Institut für Rechnergestützte Ingenieursysteme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3356

# Entwicklung von Layout-Konzepten und Algorithmen zur komponentenneutralen Beschreibung von Materialflusssystemen

Benjamin Behrens

| | |
|---|---|
| **Studiengang:** | Softwaretechnik |
| **Prüfer:** | Univ-Prof. Hon-Prof. Dr. Dieter Roller<br>Prof. Dr.-Ing. Peter Klemm |
| **Betreuer:** | Dipl.-Inf. Akram Chamakh<br>Dipl.-Ing. Adrian Neyrinck |
| **begonnen am:** | 11. Juni 2012 |
| **beendet am:** | 11. Dezember 2012 |
| **CR-Klassifikation:** | F.2.2, I.2.9, I.6.7 |

# Abstract

Nowadays when designing an industrial facility, different versions for the realization of the material flow are build virtually and simulated by corresponding software tools. In the course of this, various variants for the realization of the industrialfacility are modeled manually on the basis of several components for purposes of comparison.

The approach of component-neutral description does not include the modeling of the components, but only specifies the required material flow. On this basis, the different variants are calculated automatically based on the available components. Thus, only a single modeling effort is required.

In this paper, an approach to implementing the component-neutral description is presented by means of a prototype. First, the characteristics of the material flow to be modeled are analyzed and how it can be presented, and what further requirements are resulting from this task.

Furthermore, a basic structure in the EPLAN Engineering Center is designed to describe the modeled components. The material flow and its surroundings are then modeled by an editor and mapped to the completed structure.

From the modeled material flow various variants of predefined groups of components are calculated. For this purpose, different algorithms for pathfinding are examined. One approach of this algorithms is adopted, modified and enhanced in order to implement the previously established requirements.

The results of the algorithm are represented by a structure and a graphical depiction of each variant.

# Kurzfassung

Bei der Planung von Industrieanlagen werden heutzutage verschiedene Möglichkeiten zu der Realisierung des Materialflusses virtuell erstellt und mit entsprechenden Softwarewerkzeugen simuliert. Dabei werden meist verschiedene Varianten der umzusetzenden Industrieanlage einzeln anhand von vorhandenen Komponenten erstellt, um diese anschließend vergleichen zu können.

Der Ansatz der komponentenneutralen Beschreibung sieht vor, nicht die Komponenten zu modellieren, sondern ausschließlich den notwendigen Materialfluss anzugeben. Anhand von diesem werden die verschiedenen Varianten der zu Verfügung stehenden Komponenten automatisch berechnet. Somit ist nur ein einmaliger Modellierungsaufwand notwendig.

In dieser Arbeit wird ein Ansatz zur Umsetzung der komponentenneutralen Beschreibung unter Verwendung eines Prototypen vorgestellt. Hierzu wird zunächst analysiert, welche Eigenschaften der zu modellierende Materialfluss aufweisen muss, wie dieser dargestellt werden kann und welche weiteren Anforderungen sich aus der Aufgabenstellung zur Umsetzung des Materialflusses ergeben.

Weiterhin wird eine grundlegende Struktur im *EPLAN Engineering Center* zur Beschreibung der modellierten Komponenten entworfen. Der Materialfluss und dessen Umgebung wird anschließend über einen Editor modelliert und auf die erstellte Struktur abgebildet.

Aus dem modellierten Materialfluss werden verschiedene Varianten aus vorgegebenen Gruppen von Komponenten berechnet. Hierzu werden verschiedene Wegfindungsalgorithmen untersucht. Ein Ansatz dieser Algorithmen wird übernommen und entsprechend modifiziert und erweitert, um die vorher ermittelten Anforderungen umzusetzen.

Die Ergebnisse des Algorithmus werden über eine Struktur und eine grafische Abbildung jeder Variante dargestellt.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis



# Verzeichnis der Algorithmen

# 1 Einleitung

Dieses Kapitel gibt einen Einblick in den Hintergrund und die Aufgabenstellung dieser Arbeit. Weiterhin wird der Projektverlauf und dessen Planung kurz erläutert.

## 1.1 Hintergrund

Bevor eine Industrieanlage gebaut wird, werden üblicherweise zunächst Angebote von verschiedenen Anbietern eingeholt, welche ein oder mehrere Konzepte zur Lösung einer gegebenen Problemstellung vorstellen. Diese werden meist anhand von spezieller Software virtuell erstellt und deren Verhalten simuliert.

Um fördertechnische Aufgaben umzusetzen, werden heutzutage verschiedene Transfersysteme verwendet, um Material zu transportieren und zu lagern. Diese Transfersysteme sind oft in verschiedene Gruppen gegliedert, innerhalb der diese beliebig kombiniert werden können. Welche Transfersysteme dabei genau Verwendung finden, hängt dabei unter anderem von den zu Verfügung stehenden Transfersystemen, der umzusetzenden Aufgabe, der Umgebung und schließlich auch von subjektiven Vorgaben des Planers ab.

Eine solche Problemstellung kann daher anhand von unterschiedlichen Transfersystemen erfolgreich gelöst werden. Welche der vorgeschlagenen Lösungen sich am Besten für eine tatsächliche Umsetzung eignet, kann dabei nicht ohne weiteres geklärt werden. Die Anschaffungskosten der verschiedenen Lösungen werden sich wahrscheinlich unterscheiden, jedoch spielen neben diesen auch weitere Faktoren eine Rolle. So könnten zum Beispiel die zu erwartenden kompletten Kosten innerhalb eines Lebenszyklus der Maschinen oder auch die Energieeffizienz der Industrieanlage von Bedeutung sein.

Daher sollen möglichst mehrere Lösungen mittels unterschiedlicher Transfersysteme erstellt beziehungsweise simuliert werden, um diese anschließend anhand der geforderten Kriterien analysieren und auswerten zu können. Jede solche Lösung wird dabei als eine Variante der (zunächst weiterhin virtuellen) Industrieanlage bezeichnet.

## 1.2 Motivation und Zielsetzung

Bei Konzeption und Entwurf von Industrieanlagen wird bisher meist der Ansatz verfolgt, in einer modellierten Umgebung anhand vorgegebener Komponenten eine möglichst gute Realisierung der gegebenen Aufgabenstellung umzusetzen. Die so erstellte Lösung kann heute anhand guter Simulationswerkzeuge schnell ausgewertet und bei Bedarf angepasst werden. Bei der Modellierung der Lösung wird sich dabei

oft auf Erfahrungswerte verlassen, was jedoch auch zu einer subjektiven Prägung der Lösung führt. Ein großer Teil des Entwicklungsaufwandes besteht dabei aus der Modellierung und Anpassung der verwendeten Komponenten sowie deren Anordnung.

Oft ist es jedoch von Vorteil, nicht nur eine Lösung für eine geforderte Aufgabe zu erstellen, sondern eine Umsetzung anhand verschiedener Komponenten zu simulieren. Ein Grund hierfür kann zum Beispiel sein, dass nicht im Vorfeld eindeutig zu klären ist, welches Fördersystem sich zur Umsetzung der Aufgabe mit den erforderlichen Vorgaben am Besten eignet. Möchte man mehrere Lösungen vergleichen, so muss jede von ihnen einzeln modelliert werden, was einen hohen Arbeitsaufwand erfordert.

Ein Ansatz um dies zu umgehen, ist die komponentenneutrale Beschreibung der gegebenen Problemstellung. Dabei wird statt konkreten Transfersystemen ein Materialfluss modelliert, welcher um geforderte Funktionalitäten an definierten Punkten ergänzt werden kann. Dabei müssen mindestens die Teilfunktionen der Handhabungsfunktion (siehe 2.2) umgesetzt werden, um eine ausreichende Beschreibung des Materialflusses zu gewährleisten.

Aus der Beschreibung des Materialflusses können anschließend beliebige Varianten der Transfersysteme mit unterschiedlichen Komponenten erzeugt werden, sofern deren Komponentengruppen bekannt sind.

Ziel dieser Arbeit ist es, die Grundlagen zur Umsetzung der komponentenneutralen Beschreibung und der Variantenbildung zu entwerfen und anhand eines Prototyps eine eingeschränkte Minimalversion dieses Ansatzes umzusetzen. Dies bedeutet vor allem, dass zunächst nur grundlegende Eigenschaften von Transfersystemen berücksichtigt werden sollen und diese auf einfache geometrische Formen abgebildet werden. Das Konzept soll dabei jedoch so ausgelegt sein, dass es später um weitere Funktionalitäten ergänzt werden kann und durch Anpassungen auch die Verwendung komplexerer Geometrien ermöglicht wird.

Hierzu muss zunächst eine Struktur entworfen werden, welche eine Beschreibung des Materialflusses abbildet. Diese Struktur soll dabei die vorhandenen Bausteine des *EPLAN Engineering Centers* (siehe 2.3.1) nutzen. Entsprechend werden mittels dieses Werkzeugs auch die später erforderlichen Komponenten erstellt.

Weiterhin soll ein Editor zur grafischen Erstellung des Materialflusses erstellt werden, welcher auf die Struktur abgebildet werden kann. Anhand dieser Struktur sollen automatisch passende Instanzen der Bausteine mitsamt deren Abhängigkeiten generiert werden.

Aus dieser modellierten Beschreibung sind anschließend Varianten zu berechnen. Hierzu muss ein passender Wegfindungsalgorithmus gefunden beziehungsweise

entwickelt werden, welcher den Ansprüchen zur Berechnung der Wege anhand beliebiger Komponenten genügt. Die dabei verwendeten Komponenten sollen anschließend ebenfalls erzeugt und in einer stark vereinfachten Darstellung grafisch angezeigt werden. Für jede der zu erstellenden Varianten ist dabei eine grafische Ausgabe zu erzeugen.

## 1.3 Kooperationspartner

Das vom Ministerium für Wirtschaft und Finanzen Baden-Württemberg geförderte Forschungsprojekt „*Simulationsgestützte Berücksichtigung von Varianten bei der Konzeption von Maschinen und Anlagen – SimVar*“soll Methoden und Werkzeuge für eine Variantenbildung und den simulationsgestützten Vergleich von diesen entwickeln. [Ney12]

Geleitet wird das Projekt vom *Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart (ISW)*. Ebenso beteiligt sind das *Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA)*, die *Mind8 GmbH & Co. KG*, die *Industrielle Steuerungstechnik GmbH (ISG)* und die *Schnaithmann Maschinenbau GmbH*. [Sch12]

Die hier vorgestellte Arbeit findet ergänzend zu diesem Projekt statt und die gegebene Aufgabenstellung wurde in Kooperation mit dem an dem Projekt beteiligten Mitarbeitern des ISW ausgegeben. Dabei fungiert Dipl.-Ing. Adrian Neyrinck als zusätzlicher Betreuer für diese Ausarbeitung.

Weiterhin besteht ein enger Kontakt zu Mitarbeitern der Firma Mind8, welche für die Entwicklung des EPLAN Engineering Centers (siehe 2.3.1) verantwortlich sind. Dabei erfolgt eine Zusammenarbeit vorrangig im Bereich der Erstellung der Struktur und des Materialflusseditors anhand deren Anwendung.

## 1.4 Projektverlauf

Das Vorgehen des Projektes zur Erstellung dieser Arbeit wurde in drei verschiedene Abschnitte eingeteilt. Der erste Abschnitt umfasste die Konzeption und Umsetzung einer geeigneten Struktur und die Darstellung des Materialflusses, der zweite Abschnitt die Analyse und Umsetzung von Wegfindungsalgorithmen anhand dieser Struktur. Im dritten Abschnitt wurde dieses Dokument erstellt und iterativ erweitert.

Die ersten beiden Abschnitte unterteilten sich in weitere Phasen, welche mit dem Vorgehen eines vereinfachten Wasserfallmodells realisiert wurden. [Ludo7]

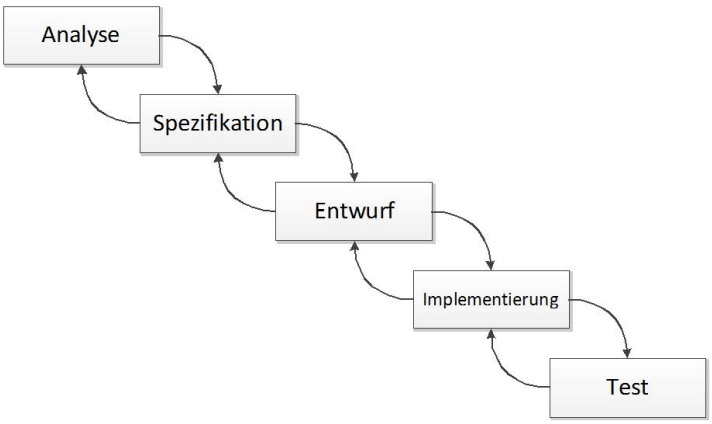



**Abbildung 1.1:** Darstellung des vereinfachten Wasserfallmodells zur Umsetzung der ersten beiden Projektabschnitte

Die geplante Arbeitszeit orientierte sich an den in den Richtlinien der im *Studienplan des Diplomstudiengangs Softwaretechnik* vorgegebenen 900 Stunden [Uni10]. Dies ergibt bei einer 6-monatigen oder 26-wöchigen Gesamtzeit eine wöchentliche Arbeitszeit von etwa 35 Stunden. Aufgrund der Kooperation mit Mitarbeitern von verschiedenen Instituten sowie der Firma Mind8 war dabei eine hohe Flexibilität erforderlich.

Eine genauere zeitliche Abfolge dieses Projektes stellt folgendes Gantt-Diagramm dar.

| Kennung | Aufgabenname | Anfang | Abschluss | Dauer |
|---|---|---|---|---|
| 1 | Analyse – Phase I | 11.06.2012 | 18.06.2012 | 1,2w |
| 2 | Spezifikation – Phase I | 18.06.2012 | 25.06.2012 | 1,2w |
| 3 | Entwurf - Phase I | 27.06.2012 | 06.07.2012 | 1,6w |
| 4 | Implementierung – Phase I | 09.07.2012 | 20.07.2012 | 2w |
| 5 | Test – Phase I | 19.07.2012 | 24.07.2012 | ,8w |
| 6 | Analyse – Phase II | 25.07.2012 | 03.08.2012 | 1,6w |
| 7 | Spezifikation – Phase II | 06.08.2012 | 17.08.2012 | 2w |
| 8 | Entwurf - Phase II | 20.08.2012 | 31.08.2012 | 2w |
| 9 | Implementierung – Phase II | 03.09.2012 | 19.10.2012 | 7w |
| 10 | Test – Phase II | 15.10.2012 | 23.10.2012 | 1,4w |
| 11 | Strukturdesign Diplomarbeit | 26.10.2012 | 26.11.2012 | 4,4w |
| 12 | Inhalt Diplomarbeit | 01.11.2012 | 30.11.2012 | 4,4w |
| 13 | Abgabe Diplomarbeit | 03.12.2012 | 05.12.2012 | ,6w |
| 14 | Puffer | 06.12.2012 | 10.12.2012 | ,6w |



**Abbildung 1.2:** Gantt-Diagramm zur terminlichen Planung des Projektverlaufes

## 1.5 Gliederung

Dieses Dokument ist in verschiedene Kapitel gegliedert, deren Abfolge sich grob an dem Vorgehen während der Durchführung des Projektes orientiert. Zuerst werden die Struktur und die Darstellung des Materialflusses analysiert und erläutert. Anschließend erfolgt eine ähnliche Abfolge zur Berechnung der Varianten unter Zuhilfenahme eines Wegfindungsalgorithmus.

Die Kapitel besitzen folgende Inhalte:

**Kapitel 1** beschreibt den Hintergrund, die allgemeine Aufgabenstellung, die Kooperationspartner und den Projektverlauf zur Erstellung dieser Arbeit.

**Kapitel 2** gibt eine Einführung in die in dieser Arbeit verwendeten Begriffe der Fördertechnik. Zudem werden die verwendeten Werkzeuge beschrieben.

**Kapitel 3** enthält eine Analyse zur genaueren Festlegung und Beschreibung der Anforderungen für das durchgeführte Projekt.

**Kapitel 4** beinhaltet die Konzeptfindung zur Darstellung des Materialflusses.

**Kapitel 5** erläutert die erstellte Datenstruktur zur Umsetzung des Materialflusses und der Berechnung der Varianten.

**Kapitel 6** beschreibt die Erstellung des Materialflusseditors im *EPLAN Engineering Center*.

**Kapitel 7** trifft eine Auswahl von Ansätzen zur Wegfindung innerhalb der modellierten Umgebung und bewertet diese.

**Kapitel 8** enthält eine Beschreibung der programmierten Klassen und erläutert das Vorgehen des implementierten Wegfindungsalgorithmus.

**Kapitel 9** veranschaulicht die Ergebnisse des Algorithmus und zeigt somit die Darstellung der berechneten Varianten.

**Kapitel 10** fasst diese Arbeit zusammen und gibt einen Ausblick über Erweiterungsmöglichkeiten des Algorithmus.

# 2 Grundlagen und Erläuterungen

Dieses Kapitel gibt eine Einführung in die Begrifflichkeiten der Fördertechnik. Zudem wird die für diese Arbeit wichtige *Handhabungsfunktion* erläutert und ein Einblick in die verwendeten Software-Werkzeuge gegeben.

## 2.1 Begriffe der Fördertechnik

In der Fördertechnik werden viele Begriffe unterschiedlich verwendet. Oft wird die genaue Bedeutung dabei nur aus dem Kontext ersichtlich. Daher werden folgend einige wichtige Begriffe erläutert und von einander abgegrenzt, um deren Verwendung im Kontext dieses Dokumentes zu klären. Dabei wird versucht, die Begrifflichkeiten der angegeben Quellen einzuhalten.

Bei der Verwendung dieser Begriffe ist weiterhin zu beachten, dass oft keine Unterscheidung zwischen tatsächlichen Gegenständen und deren Modell in einer Software getroffen wird. Da hier die Umsetzung einer Software beschrieben wird, sind die Begriffe im Zweifelsfall immer als Modell des Gegenstandes zu verstehen.

### Materialfluss

Materialfluss ist als die „*Verkettung aller Vorgänge beim Gewinnen, Be- und Verarbeiten sowie bei der Verteilung von stofflichen Gütern innerhalb festgelegter Bereiche*„definiert [Ver07].

Vereinfacht kann dies als eine Beschreibung der Orte und deren Reihenfolge verstanden werden, welche Material zum Erreichen der verschiedenen Bearbeitungsschritte im Fertigungsprozess ablaufen muss. Dies kann man auch mit folgender Frage beschreiben: Was wird in welcher Reihenfolge wohin transportiert?

### Industrieanlagen

Unter einer Industrieanlage, oder folgend nur Anlage, wird die Gesamtheit von Maschinen und Geräten verstanden, welche zur Umsetzung einer gemeinsamen Aufgabe eingesetzt werden. Je nach Definition der Aufgabe kann dieser Begriff daher sowohl die Maschinen zur Lösung einer Teilaufgabe wie auch das Komplettsystem einer Fertigungsreihe umfassen.

Im Nachfolgenden wird unter dieser Aufgabe die Umsetzung eines durchgängigen Materialflusses verstanden. Somit fallen unter diesen Begriff alle Maschinen, welche ohne Unterbrechungen wie Zwischenlagerung oder Umtransport für die Bewegung des Materials und deren Verarbeitung notwendig sind.

### Materialflusssysteme und Transfersysteme

Die Begriffe Materialflusssystem und Transfersystem sind nur schwer voneinander zu unterscheiden und werden häufig synonym verwendet. Sofern kein Namenskonflikt mit angegebenen oder zitierten Quellen vorliegt, wird hier jedoch der Begriff Transfersystem verwendet.

Unter diesen Begriffen wird nachfolgend ein einzelnes Element zur Umsetzung eines durchgängigen Materialflusses verstanden.

### Bandsysteme und Fördersysteme

Bandsysteme sind eine Untergruppe der Transfersysteme. Wörtlich genommen sind dies alle Transfersysteme, welche den Materialfluss über Förderbänder umsetzen. Häufig werden darunter jedoch alle Transfersysteme verstanden, welche einen Strom aus Materialien auf einer festgelegten Bahn anhand eines angetriebenen Transportnetzes umsetzt. Somit fallen unter diesen Begriff ebenfalls z.B. Platten- und Kettenförderer.

Um eine eindeutigere Bezeichnung zu verwenden, wird daher vorrangig der Begriff *Fördersystem* verwendet.

### Komponenten

Die Bedeutung des Begriffs Komponente weicht je nach Kontext teilweise stark voneinander ab. Hier wird der Begriff als allgemeine Beschreibung für ein nicht weiter zerlegbares Element zur Umsetzung der Handhabungsfunktion (siehe 2.2) verstanden. Darunter fallen somit einzelne Transfersysteme, aber auch z.B. einzelne Sensoren oder Stopper zum Blockieren des Materialflusses an einem festgelegten Punkt.

### Komponentengruppen

Dieser Begriff ist kein allgemein verwendeter Begriff und wird in diesem Dokument zum besseren Verständnis neu eingeführt.

Vereinfachend für die folgende Aufgabe wird angenommen, dass nicht alle Komponenten miteinander kompatibel sind, so kann z.B. ein Bandförderer nicht ohne weiteres mit einem Plattenförderer kombiniert werden. Eine Komponentengruppe enthält nur Komponenten, welche auch miteinander verbunden werden können.

### Varianten (einer Industrieanlage)

Eine Variante ist eine konkrete Kombination verschiedener Komponenten zur Lösung einer fördertechnischen Aufgabe. Für eine Anlage können mehrere Varianten erstellt werden, welche die gleiche Aufgabe erfüllen. Die genaue Umsetzung der Aufgabe unterscheidet sich jedoch innerhalb der Varianten.

