekt) ausgegeben, welches eine „List“ [24] vom Typen *IfcRepresentation* enthält. In der IFC-Beispieldatei enthält diese List-Objekte der Klasse *IfcShapeRepresentation*, was aufgrund der Zuweisungskompatibilität erbender Klassen möglich ist. Ein IfcShapeRepresentation-Objekt steht dabei immer für eine bestimmte Art der Bauteilrepräsentation, was durch den String-Typen [25] „RepresentationType“ angegeben ist. Es müssen nun die ent-

[21] Eine *Collection* ist in Java das Ausgangselement der Vererbungshierarchie, zu der bspw. auch ArrayList gehört. In einer Collection können ebenfalls mehrere Instanzen einer bestimmten Klasse ohne vorherige Angabe einer Dimension dieses Arrays enthalten sein.

[22] Schließt auch die Instanzen von IfcWallStandardCase ein, da diese Klasse von IfcWall erbt.

[23] IfcWall erbt von IfcProduct, die wiederum von IfcRoot erbt. Dort sind die Attribute *GlobalID* sowie *Name* definiert, sodass Instanzen von IfcWall diese Attribute ebenfalls enthalten. Siehe dazu auch Abbildung 5.3 auf Seite 49.

[24] *List* erbt von Collections und bietet ebenfalls die Möglichkeit, eine variable Anzahl von Instanzen einer bestimmten Klasse zu speichern.

[25] Korrekterweise eigentlich durch IfcLabel, was jedoch ein String-Attribut ist.

haltenden/referenzierten (IfcRepresentationItem-)Objekte des IfcRepresentation-Objekts ermittelt werden, welches vom RepresentationType „BoundingBox“ ist. Da es ein oder mehrere IfcRepresentationItem-Objekte sein können, müssen diese in einem „Set“ [26] ihres Datentyps zwischengespeichert werden. Im betrachteten Fall liegt hier zwar immer nur ein Objekt vor (da es nur eine Bounding-Box-Beschreibung pro Bauteil gibt), es muss aber dennoch durch dieses Set „iteriert“ werden, um das IfcRepresentationItem-Objekt zu erhalten. In der IFC-Beispieldatei ist in dem Set direkt ein IfcBoundingBox-Objekt vorhanden (möglich, weil IfcBoundingBox von IfcRepresentationItem erbt). Da es mit den Angaben zu den Abmessungen jedoch Attribute enthält, welche es nicht von IfcRepresentation erbt, kann auf diese zunächst nicht zugegriffen werden. [27] An dieser Stelle muss ein „Casting“ [28] des Objekts vom Datentypen *IfcRepresentationItem* in den Datentypen *IfcBoundingBox* erfolgen. In dem IfcBoundingBox-Objekt sind nun alle Attribute enthalten, die zur geometrischen Beschreibung der Bounding-Box erforderlich sind.

Die Bounding-Box hat mit *XDim*, *YDim* und *ZDim* drei Längenangaben zur Beschreibung ihrer Ausdehnung in alle drei Richtungen. Die z-Richtung ist für den hier betrachteten Anwendungsfall der Rettungswegermittlung irrelevant, da nur die Grundrissdaten benötigt werden. Aus der x- und y-Ausdehnung lassen sich vier Koordinaten zur Beschreibung des Wandgrundrisses errechnen. Dazu wird der „Corner“ der BoundingBox als Koordinatenursprung mit den x- und y-Koordinaten (0|0) definiert und die übrigen Koordinaten mit den entsprechenden Werten aus den Abmessungen belegt (siehe Abbildung 6.7).

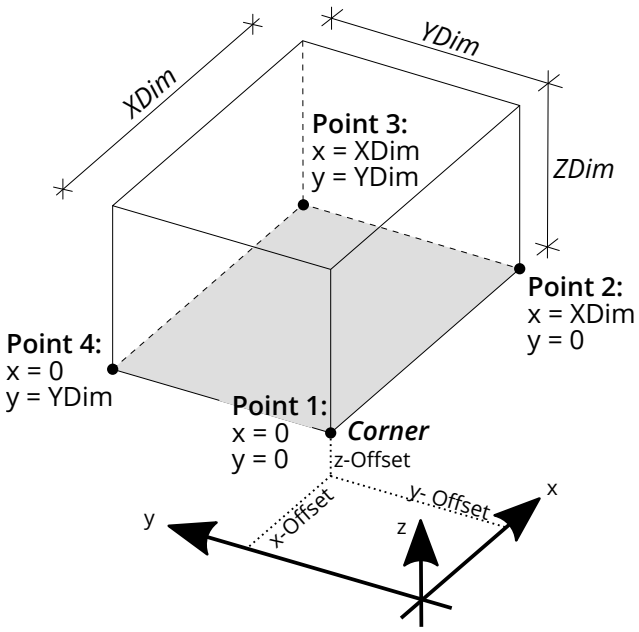



Abbildung 6.7: Filterung und Datenkonvertierung von Geometriedaten der IfcBoundingBox

Gleichzeitig kann der Versatz des „Corners“ der Bounding-Box zum Koordinatenursprung des lokalen Koordinatensystems ausgelesen werden. Auch hier ist nur der Versatz in x- und

[26] *Set* erbt ebenfalls von Collections und ist *List* ähnlich. Im Gegensatz dazu können in Set jedoch keine Duplikate gespeichert werden.

[27] Es kann deshalb nicht auf die Attribute von IfcBoundingBox zugegriffen werden, da das Objekt IfcRepresentation bzw. IfcShapeRepresentation nur die Objektmethoden von IfcRepresentationItem, aber nicht die von IfcBoundingBox kennt.

[28] Casting ist das Überführen eines Datentyps in einen anderen und funktioniert nur bei zuweisungskompatiblen Datentypen.

y-Richtung relevant. Dazu werden die IfcLengthMeasure-Werte des *IfcCartesianPoint* aus der IfcBoundingBox ausgelesen und als Fließkommazahlen gespeichert. Dieser Versatz wird bei der später folgenden Koordinatentransformation berücksichtigt. Sowohl die Werte der Eckkoordinaten als auch die des Versatzes werden in Hilfsvariablen zwischengespeichert, da in das PlanElement-Objekt erst die transformierten Koordinaten eingetragen werden.

Im nächsten Schritt muss der Versatz des lokalen Koordinatensystems der Bounding-Box zum globalen Koordinatensystem WCS ermittelt werden. Dazu wird das IfcObjectPlacement-Objekt aus der IfcWall-Instanz gefiltert. Da in der IFC-Datei an dieser Stelle ein Objekt der Klasse IfcLocalPlacement gespeichert wird, auf dessen Attribute nicht zugegriffen werden kann, muss zunächst ein Casting des Datentyps IfcObjectPlacement zu IfcLocalPlacement erfolgen. Wie in Abschnitt 5.2.4 bereits erläutert wurde, enthält IfcLocalPlacement sowohl eine die Angabe des Versatzes zu einem Bezugskoordinatensystem, als auch eine Referenz zum IfcLocalPlacement des Objekts, dessen lokales Koordinatensystem das zuvor genannte Bezugskoordinatensystem ist. Die Abhängigkeiten der verschiedenen Koordinatensysteme von dem Wandelement bis zum WCS sind in Abbildung 6.8 dargestellt.

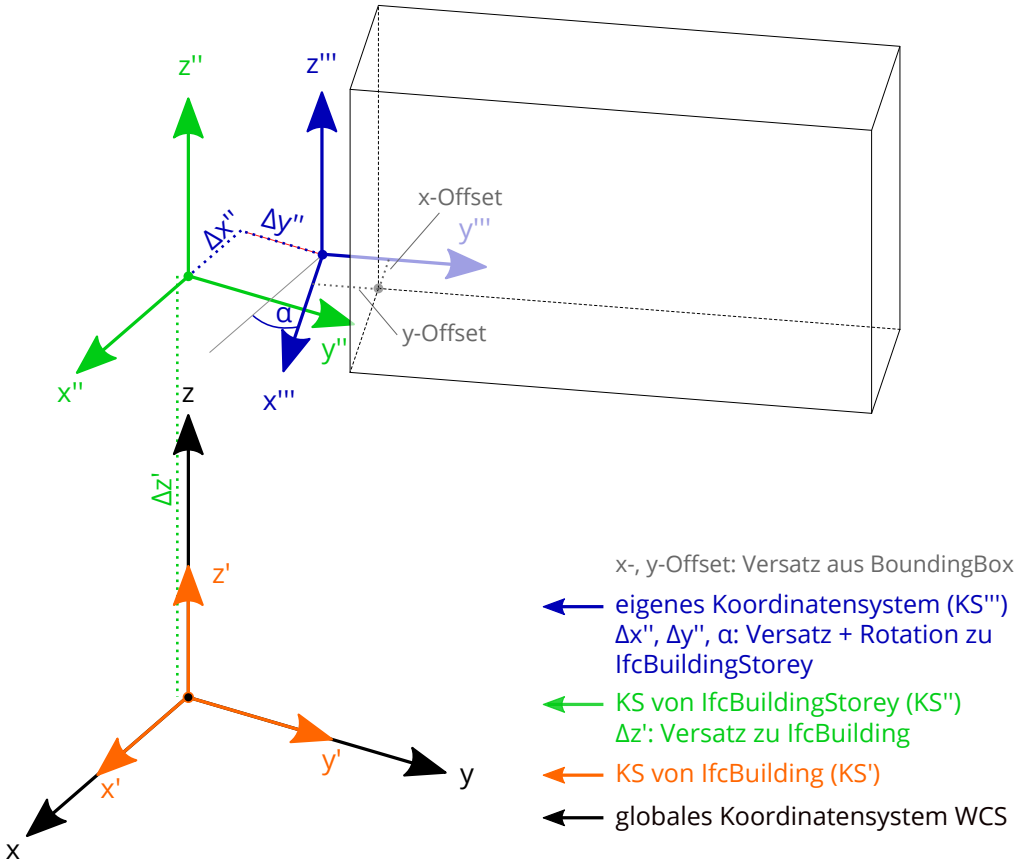



Abbildung 6.8: LocalPlacement von IfcWall in Abhängigkeit von anderen Koordinatensystemen

Wie oben bereits beschrieben, hat die Bounding-Box einen Versatz in x- und y-Richtung von ihrem „Corner“ zu ihrem lokalen Koordinatensystem $KS^{III}$. Dieses bezieht sich wiederum auf das lokale Koordinatensystem des Geschosses, in welchem die Bounding-Box liegt ($KS^{II}$). Zwischen $KS^{II}$ und $KS^{III}$ gibt es einen Versatz in x- und y-Richtung, der (mögliche) Versatz in z-Richtung ist an dieser Stelle irrelevant. Ebenso kann eine Rotation zwischen

diesen beiden Koordinatensystemen angegeben sein ($\alpha$). Ein Geschoss wird über die IFC-Klasse *IfcBuildingStorey* ausgedrückt, die hier aber nicht weiter behandelt wird. Das $KS^{II}$ hat seinen Bezug zum lokalen Koordinatensystem des Bauwerks ($KS^{I}$). Mit diesem Versatz wird die Höhenlage des Geschosses angegeben, was für die Zuordnung der Bauteile zu ihren jeweiligen Geschossen im weiteren Programmablauf relevant ist. Das $KS^{I}$ bezieht sich schlussendlich auf das globale Koordinatensystem WCS, sodass der Versatz und die Rotation zwischen drei Koordinatensystemen sowie der Versatz zwischen lokalem Koordinatensystem und Bounding-Box bei der Transformation der Koordinaten berücksichtigt werden muss.

Sowohl in der IFC-Beispieldatei, als auch in weiteren erstellen IFC-Dateien, ist zu erkennen, dass $KS^{I}$ mit dem WCS übereinstimmt und das $KS^{II}$ zum $KS^{I}$ lediglich einen Versatz in z-Richtung hat. Aus diesem Grund wurde die in Abbildung 6.8 gegebene Darstellungsweise gewählt. Aus Gründen der Vereinfachung ist daher in dem Softwareprototypen nur eine Filterung und Verarbeitung des x- und y-Versatzes sowie der Rotation zwischen dem eigenen lokalen und dem $KS^{II}$ sowie des z-Versatzes zwischen $KS^{I}$ und $KS^{II}$ implementiert.

Um die Bezugshöhe zur Geschossbestimmung zu erhalten, muss der z-Versatz des lokalen Koordinatensystems des Geschosses ($KS^{II}$) in Bezug zum WCS, ausgehend vom eigenen lokalen Koordinatensystem ($KS^{III}$) ermittelt werden. Dazu wird das *PlacementRelTo* des eigenen IfcLocalPlacement ausgegeben, welches das IfcLocalPlacement des Geschosses ist. Gemäß dem Klassenschema aus Abbildung 5.7 (Seite 54) wird von diesem das *RelativePlacement* ermittelt, welches vom Typ *IfcAxis2Placement* sein muss. In der IFC-Datei liegt es an dieser Stelle als Instanz von *IfcAxis2Placement3D* vor, was aufgrund der Zuweisungskompatibilität erbender Klassen auch korrekt ist. Da IfcLocalPlacement nur die Objektmethoden von IfcAxis2Placement, nicht aber die von IfcAxis2Placement3D kennt, muss das ermittelte Objekt zunächst in diesen Datentyp überführt werden (Casting). Aus dem IfcAxis2Placement3D-Objekt wird der Versatz (Location) über die Referenz zu IfcCartesianPoint bestimmt, indem die dort abgelegte Fließkommazahl [29] für z (dritter Eintrag in der *List*) ausgelesen wird. Dabei handelt es sich um die Werte für den vertikalen Versatz der Koordinatensysteme $KS^{I}$ und $KS^{II}$, wobei $KS^{I}$ mit dem WCS übereinstimmt und der Wert somit den globalen Versatz in z-Richtung beschreibt.

Der horizontale Versatz und die Rotation wird ebenfalls dem IfcLocalPlacement-Objekt gemäß dem Klassenschema aus Abbildung 5.7 entnommen. Um den Bezug zum globalen Koordinatensystem WCS herzustellen genügt es, den Versatz zwischen dem eigenen lokalen Koordinatensystem $KS^{III}$ und dem Bezugskoordinatensystem $KS^{II}$ zu ermitteln, da $KS^{II}$ gegenüber dem WCS keinen horizontalen Versatz aufweist. Im Gegensatz zur Ermittlung des z-Versatzes wird daher für den horizontalen Versatz direkt das *RelativePlacement* ermittelt. Die weitere Vorgehensweise der Filteriteration entspricht dann im Wesentlichen der des vorhergehenden Absatzes. Hier werden jedoch aus dem IfcCartesianPoint des IfcAxis2Placement3D-Objekts die Fließkommazahlen für x (erster Eintrag in der *List*) und y (zweiter Eintrag in der *List*) ausgelesen.

Die Rotation des lokalen Koordinatensystems $KS^{III}$ zum Bezugskoordinatensystem $KS^{II}$ wird vom Objekt IfcAxis2Placement3D über die *RefDirection* bestimmt. Diese verweist auf ein Objekt der Klasse *IfcDirection*, welche zwei (oder drei) Fließkommazahl-Attribute besitzt. In der IFC-Beispieldatei sind zwei Werte angegeben, welche über x-y-Koordinaten den Winkel der horizontalen Rotation angeben. Für die Verarbeitung in dem Softwareprototypen müssen diese Koordinaten in einen Winkel (Radiant) umgerechnet werden. Dies geschieht über den Arcustangens dieser beiden Werte, wobei eine quadrantenabhängige

[29] Korrekterweise eigentlich IfcLengthMeasure, was jedoch ein Fließkommazahl-Attribut ist.

Fallunterscheidung durchzuführen ist. Liegen die Koordinaten auf den Achsen, resultieren daraus feste Werte für die Winkel. Die entsprechenden Umrechnungsformeln und Werte sind in der Abbildung 6.9 gegeben.

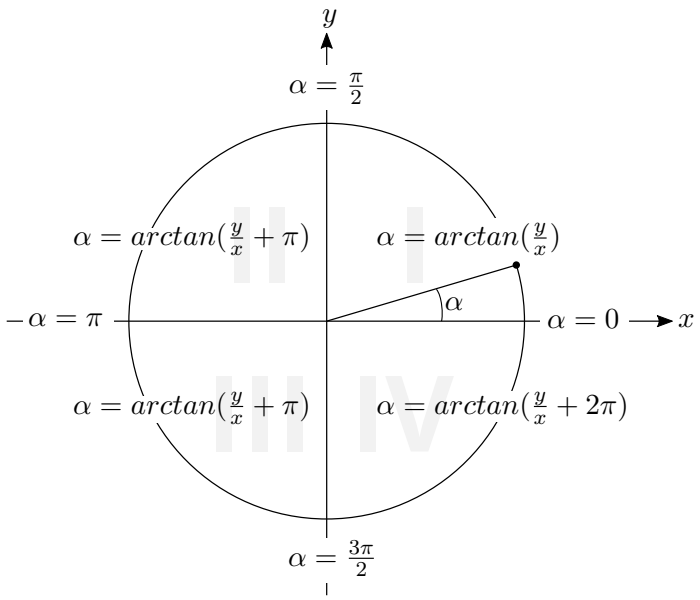



Abbildung 6.9: Quadrantenabhängige Transformation von x-y-Koordinaten in Radiant

Ist der Winkel als Radiant berechnet, liegen mit dem horizontalen Versatz der Koordinatensysteme $KS^{II}$ und $KS^{III}$, dem Versatz der Bounding-Box zu deren lokalem Koordinatensystem ($KS^{III}$) und eben der Rotation alle Informationen vor, um die Koordinaten der Bounding-Box in das globale Koordinatensystem WCS zu transformieren. Die Transformation der vier Eckpunkte erfolgt nacheinander. Zunächst wird der Versatz zwischen *Corner* und $KS^{III}$ berechnet:

$$x_{corner} = PointX + xOffset \tag{6.1}$$
$$y_{corner} = PointY + yOffset \tag{6.2}$$

Um die Rotation des Koordinatensystems $KS^{II}$ zu $KS^{III}$ auf $x_{corner}$ und $y_{corner}$ anzuwenden, ist das Lösen eines linearen Gleichungssystems mit zwei Unbekannten erforderlich. Nachfolgend ist das Gleichungssystem gegeben, mit welchem die transformierten Koordinaten $x'$ und $y'$ berechnet werden:

$$\begin{matrix} x'' \cdot cos(\alpha) + y'' \cdot sin(\alpha) \\ -x'' \cdot sin(\alpha) + y'' \cdot cos(\alpha) \end{matrix} = \begin{matrix} x_{corner} \\ y_{corner} \end{matrix} \tag{6.3}$$

Zuletzt ist der lineare Versatz zwischen den Koordinatensystemen $KS^{II}$ und $KS^{III}$ zu addieren, um die Koordinaten der Bounding-Box endgültig in das globale Koordinatensystem zu transformieren:

$$x = x'' + \Delta x'' \tag{6.4}$$
$$y = y'' + \Delta y'' \tag{6.5}$$

Sofern diese Transformation für alle vier Eckpunkte durchgeführt worden ist, liegt die Bounding-Box vollständig in globalen Koordinaten vor. An dieser Stelle können diese Koordinaten in das PlanElement-Objekt für *point1X, point1Y, point2X...* eingetragen werden. Der z-Versatz muss nicht transformiert werden und wird dem PlanElement direkt zugewiesen. Damit sind alle Informationen in dem PlanElement abgelegt, welche zur Beschreibung eines Wandelements im weiteren Programmablauf erforderlich sind. Die Filterung und Aufbereitung der Daten eines Objekts der Klasse IfcWall ist an dieser Stelle abgeschlossen.

### Datenfilter für Türöffnungen

Die Geometrie von Öffnungen in Bauteilen wird durch die Klasse *IfcOpeningElement* über eine Bounding-Box genauso beschrieben wie Wände oder Stützen. Auch die Bauteilplatzierung erfolgt auf die gleiche Art und Weise mittels IfcLocalPlacement. Aus diesem Grund wird für die Verarbeitung in dem Softwareprototypen jede Öffnung durch eine Instanz von PlanElement repräsentiert. Der Datenfilter für *Tür*öffnungen unterscheidet sich zu dem der Wandelemente dahingehend, dass dieser eine *bedingte* Filterung durchführt, d.h., Elemente nur bei der Erfüllung bestimmter Bedingungen filtert.

Gebäude enthalten üblicherweise Fenster, die in IFC auch durch eine eigene Klasse *IfcWindow* repräsentiert werden. Gleichzeitig wird für jedes Fenster ein IfcOpeningElement erzeugt, das sich von denen der Türen nicht unterscheidet. Daher dürfen nur IfcOpeningElement-Objekte gefiltert werden, welche infolge der Platzierung eines Bauteils vom Typ *IfcDoor* erzeugt worden sind. Wie in Abschnitt 5.2.6 bereits erläutert wurde, wird die Verbindung zwischen IfcDoor und IfcOpeningElement über inverse Beziehungen zu einem IfcRelFillsElement-Objekt hergestellt. Es müssen also zunächst alle Instanzen von IfcRelFillsElement gefiltert und in einem Collection-Objekt gespeichert werden. Bei der Iteration durch diese Collection werden nur dann referenzierte IfcOpeningElement-Instanzen ausgegeben, wenn das zugehörige *RelatingBuildingElement* vom Typ *IfcDoor* ist. Dieses wird auch mit ausgegeben, da insbesondere dessen globale ID im folgenden Programmschritt benötigt wird.

Um Ausgangspunkte für die Rettungswegermittlung zu erhalten, müssen bestimmte Türen als „Notausgang“ im Gebäude definiert sein. Diese Information wird in einem IfcPropertySet-Objekt gespeichert, welches gemäß IFC-Schema über eine inverse Beziehung zu einem IfcRelDefinesByProperties-Objekt mit dem IfcDoor-Element verbunden ist (siehe Abschnitt 5.2.6). Dementsprechend sind auch alle Instanzen von IfcRelDefinesByProperties aus dem IfcModel zu filtern. Von diesen werden in einer weiteren Iteration die zugehörigen IfcDoor-Objekte ausgegeben und mit der ID des zu dem Opening-Element gehörenden Door-Elements abgeglichen. Ist diese identisch, kann von dem aktuellen IfcRelDefinesByProperties-Objekt die zugehörige *RelatingPropertyDefinition* ausgegeben werden, welche die Angabe zu einem Notausgang enthält.

Zur Ermittlung der Bauteileigenschaft „Notausgang“ wird zunächst das ausgegebene IfcPropertySetDefinition=Objekt in den Datentypen *IfcPropertySet* umgewandelt. [30] Aus diesem wird eine *List* des Typs *IfcProperty* ausgelesen, durch welche iteriert wird, um die einzelnen Bauteileigenschaften als IfcProperty-Instanzen zu erhalten. In der IFC-Datei liegen diese Elemente als IfcPropertySingleValue-Objekte vor. Hier ist ein Casting in den Datentypen *IfcPropertySingleValue* erforderlich, da die Attribute der Bauteileigenschaften

[30] In der IFC-Datei liegt die *RelatingPropertyDefinition* direkt als IfcPropertySet vor. Da IfcRelDefinesByProperties die Objektmethoden von IfcPropertySet nicht kennt, ist das Casting an dieser Stelle erforderlich.

erst in IfcPropertySingleValue definiert werden und die Objektmethoden zum Auslesen dieser den Instanzen von IfcPropertySet nicht bekannt sind. Jedes IfcPropertySingleValue-Objekt hat eine Bezeichnung (*Name*), welche Aufschluss über die Bedeutung der Bauteileigenschaft gibt (siehe dazu auch Abschnitt 5.2.5). Es ist die IfcPropertySingleValue-Instanz aus der List zu filtern, welche die Bezeichnung „FireExit“ hat. Diese Instanz beinhaltet als *NominalValue* ein *Boolean*-Attribut [31] , welches angibt, ob es sich bei der betrachteten Tür um einen Notausgang handelt („*true*“) oder nicht („*false*“). Dieser Wert wird in das PlanElement-Objekt der betrachteten Türöffnung für das Boolean-Attribut *isFireExit* eingetragen.

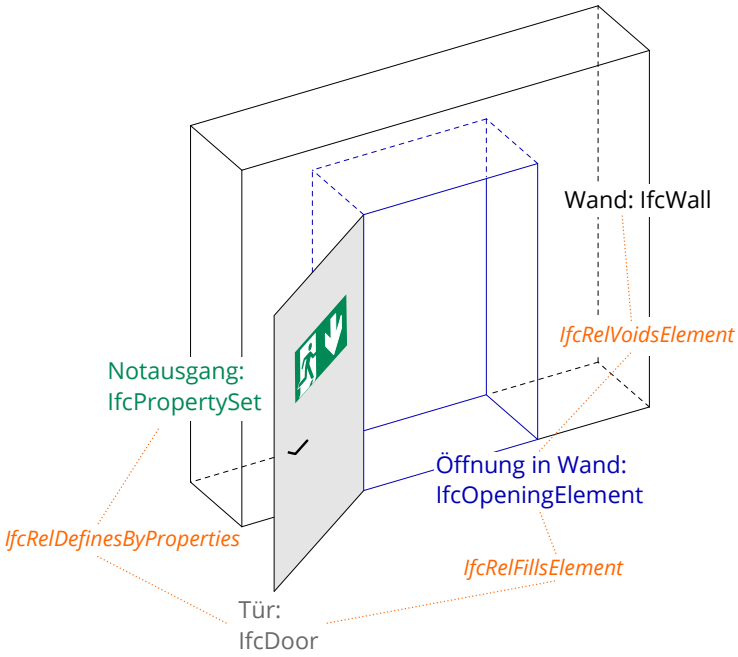



Abbildung 6.10: Über inverse Beziehungen verbundene Bauteile und Eigenschaften einer Tür

Nachdem die Filterung der Information bezüglich des Notausgangs abgeschlossen ist, wird das Wandelement ermittelt, in welchem die betrachtete Türöffnung liegt. Von diesem wird die globale ID ausgelesen und in dem PlanElement-Objekt als *relatedElementID* gespeichert. Im späteren Programmverlauf wird auf dieses Attribut zurückgegriffen, um das Wandelement zu filtern, in welchem die Öffnung erzeugt werden soll (siehe *Aufbereitung der gefilterten Elemente und Datenübergabe*). Die Verbindung zwischen einem IfcOpeningElement-Objekt und dem zugehörigen IfcWall-Objekt wird über inverse Beziehungen zu einem IfcRelVoidsElement-Objekt hergestellt (siehe Abbildungen 5.11 und 6.10). Das IfcOpeningElement-Objekt ist bereits bekannt (siehe oben); es muss von diesem noch die Element-ID ermittelt werden. Anschließend werden alle IfcRelVoidsElement-Instanzen gefiltert und in einer Collection dieses Typs gespeichert. Durch diese Collection wird solange iteriert, bis anhand der ID das aktuell betrachtete IfcOpeningElement-Objekt *RelatedOpeningElement* zu einer IfcRelVoidsElement-Instanz gefunden wurde. Dementsprechend handelt es sich bei dem *RelatingBuildingElement* um dasjenige IfcWall-Objekt, in welchem die Öffnung liegt. Von diesem Objekt wird die globale ID ermittelt und in dem PlanElement-Objekt als *relatedElementID* abgelegt. Die globale ID des IfcOpeningElement-Objekts wird

[31] Korrekterweise ist das Attribut vom Typ *IfcValue*, liegt in der IFC-Datei aber als *boolean* vor. Zur Datentypkonvertierung ist hier ein Casting erforderlich.

in dem PlanElement-Objekt als *elementID* gespeichert. An dieser Stelle sind alle spezifischen Attribute einer Öffnung der PlanElement-Instanz hinzugefügt worden, sodass mit der Ermittlung und Transformation der geometrischen Repräsentation fortgefahren werden kann.

