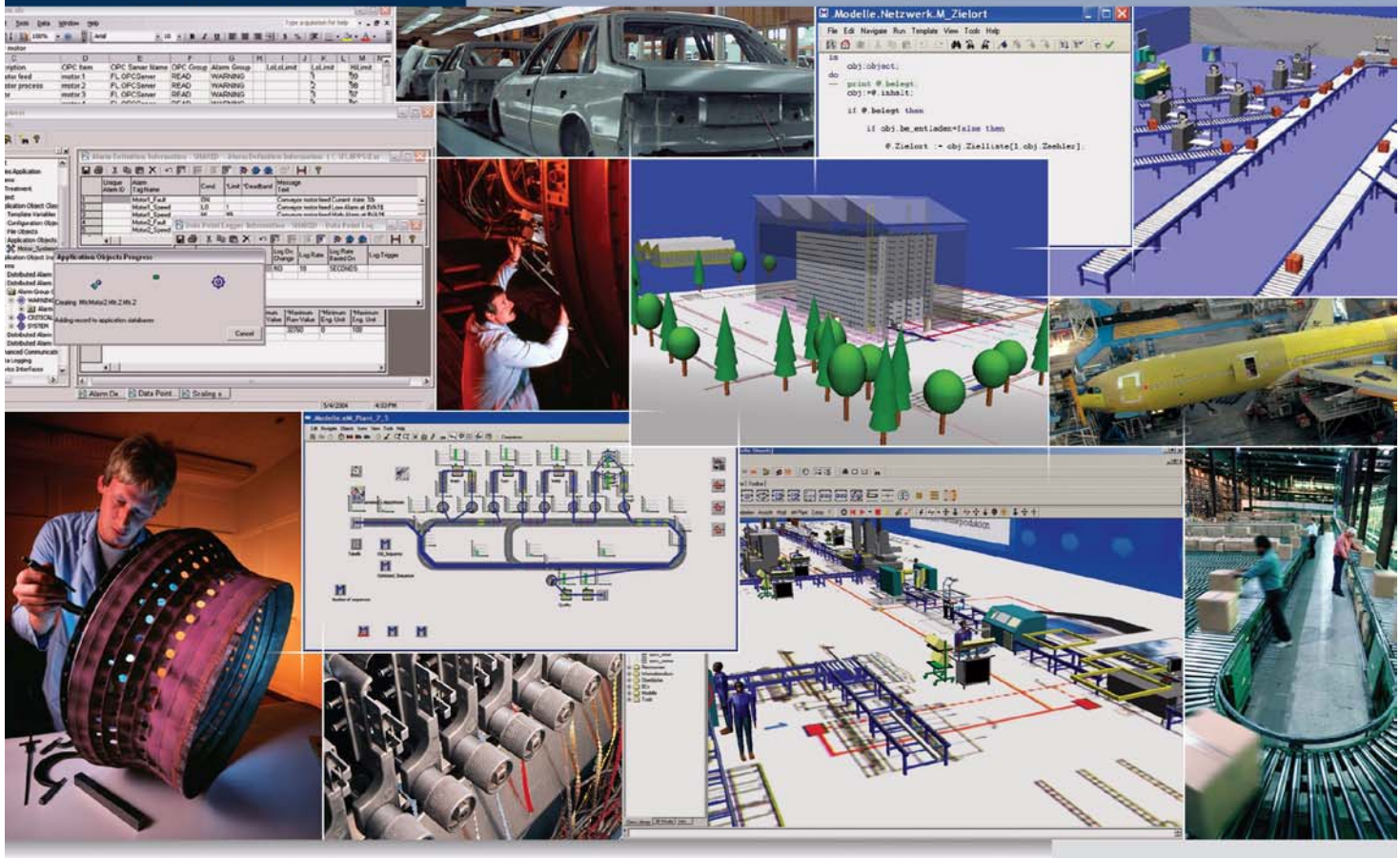
Plant Simulation Transport library

Reference manual

Siemens PLM Software

www.siemens.com/plm



TECNOMATIX

SIEMENS

Hinweise zu Eigentumsrechten

TECNOMATIX
Plant Simulation

SIEMENS

Bibliotheken für Fördersysteme

Dezember 2008

Inhaltsverzeichnis

1

Überblick zu dem Bibliotheken für Fördersysteme

Die Objektbibliotheken Plant Simulation AGVS (**A**utomated **G**uided **V**ehicle **S**ystem), Plant Simulation Conveyor, Plant Simulation EOM (**E**lectric **O**verhead **M**onorail) und Plant Simulation HBW (**H**igh **B**ay **W**arehouse) sind für die Nachbildung typischer Materialflusssyteme entwickelt worden. Diese Transport- und Lagerhaltungssysteme dienen der Unterstützung von Produktionssystemen, die mit der Objektbibliothek Plant Simulation Shop Light abgebildet werden können. Der Materialfluß in einem Produktionssystem wird durch Auftragslisten und Arbeitsplänen beschrieben. Sie können natürlich auch eine andere Modellierung eines Produktionssystems verwenden, wenn die verwendeten BEs die erfolderlichen Attribute besitzen. Wollen Sie die Performance eines Materialflusssystem ohne Anbindung an ein konkretes Produktionssystems bestimmen, so können Sie lokale Routingtabellen verwenden. Mit Routingtabellen bestimmen Sie das nächste Ziel eines BEs an einer Station des Materialflusses. Es ist auch möglich mit anderen Materialflußsystemen eine Kopplung über Schnittstellen zu simulieren.

Notwendige technische Parameter und Steuerungsregeln können einfach in Dialogen festgelegt werden. Bei der Modellierung mit der Objektbibliothek EOM müssen die Daten in Tabellen und Variablen der Bausteine eingegeben werden. Alle Bausteinkästen sind auf der Grundlage von Plant Simulation entwickelt. Es ist eine Plant Simulation Professional License, Plant Simulation Standard License oder Plant Simulation Application License erforderlich. Zur Anwendung der Bausteinkästen werden Kenntnisse in Plant Simulation vorausgesetzt.