Hier wird unter verschiedenen Varianten einer Anlage verstanden, dass der gleiche grundlegende Materialfluss abgebildet wird, jedoch zu dessen Umsetzung unterschiedliche Transfersysteme eingesetzt werden. Für jede Variante wird dabei eine unterschiedliche Komponentengruppen zur Umsetzung des Materialflusses verwendet.

In Bezug auf eine Industrieanlage wird der Begriff Variante außerhalb dieses Dokuments selten verwendet.

## 2.2 Handhabungsfunktionen

Die Richtlinie 2860 des *Vereins Deutscher Ingenieure (VDI)* beschreibt die Funktion *Handhaben* und definiert deren Teilfunktionen. Handhaben ist eine Teilfunktion des Materialflusses und dient dessen Beschreibung.

*Handhaben* ist als *„Schaffen, Verändern oder Aufrechterhalten einer räumlichen Anordnung“*[Ver90] definiert.

Die räumliche Anordnung wird durch zwei Größen bestimmt:

- **Orientierung:** Gibt anhand eines kartesischen körpereigenen Koordinatensystems die Ausrichtungen des Körpers (u,v,w) an. Sie soll in Bezug auf seine drei rotatorischen Freiheitsgrade jeweils als Winkelgrad (Eulersche Winkel) angegeben werden. Dabei gibt der Orientierungsgrad an, wie viele der drei Winkel bestimmt werden.
- **Position:** Der Ort den ein körpereigener Punkt (in der minimalen Hüllfläche des Körpers) einnimmt. Er wird als Weglänge (Positionsvektor) des kartesischen umgebenden Koordinatensystems (x,y,z) angegeben. Dabei gibt der Positionsgrad an, wie viele der drei Weglängen bestimmt werden.

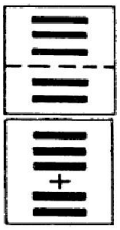
Der Ordnungszustand wird durch die Angabe der Kennzahlen Orientierungsgrad und Positionsgrad angegeben (z.B. 3/3). Eine Handhabungsfunktion muss per Definition mindestens einen Ordnungszustand von 1/1 besitzen. [Ver90]

In dem aktuellen Projekt soll der höchste Ordnungszustand aus Gründen der Vereinfachung mit 2/2 angenommen werden. Dies entspricht einer zweidimensionalen Darstellung.

Die Funktion *Handhaben* unterteilt sich in weitere *Teilfunktionen*, welche wiederum aus verschiedenen *Unterfunktionen* bestehen. Folgend werden nur die *Elementarfunktionen* erläutert, welche nicht weiter zerlegbar sind.

## Teilfunktion *Mengen verändern*

Steht für das Verändern, also Vereinigen oder Teilen, von Mengen. Diese sind nicht zwingend Handhabungsfunktionen, sondern nur, falls sich auf bestimmte Körper mit Orientierungsgrad >= 1 bezogen wird.

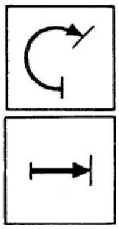


**Unterfunktion Teilen**
Das Bilden von Teilmengen aus einer Menge.

**Unterfunktion Vereinigen**
Das Zusammenführen von Teilmengen zu einer Menge.

## Teilfunktion *Bewegen*

Verändert die Orientierung oder Position von Körpern.



**Unterfunktion Drehen**
Verändern der Orientierung eines Körpers.

**Unterfunktion Verschieben**
Verändern der Position eines Körpers.

## Teilfunktion *Sichern*

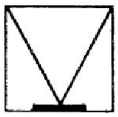
Dient zum Erhalten definierter Zustände beziehungsweise der räumlichen Anordnung von Körpern. Es handelt sich hierbei nur um eine Handhabungsfunktion, wenn diese Sicherung nur vorübergehend ist.

**Unterfunktion Halten**
Sichern eines Körpers in einer bestimmten Position und Orientierung.

**Unterfunktion Lösen**
Auflösen der Sicherung des Haltens.

## Teilfunktion *Kontrollieren*

Dient zum Messen der Anwesenheit von Material oder dessen Eigenschaften.



**Unterfunktion Prüfen**
Feststellen, ob der Körper vorgegebene Bedingungen erfüllt.

## 2.3 Verwendete Werkzeuge

Die Umsetzung der beschriebenen Arbeit erfolgt unter Zuhilfenahme verschiedener Software-Werkzeuge. Das *EPLAN Engineering Center* dient zur Modellierung der Struktur und Erstellung von grafischen Diagrammen. Anhand des Simulations-Werkzeugs *virtuos* werden die erstellten Ergebnisse veranschaulicht und simuliert.

### 2.3.1 EPLAN Engineering Center

Das *EPLAN Engineering Center (EEC)* ist eine Softwarelösung zur baukastenbasierten Beschreibung von Engineering-Prozessen. Durch eine spezielle Beschreibung der Anlagen soll eine Brücke zwischen Elektrotechnik, Mechanik, Steuerungstechnik und Dokumentation geschlagen werden.

Eine geplante Anlage wird in funktionale Baukästen unterteilt. Diese Baukästen werden innerhalb einer Bibliothek als Baumstruktur dargestellt und setzen entsprechende Standards zur Erhöhung der Wiederverwendbarkeit um. Durch die Kommunikation mit anderen Systemen mit Hilfe verschiedener Schnittstellen können auch bereits vorhandene Beschreibungen eingebunden und deren Funktionalität genutzt werden.

Die Baukästen werden dabei allgemein beschrieben und deren Verhalten über Parameter angepasst. So können funktional ähnliche Baukästen durch Anpassung der Parameter wiederverwendet werden, ohne das eine Beschreibung eines neuen Baukastens notwendig wird. [EPL12]

Das EEC trennt zwischen allgemeinen Beschreibungen der Baukästen samt deren Abhängigkeiten und den aus diesen anhand der Parameter gebildeten Instanzen. Hierfür werden Projekte erstellt, in welche Instanzen der Baukästen geladen werden können. Durch Angabe der Parameter der Instanzen können diese dann genauer spezifiziert werden, wobei deren Grundfunktionalität erhalten bleibt. Dies kann als ein objektorientierter Ansatz interpretiert werden, wobei die Baukästen als Klassen und deren Instanzen als Objekte zu verstehen sind.

Die Funktionalitäten der Baukästen können sowohl über Formeln in den Parametern als auch durch die Einbindung von Java Code erweitert werden. Der Code kann entweder direkt bei den Baukästen angegeben oder über ein Plugin eingebunden werden. Letzteres empfiehlt sich vor allem bei komplexeren Funktionalitäten.

Anhand von disziplinspezifischen Baukästen, von denen die neu erstellten Baukästen abgeleitet werden, können Software- und Hardware-Informationen generiert werden. Hierzu gehören beispielsweise Elektro-Schaltpläne, Sensor-/ Aktorlisten,

Stücklisten, Mechanik-Zeichnungen, technische Beschreibungen oder auch Software zu Steuerung und Regelung von Maschinen und Anlagen.

Weiterhin bietet das EEC Baukästen zur Generierung von grafischen Diagrammen und Editoren. Sowohl die Diagramme als auch weiterführende Oberflächen zur Anzeige von Daten der Baukästen werden dabei über ein vorgegebenes XML-Schema modelliert. [Cas12]

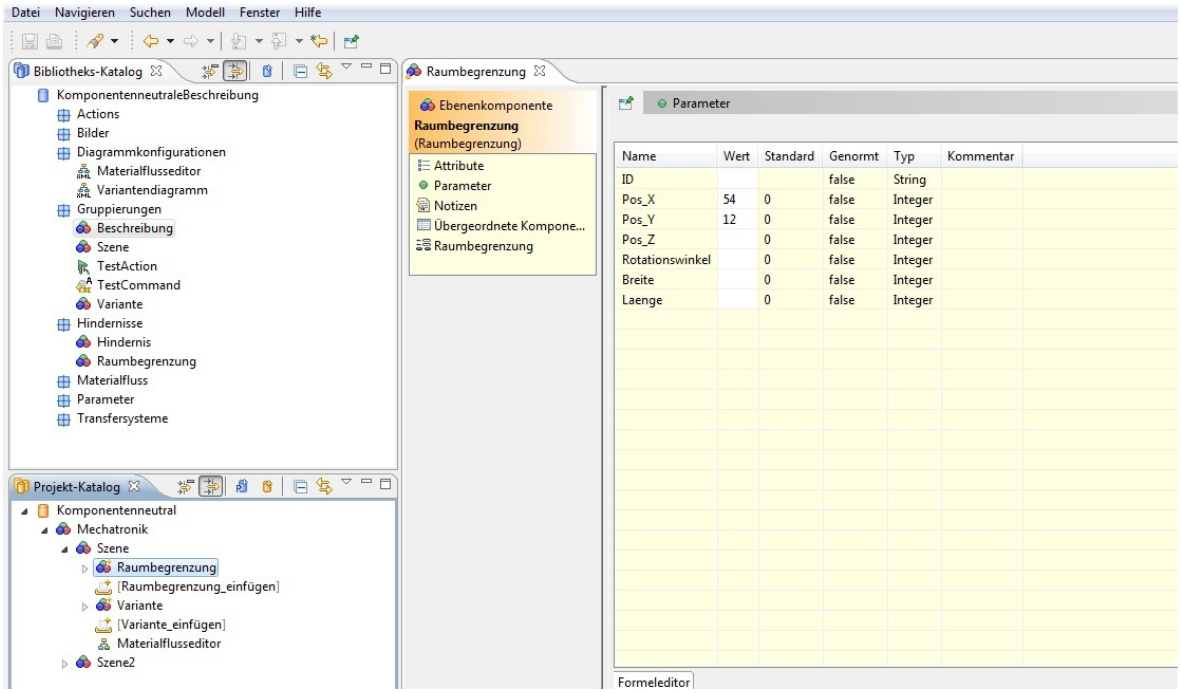



**Abbildung 2.1:** Ausschnitt des EPLAN Engineering Centers (Bereiche Bibliothek, Projekte, Parametereingabe)

### 2.3.2 ISG-virtuos

*Virtuos* ist eine Software der *Industriellen Steuerungstechnik GmbH (ISG)* zur Simulation von Maschinen und Anlagen in Echtzeit. Dabei können sowohl virtuelle Maschinen als auch reale Anlagen mit *virtuos* verbunden werden, um das Verhalten von Anlagen zu simulieren und zu analysieren.

Die Anlagen können dabei über einen grafischen Editor erstellt oder über eine objektorientierte XML-Modellbeschreibung importiert werden. Somit können auch

Modelle von anderen Softwarelösungen wie dem EPLAN Engineering Center simuliert werden, sofern diese eine Exportfunktion in dieses Format aufweisen.

Neben dem Ansprechen von Maschinensteuerungen existiert auch eine grafische Materialflussdarstellung. Somit kann auch der korrekte Ablauf der Materialsysteme angezeigt und überprüft werden. [ISG12]

Bei der Umsetzung der in diesem Dokument geschilderten Aufgabe wird *virtuos* ausschließlich zur Veranschaulichung des Materialflusses verwendet. Dazu werden die im *EPLAN Engineering Center* erstellten Modelle in *virtuos* importiert.

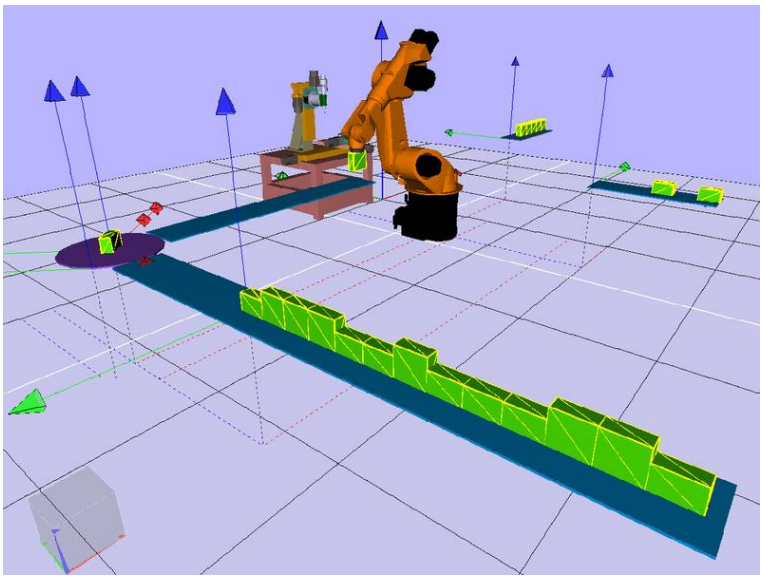


**Abbildung 2.2:** Grafische Darstellung von Materialfluss in virtuos [ISG12]

# 3 Anforderungsanalyse und -beschreibung

Alle beschriebenen Teilaufgaben der Aufgabenstellung (siehe 1.2) wurden im Vorfeld nur geringfügig eingeschränkt, um einen möglichst großen Freiraum bei deren Gestaltung zu erlauben. Daher werden weitere Anforderungen und Einschränkungen erarbeitet, welche den zeitlichen Rahmen dieser Arbeit berücksichtigen.

Die ermittelten Anforderungen werden in den folgenden Unterkapiteln genauer erläutert. Dabei werden zur Vereinfachung teilweise Annahmen getroffen, welche die Umsetzung der Aufgabe erleichtern.

Aufgrund der Komplexität besonders im Bereich des Algorithmus zur Berechnung der Varianten werden nicht alle Anforderungen tatsächlich von dieser Arbeit umgesetzt. Erkannte Anforderungen werden zur Erleichterung von eventuellen folgenden Erweiterungen dieser Arbeit dennoch aufgeführt und in späteren Kapiteln mögliche Ansätze zu deren Lösungen vorgeschlagen.

## 3.1 Anforderungen an die Struktur

Die zu erstellende Struktur ist im *EPLAN Engineering Center* anhand von verschiedenen Baukästen zu erstellen. Die Struktur soll eine Darstellung und Speicherung aller notwendigen Eigenschaften zur Modellierung eines Materialflusses enthalten. Dabei ist zu berücksichtigen, dass die grundlegenden Teilfunktionen der Handhabungsfunktion ( siehe 2.2) umgesetzt werden können.

Die Struktur soll zudem eine Hierarchie vorgeben, welche die Abhängigkeiten der erstellten Baukästen abbildet und eine fehlerhafte Modellierung ausschließt.

In der Tabelle 3.1 werden die zwingend zu erstellenden Baukästen und deren Funktion für den Materialfluss kurz erläutert.

| **Baukasten** | **Funktion** |
|---|---|
| Materialflusspunkt | Dient zur Beschreibung des Materialflusses und dessen Eigenschaften an dem gegebenen Punkt. Ein Materialflusspunkt soll dabei die Handhabungsfunktionen (siehe 2.2) umsetzen können. Im Besonderen sind dabei auch Position und Richtung zu berücksichtigen, wie von der VDI-Richtlinie 2860 [Ver90] vorgegeben. |

| **Baukasten** | **Funktion** |
|---|---|
| Raumbegrenzung | Beschreibt die zu Verfügung stehende Fläche und deren Form. Nur innerhalb einer Raumbegrenzung darf der Materialfluss stattfinden. |
| Hindernis | Ein verallgemeinerter Begriff für alle Dinge, durch die kein Materialfluss erfolgen kann. |
| Maschine | Eine Maschine ist ein Hindernis, durch welches jedoch auch in speziellen Fällen Materialfluss verlaufen kann, z.B. wenn ein Förderband durch diese verläuft. Eine Maschine soll dabei nicht zwingend ein eigener Baukasten sein, sondern kann möglicherweise auch durch Hindernisse dargestellt werden. |
| Transfersystem | Wurde der Materialfluss berechnet, so sind Varianten mit den entsprechenden Transfersystemen zu erstellen. Instanzen von Transfersystemen werden nicht zur Modellierung der Umgebung verwendet, sondern sind in den Varianten als Teil des Ergebnisses enthalten. |

**Tabelle 3.1:** Übersicht der Baukästen und deren Funktion

## 3.2 Anforderungen an die Darstellung des Editors und der Varianten

Der Editor zur Erstellung des Materialflusses und deren Umgebung wird folgend als *Materialflusseditor* bezeichnet, die Darstellung einer Variante als *Variantendiagramm*. Bezieht sich eine Aussage auf beide grafische Darstellungen, so wird der Begriff *Diagramm* verwendet.

Sowohl der Materialflusseditor als auch das Variantendiagramm sind grafische Darstellungen von Instanzen aus der durch die Struktur vorgegebenen Baukästen. Dabei generiert der Materialflusseditor eine Struktur aus der modellierten Umgebung und ein Variantendiagramm zeigt die berechnete Struktur einer Variante an.

Die Diagramme sollen weiterhin folgende Eigenschaften aufweisen.

- **Abbildung der Instanzen:** Jede erstellte beziehungsweise berechnete Instanz der Struktur soll eine grafische Darstellung besitzen. Jedes Element des Diagramms soll gleichfalls eine Entsprechung innerhalb der Struktur besitzen. Dies kann eine eigene Instanz oder abgebildete Daten einer Instanz sein.
- **Anzeige der Eigenschaften der Instanzen:** Zu jeder Instanz soll eine grafische Benutzeroberfläche existieren, welche deren Daten anzeigt. Die Anzeige von diesen soll direkt aus dem Diagramm erfolgen und möglichst übersichtlich gehalten werden. Im Falle des Materialflusseditors soll die Benutzeroberfläche auch Änderungen der Daten über die Benutzeroberfläche ermöglichen.
- **Einfache Darstellung:** Die Diagramme sollen möglichst einfach und intuitiv verständlich sein. Es soll kein tiefgehendes Wissen zur Erstellung und Analyse der Diagramme notwendig sein. Dies schließt nicht notwendigerweise die fachspezifischen Daten ein, welche die Instanzen beinhalten.
- **Erstellung und Bearbeitung der Instanzen:** Der Materialflusseditor soll neue Instanzen erstellen und vorhandene Instanzen bearbeiten können. Für das Variantendiagramm ist eine reine Anzeige ausreichend.

## 3.3 Anforderungen an den Algorithmus zur Berechnung der Varianten

Aus der modellierten Umgebung im Materialflusseditor werden mehrere Varianten berechnet. Eine zentrale Aufgabe ist es hierbei, einen passenden Algorithmus zu finden, welcher die möglichen Verbindungen der Materialflusspunkte berechnet und diese anhand der für die Variante zu Verfügung stehenden Transfersysteme umsetzt.

Bei der Erstellung des Algorithmus sind im Besonderen die Eigenschaften der Transfersysteme und deren Zusammenspiel zu beachten.

Nicht alle der ermittelten Anforderungen werden in der Arbeit umgesetzt. Diese werden dennoch aus Gründen der Vollständigkeit nachfolgend erläutert.

- **Vermeidung von Kurven:** Der Materialfluss erfolgt im günstigsten Falle über eine direkte Verbindung der zu verbindenden Materialflusspunkte. Es wird davon ausgegangen, dass gerade Verbindungen generell zu bevorzugen sind und Kurven nur für eine Richtungsänderung des Materialflusses verwendet werden sollen. Dies ist eine getroffene Annahme, da sich Szenarien konstruieren lassen, in denen Kurven theoretisch zu bevorzugen sind. Diese treten in der Realität jedoch nur selten auf.

- **Einschränkung der Winkel:** Transfersysteme sind meist zur Umsetzung definierter Winkel konstruiert und können oft nur festgelegte Winkel umsetzen. Daher wird der Pfad auf exakt die Winkel der Transfersysteme eingeschränkt, welche für die aktuell zu berechnende Variante zu Verfügung stehen. Dies kann dazu führen, dass sich die berechneten Pfade der Varianten einer Industrieanlage maßgeblich unterscheiden. Als Annahme wird davon ausgegangen, dass immer ein gerades Transfersystem vorhanden ist, also eine Winkeländerungen von o Grad umgesetzt werden kann.
- **Berücksichtigung der Breite der Transfersysteme:** Transfersysteme setzen den Materialfluss nicht in einer idealisierten Linie um, sondern besitzen eine bestimmte Breite. Vereinfachend wird angenommen, dass alle Transfersysteme einer Komponentengruppe die gleiche Breite besitzen.
- **Berücksichtigung der Länge der Transfersysteme:** Insbesondere gerade Transfersysteme können heutzutage in ihrer Länge variiert werden. Jedoch haben auch diese eine minimale und eine maximale Länge. Daher wird die minimale Länge bei der Berechnung ebenfalls berücksichtigt. Da Kurven vermieden und somit auch mit einer geringen Länge von diesen gerechnet werden kann, wird angenommen, dass die maximale Länge nur bei geraden Transfersystemen überschritten wird. Diese lassen sich jedoch ohne großen Aufwand durch mehrere gleichartige Transfersysteme umsetzen, weshalb die maximale Länge vereinfachend ignoriert wird.
- **Einhaltung von Abständen:** Um den Zugang für die Wartung und Bedienung zu erleichtern, ist es sinnvoll, dass nicht miteinander verbundene Transfersysteme einen frei wählbaren Abstand zu anderen Transfersystemen und Hindernissen einhalten.
- **Vermeidung von Überschneidungen:** Es existieren Transfersysteme, welche Überschneidungen von Materialflüssen umsetzen können. Weiterhin kann in einer dreidimensionalen Umgebung der Materialfluss auch in mehreren Ebenen übereinander erfolgen. Somit können tatsächliche Überschneidungen vermieden werden. Da ersteres jedoch meist nicht sinnvoll ist und zudem eine Beschränkung auf zwei Dimensionen vorliegt, werden Überschneidungen verschiedener Materialflüsse ausgeschlossen.
- **Berücksichtigung von Umläufen:** Einige Transfersysteme, wie z.B. Plattenförderer, benötigen ein geschlossenes System. Das heißt, alle Transfersysteme müssen in einer geschlossenen Kette miteinander verbunden sein. Sind solche Transfersysteme vorhanden und es liegt kein geschlossenes System vor, so wird dieses erzeugt.