Wie oben bereits genannt, wird die Geometrie von Öffnungen wie bei Wänden durch eine Bounding-Box beschrieben, da sowohl IfcOpeningElement als auch IfcWall ihre Repräsentation von der Klasse *IfcProduct* erben. Somit müssen aus der IfcOpeningElement-Instanz ebenfalls über deren Bounding-Box-Repräsentation die *XDim*- und *YDim*-Längenangaben zur Berechnung der vier Grundrisskoordinaten gefiltert werden (Abbildung 6.7). Auch der Versatz des „Corners“ zum lokalen Koordinatensystem ist hier für die Transformation der Grundrisskoordinaten zu ermitteln. Auf die nähere Beschreibung, welche Filterroutinen anzuwenden sind, um die Bounding-Box-Attribute zu erhalten, wird an dieser Stelle verzichtet und stattdessen auf die Beschreibung in dem vorhergehenden Abschnitt *Datenfilter für Wände und Stützen* verwiesen.

Der Versatz des lokalen Koordinatensystems der Bounding-Box zum globalen Koordinatensystem WCS wird grundsätzlich auf die gleiche Art ermittelt wie bei den IfcWall-Instanzen. Es ist jedoch ein weiterer Versatz zu berücksichtigen, da sich das lokale Koordinatensystem der Bounding-Box des IfcOpeningElement-Objekts nicht auf das Koordinatensystems des Geschosses, sondern auf das des Wandelements bezieht. Abbildung 6.11 zeigt eine vollständige Darstellung aller Abhängigkeiten vom lokalen Koordinatensystem bis zum WCS.

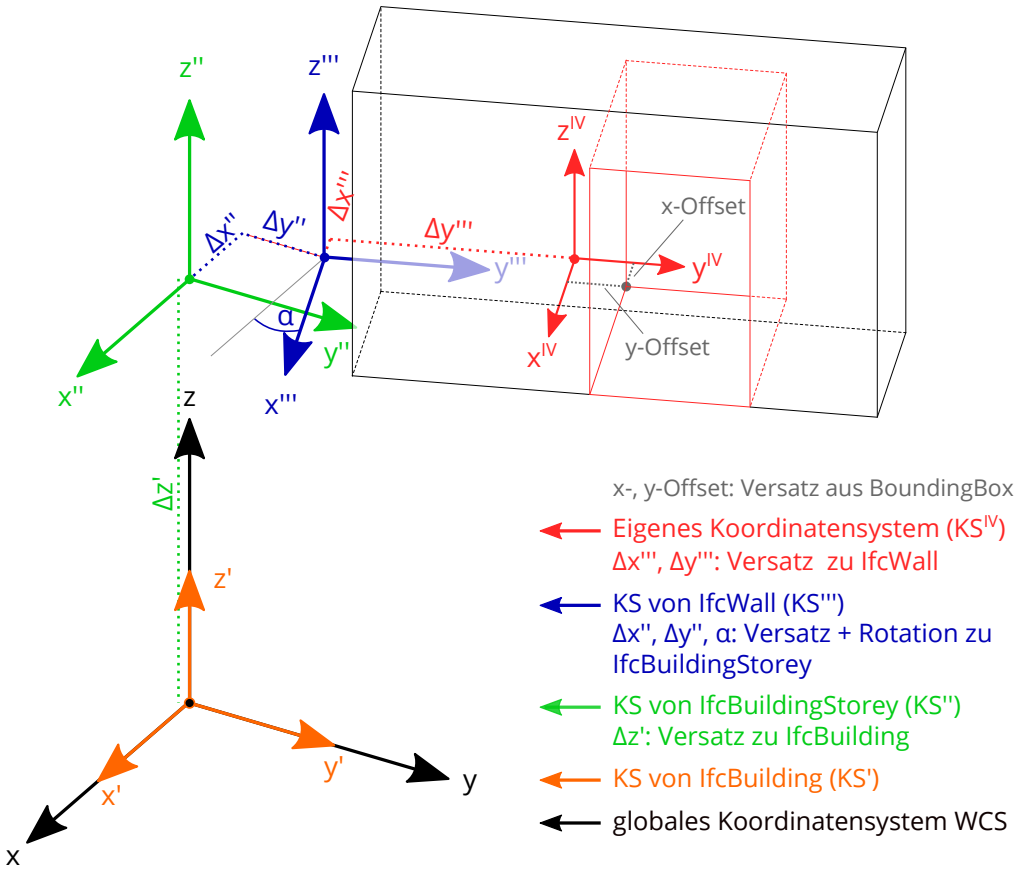



Abbildung 6.11: LocalPlacement von IfcOpeningElement in Abhängigkeit von anderen Koordinatensystemen

Zwischen den verschiedenen Koordinatensystemen WCS, $KS^{I}$ – $KS^{IV}$ sind ein Versatz in x-, y- und z-Richtung sowie eine Rotation theoretisch möglich. Aus dem Abschnitt *Datenfilter für Wände und Stützen* ist jedoch bereits bekannt, dass WCS und $KS^{I}$ immer übereinstimmen und zwischen $KS^{I}$ und $KS^{II}$ nur ein Versatz in z-Richtung besteht. Bei den IfcOpeningElement-Instanzen wurde ergänzend festgestellt, dass zwischen $KS^{III}$ und $KS^{IV}$ eine Rotation weder in der IFC-Beispieldatei, noch in weiteren versuchsweise erstellten Dateien auftraten, wodurch aus Gründen der Vereinfachung eine solche bei der Transformation ebenfalls nicht berücksichtigt wird.

Um die Bezugshöhe zur Geschossbestimmung für das Öffnungselement zu erhalten, muss der z-Versatz des $KS^{I}$ zu $KS^{II}$ nun über ein weiteres Koordinatensystem ($KS^{III}$) ermittelt werden. Hierfür wird zunächst das *PlacementRelTo* des eigenen IfcLocalPlacement-Objekts [32] ausgegeben, wobei es sich um das IfcLocalPlacement-Objekt der Wand handelt ($KS^{III}$). Von diesem muss wieder das PlacementRelTo ermittelt werden, um das IfcLocalPlacement des Geschosses zu erhalten ($KS^{II}$). Der z-Versatz dieses Koordinatensystems zum WCS lässt sich nun über das *RelativePlacement* bestimmen. Da die Vorgehensweise hierfür ab dieser Stelle derer aus dem Abschnitt *Datenfilter für Wände und Stützen* entspricht, wird sie hier nicht näher erläutert und stattdessen auf jenen Abschnitt verwiesen. Der Wert des ermittelten z-Versatzes wird in dem PlanElement-Objekt als *storeyZOffset* gespeichert.

Für den gesamten horizontalen Versatz des lokalen Koordinatensystems $KS^{IV}$ zum WCS müssen sowohl der x-y-Versatz von $KS^{IV}$ zu $KS^{III}$ als auch der x-y-Versatz sowie die Rotation von $KS^{III}$ zu $KS^{II}$ ermittelt werden. Ausgehend vom IfcLocalPlacement-Objekt des Öffnungselements wird der Versatz zwischen $KS^{IV}$ und $KS^{III}$ über das *RelativePlacement* bestimmt. Um den Versatz zwischen $KS^{III}$ und $KS^{II}$ zu erhalten, muss ausgehend vom IfcLocalPlacement-Objekt des Öffnungselements das *RelativePlacement* ausgegeben werden, was wieder dem IfcLocalPlacement-Objekt der Wand entspricht. Über das *RelativePlacement* dieses Objekts werden Versatz und Rotation ermittelt.

Die Vorgehensweise zur Bestimmung von Rotation und Versatz ist ab der Ermittlung des *RelativePlacements* sowohl für $KS^{IV}$ zu $KS^{III}$, als auch für $KS^{III}$ zu $KS^{II}$ die gleiche wie im Abschnitt *Datenfilter für Wände und Stützen*, weswegen sie mit Hinweis auf diesen Abschnitt nicht erneut beschrieben wird.

An dieser Stelle liegen der Versatz des „Corners“ der Bounding-Box zu ihrem lokalen Koordinatensystem $KS^{IV}$, der x-y-Versatz zwischen $KS^{IV}$ und $KS^{III}$ sowie der x-y-Versatz und die Rotation zwischen $KS^{III}$ und $KS^{II}$ vor, wobei der Winkel der Rotation gemäß Abbildung 6.9 noch in einen Radianten umgerechnet werden muss. Damit sind alle Informationen vorhanden, die es für eine korrekte Transformation der Grundrisskoordinaten der Bounding-Box in das globale Koordinatensystem bedarf.

Für den linearen Versatz zwischen *Corner* und $KS^{III}$ gelten die Gleichungen 6.1 und 6.2 entsprechend:

$$x_{corner} = PointX + xOffset$$
$$y_{corner} = PointY + yOffset$$

Da gegenüber dem Wandelement beim Öffnungselement ein weiterer Versatz zu berücksichtigen ist, werden die Transformationsvorschriften um eine weitere ergänzt. So wird der zusätzliche Versatz zwischen $KS^{IV}$ und $KS^{III}$ durch Addition berechnet:

[32] IfcLocalPlacement-Objekt zum Öffnungselement mit dem lokalen Koordinatensystem $KS^{IV}$.

$$x = x''' + \Delta x''' \tag{6.6}$$
$$y = y''' + \Delta y''' \tag{6.7}$$

Die Rotationstransformation wird über das Gleichungssystem nach Gleichung 6.3 gelöst, da sie ebenfalls zwischen den Koordinatensystemen $KS^{II}$ und $KS^{III}$ erfolgt:

$$\begin{matrix} x'' \cdot cos(\alpha) + y'' \cdot sin(\alpha) \\ -x'' \cdot sin(\alpha) + y'' \cdot cos(\alpha) \end{matrix} = \begin{matrix} x_{corner} \\ y_{corner} \end{matrix}$$

Auch der lineare Versatz zwischen $KS^{II}$ und $KS^{III}$ wird auf die gleiche Art berechnet wie im Abschnitt *Datenfilter für Wände und Stützen*. Daher gelten die Gleichungen 6.4 und 6.5 hier ebenfalls:

$$x = x'' + \Delta x''$$
$$y = y'' + \Delta y''$$

Nach Anwendung dieser Transformationsvorschriften auf alle vier Eckpunkte des Grundrisses der Bounding-Box liegt dieser in globalen Koordinaten vor und werden als *point1X, point1Y, point2X...* dem PlanElement-Objekt hinzugefügt. Damit enthält dieses nun alle erforderlichen Informationen zur Beschreibung einer Türöffnung. Das „Öffnen“ des zugehörigen Wandgrundrisses erfolgt an späterer Stelle (siehe Abschnitt *Aufbereitung der gefilterten Elemente und Datenübergabe*). In Abbildung 6.11 ist die Bounding-Box der Öffnung im Grundriss größer als die des Wandelements. Diese Darstellung ist bewusst gewählt worden, da dies auch die Gegebenheiten in der IFC-Datei widerspiegelt. Dies stellt für den Öffnungsalgorithmus allerdings kein Problem dar, da die Öffnung in den Wandgrundriss über Schnittpunkte mit dem Grundriss des Öffnungselements erzeugt wird.

### Datenfilter für Räume

Geometrische Informationen von Raumflächen werden in referenzierten Objekten von IfcSpace-Instanzen gespeichert. Diese werden zunächst aus dem Gebäudemodell gefiltert und in einer Collection des Typs IfcSpace abgelegt. Analog zu den Objekten, welche in den vorhergehenden Abschnitten erläutert wurden, wird durch diese Collection iteriert, um jedes einzelne IfcSpace-Objekt zu erhalten. Von diesem werden direkt die Attribute *GlobalId*, *Name* sowie *LongName* ausgelesen und in ein SingleRoom-Objekt übertragen.

Nach diesem Schritt wird zunächst die Personenbelegung des betrachteten Raums ermittelt. Diese wird durch ein IfcPropertySet-Objekt angegeben, welches über eine inverse Beziehung mit dem Raumobjekt in Verbindung steht. Die Referenzen werden durch eine Instanz der Klasse *IfcRelDefinesByProperties* hergestellt (siehe Abbildung 5.12). Dementsprechend wird an dieser Stelle eine Collection vom Typ IfcRelDefinesByProperties erstellt, in welche alle Instanzen dieser Klasse eingefügt werden. Danach wird für jede Instanz das zugehörige *RelatingBuildingElement* ausgegeben und von diesem die globale ID mit der globalen ID des aktuell betrachteten Raums abgeglichen. Stimmen diese überein, handelt es sich um denselben Raum und es kann die *RelatingPropertyDefinition* der IfcRelDefinesByProperties-Instanz zur weiteren Filterung ausgegeben werden.

Ab dieser Stelle erfolgt der Filteralgorithmus für dieses IfcPropertySet-Objekt analog zu dem des Notausgang-Filters, sodass hier auf die nähere Erläuterung dieses Algorithmus

verzichtet und auf den Abschnitt 6.4.2 verwiesen wird. Lediglich an der Stelle, an welcher die IfcPropertySingleValue-Instanz aus der *List* anhand ihrer Bezeichnung gefiltert wird, muss in diesem Fall nach der Instanz mit der Bezeichnung „OccupancyNumber“ gesucht werden. Als *NominalValue* beinhaltet diese Instanz ein *IfcCountMeasure*-Attribut. [33] Aus diesem kann der *integer*-Wert [34] für die Raumbelegung ausgelesen werden, welcher dem SingleRoom-Objekt als *occupancy* hinzugefügt wird.

Zu beachten ist, dass zuerst ein IfcSpace-Objekt aus der Collection betrachtet und danach erst die IfcRelDefinesByProperties-Objekte untersucht werden, von denen aus nach demselben IfcSpace-Objekt gesucht wird. Ein hypothetischer Optimierungsansatz wäre hier, die Iteration durch die IfcSpace-Collection wegzulassen und nur die IfcRelDefinesByProperties-Objekte durchzusuchen, um darüber das IfcSpace-Objekt zu erhalten, dessen Informationen gefiltert werden sollen. Dies minimiert den Programmieraufwand jedoch nicht, da mehrere IfcRelDefinesByProperties-Instanzen mit Referenzen zu demselben IfcSpace-Objekt bestehen können und eine Untersuchung, ob dieses Objekt bereits betrachtet worden ist, an dieser Stelle durchzuführen wäre.

Nach Abschluss der Belegungsermittlung wird die geometrische Repräsentation des IfcSpace-Objekts gefiltert. Der Filterprozess unterscheidet sich zu den Wand- und Öffnungselementen dahingehend, dass hier nicht nach einer IfcBoundingBox-, sondern nach einer IfcGeometricCurveSet-Instanz gefiltert wird. Diese gibt, ausgehend vom lokalen Koordinatensystem des IfcSpace-Objekts, ein Polygon über x-y-Koordinaten an.

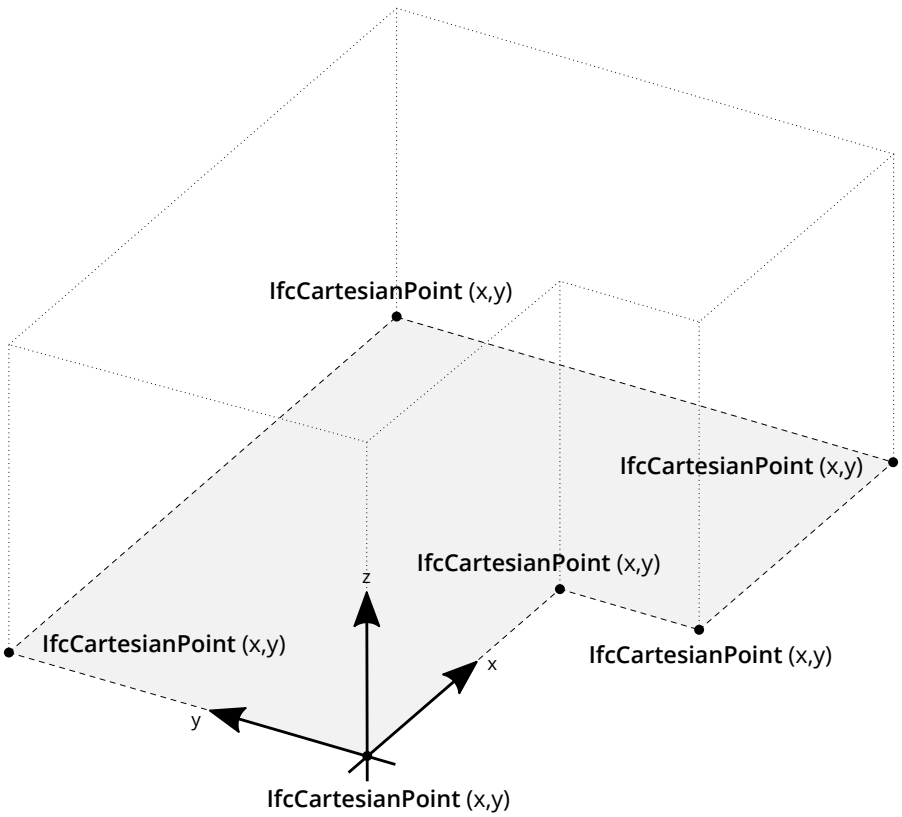



Abbildung 6.12: Filterung von Geometriedaten aus IfcGeometricCurveSet (beispielhaft mit sechs Koordinaten)

[33] Korrekterweise ist das Attribut vom Typ *IfcValue*, liegt in der IFC-Datei aber als IfcCountMeasure vor. Zur Datentypkonvertierung ist hier ein Casting erforderlich.
[34] Hier ist ebenfalls ein Casting vom Datentyp *IfcCountMeasure* zum Datentyp *integer* erforderlich.

Der prinzipielle Ablauf des Filteralgorithmus ist zunächst jedoch gleich. Es wird die *Representation* [35] des IfcSpace-Objekts ausgegeben, welche in der IFC"Beispieldatei als Objekt des zuweisungskompatiblen Datentyps *IfcProductDefinitionShape* vorliegt. Dieses beinhaltet eine *List* des Typs *IfcRepresentation*, in welcher ein oder mehrere Objekte zur weiteren Geometriebeschreibung vorhanden sind. In der IFC-Beispieldatei sind diese Objekte von der zuweisungskompatiblen Klasse *IfcShapeRepresentation*. Unter diesen Objekten muss nun dasjenige Objekt gefiltert werden, welches vom RepresentationType „GeometricCurveSet“ ist. Dieses referenziert zu einem *Set* des Typs *IfcRepresentationItem*, wobei in der IFC="Beispieldatei hier ein Objekt der von IfcRepresentationItem erbenden Klasse *IfcGeometricCurveSet* vorhanden ist. Das IfcGeometricCurveSet-Objekt beinhaltet eine List des Typs *IfcGeometricSetSelect*, wobei in der IFC-Beispieldatei in dieser List ein Objekt des zuweisungskompatiblen Typs *IfcPolyline* vorhanden sind. [36] Dieses Objekt beinhaltet erneut eine List vom Typen *IfcCartesianPoint*, in welcher sich ein oder mehrere Objekte dieses Typs befinden. Diese geben die Koordinaten an, aus welchen sich das Raumpolygon zusammensetzt. Die Werte sind als einzelne IfcLengthMeasure-Attribute getrennt für x und y in den jeweiligen IfcCartesianPoint-Objekten enthalten und können als Fließkommazahl ausgegeben werden. Damit liegen die Polygonpunkte des Raums in lokalen Koordinaten vor und müssen im übernächsten Schritt in globale Koordinaten transformiert werden.

Zuvor muss noch der Versatz des lokalen Koordinatensystems zum globalen Koordinatensystem WCS ermittelt werden. Der Filteralgorithmus hierfür entspricht im Wesentlichen dem bei Wänden, da sich das lokale Koordinatensystem von IfcSpace-Objekten ebenfalls

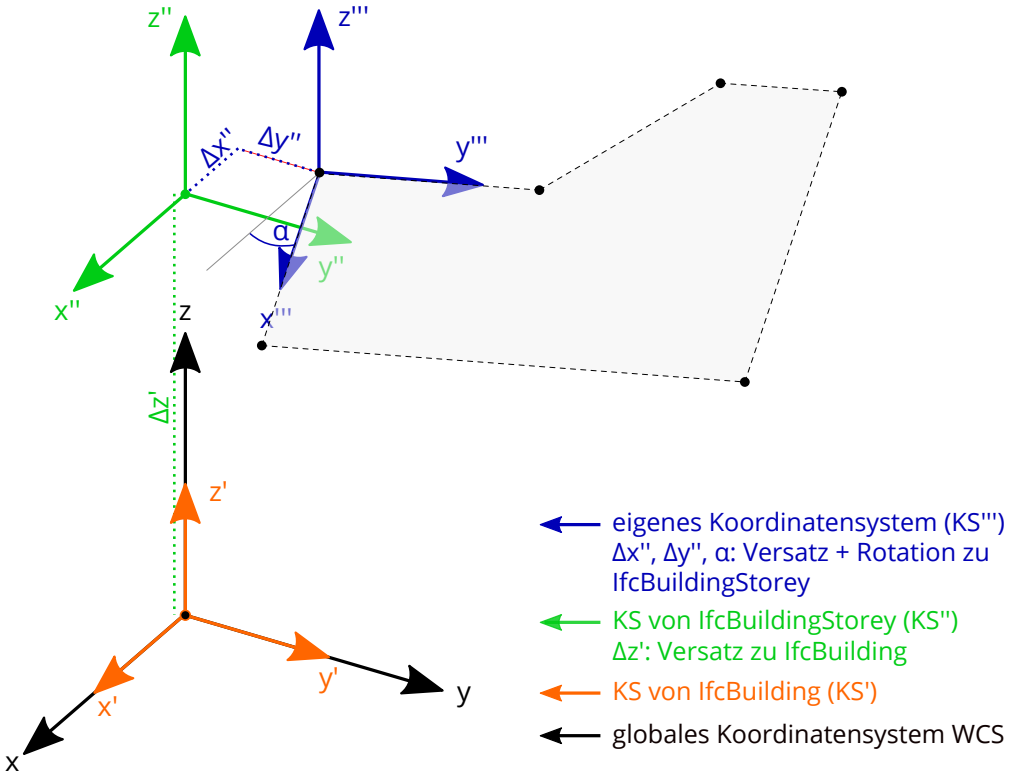



Abbildung 6.13: LocalPlacement von IfcSpace in Abhängigkeit von anderen Koordinatensystemen

[35] Die *Representation* eines Objekts muss gemäß IFC-Schema ein Objekt einer Klasse sein, welche von IfcProductRepresentation erbt.

[36] Im Softwareprototypen muss hier ein Casting zu dem Datentyp IfcPolyline erfolgen, um auf dessen Attribute zugreifen zu können.

auf das lokale Koordinatensystem von IfcBuildingStorey bezieht. Daher wird dieser Algorithmus hier nicht näher erläutert und stattdessen auf den Abschnitt Datenfilter für Wände und Stützen verwiesen. Mithilfe dieses Filteralgorithmus werden Werte für den horizontalen Versatz und die Rotation der Koordinatensysteme $KS^{III}$ und $KS^{II}$ sowie für den vertikalen Versatz in Bezug auf das WCS ermittelt. Die für die Rotation ermittelten Koordinaten werden wieder gemäß Abbildung 6.9 in einen Radianten umgerechnet, sodass mit diesen Werten die Transformation der lokalen Polygon-Koordinaten in globale Koordinaten durchgeführt werden kann.

Für die Transformation der Raumgrundrisskoordinaten können ebenfalls dieselben Gleichungen angewendet werden wie bei denen der Wandelemente. Lediglich ein Versatz eines „Corners“, wie der Bounding-Box, muss nicht mit eingerechnet werden, da ein solcher Versatz bei IfcGeometricCurveSet nicht vorhanden ist. Demzufolge wird für die Rotation das Gleichungssystem aus Gleichung 6.3 gelöst:

$$\begin{matrix} x'' \cdot cos(\alpha) + y'' \cdot sin(\alpha) \\ -x'' \cdot sin(\alpha) + y'' \cdot cos(\alpha) \end{matrix} = \begin{matrix} CartesianPointX \\ CartesianPointY \end{matrix}$$

Anschließend erfolgt die lineare Transformation nach den Gleichungen 6.4 und 6.5:

$$\begin{aligned} x &= x'' + \Delta x'' \\ y &= y'' + \Delta y'' \end{aligned}$$

Nach Anwendung des Transformationsalgorithmus auf alle Polygonkoordinaten liegen diese in globalen Koordinaten vor und werden als *Point* [37] in der ArrayList des SingleRoom-Objekts für *coordinates* hinzugefügt. Der z-Versatz zwischen den Koordinatensystemen $KS^{II}$ und $KS^{I}$ muss nicht transformiert werden und wird direkt als *storeyZOffset* in dem SingleRoom-Objekt gesetzt.

Damit sind alle Attribute des SingleRoom-Objekts mit Werten belegt (bis auf das *containedVertices*-Attribut, welches später folgt), sodass der Filter- und Datenaufbereitungsprozess für einen Raum abgeschlossen ist.

### Aufbereitung der gefilterten Elemente und Datenübergabe

An dieser Stelle liegen nun jede Wand und jede Stütze als einzelne PlanElement-Objekte vor, die in einem Floor-Objekt (siehe Seite 68) zusammengefasst sind. Der Begriff „Floor“ ist an dieser Stelle noch etwas irreführend, da hier alle Elemente sämtlicher Geschosse abgelegt sind. Eine geschossweise Aufteilung erfolgt erst in diesem Programmabschnitt.

Alle (Tür)Öffnungen liegen ebenfalls als einzelne PlanElement-Objekte vor, die in einem separaten Floor-Objekt zusammengefasst sind. Auch hier erfolgt die Geschosszuordnung in diesem Programmabschnitt, nachdem die Wände anhand der Öffnungselemente geteilt und angepasst wurden.

Die SingleRoom-Objekte, welche die Räume in dem Softwareprototypen repräsentieren, sind in einem Room-Objekt (siehe Seite 69) zusammengefasst. Sie sind ebenso noch nicht nach den unterschiedlichen Geschossen sortiert.

In Vorbereitung auf die Öffnung der Wandelemente, im Folgenden als „Öffnungsalgorithmus“ bezeichnet, wird das Floor-Objekt, in welchem die Objekte zur Beschreibung von

[37] Selbstdefinierter Datentyp mit x- und y- Fließkommazahl-Attributen. Siehe Seite 68.