Hinweis: Die Bausteinkästen sollen die jeweilige Modellierungsaufgabe möglichst vollständig abdecken. Aufgrund seiner Flexibilität und Allgemeinheit sind die Bausteine selbst und die Beziehungen der Bausteine untereinander oft komplex. Die Anpassung verlangt ein gutes Verständnis der Funktionalität und ist daher nicht immer einfach. Deshalb ist vor jedem Einsatz zu prüfen, ob die vorliegende Funktionalität die Anforderungen des einzelnen Einsatzfalles abdeckt. Sind Anpassungen zu erwarten, sollte das Nutzen-Aufwand-Verhältnis für den aktuellen Einsatz sorgfältig geprüft werden. Jedenfalls eignet sich diese Objektbibliothek für die schnelle Erstellung von Prototypen und als Referenz für eigene Bibliotheken.

Zur automatischen Registrierung aller Bausteine steht der Assistenzbaustein *Assistant* zur Verfügung.

Alle Bausteine sind offen für individuelle Anpassungen. Beispiele zur Anwendung der Bibliotheken finden Sie in der Kategorie *Freie Bibliotheken* der Beispielmodellsammlung.

2

AGVS

Einführung

Der Objektbibliothek Plant Simulation AGVS (**A**utomated **G**uided **V**ehicle **S**ystem) ist für die Nachbildung eines **F**ahrerlosen **T**ransport**s**ystems (FTS) entwickelt worden. Um ein Fahrerloses Transportsystem zu modellieren, muß mit den bereitgestellten Wegelementen das Layout nachgebildet werden.

Die zu befördernden Teile werden von dem Transportsystem übernommen und an ein Bestimmungsort transportiert. Der Bestimmungsort kann durch

- Auftragslisten und Arbeitsplänen oder durch
- lokale Routingtabellen

erfolgen. Zur Modellierung eines Produktionssystems mit Auftragslisten und Arbeitsplänen können Sie die Objektbibliothek Plant Simulation Shop Light verwenden. Die zu befördernden Teile werden auf dem kürzesten Weg an den Bestimmungsort befördert. Es ist auch möglich mit anderen Materialflußsystemen eine Kopplung über Schnittstellen zu simulieren.