Neben diesen Anforderungen existieren noch allgemeine Anforderungen, welche sich für die Variantenbildung ergeben.

- **Ermittlung kurzer Pfade:** Es soll nicht nur ein beliebiger, sondern ein möglichst kurzer Pfad generiert werden. Daher werden mehrere Pfade ermittelt, von denen der kürzeste gewählt wird. Der Algorithmus ermittelt den kürzesten Pfad unter Berücksichtigung der Vorgaben der Transfersysteme. Aufgrund der hohen Komplexität werden geringe Abweichungen in Sonderfällen jedoch toleriert.
- **Umgehung nicht geeigneter Lücken:** Kann zwar der kürzeste Pfad zwischen zwei Punkten gefunden werden, aber dieser kann aufgrund der Einschränkungen nicht umgesetzt werden, so wird dieser verworfen und der Pfad an einer geeigneten Stelle fortgesetzt. Dies tritt zum Beispiel auf, wenn eine Lücke zwischen Hindernissen zu schmal ist oder einen nicht geeigneten Winkel aufweist. Da einige Ansätze dies nur mit großem Aufwand umsetzen können, wird dies vor allem bei der Auswahl eines Grundansatzes des Wegfindungsalgorithmus berücksichtigt.

## 3.4 Ausgewählte Anforderungen

Um den Rahmen der Diplomarbeit einzuhalten, werden einige der Anforderungen ausgewählt und somit die zu leistende Arbeit eingeschränkt. Weiterhin werden einige vereinfachende Einschränkungen für Struktur und Algorithmus festgelegt, welche für den zu erstellenden Prototypen gelten sollen.

Die wichtigsten allgemeinen Einschränkungen der Aufgabe sind:

- **Beschränkung auf 2D:** Alle Verfahren und Strukturen sind auf zwei Dimensionen eingeschränkt. Eine Erweiterbarkeit auf 3D soll jedoch möglich sein.
- **Einschränkung der Geometrie auf Rechtecke:** Alle komplexen Elemente werden als Rechtecke dargestellt. Diese sollen auch als Grundlage für den Algorithmus dienen. Die Möglichkeit der späteren Erweiterung auf beliebige Polygone soll jedoch gegeben sein.

Die Struktur erfüllt unter diesen Einschränkungen alle ermittelten Anforderungen (siehe 3.1) Ebenso werden die Anforderungen für den Editor anhand einer einfachen grafischen Darstellung umgesetzt (siehe 3.2).

Der Algorithmus zur Berechnung der Varianten wird mit den grundlegenden Anforderungen berücksichtigt. Folgende der ermittelten Anforderungen werden hierzu umgesetzt (siehe 3.3):

- **Vermeidung von Kurven**
- **Einschränkung der Winkel**
- **Berücksichtigung der Breite der Transfersysteme**
- **Einhaltung von Abständen**
- **Vermeidung von Überschneidungen**
- **Ermittlung kurzer Pfade**
- **Umgehung nicht geeigneter Lücken**

# 4 Konzeptfindung zur Darstellung des Materialflusses

Die Struktur wird mit Blick auf den Materialflusseditor und den Algorithmus zur Erstellung der Varianten entworfen. Die wichtigsten Elemente sind dabei die Materialflusspunkte, da über diese die Teilfunktionen der Handhabungsfunktion (siehe 2.2) abgebildet werden. Daher wird zunächst analysiert, wie diese abzubilden sind, bevor die weitergehende Struktur entworfen wird. Dazu werden Möglichkeiten beschrieben, wie die Orientierung des Materials und die generelle Darstellung der Teilfunktionen berücksichtigt werden können.

Weiterhin werden Darstellungsmöglichkeiten für Maschinen untersucht, da unter Umständen der Materialfluss auch durch diese verlaufen kann.

## 4.1 Berücksichtigung der räumlichen Anordnung des Materials

Für die Handhabungsfunktion ist die räumliche Anordnung des Materials über Orientierung und Position definiert (siehe 2.2). Diese soll auch an jedem Materialflusspunkt bestimmbar sein. Die Position kann dabei direkt über die Position des Materialflusspunktes abgebildet werden und muss daher nicht weiter bestimmt werden.

Ursprünglich war vorgegeben, dass zur Umsetzung der Orientierung eine Richtung bei jedem Materialflusspunkt angegeben wird. Diese wurde vom Modellierer erstellt und auch grafisch dargestellt. Daher wird nachfolgend untersucht, ob eine solche Angabe bereits bei der Modellierung sinnvoll ist.

Materialflusspunkte werden miteinander verbunden, um deren Abfolge darzustellen. Während der Berechnung der Varianten werden die Winkel der Transfersysteme zueinander ermittelt. Daher kann die Orientierung des Materials auch berechnet werden und die Voraussetzung der räumlichen Anordnung ist somit ebenfalls erfüllt.

Ebenso ist die Interpretation einer angegeben Richtung von grundlegender Bedeutung. So kann damit entweder eine eingehende, eine ausgehende oder eine punktuelle Richtung gemeint sein. Also ob der Materialfluss direkt vor, direkt nach oder genau an diesem Punkt diese Richtung besitzen soll.

Intuitiv wird hierbei meist zuerst die punktuelle Richtung verstanden. Da Transfersysteme Kurven kontinuierlich umsetzen, müsste bei einer solchen Angabe jeweils der Materialflusspunkt an der Position erstellt werden, an dem diese Kurve exakt den gewünschten Winkel einnimmt. Da dies abseits von sehr einfachen Fällen

mit einem großen Aufwand verbunden ist und auch keine genauen Kurven bei der Modellierung vorgegeben sind (diese werden später berechnet), so ist diese Interpretation ohne weitere Analyse als nicht geeignet zu verwerfen.

Für die beiden anderen Interpretationen existieren einfache Beispiele, welche zu einer Erhöhung der Kurven und Verlängerung der Wege führen:

Wird die Richtung als *eingehend* aufgefasst und liegt der Nachfolger eines Punktes exakt in der Richtung des Vorgängers und besitzt selbst eine andere Richtung, so kann keine Gerade Verbindung zwischen diesen erstellt werden. Somit sind mindestens zwei statt nur einer Kurve zur Umsetzung des Materialflusses notwendig.

**Abbildung 4.1:** Skizze: Beispiel bei eingehender Interpretation der Richtung

Wird die Richtung als *ausgehend* aufgefasst und hat der Nachfolger einen Winkel von 90 Grad oder mehr in Bezug auf die Richtung des Vorgängers, so sind mindestens zwei statt einer Kurve zur Änderung notwendig.

**Abbildung 4.2:** Skizze: Beispiel bei ausgehender Interpretation der Richtung

Da es wahrscheinlich ist, dass ein Modellierer häufig die Materialflusspunkte auf einer Linie (z.B. parallel zu den Koordinatenachsen) erstellt, wird angenommen, dass solche Verlängerungen in der Praxis recht häufig vorkommen werden.

Die Vorgabe einer Richtung an einem Materialflusspunkt hat dabei im Vergleich ohne diese folgende Vor- und Nachteile, welche nicht alle auf den ersten Blick ersichtlich sind:

**Vorteile**

- **Festgelegte räumliche Anordnung bei Modellierung:** Bereits bei der Modellierung wird die räumliche Anordnung der Punkte bestimmt. Somit ist die grundlegende Eigenschaft der Handhabungsfunktion immer vorhanden. Besitzt ein Materialflusspunkt keine Verbindungen, kann dort andernfalls auch keine Orientierung berechnet werden.
- **Genauere Vorgabe der Richtung:** Manchmal ist es sinnvoll, bei einem bestimmten Abschnitt des Materialflusses die Richtung vorgeben zu können, z.B. wenn eine Maschine auf ein gerades Transfersystem angewiesen ist. Dies könnte mit der Angabe der Richtung schnell modelliert werden. Eine solche Vorgabe ohne die Möglichkeit der Richtungsangabe kann zwar angenähert, jedoch nicht sichergestellt werden.

**Nachteile**

- **Erhöhung des Modellierungs-Aufwandes:** Das Erstellen des Materialflusses im Editor wird erschwert, da nicht nur Materialflusspunkte und deren Verbindungen, sondern auch die jeweiligen Richtungen angepasst werden müssen. Dabei können schon geringe ungewollte Abweichungen der frei wählbaren Richtungen zu unerwünschten Ergebnissen führen.
- **Einschränkung der Umsetzbarkeit:** Die Komponentengruppen enthalten eine eingeschränkte Auswahl von Kurven mit fest vorgegebenen Winkeln. Durch die Vorgabe der Richtungen wird die Ausrichtung der Transfersysteme an diesem Punkt zwingend vorgeschrieben. Es ist möglich, dass kein Materialfluss erzeugt werden kann, welcher diese Vorgabe erfüllt.
- **Komplexere Berechnungen:** Die angegebenen Richtungen müssen bei der Berechnung berücksichtigt werden und der Code wird somit komplizierter. Auch weitere Sonderfälle an den Materialflusspunkten sind zu erwarten.
- **Längere Wege:** Wie zuvor erläutert, können einige Konstellationen zu längeren Wegen mit mehr Kurven führen.

Eine mögliche Undefiniertheit der räumlichen Anordnung kann hingenommen werden, da diese nur bei der wenig sinnvollen Angabe einzelner Materialflusspunkte ohne Verbindungen auftritt. Daher ist der einzige Vorteil der Richtungsangabe die Möglichkeit der Angabe einer genauen Richtung auf Teilabschnitten. Dies widerspricht teilweise dem Ansatz der undefinierten Pfade, da nur eine Umsetzbarkeit mit

bestimmten Transfersystemen beziehungsweise bestimmter Komponentengruppen gegeben ist. Unter Berücksichtigung der weiteren Nachteile einer Richtungsangabe bei der Modellierung wird diese daher nicht verwendet.

## 4.2 Umsetzung der Handhabungsfunktionen

Transfersysteme dienen grundsätzlich zum Befördern von Material. Sie können jedoch auch weitere Fähigkeiten aufweisen, welche zum Beispiel die Verarbeitung des Materials erleichtern oder der Messung des Materialstroms dienen.

Eine allgemeine Beschreibung dieser Fähigkeiten wird durch die Teilfunktionen der Handhabungsfunktion (siehe 2.2) beschrieben, welche bei der Modellierung ebenfalls berücksichtigt werden. Daher wird nachfolgend untersucht, ob und wie sich die Teilfunktionen auf die Modellierung der Materialflusspunkte auswirken und wie diese umgesetzt werden können.

| **Teilfunktion** | **Auswirkung und Umsetzung** |
|---|---|
| Mengen verändern | Das Trennen und Vereinigen des Materialflusses kann nur anhand spezieller Transfersysteme erfolgen. Sind solche nicht in der Komponentengruppe enthalten, so kann diese Teilfunktion auch nicht umgesetzt werden. Diese Fähigkeit kann dadurch dargestellt werden, dass mehrere Materialflüsse aus einem Punkt entspringen (Trennen) oder zu diesem führen (Vereinigen). Daher können Materialflusspunkte mehrere Vorgänger und Nachfolger besitzen. |
| Bewegen | Das Drehen und Verschieben sind grundlegende Eigenschaften der Transfersysteme. Diese werden über deren umgesetzte Strecke und den Winkel komplett beschrieben. Die Angabe der Materialflusspunkte gibt dabei deren Ortsveränderung an (Verschieben). Die tatsächliche Drehung wird erst mit den Transfersystemen beziehungsweise bei deren Berechnung angegeben. Eine Modellierung von dieser ist nicht gewollt (siehe auch 4.1). Daher sind auch keine weiteren Vorgaben zu deren Umsetzung nötig. |

| Teilfunktion | Auswirkung und Umsetzung |
|---|---|
| Sichern | Das Halten und Lösen von Material ist eine Eigenschaft der Transfersysteme und schränkt deren Auswahl ein. Diese Fähigkeit kann entweder direkt bei dem Transfersystem angegeben sein oder auch innerhalb einer Komponente, um die das Transfersystem optional ergänzt werden kann. Auf den Materialfluss hat dies jedoch keine Auswirkungen. Daher wird als Eigenschaft eines Materialflusspunktes lediglich angegeben, ob an dem gegebenen Punkt eine Möglichkeit zum Sichern vorliegt oder nicht. |
| Kontrollieren | Ähnlich wie das Sichern ist auch das Kontrollieren eine Eigenschaft eines Transfersystems. Diese müssen jedoch zu deren Umsetzung immer mit einer anderen Komponente, einem Sensor, versehen werden. Dies stellt somit ebenfalls eine Eigenschaft eines Materialflusses dar, welche die Transfersysteme jedoch nicht einschränkt, sondern nur angibt, dass diese an dem Punkt um einen Sensor erweitert werden sollen. |

**Tabelle 4.1:** Umsetzung der Handhabungs-Teilfunktionen

Bei den Eigenschaften der Materialflusspunkte wird somit folgendes berücksichtigt: Es sind mehrere Vorgänger und Nachfolger erlaubt und die Fähigkeiten *Sichern* und *Kontrollieren* müssen ausgewählt werden können.

## 4.3 Darstellung des Materialflusses durch Maschinen

Maschinen zur Verarbeitung des Materials können einen Materialfluss enthalten. Das bedeutet, durch diese verläuft ein Transfersystem, welches Material in die Maschine befördert. Zur Umsetzung einer solchen Option wurde vorgeschlagen, Maschinen durch mehrere Hindernisse zu modellieren, welche eine entsprechende Lücke lassen, durch die der Materialfluss verlaufen kann. Alternativ soll ein eigener Baukasten für Maschinen erstellt werden, in denen ein Materialfluss stattfinden kann, sofern dieser zur Belieferung der Maschine dient. Dies grenzt Maschinen von Hindernissen ab, durch welche niemals ein Materialfluss erfolgen darf.

Beide Möglichkeiten werden untersucht und miteinander verglichen.

Abbildung 4.3 zeigt die verschiedenen Möglichkeiten der Modellierung von Maschinen mitsamt eines durch diese verlaufenden Materialflusspfades.

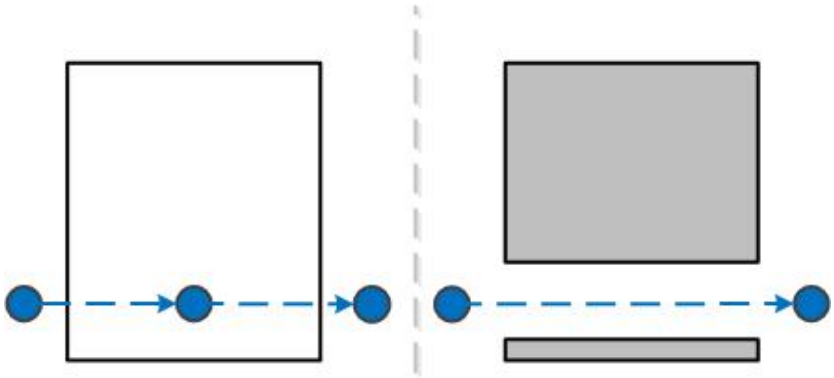


**Abbildung 4.3:** Modellierung einer Maschine als eigenes Element und über Hindernisse

Die Erstellung eigener Baukästen für Maschinen bietet folgende Vor- und Nachteile:

**Vorteile**

- **Abgrenzung zu Hindernissen:** Maschinen haben im Gegensatz zu Hindernissen Funktionen, welche diese umsetzen. Bei einer Modellierung durch Hindernisse könnte nur schwer dargestellt werden, ob ein Hindernis eine Maschine ist und somit auch eine Funktion besitzt oder diese ausschließlich den Materialfluss einschränkt.
- **Einfache Modellierung:** Bei der Modellierung durch Hindernisse müssen Lücken entworfen werden, durch die der Materialfluss erfolgen soll. Dabei sind die Winkel und auch die Abstände der Hindernisse zu beachten, was die Modellierung aufwändiger macht. Dies entfällt komplett, wenn ein eigenes Element zur Darstellung von Maschinen vorhanden ist. Auch ein komplexer Materialfluss ist durch die Modellierung innerhalb einer Maschine möglich.
- **Direkte Zuordnung des Materialflusses:** Verläuft ein Materialfluss durch eine Maschine, so werden in dieser Materialflusspunkte erstellt. Somit enthält die Maschine entsprechende Punkte als Tochterelemente. Diese können daher direkt der Maschine zugeordnet werden.
- **Sicherstellung des gewünschten Materialflusses:** Lücken zwischen Hindernissen können umgangen werden, wenn diese einen ungünstigen Weg für den Materialfluss bedeuten. Durch die Angabe von Materialflusspunkten wird der

Materialfluss jedoch unter allen Umständen durch die Maschine verlaufen, sofern dies modelliert wurde.

- **Höhere Umsetzbarkeit:** Verläuft der Materialfluss zwischen Hindernissen und die Materialflusspunkte können durch die gegebenen Transfersysteme oder deren Anordnung nicht erreicht werden, so kann kein Ergebnis zur Lösung des Materialflusses ermittelt werden. Das Auftreten eines solchen Falles wird bei einer Modellierung der Maschinen als Hindernisse wesentlich wahrscheinlicher.

**Nachteile**

- **Keine genaue Vorgabe des Materialflusses:** Es kann nicht sicher gestellt werden, dass der Materialfluss innerhalb der Maschine einen definierten Verlauf nimmt, zum Beispiel durch einen geraden Verlauf am Rand der Maschine. Dieser ist nämlich weiterhin von den vorhandenen Transfersystemen und deren Anordnung abhängig. Bei der Modellierung über Hindernisse würde dies jedoch gleichfalls zu Problemen führen, da der gewünschte Verlauf wahrscheinlich nicht umgesetzt werden kann.

Die Darstellung von Maschinen als ein eigenes Element bietet überwiegend Vorteile. Daher wird ein eigener Baukasten für Maschinen erstellt. In diesen dürfen ebenfalls Materialflusspunkte erzeugt werden. Ein Materialfluss außerhalb der Maschine darf nur durch diese verlaufen, wenn er direkt mit einem ein- oder ausgehenden Materialflusspunkt der Maschine verbunden ist.

# 5 Datenstruktur der Baukästen

Im EPLAN Engineering Center (siehe 2.3.1) wurde eine Bibliothek angelegt, welche die nachfolgend beschriebenen Baukästen enthält. Die Bibliothek trägt den Namen KomponentenneutraleBeschreibung. Die Baukästen dienen zur Speicherung der Daten und der Abhängigkeiten der in den Diagrammen dargestellten Elemente.

## 5.1 Übersicht der Baukästen

Die Bibliothek enthält 4 Unterbereiche, welche die Baukästen in Gruppen aufteilt:

- **Gruppierungen**
- **Hindernisse**
- **Materialfluss**
- **Transfersysteme**

Zusätzlich ist der Bereich **Diagrammkonfigurationen** vorhanden, welcher keine Baukästen enthält, sondern zur Beschreibung des Materialflusseditors und der grafischen Darstellung der Varianten dient. Diese Unterbereiche enthalten jeweils verschiedene Baukästen, welche in den folgenden Unterkapiteln genauer erläutert werden.

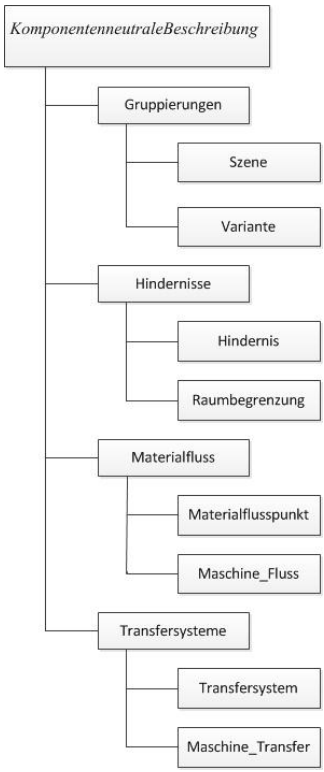



**Abbildung 5.1:** Übersicht der Unterbereiche und der enthaltenen Baukästen

## 5.2 Gruppierungen

Gruppierungen dienen ausschließlich zur Erzeugung einer Hierarchie. Sie weisen neben dem Namen keine weiteren Eigenschaften auf.

### Szene

Szenen werden zur strukturierten Anordnung der Diagramme und der dort erstellten Instanzen erzeugt. Eine Szene enthält einen Materialflusseditor, über den die weiteren Instanzen automatisch angelegt werden. Weiterhin werden die erzeugten Raumbegrenzungen unterhalb dieser abgelegt. Die generierten Varianten werden ebenfalls hierarchisch unterhalb dieser Szene erzeugt.

### Variante

Eine Variante dient zur hierarchischen Ordnung der berechneten Varianten. Unterhalb dieser Struktur werden die Instanzen für jeweils eine Variante sowie ein Variantendiagramm zur grafischen Veranschaulichung abgelegt.

## 5.3 Hindernisse

Hindernisse werden sowohl im Materialflusseditor als auch in den Varianten verwendet. Sie stellen Bereiche dar, die andere Elemente enthalten dürfen (Raumbegrenzung) oder nicht enthalten dürfen (Hindernis). Sie schränken somit den Materialfluss ein.

### Hindernis

Hindernisse sind ein allgemeiner Begriff für Dinge, welche den Materialfluss blockieren. Das heißt, durch Hindernisse kann kein Materialfluss stattfinden. Hindernisse werden bei der Modellierung erstellt und bei der Generierung der Varianten berücksichtigt.