Wänden und Stützen enthalten sind, dupliziert. Der Öffnungsalgorithmus wird dann auf beide Floor-Objekte getrennt angewendet, wobei dieser bei einem der beiden diejenigen Öffnungselemente nicht berücksichtigt, deren *isFireExit*-Wert als *true* gesetzt ist. Dies hat den Hintergrund, dass das Polygon, welches den gesamten Gebäudegrundriss umschließt, nur bei einem vollständig geschlossenen Kantenzug korrekt ermittelt wird (siehe Abbildung 6.14). [38] Gleichzeitig benötigt die kürzeste-Pfad-Ermittlung einen „offenen“ Grund-

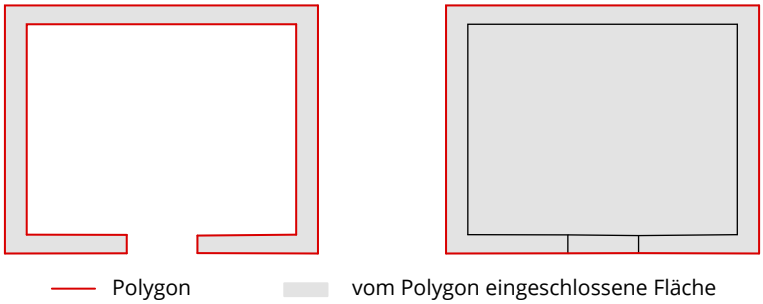



Abbildung 6.14: Falsch (links) und korrekt ermitteltes Grundrisspolygon

riss, da es mit geschlossenen Wänden bei Treppenräumen zu Problemen kommen könnte. Als Startknoten für die kürzeste-Pfad-Ermittlung wird der dem Notausgang nächstgelegene Rasterknoten ausgewählt. Dieser könnte bei „geschlossenen“ Wänden jedoch auf der „falschen“ Seite liegen (siehe Abbildung 6.15), sodass von diesem Knoten aus nur die Verbindungen zu Knoten innerhalb des Treppenraums gefunden werden, nicht aber im übrigen (und relevanten) Teil des Gebäudes. Durch das Duplikat ist es möglich, sowohl eine Grund-

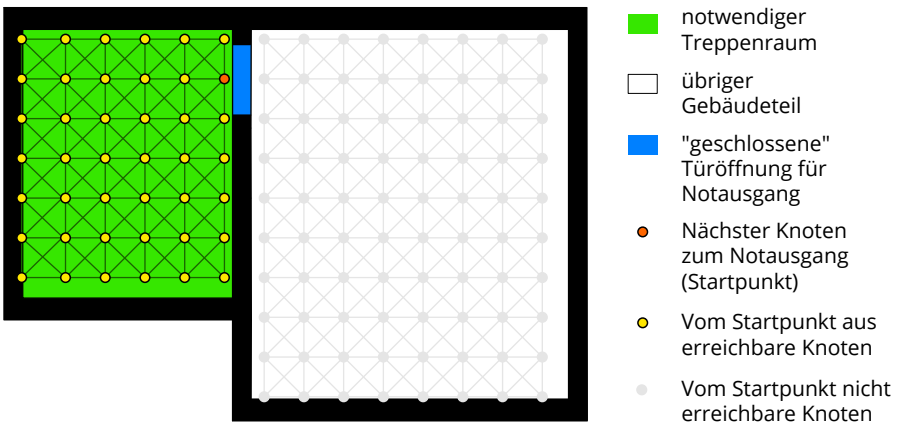



Abbildung 6.15: Nicht erreichbare Knoten aufgrund „geschlossener“ Wand

rissgeometrie mit offenen Notausgängen, als auch eine mit geschlossenen herzustellen. So können die Polygon- und Kürzeste-Pfad-Ermittlungen jeweils auf Basis ihrer spezifischen Grundrissdaten erfolgen.

Im nächsten Schritt werden die Öffnungen in den Wandelementen hergestellt. Wie oben bereits erwähnt, wird der Öffnungsalgorithmus zweimal angewendet; jeweils auf eines der Floor-Objekte mit den Wandelementen. Unterschieden wird dies über das Attribut *isFireExit* bei den Öffnungs-PlanElement-Objekten. Zuerst wird aus dem Floor-Objekt mit

[38] Dieses Polygon wird für den Punkt-in-Polygon-Test benötigt, bei welchem geprüft wird, ob ein Rasterknoten innerhalb oder außerhalb des Gebäudes liegt. Es wird mithilfe einer Java-Methode automatisch erkannt.

den Öffnungselementen ein PlanElement-Objekt entnommen [39] und von diesem die *relatedElementID* ausgegeben, welche entsprechend zu einem Wandelement gehört. Dieses Wandelement wird aus dem Floor-Objekt der Wände gefiltert, sodass nun die Wand und ihre zugehörige Öffnung vorliegen.

Das Wandelement wird nun anhand des Öffnungselements geteilt, sodass daraus zwei neue Wandelemente entstehen (siehe Abbildung 6.16). Hierzu werden die Schnittpunktkoordinaten (in der Abbildung rot dargestellt) zwischen den *LineSegment*-Strecken [40] des Wand- und des Öffnungselements ermittelt. Aus jeweils zwei der Wandkoordinaten und zwei der

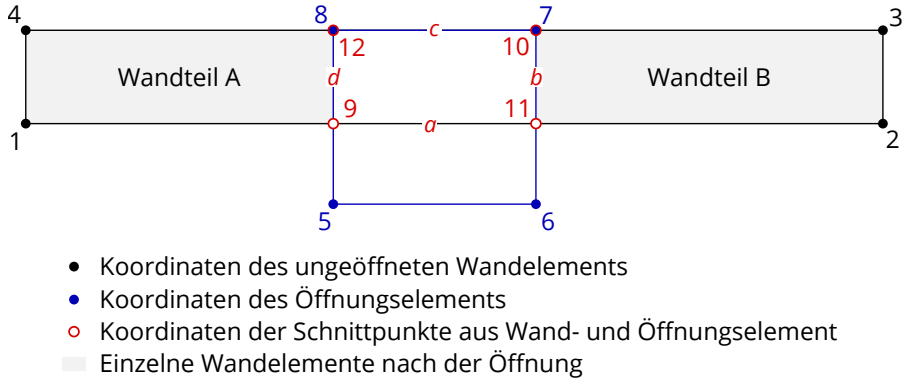



Abbildung 6.16: Darstellung eines Wand- und des zugehörigen Öffnungselements sowie die Aufteilung nach der Öffnung, horizontal

Schnittpunktkoordinaten lassen sich die beiden neuen Wandelemente zusammensetzen. Es muss jedoch noch eine Zuordnung erfolgen, welche dieser Koordinaten zusammen die korrekten Teilelemente erzeugen. Nur bei den PlanElement-Objekten der Wand und des Öffnungselements ist die geometrische Reihenfolge und damit die Lage der Eckpunkte zueinander bekannt, während die Lage der Schnittpunkte zu den Eckpunkten der Wand für den Algorithmus noch unbekannt ist. So steht nicht fest, zwischen welchen Punkten das Öffnungselement liegt (vergleiche dazu die Abbildungen 6.16 und 6.17 – es kann zwischen den Punkten 1–2, aber auch 2–3 liegen). Zudem werden die Schnittpunkte nicht in geometrischer Reihenfolge (so wie die Punkte des Wand- und des Öffnungselements) ermittelt, da dies aufgrund der unbekannten Lage des Öffnungselements nicht möglich ist.

Um die Schnittpunkte den übrigen Eckpunkten korrekt zuzuordnen, erfolgt eine Längenberechnung verschiedener Strecken zwischen diesen Punkten. Zunächst werden zwei Arrays erzeugt, in welchen die Punkte (in der richtigen geometrischen Reihenfolge) der beiden Wandteile gespeichert werden. Grundsätzlich gilt, dass sich der erste sowie der dritte Punkt in einem Rechteck immer gegenüberliegen. Da das Öffnungselement auch immer zwischen diesen Punkten liegt, lässt sich daraus folgern, dass diese beiden Punkte immer zu unterschiedlichen Wandelementen gehören. Sie stellen somit in ihrem Wandelement jeweils den ersten Punkt dar. Von diesem aus (im Folgenden wird der Punkt 1 aus den Abbildungen 6.16 und 6.17 betrachtet) werden Strecken zu den Wandpunkten 2 und 4 gezogen.

[39] An dieser Stelle findet die Unterscheidung zwischen den beiden möglichen Grundrissen statt. Für den Grundriss, dessen Notausgänge geschlossen bleiben sollen, werden diejenigen Öffnungs-PlanElement-Objekte nicht entnommen, deren *isFireExit*-Attribut auf *true* gesetzt ist. Für den vollständig offenen Grundriss werden alle Öffnungs-PlanElement-Objekte entnommen.

[40] Durch den Konstruktor von *PlanElement* werden bei Eingabe der Eckpunktkoordinaten Vektoren als Objekte des Typs *LineSegment* erzeugt, welche die Verbindungen zwischen diesen Eckpunktkoordinaten darstellen. Siehe dazu auch Abbildung B.1 in Anhang B.1.

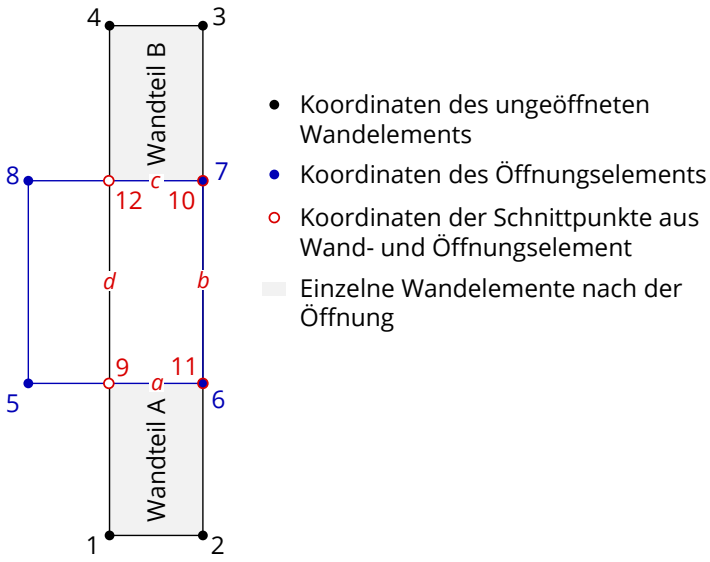



Abbildung 6.17: Darstellung eines Wand- und des zugehörigen Öffnungselements sowie die Aufteilung nach der Öffnung, vertikal

Es gehört derjenige Punkt zum selben Wandteil wie Punkt 1, dessen Strecke zu Punkt 1 nicht von einer Strecke $a$ bis $d$ geschnitten wird. Dieser Punkt stellt den zweiten Punkt des Wandelements dar (in Abbildung 6.17 ist dies zufällig auch Punkt 2 des Wandelements, in Abbildung 6.16 wäre das der Punkt 4).

Danach werden die Strecken zu den Schnittpunkten untersucht. Es gehören die beiden Schnittpunkte demselben Wandelement an, deren Strecke zu Punkt 1 nur jeweils einmal von einer Strecke $a$ bis $d$ geschnitten wird. In Abbildung 6.17 sind dies beispielsweise die Schnittpunkte 9 und 11, da ihre Strecken zum Punkt 1 jeweils nur einmal von der Strecke $a$ geschnitten werden. Die Strecken der Schnittpunkte 10 und 12 zum Punkt 1 schneiden jeweils die Strecken $a$ und $c$, also insgesamt zweimal. Sind die beiden Punkte bekannt, die zum selben Wandelement wie Punkt 1 gehören, ist die geometrische Reihenfolge in dem Rechteck zu prüfen. Dafür werden erneut vom Punkt 1 aus Strecken zu diesen beiden Punkten erzeugt (als Beispiel hier die Punkte 9 und 11 aus Abbildung 6.17) und die Längen dieser verglichen. Bei dem Punkt mit der längeren Strecke zu Punkt 1 (hier 11) handelt es sich immer um den gegenüberliegenden Punkt, sodass dieser an dritter Stelle der Reihenfolge kommt. Gleichzeitig ist dann auch bekannt, dass der andere Punkt (hier 9) an vierter Stelle der geometrischen Reihenfolge steht. Analog wird diese Vorgehensweise auch für den gegenüberliegenden Wandpunkt (hier 3) durchgeführt, da die geometrische Reihenfolge der Punkte in dessen Wandelement ebenfalls bestimmt werden muss.

Abschließend müssen die neuen Wandteile noch dem Floor-Objekt hinzugefügt werden. Dafür erhält das bestehende (noch „ungeöffnete“) Wandelement die Koordinaten eines Wandteils, während das andere Wandteil als neues PlanElement-Objekt die übrigen Attribute des Wandelements erhält.

Als letzte Handlung in diesem Programmabschnitt wird die Aufteilung der Elemente anhand ihrer Geschosse vorgenommen. In jedem PlanElement- und SingleRoom-Objekt ist der z-Versatz des lokalen Koordinatensystems des Geschosses zum globalen Koordinatensystem hinterlegt, sodass die Elemente anhand dieses Wertes den Geschossen zugeordnet werden können. Für jede Bauteilgruppe (Plan-Element-Objekte der Wände (inkl. Notausgangsöffnung), Plan-Element-Objekte der Wände ohne Notausgangsöffnung, PlanElement-

Objekte der Öffnungen und SingleRoom-Objekte) wird ein Floor-Array bzw. Rooms-Array mit der Dimension der Anzahl der Geschosse erzeugt. Die Elemente werden dann entsprechend ihres *globalZOffset*-Attributes einem Index dieses Arrays zugeordnet. Zuletzt werden all diese Arrays in einem Objekt der Klasse *Building* [41] zusammengefasst, welches an den nächsten Programmteil übergeben wird.

### 6.4.3 Rastererzeugung und kürzeste-Pfade-Ermittlung

In diesem Programmteil findet die Rastererzeugung, die Prüfung möglicher Verbindungen der Rasterknoten zueinander, und schließlich die Rettungswegermittlung statt. Der Ablauf der Rasterung und Verbindungsprüfung entspricht im Wesentlichen der in Abschnitt 4.3.2 (Seite 38) vorgestellten Methodik und wird daher hier nicht nochmals detaillierter erläutert. Dafür wird im Folgenden auf die spezifischen Besonderheiten der Umsetzung in dem Softwareprototypen näher eingegangen, da bestimmte (in diesem Programmabschnitt eingeführte) Klassen und Objekte auch für die Ausgabe von Bedeutung sind.

Zu Beginn werden die Geometriedaten aus dem Building-Objekt eingelesen und darauf basierend die Rasterknoten erzeugt (Schritte 1 – 6 in Abbildung 4.12). Bei den Knoten handelt es sich um Instanzen der selbstdefinierten Klasse *Vertex*, welche neben den in Abschnitt 4.3.2 genannten Attributen (*exist*, $x$, $y$) weitere definiert, in denen Informationen zu erreichbaren Knoten, den Rettungswegen und beinhaltenden Räumen vorliegen. Jedes Vertex-Objekt besitzt einen *adjacencyIndex*, welcher eine fortlaufende Nummer zur eindeutigen Identifikation ist. Die Attribute *exist*, $x$, $y$ wurden bereits genannt, wobei in der Klasse Vertex das Attribut *exist* als *isReachable* bezeichnet wird. [42] Das *connectedRoom*-Attribut gibt an, in welchem Raum sich der Knoten befindet, was im nächsten Programmabschnitt zugeordnet wird (siehe Abschnitt 6.4.4). Die Attribute *occupancy* und *description* beziehen sich nur auf diejenigen Knoten, die den Notausgängen am nächsten liegen und daher als Startknoten der kürzesten-Pfade-Ermittlungen dienen. Sie repräsentieren in der Ausgabe die Notausgänge, sodass in ihnen die Anzahl der auf diesen Notausgang entfallenden Personen sowie die Bezeichnung des Notausgangs gespeichert wird. Bei den übrigen Knoten werden diese Attribute leer gelassen.

In der Vertex-Klasse werden darüber hinaus zwei Attribute definiert, die sich auf weitere selbstdefinierte Klassen beziehen. Eines davon ist das Attribut *edges*, welches eine ArrayList des Typs *Edge* beinhaltet. Mit den *edges*-Attributen werden die existierenden Verbindungen der Knoten untereinander angegeben. Durch die Rasterung kann jeder einzelne Knoten bis zu acht erreichbare Nachbarknoten haben, zu denen eine direkte Verbindung besteht. [43] Diese Verbindungen werden über die Klasse *Edge* repräsentiert. Ein Edge-Objekt hat drei Attribute: *OwnVertex* bezieht sich rekursiv auf den Knoten, welcher das Edge-Objekt beinhaltet, *neighbourVertex* gibt den Knoten an, zu welchem die Verbindung hingeführt wird. Die Streckenlänge *distance* wird als Fließkommazahl angegeben. Dies entspricht den Informationen, die in einer Adjazenzmatrix gespeichert werden, um die Verbindungen innerhalb eines Graphen auszudrücken. In dem Softwareprototypen werden diese Verbin-

[41] *Building* ist eine selbstdefininerte Klasse, die nur der leichteren programminternen Handhabung dient. In Anhang B.3 ist das UML-Klassendiagramm zu *Building* dargestellt.

[42] Mit beiden Bezeichnungen ist dasselbe gemeint. Es wird angegeben, ob ein Knoten grundsätzlich erreichbar ist, d.h., dass er innerhalb des Raumpolygons liegt sowie nicht auf oder innerhalb eines Wandelements.

[43] In dem *edges*-Attribut können auch mehr als acht Verbindungen gespeichert werden, was als Vorbereitung auf eine Erweiterung des Softwareprototypen hinsichtlich der Genauigkeit der Rastermethode (siehe Abschnitt 4.4) dient.

dungen jedoch nicht in einer Adjazenzmatrix, sondern entsprechend direkt in den Knoten gespeichert, was aus objektorientierter Sicht das bessere Datenschema darstellt. So hat eine Adjazenzmatrix sehr viele Null-Einträge (siehe Abschnitt 4.3.2 – Allgemeines), die während des kürzesten-Pfad-Algorithmus dennoch immer wieder überprüft würden. Bei dem objektorientierten Ansatz hingegen werden direkt die existierenden Nachbarknoten überprüft, was in Verbindung mit den in dem Knoten gespeicherten *escapeRoutes* (siehe nächster Absatz) einen effizienteren Programmablauf ermöglicht. Darüber hinaus können diese Informationen an jeder Stelle des Programms zu jeder Zeit abgefragt werden, sodass sich diese Datenstruktur auch als sehr flexibel erweist. Die Erzeugung der Edge-Objekte für jeden Knoten entspricht den Schritten 15 – 26 aus Abbildung 4.12 mit zwei Unterschieden: Bei einer existierenden Verbindung zwischen dem betrachteten Knoten $k_i$ zum Nachbarknoten $k_m$ wird kein Eintrag in eine Adjazenzmatrix (Schritt 22), sondern ein Edge-Objekt mit dem Nachbarknoten erzeugt. Existiert keine Verbindung (Schritt 24), wird auch kein Edge-Objekt erzeugt.

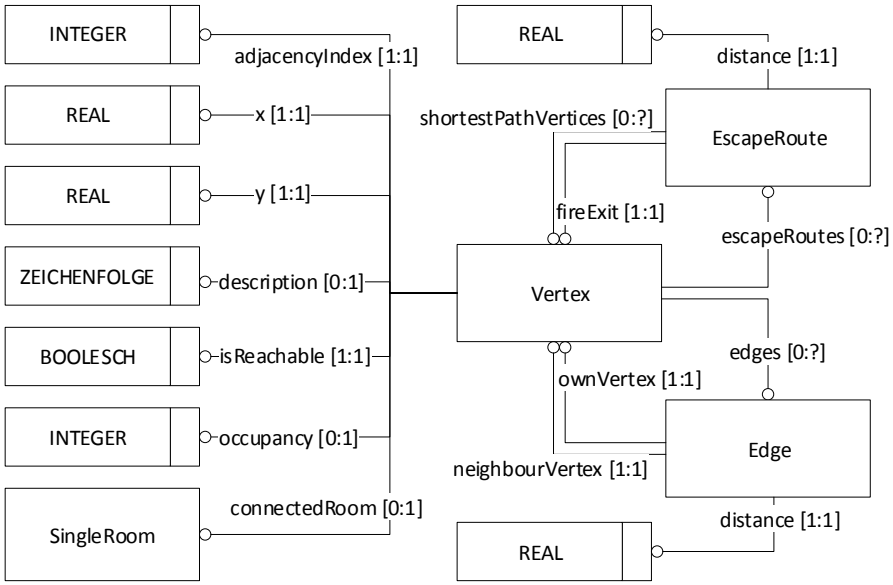



Abbildung 6.18: EXPRESS-G Darstellung der Klassen Vertex, Edge und EscapeRoute mit ihren Attributen

Wie im vorherigen Absatz erwähnt, besitzen Objekte der Klasse Vertex ein weiteres Attribut, was eine ArrayList vom Typen *EscapeRoute* ist. Auch dabei handelt es sich um eine selbstdefinierte Klasse, welche drei Attribute definiert, die einen Rettungsweg vollständig beschreiben. In einem Vertex-Objekt können unbegrenzt viele Rettungswege gespeichert werden, praktisch sind es so viele, wie es Notausgänge in dessen Geschoss gibt. Relevant ist letztlich nur der Nachweis von zwei unabhängigen Rettungswegen, was in der Ausgabe geschieht (Abschnitt 6.4.5). Das Attribut *fireExit* eines EscapeRoute-Objekts gibt den Knoten (Vertex-Objekt) an, welcher den Notausgang repräsentiert und somit den Startknoten des kürzesten-Pfad-Algorithmus bildet. Das Attribut *shortestPathVertices* ist eine ArrayList des Typs Vertex und beinhaltet alle Knoten, die im Verlauf des Rettungswegs von dem aktuell betrachteten Knoten liegen. Mit der Fließkommazahl *distance* wird die Streckenlänge des gesamten Rettungswegs angegeben. Diese Informationen dienen sowohl der Auswertung, da die Streckenlänge und der Verlauf eines Rettungsweges darüber ausgegeben werden kann, aber auch dem kürzesten-Pfad-Algorithmus während der Berechnung.

Der in diesem Softwareprototypen zur kürzesten-Pfad-Ermittlung eingesetzte *Dijkstra-Algorithmus* ermittelt immer von einem aktuell betrachteten Knoten aus alle weiteren erreichbaren Knoten, wofür er auf die *edges*-Attribute des betrachteten Knoten zurückgreifen kann. Während des Ablaufs muss er bereits bekannte (und noch nicht besuchte) Knoten und deren Pfadlängen zum Startpunkt zwischenspeichern (siehe Abschnitt 4.2.1). Hierzu kann er die *EscapeRoute*-Objekte nutzen, da dort die aktuelle Länge jedes Pfades zu einem Zielknoten gespeichert ist. Wird ein neuer Knoten als „besucht“ markiert, kann dessen Vorgängerknoten der *escapeRoutes*-Liste hinzugefügt und die Gesamtlänge des Pfades bis zum Zielknoten aktualisiert werden. Hier offenbart sich der große Vorteil dieses objektorientierten Ansatzes gegenüber einer statischen Adjazenzmatrix, da der Algorithmus mit den vorhandenen Objekten arbeitet, nur wenige Hilfsobjekte zum Zwischenspeichern erstellt werden müssen und nach Beendigung des Algorithmus sofort alle Ergebnisse vorliegen.

Die kürzeste-Pfad-Ermittlung wird für jeden Notausgang desselben Geschosses durchgeführt, wodurch sich insgesamt ein erhöhter Rechenaufwand ergibt. Anschließend liegen alle Rettungswege für jeden Knoten vor und es kann die Aufbereitung dieser Informationen vorgenommen werden.

### 6.4.4 Aufbereitung der Rettungswegermittlung

Aus der kürzesten-Pfad-Ermittlung liegen für jeden (erreichbaren) Knoten ein oder mehrere Rettungswege vor, abgespeichert in der ArrayList seines *escapeRoutes*-Attributs. Sie sind aber noch unsortiert, sodass die ersten und zweiten Rettungswege der einzelnen Knoten an dieser Stelle noch nicht bestimmt sind. Die Sortierung erfolgt anhand der Rettungswegläng (*distance*) der jeweiligen EscapeRoute-Objekte. Da nur der erste Rettungsweg eine maximale Länge nicht überschreiten darf (siehe Abschnitt 2.1.2 – *Notwendiger Treppenraum und maximale Rettungsweglänge*) ist es naheliegend, den kürzesten unter allen ermittelten Rettungswegen auszuwählen. Dieser wird in der ArrayList an erster Stelle eingetragen. Der Rettungsweg mit der zweit-kürzesten Strecke wird an der zweiten Stelle in der ArrayList eingetragen und gibt damit den zweiten Rettungsweg an. [44]

Anschließend erfolgt die Zuordnung der Knoten zu den Räumen, in denen sie liegen. Da die umschlossene Grundfläche eines Raums in dem SingleRoom-Objekt bereits als Polygon vorliegt, kann der Punkt-in-Polygon-Test relativ einfach erfolgen. Dabei wird ein Knoten betrachtet und geprüft, in welchem der vorliegenden Räume des Geschosses er liegt. Sobald dieser Raum ermittelt wurde, wird er in das *connectedRoom*-Attribut des Knotens eingetragen. Ebenso wird der Knoten der ArrayList des *containedVertices*-Attributs des Raums hinzugefügt, sodass auch ermittelbar ist, welche Knoten in einem Raum liegen. Damit kann im nächsten Schritt untersucht werden, von welchem Knoten (und damit von welcher Stelle) aus in einem Raum die maximale Wegstrecke zu einem Notausgang besteht.