Ein Wegesystem ist einem Transportmanager *FTS Manager* zugeordnet. In einem Simulationsmodell können mehrere Wegesysteme abgebildet werden. Die Zielfindung der **F**ahrerlosen **T**ransport**f**ahrzeuge (FTF) wird vom System automatisch erledigt. Verschiedene Dispositionsregeln für die Fahrzeuge wurden bei der Implementierung der Objektbibliotheks berücksichtigt.

Zur Modellierung des Wegesystems stehen folgende Bausteine zur Verfügung:

- *AGVS Andockstation* (Übergang zu einer Bearbeitungsstation eines Produktionssystems)
- *AGVS Polygon* (Modellierung durch einen Polygonzug)
- *AGVS Gerade* (standardisiertes gerades Wegestück)
- *AGVS Kurve* (standardisiertes 90°-Kurvenstück)
- *AGVS Verzweigung* (Verzweigungen in 2 oder 3 Richtungen)
- *AGVS Zusammenführung* (Zusammenführung aus 2 oder 3 Richtungen)
- *AGVS Kreuzung* (Kreuzung von 2 Wegestücken)
- *AGVS Pulkstrecke* (Bildung von Teilegruppen)
- *AGVS Einschleußen* (Übernahme von Teilen)
- *AGVS Ausschleußen* (Abgabe von Teilen)
- *AGVS Schnittstelle* (Schnittstelle zu anderen Transportsystemen)
- *AGVS Blockstrecke* (Festlegen von Regionen, in denen nur ein Fahrzeug fahren darf.)
- *AGVS Stichstrecke* (Eintrittspunkt für ein Wegeteilsystem, auf dem die Fahrzeuge rückwärts fahren dürfen.)

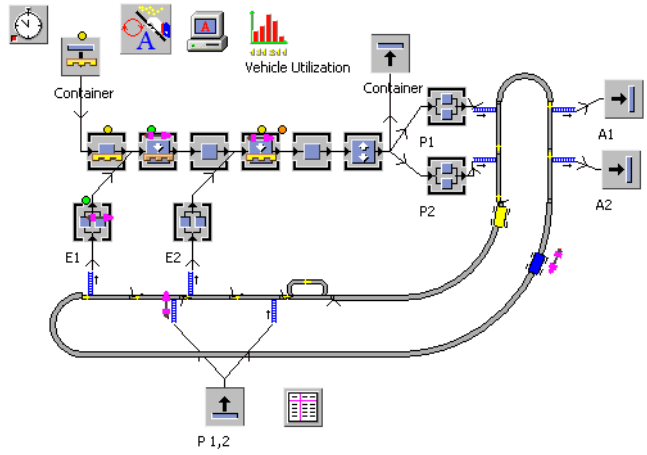
Zur automatischen Registrierung aller Bausteine steht der Assistenzbaustein *Assistant* zur Verfügung.

Beispiele

Bevor die Details genauer erklärt werden, sollen zwei Beispiele betrachtet werden.

Modell mit lokalen Routingtabellen

Das Modell *AssemblyLineWithAGVS* zeigt die Anwendung der Routingtabellen in den Bausteinen *agvs_dock_Station* und *agvs_PutIn*. Mit Routingtabellen können Sie den Materialfluss ohne eine Produktionssteuerung und ohne Methodenprogrammierung realisieren.



Das Modell *AssemblyLineWithAGVS*

Wird ein Teil in das Transportsystem eingeschleust, so wird der nächste Bestimmungsort in Abhängigkeit von dem BE-Typ oder eines freien Attributes in einer Zieletabelle festgelegt.

	string 0	object 1
string	Typ der BEs	Nächste Bearbeitungsstation
1	T1	root.E1
2	T2	root.E2
3		

Die Routingtabelle legt die nächsten Ziele der Teile fest.