Hindernisse werden zunächst auf Rechtecke beschränkt und grau unterlegt dargestellt.

Ein Hindernis hat folgende Eigenschaften:

- Geometrie (Rechteck)
- Position (Pos_X, Pos_Y)
- Raumwinkel

### Raumbegrenzung

Eine Raumbegrenzung stellt die Umgebung der Elemente dar. Sie wird im Materialflusseditor erstellt, und enthält alle anderen Elemente zur Darstellung des Materialflusses. Andere Elemente dürfen daher nur innerhalb einer Raumbegrenzung erzeugt werden. Bei der Erzeugung der Varianten darf der Materialfluss eine Raumbegrenzung nicht verlassen.

Sie ist zunächst ebenfalls auf ein Rechteck eingeschränkt und wird durch eine gestrichelte Linie dargestellt.

Eine Raumbegrenzung hat folgende Eigenschaften:

- Geometrie (Rechteck)
- Position (Pos_X, Pos_Y)
- Raumwinkel

## 5.4 Materialfluss

Diese Elemente werden zur Modellierung des Materialflusses verwendet und sind in den Varianten nicht mehr enthalten.

### Materialflusspunkt

Ein Materialflusspunkt stellt die Basis für die spätere Berechnung der Varianten dar. Anhand der Materialflusspunkte werden die Transfersysteme berechnet. Sie dürfen nur in Raumbegrenzungen und Maschinen erzeugt werden.

Die Materialflusspunkte besitzen gerichtete Verbindungen untereinander. Diese werden in Listen als Vorgänger und Nachfolger gespeichert.

Ein Materialflusspunkt wird als blauer Kreis dargestellt und deren Verbindungen als blauer gestrichelter Pfeil.

Ein Materialflusspunkt hat folgende Eigenschaften:

- Geometrie (Kreis)
- Position (Pos_X, Pos_Y)
- Liste der Vorgänger
- Liste der Nachfolger
- Markierung für Sicherung (Fähigkeit)
- Markierung für Messung (Fähigkeit)

### Maschine_Fluss

Eine Maschine_Fluss steht für eine Maschine, welche im Materialflusseditor verwendet wird. Aus dieser wird später eine Maschine_Transfer erzeugt. Sie stellt eine Kombination aus Materialflusspunkten und einem Hindernis dar. Der Grund der Trennung von Maschinen sind die verschiedenen Unterelemente. Würde nur ein Baukasten für eine Maschine vorliegen, welche sowohl Materialflusspunkte als auch Transfersysteme enthalten kann, wäre es möglich, bereits bei der Modellierung Transfersysteme zu erzeugen. Dies soll aber vermieden werden.

Eine Maschine_Fluss wird durch ein Rechteck dargestellt und kann Materialflusspunkte enthalten. Bei der Berechnung des Materialflusses darf eine Maschine nur passiert werden, wenn eine Materialflussverbindung direkt in eine Maschine hinein oder hinaus läuft. Andernfalls ist diese als Hindernis zu betrachten.

Sie hat folgende Eigenschaften:

- Geometrie (Rechteck)
- Position (Pos_X, Pos_Y)
- Raumwinkel
- Liste von Materialflusspunkten

## 5.5 Transfersysteme

Transfersysteme werden nicht im Materialflusseditor verwendet, sondern aus den unter Materialfluss genannten Elementen bei den Varianten erzeugt.

### Transfersystem

Ein Transfersystem wird zur Umsetzung des vorher modellierten Materialflusses benötigt und stellt zum Beispiel ein Förderband dar.

Es hat folgende Eigenschaften:

- Geometrie
- Position (Pos_X, Pos_Y)
- Raumwinkel
- Länge
- Breite
- Liste der Vorgänger-Transfersysteme
- Liste der Nachfolger-Transfersysteme
- Markierung für Sicherung (Fähigkeit)
- Markierung für Messung (Fähigkeit)

### Maschine_Transfer

Eine Maschine_Transfer entsteht durch die Umsetzung einer Maschine_Fluss. Es ist eine Kombination aus Hindernis und Transfersystemen.

Wie Maschine_Fluss ist die Maschine_Transfer auf Rechtecke eingeschränkt und wird als ein solches dargestellt.

- Geometrie (Rechteck)
- Position (Pos_X, Pos_Y)
- Raumwinkel
- Liste von Transfersystemen

## 5.6 Diagrammkonfigurationen

Dieser Bereich dient zur Definition der verschiedenen Diagramme. Deren Aufbau wird genauer im Kapitel 6 beschrieben.

### Materialflusseditor

Für eine Szene werden über den Materialflusseditor die Instanzen erzeugt, welche zur Modellierung des Materialflusses und dessen Umgebung notwendig sind.

Der Materialflusseditor verwendet die Baukästen *Hindernis*, *Maschine_Fluss*, *Raumbegrenzung* und *Materialflusspunkt*.

### Variantendiagramm

Der Aufbau des Diagramms zur Darstellung der verschiedenen Varianten wird in diesem Baukasten beschrieben. Es werden automatisch ein oder mehrere Variantendiagramme aus dem Materialflusseditor erstellt, sobald die Verarbeitung angestoßen wird. Eine Modellierung in diesen Diagrammen ist nicht vorgesehen.

Das Variantendiagramm verwendet die Baukästen *Hindernis*, *Maschine_Transfer*, *Raumbegrenzung* und *Transfersystem*.

# 6 Erstellung des Materialflusseditors

Das *EPLAN Engineering Center* stellt einen Baukasten zu Verfügung, über welchen anhand eines definierten XML-basierten Schemas ein Diagramm zur grafischen Darstellung und Erzeugung von Instanzen anderer Baukästen beschrieben werden kann. Aus dieser Beschreibung wird automatisch ein Diagramm erzeugt, sobald eine Instanz von diesem angelegt wird. In dem Projekt wurden die Baukästen des Materialflusseditors und des Variantendiagramms mittels des genannten Schemas erstellt.

Das Vorgehen zur Erstellung eines Diagrammes ist für einige der Kooperationspartner von Interesse, daher wird dieses folgend teils sehr detailliert beschrieben. Für ein allgemeines Verständnis dieser Arbeit ist jedoch lediglich das Unterkapitel *Grafische Darstellung des Materialflusseditors* (6.1) erforderlich. Die weiteren Unterkapitel können somit bei Bedarf übersprungen werden.

## 6.1 Grafische Darstellung des Materialflusseditors

Der Materialflusseditor enthält eine Werkzeugleiste zur Erstellung verschiedener Instanzen. Diese Instanzen werden im Editor angezeigt und deren Eigenschaften können über grafische Oberflächen bearbeitet werden. Die Instanzen werden dabei automatisch in einer hierarchischen Struktur abgelegt. Abbildung 6.1 zeigt eine im Materialflusseditor modellierte Szene zur Beschreibung des Materialflusses und dessen Umgebung.

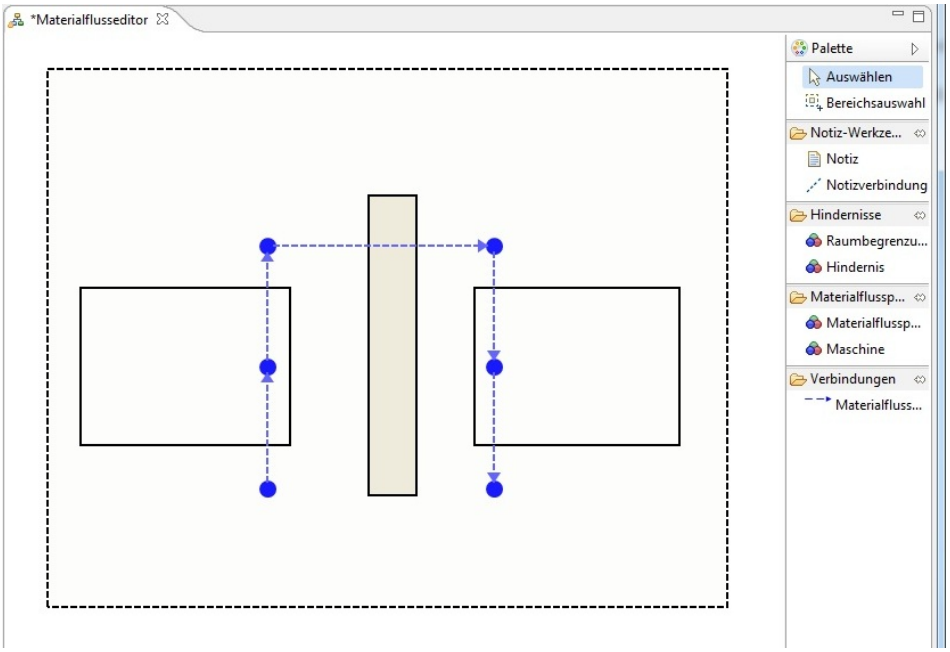



**Abbildung 6.1:** Darstellung des Materialflusseditors mit modellierten Beispiel

## 6.2 Initialisierung und Nutzung der modellierten Komponenten

Zur Definition eines Diagrammes anhand des vorhandenen XML-Schemas müssen zunächst allgemeine Vorgaben des Diagramms angegeben werden:

```
<diagramEditor
       id="diagram"
       title="Materialflusseditor"
       packageable="=isInstanceOf('Komponentenneutral.Szene.Szene')"
       version="1"
       acceptedChildren="Raumbegrenzung"
>
```

Der *title* gibt hierbei an, welchen Namen Instanzen des Editors standardmäßig tragen. Die Eigenschaft *packageable* enthält die Komponenten, unter denen der Editor erstellt werden kann. So ist es hier möglich, eine Instanz des Materialflusseditors unterhalb einer Instanz des Baukastens Szene zu erstellen.

Unter der Eigenschaft *acceptedChildren* wird aufgelistet, welche Instanzen der Editor auf der obersten Hierarchie-Ebene enthalten darf. Hierbei wird nicht auf vorhandene Baukästen referenziert, sondern auf einen Knoten (beziehungsweise *node*), welcher in der XML-Datei aufgeführt wird.

```
<node id="Raumbegrenzung" valid="= isInstanceEO() and
       isInstanceOf('Komponentenneutral.Hindernisse.Raumbegrenzung')"
       acceptedChildren="AbstraktesElement">
       <figurePolygon
          figure="com.mind8.graph2d.figure.container.Polygon">
              <configuration>
                     <editRequestHandler
                        xsi:type="formui:formui-handler" />
                     <editRequestHandlerInfo
                        xsi:type="formui:formui-handler-info"
                        refId="Raum_UI" />
              </configuration>
              <properties>
              ...
              </properties>
       </figurePolygon>
</node>
```

Die *id* des Knotens enthält den Namen, welcher innerhalb der XML-Datei verwendet wird. In *valid* wird gekennzeichnet, dass diesem Knoten eine Instanz (*isInstanceEO()*) zugeordnet wird und diese vom Typ Raumbegrenzung sein soll. Mögliche Unterelemente für diesen Knoten müssen vom Typ *AbstraktesElement* sein. Dieses wird nur innerhalb der XML-Datei verwendet und dient als Elternklasse für die Komponenten Maschine_Fluss, Materialflusspunkt und Hindernis. Dies ist notwendig, da bei der Angabe der *acceptedChildren* derzeit nur ein Knotentyp angegeben werden kann.

Bei der Angabe der *acceptedChildren* ist weiterhin zu beachten, dass die Ebenenhierarchie der modellierten Baukästen ebenfalls eine solche Hierarchie zulassen muss. Dort muss angegeben werden, dass eine Raumbegrenzung Instanzen der Baukästen Maschine_Fluss, Materialflusspunkt und Hindernis enthalten darf.

[Cas12]

## 6.3 Palettenerstellung und Erzeugung der Instanzen

Die bisher genannten Einträge beschreiben die zu verwendenden Elemente, deren zugewiesenen Baukästen und der Hierarchie zwischen diesen. Damit diese Instanzen auch innerhalb des Editors erzeugt werden können, sind weitere Einträge notwendig.

Eine Palette mit Standard-Werkzeugen wird automatisch von dem Baukasten zur Erstellung eines Diagramms generiert. Diese Palette wird entsprechend erweitert, so dass Instanzen der Baukästen erstellt werden können. Hierzu dient der Eintrag *paletteEntry*:

```
<paletteEntry id="RaumbegrenzungPalette" group="Hindernisse
   text="Raumbegrenzung"
   smallIcon="=type('Komponentenneutral.Hindernisse.Raumbegrenzung').
   getImage" >
       <eoCreation command="commandCreateRaumbegrenzungFromPalette"
/>
```

Die *id* beschreibt wieder einen internen Namen, welcher nur innerhalb der XML-Datei verwendet wird. Durch die Angabe einer *group* wird automatisch eine Gruppe in der Palette mit entsprechenden Namen angelegt. Alle Einträge mit der gleichen *group* werden hierbei unter einer Gruppierung angezeigt. Der *text* gibt den Namen des Eintrags an und die *smallIcons* deren grafische Darstellung innerhalb der Palette. Die Icons werden den in den Baukästen hinterlegten Bildern entnommen. Mit *eoCreation* wird ein *command* angegeben, welches aufgerufen wird, sobald im Editor dieser Eintrag ausgewählt wird. Diese *commands* sind ebenfalls in der XML-Datei

anzugeben.

```
<command xsi:type="instantiateAndCreateNode" src="=isClassEO() and
   isAssignableTo('Komponentenneutral.Hindernisse.Raumbegrenzung')"
   eoClassPath="Komponentenneutral.Hindernisse.Raumbegrenzung"
   id="commandCreateRaumbegrenzungFromPalette" />
```

Einem *command* wird ein entsprechender Baukasten zugewiesen. In diesem Fall wird eine neue Instanz des Baukastens Raumbegrenzung angelegt. Dabei gibt *eoClassPath* den zu instanziierenden Baukasten an. In *src* kann ein Verweis auf Quellcode oder wie hier auch fest programmierte Bedingungen eingefügt werden, um die Instanziierung zu überprüfen oder einzuschränken.

[Cas12]

## 6.4 Verbindungsdefinition der Materialflusspunkte

Um Materialflusspunkte zu verbinden, reicht eine Definition innerhalb der XML-Datei aus. Es müssen daher keine weiteren Instanzen erzeugt werden.

```
<reference id="Materialfluss_Referenz" type="connectable"
   src="=isInstanceEO() and isInstanceOf
   ('Komponentenneutral.Materialfluss.Materialflusspunkt')"
   target="=isInstanceEO() and isInstanceOf
   ('Komponentenneutral.Materialfluss.Materialflusspunkt')">
       <forward multiplicity="*"
          connector="=getParameter('Nachfolger')" />
       <backward multiplicity="*"
          connector="=getParameter('Vorgaenger')" />
</reference>

<connection id="Materialfluss_Pfad" reference="Materialfluss_Referenz"
   router="Bendpoint" width="2" style="2" color="100,100,255"
   sourceDeco="empty" targetDeco="filled-arrow" />
```

Die *reference* beschreibt dabei die Art der Verbindungen und an welcher Stelle diese Verbindung innerhalb des Materialflusspunkts vermerkt wird. Es werden somit beim Erstellen einer Verbindung automatisch Vorgänger und Nachfolger gesetzt.

Der Eintrag *connection* beschreibt die grafische Darstellung der Verbindung innerhalb des Editors. Um eine Verbindung erzeugen zu können, wurde für diese ebenfalls ein

Eintrag in der Palette erstellt.

```
<paletteEntry id="MaterialflussverbindungPalette" group="Verbindungen"
   text="Materialflussverbindung"
   smallIcon="=type('Komponentenneutral.Bilder.Materialflussverbindung').
   getImage"
   largeIcon="=type('Komponentenneutral.Bilder.Materialflussverbindung').
   getImage"
>

       <connectionCreation connection="Materialfluss_Pfad" />
```

Da Verbindungen auf keine Baukästen verweisen, müssen die Icons für die Palette direkt angegeben werden. Im Gegensatz zu den Einträgen der Baukästen ist jedoch keine Angabe und Definition von *commands* notwendig.

[Cas12]

## 6.5 Beschreibung grafischer Benutzeroberflächen

Für jede im Editor erstellte Instanz kann eine grafische Oberfläche zur Anzeige und Änderung der Daten dieses Elements angezeigt werden. Für jeden Baukasten wird daher eine Oberfläche erstellt, welche bei Aufruf mit den entsprechenden Daten der angewählten Instanz gefüllt wird. Die Erstellung der Oberflächen erfolgt wie die Erstellung des Editors anhand eines vorgegebenen XML-Schemas. Dieses Schema enthält Elemente zur Beschreibung einfacher Strukturen zum Aufbau der Oberfläche sowie Elemente zur Formulierung von Schleifen und der Instanziierung von Elementen.

```
<form title="Materialflusspunkt" id="Materialflusspunkt_UI">
<line>
       <label>Name:</label>
       <input type="text" editable="true" text="=name()"
          receiver="this"></input>
</line>
<group title="Position" expanded="true">
</group>
<group title="Vorgaenger" expanded="true">
       <loop variable="x" receiver="=$Vorgaenger.ifNull('')">
       <line visible="x.isInstanceOf
          ('Komponentenneutral.Materialfluss.Materialflusspunkt')" >
```

```
        <label text="x.name" />
      </line>
</loop>
</group>
...
```

Für jeden Baukasten wird eine Oberfläche beschrieben. Dennoch muss im Editor für jeden von diesen noch der Name der Oberfläche eingegeben werden, so dass diese bei Aufruf der erstellten Instanzen aufgerufen werden kann.

[Cas12]

# 7 Algorithmenauswahl

Es wird ein Algorithmus entwickelt, um die Varianten aus dem modellierten Materialfluss und dessen Umgebung zu berechnen. Der modellierte Materialfluss gibt keinen genauen Pfad an, auf dem dieser erfolgen muss, sondern lediglich die Abfolge der Orte, zu welchen das Material transportiert werden soll.

Daher werden verschiedene Ansätze von Algorithmen und Strukturen zur Wegfindung untersucht und miteinander verglichen. Dabei wird insbesondere darauf geachtet, dass die Umsetzung der ermittelten Anforderungen (siehe 3) für die verschiedenen Ansätze möglich ist. Ebenso wird abgeschätzt, wie hoch der Aufwand zu deren Anpassung ist.

Die Anforderungen *Berücksichtigung der Breite der Transfersysteme*, *Einhaltung von Abständen* und *Vermeidung von Überschneidungen* werden hierbei nicht beachtet, da diese mit leichten Anpassungen der Umgebung in jedem der Ansätze umgesetzt werden können (siehe 8).

## 7.1 Wegfindungsalgorithmen

Üblicherweise wird bei der Ermittelung eines Weges von einem Startpunkt zu einem Endpunkt die Umgebung ausgehend von dem Startpunkt untersucht. Ist eine optimale Verbindung, üblicherweise eine direkte Verbindung mit minimal möglichen Aufwand, nicht möglich, so wird die Suche ausgedehnt und weitere Punkte ermittelt, um eine Verbindung zu ermöglichen. Dabei werden verschiedene Pfade untersucht, von denen der Beste gewählt wird. Meist werden Pfade jedoch nicht komplett berechnet, sondern es werden bereits Teilpfade miteinander verglichen. Führen verschiedene Teilpfade zu dem gleichen Punkt, wird nur der beste Teilpfad weiter verfolgt.

### 7.1.1 Rasterisierung des Raumes

Bei der Rasterisierung wird der modellierte Raum in Zellen zerlegt, welche zusammen ein Gitter darstellen. Oft werden hierbei für die Zellen gleichartige Geometrien, wie zum Beispiel Rechtecke, in einem einheitlichen Muster verwendet. Diese können mit schnellen Rechenoperationen durchlaufen werden und benötigen einen geringeren Speicheraufwand. Jedoch ist auch die Zerlegung des Raumes durch komplexere Geometrien und auch die Verwendung unterschiedlicher Geometrien im gleichen Gitter möglich.

Dadurch kann ein Graph aus *Knoten* und *Kanten* erstellt werden, für deren Analyse und Auswertung ausgereifte Algorithmen zu Verfügung stehen. Innerhalb des Netzes

können daher schnell gute Wege gefunden werden. Den Zellen oder Knoten, seltener auch den Kanten, werden Werte zugewiesen, welche die Umgebung anhand von Kosten beschreiben. Zum Beispiel können Hindernisse sehr hohe Kosten aufweisen, während freie Flächen geringe Kosten besitzen. [ES11]

Diese Kosten können auch variabel angepasst werden, um weitere Vorgaben umsetzen zu können. Es ist möglich, Richtungsänderungen bei Netzen mit rechteckigen Zellen mit höheren Kosten zu versehen. Dadurch werden gerade Verbindungen bevorzugt und Kurven vermieden. [Sie09]

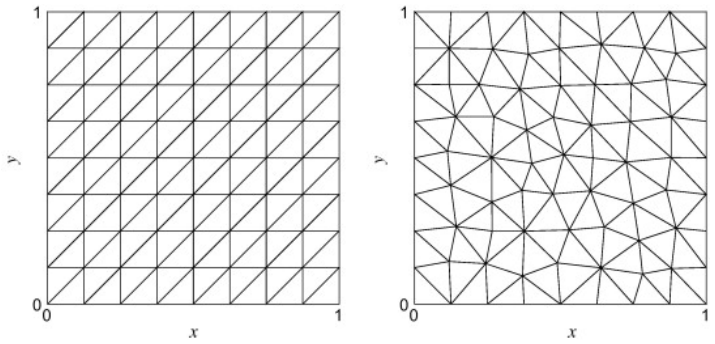



**Abbildung 7.1:** Beispiele typischer Gitter [Hir12]

In Bezug auf die ermittelten Anforderungen werden die Eigenschaften einer Rasterisierung zur Wegfindung wie folgt eingeschätzt:

- **Vermeidung von Kurven:** Ist anhand einer Kostenanpassung ohne weiteres möglich.
- **Einschränkung der Winkel:** Schwer umzusetzen, da das gesamte Netz auf variable Winkel angepasst werden muss und die Berechnung der Geometrien somit sehr komplex werden.
- **Ermittlung kurzer Pfade:** Aufgrund bekannter Algorithmen im Rahmen des erstellten Graphen leicht umsetzbar.
- **Umgehung nicht geeigneter Lücken:** Muss bei der Erstellung des Graphen berücksichtigt werden, was deren Komplexität weiter erhöht und fehleranfälliger macht.