Um die maximale Wegstrecke aus einem Raum zum nächstgelegenen Notausgang zu ermitteln, müssen die enthaltenen Knoten nach der Streckenlänge ihrer ersten Rettungswege sortiert werden. Da die Rettungswege der jeweiligen Knoten bereits sortiert vorliegen, ist bekannt, dass der erste Eintrag in der ArrayList des *escapeRoutes*-Attributs den ersten Rettungsweg und damit die kürzeste Strecke zum nächstgelegenen Notausgang angibt. Anhand dieses Eintrags werden die Knoten in der *containedVertices*-ArrayList der Län-

[44] Existieren mehr als zwei Rettungswege für einen Knoten, so ist es im Prinzip egal, welcher davon als zweiter Rettungsweg ausgewählt wird. Da die ArrayList nach den Rettungsweglängen aufsteigend sortiert wird, ergibt sich diese Zuteilung automatisch. Es ist aber aus brandschutztechnischer Sicht sicherlich ein Vorteil, wenn der zweite Rettungsweg ebenfalls möglichst kurz ist.

ge nach absteigend sortiert. Durch die Sortierung der EscapeRoute-Objekte innerhalb der ArrayList eines jeden Knotens ist auch berücksichtigt, dass unterschiedliche Knoten innerhalb eines Raums verschiedene Notausgänge als kürzesten ersten Rettungsweg haben. Abbildung 6.19 zeigt hierzu ein Beispiel. Der in einem Raum längste Weg bis zu einem Ausgang ist theoretisch der in rot eingezeichnete Weg. Dieser Weg kann für den Nachweis

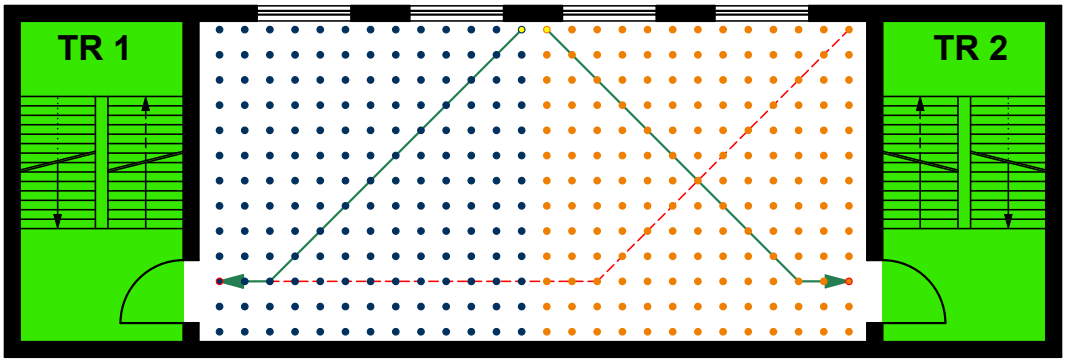



Abbildung 6.19: Aufteilung der Knoten eines Raums auf verschiedene Notausgänge

der maximalen Rettungswegländge ausreichend sein. Solange er kürzer als 35 m ist, wird der Rettungswegnachweis damit auch erfüllt. Problematisch wird es, wenn dieser Weg länger ist und die bauordnungsrechtlichen Vorschriften nicht mehr eingehalten werden können. Daher ist immer zu prüfen, ob alle Stellen durch das Erreichen von verschiedenen Notausgängen mindestens einen davon in weniger als 35 m erreichen können. So ist für die blau eingezeichneten Knoten in Abbildung 6.19 TR 1 der am kürzesten erreichbare Rettungsweg, für die orangenen Knoten TR 2. Grün eingezeichnet sind die längsten Rettungswegstrecken jedes Notausgangs unter der Berücksichtigung, dass der jeweils andere Notausgang nicht auf kürzerer Strecke erreichbar ist. Da in jedem Knoten der erste Rettungsweg durch den kürzesten Pfad zum nächstgelegenen Notausgang bestimmt worden ist, kann somit die Situation, dass ein Knoten einen kürzeren zweiten Rettungsweg hat (wie es ja bei dem Endknoten der roten Linie aus Abb. 6.19 der Fall ist, da TR 2 für diesen kürzer erreichbar ist) niemals eintreten. Als längster Rettungsweg dieses Raums würde entweder der zu dem am weitesten entfernten blauen oder orangenen Knoten angegeben werden, je nachdem, welcher zuerst ermittelt wurde. [45]

Mit diesen Zuordnungen und Sortierungen sind die Daten innerhalb der Vertex- und SingleRoom-Objekte soweit aufbereitet, dass sie aussagekräftige Ergebnisse liefern und im letzten Programmteil ausgegeben werden können.

### 6.4.5 Ausgabe

Grundsätzlich besteht die Ausgabe aus einem textlichen Teil sowie einer visuellen Aufbereitung. In der Textausgabe werden für jeden Raum der erste und zweite Rettungsweg (inkl. ihrer Entfernungen zu den jeweiligen Notausgängen) angegeben. Zusätzlich werden für alle Notausgänge die zu diesen führenden (ersten) Rettungswege mit der größten Distanz angegeben. [46] Darüber hinaus erfolgt auch eine Angabe der Summe aller Personen,

[45] Programmintern wird nur ein neuer maximaler Rettungsweg für einen Raum festgelegt, wenn dessen Entfernung zu einem Notausgang größer ist als eine andere. Sind die beiden Entfernungen gleich (so wie im Beispiel aus Abb. 6.19), bleibt der zuerst eingetragene maximale Rettungsweg bestehen.

[46] Unter der Berücksichtigung, dass für den Knoten mit der maximalen Distanz zu einem Notausgang kein zweiter Rettungsweg mit einer kürzeren Entfernung zu einem anderen Notausgang existiert.

die auf den jeweiligen Notausgang angewiesen sind. Abbildung 6.20 zeigt beispielhaft die Textausgabe eines Geschosses mit zwei Räumen.

```
************************************************************
                    2. Obergeschoss
************************************************************

Rettungswege für jeden Raum:

R-001 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR2-2OG in maximal 28.03 m.
2. Rettungsweg: TR1-2OG in maximal 29.38 m.

R-002 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR2-2OG in maximal 15.08 m.
2. Rettungsweg: TR1-2OG in maximal 38.41 m.

____________________________________________________________

Längste Rettungswege in diesem Geschoss:

Von Knoten 626 (x: 1.409 y: 0.377) zu TR2-2OG.
in Raum: R-001 | Büro
Maximale Rettungsweglänge: 28.03 m.
2. Rettungsweg: TR1-2OG (Länge: 29.38 m.)


Summe des Personenstroms je Rettungsweg:

TR2-2OG: 3 Personen.
```

Abbildung 6.20: Beispielhafte Textausgabe

Für die Ausgabe der Rettungswege jedes Raums liegen alle Daten in den jeweiligen SingleRoom-Objekten vor. Raumbezeichnung, Raumnummer und Belegung sind direkt als Attribute enthalten, für die Angaben des ersten und zweiten Rettungswegs werden die Daten des ersten und zweiten Elements der *escapeRoutes*-ArrayList dieses SingleRoom-Objekts ausgegeben.

Die Ausgabe der längsten Rettungswege pro Geschoss für jeden Notausgang kann nicht direkt erfolgen, da diese Daten nicht aufbereitet in einem Objekt hinterlegt sind. Es müssen zuvor alle Knoten anhand ihres Notausgangs des ersten Rettungswegs aufgeteilt werden, sodass für jeden Notausgang bekannt ist, von welchen Knoten aus deren erster Rettungsweg zu diesem Notausgang führt. [47] Anschließend werden die auf die verschiedenen Notausgänge aufgeteilten Knoten nach ihrer Entfernung bis zu ihren Notausgängen sortiert, um die Knoten mit der jeweils größten Distanz zu erhalten. Letztlich sind diese Knoten auch diejenigen, welche für den Rettungsnachweis ausschlaggebend sind. Ist deren maximale Rettungsweglänge eingehalten, gilt dies automatisch auch für alle anderen Knoten und damit Stellen in einem Geschoss.

Der Personenstrom je Rettungsweg ist ebenfalls noch in diesem Programmabschnitt zu addieren. Hierfür werden zunächst alle Räume des Geschosses aufgelistet und von jedem der erste Knoten aus der *containedVertices*-ArrayList ausgegeben. [48] Da der Notausgang, zu dem der erste Rettungsweg dieses Knotens führt, ermittelbar ist, können die verschiedenen Räume entsprechend diesen Notausgängen aufgeteilt werden. Da nun bekannt ist, aus welchen Räumen Rettungswege zu den jeweiligen Notausgängen führen, können entsprechend die Personen pro Notausgang addiert und ausgegeben werden.

[47] Aus Abschnitt 6.4.4 ist bekannt, dass der erste Rettungsweg immer zum nächstgelegenen Notausgang führt, sodass auch der Problematik aus Abbildung 6.19 vorgebeugt wird.

[48] Da die Knoten in dieser ArrayList bereits nach ihrer Rettungsweglänge zum nächstgelegenen Notausgang sortiert sind (siehe Abschnitt 6.4.4), besteht von dem ersten Knoten in der ArrayList die größte Entfernung innerhalb des Raums bis zum nächstgelegenen Notausgang.

Für die visuelle Darstellung der Rettungswegermittlung werden die Daten aus der Ermittlung der längsten ersten Rettungswege pro Geschoss für jeden Notausgang zugrunde gelegt. Wie bereits erläutert wurde sind diese Rettungswege für den Rettungswegnachweis maßgebend, sodass deren Angabe ausreichend ist. Bei der grafischen Aufbereitung wird der Übersichtlichkeit halber auf weitere Darstellungen anderer Rettungswege verzichtet.

## 6.5 Evaluation

### 6.5.1 Beispielgebäudemodell

Zur Evaluation der Ergebnisse ist eine IFC-Beispieldatei exportiert worden, auf welche in dieser Arbeit bereits mehrfach Bezug genommen wurde. Der Export erfolgte aus einem Gebäudemodell, welches mit „ArchiCAD 21“ erstellt worden ist. Bei dem Beispielobjekt

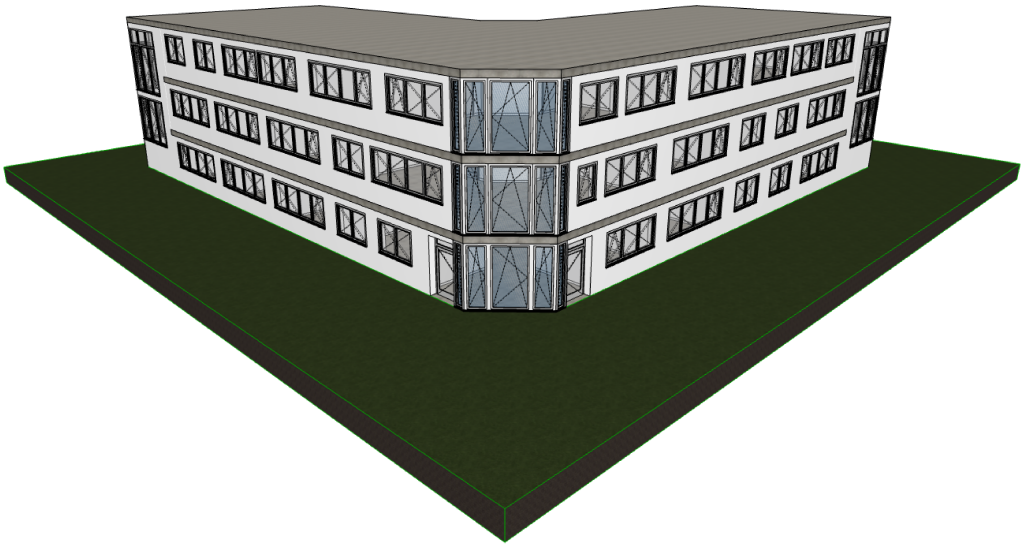


Abbildung 6.21: 3D-Ansicht des Beispielgebäudemodells

handelt es sich um ein dreigeschossiges Bürogebäude, welches einen „L-förmigen“ Grundriss mit einer Bruttogrundfläche von 646 $m^2$ pro Geschoss aufweist. Die maximalen Abmessungen betragen in den beiden Hauptrichtungen jeweils rund 31,5 m. Es existieren zwei notwendige Treppenräume an den Gebäudeenden, zu denen es in jedem Geschoss einen Zugang gibt. Im Erdgeschoss sind zwei weitere direkte Ausgänge ins Freie vorhanden, die ebenfalls als Notausgang dieses Geschosses betrachtet werden.

Im Erdgeschoss sind überwiegend Büroräume mit einer Belegung von ein bis drei Personen angeordnet. Es existieren aber auch Räume, in denen keine Personenbelegung vorgegeben wurde (keine Aufenthaltsräume, wie z.B. WC-Räume) und ein durch Wände abgetrennter Bereich, der nicht zugänglich ist (was z.B. einen Versorgungsschacht darstellen könnte). Der Grundriss des ersten Obergeschosses entspricht genau dem des Erdgeschosses, mit der Ausnahme, dass hier anstelle des Foyers mit den direkten Ausgängen ins Freie ein großer Pausenraum mit 20 Personen vorhanden ist. Die Personenbelegung der übrigen Räume ist gegenüber dem Erdgeschoss leicht abgeändert. Das zweite Obergeschoss weist einen veränderten Grundriss auf; auffälligste Änderung gegenüber den anderen beiden Geschossen ist der Konferenzraum mit einer Belegung von 32 Personen, in welchem zusätzlich eine

freie Wand (ohne Anschluss an andere Bauteile) vorhanden ist. Abbildung 6.22 zeigt den Erdgeschossgrundriss sowie die Notausgänge; eine vollständige Darstellung aller Rauminformationen (Raumbezeichnung, Raumnummer, Belegung) ist für alle Geschosse in Anhang C.1 in den Abbildungen C.1 – C.3 gegeben.

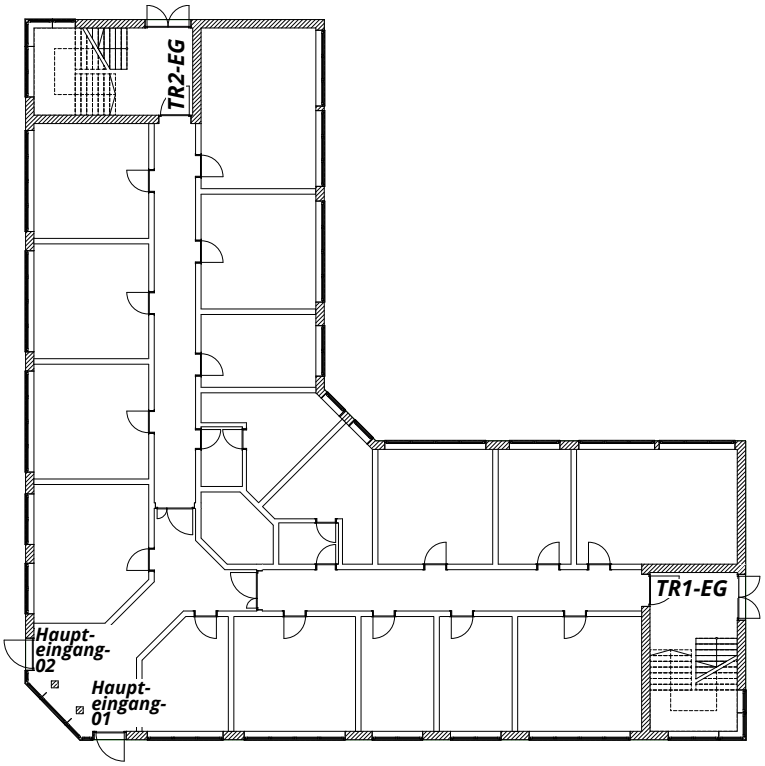



Abbildung 6.22: Erdgeschossgrundriss des Beispielgebäudemodells mit Beschreibung der Notausgänge

Das Beispielgebäude ist so konstruiert, dass alle Elemente, welche durch den Softwareprototypen betrachtet werden (siehe Abschnitt 5.1), auch in dem Gebäude vorkommen. Bei der Einbringung von Stützen wurde sich auf jeweils zwei pro Geschoss beschränkt, da sie in der Regel keinen signifikanten Einfluss auf eine Rettungswegermittlung haben. Es sind Wände mit unterschiedlichen Dicken und Materialien vorhanden, die zum Teil auch schräg angeordnet sind. Anhand dessen kann der korrekte Linienzusammenschluss für die Schnittpunktberechnung getestet werden. Neben den Türöffnungen sind auch Fenster vorhanden, die durch den Softwareprototypen nicht als Öffnungen erkannt werden sollten.

Die Notausgänge sind mittels Bauteileigenschaften der Türen definiert worden, welche als IfcPropertySet in die IFC-Datei übersetzt werden. Der Softwareprototyp sollte diese erkennen und die jeweils nächstgelegenen Rasterknoten als Startpunkte für die kürzeste-Pfad-Ermittlung auswählen. Die Räume wurden mittels des „Raumflächen-Werkzeug“ von ArchiCAD erzeugt, die Angabe der Belegung erfolgte ebenfalls über die Bauteileigenschaften. Der Softwareprototyp sollte die einzelnen Räume erkennen und die Personenströme je Notausgang korrekt addieren.

Für die Evaluation wurde die grafische Ausgabe dahingehend angepasst, dass auch das Raster, die vorhandenen Verbindungen zwischen den Knoten und das äußerste Polygon mit angezeigt werden. Im Praxiseinsatz sollten diese Elemente ausgeblendet sein, da sie für Rettungswegnachweise unerheblich sind. Zur Kontrolle von Zwischenergebnissen ist die Ausgabe mit Raster und Polygon jedoch erforderlich.

Auch wenn das Beispielgebäude ausschließlich zu Forschungszwecken im Rahmen dieser Arbeit entworfen wurde, könnte es auch, bei Anpassung kleinerer architektonischer Details, ein reales Gebäude sein. Somit ist eine sehr praxisnahe Evaluation des Softwareprototypen möglich.

### 6.5.2 Rechenzeit und Genauigkeit

Die Anzahl an Rasterknoten hat einen erheblichen Einfluss auf die Rechenzeit des Softwareprototypen. Ein zusätzlicher Knoten bedeutet acht bis sechzehn weitere zu untersuchende Verbindungen, die bei der Prüfung auf Kollision mit Wandstrecken sowie bei der kürzesten-Pfad-Berechnung zu berücksichtigen sind. Bei der Grundrissausdehnung handelt es sich um einen statischen Wert, der durch die Gebäudegröße vorgegeben ist. Der Abstand der Rasterknoten ist hingegen ein beeinflussbarer Parameter, sodass dieser unter Abwägung von Genauigkeit und Rechenzeit möglichst günstig ausgewählt werden muss.

Im Rahmen der Evaluation wurde anhand des Beispielobjektes auch der Einfluss des Knotenabstands auf die Rechenzeit ausgewertet (siehe Anhang C.4). Als maximaler Abstand ist 0,90 m gewählt worden, da ein noch größerer Wert dazu geführt hätte, dass der Abstand größer wäre als die Türöffnungen breit sind und ggf. bestimmte Räume nicht erreicht werden können. Bei einem Abstand von weniger als 0,3 m kam es bei Tests immer zu einem sogenannten „Stapelüberlauf“, was bedeutet, dass der Arbeitsspeicher des Computers ausgelastet war und die Berechnungen nicht durchgeführt werden konnten. Dementsprechend beschränken sich die Untersuchungen auf Knotenabstände zwischen 0,3 m und 0,9 m.

Anhang C.4 ist zu entnehmen, dass ein Knotenabstand von 0,3 m mit über 20 Minuten eine äußerst praxisferne Rechenzeit darstellt. Wird der Abstand allerdings nur um zehn Zentimeter auf 0,4 m vergrößert, so verringert sich die Rechenzeit deutlich auf 3,5 Minuten. Bei einem Abstand von 0,5 m beträgt sie noch eine Minute, bei 0,6 m nur noch 30 Sekunden. Auch wenn 30, 20 oder sogar 11 Sekunden (bei 0,9 m Abstand) einen sehr guten Wert darstellen ist dabei immer zu berücksichtigen, dass ein größerer Abstand unweigerlich eine höhere Ungenauigkeit mit sich bringen. Als guten Kompromiss zwischen Rechenzeit und Genauigkeit wird daher ein Rasterabstand von 0,5 m angesehen.

Es ist zu beachten, dass diese Messungen nur zur groben Einordnung dienen und keinen Aufschluss über die Rechenzeiten eines praxistauglichen Programms geben. Zudem sind sie nur anhand des Beispielgebäudemodells ermittelt worden, sodass bei einem Gebäude mit anderen Abmessungen zum Teil deutlich andere Rechenzeiten möglich wären. In dem Softwareprototypen sind keine besonderen Maßnahmen zur Rechenzeitoptimierung vorgenommen worden. Ein optimierter Algorithmus könnte eventuell schneller Ergebnisse liefern, was eine Verkürzung des Knotenabstands ermöglichen würde.

### 6.5.3 Beschreibung und Auswertung der Ausgabe

Die Ausgabe der Rettungswegermittlung für das Beispielobjekt bei einem Rasterknotenabstand von 0,5 m umfasst mehrere Seiten, weswegen sie daher dem Anhang beigefügt sind. Die Textausgabe beginnt im Abschnitt C.2.1 (ab Seite 126), die grafische Ausgabe für alle Geschosse ist in Abschnitt C.3.1 (ab Seite 134) dargestellt. Nachfolgend werden die wichtigsten Erkenntnisse zusammengefasst und ausgewertet.

Der grafischen Ausgabe ist zu entnehmen, dass alle relevanten Bauteile (Wände, Stützen und Türöffnungen) durch den Algorithmus erkannt und korrekt wiedergegeben wurden.

Ebenso sind Fensteröffnungen richtigerweise nicht erfasst und dargestellt worden. Bei den schräg verlaufenden Wandelementen ist zu erkennen, dass diese über die eigentlichen Wandbegrenzungen hinausragen. Dies ist dem Umstand geschuldet, dass sich die „Bounding-Box“

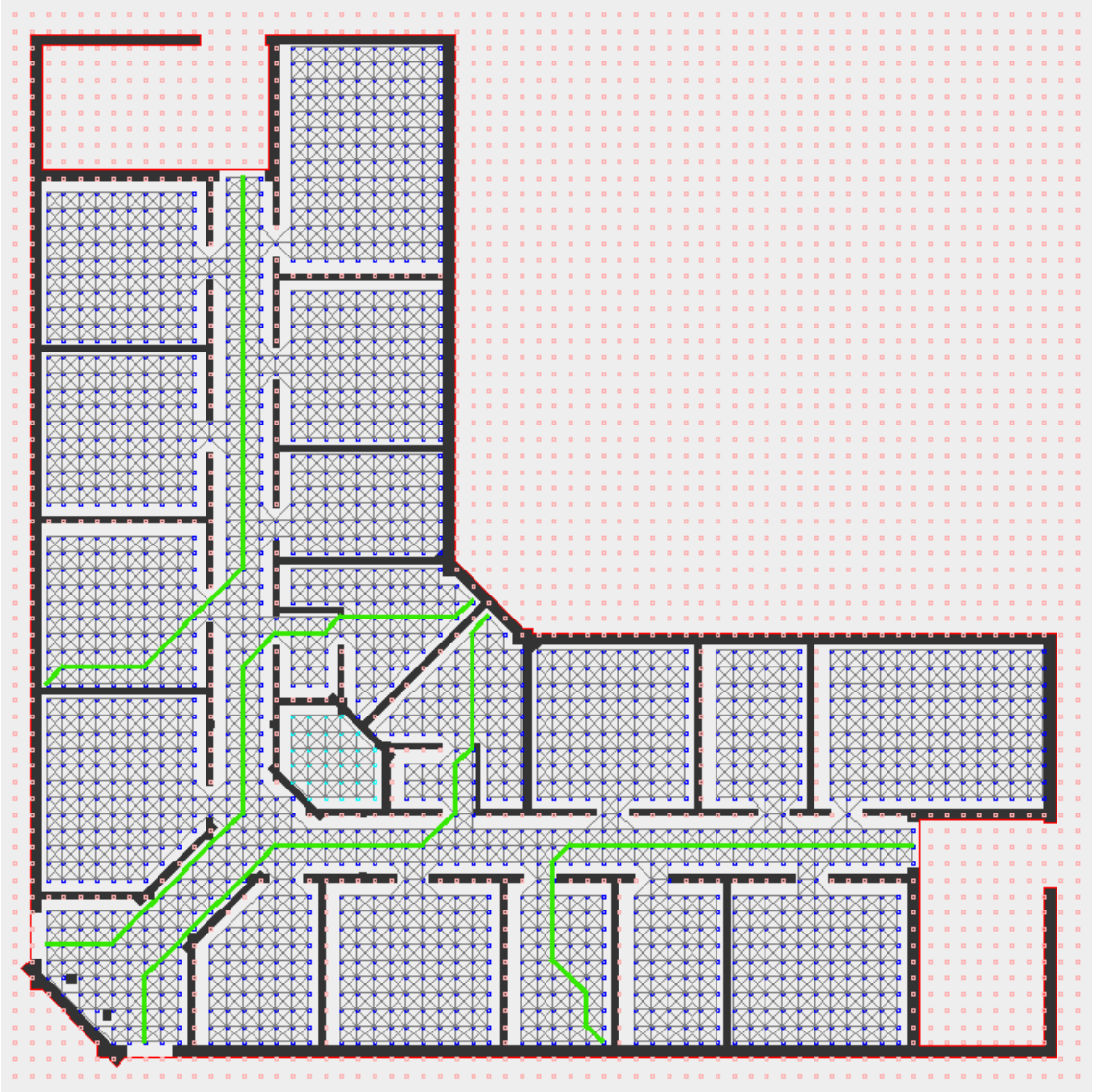


Abbildung 6.23: Grafische Ausgabe der Rettungswegermittlung für das Erdgeschoss

über die maximalen Abmessungen eines Bauteils erstreckt und immer rechteckige Winkel aufweist. Dadurch bilden sich diese Überstände, welche die Rettungswegermittlung aber nicht negativ beeinflussen. Bei einem praxistauglichen Programm würden ohnehin genauere Geometriedarstellungen zugrunde gelegt werden (siehe Abschnitt 5.2.3 – Weitere Klassen zur Geometriebeschreibung), sodass diese Situation dort nicht auftreten wird.

Weiterhin ist zu erkennen, dass alle Rasterknoten gleichmäßig verteilt und deren *isReachable*-Attribute korrekt gesetzt wurden. Die rosa eingefärbten Knoten werden nicht betrachtet, da sie entweder außerhalb des Grundrisspolygons oder auf einem Wandelement liegen. So ist auch erkennbar, dass der Grundriss an der richtigen Stelle von dem Polygon umschlossen wird (rote Linie). Blau eingezeichnet sind alle Knoten, welche die vorstehenden Bedingungen erfüllen. Eine Besonderheit stellen die hellblauen Knoten dar, welche die vorstehenden Bedingungen auch erfüllen, von den Notausgängen aber nicht erreichbar sind.

Die grauen Linien zeigen die existierenden Verbindungskanten zwischen den Knoten, welche nicht durch Bauteile o.Ä. unterbrochen werden. Auch hier ist zu erkennen, dass diese korrekt ermittelt worden sind.

Die grünen Linien stellen letztlich die Rettungswege von den am weitesten entfernten Punkten zu den nächstgelegenen Notausgängen dar. [49] Diese werden in der Textausgabe für jedes Geschoss unter den Punkt *Längste Rettungswege in diesem Geschoss* mit konkreten Zahlenwerten angegeben. Die dort aufgeführten Ergebnisse entsprechen im Wesentlichen den tatsächlichen Strecken, sodass auch die kürzeste-Pfad-Ermittlung korrekt ausgeführt wurde. Durch die wegen der Rasterung festgelegten Winkel der Laufrichtung ergeben sich hier längere Rettungswege als bei der händischen Rettungswegermittlung, was bereits in den Abschnitten 4.3.2 und 4.4 ausführlich erläutert wurde.

Im Rahmen weiterführender Untersuchungen zum Rettungswegverlauf konnten Unterschiede bei verschiedenen Abständen der Rasterknoten festgestellt werden. Abbildung 6.24 zeigt die grafische Ausgabe der Rettungswegermittlung bei einem Rasterknotenabstand von 0,42 m und 0,45 m. Zu erkennen ist, dass die Rettungswegführung bei einem Kno-

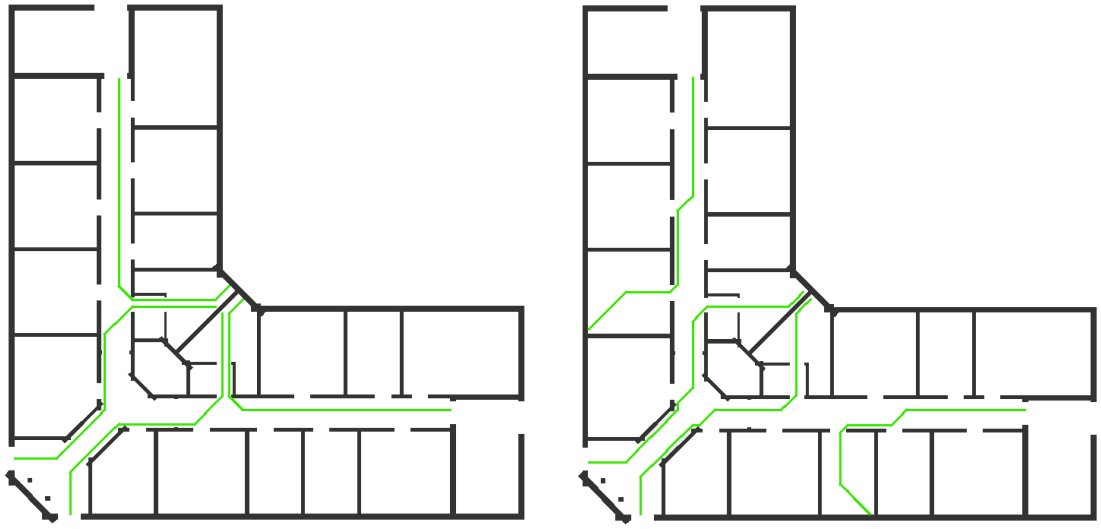


Abbildung 6.24: Rettungswegverlauf im Erdgeschoss bei einem Rasterknotenabstand von 0,42 m (links) und 0,45 m (rechts)

tenabstand von 0,45 m derer bei einem Abstand von 0,50 m ähnelt, während bei einem Knotenabstand von 0,42 m der von den beiden Treppenräumen entferntest gelegene Punkt jeweils in einem anderen Raum liegt. In Anhang C.2.2 und C.2.3 ist zusätzlich die Textausgabe für diese Rettungswege gegeben. Werden einzelne Rettungswege zu denselben Notausgängen bei unterschiedlichen Knotenabständen betrachtet, so sind zunächst deutliche Unterschiede zu erkennen. Bei einem Abstand von 0,42 m beträgt beispielsweise die maximale Rettungsweglänge zu *TR1-EG* 20,02 m, während sie bei dem Abstand von 0,50 m nur 17,33 m beträgt. Diese Unterschiede deuten jedoch auf kein Problem hin. Betrachtet man alle Längen bei allen Abständen, so ist festzustellen, dass die insgesamt maximale Rettungsweglänge unter diesen immer etwa 20 m beträgt. Das bedeutet, dass durch den Algorithmus die entferntest gelegenen Punkte grundsätzlich gefunden werden und sich durch die unterschiedlichen Knotenabstände lediglich die „Einzugsbereiche“ verschieben, in denen die Rettungswege der übrigen Knoten zu den jeweiligen Notausgängen führen.

[49] Unter der Berücksichtigung, dass es für diese Knoten keinen kürzeren Rettungsweg zu einem anderen Notausgang gibt.

Aus der Textausgabe (0,50 m Knotenabstand, Anhang C.2.1) gehen die zweiten Rettungswege sowie die Informationen zu den einzelnen Räumen hervor. Für letztere kann beim Abgleich mit den Eingangsdaten (Anhang C.1) festgestellt werden, dass die Raumbezeichnungen und Belegungen richtig zugeordnet worden sind. Auch die Angaben der Personenströme je Rettungsweg entsprechen den zu erwartenden Ergebnissen. Die von dem Softwareprototypen ermittelten zweiten Rettungswege wurden für jeden Raum ebenfalls korrekt ermittelt. Somit kann die Lauffähigkeit des Softwareprototypen auch für die Raumanalyse sowie die Ermittlung zweiter Rettungswege bestätigt werden.

Dass im ersten und zweiten Obergeschoss auch Rettungswege für Knoten in den Treppenräumen ermittelt wurden, ist aus brandschutztechnischer Sicht widersprüchlich, da diese als sichere Bereiche gelten und für die übrigen Rettungswege das Ziel darstellen. Es führt jedoch nicht zu falschen Rettungswegnachweisen, da es auf die übrigen Knoten keinen Einfluss hat. Dennoch sollte dies in einem Praxiseinsatz ausgeschlossen werden, was aber mit Anwendung von Raumpolygonen in Treppenräumen problemlos machbar ist. [50]

Abschließend ist festzustellen, dass anhand der Rettungswegermittlung für das Beispielgebäude nachgewiesen wurde, dass der Softwareprototyp alle an ihn gestellten Anforderungen erfüllen konnte. Gleichzeitig wird damit auch die Anwendbarkeit der entwickelten Methodik zur BIM-basierten Rettungswegermittlung bestätigt. Schwierigkeiten und Fehler, die mit dem Beispielgebäudemodell nicht entdeckt wurden, werden im folgenden Abschnitt anhand von weiteren Minimalbeispielen erläutert.

### 6.5.4 Schwierigkeiten bei anderen Objekten

#### Aufteilung des Personenstroms

Anhand der Ausgabe für die Rettungswegermittlung des Beispielgebäudes wurde festgestellt, dass die Personenstromaufteilung auf die jeweiligen Notausgänge korrekt funktioniert. Mathematisch gesehen ist dies bei dem Softwareprototypen immer der Fall. Es kann jedoch die Situation auftreten, dass der Personenstrom aufgeteilt werden soll oder muss. Abbildung 6.25 zeigt die grafische Ausgabe für die Rettungswegermittlung eines einzelnen Raums. An den beiden Raumenden sind jeweils Notausgänge vorhanden, zu denen die Rettungswege auch korrekt hingeführt werden. Der am weitesten entfernte (relevante) Punkt von beiden Notausgängen befindet sich an einer Wandseite mittig, da noch weiter entfernte Punkte den jeweils anderen Notausgang als ersten Rettungsweg haben. Für diesen Raum ist eine Personenbelegung von 240 Personen festgelegt worden. Um den Personenstrom zügig abzuführen, wäre eine Aufteilung von 120 Personen je Notausgang günstig. Der Softwareprototyp teilt diesen Personenstrom jedoch nicht auf, sondern ordnet ihn einem der beiden Notausgänge zu (siehe Textausgabe in Anhang D.1). Ein eventueller Nachweis der Rettungswegbreiten könnte dann negativ ausfallen, wenn der Notausgang für den anfallenden Personenstrom zu schmal ist.

An dieser Stelle gibt es zwei denkbare Lösungsansätze. Die einfache Lösung wäre, den Personenstrom anhand der Anzahl der Ausgänge aus einem Raum aufzuteilen. Dadurch werden die vorhandenen Rettungswege gleichmäßig ausgelastet, sodass für den Nachweis der Rettungswegbreiten der minimale Personenstrom pro Rettungsweg angesetzt werden kann (was sich entsprechend günstig auf die Dimensionierung der Rettungswege auswirkt).

[50] Da die IfcRelSpaceBoundary-Zuordnung noch nicht zuverlässig funktioniert (siehe Abschnitt 6.1), wurde in dem Softwareprototypen auf eine Erstellung von Raumpolygonen in Treppenräumen vorerst verzichtet.

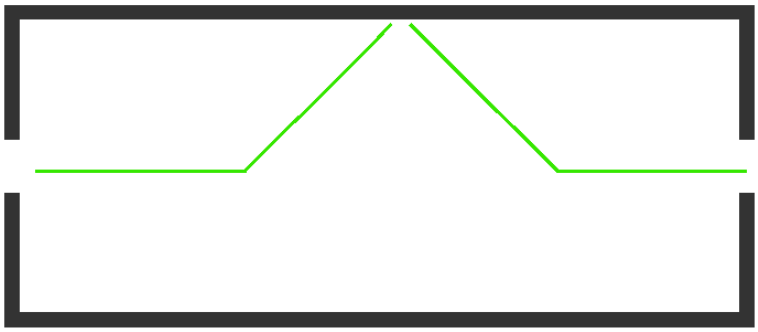


Abbildung 6.25: Grafische Ausgabe für die Rettungswegermittlung des Minimalbeispiels zur „Aufteilung des Personenstroms“

Eine zweite Lösungsmöglichkeit für diese Problematik wäre, die Personenzahl anteilig anhand der Anzahl von Rasterknoten aufzuteilen, die innerhalb eines Raums ihren ersten Rettungsweg zu den verschiedenen Notausgängen haben. In dem Beispiel aus Abbildung 6.25 sind von der Raummitte beide Notausgänge gleichermaßen weit entfernt, wodurch die hälftige Aufteilung problemlos anwendbar ist. Liegen die Notausgänge jedoch nicht wie in diesem Beispiel direkt an dem Raum, sondern weiter entfernt und mit unterschiedlichen Distanzen, kann der Fall eintreten, dass beispielsweise nur ein geringer Teil der Rasterknoten seinen ersten Rettungsweg zu einem bestimmten Notausgang führt. Würden die Personen in einem solchen Raum hälftig aufgeteilt werden, könnte es passieren, dass der Rettungswegnachweis für einen Teil nicht mehr erfüllt wird, da sie eigentlich dem anderen Notausgang zugeordnet wären.

Da mit dem zweiten Lösungsansatz die vorhandenen Rettungswegbreiten nicht optimal ausgenutzt werden und der Fall, dass der Rettungswegnachweis für die Rettungsweglängen nicht erfüllt wird, auch nicht eintreten muss, wäre ein iterativer Prozess die beste Lösung. Dabei werden die Personen zuerst nach Anzahl der vorhandenen Ausgänge aufgeteilt. Falls der Rettungsweglängennachweis nicht erfüllt werden sollte, werden entsprechend so viele Personen dem anderen Notausgang zugeordnet, bis dieser erfüllt ist. Eine Implementation der iterativen Lösung würde den Rahmen dieser Arbeit jedoch übersteigen, sodass der Lösungsansatz hier nur theoretisch erwähnt wird.

#### Fehlerhaft ermittelter zweiter Rettungsweg

Eine weitere Problemstellung, welche durch das Beispielgebäudemodell nicht aufgedeckt wird, ist das Finden eines „nicht-vorhandenen“ zweiten Rettungswegs. In dem Beispielgebäude führt pro Geschoss immer nur eine Tür zu jedem notwendigen Treppenraum. Somit kann der Notausgang immer dem notwendigen Treppenraum (Rettungsweg) gleichgesetzt werden. Es kann jedoch vorkommen, dass in einem Geschoss mehrere Türen zu einem notwendigen Treppenraum führen. In diesem Fall gibt es dann zwar zwei Notausgänge, aber zu demselben Rettungsweg.

Abbildung 6.26 zeigt die grafische Ausgabe für einen Grundriss, bei welchem die genannte räumliche Situation gegeben ist. Dieser Grundriss beinhaltet zwei Räume, die jeweils direkt an demselben notwendigen Treppenraum (Raum in der Mitte) angeschlossen sind. Dieser bildet den ersten Rettungsweg für die angrenzenden Räume, ein zweiter Rettungsweg existiert hier nicht. Damit die Türen als Notausgang erkannt werden, wurden diese in den IFC-Bauteileigenschaften als *fireExit* definiert.

Der erste Rettungsweg wird für jeden Raum bzw. für jede Stelle in dem Geschoss korrekt ermittelt. Die Textausgabe (Abbildung 6.27) zeigt jedoch, dass ein zweiter Rettungsweg ebenfalls ermittelt worden ist, wobei es sich in diesem Fall um den jeweils anderen Notausgang handelt. Dies stellt aus brandschutztechnischer Sicht einen schweren Fehler dar, weil hier ein zweiter Rettungsweg ermittelt worden ist, obwohl er nicht existiert. Geschuldet ist

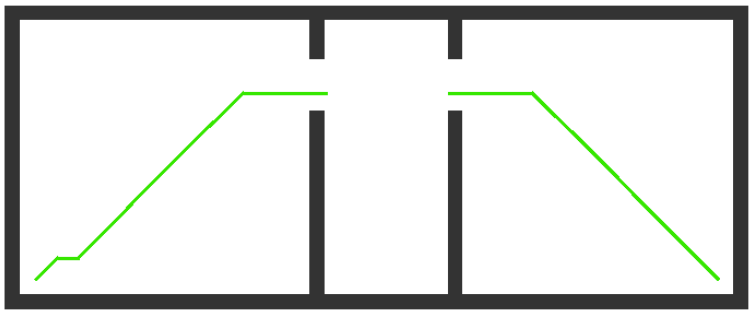


Abbildung 6.26: Grafische Ausgabe für die Rettungswegermittlung des Minimalbeispiels zum „fehlerhaft ermittelten zweiten Rettungsweg“

dieses Problem der „Übergangslösung“ mittels IfcPropertySet. In Abschnitt 6.1 ist bereits darauf hingewiesen worden, dass die Fluchtzieldefinition durch eine Erkennung mittels *IfcRelSpaceBoundary* die bessere Lösung wäre, da dieser Prozess automatisiert abliefe und die händische Definition der Notausgänge entfallen könnte. Da der Export in IfcRelSpaceBoundary-Objekte derzeit jedoch nicht zuverlässig funktioniert, konnte diese Lösung noch nicht in dem Softwareprototypen implementiert werden.

```
************************************************************
                    2. Obergeschoss
************************************************************

Rettungswege für jeden Raum:

R-001 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR2-2OG in maximal 28.03 m.
2. Rettungsweg: TR1-2OG in maximal 29.38 m.

R-002 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR2-2OG in maximal 15.08 m.
2. Rettungsweg: TR1-2OG in maximal 38.41 m.

____________________________________________________________

Längste Rettungswege in diesem Geschoss:

Von Knoten 626 (x: 1.409 y: 0.377) zu TR2-2OG.
in Raum: R-001 | Büro
Maximale Rettungsweglänge: 28.03 m.
2. Rettungsweg: TR1-2OG (Länge: 29.38 m.)


Summe des Personenstroms je Rettungsweg:

TR2-2OG: 3 Personen.
```

Abbildung 6.27: Textausgabe für die Rettungswegermittlung des Minimalbeispiels zum „fehlerhaft ermittelten zweiten Rettungsweg“ (Auszug)

Über Objekte der IfcRelSpaceBoundary-Klasse würden Beziehungen zwischen Räumen und angrenzenden Bauteilen hergestellt werden, sodass mehrere Türen einem Treppenraum eindeutig zugeordnet werden können. Bei der Rettungswegermittlung bleiben dann immer noch die Türen Ziele der Rettungswegführung, es können dann aber die zugeordneten Treppenräume abgefragt und damit eine mehrfache Rettungswegführung zu demselben

Treppenraum ausgeschlossen werden. In diesem Zusammenhang ist zusätzlich noch ein Algorithmus zu implementieren, welcher prüft, ob ein zweiter Rettungsweg durch den „Ziel-Treppenraum“ des ersten Rettungsweges hindurchführt. Auch diese Situation muss ausgeschlossen werden, um die Unabhängigkeit der Rettungswege zu gewährleisten.

# 7 Schlussbetrachtung

## 7.1 Zusammenfassung

Mit der vorliegenden Diplomarbeit wurde nachgewiesen, dass die Nutzung von BIM-Daten zur Rettungswegermittlung anhand bauordnungsrechtlicher Vorschriften grundsätzlich möglich ist und der entwickelte Softwareprototyp bei weiterem Ausbau auch praxisrelevant eingesetzt werden kann. Die aktuellen Möglichkeiten, die das IFC-Schema bietet, reichen jedoch für eine vollumfängliche Automatisierung dieses Prozesses noch nicht aus.

Für die prinzipielle Machbarkeitsstudie sind Informationen zu Wänden, Stützen, (Tür-)Öffnungen sowie Räumen relevant. Diese Elemente werden durch das IFC-Datenschema unterstützt, was eine objektorientierte Betrachtung ermöglicht. Die Filterung dieser Daten aus einer IFC-Datei gelingt problemlos, sodass die Bauwerksdaten vollständig ausgelesen und für die Verarbeitung durch den Softwareprototypen bereitgestellt werden können. Hierbei erweist sich das eingesetzte Java-Framework der Firma Apstex als geeignete Schnittstelle zwischen dem IFC-Datenformat und der programmierten Umsetzung in Java.

Durch die Verwendung einer Rasterunterteilung des Gebäudegrundrisses wird die Grundlage zur Erzeugung eines Graphen bereitet, anhand dessen die „kürzester-Pfad“-Berechnungen erfolgen. Mit der Methodik der Rasterung werden alle potentiellen Punkte in einem Grundriss erfasst, sodass auch bei komplexeren Bauteilkonstellationen immer der tatsächlich längste Weg ermittelt wird. Die Genauigkeit dieser Methode ist für einen praxistauglichen Einsatz jedoch noch zu optimieren, da durch das Raster für die Laufrichtung lediglich drei mögliche Winkel vorgegeben sind. Dies führt zu längeren Wegstrecken gegenüber den tatsächlichen Laufwegen. Zur Ermittlung der kürzesten Pfade hat sich der Dijkstra-Algorithmus bewährt, da mit diesem die optimale Lösung und damit die kürzesten Laufwege gefunden werden.

Die gefilterte geometrische Struktur des Gebäudes sowie die Ergebnisse der Rettungswegermittlung werden in einem separaten objektorientierten Datenmodell gespeichert. Dies ermöglicht eine fachspezifische Weiterverarbeitung dieser Daten, indem die aussagekräftigen Informationen aus diesem Modell geschossweise herausgefiltert und sowohl grafisch als auch in Textform ausgegeben werden. Mit der aktuellen Version des Softwareprototypen gelingt es, die Wegstrecken von jedem Rasterknoten zu allen Rettungswegen des jeweiligen Geschosses zu ermitteln und auf Basis ihrer Länge eine Unterscheidung zwischen erstem und zweitem Rettungsweg vorzunehmen. Zudem können die Personen der einzelnen Räume ihren jeweiligen ersten Rettungswegen hinzuaddiert werden, wodurch sich ein Gesamtpersonenstrom pro Rettungsweg angeben lässt.

Die Visualisierung zeigt den Verlauf der Rettungswege von den am weitesten entfernt gelegenen Punkten zu den jeweils nächstgelegenen Notausgängen. In der Textausgabe werden die entsprechenden Weglängen und Ziel-Notausgänge des ersten und zweiten Rettungswegs für jeden Raum sowie für die maximal entfernten Punkte angegeben. Des weiteren wird die Personendichte je Rettungsweg dargestellt, anhand derer ein Nachweis zur ausreichenden Rettungswegbreite vorgenommen werden kann. Ein automatischer Abgleich der vorhandenen Rettungswegbreiten erfolgt in der aktuellen Version des Softwareprototypen noch

nicht, da Regelungen bezüglich der Rettungswegbreiten nur in Sonderbauvorschriften getroffen werden, die im Rahmen dieser Arbeit nicht im erforderlichen Maße berücksichtigt werden konnten.

Die Unabhängigkeit der Rettungswege kann mit den vorliegenden IFC-Daten nicht abschließend nachgewiesen werden. Das IFC-Schema unterstützt keine direkte Anordnung von Nutzungseinheiten, sodass die Betrachtung einer unabhängigen Rettungswegführung je Nutzungseinheit nicht möglich ist. Darüber hinaus erweist sich die Angabe von Notausgängen zu notwendigen Treppenräumen mittels IfcPropertySet in der gegenwärtig implementierten Form als ungünstig. Bei zwei möglichen Zugängen zu demselben Treppenraum werden diese als unterschiedliche Rettungswege erkannt, wodurch sowohl der erste als auch der zweite Rettungsweg zum selben Treppenraum geführt werden. Dies widerspricht jedoch dem Unabhängigkeitsprinzip der Rettungswege, sodass der Rettungswegnachweis in diesem Falle fehlerhaft wäre.

## 7.2 Ausblick

Trotz der aktuell vorhandenen Probleme besteht das Potential, den Softwareprototypen weiterzuentwickeln und zu einem praxistauglichen Programm auszubauen. Zunächst muss dabei die Genauigkeit der Rastermethode verbessert werden. Dies ist mit dem Verfahren möglich, die Wegstrecke nicht mit statischen Winkeln von einem Knoten zu den unmittelbaren Nachbarknoten zu führen, sondern auf direktem Wege zu entfernt liegenden Knoten des Rasters.

Unbedingt notwendig ist es, die Unabhängigkeit der Rettungswege fehlerfrei nachzuweisen. Es muss ausgeschlossen werden, dass der erste Rettungsweg über eine benachbarte Nutzungseinheit geführt wird und dass beide Rettungswege zu verschiedenen Türen desselben Treppenraums führen. Letzteres wäre durch die Einbeziehung der IfcRelSpaceBoundary-Klasse möglich, welche Beziehungen zwischen Räumen und angrenzenden Bauteilen herstellt. Damit können auch mehrere Türen einem Treppenraum eindeutig zugeordnet werden, sodass bei der Rettungswegermittlung zu den Notausgangstüren eine Abfrage nach dem zugehörigen Treppenraum erfolgen kann. Dadurch lässt sich eine Führung des zweiten Rettungswegs zum selben Treppenraum des ersten Rettungswegs ausschließen. Eine derartige Erweiterung des Softwareprototypen kann aktuell jedoch noch nicht validiert werden, da der IFC-Export in die IfcRelSpaceBoundary-Klasse noch nicht zuverlässig funktioniert.

Um die Erkennung von unzulässigen Rettungswegführungen über benachbarte Nutzungseinheiten zu automatisieren, müsste das IFC-Schema die Festlegung von Nutzungseinheiten unterstützen. Möglich wäre dies über eine Klasse, in welcher mehrere IfcSpace-Instanzen zusammengefasst werden können, um eine Nutzungseinheit zu repräsentieren. Dementsprechend könnte für die Rasterknoten eine Zugehörigkeit zu einer Nutzungseinheit geprüft und eine unzulässige Rettungswegführung ausgeschlossen werden.

Neben der Verwendung in einem Programm zur Rettungswegermittlung anhand bauordnungsrechtlicher Vorschriften könnten die durch den Datenfilter erhältlichen Bauwerksinformationen auch für die Verarbeitung in einer Simulationssoftware genutzt werden. Damit ließen sich auch rechnerische Rettungswegnachweise einfacher durchführen, da das händische Erstellen eines eigenen Modells für diese Software entfallen würde. Sofern sich auf diese Art und Weise schnell gute Ergebnisse erzielen lassen, könnten rechnerische Verfahren in Zukunft das Brandschutzwesen stärker dominieren oder die Planung anhand bauordnungsrechtlicher Vorschriften sogar ablösen.

Unabhängig von der Entwicklung des Brandschutzes bleibt jedoch offen, ob die Integration der Brandschutzfachplanung in ein Bauwerksmodell der zukunftsorientierte Weg im Building Information Modeling ist. Als alternativer Ansatz dazu ist die Entwicklung sogenannter Multimodelle Gegenstand wissenschaftlicher Forschung. Hierbei arbeiten die jeweiligen Fachdisziplinen weiterhin mit ihren eigenen Fachmodellen, die Verknüpfung zwischen diesen erfolgt dann über Linkmodelle. Eine solche Verbindung zu einem Brandschutz- bzw. Rettungswege-Fachmodell wäre denkbar, da mit dem Datenschema des entwickelten Softwareprototypen bereits ein vom Bauwerksmodell unabhängiges Schema vorliegt. Da sich der Multimodellansatz aufgrund der Vielzahl von notwendigen Schnittstellen auch noch im Entwicklungsstadium befindet, ist ein bevorzugter Ansatz derzeit noch nicht absehbar.

Die vorliegende Arbeit hat in jedem Fall nachgewiesen, dass eine BIM-basierte Rettungswegermittlung mittels Graphentheorie ein zukunftsfähiges Konzept für die Brandschutzplanung ist und eine Weiterentwicklung dieses Ansatzes einen sinnvollen Beitrag zum digitalen Bauen leistet.

# Literatur


[1] AREO BLOG: *IFC4 - is it ready yet?* 9. Nov. 2016. URL: https://blog.areo.io/ifc4-is-it-ready-yet/ (besucht am 13. 03. 2019).

[2] BARTELME, N.: *Geoinformatik : Modelle, Strukturen, Funktionen.* 4., vollst. überarb. Aufl. Berlin ; , Heidelberg [u.a.] : Springer, 2005.

[3] BAUMINISTERKONFERENZ. 2019. URL: https://www.bauministerkonferenz.de/verzeichnis.aspx?id=762&o=7590762 (besucht am 26. 01. 2019).

[4] BAUMINISTERKONFERENZ: *Muster-Richtlinie über bauaufsichtliche Anforderungen an Schulen. Muster-Schulbau-Richtlinie.* MSchulbauR. Hrsg. von BAUAUFSICHT, F. 1. Apr. 2009.

[5] BAUMINISTERKONFERENZ: *Musterbauordnung.* MBO. Hrsg. von FACHKOMMISSION BAUAUFSICHT. Fassung November 2002; zuletzt geändert durch Beschluss der Bauministerkonferenz vom 13.05.2016. 21. Sep. 2012.

[6] BAUMINISTERKONFERENZ: *Musterverordnung über den Bau und Betrieb von Versammlungsstätten. Muster-Versammlungsstättenverordnung.* MVStättVO. Hrsg. von FACHKOMMISSION BAUFAUFSICHT. 1. Juli 2014.

[7] BAUNETZ MEDIA GMBH: *BIM. Welche Arten von BIM-Daten gibt es?* 2019. URL: https://www.baunetzwissen.de/bim/fachwissen/grundlagen/welche-arten-von-bim-daten-gibt-es-5262646 (besucht am 30. 01. 2019).

[8] BAUNETZ MEDIA GMBH: *IFC der offene Standard für BIM-Modelle.* 2019. URL: https://www.baunetzwissen.de/bim/fachwissen/standardisierung/ifc-der-offene-standard-fuer-bim-modelle-5288161 (besucht am 03. 02. 2019).

[9] BUILDINGSMART: *About.* 2019. URL: https://www.buildingsmart.org/about/ (besucht am 03. 02. 2019).

[10] BUNDESANSTALT FÜR ARBEITSSCHUTZ UND ARBEITSMEDIZIN: *Technische Regeln für Arbeitsstätten.* Fluchtwege und Notausgänge, Flucht- und Rettungsplan. 2017.

[11] CONCEPTURE GRUPPE BRANDSCHUTZ: *Fluchtwege und Rettungswege.* Hrsg. von HUCK, A. 28. Aug. 2017. URL: https://concepture.de/brandschutz/rettungsweg/ (besucht am 27. 01. 2019).

[12] CORMEN, T. H.: *Algorithmen - eine Einführung.* 4., durchges. u. korr. Aufl. München : Oldenbourg, 2013.

[13] DATACUBIST OY: *Space Boundary. Background.* 2019. URL: http://datacubist.com/support/addon-spaceboundary.html (besucht am 26. 03. 2019).

[14] DIESTEL, R.: *Graphentheorie.* 3., neu bearb. und erw. Aufl. Berlin ; , Heidelberg [u.a.] : Springer, 2006.

[15] HIERHOLZER, C.: „Ueber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren“. ger. *Mathematische Annalen* 6 (1873), S. 30–32.