### 7.1.2 A* Algorithmus

Der A*-Algorithmus ist die Basis für viele bekannte Wegfindungsalgorithmen. Im grundlegenden Ansatz des Algorithmus wird der kürzeste Pfad in einem Graphen von einem Knoten zu einem anderen gesucht. Um nicht den ganzen Graphen durchsuchen zu müssen, wird eine Schätzfunktion verwendet.

Ein einfacher Ansatz zur Übertragung eines Graphen auf eine Umgebung ist dabei die Einteilung der Umgebung durch ein Raster. Jede so entstehende Zelle des Rasters bildet dabei einen Knotenpunkt. Eine sehr feine Rasterisierung ist zum Beispiel die Einteilung einer modellierten Umgebung auf einzelne Pixel, aber auch gröbere Einteilungen sind möglich. Setzt man dies um, stellt der A*-Algorithmus eine besondere Form des beschriebenen Ansatzes der *Rasterisierung des Raumes* (7.1.1) dar.

Der A*-Algorithmus verwendet eine heuristische Schätzfunkion H, welche die minimal möglichen Kosten eines Knotens zu einem anderen Knoten angibt. Um eine sinnvolle Schätzung vornehmen zu können, muss ein Zusammenhang zwischen Position und Kosten gefunden werden. Dabei ist es wichtig, dass eine solche Schätzung die minimal möglichen Kosten vorgibt, welche zwischen diesen Knoten bestehen kann. Es ist möglich, dass diese Verbindung mit den angegebenen Kosten niemals tatsächlich erreicht werden kann. Jedoch dürfen diese Kosten in keinem Fall unterschritten werden können. Im Zweifelsfall sind daher möglichst geringe statt zu hohe Kosten innerhalb der Schätzfunktion zu berechnen.

Im Fall einer rasterisierten Umgebung in einem Gitternetz und gleichmäßigen Gewichtungen der Zellen kann der Abstand des aktuellen Knotens (bzw. Zelle) zum Zielknoten als Schätzfunktion verwendet werden, wobei die zurückgelegte Entfernung die Kosten angibt. Die minimal möglichen Kosten sind somit bei einer Verbindung über eine Gerade gegeben – jede andere Verbindung müsste eine größere Strecke zurücklegen würde somit zu höheren Kosten führen.

Jeder bekannte Knoten enthält dabei Kosten, welche sich aus folgender Formel ergeben:

$$\textit{Kosten} = \textit{ermittelte Kosten von Startknoten} + \textit{Abstand zu Zielknoten}$$

Es gibt immer einen aktiven Knoten, von dem aus ein weiterführender Pfad gesucht wird. Zu Beginn ist dies der Startknoten. Ausgehend von diesem wird deren Nachbarschaft durchsucht und der nächste Knoten mit den geringsten Kosten durchsucht, sofern dieser nicht bereits untersucht wurde. Durch die Einbeziehung des Abstandes zum Ziel in die Kosten wird automatisch der kürzest mögliche Pfad zuerst überprüft. [Mato2] [SF04]

Die Eigenschaften des A*-Algorithmus mit einer Rasterisierung anhand der Pixel werden in Bezug auf die Anforderungen nachfolgend bewertet:

- **Vermeidung von Kurven:** Anhand von weiteren Einschränkungen der Suchrichtung bzw. deren Kosten wahrscheinlich umsetzbar.
- **Einschränkung der Winkel:** Auch hier müssen weitere Anpassungen der Suchrichtung vorgenommen werden, welche auch mehrere Schritte überdauern müssen.
- **Ermittlung kurzer Pfade:** Der A*-Algorithmus wird als *optimal* beschrieben, womit immer der kürzeste Weg gefunden wird.
- **Umgehung nicht geeigneter Lücken:** Im grundlegenden Algorithmus nicht möglich, jedoch sind variable Anpassungen der Umgebung denkbar, welche nachträglich der Umgehung dienen können.

### 7.1.3 Navigation Meshes

Navigation Meshes (englisch für *Navigations-Netze*) sind zunächst eine Datenstruktur, welche Wegpunkte enthält. Dabei wird üblicherweise die begehbare Fläche der Umgebung, ähnlich wie beim Rendern von Objekten, durch Dreiecke eingeteilt. Je größer und unverzweigter dabei eine Fläche ist, desto weniger Dreiecke werden für deren Darstellung benötigt. Es können auch andere konvexe Polygone verwendet werden, was deren Bestimmung jedoch aufwändiger macht.

Die Dreiecke enthalten Wegpunkte, welche üblicherweise an deren Kanten oder deren Mittelpunkt angeordnet werden. Anhand der Nachbarschaft der Dreiecke kann überprüft werden, welche der Wegpunkte miteinander verbunden werden können. Werden Start- und Endpunkt ebenfalls einem Dreieck zugeordnet, können beginnend mit den Wegpunkten des Start-Dreiecks die möglichen Pfade ermittelt werden. [Hayo8] [DO12]

Der größte Aufwand ist hierbei, neben der Triangulation des Raumes, das sinnvolle Platzieren der Wegpunkte.

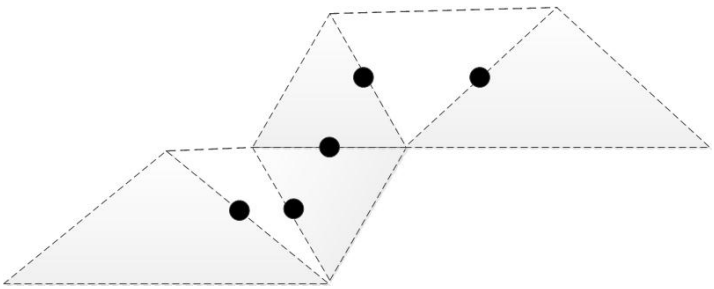


**Abbildung 7.2:** Navigation Mesh mit Wegpunkten an Schnittflächen

Die Eigenschaften von Navigation Meshes in Bezug auf die Anforderungen sind:

- **Vermeidung von Kurven:** Bei geeigneter Anordnung der Wegpunkte leicht möglich.
- **Einschränkung der Winkel:** Schwer umzusetzen, da mehrere Wegpunkte gesetzt werden müssen, welche nur von anderen Wegpunkten gewählt werden dürfen. Widerspricht teilweise dem Punkt *Vermeidung von Kurven*.
- **Ermittlung kurzer Pfade:** Im Rahmen der Wegpunkte sehr leicht möglich. Diese können jedoch von den tatsächlich kürzesten Pfaden abweichen.
- **Umgehung nicht geeigneter Lücken:** Muss bei der Erstellung der Wegpunkte oder des Netzes berücksichtigt werden, sollte aber möglich sein.

### 7.1.4 Geometrische Ansätze

Der geometrische Ansatz ist ein hier verwendeter Sammelbegriff für Algorithmen, welche zur Wegfindung keine neue Struktur aufbauen, sondern direkt anhand der vorhandenen Umgebung versuchen, den Weg zu ermitteln. Dabei wird meist eine direkte Verbindung zwischen Start und Ziel über eine Gerade gesucht. Ist eine solche nicht möglich, werden die gekreuzten Hindernisse umgangen, bis eine direkte Verbindung möglich ist.

Solche Ansätze sind meist nur bei vereinfachten Umgebungen sinnvoll, welche nur aus Hindernissen und freien Flächen bestehen. Eventuelle Mehrkosten für zum Beispiel unterschiedliches Terrain können nur schwer berücksichtigt werden. Weiterhin entfällt zwar ein initialer Aufwand zur Erstellung einer geeigneten Struktur, dafür sind die Berechnungen in jedem Schritt meist wesentlich komplexer und aufwändiger. Dadurch ist ein solcher Ansatz meist nicht für zeitkritische Berechnungen geeignet.

Der Ansatz ist jedoch sehr flexibel und kann mit entsprechendem Aufwand an fast beliebige Anforderungen angepasst werden.

Bezogen auf die Anforderungen hat dieser Ansatz folgende Eigenschaften:

- **Vermeidung von Kurven:** Aufgrund des Vorgehens über Geraden leicht möglich, jedoch müssen zu kurze Geradenabschnitte beachtet werden.
- **Einschränkung der Winkel:** Aufgrund einer Richtungsvorgabe bei der Umgehung von Hindernissen leicht möglich.

- **Ermittlung kurzer Pfade:** Die Pfade könnten bei komplexen Hindernissen leicht verlängert werden, was größere Anpassungen notwendig machen würde. Dennoch sind im Allgemeinen gute und kurze Pfade zu erwarten.
- **Umgehung nicht geeigneter Lücken:** Nicht geeignete Lücken führen zu vielen Schnittpunkten mit den Hindernissen und könnten mit Anpassungen erkannt werden.

## 7.2 Vergleich und Auswahl eines Algorithmus

Die ausgewählten Verfahren wurden in den vorigen Kapiteln beschrieben und grob bewertet. Um einen direkten Vergleich zu ermöglichen, werden die Bewertungen verfeinert. Hierfür wird für jede Anforderung eine Punktzahl zwischen *1* und *10* vergeben, wobei *10* für eine sehr gute Eignung mit voraussichtlich wenig Anpassungen und *1* für eine schlechte Umsetzbarkeit steht.

Es wird davon ausgegangen, dass mit entsprechenden Modifikationen alle Ansätze jede Anforderung umsetzen können. Daher bedeutet auch eine niedrige Punktzahl nicht zwingend, dass eine Umsetzung unmöglich ist. Weiterhin sind die Punkte als Abschätzungen der Umsetzbarkeit gemessen an derem Aufwand zu sehen. So hat auch eine volle Punktzahl einen gewissen Aufwand, welcher beachtet werden sollte.

Die ausgewählten Anforderungen sind alle für die Umsetzung des Algorithmus von Bedeutung, jedoch sind diese verschieden stark zu bewerten. Daher werden diesen Anforderungen wiederum Gewichte zugeteilt, welche als Multiplikator für die vergebenen Punkte zu verstehen sind. Die Summe aller mit den entsprechenden Gewichten multiplizierten Punkte ergibt somit die Gesamtpunktzahl für einen Ansatz. Dabei werden die Gewichte so gewählt, dass deren Summe ebenfalls *10* beträgt. Die maximale Punktzahl ist somit *100*.

Die Gewichtung der Anforderungen wird dabei wie folgt gesetzt:

- **Einschränkung der Winkel, Gewicht 4:** Die verschiedenen Transfersysteme müssen zwingend die vorgegebenen Winkel einhalten.
- **Umgehung nicht geeigneter Lücken, Gewicht 3:** Nicht geeignete Lücken müssen beachtet und vom Algorithmus auch übersprungen werden. Diese treten jedoch nur bei einigen komplexen Umgebung auf. Somit ist die Anforderung nicht für alle Modellierungen relevant.

- **Ermittlung kurzer Pfade, Gewicht 2:** Es sollen möglichst kurze Pfade gefunden werden, jedoch sind geringe Abweichungen hinnehmbar, sofern im Allgemeinen gute Ergebnisse zu erwarten sind.
- **Vermeidung von Kurven, Gewicht 1:** Kurven sollen generell vermieden werden, jedoch sind auch Ansätze mit vielen Kurven denkbar, sofern ein guter Pfad erzeugt wird.

| | **Rasterisierung des Raumes** | **A* Algorithmus** | **Navigation Meshes** | **Geometrische Ansätze** |
|---|---|---|---|---|
| **Einschränkung der Winkel (x4)** | 3 | 5 | 2 | 8 |
| **Umgehung nicht geeigneter Lücken (x3)** | 5 | 2 | 5 | 7 |
| **Ermittlung kurzer Pfade (x2)** | 8 | 10 | 7 | 4 |
| **Vermeidung von Kurven (x1)** | 9 | 4 | 8 | 8 |
| **Gesamtpunkte mit Gewichtung** | 52 | 50 | 45 | 69 |

**Tabelle 7.1:** Punktevergabe und gewichtete Ergebnisse für die algorithmischen Ansätze in Bezug auf die Anforderungen

Alle Ansätze der Verfahren sind zur Umsetzung der Anforderungen mit einigem Aufwand verbunden. Dabei stechen jedoch die geometrischen Ansätze aufgrund deren Punktzahl hervor. Besonders bei den wichtigsten Anforderungen, der *Einschränkung der Winkel* und der *Umgehung nicht geeigneter Lücken*, weisen diese die höchste Punktzahl der verschiedenen Ansätze auf. Daher wird ein solcher Ansatz gewählt.

# 8 Algorithmusentwicklung

Dieses Kapitel enthält eine Beschreibung des umgesetzten Algorithmus zur Wegfindung. Hierfür werden zunächst die Besonderheiten der entwickelten Struktur erläutert. Anschließend wird der grundlegende Algorithmus beschrieben, welcher schrittweise erweitert wird, um weitere Vorgaben abzudecken.

Hierbei wird zunächst das Grundgerüst zur Umsetzung des Algorithmus und der Anforderungen beschrieben. Anschließend werden Sonderfälle erläutert, welche bei der Entwicklung des Algorithmus eine besondere Aufmerksamkeit erhalten haben.

## 8.1 Kurzbeschreibung der strukturgebenden Klassen

Der Algorithmus verwendet mehrere Klassen, welche sich in gewissen Punkten von üblichen Implementierungen unterscheiden. Teilweise wirkt sich die gewählte Implementierung stark auf das Vorgehen innerhalb des Algorithmus aus, daher werden die erstellten Klassen zur Speicherung und Verarbeitung der Strukturen und deren Besonderheiten nachfolgend kurz erläutert.

### 8.1.1 Klasse Point

Die Klasse Point stellt zunächst einen typischen Punkt mit je einer X- und Y-Koordinate dar. Weiterhin wird dort jedoch auch gespeichert, ob es sich um einen generierten Punkt oder einen festgelegten Materialflusspunkt handelt, da durch diese Angabe die Punkte unterschiedlich behandelt werden müssen. So dürfen Materialflusspunkte niemals verschoben werden, was bei den generierten Punkten jedoch erlaubt ist.

### 8.1.2 Klasse Path

Der vom Algorithmus berechnete Pfad wird in einer Verkettung von Punkten gespeichert. Beim Hinzufügen und Entfernen von Punkten aus dem Pfad wird hierbei direkt die Gesamtstrecke des Pfades berechnet beziehungsweise aktualisiert.

### 8.1.3 Klasse Line

Diese Klasse beinhaltet zwei Punkte, welche den Beginn und das Ende einer Linie darstellen. Sowohl in der Klasse Obstacle (siehe 8.1.4) als auch bei der Umsetzung des Algorithmus (siehe 8.2.1) ist im Besonderen darauf zu achten, welcher Punkt der Startpunkt und welcher der Endpunkt ist, da die somit angegebene Richtung dort einen großen Einfluss haben kann.

### 8.1.4 Klasse Obstacle

Die durch diese Klasse dargestellten Hindernisse sind in der derzeitigen Implementierung auf Rechtecke beschränkt. Durch eine Anpassung weniger Methoden kann die Klasse jedoch auf beliebige Vielecke erweitert werden. Ein Hindernis wird als eine Liste von Punkten gespeichert, welche die Seiten des Rechtecks angeben. Dabei wird ausgehend von einem definierten Startpunkt immer der nächste Eckpunkt angegeben, welcher sich zu diesem in logischer Reihenfolge entgegen dem Uhrzeigersinn befindet. Alle Methoden dieser Klasse beachten diese Reihenfolge. So werden bei allen zurückgegebenen Linien (siehe 8.1.3) die Start- und Endpunkte entsprechend dieser Vorgabe gesetzt.

Somit enthält jede Seite des Hindernisses indirekt eine Richtungsangabe, welche für den späteren Algorithmus von Nutzen ist (siehe 8.2.4).

**Abbildung 8.1:** Linienrichtung der Hindernisse

### 8.1.5 Klasse Conveyor

Die Klasse Conveyor (englisch für *Fördergerät*) enthält die Daten eines Transfersystems. Enthalten sind neben dessen Namen die umgesetzte Richtung, der Radius bei der Richtungsänderung und die Mindestlänge des Transfersystems. Hierbei ist zu beachten, dass jedes Transfersystem deren umgesetzten Winkel immer in beide

Richtungen umsetzen kann. So kann ein Transfersystem mit 90 Grad Kurve auch 270 Grad umsetzen.

Diese Daten werden für die Berechnung des Materialflusses (8.2.5) und bei Abbildung des Materialflusses durch die Transfersyteme benötigt.

### 8.1.6 Klasse ConveyorSet

Die Klasse ConveyorSet enthält die Transfersysteme einer Komponentengruppe sowie zusätzliche Informationen zu diesen. In dieser Klasse wird beispielsweise vermerkt, ob eine Komponentengruppe einen Umlauf benötigt oder nicht (siehe 3.3).

Die verschiedenen Komponentengruppen werden für diese Arbeit zunächst von Hand eingetragen. Später sollen diese aus einer Datenbank eingelesen werden.

### 8.1.7 Klasse PathCalculator

Alle nicht von obigen Strukturen abgedeckten Funktionalitäten zur Berechnung des Pfades zwischen zwei Punkten werden von der Klasse PathCalculator umgesetzt. Diese Klasse enthält eine Liste der vorhandenen Hindernisse (siehe 8.1.4). Weiterhin ist dort eine Enumeration definiert worden, um die aktuelle Suchrichtung zu kennzeichnen und die erstellten Algorithmen einzuschränken (siehe 8.2.1).

```
enum EnumRunningDirection {
        BOTH_DIRECTIONS,
        START_DIRECTION,
        END_DIRECTION
}
```

## 8.2 Algorithmusbeschreibung

Der implementierte Algorithmus soll einen möglichst kurzen Pfad zwischen zwei Punkten ermitteln. Um alle Verbindungen zwischen den gegebenen Materialflusspunkten zu erhalten, wird das beschriebene Vorgehen daher für jede Verbindung von Materialflusspunkten durchgeführt. Um Hindernisse zu umgehen, werden dabei temporär weitere Materialflusspunkte angelegt.

Um Materialflusspfade in beliebigen Verzweigungen untersuchen zu können, wird in mehreren Methoden des beschriebenen Algorithmus Rekursion verwendet.

### 8.2.1 Grundalgorithmus

Ausgehend von dem Startpunkt wird eine Gerade zu dem Endpunkt gezogen. Sollte diese Gerade dabei auf ein Hindernis treffen, können beide Punkte nicht direkt miteinander verbunden werden. Daher wird das Hindernis sowie die betroffene Seite des Hindernisses ermittelt, welche zuerst geschnitten wird. Das Hindernis soll umgangen werden, indem von den Eckpunkten dieser Seite aus jeweils ein Umgehungspunkt in einem festgelegten Abstand (siehe 8.2.5) berechnet wird. Von diesen ausgehend beginnt das Vorgehen von neuem, solange bis der Endpunkt erreicht werden kann. Um Endlos-Schleifen und unnötige Wiederholungen von Pfaden zu vermeiden, wird die aktuelle Suchrichtung angegeben. Sollte der nächste Schritt das selbe Hindernis betreffen, so wird nur in der aktuellen Suchrichtung weiter gesucht, andernfalls in beide Richtungen.

Aufgrund der Umgehung der Hindernisseite in beide Richtungen entstehen mehrere Pfade, von denen der kürzeste ausgewählt wird.

Die Abbildung 8.2 stellt den Ablauf des Grundalgorithmus dar.

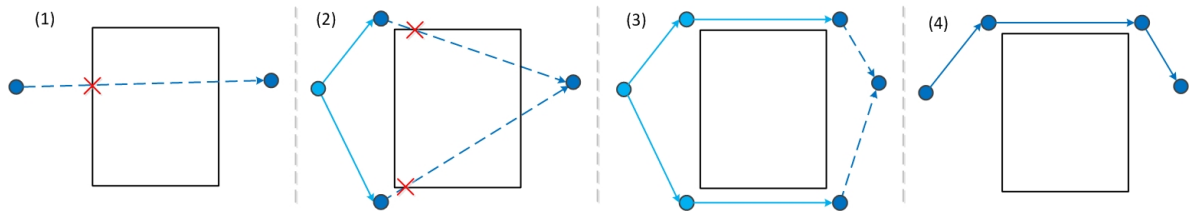



**Abbildung 8.2:** Ablauf des Grundalgorithmus

**Algorithmus 8.1** Grundalgorithmus (Klasse PathCalculator)

```
procedure BERECHNEPFAD(Point startPunkt, Point endPunkt, Path bisherigerPfad,
EnumRunningDirection laufrichtung)
    Line linie = ermittleNaechsteHindernisLinie(startPunkt,endPunkt)
    if linie gefunden then
        if START_DIRECTION für laufrichtung erlaubt then
            Path startPfad = bisherigerPfad          // In StartPunkt-Richtung laufen
            Point p1 = geheZuDistanz(linie.startPunkt)
            if !Pfad.enthaelt(p1) then
                startPfad.ergaenzePunkt(p1)
                if Nächster Schritt trifft gleiches Hindernis then
                    laufrichtung = START_DIRECTION   // In START_DIRECTION
                else
                    laufrichtung = BOTH_DIRECTIONS       // In beide Richtungen
                end if
                startPfad = berechnePfad(p1,endPunkt,startPfad,laufrichtung)
            end if
        end if
        if END_DIRECTION für laufrichtung erlaubt then
            Path endePfad = bisherigerPfad          // In EndPunkt-Richtung laufen
            Point p2 = geheZuDistanz(linie.endPunkt)
            if !Pfad.enthaelt(p2) then
                endePfad.ergaenzePunkt(p2)
                if Nächster Schritt trifft gleiches Hindernis then
                    laufrichtung = END_DIRECTION         // In END_DIRECTION
                else
                    laufrichtung = BOTH_DIRECTIONS       // In beide Richtungen
                end if
                endePfad = berechnePfad(p2,endPunkt,endePfad,laufrichtung)
            end if
        end if
    else
        bisherigerPfad.ergaenzePunkt(endPunkt)         // Kein Hindernis im Weg
    end if
    Path kuerzesterPfad = bisherigerPfad                // Wähle den kürzesten Pfad
    if neuer Pfad ermittelt then
        if startPfad < endePfad then  kuerzesterPfad = startPfad
        else kuerzesterPfad = endePfad
        end if
    end ifreturn kuerzesterPfad
end procedure
```

Die nächste Hindernislinie wird berechnet, indem der Schnittpunkt der gegebenen Linie mit jeder Seite jedes Hindernisses (siehe 8.1.4) berechnet wird.