[16] Holzem, D.: *Das Multimodell unter Ausnutzung des Variationsprinzips am Beispiel der Brandsimulation mit dem Fire Dynamics Simulator*. Diplomarbeit. Technische Universität Dresden, 2018.

[17] Hosser, D.: *Leitfaden Ingenieurmethoden des Brandschutzes*. Techn. Ber. Vereinigung zur Förderung des Deutschen Brandschutzes e.V. (vfdb), 1. Mai 2009, S. 386.

[18] IAI International Council Limited: *IFC2x Edition 3 Technical Corrigendum 1*. 2007. URL: http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm (besucht am 17. 03. 2019).

[19] Ingenieure für Brandschutz GmbH, hhpberlin: *Personenstromanalyse bei Sonderbauten mit rechnergestützten Methoden*. Hrsg. von Tilly, R. 29. Jan. 2019.

[20] Jäger, A. P. G.: *Evakuierungssimulation im Rahmen von Sicherheitskonzepten*. von der Konzeption bis zur Realisierung an Beispielen. Beitrag. BFT Cognos, 24. Nov. 2011.

[21] Linxweiler, M. E. J.: *Berechnung von Evakuierungszeiten bei Sonderbauten mit dem Programm buildingExodus*. 2004. URL: https://bau-ings.de/exodus/kapitel3.htm (besucht am 29. 01. 2019).

[22] N+P Informationssysteme GmbH: *Der richtige Einstieg in BIM*. 8. März 2017. URL: https://blog.nupis.de/einstieg-open-bim-closed-bim/ (besucht am 11. 03. 2019).

[23] N+P Informationssysteme GmbH: *Vergleich von open BIM, closed BIM, little BIM, big BIM und connected BIM*. 8. Feb. 2018. URL: https://blog.nupis.de/vergleich-open-closed-little-big-bim-connected-bim/ (besucht am 11. 03. 2019).

[24] Plandata BIM Solutions: *BIM Level. Entwicklungsstufen der BIM Methode*. 2019. URL: https://www.bimpedia.eu/node/1003 (besucht am 08. 03. 2019).

[25] Probst, M.: *Was ist openBIM?* 2018. URL: https://bimconnect.org/wiki/was-ist-openbim/ (besucht am 31. 01. 2019).

[26] Rovas, G. N. L. G. I. G. D.: „Automatic generation of second-level space boundary topology from IFC geometry inputs“. *Automation in Construction* (19. Jan. 2017).

[27] Schapke, R. J. S. S.-E.: *Informationssysteme im Bauwesen. 1. Modelle, Methoden und Prozesse*. Berlin ; , Heidelberg : Springer Vieweg, 2014.

[28] Scherer, R. J.: *WP4-70: Modellbasiertes Arbeiten - BIM. Vorlesung 1: Einführung*. 8. Nov. 2013.

[29] Schwartz, A.: „Einführung in die Graphentheorie“. 16. Aug. 2013.

[30] Torres, Y.: *Alliance polynomial and hyperbolicity in regular graphs*. Requirement for the degree of Ph. Doctor in Mathematical Engineering. University Carlos III de Madrid, 1. Juli 2014.

[31] Turau, V.: *Algorithmische Graphentheorie*. 3., überarb. Aufl. München : Oldenbourg, 2009.

[32] Upmeier, B. S. L. B. H.: *Königsberger Brückenproblem. Ausarbeitung des Seminarvortrags vom 21.10.2009*. Hrsg. von Schwarz, B. 21. Okt. 2009.

[33] *Verwaltungsvorschrift des Sächsischen Staatsministeriums des Innern zur Sächsischen Bauordnung*. 18. März 2005.

# Anhang

# A Darstellung von IFC-Klassen in EXPRESS-Notation

## A.1 IfcWall

Quellcode A.1: Spezifikation der Klasse IfcWall in EXPRESS-Notation [18]

```
ENTITY IfcWall
  SUPERTYPE OF  (IfcWallStandardCase)
  SUBTYPE OF (  IfcBuildingElement);
  WHERE
    WR1  :  SIZEOF (QUERY(temp <* SELF\IfcObjectDefinition.HasAssociations
        | 'IFCPRODUCTEXTENSION.IFCRELASSOCIATESMATERIAL' IN TYPEOF(temp) ))
        <= 1;
END_ENTITY;
```

Quellcode A.2: Darstellung der Klasse IfcWall mit allen Vererbungen in EXPRESS-Notation [18]

```
ENTITY IfcWall;
ENTITY IfcRoot;
  GlobalId      :   IfcGloballyUniqueId;
  OwnerHistory  :   IfcOwnerHistory;
  Name          :   OPTIONAL IfcLabel;
  Description   :   OPTIONAL IfcText;
ENTITY IfcObjectDefinition;
INVERSE
  HasAssignments  :  SET OF IfcRelAssigns FOR RelatedObjects;
  IsDecomposedBy  :  SET OF IfcRelDecomposes FOR RelatingObject;
  Decomposes    :   SET [0:1] OF IfcRelDecomposes FOR RelatedObjects;
  HasAssociations :  SET OF IfcRelAssociates FOR RelatedObjects;
ENTITY IfcObject;
  ObjectType    :   OPTIONAL IfcLabel;
INVERSE
  IsDefinedBy   :   SET OF IfcRelDefines FOR RelatedObjects;
ENTITY IfcProduct;
  ObjectPlacement :  OPTIONAL IfcObjectPlacement;
  Representation  :  OPTIONAL IfcProductRepresentation;
INVERSE
  ReferencedBy  :   SET OF IfcRelAssignsToProduct FOR RelatingProduct;
ENTITY IfcElement;
  Tag           :   OPTIONAL IfcIdentifier;
INVERSE
  HasStructuralMember  :  SET OF IfcRelConnectsStructuralElement FOR
      RelatingElement;
  FillsVoids      :   SET [0:1] OF IfcRelFillsElement FOR
      RelatedBuildingElement;
  ConnectedTo     :   SET OF IfcRelConnectsElements FOR RelatingElement;
  HasCoverings    :   SET OF IfcRelCoversBldgElements FOR
      RelatingBuildingElement;
  HasProjections    :   SET OF IfcRelProjectsElement FOR RelatingElement;
```

```
  ReferencedInStructures  :  SET OF IfcRelReferencedInSpatialStructure FOR
      RelatedElements;
  HasPorts        :   SET OF IfcRelConnectsPortToElement FOR RelatedElement;
  HasOpenings     :   SET OF IfcRelVoidsElement FOR RelatingBuildingElement;
  IsConnectionRealization :  SET OF IfcRelConnectsWithRealizingElements FOR
      RealizingElements;
  ProvidesBoundaries   :  SET OF IfcRelSpaceBoundary FOR
      RelatedBuildingElement;
  ConnectedFrom    :  SET OF IfcRelConnectsElements FOR RelatedElement;
  ContainedInStructure  :  SET [0:1] OF IfcRelContainedInSpatialStructure
      FOR RelatedElements;
ENTITY IfcBuildingElement;
ENTITY IfcWall;
END_ENTITY;
```

## A.2 IfcElement

Quellcode A.3: Spezifikation der Klasse IfcElement in EXPRESS-Notation [18]

```
ENTITY IfcElement
  ABSTRACT SUPERTYPE OF  (ONEOF(IfcBuildingElement,
     IfcFurnishingElement, IfcElectricalElement,
     IfcDistributionElement, IfcTransportElement,
     IfcEquipmentElement, IfcFeatureElement, IfcElementAssembly,
     IfcVirtualElement))
  SUBTYPE OF (  IfcProduct);
    Tag  :  OPTIONAL IfcIdentifier;
  INVERSE
    HasStructuralMember  :  SET OF IfcRelConnectsStructuralElement
       FOR RelatingElement;
    FillsVoids  :  SET [0:1] OF IfcRelFillsElement FOR
       RelatedBuildingElement;
    ConnectedTo  :  SET OF IfcRelConnectsElements FOR
       RelatingElement;
    HasCoverings  :  SET OF IfcRelCoversBldgElements FOR
       RelatingBuildingElement;
    HasProjections  :  SET OF IfcRelProjectsElement FOR
       RelatingElement;
    ReferencedInStructures  :  SET OF
       IfcRelReferencedInSpatialStructure FOR RelatedElements;
    HasPorts  :  SET OF IfcRelConnectsPortToElement FOR
       RelatedElement;
    HasOpenings  :  SET OF IfcRelVoidsElement FOR
       RelatingBuildingElement;
    IsConnectionRealization  :  SET OF
       IfcRelConnectsWithRealizingElements FOR RealizingElements;
    ProvidesBoundaries  :  SET OF IfcRelSpaceBoundary FOR
       RelatedBuildingElement;
    ConnectedFrom  :  SET OF IfcRelConnectsElements FOR
       RelatedElement;
    ContainedInStructure  :  SET [0:1] OF
       IfcRelContainedInSpatialStructure FOR RelatedElements;
END_ENTITY;
```

Quellcode A.4: Darstellung der Klasse IfcElement mit allen Vererbungen in EXPRESS-Notation [18]

```
ENTITY IfcElement;
  ENTITY IfcRoot;
    GlobalId  :  IfcGloballyUniqueId;
    OwnerHistory  :  IfcOwnerHistory;
    Name  :  OPTIONAL IfcLabel;
    Description  :  OPTIONAL IfcText;
  ENTITY IfcObjectDefinition;
  INVERSE
    HasAssignments  :  SET OF IfcRelAssigns FOR RelatedObjects;
    IsDecomposedBy  :  SET OF IfcRelDecomposes FOR RelatingObject;
    Decomposes  :  SET [0:1] OF IfcRelDecomposes FOR RelatedObjects;
    HasAssociations  :  SET OF IfcRelAssociates FOR RelatedObjects;
  ENTITY IfcObject;
    ObjectType  :  OPTIONAL IfcLabel;
  INVERSE
    IsDefinedBy  :  SET OF IfcRelDefines FOR RelatedObjects;
  ENTITY IfcProduct;
    ObjectPlacement  :  OPTIONAL IfcObjectPlacement;
    Representation  :  OPTIONAL IfcProductRepresentation;
  INVERSE
    ReferencedBy  :  SET OF IfcRelAssignsToProduct FOR
       RelatingProduct;
  ENTITY IfcElement;
    Tag  :  OPTIONAL IfcIdentifier;
  INVERSE
    HasStructuralMember  :  SET OF IfcRelConnectsStructuralElement
       FOR RelatingElement;
    FillsVoids  :  SET [0:1] OF IfcRelFillsElement FOR
       RelatedBuildingElement;
    ConnectedTo  :  SET OF IfcRelConnectsElements FOR
       RelatingElement;
    HasCoverings  :  SET OF IfcRelCoversBldgElements FOR
       RelatingBuildingElement;
    HasProjections  :  SET OF IfcRelProjectsElement FOR
       RelatingElement;
    ReferencedInStructures  :  SET OF
       IfcRelReferencedInSpatialStructure FOR RelatedElements;
    HasPorts  :  SET OF IfcRelConnectsPortToElement FOR
       RelatedElement;
    HasOpenings  :  SET OF IfcRelVoidsElement FOR
       RelatingBuildingElement;
    IsConnectionRealization  :  SET OF
       IfcRelConnectsWithRealizingElements FOR RealizingElements;
    ProvidesBoundaries  :  SET OF IfcRelSpaceBoundary FOR
       RelatedBuildingElement;
    ConnectedFrom  :  SET OF IfcRelConnectsElements FOR
       RelatedElement;
    ContainedInStructure  :  SET [0:1] OF
       IfcRelContainedInSpatialStructure FOR RelatedElements;
END_ENTITY;
```

## A.3 IfcProduct

Quellcode A.5: Spezifikation der Klasse IfcProduct in EXPRESS-Notation [18]

```
ENTITY IfcProduct
  ABSTRACT SUPERTYPE OF  (IfcProxy)
  SUBTYPE OF (  IfcObject);
    ObjectPlacement  :  OPTIONAL IfcObjectPlacement;
    Representation  :  OPTIONAL IfcProductRepresentation;
  INVERSE
    ReferencedBy  :  SET OF IfcRelAssignsToProduct FOR
       RelatingProduct;
  WHERE
    WR1  :   (EXISTS(Representation) AND EXISTS(ObjectPlacement))
       OR (EXISTS(Representation) AND
       (NOT('IFCREPRESENTATIONRESOURCE.IFCPRODUCTDEFINITIONSHAPE'
       IN TYPEOF(Representation)))) OR
       (NOT(EXISTS(Representation))) ;
END_ENTITY;
```

Quellcode A.6:
Darstellung der Klasse IfcProduct mit allen Vererbungen in EXPRESS-Notation [18]

```
ENTITY IfcProduct;
  ENTITY IfcRoot;
    GlobalId  :  IfcGloballyUniqueId;
    OwnerHistory  :  IfcOwnerHistory;
    Name  :  OPTIONAL IfcLabel;
    Description  :  OPTIONAL IfcText;
  ENTITY IfcObjectDefinition;
  INVERSE
    HasAssignments  :  SET OF IfcRelAssigns FOR RelatedObjects;
    IsDecomposedBy  :  SET OF IfcRelDecomposes FOR RelatingObject;
    Decomposes  :  SET [0:1] OF IfcRelDecomposes FOR RelatedObjects;
    HasAssociations  :  SET OF IfcRelAssociates FOR RelatedObjects;
  ENTITY IfcObject;
    ObjectType  :  OPTIONAL IfcLabel;
  INVERSE
    IsDefinedBy  :  SET OF IfcRelDefines FOR RelatedObjects;
  ENTITY IfcProduct;
    ObjectPlacement  :  OPTIONAL IfcObjectPlacement;
    Representation  :  OPTIONAL IfcProductRepresentation;
  INVERSE
  ReferencedBy  :  SET OF IfcRelAssignsToProduct FOR
     RelatingProduct;
END_ENTITY;
```

## A.4 IfcPropertySet

Quellcode A.7: Spezifikation der Klasse IfcPropertySet in EXPRESS-Notation [18]

```
ENTITY IfcPropertySet
  SUBTYPE OF (  IfcPropertySetDefinition);
    HasProperties  :  SET [1:?] OF IfcProperty;
  WHERE
    WR31  :  EXISTS(SELF\IfcRoot.Name);
    WR32  :  IfcUniquePropertyName(HasProperties);
END_ENTITY;
```

Quellcode A.8: Darstellung der Klasse IfcPropertySet mit allen Vererbungen in EXPRESS-Notation [18]

```
ENTITY IfcPropertySet;
  ENTITY IfcRoot;
    GlobalId  :  IfcGloballyUniqueId;
    OwnerHistory  :  IfcOwnerHistory;
    Name  :  OPTIONAL IfcLabel;
    Description  :  OPTIONAL IfcText;
  ENTITY IfcPropertyDefinition;
  INVERSE
    HasAssociations  :  SET OF IfcRelAssociates FOR RelatedObjects;
  ENTITY IfcPropertySetDefinition;
  INVERSE
    PropertyDefinitionOf  :  SET [0:1] OF IfcRelDefinesByProperties
       FOR RelatingPropertyDefinition;
    DefinesType  :  SET [0:1] OF IfcTypeObject FOR HasPropertySets;
  ENTITY IfcPropertySet;
    HasProperties  :  SET [1:?] OF IfcProperty;
END_ENTITY;
```

## A.5 IfcLocalPlacement

Quellcode A.9: Spezifikation der Klasse IfcLocalPlacement in EXPRESS-Notation [18]

```
ENTITY IfcLocalPlacement
  SUBTYPE OF (IfcObjectPlacement);
    PlacementRelTo  :  OPTIONAL IfcObjectPlacement;
    RelativePlacement  :  IfcAxis2Placement;
  WHERE
    WR21  : IfcCorrectLocalPlacement(RelativePlacement,
       PlacementRelTo);
END_ENTITY;
```

Quellcode A.10: Darstellung der Klasse IfcLocalPlacement mit allen Vererbungen in EXPRESS-Notation [18]

```
ENTITY IfcLocalPlacement;
  ENTITY IfcObjectPlacement;
  INVERSE
    PlacesObject  :  SET [1:1] OF IfcProduct FOR ObjectPlacement;
    ReferencedByPlacements  :  SET OF IfcLocalPlacement FOR
       PlacementRelTo;
  ENTITY IfcLocalPlacement;
    PlacementRelTo  :  OPTIONAL IfcObjectPlacement;
    RelativePlacement  :  IfcAxis2Placement;
END_ENTITY;
```

## A.6 IfcProductDefinitionShape

Quellcode A.11: Spezifikation der Klasse IfcProductDefinitionShape in EXPRESS-Notation [18]

```
ENTITY IfcProductDefinitionShape
  SUBTYPE OF (  IfcProductRepresentation);
  INVERSE
    ShapeOfProduct  :  SET [1:1] OF IfcProduct FOR Representation;
    HasShapeAspects  :  SET OF IfcShapeAspect FOR
       PartOfProductDefinitionShape;
  WHERE
    WR11  :  SIZEOF(QUERY(temp <* Representations |
       (NOT('IFCREPRESENTATIONRESOURCE.IFCSHAPEMODEL' IN
       TYPEOF(temp))) )) = 0;
END_ENTITY;
```

Quellcode A.12: Darstellung der Klasse IfcProductDefinitionShape mit allen Vererbungen in EXPRESS-Notation [18]

```
ENTITY IfcProductDefinitionShape
  SUBTYPE OF (  IfcProductRepresentation);
  INVERSE
    ShapeOfProduct  : SET [1:1] OF IfcProduct FOR Representation;
    HasShapeAspects : SET OF IfcShapeAspect FOR
       PartOfProductDefinitionShape;
  WHERE
    WR11  :  SIZEOF(QUERY(temp <* Representations |
       (NOT('IFCREPRESENTATIONRESOURCE.IFCSHAPEMODEL' IN
       TYPEOF(temp))) )) = 0;
  END_ENTITY;
```

## A.7 IfcRoot

Quellcode A.13: Spezifikation der Klasse IfcRoot in EXPRESS-Notation [18]

```
ENTITY IfcRoot
  ABSTRACT SUPERTYPE OF  (ONEOF(IfcPropertyDefinition,
     IfcRelationship, IfcObjectDefinition));
    GlobalId  :  IfcGloballyUniqueId;
    OwnerHistory :  IfcOwnerHistory;
    Name  :  OPTIONAL IfcLabel;
    Description :  OPTIONAL IfcText;
  UNIQUE
    UR1  :  GlobalId;
END_ENTITY;
```

Quellcode A.14: Darstellung der Klasse IfcRoot mit allen Vererbungen in EXPRESS-Notation [18]

```
ENTITY IfcRoot;
  ENTITY IfcRoot;
    GlobalId  :  IfcGloballyUniqueId;
    OwnerHistory :  IfcOwnerHistory;
    Name  :  OPTIONAL IfcLabel;
    Description :  OPTIONAL IfcText;
END_ENTITY;
```

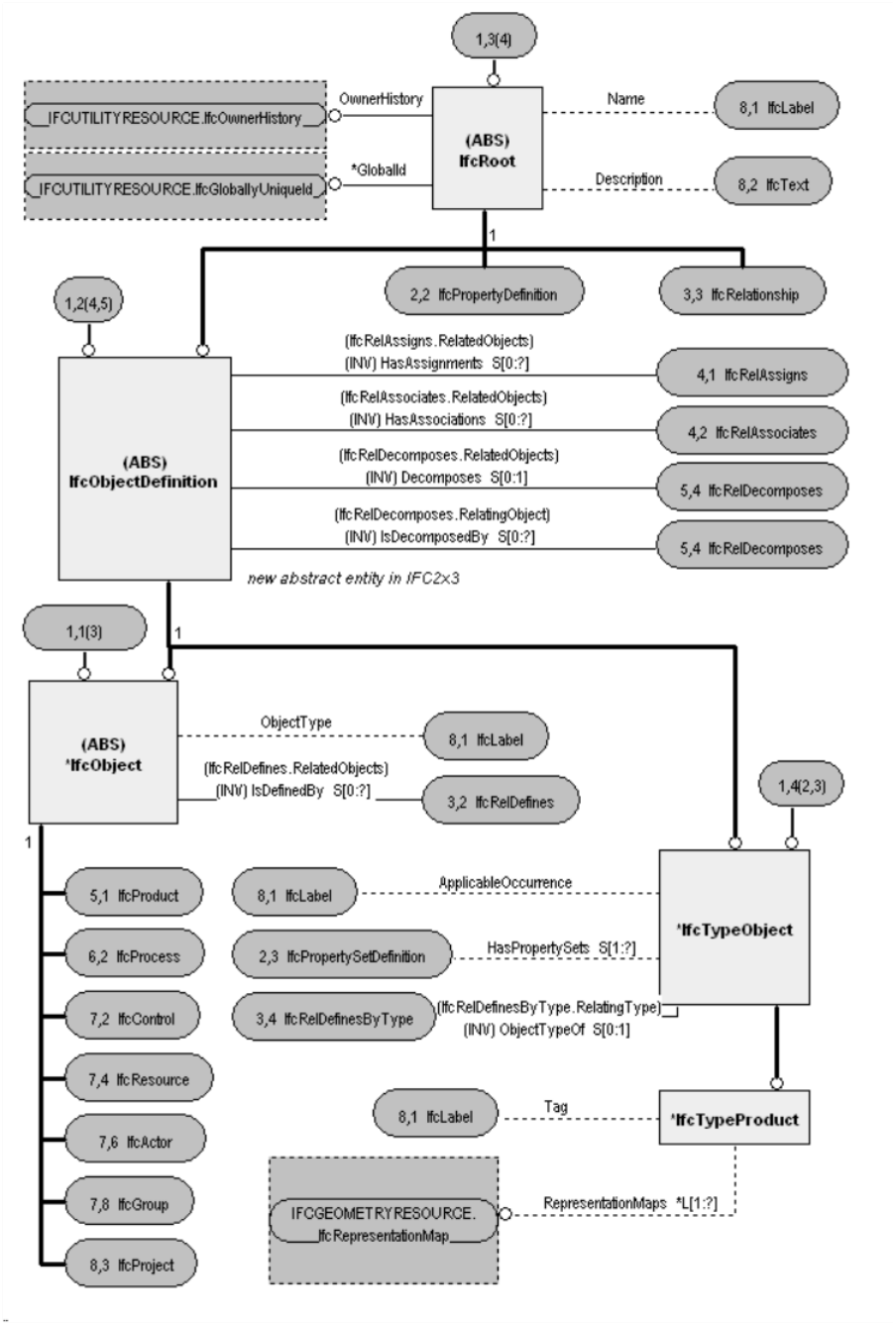



Abbildung A.1: Die Klasse IfcRoot mit ihrer Vererbungshierarchie in der graphischen EXPRESS-G Darstellung

# B UML-Diagramme von Klassen des Softwareprototypen

## B.1 PlanElement, Floor und LineSegement

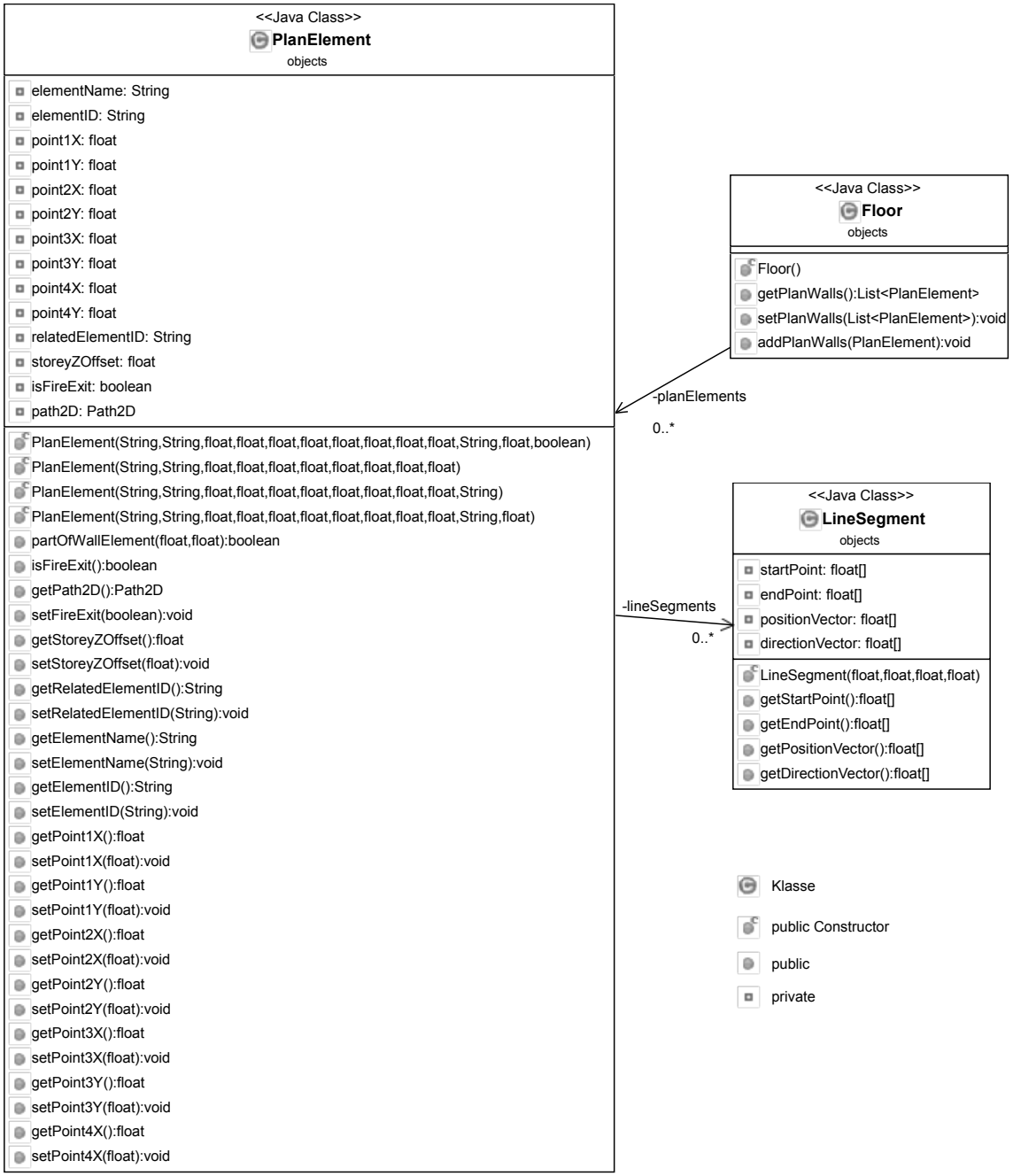



Abbildung B.1: UML-Klassendiagramm zu den Klassen Floor, PlanElement und LineSegment (abweichende Notation durch Export mit Eclipse-AddOn „ObjectAid“)