Die geschnittene Hindernisseite mit der geringsten Distanz zum Startpunkt soll anschließend als nächstes umgangen werden. Entsprechend wird diese als neue Linie (siehe 8.1.3)zurückgeliefert.

**Algorithmus 8.2** Berechnung der kürzesten Distanz zu allen Hindernisseiten (Klasse PathCalculator)

```
procedure ERMITTLENAECHSTEHINDERNISLINIE(Point startPunkt, Point endPunkt)
    Line kuerzesteLinie = null
    double kuerzesteDistanz = unendlich
    for all Obstacles : hindernis do
        Point schnittpunkt= hindernis.naechsterSchnittpunkt(startPunkt, endPunkt)
        double distanz = Abstand schnittpunkt zu startPunkt
        if schnittpunkt vorhanden UND distanz < kuerzesteDistanz then
            kuerzesteDistanz = distanz
            kuerzesteLinie = hindernis.naechsteHindernisLinie(startPunkt, end-
Punkt)
        end if
    end for
     return kuerzesteLinie
end procedure
```

Um die am nächsten gelegene Seite eines von der Linie geschnittenen Hindernisses zu bestimmen, wird der Schnittpunkt mit jeder Seite berechnet. Die geschnittene Seite mit dem geringsten Abstand des Schnittpunktes zum Startpunkt wird ausgewählt.

Zudem wird geprüft, ob beide Eckpunkte der nächsten Seite vom Startpunkt aus erreichbar sind. Dies ist dann von Bedeutung, wenn das Hindernis in einem der Eckpunkte geschnitten wird. Somit sollen nur Seiten gewählt werden, welche sich ohne weitere Zwischenschritte erreichen lassen.

Sollte dies auf beide Seiten zutreffen, wird in der aktuellen Implementierung die Seite mit den höchsten Indices der Eckpunkte gewählt. Dies kann unter Umständen zu längeren Pfaden führen (siehe 10.2.4).

**Algorithmus 8.3** Berechnung der nächsten Seite eines Hindernisses (Klasse Obstacle)

```
procedure NAECHSTEHINDERNISLINIE(Point startPunkt, Point endPunkt)
    double distanz01 = Abstand von startPunkt zu Seite der Punkte mit Index 0
und 1
    double distanz12 = Abstand von startPunkt zu Seite der Punkte mit Index 1
und 2
    double distanz23 = Abstand von startPunkt zu Seite der Punkte mit Index 2
und 3
    double distanz30 = Abstand von startPunkt zu Seite der Punkte mit Index 3
und 0
    Line naechsteLinie = null
    if Schnittpunkt gefunden then
        naechsteLinie = Linie mit geringster Distanz
        if distanz01 am geringsten und Eckpunkte 0 und 1 erreichbar then
            naechsteLinie = Linie mit Punkten 0 und 1
        end if
        if distanz12 am geringsten und Eckpunkte 1 und 2 erreichbar then
            naechsteLinie = Linie mit Punkten 1 und 2
        end if
        if distanz23 am geringsten und Eckpunkte 2 und 3 erreichbar then
            naechsteLinie = Linie mit Punkten 2 und 3
        end if
        if distanz30 am geringsten und Eckpunkte 3 und 0 erreichbar then
            naechsteLinie = Linie mit Punkten 3 und 0
        end if
    end if
     return naechsteLinie
end procedure
```

### 8.2.2 Einhaltung eines Abstands zu Hindernissen

Der berechnete Pfad soll einen frei einstellbaren Mindestabstand zu jedem Hindernis (8.1.4) einhalten. Diese zusätzliche Anforderung wird durch eine Vergrößerung jedes Hindernisses umgesetzt, bevor der eigentliche Algorithmus gestartet wird. Diese Vergrößerung beinhaltet zum einen den Mindestabstand und zum anderen die halbe Breite der Transfersysteme des *ConveyorSets*. Dies liegt darin begründet, dass die Berechnung des Materialflusses linienförmig stattfindet und somit die Breite der Transfersysteme andernfalls nicht berücksichtigt wird.

Dabei wird für jeden Eckpunkt anhand der angrenzenden Seiten des Hindernisses ein Richtungsvektor mit der vorgegebenen Länge bestimmt. Anschließend wird aus beiden Richtungsvektoren der Gesamtvektor für diesen Eckpunkt berechnet und der Punkt entsprechend des Vektors verschoben.

Um Wechselwirkungen durch eine bereits durchgeführte Verschiebung zu vermeiden, werden die verschobenen Eckpunkte temporär in einer eigenen Liste gespeichert. Sobald alle verschobenen Eckpunkte ermittelt wurden, wird die Liste der alten Eckpunkte mit dieser überschrieben.

Das Ergebnis ist eine Vereinfachung der Anforderung. Die resultierenden Seiten des vorherigen Hindernisses halten den vorgegebenen Abstand ein. An den Eckpunkten jedoch wird durch diese Vergrößerung ein etwas größerer Abstand als gefordert eingehalten. Um eine möglichst einfache Form der Struktur der Hindernisse beizubehalten und den Algorithmus nicht weiter zu verkomplizieren wird diese Vereinfachung jedoch in Kauf genommen.

Abbildung 8.3 zeigt die durchgeführte Vergrößerung eines Hindernisses und die korrekte Vergrößerung zur Einhaltung des Abstandes.

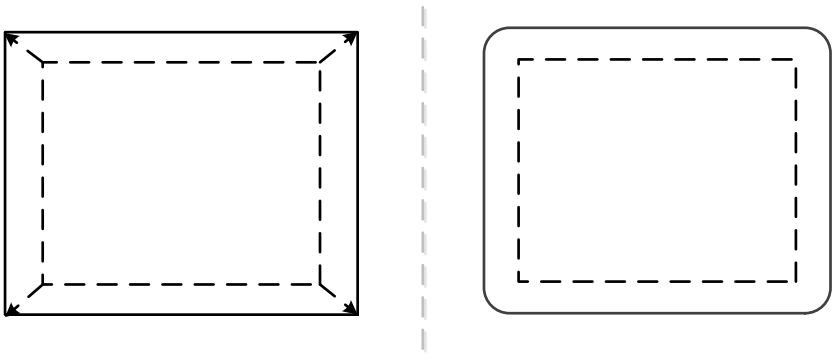


**Abbildung 8.3:** Durchgeführte und korrekte Vergrößerung der Hindernisse

### 8.2.3 Erweiterung zur Umgehung der Hindernisse

Manche Hindernisse können mit dem bisherigen Vorgehen nicht umgangen werden. Es ist möglich, dass die Linie ausgehend von einem ermittelten Punkt zur Umgehung eines Hindernisses wieder die gleiche Seite des Hindernisses schneidet, welche auch zuvor geschnitten wurde. Dabei werden bisher die gleichen Umgehungspunkte ermittelt, welche schon im vorigen Schritt berechnet wurden. Somit kann der gewünschte Pfad nicht weiter verfolgt werden und der Algorithmus bricht für diesen die Suche ab.

Das Auftreten dieses Falles ist abhängig von der Lage des gewünschten Endpunktes und den Ausmaßen des Hindernisses. Je näher der Endpunkt dem zu umgehenden Hindernis ist, desto wahrscheinlicher kann dieses Verhalten Auftreten. Ebenso begünstigen Hindernisse mit großen Seitenlängen diesen Sonderfall.

Dieser Fall tritt häufig auf und muss daher zwingend behoben werden.

Durch eine einfache Erweiterung kann dieses Problem jedoch gelöst werden. Da das gleiche Hindernis erneut geschnitten wird, wird der Pfad nur in bisherige Richtung weiter verfolgt. Es gibt somit keine erneute Aufteilung des Pfades. Sollte der ermittelte Umgehungspunkt dem aktuellen Startpunkt entsprechen, liegt beschriebener Sonderfall vor. Daher wird ein neuer Punkt zur Umgehung des Hindernisses gewählt, welcher über den nächsten Eckpunkt des Hindernisses in der vorgegebenen Richtung (siehe 8.1.4) berechnet wird.

Abbildung 8.4 stellt das Vorgehen anhand der Pfadsuche in Startrichtung grafisch dar.

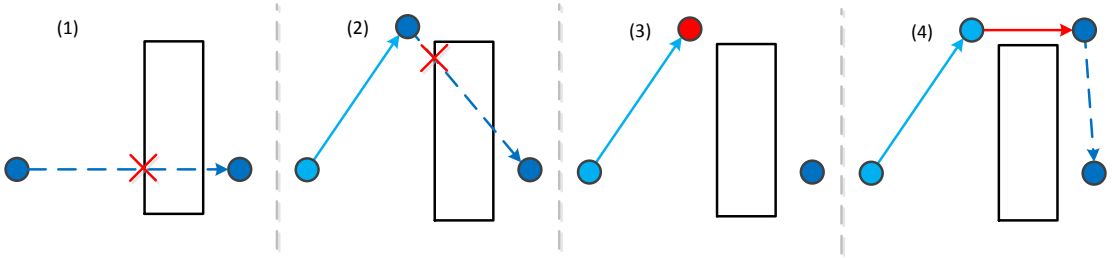



**Abbildung 8.4:** Wahl des Umgehungspunktes des nächsten Eckpunktes bei Suche in gegebener Richtung

**Algorithmus 8.4** Codeausschnitt des erweiterten Grundalgorithmus zur Umgehung von Hindernissen (Klasse PathCalculator)

```
procedure BERECHNEPFAD(Point startPunkt, Point endPunkt, Path bisherigerPfad,
EnumRunningDirection laufrichtung)
    ...
    Path startPfad = bisherigerPfad
    Point p1 = geheZuDistanz(linie.startPunkt)
    if p1 == startPunkt then          // Ergänzte Abfrage für START_DIRECTION
        p1 = Vorgänger von linie.startPunkt
        p1 = geheZuDistanz(p1)
    end if
    ...
    Path endePfad = bisherigerPfad
    Point p2 = geheZuDistanz(linie.endPunkt)
    if p2 == startPunkt then          // Ergänzte Abfrage für END_DIRECTION
        p2 = Nachfolger von linie.endPunkt
        p2 = geheZuDistanz(p2)
    end if
    ...
end procedure
```

### 8.2.4 Umgehung von Hindernisketten

Der Algorithmus liefert für einzeln stehende Hindernisse gute Ergebnisse. Jedoch kann es notwendig sein, mehrere aneinander angrenzende Hindernisse umgehen zu können. Hindernisse können sich ihre Seiten ganz oder teilweise teilen. Auch resultierend aus der Vergrößerung der Hindernisse (siehe 8.2.2), können sich Hindernisse jedoch möglicherweise überschneiden, so dass deren Seiten nur gemeinsame Schnittpunkte aufweisen.

Anhand der bisherigen Berechnung kann nur sicher gesagt werden, dass der Schnittpunkt mit dem Hindernis von dem Startpunkt der Suchlinie aus erreichbar ist. Es ist jedoch nicht sichergestellt, dass auch die Eckpunkte der geschnittenen Hindernisseite erreichbar sind - diese können auch von einem anderen Hindernis verdeckt sein und wären somit nicht erreichbar.

Daher wird weitergehend überprüft, ob der Eckpunkt in Laufrichtung erreichbar ist. Ist dies nicht der Fall, wird das im Weg liegende Hindernis ermittelt. Statt zum nächsten Eckpunkt des eigentlichen Hindernisses zu gehen, wird der Eckpunkt der geschnittenen Seite des neu ermittelten Hindernisses in Laufrichtung gewählt. Da auch dieses wieder verdeckt sein kann, wird dieses Verfahren so oft rekursiv fortgesetzt, bis ein Eckpunkt erreicht werden kann. Somit wird immer der äußerste Rand einer Verkettung von Hindernissen in der Laufrichtung erreicht.

Abbildung 8.5 veranschaulicht die Auswahl der nächsten Eckpunkte bei der Suche in Startrichtung.

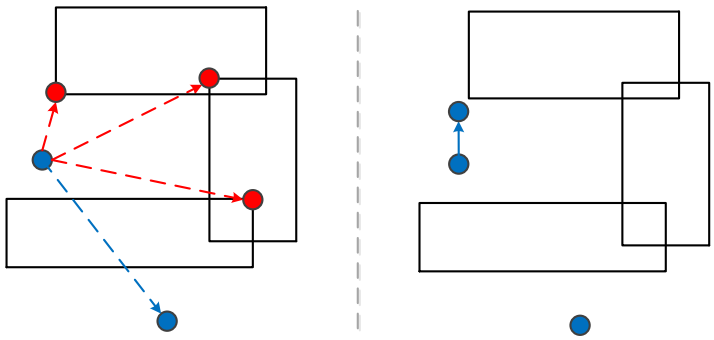


**Abbildung 8.5:** Umgehen der nicht erreichbaren Eckpunkte bei Verkettungen von Hindernissen

Ist das Ende einer Verkettung erreicht, ist es je nach Lage des Startpunktes möglich, dass auch der nächste Eckpunkt des Endhindernisses vom Startpunkt aus erreicht

werden kann. Dieser Eckpunkt kann sich ebenfalls in einem anderen Hindernis befinden, daher wird auch für diesen geprüft, ob eine Hindernisverkettung vorliegt. So können zum einen längere Pfade vermieden werden und zum anderen werden kurze Unterbrechungen bei Hindernisketten ebenfalls erkannt und übersprungen.

Diese Erweiterung betrifft auch das Vorgehen bei einzelnen Hindernissen und wird auch dort angewendet. So werden immer längere Pfade durch das Einfügen zusätzlicher Punkte vermieden, sofern der nächste Eckpunkt des Hindernisses in Suchrichtung ebenfalls erreichbar ist.

Abbildung 8.5 zeigt die Auswahl des nächsten erreichbaren Eckpunktes und dessen Auswirkung bei der Suche in Startrichtung.

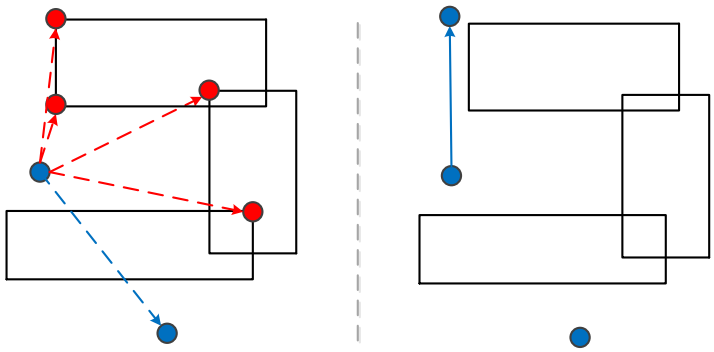


**Abbildung 8.6:** Wahl des nächsten Eckpunktes bei Hindernisketten (in Startrichtung)

**Algorithmus 8.5** Codeausschnitt des erweiterten Grundalgorithmus zur Umgehung von Hindernisketten (Klasse PathCalculator)

```
procedure BERECHNEPFAD(Point startPunkt, Point endPunkt, Path bisherigerPfad,
EnumRunningDirection laufrichtung)
    ...
    Path startPfad = bisherigerPfad
    Point p1 = geheZuHindernisEnde(startPunkt,linie.startPunkt,START_DIRECTION)
    p1 = geheZuDistanz(p1)
    ...
    Path endePfad = bisherigerPfad
    Point p2 = geheZuHindernisEnde(startPunkt,linie.endPunkt,END_DIRECTION)
    p2 = geheZuDistanz(p2)
    ...
end procedure
```

**Algorithmus 8.6** Umgehung von Hindernisketten (Klasse PathCalculator)

```
procedure geheZuHindernisEnde(Point startPunkt, Point eckpunkt, EnumRunningDirection laufrichtung)
    Point endPunkt = eckPunkt
    Obstacle neuesHindernis = Von der Linie (startPunkt,eckPunkt) geschnittes Hindernis
    if neuesHindernis gefunden then
        Line geschnitteneSeite = geschnittene Seite von neuesHindernis
        if laufrichtung ist START_DIRECTION then
            endPunkt=geheZuHindernisEnde(startPunkt,geschnitteneSeite.startPunkt,
            START_DIRECTION)
        else
            endPunkt=geheZuHindernisEnde(startPunkt,geschnitteneSeite.endPunkt,
            END_DIRECTION)
        end if
    else
        Point naechsterPunkt = nächster Eckpunkt von neuesHindernis in laufrichtung
        if naechsterPunkt erreichbar von startPunkt then
            endPunkt = geheZuHindernisEnde(startPunkt, naechsterPunkt, laufrichtung)
        end if
    end if
     return endPunkt
end procedure
```

### 8.2.5 Einschränkung der Winkel

Mit der Anforderung *Einschränkung der Winkel* (siehe 3.3) wird festgelegt, dass Transfersysteme auf bestimmte Winkel zur Umsetzung des Materialflusses eingeschränkt sind. Dies wird auch bei der Erstellung des Materialflusspfades berücksichtigt. Die Winkel sind dabei spezifisch für jedes *ConveyorSet* (siehe 8.1).

Jeder Punkt kann durch Polarkoordinaten dargestellt werden, welche deren Winkel im Raum in Bezug auf den Ursprung des Koordinatensystems angibt.

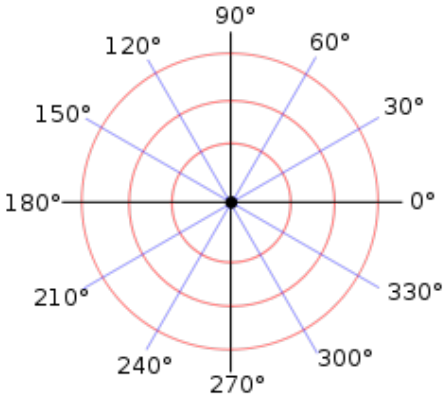



**Abbildung 8.7:** Winkel von Polarkoordinaten zum Ursprung (in Grad) [Wik12]

Durch die Funktion *atan2* kann leicht auch ein Winkel eines Punktes P1 zu einem anderen Punkt P2 berechnet werden.

$$\begin{aligned} \Delta x &= P1.x - P2.x \\ \Delta y &= P1.y - P2.y \\ Winkel(P1, P2) &= atan2(\Delta y, \Delta x) \end{aligned}$$

Mit dieser Funktion kann der Winkel von einem Materialflusspunkt zu seinem Vorgänger bestimmt werden. Setzt man diesen wiederum in Bezug zu dessen Vorgänger, kann man die Winkeländerung ermitteln, indem man beide Winkel direkt miteinander verrechnet. Hierbei muss lediglich die Richtung berücksichtigt werden und ob sich ein neuer Winkel über 360 oder unter 0 Grad ergibt.

Die Abbildung 8.8 veranschaulicht diese Winkelbeziehung, aus welcher sich die Winkeländerung ergibt.

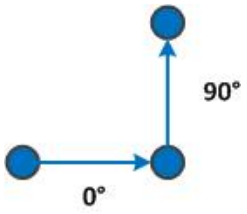



**Abbildung 8.8:** Beispiel der benötigten Winkel zur Berechnung der Winkeländerung

Ein berechneter Materialflusspunkt zur Umgehung eines Hindernisses entspricht in allen Fällen einem Eckpunkt eines Hindernisses. Daher wird dieser so verschoben, dass er den nächstmöglichen Winkel des *ConveyorSets* in Bezug auf seinen Vorgänger einnimmt. Der Eckpunkt entspricht immer dem äußersten Punkt des Hindernisses in Suchrichtung, welcher von dem Vorgänger erreicht werden kann. Für die Ermittlung, ob der nächst größere oder kleinere Winkel verwendet wird, wird die gegebene Struktur der Hindernisse (siehe 8.1.4) genutzt. Alle Winkel bei der Suche in Startrichtung werden auf den Winkel gegen den Uhrzeigersinn, alle in Endrichtung im Uhrzeigersinn gerundet. Welcher der beiden Winkel verwendet werden soll muss daher nicht berechnet werden.

An diesem Vorgehen bei der Winkelrundung ändert sich auch bei anderen Positionen des Startpunktes nichts. Es muss nur sichergestellt sein, dass der äußerste Eckpunkt eines Hindernisses der entsprechenden Richtung erreicht wurde.