## B.2 SingleRoom, Point und Rooms

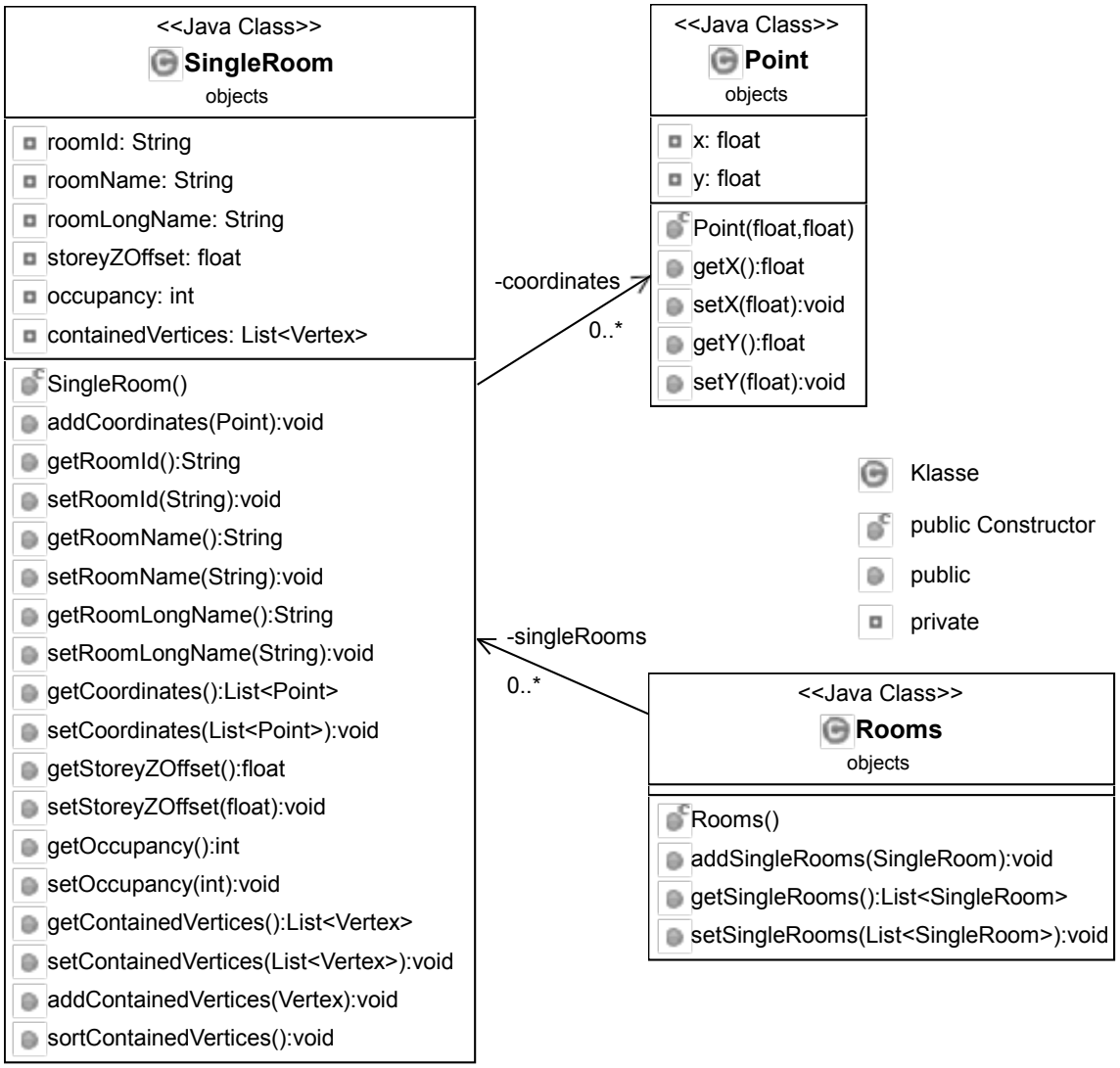



Abbildung B.2: UML-Klassendiagramm zu den Klassen SingleRoom, Point und Rooms (abweichende Notation durch Export mit Eclipse-AddOn „ObjectAid“)

## B.3 Building mit Floor-Array und Rooms-Array

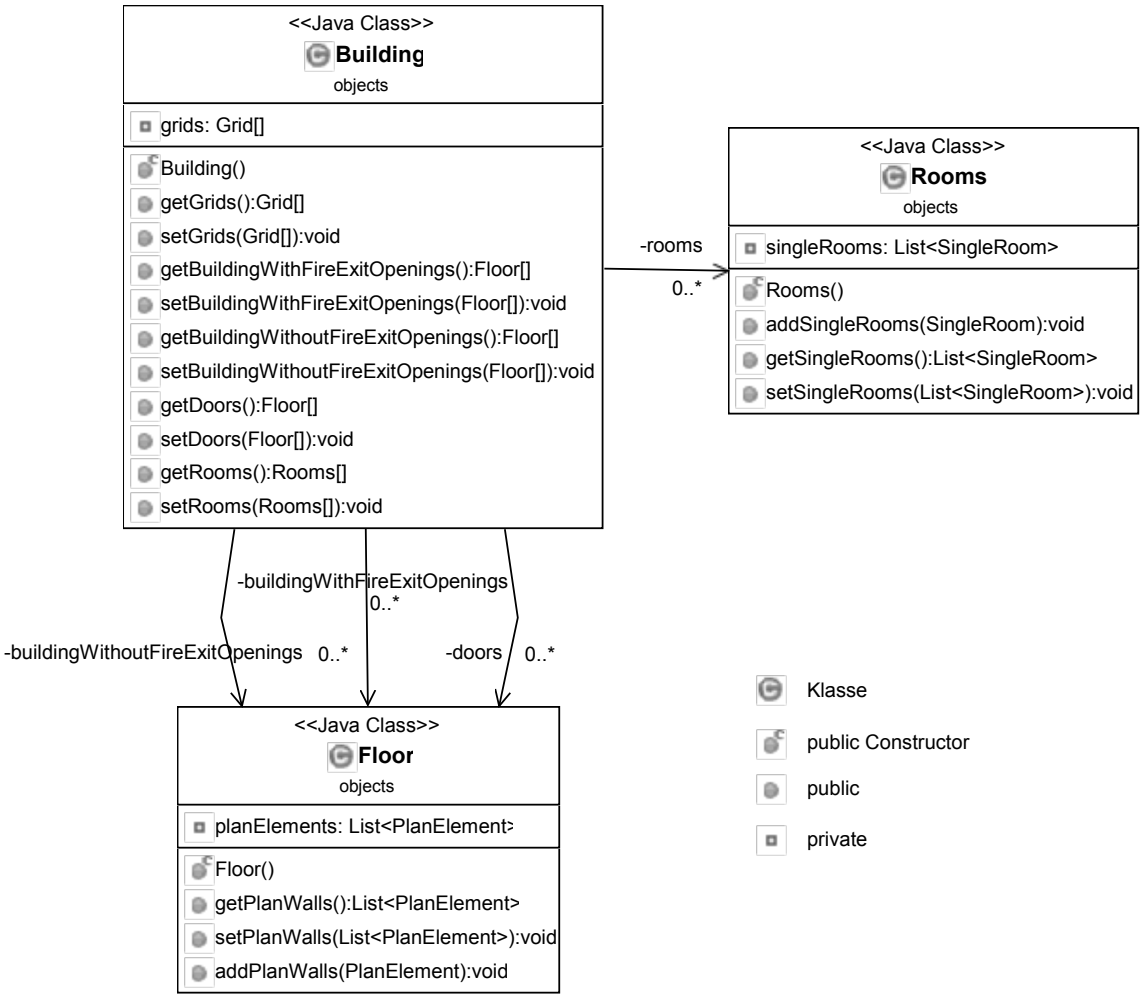



Abbildung B.3: UML-Klassendiagramm zu den Klassen Building, Floor und Rooms (abweichende Notation durch Export mit Eclipse-AddOn „ObjectAid“)

## B.4 Vertex, EscapeRoute, Edge, Grid

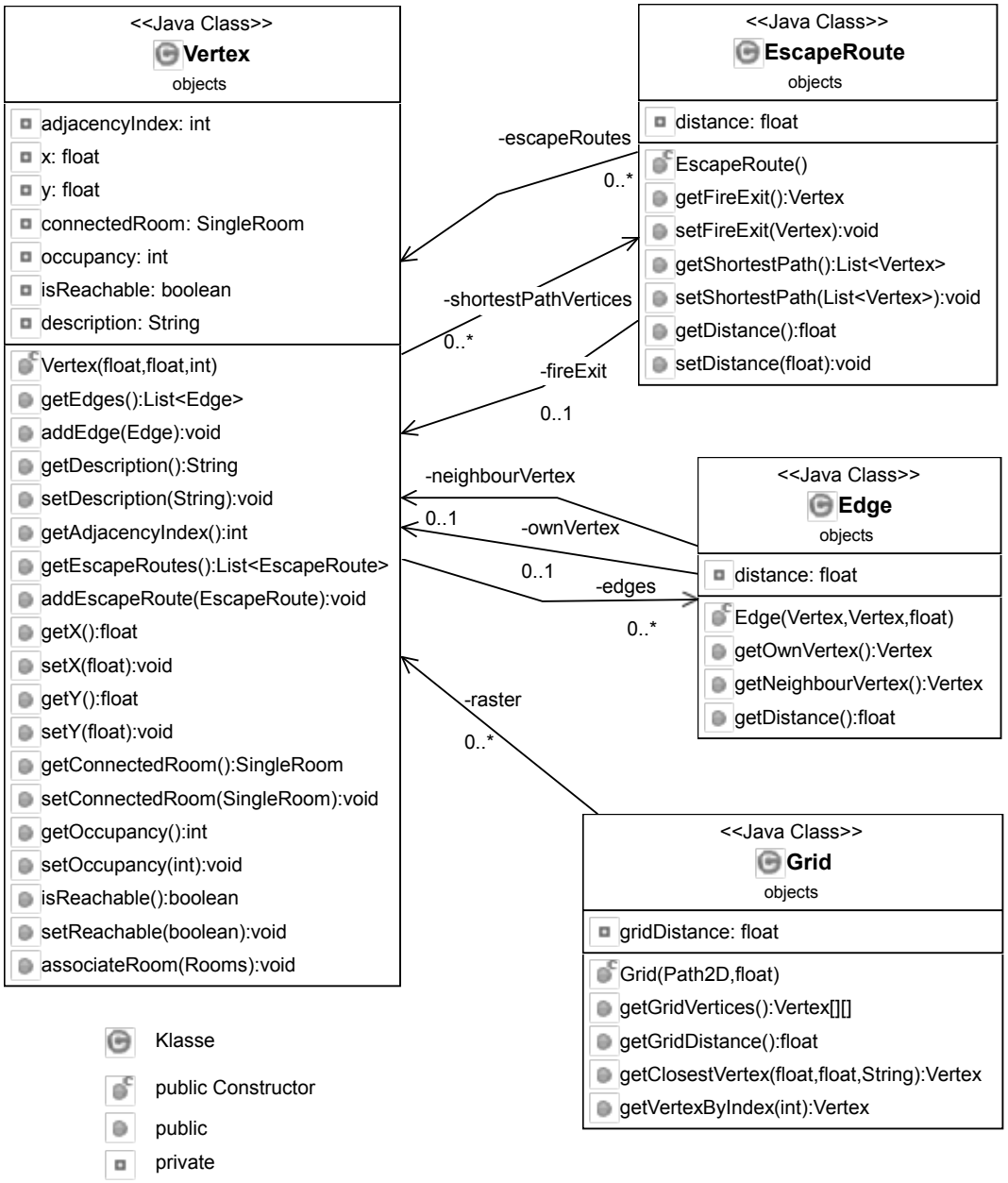



Abbildung B.4: UML-Klassendiagramm zu den Klassen Vertex, EscapeRoute, Edge und Grid (abweichende Notation durch Export mit Eclipse-AddOn „ObjectAid“)

# C Rettungswegermittlung anhand des Beispielgebäudemodells

## C.1 Grundrisse aller Geschosse und Rauminformationen

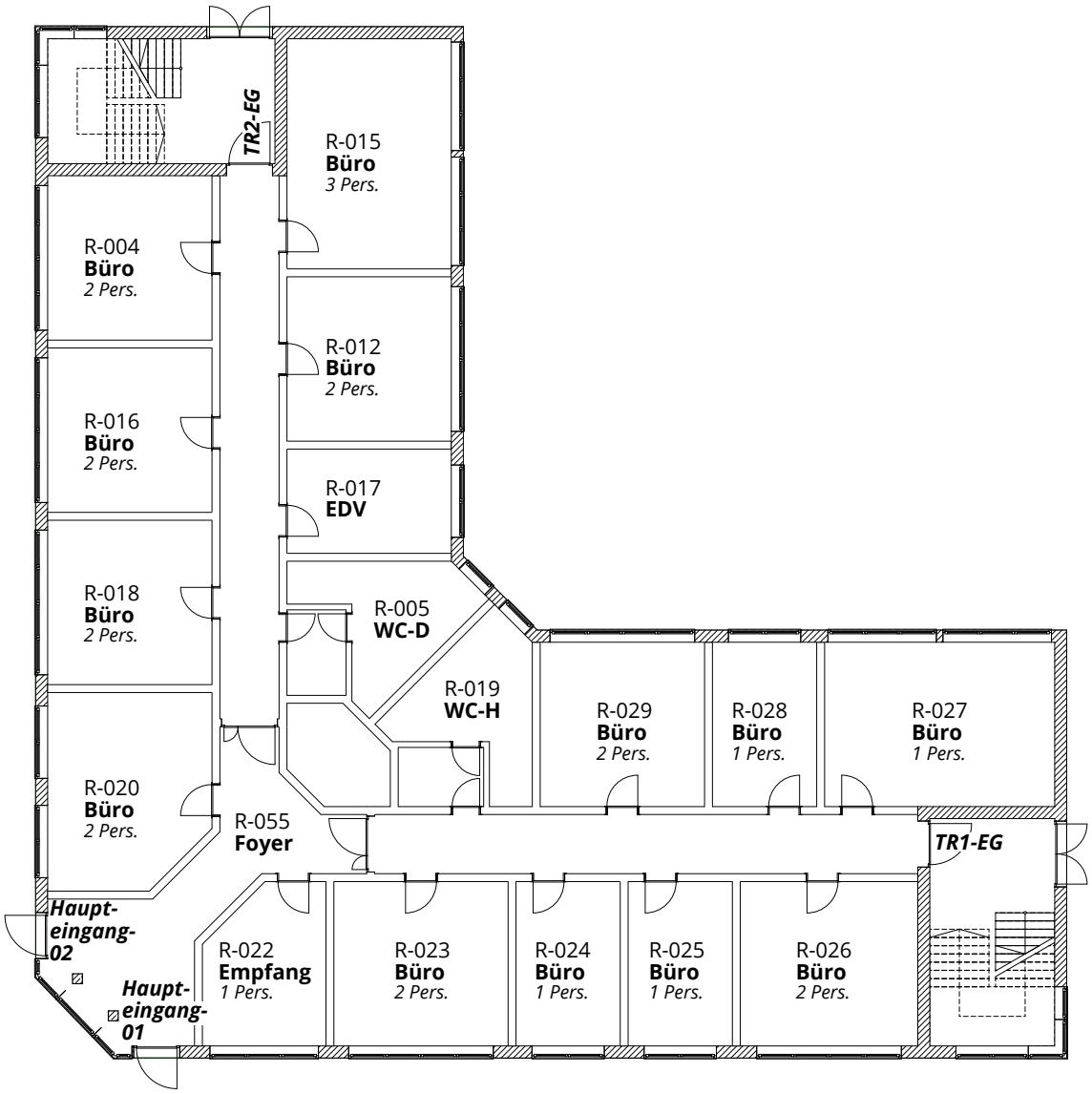



Abbildung C.1: Erdgeschoss des Beispielgebäudes im Grundriss

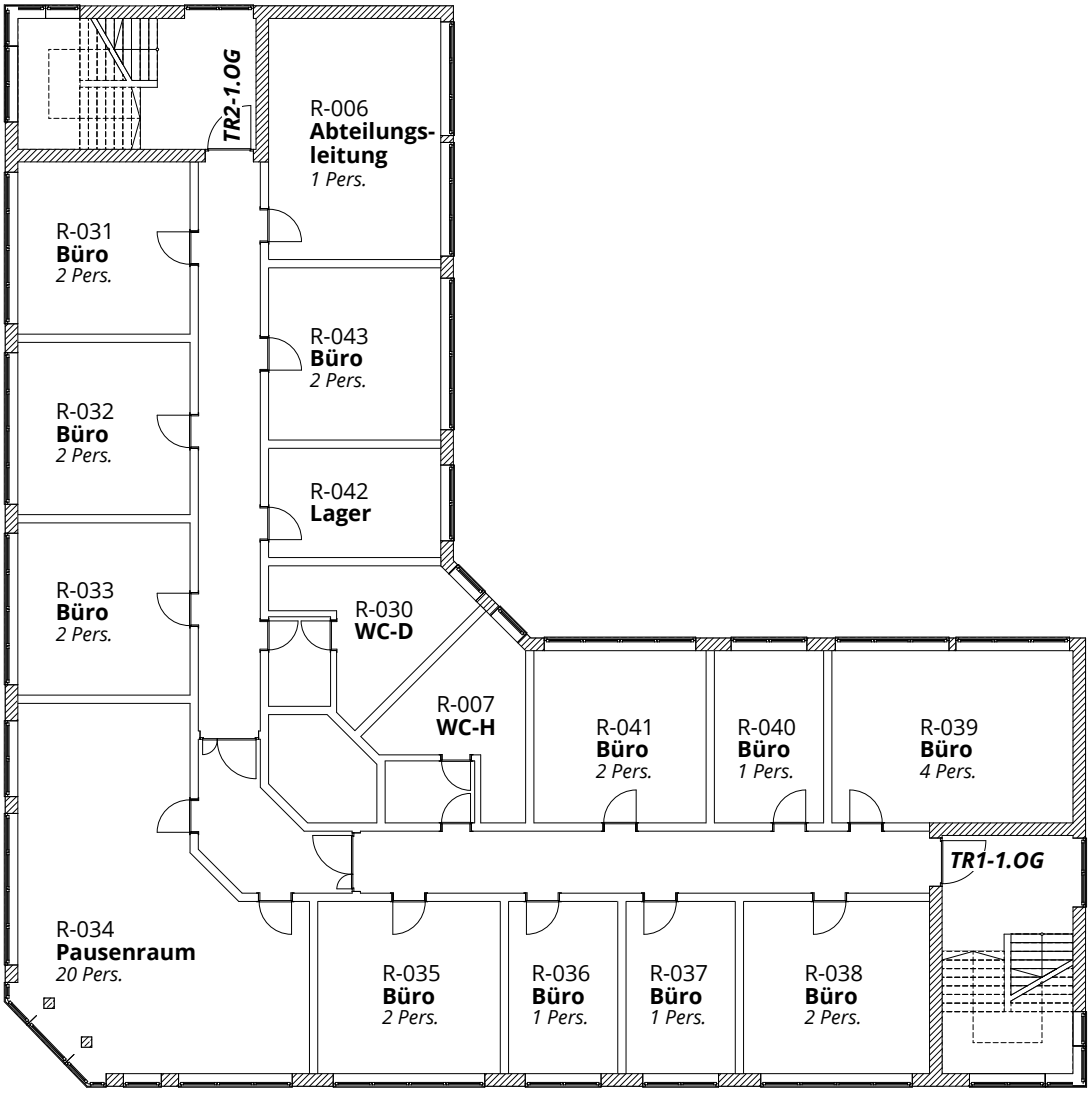



Abbildung C.2: 1. Obergeschoss des Beispielgebäudes im Grundriss

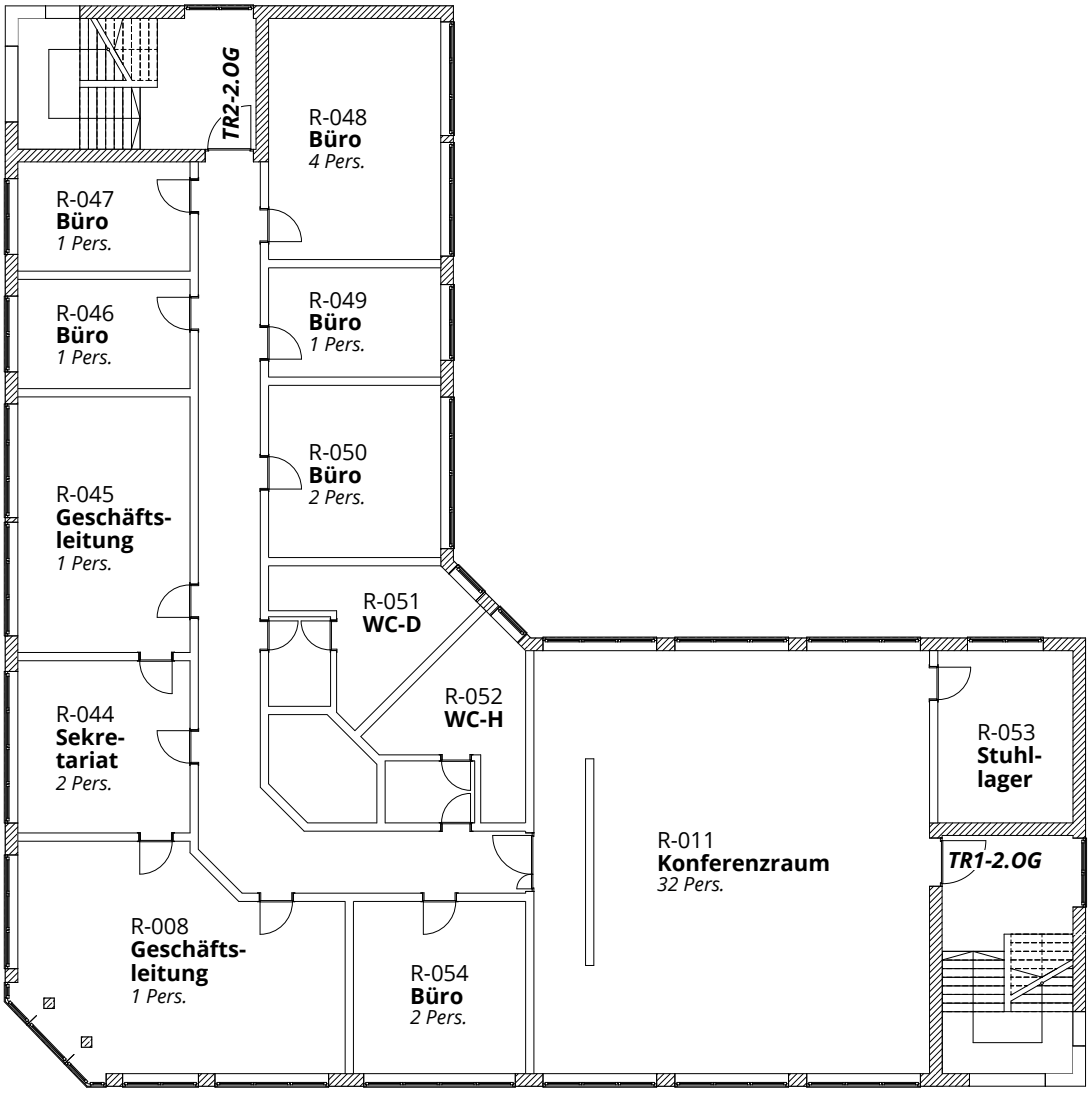



Abbildung C.3: 2. Obergeschoss des Beispielgebäudes im Grundriss

## C.2 Textausgabe für das Beispielgebäude

### C.2.1 Rettungswegermittlung bei 0,50 m Knotenabstand

```
************************************************************
                    Erdgeschoss
************************************************************

Rettungswege für jeden Raum:

R-015 | Büro
Belegung: 3 Personen.
1. Rettungsweg: TR2-EG in maximal 10.49 m.
2. Rettungsweg: Haupteingang-02 in maximal 33.64 m.

R-018 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR2-EG in maximal 19.45 m.
2. Rettungsweg: Haupteingang-02 in maximal 19.49 m.

R-022 | Empfang
Belegung: 1 Person.
1. Rettungsweg: Haupteingang-01 in maximal 13.49 m.
2. Rettungsweg: Haupteingang-02 in maximal 14.07 m.

R-016 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR2-EG in maximal 14.24 m.
2. Rettungsweg: Haupteingang-02 in maximal 24.99 m.

R-023 | Büro
Belegung: 2 Personen.
1. Rettungsweg: Haupteingang-01 in maximal 17.69 m.
2. Rettungsweg: Haupteingang-02 in maximal 18.28 m.

R-055 | Foyer
Belegung: 0 Personen.
1. Rettungsweg: Haupteingang-02 in maximal 10.36 m.
2. Rettungsweg: Haupteingang-01 in maximal 10.95 m.

R-029 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR1-EG in maximal 15.45 m.
2. Rettungsweg: Haupteingang-01 in maximal 23.9 m.

R-005 | WC-D
Belegung: 0 Personen.
1. Rettungsweg: Haupteingang-02 in maximal 19.99 m.
2. Rettungsweg: TR2-EG in maximal 20.41 m.
```

```
R-026 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR1-EG in maximal 9.45 m.
2. Rettungsweg: Haupteingang-01 in maximal 29.78 m.

R-027 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR1-EG in maximal 10.49 m.
2. Rettungsweg: Haupteingang-01 in maximal 33.64 m.

R-017 | EDV
Belegung: 0 Personen.
1. Rettungsweg: TR2-EG in maximal 16.74 m.
2. Rettungsweg: Haupteingang-02 in maximal 22.61 m.

R-019 | WC-H
Belegung: 0 Personen.
1. Rettungsweg: Haupteingang-01 in maximal 19.99 m.
2. Rettungsweg: TR1-EG in maximal 20.41 m.

R-025 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR1-EG in maximal 13.83 m.
2. Rettungsweg: Haupteingang-01 in maximal 24.99 m.

R-004 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR2-EG in maximal 8.95 m.
2. Rettungsweg: Haupteingang-02 in maximal 29.99 m.

R-028 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR1-EG in maximal 10.24 m.
2. Rettungsweg: Haupteingang-01 in maximal 28.69 m.

R-024 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR1-EG in maximal 17.33 m.
2. Rettungsweg: Haupteingang-01 in maximal 21.49 m.

R-012 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR2-EG in maximal 11.95 m.
2. Rettungsweg: Haupteingang-02 in maximal 27.4 m.

R-020 | Büro
Belegung: 2 Personen.
1. Rettungsweg: Haupteingang-02 in maximal 14.11 m.
2. Rettungsweg: Haupteingang-01 in maximal 14.69 m.
_____________________________________________________________

Längste Rettungswege in diesem Geschoss:

Von Knoten 156 (x: -0.991 y: 11.497) zu TR2-EG.
in Raum: R-018 | Büro
Maximale Rettungsweglänge: 19.45 m.
2. Rettungsweg: Haupteingang-02 (Länge: 19.49 m.)
```

```
Von Knoten 1877 (x: 12.009 y: 13.997) zu Haupteingang-02.
in Raum: R-005 | WC-D
Maximale Rettungsweglänge: 19.99 m.
2. Rettungsweg: TR2-EG (Länge: 20.41 m.)

Von Knoten 1942 (x: 12.509 y: 13.497) zu Haupteingang-01.
in Raum: R-019 | WC-H
Maximale Rettungsweglänge: 19.99 m.
2. Rettungsweg: TR1-EG (Länge: 20.41 m.)

Von Knoten 2378 (x: 16.009 y: 0.497) zu TR1-EG.
in Raum: R-024 | Büro
Maximale Rettungsweglänge: 17.33 m.
2. Rettungsweg: Haupteingang-01 (Länge: 21.49 m.)


Summe des Personenstroms je Rettungsweg:

TR1-EG: 8 Personen.
Haupteingang-02: 2 Personen.
TR2-EG: 11 Personen.
Haupteingang-01: 3 Personen.



************************************************************
                    1. Obergeschoss
************************************************************

Rettungswege für jeden Raum:

R-006 | Abteilungsleitung
Belegung: 1 Person.
1. Rettungsweg: TR2-1OG in maximal 10.99 m.
2. Rettungsweg: TR1-1OG in maximal 45.61 m.

R-033 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR2-1OG in maximal 19.95 m.
2. Rettungsweg: TR1-1OG in maximal 32.28 m.

R-037 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR1-1OG in maximal 13.83 m.
2. Rettungsweg: TR2-1OG in maximal 38.28 m.

R-040 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR1-1OG in maximal 10.24 m.
2. Rettungsweg: TR2-1OG in maximal 41.16 m.

R-041 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR1-1OG in maximal 15.45 m.
2. Rettungsweg: TR2-1OG in maximal 36.36 m.
```

```
R-007 | WC-H
Belegung: 0 Personen.
1. Rettungsweg: TR1-1OG in maximal 20.41 m.
2. Rettungsweg: TR2-1OG in maximal 32.45 m.

R-035 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR1-1OG in maximal 21.45 m.
2. Rettungsweg: TR2-1OG in maximal 30.78 m.

R-032 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR2-1OG in maximal 14.74 m.
2. Rettungsweg: TR1-1OG in maximal 37.78 m.

R-038 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR1-1OG in maximal 9.45 m.
2. Rettungsweg: TR2-1OG in maximal 43.07 m.

R-031 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR2-1OG in maximal 9.45 m.
2. Rettungsweg: TR1-1OG in maximal 42.78 m.

R-043 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR2-1OG in maximal 12.45 m.
2. Rettungsweg: TR1-1OG in maximal 39.36 m.

R-039 | Büro
Belegung: 4 Personen.
1. Rettungsweg: TR1-1OG in maximal 10.49 m.
2. Rettungsweg: TR2-1OG in maximal 46.11 m.

R-036 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR1-1OG in maximal 17.33 m.
2. Rettungsweg: TR2-1OG in maximal 34.78 m.

R-034 | Pausenraum
Belegung: 20 Personen.
1. Rettungsweg: TR2-1OG in maximal 27.78 m.
2. Rettungsweg: TR1-1OG in maximal 27.86 m.

R-042 | Lager
Belegung: 0 Personen.
1. Rettungsweg: TR2-1OG in maximal 17.24 m.
2. Rettungsweg: TR1-1OG in maximal 34.57 m.

R-030 | WC-D
Belegung: 0 Personen.
1. Rettungsweg: TR2-1OG in maximal 20.91 m.
2. Rettungsweg: TR1-1OG in maximal 31.95 m.
```