Abbildung 8.9 zeigt eine Verschiebung des Punktes und dessen Winkeländerung sowohl für die Start- als auch für die Endrichtung.

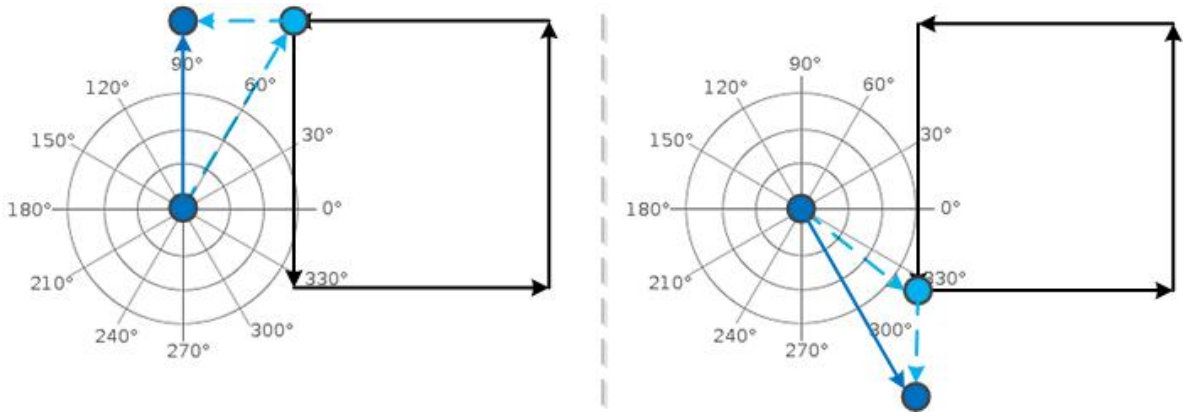



**Abbildung 8.9:** Veranschaulichung der Winkelrundung in Start- und in Endrichtung

Die Richtung des neuen Materialflusspunktes ist nun festgelegt, jedoch nicht dessen genaue Position. Da möglichst kurze Wege gewählt werden sollen, wird der Ort so gewählt, dass der minimale Abstand zum Hindernis zur Umgehung der nächsten Seite des Hindernisses eingehalten wird.

Daher wird eine Parallele zur nächsten, nicht erreichbaren Seite des Hindernisses im Abstand von 1 Pixel gezogen. Der Schnittpunkt dieser Parallelen mit dem Vektor des neuen Winkels wird als neue Position verwendet.

Abbildung 8.10 zeigt eine Verschiebung des Punktes anhand von Winkel und Parallele für die Start- als auch für die Endrichtung.

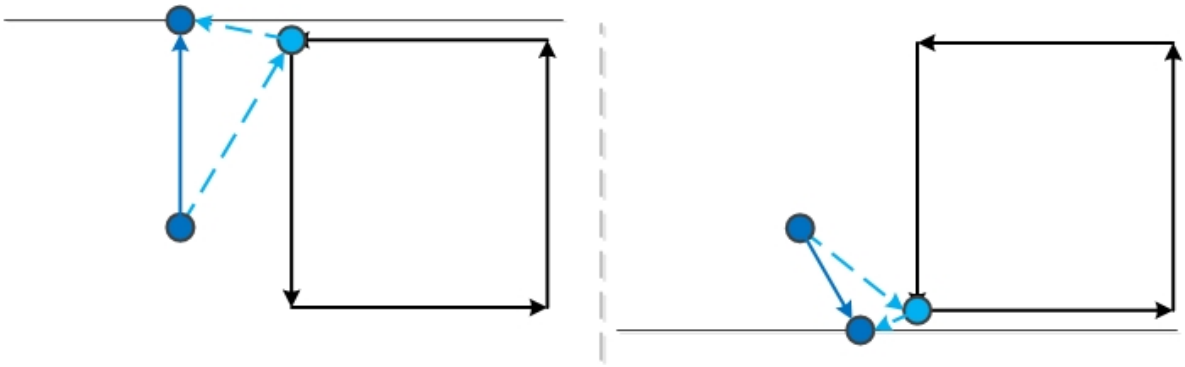


**Abbildung 8.10:** Veranschaulichung der Positionsberechnung in Start- und Endrichtung

Dieses Verfahren wird für alle berechneten Materialflusspunkte angewendet. Die modellierten Materialflusspunkte dürfen jedoch nicht verschoben werden, daher sind diese gesondert zu behandeln.

Ist der Vorgänger des Punktes ein modellierter Materialflusspunkt und dieser besitzt selbst keinen Vorgänger, es ist also der erste Punkt eines Pfades, kann auch kein Winkel zu diesem berechnet werden.

Daher wird eine Verschiebung auf die Parallele der nächsten Seite vorgenommen und der Winkel übernommen. Dies wirkt sich auf alle nachfolgenden Materialflusspunkte aus, da dort die initiale Richtung vorgegeben wird. Diese sollte später optimiert werden (siehe 10.2.2). Besitzt der Materialflusspunkt einen Vorgänger, wird das bisherige Verfahren angewendet.

Kann der Zielpunkt der Berechnung, also ein modellierter Materialflusspunkt, erreicht werden, so darf dieser nicht verschoben werden. Ist der Winkel zu diesem ein erlaubter Winkel, wird dieser direkt verbunden.

Ist dies nicht der Fall, wird ein weiterer Materialflusspunkt eingefügt, so dass von diesem zu dem Zielpunkt ein erlaubter Winkel entsteht. Dazu wird entgegengesetzt wie bisher der nächste Winkel des *ConveyorSets* ermittelt. Dieser Winkel wird nicht wie bisher an den Vorgänger des Materialflusspunktes angelegt, sondern an den Zielpunkt. Der Schnittpunkt dieses Vektors mit der vektorisierten vorigen Verbindung des Vorgängers wird die neue Position des zusätzlichen Materialflusspunktes.

Abbildung 8.11 stellt die Berechnung eines weiteren Materialflusspunktes zur Erreichung eines vorgegebenen Punktes mit einem möglichen Winkel dar.

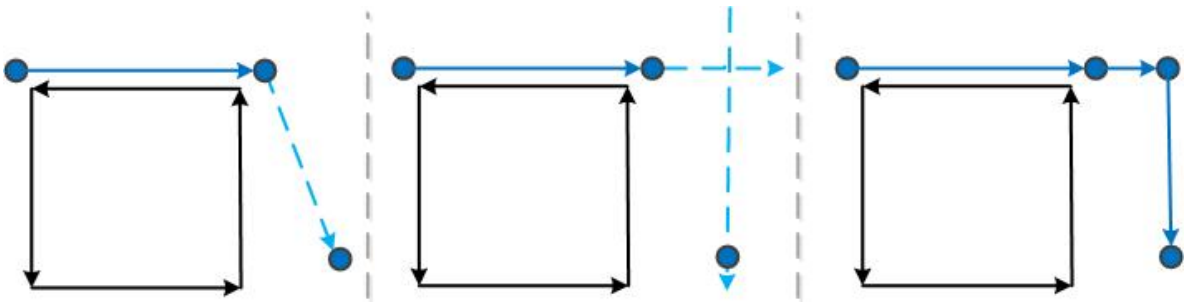


**Abbildung 8.11:** Veranschaulichung der Punktberechnung bei Zielerreichung

**Algorithmus 8.7** Codeausschnitt geänderter Grundalgorithmus zur Winkeleinschränkung

```
procedure BERECHNEPFAD(...)
    ...
    Point p1 = geheZuHindernisEnde(startPunkt,linie.startPunkt,START_DIRECTION)
    p1 = berechnePunktMitWinkel(startPunkt, p1, START_DIRECTION)
    ...
    Point p2 = geheZuHindernisEnde(startPunkt,linie.endPunkt,END_DIRECTION)
    p2 = berechnePunktMitWinkel(startPunkt, p2, END_DIRECTION)
    ...
end procedure
```

**Algorithmus 8.8** Einschränkung der Winkel (Klasse PathCalculator)

**procedure** BERECHNEPUNKTMITWINKEL(Point startPunkt, Point eckpunkt, EnumRunningDirection laufrichtung)
  **if** startPunkt hat Vorgänger **then**
    double winkel = Winkel von startPunkt zu eckPunkt
    **if** laufrichtung ist START_DIRECTION **then**
      winkel = nächster Winkel gegen Uhrzeigersinn
    **else**
      winkel = nächster Winkel in Uhrzeigersinn
    **end if**
    Point endPunkt = Schnittpunkt der Parallelen mit winkel ausgehend von startPunkt
  **else**
    Point endPunkt = verschobener eckPunkt auf Parallele um 1 Pixel
  **end if**
  **return** endPunkt
**end procedure**

**Algorithmus 8.9** Erweitern der Methode ergaenzePunkt(Klasse PathCalculator)

**procedure** ERGAENZEPUNKT(Point endPunkt, Point vorgaenger, EnumRunningDirection laufrichtung)
  **if** vorgaenger hat auch Vorgänger **then**
    double winkel = Winkel von vorgaenger zu endPunkt
    **if** laufrichtung ist START_DIRECTION **then**
      winkel = nächster Winkel in Uhrzeigersinn
    **else**
      winkel = nächster Winkel gegen Uhrzeigersinn
    **end if**
    Point neuerPkt = Schnittpunkt voriger Vektor und endPunkt mit winkel als Vektor
    Hinzufügen von neuerPkt zu Pfad
  **end if**
  Hinzufügen von endPunkt zu Pfad
**end procedure**

### 8.2.6 Umgehung nicht geeigneter Lücken

Die erzeugten Materialflusspunkte werden aufgrund der Winkelanpassung (siehe 8.2.5) verschoben. Dabei ist es möglich, dass dieser Winkel nicht umgesetzt werden kann, da sich ein weiteres Hindernis auf dem neu erzeugten Pfad befindet. Dies war bei den meisten anderen Verfahren zur Wegfindung (siehe 7.1) ein großes Problem, da solche Lücken zwischen mehreren Hindernissen nur schwer umgangen werden können.

Bei dem aktuellen Ansatz ist dies jedoch recht einfach möglich. Der Pfad zu dem neuen Materialflusspunkt wird verfolgt. Befindet sich auf diesem ein weiteres Hindernis, so wird der Materialflusspunkt verworfen. Beginnend mit dem Schnittpunkt des Pfades mit dem im Weg liegenden Hindernis wird nach dem üblichen Vorgehen ein neuer Materialflusspunkt ermittelt. Dabei wird die bisherige Suchrichtung beibehalten.

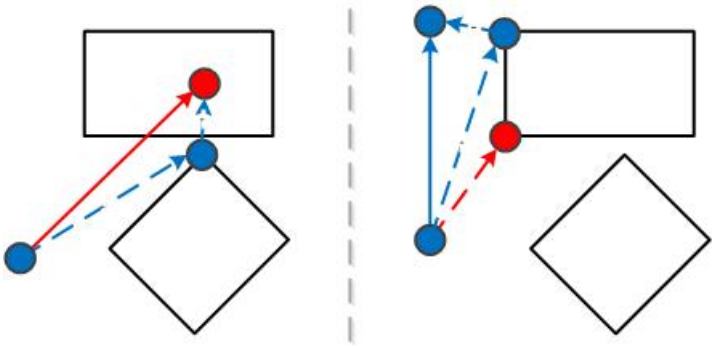


**Abbildung 8.12:** Veranschaulichung der Umgehung nicht geeigneter Lücken

### 8.2.7 Vermeiden von Überschneidungen

Es wird davon ausgegangen, dass sich Transfersysteme nicht überschneiden dürfen. Für die aktuelle Implementierung werden daher für die bisherigen Strecken des Materialflusses Hindernisse erstellt. Diese verhindern, dass ein weiterer Materialfluss durch den erstellten Pfad erfolgt. Dies ist ein vereinfachender Ansatz, welcher später verbessert werden könnte (siehe 10.2.5).

Bei der Erstellung dieser Hindernisse wird die berechnete Verbindung zwischen den Materialflusspunkten um die halbe Breite der Transfersysteme des *ConveyorSets* plus den Mindestabstand (siehe auch 8.2.2) expandiert. Somit können sich Überschneidungen mit anderen Hindernissen ergeben. Trotz Überschneidungen werden

aufgrund der Expansion der Hindernisse (8.2.2) jedoch weiterhin die geforderten Abstände eingehalten.

Der bisherige Algorithmus berücksichtigt bereits Überschneidungen von Hindernissen, daher müssen diese nicht gesondert beachtet werden.

Hierbei ist zu beachten, dass von dem Zielpunkt des Materialflusses weitere Verbindungen möglich sind. Dieser darf somit nicht innerhalb eines Hindernisses liegen, wenn man nicht weitere komplexe Ausnahmen zu dem Algorithmus hinzufügen will. Daher wird das Hindernis so erstellt, dass der Zielpunkt des Materialflusses gerade noch außerhalb des Hindernisses liegt. In Richtung des Startpunktes wird um den Minimalabstand expandiert.

Abbildung 8.13 zeigt die Erstellung der Hindernisse für den Materialflusspfad anhand der halben Breite der Transfersysteme und dem geforderten Mindestabstand.

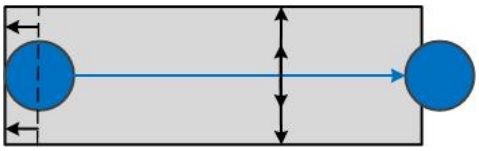


**Abbildung 8.13:** Veranschaulichung der Erstellung von Hindernissen für die Pfade

### 8.2.8 Berücksichtigung von Umläufen

Die Anforderung *Berücksichtigung von Umläufen* (siehe 3.3) besagt, dass bestimmte Transfersysteme in sich geschlossen sein müssen.

Der Modellierer kann dies bereits bei der Erstellung der Materialflusspunkte berücksichtigt haben, indem die Verbindungen der Materialflusspunkte eine geschlossene Kette bilden. Somit existiert kein klar ersichtlicher Beginn und kein eindeutiges Ende des Materialflusses. Ist dies der Fall, wird der Umlauf auch bereits beim Algorithmus berücksichtigt und keine weitere Anpassung ist notwendig.

Wurde keine Verkettung modelliert, wird eine solche Verbindung hinzugefügt, sofern das *ConveyorSet* dies voraussetzt. Hierbei wird ausgehend vom letzten erreichten Materialflusspunkt eine Verbindung zum ersten Materialflusspunkt des Pfades erstellt und der entsprechende Pfad berechnet.

In beiden Fällen ist lediglich zu beachten, dass der erste Materialflusspunkt eines erzeugten Hindernisses (siehe 8.2.7) liegen kann. Dieses ist daher bei der Verbindung der Materialflusspunkte zu ignorieren.

Abbildung 8.14 stellt die Erzeugung eines Umlaufes und den neu berechneten resultierenden Pfad dar.

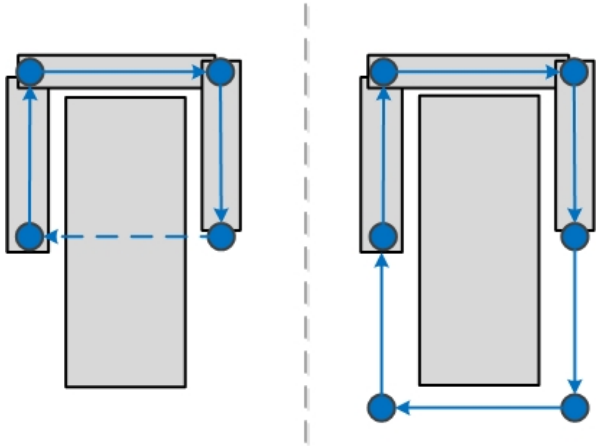


**Abbildung 8.14:** Veranschaulichung einer Verbindung zur Berücksichtigung des Umlaufes

# 9 Ergebnisse

Für die modellierte Umgebung wird jeweils eine Variante für jede *Komponentengruppe* erstellt. Die Varianten können sich stark voneinander unterscheiden, da für diese unterschiedliche Transfersysteme und somit auch Winkel zugrunde liegen.

Der Algorithmus berechnet für jede Variante einen Pfad, entlang dem passende Transfersysteme mit den entsprechenden Eigenschaften generiert werden. Jedes Transfersystem wird als Instanz unterhalb des Elements *Variante* angelegt. Zudem wird ein *Variantendiagramm* zur grafischen Anzeige jeder erzeugten Variante erstellt.

Die Komponenten des *EPLAN Engineering Centers* sind derzeit auf rechtwinklige Geometrien beschränkt. Daher wird folgend eine Variante dargestellt, deren *Komponentengruppe* nur gerade und rechtwinklige Transfersysteme enthält. Komplexere Varianten können zur Zeit berechnet, aber nur unzureichend grafisch dargestellt werden.

Abbildung 9.1 zeigt eine modellierte Materialflussumgebung und die Struktur von deren generierten Instanzen.

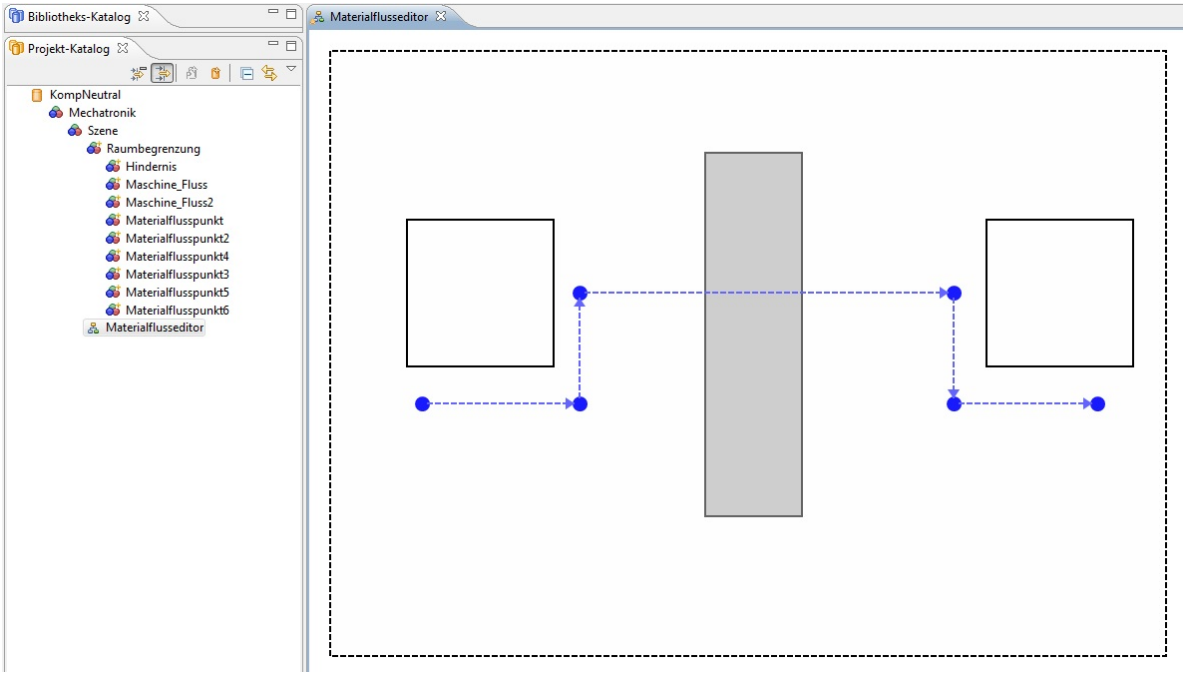



**Abbildung 9.1:** Modellierter Materialfluss mitsamt der generierten Instanzen

Für jeden Materialflusspunkt können neben deren modellierten Eigenschaften wie Position, Vorgänger und Nachfolger weitere Eigenschaften angegeben werden. Hierfür wurden die Eigenschaften zur Angabe der Teilfunktionen *Sichern* und *Kontrollieren*

erstellt. Derzeit wird lediglich angegeben, ob diese Teilfunktionen vorhanden sind oder nicht.

Abbildung 9.2 zeigt die grafische Benutzeroberfläche zur Anzeige der Eigenschaften bei der Anwahl eines Materialflusspunktes.

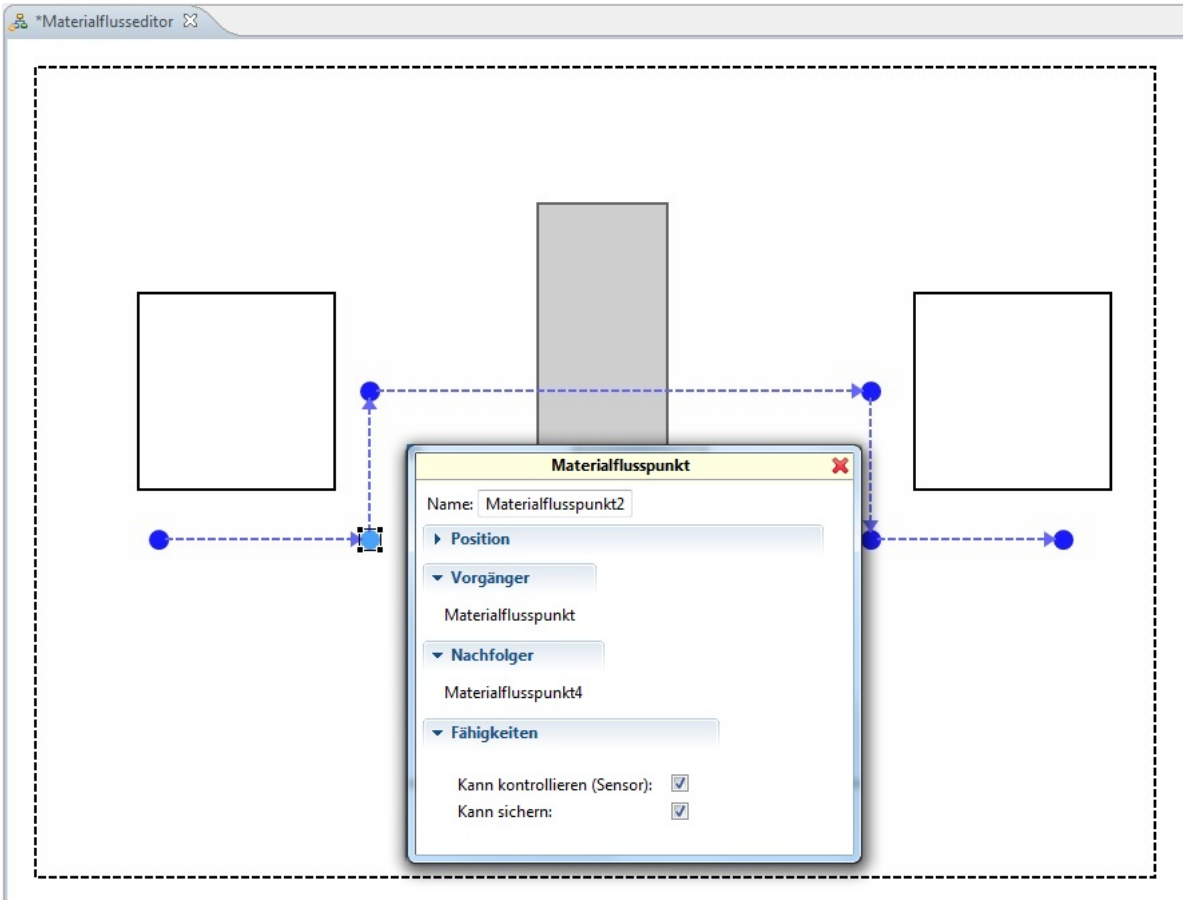



**Abbildung 9.2:** Anzeige der Eigenschaften eines Materialflusspunktes

Aus dem modellierten Materialfluss können anschließend die Varianten berechnet werden. Dabei wird die modellierte Umgebung übernommen und ein Pfad aus den Materialflusspunkten berechnet, welcher über die Transfersysteme der entsprechenden *Komponentengruppe* umgesetzt wird.

Abbildung 9.3 zeigt das *Variantendiagramm* einer Variante einschließlich der generierten Struktur der Instanzen. Dieser Variante liegt der in Abbildung 9.1 gezeigte Materialfluss zugrunde.

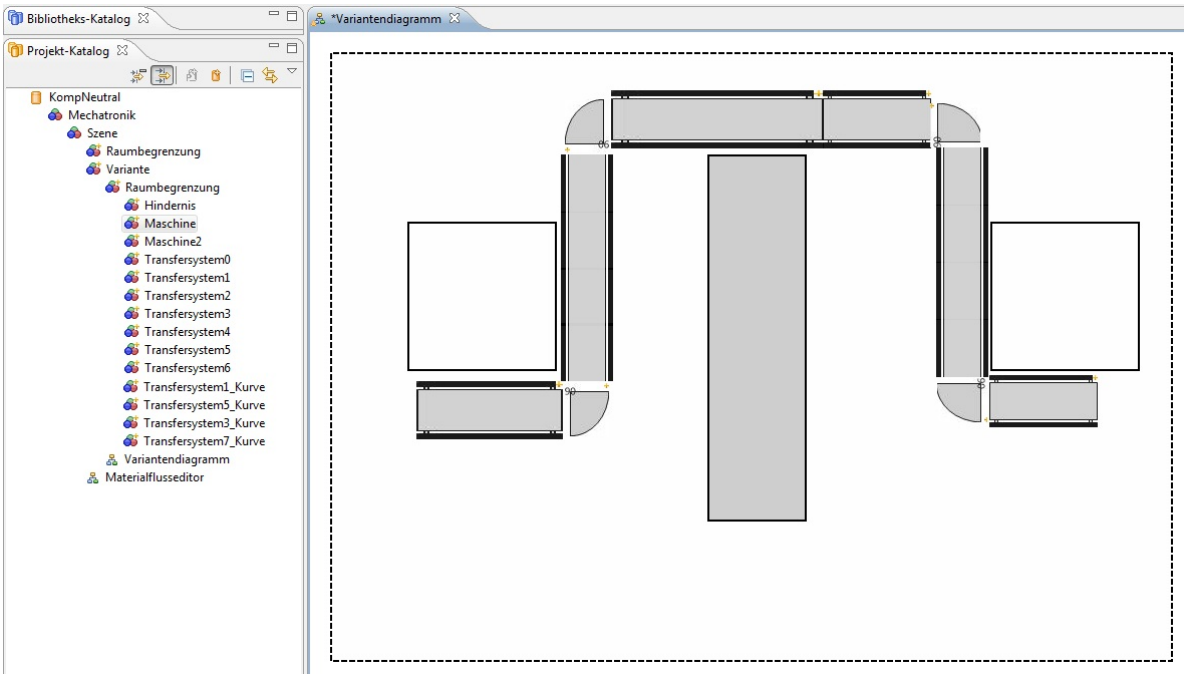



**Abbildung 9.3:** Berechnete Variante mitsamt der generierten Instanzen

Die zusätzlichen Eigenschaften der Materialflusspunkte werden für die Transfersysteme an der entsprechenden Position übernommen. Abbildung 9.4 zeigt die grafische Benutzeroberfläche zur Anzeige der Eigenschaften eines Transfersystems.

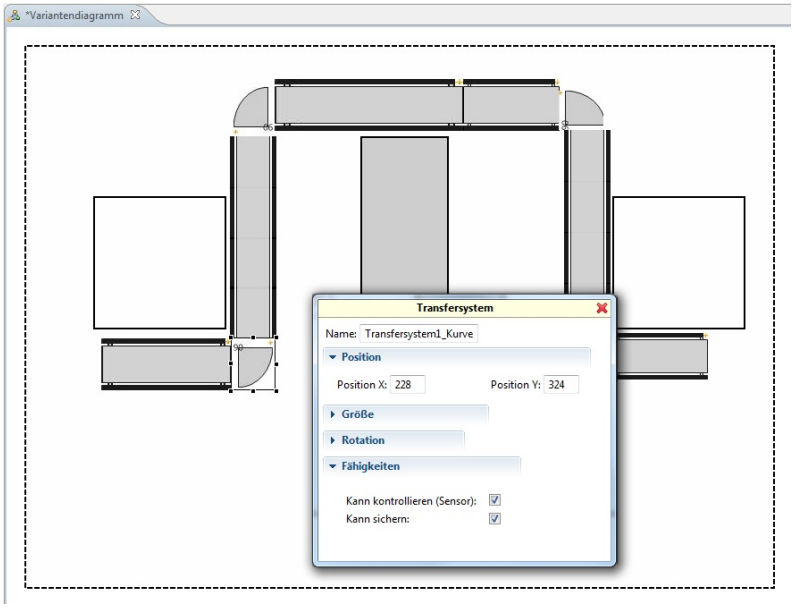



**Abbildung 9.4:** Anzeige der übernommenen Eigenschaften eines Transfersystems

# 10 Fazit und Ausblick

Dieses Kapitel beinhaltet eine Zusammenfassung der Arbeit und gibt einen Ausblick über Erweiterungsmöglichkeiten des ausgewählten Ansatzes zur Generierung der Varianten.

## 10.1 Fazit

Die Aufgabenstellungen zur Umsetzung der komponentenneutralen Beschreibung und der Variantenbildung war sehr frei formuliert. Daher wurden weitere Anforderungen sowohl für die Struktur, die Modellierung des Materialflusses wie auch für die Berechnung der Varianten erarbeitet.

Aufgrund des zeitlichen Rahmens dieser Arbeit wurde eine Auswahl der umzusetzenden Anforderungen getroffen. Die ausgewählten Anforderungen wurden mit weiteren Einschränkungen wie der Verwendung einer zweidimensionalen Umgebung umgesetzt.

Zur Modellierung des Materialflusses wurde ein Konzept erstellt, wobei mehrere Vorgaben analysiert und teilweise verworfen oder abgeändert wurden. Unter Berücksichtigung der Anforderungen wurde eine Struktur entworfen, welche eine Modellierung des Materialflusses ermöglicht. Auf deren Basis wurde im *EPLAN Engineering Center* ein Editor zur Erzeugung von Materialflüssen und deren Umgebung erstellt. Aus diesem werden automatisch Instanzen der Baukästen generiert, welche durch die Struktur beschrieben werden.

Um aus der modellierten Umgebung die Varianten berechnen zu können, wurden mehrere Ansätze zur Wegfindung untersucht und bewertet. Dabei wurde der geometrische Ansatz ausgewählt. Dieser ermittelt die Wege zwischen Punkten anhand von Geraden und deren Schnittpunkten mit den Hindernissen.

Der Algorithmus wurde beschrieben und Erweiterungen und Sonderfälle wurden erläutert. Dabei wurde vor allem auf die Einschränkung der Winkel, aber auch auf die Umsetzung der anderen ausgewählten Anforderungen eingegangen.

Der ausgewählte Ansatz eignet sich gut zur Generierung der Varianten. Allerdings weist er, wie die anderen Ansätze, auch Schwächen bei der Umsetzung einiger Anforderungen auf. Daher sollte der Algorithmus noch erweitert werden, um die nicht umgesetzten Anforderungen zu berücksichtigen.

## 10.2 Erweiterungsmöglichkeiten

Das vorgestellte Verfahren besitzt einige Vereinfachungen, welche aufgelöst werden könnten, um eine größere Anwendbarkeit auf verschiedene Umgebungen zu erreichen. Abseits der Handhabungsfunktion könnten auch weitere Aufgaben der Materialflusspunkte und dementsprechend auch der Transfersysteme hinzugefügt werden.

Bei der Analyse der Anforderungen (3.3) wurden bereits weitere Punkte genannt, welche bei einer Erweiterung des Algorithmus beachtet werden sollten, um den gewünschten Materialfluss zu erstellen. Die während dieser Arbeit nicht umgesetzten Anforderungen werden in den nachfolgenden Unterkapiteln nochmals genauer erläutert. Auch Lösungsansätze sowie mögliche Probleme für diese werden beschrieben.

### 10.2.1 Einhaltung von Mindestlängen

Transfersysteme können nicht beliebig klein hergestellt werden, sondern besitzen gewisse Mindestgrößen. Diese wurden bisher noch nicht berücksichtigt. Bei der Erstellung der Pfade sollte daher geprüft werden, ob die Mindestlänge der Transfersysteme zur Umsetzung der Geraden sowie der anschließenden Kurve eingehalten werden kann.

### 10.2.2 Wahl der Startrichtung

Die initiale Richtung wirkt sich auf alle Pfade zwischen den Punkten aus, da diese jeweils ausgehend von der vorherigen Richtung berechnet werden. Um eine möglichst gute Startrichtung zu ermitteln, ist es daher sinnvoll, alle Verbindungen zwischen den Punkten zu berücksichtigen. Dabei sollte beachtet werden, wie häufig unterschiedliche Winkel bei der Wegfindung benötigt werden und auch, mit welchen Winkeln Hindernisse besonders gut umgangen werden können. Manche Lücken zwischen Hindernissen können sogar nur mittels bestimmter Winkel genutzt werden. Ist ein solcher nicht verfügbar, muss das gesamte Hindernis umgangen werden, was zu einem deutlich längeren Pfad führen kann.

Weiterhin könnten auch die Materialflusspfade innerhalb der Maschinen berücksichtigt werden. Meist ist es erwünscht, dass die Pfade innerhalb der Maschine exakt dem modellierten Materialfluss entspricht. Dies ist liegt vor allem darin begründet, dass innerhalb einer Maschine meist nicht ein beliebiger Materialfluss erfolgen kann.

Abbildung 10.1 veranschaulicht die Auswirkungen unterschiedlicher initialer Richtungen anhand einer Hindernislücke.

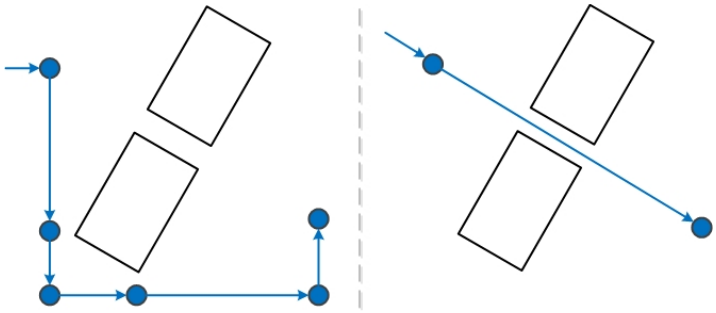


**Abbildung 10.1:** Beispiele der Pfadberechnung bei unterschiedlicher Winkelvorgabe der Startrichtung

### 10.2.3 Zusammenfassen von Punkten gleicher Richtung

Aufgrund der Verschiebung von Materialflusspunkten ist es möglich, dass sich mehrere von diesen auf einer Geraden befinden. Dies ist zur Berechnung der Umgehung von Hindernissen erforderlich. Zur Vereinfachung der Berechnung von Transfersystemen könnten Materialflusspunkte auf einer Geraden gelöscht werden.

Dies wäre durch einen Abgleich des Winkels zum Vorgänger und Nachfolger möglich. Ist dieser gleich, so kann der Materialflusspunkt entfernt werden. Nur Materialflusspunkte ohne weitere Fähigkeiten sollten gelöscht werden, da ansonsten Informationen verloren gehen. Somit sind vor allem die während dem Algorithmus berechneten Materialflusspunkte betroffen.

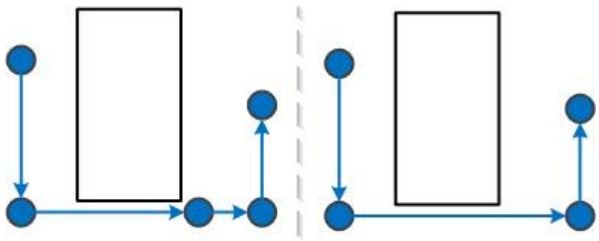


**Abbildung 10.2:** Beispiel der Entfernung von Materialflusspunkten einer Geraden

### 10.2.4 Überprüfung des Pfades

Bei sehr verwinkelten Hindernisketten ist es derzeit möglich, dass zunächst Materialflusspunkte zur Umgehung von Hindernissen erzeugt werden, welche nicht zur Umgehung der Verkettung der Hindernisse beitragen. Somit werden längere Pfade als nötig erzeugt.

Dies könnte behoben werden, indem nach jeder Erzeugung eines Materialflusspunktes geprüft wird, ob auch ein anderer Materialflusspunkt des bisherigen Pfades abseits von dessen direktem Vorgänger erreicht werden kann. Eine spätere Anpassung des Gesamtpfades ist nicht sinnvoll, da die Lage des Materialflusspunktes sich auf den kompletten nachfolgenden Pfad auswirken kann.

Das Entfernen von Materialflusspunkten, welche zu längeren Pfaden führen, ist mit einigem Aufwand verbunden. Es müssen nicht nur die Materialflusspunkte entfernt, sondern auch die kürzeren Verbindungen erneut mit den korrekten Winkeln berechnet werden.

Abbildung 10.3 zeigt eine Skizze eines leicht verlängerten Pfades aufgrund einer verwinkelten Hindernisverkettung.

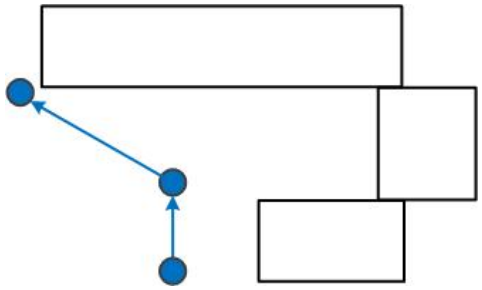


**Abbildung 10.3:** Beispiel eines Teilpfades zur Umgehung verwinkelter Hindernisse

### 10.2.5 Ermittlung des kürzesten Pfades bei Überschneidungen

Um Überschneidungen zu Vermeiden, werden bisher Hindernisse für die berechneten Pfade gesetzt. Somit werden bei Überschneidungen die zuerst berechneten Teilpfade direkt übernommen, während der später erstellte Teilpfad den vorigen umgehen muss.

Es ist jedoch auch möglich, dass ein kürzerer Gesamtpfad erreicht wird, wenn der später erstellte Teilpfad übernommen wird und der vorige Teilpfad diesen umgehen muss. Daher könnten alle Möglichkeiten zur Umgehung der Überschneidung berechnet und der kürzeste Gesamtpfad gewählt werden.

Abbildung 10.4 veranschaulicht die verschiedenen Pfade für die Überschneidung zweier Materialflusspfade. Die dunkleren Punkte stellen dabei den modellierten Materialfluss dar und die gestrichelte Linie den Pfad, welcher aufgrund der Überschneidung nicht umgesetzt werden kann.

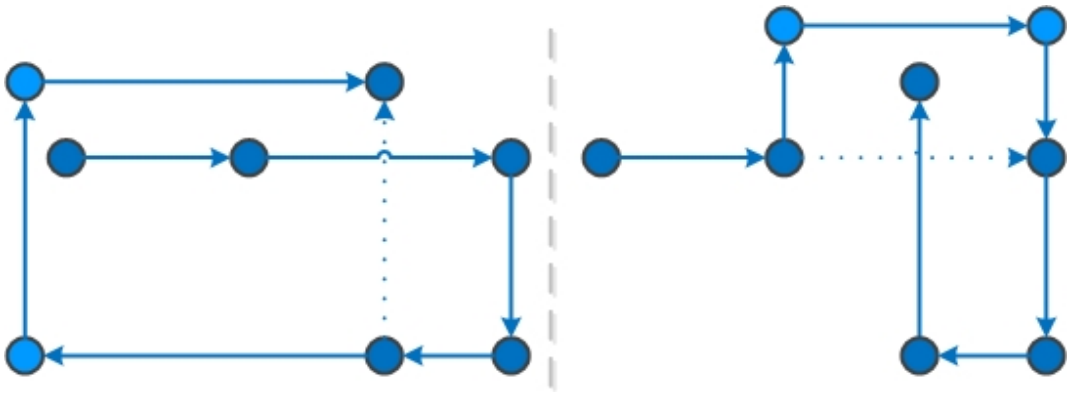


**Abbildung 10.4:** Beispiel der möglichen Pfade bei Überschneidungen

# Literaturverzeichnis


[Cas12] N. Caspari. Entire Documentation EPLAN Engineering Center, 2012. (Zitiert auf den Seiten 18, 41, 42, 43 und 44)

[DO12] Delphi-OpenGLCommunity. DGL Wiki, 2012. URL http://wiki.delphigl.com/index.php/Navigation_Meshes. (Zitiert auf Seite 48)

[EPL12] EPLAN. EPLAN Homepage, 2012. URL http://www.eplan.de/produkte/mechatronik/eplan-engineering-center/. (Zitiert auf Seite 17)

[ES11] T. Ertl, F. Sadlo. Skript zur Vorlesung: Visualization, SS2011, 2011. (Zitiert auf Seite 46)

[Hay08] S. Hayles. Autonomous and Extensible Exploration and Navigation in Unmapped Terrains, 2008. (Zitiert auf Seite 48)

[Hir12] Hiroaki Nishikawa. Divergence formulation of source term. *Journal of Computational Physics*, 231(19):6393–6400, 2012. doi:10.1016/j.jcp.2012.05.032. URL http://www.sciencedirect.com/science/article/pii/S0021999112003014. (Zitiert auf Seite 46)

[ISG12] ISG. ISG Homepage, 2012. URL http://www.isg-stuttgart.de/virtuos.html. (Zitiert auf Seite 19)

[Lud07] J. Ludewig. *Software Engineering: Grundlagen, Menschen,Prozesse, Techniken*. dpunkt.verlag, 1 Auflage, 2007. (Zitiert auf Seite 10)

[Mat02] J. Matthews. AI Game Programming Wisdom. 2002. (Zitiert auf Seite 47)

[Ney12] A. Neyrinck. Optimale Maschinen und Anlagen durch Simulation von Varianten in der Konzeptionsphase. 2012. (Zitiert auf Seite 9)

[Sch12] Schnaithmann Maschinenbau GmbH. Schnaithmann Homepage: Projekt Simvar, 2012. URL http://www.schnaithmann.de/simvar.0.html. (Zitiert auf Seite 9)

[SF04] T. Schmidt, D. Fuchs. Homepage Geosimulation, 2004. URL http://www.geosimulation.de/methoden/a_stern_algorithmus.htm. (Zitiert auf Seite 47)

[Sie09] M. Siebenhaller. Orthogonal graph drawing with constraints: Algorithms and Applications. 2009. (Zitiert auf Seite 46)

[Uni10] Universität Stuttgart. Studienplan Diplomstudiengang Softwaretechnik, 2010. URL http://www.informatik.uni-stuttgart.de/fileadmin/user_upload/fakultaet/studium/studienplan2010.pdf. (Zitiert auf Seite 10)

[Ver90] Verein Deutscher Ingenieure. VDI-Richtlinie 2860. 1990. (Zitiert auf den Seiten 14 und 20)

[Ver07] Verein Deutscher Ingenieure. VDI-Richtlinie 3300. 2007. (Zitiert auf Seite 12)

[Wik12] Wikipedia. Wikipedia: Polarkoordinaten, 2012. URL http://de.wikipedia.org/wiki/Polarkoordinaten. (Zitiert auf Seite 65)

Alle URLs wurden zuletzt am 07. 12. 2012 geprüft.

**Erklärung**

Ich versichere, diese Arbeit selbständig verfasst zu haben.
Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet.
Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens.
Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht.
Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

---

(Benjamin Behrens)