---

```
Längste Rettungswege in diesem Geschoss:

Von Knoten 203 (x: -0.491 y: 1.997) zu TR2-1OG.
in Raum: R-034 | Pausenraum
Maximale Rettungsweglänge: 27.78 m.
2. Rettungsweg: TR1-1OG (Länge: 27.86 m.)

Von Knoten 268 (x: 0.009 y: 1.497) zu TR1-1OG.
in Raum: R-034 | Pausenraum
Maximale Rettungsweglänge: 27.57 m.
2. Rettungsweg: TR2-1OG (Länge: 28.07 m.)


Summe des Personenstroms je Rettungsweg:

TR2-1OG: 29 Personen.
TR1-1OG: 13 Personen.



************************************************************
                    2. Obergeschoss
************************************************************

Rettungswege für jeden Raum:

R-052 | WC-H
Belegung: 0 Personen.
1. Rettungsweg: TR1-2OG in maximal 23.78 m.
2. Rettungsweg: TR2-2OG in maximal 31.95 m.

R-008 | Geschäftsleitung
Belegung: 1 Person.
1. Rettungsweg: TR2-2OG in maximal 28.16 m.
2. Rettungsweg: TR1-2OG in maximal 29.94 m.

R-048 | Büro
Belegung: 4 Personen.
1. Rettungsweg: TR2-2OG in maximal 10.49 m.
2. Rettungsweg: TR1-2OG in maximal 48.97 m.

R-049 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR2-2OG in maximal 11.33 m.
2. Rettungsweg: TR1-2OG in maximal 43.23 m.

R-044 | Sekretariat
Belegung: 2 Personen.
1. Rettungsweg: TR2-2OG in maximal 21.99 m.
2. Rettungsweg: TR1-2OG in maximal 31.11 m.

R-011 | Konferenzraum
Belegung: 32 Personen.
1. Rettungsweg: TR1-2OG in maximal 15.16 m.
2. Rettungsweg: TR2-2OG in maximal 28.04 m.
```

```
R-047 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR2-2OG in maximal 7.24 m.
2. Rettungsweg: TR1-2OG in maximal 47.44 m.

R-051 | WC-D
Belegung: 0 Personen.
1. Rettungsweg: TR2-2OG in maximal 20.41 m.
2. Rettungsweg: TR1-2OG in maximal 35.31 m.

R-046 | Büro
Belegung: 1 Person.
1. Rettungsweg: TR2-2OG in maximal 10.74 m.
2. Rettungsweg: TR1-2OG in maximal 44.23 m.

R-045 | Geschäftsleitung
Belegung: 1 Person.
1. Rettungsweg: TR2-2OG in maximal 21.07 m.
2. Rettungsweg: TR1-2OG in maximal 38.38 m.

R-053 | Stuhllager
Belegung: 0 Personen.
1. Rettungsweg: TR1-2OG in maximal 10.86 m.
2. Rettungsweg: TR2-2OG in maximal 47.52 m.

R-054 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR1-2OG in maximal 23.4 m.
2. Rettungsweg: TR2-2OG in maximal 31.28 m.

R-050 | Büro
Belegung: 2 Personen.
1. Rettungsweg: TR2-2OG in maximal 15.45 m.
2. Rettungsweg: TR1-2OG in maximal 39.23 m.

_____________________________________________________________

Längste Rettungswege in diesem Geschoss:

Von Knoten 398 (x: 1.009 y: 0.497) zu TR2-2OG.
in Raum: R-008 | Geschäftsleitung
Maximale Rettungsweglänge: 28.16 m.
2. Rettungsweg: TR1-2OG (Länge: 29.94 m.)

Von Knoten 1942 (x: 12.509 y: 13.497) zu TR1-2OG.
in Raum: R-052 | WC-H
Maximale Rettungsweglänge: 23.78 m.
2. Rettungsweg: TR2-2OG (Länge: 31.95 m.)


Summe des Personenstroms je Rettungsweg:

TR1-2OG: 34 Personen.
TR2-2OG: 13 Personen.
```

### C.2.2 Rettungswegermittlung bei 0,45 m Knotenabstand

```
************************************************************
                    Erdgeschoss
************************************************************

Längste Rettungswege in diesem Geschoss:

Von Knoten 173 (x: -1.066 y: 11.672) zu TR2-EG.
in Raum: R-018 | Büro
Maximale Rettungsweglänge: 19.49 m.
2. Rettungsweg: Haupteingang-02 (Länge: 19.6 m.)

Von Knoten 2295 (x: 11.984 y: 13.922) zu Haupteingang-02.
in Raum: R-005 | WC-D
Maximale Rettungsweglänge: 19.97 m.
2. Rettungsweg: TR2-EG (Länge: 20.17 m.)

Von Knoten 2367 (x: 12.434 y: 13.472) zu Haupteingang-01.
in Raum: R-019 | WC-H
Maximale Rettungsweglänge: 19.97 m.
2. Rettungsweg: TR1-EG (Länge: 20.17 m.)

Von Knoten 2922 (x: 16.034 y: 0.422) zu TR1-EG.
in Raum: R-024 | Büro
Maximale Rettungsweglänge: 17.5 m.
2. Rettungsweg: Haupteingang-01 (Länge: 21.59 m.)


Summe des Personenstroms je Rettungsweg:

TR2-EG: 11 Personen.
Haupteingang-02: 2 Personen.
Haupteingang-01: 3 Personen.
TR1-EG: 8 Personen.
```

### C.2.3 Rettungswegermittlung bei 0,42 m Knotenabstand

```
************************************************************
                    Erdgeschoss
************************************************************


Längste Rettungswege in diesem Geschoss:

Von Knoten 2450 (x: 11.069 y: 12.977) zu Haupteingang-02.
in Raum: R-005 | WC-D
Maximale Rettungsweglänge: 18.71 m.
2. Rettungsweg: TR2-EG (Länge: 19.0 m.)

Von Knoten 2527 (x: 11.489 y: 12.557) zu Haupteingang-01.
in Raum: R-019 | WC-H
Maximale Rettungsweglänge: 18.71 m.
2. Rettungsweg: TR1-EG (Länge: 19.0 m.)

Von Knoten 2609 (x: 11.909 y: 14.237) zu TR2-EG.
in Raum: R-005 | WC-D
Maximale Rettungsweglänge: 20.02 m.
2. Rettungsweg: Haupteingang-02 (Länge: 20.08 m.)

Von Knoten 2763 (x: 12.749 y: 13.397) zu TR1-EG.
in Raum: R-019 | WC-H
Maximale Rettungsweglänge: 20.02 m.
2. Rettungsweg: Haupteingang-01 (Länge: 20.08 m.)


Summe des Personenstroms je Rettungsweg:

TR2-EG: 11 Personen.
Haupteingang-02: 2 Personen.
Haupteingang-01: 3 Personen.
TR1-EG: 8 Personen.
```

## C.3 Grafische Ausgabe für das Beispielgebäude

### C.3.1 Rettungswegermittlung bei 0,50 m Knotenabstand

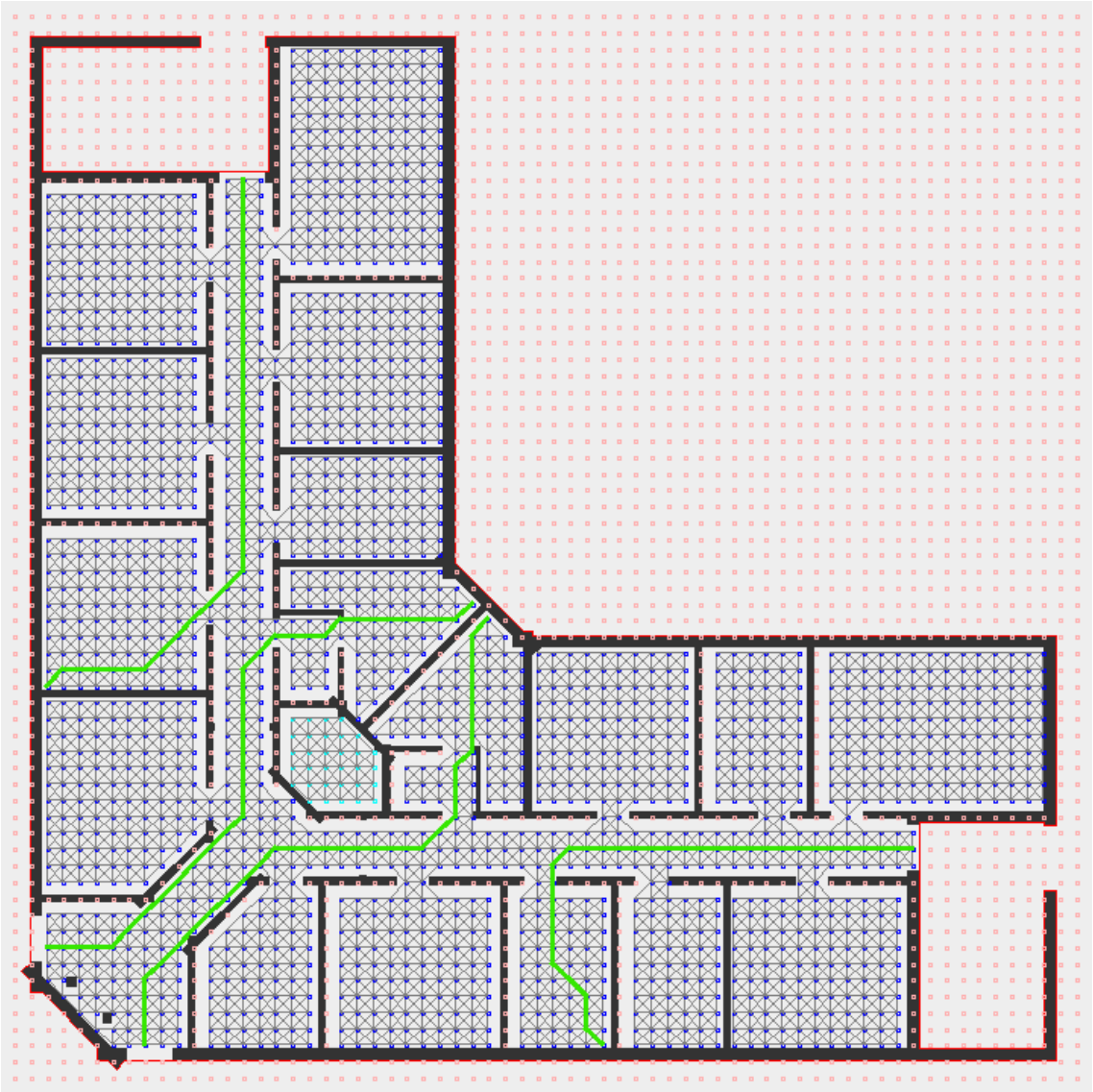


Abbildung C.4: Grafische Ausgabe der Rettungswegermittlung für das Erdgeschoss

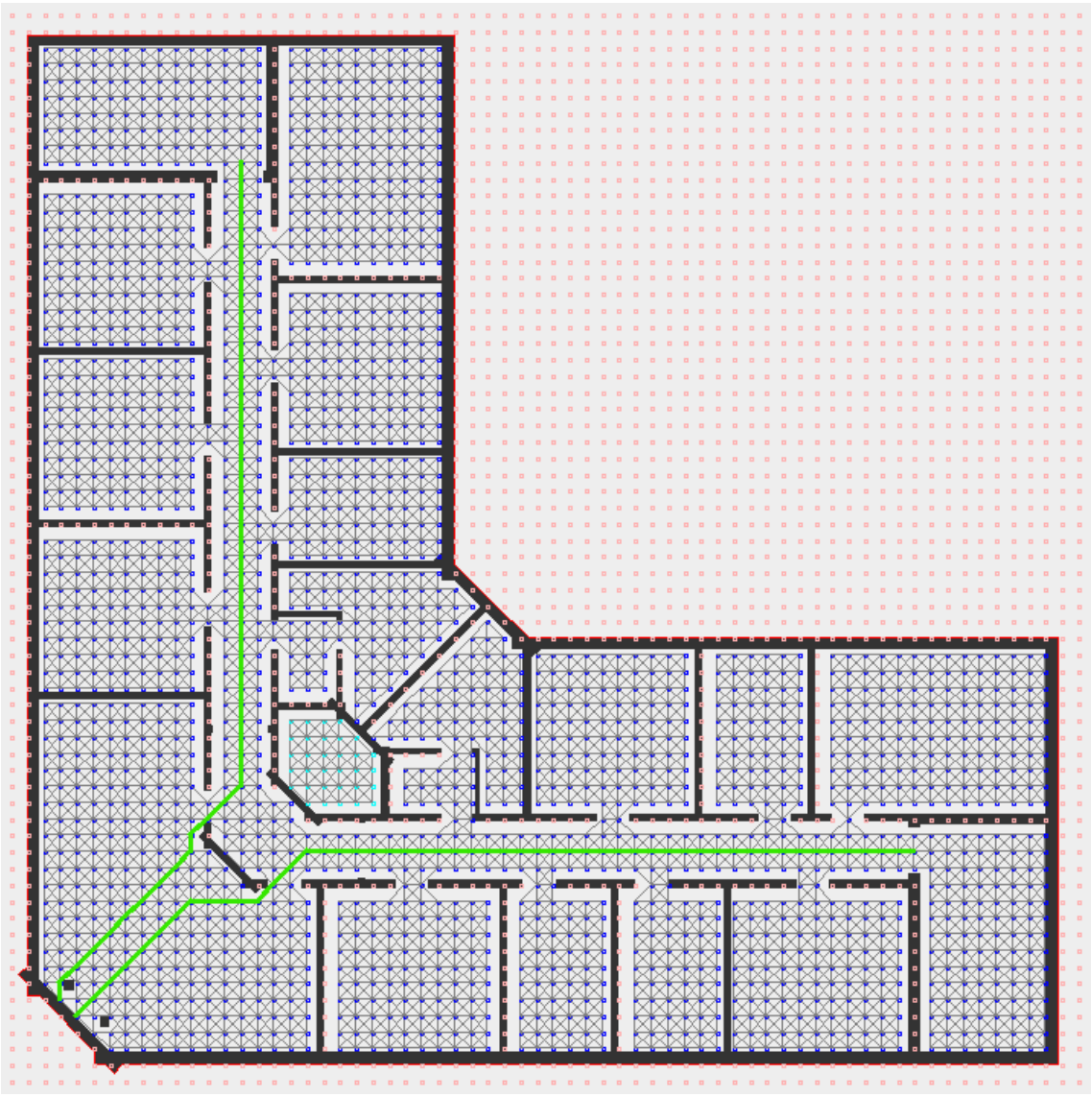


Abbildung C.5: Grafische Ausgabe der Rettungswegermittlung für das 1. Obergeschoss

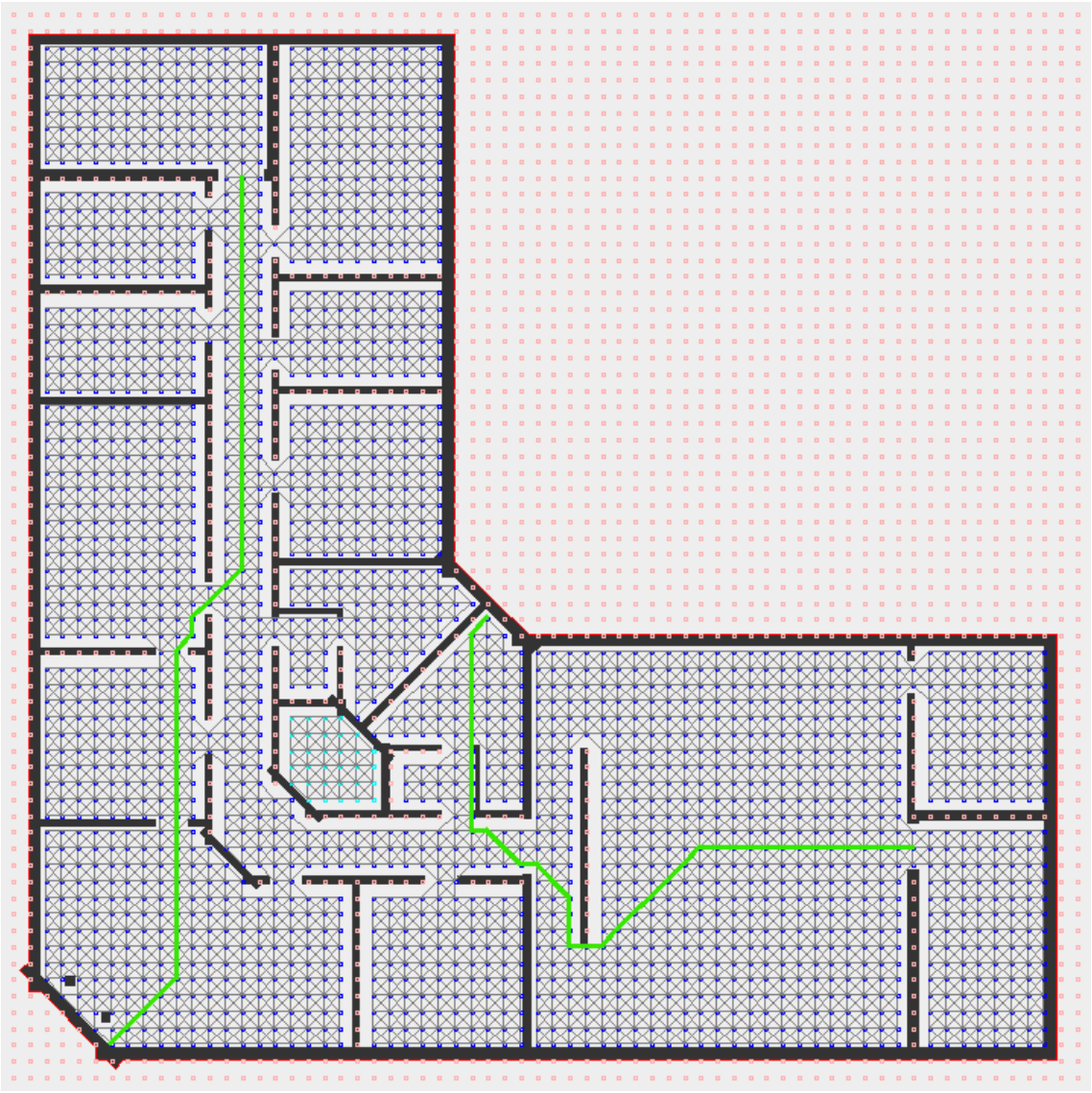
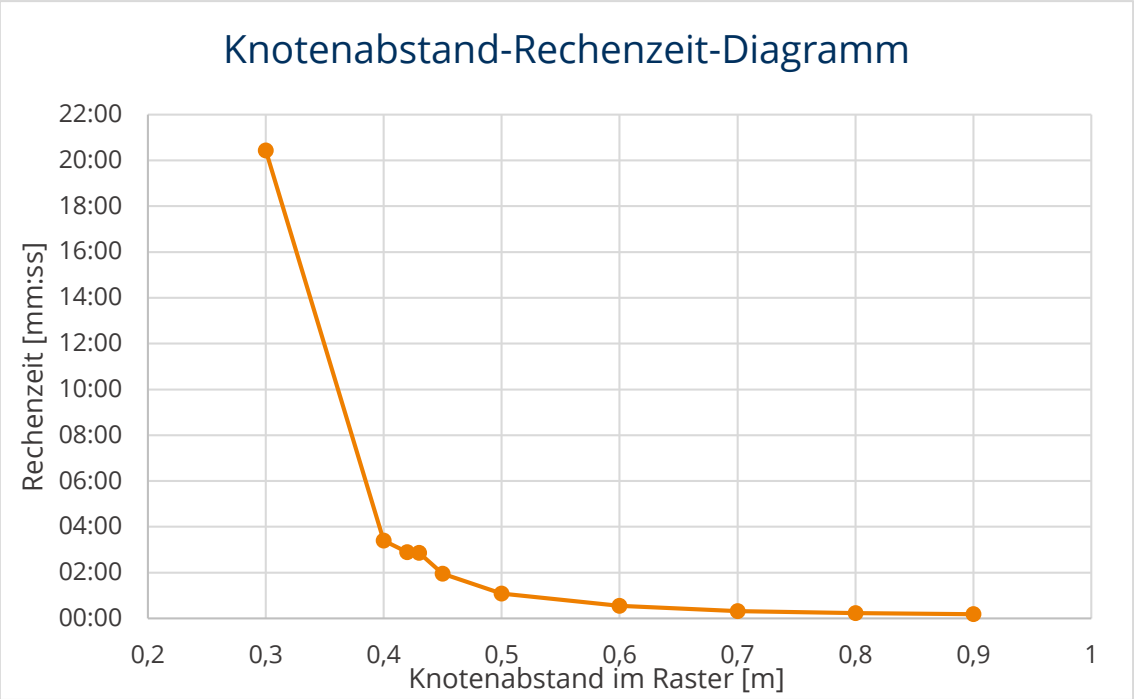


Abbildung C.6: Grafische Ausgabe der Rettungswegermittlung für das 2. Obergeschoss

## C.4 Ermittlung der Rechenzeit

| **Knotenabstand** [m] | **Rechenzeit** [mm:ss] |
|---|---|
| 0,9 | 00:11 |
| 0,8 | 00:14 |
| 0,7 | 00:19 |
| 0,6 | 00:33 |
| 0,5 | 01:05 |
| 0,45 | 01:57 |
| 0,43 | 02:52 |
| 0,42 | 02:54 |
| 0,4 | 03:24 |
| 0,3 | 20:26 |




| | |
|---|---|
| Datum: | Samstag, 06.04.2019 |
| PC: | HP EliteBook 2560p |
| Prozessor: | Intel Core i5-2520M, 2.50 GHz |
| Beispieldatei: | AC21_BeispielobjektDAnew.ifc |

# D Minimalbeispiele

## D.1 Textausgabe zum Minimalbeispiel „Aufteilung des Personenstroms“

```
************************************************************
                    Erdgeschoss
************************************************************

Rettungswege für jeden Raum:

R-002 | Veranstaltungssaal
Belegung: 240 Personen.
1. Rettungsweg: Tür-001 in maximal 9.95 m.
2. Rettungsweg: Tür-002 in maximal 9.95 m.

____________________________________________________________

Längste Rettungswege in diesem Geschoss:

Von Knoten 357 (x: 8.885 y: 6.885) zu Tür-001.
in Raum: R-002 | Veranstaltungssaal
Maximale Rettungsweglänge: 9.95 m.
2. Rettungsweg: Tür-002 (Länge: 9.95 m.)

Von Knoten 375 (x: 9.385 y: 6.885) zu Tür-002.
in Raum: R-002 | Veranstaltungssaal
Maximale Rettungsweglänge: 9.45 m.
2. Rettungsweg: Tür-001 (Länge: 10.45 m.)


Summe des Personenstroms je Rettungsweg:

Tür-001: 240 Personen.
```

## D.2 Textausgabe zum Minimalbeispiel „Falscher zweiter Rettungsweg“

```
************************************************************
                    Erdgeschoss
************************************************************

Rettungswege für jeden Raum:

R-001 | Büro
Belegung: 2 Personen.
1. Rettungsweg: Tür-001 in maximal 8.86 m.
2. Rettungsweg: Tür-002 in maximal 11.86 m.

R-002 | Büro
Belegung: 3 Personen.
1. Rettungsweg: Tür-002 in maximal 8.36 m.
2. Rettungsweg: Tür-001 in maximal 11.36 m.

____________________________________________________________

Längste Rettungswege in diesem Geschoss:

Von Knoten 36 (x: 3.226 y: 0.044) zu Tür-001.
in Raum: R-001 | Büro
Maximale Rettungsweglänge: 8.86 m.
2. Rettungsweg: Tür-002 (Länge: 11.86 m.)

Von Knoten 597 (x: 19.726 y: 0.044) zu Tür-002.
in Raum: R-002 | Büro
Maximale Rettungsweglänge: 8.36 m.
2. Rettungsweg: Tür-001 (Länge: 11.36 m.)


Summe des Personenstroms je Rettungsweg:

Tür-001: 2 Personen.
Tür-002: 3 Personen.
```