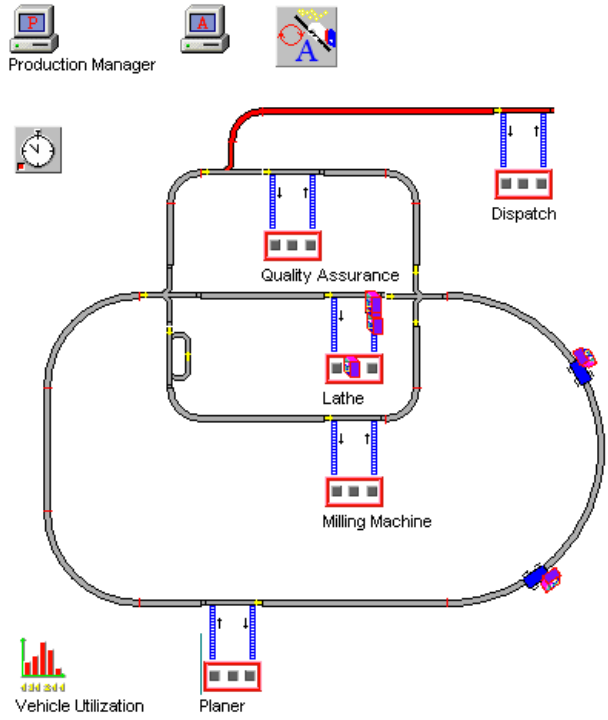
Öffnen Sie auf der Registerkarte Ziele die Routingtabellen in den Bausteinen *agvs_PutIn*.

Modell einer Fertigung mit Stückgutcharakter

Für eine Firma ist ein FTS zu entwickeln. Die Fertigung besteht aus dem Wareneingangslager, einer Drehmaschine, einem Fräswerk, der Qualitätssicherung und dem Versand. Dazu wird ein Layout erstellt, das folgendes Aussehen hat:

Zuerst wird ein leeres Netzwerk angelegt. Die zentralen Steuerung *ProductionManager* und *AGVS_manager* werden eingesetzt. Die Elemente des Transportsystems werden entsprechend dem Layout angeordnet. Zur Modellierung von Wegestücken, die aus Geraden- und Kreisbögen zusammengesetzt sind, verwenden Sie den Baustein *agvs_track*.

Tragen Sie in dem Dialog der Station den Typ der Station (z.B. *milling_machine* etc.) ein. Dann werden für jedem Stationstyp Operationen definiert, die diese Stationen ausführen können. Doppelklicken Sie dazu auf die Station und klicken im sich öffnenden Dialog auf die Schaltfläche Anzeigen neben Operationenliste.



Layout der Fertigung mit FTS

Nach dem Aufbau des Fahrkurses soll geprüft werden, ob alle Elemente miteinander verbunden sind. Nun sind die zu fertigenden Teile zu definieren und deren Arbeitspläne zu erstellen. Hierzu wird der Baustein *ProductionManager* der Fertigungssteuerung geöffnet. Nach einem Doppelklick auf den Baustein erscheint ein Dialog. Ein Klick auf die Schaltfläche Stammdaten der Teile öffnet folgende Tabelle:

string 0	string 0	table 1
0	Teiletyp	Arbeitsplan
1	part	Plan
2		

Die Tabelle *masterdataL* des Produktionsmanagers

Als nächstes wird der Typ des zu erzeugenden Teiles (in unserem Beispiel *part*) eingetragen. Durch einen Doppelklick auf den Eintrag in der Spalte *Arbeitsplan* öffnet sich eine weitere Tabelle, in der ein Arbeitsplan einzugeben ist. Dieser Arbeitsplan legt die Reihenfolge der Operationen der Maschinen fest, in der das Teil durchgeschleust werden soll.

Die Anzahl der zu erzeugenden Teile wird ebenfalls in der Fertigungssteuerung durch Anklicken der Schaltfläche Aufträge definiert. Es öffnet sich eine Tabelle, in die die entsprechenden Daten eingetragen werden. Unbedingt eingetragen werden muß der *TeileTyp* (hier: *part*) sowie die Anzahl der zu fertigenden Teile.

Zur automatischen Registrierung aller Wegelemente und Stationen wird der Assistent *ASSI* eingesetzt. Die Aktionen des Assistenten werden in der Konsole protokolliert. Nach der Benutzung des Assistenten kann dieser gelöscht werden.

string 0	string 1	time 2	time 3	real 4
	Operation	Ruestzeit	Bearbeitungszeit	Faktor
1	prepare	3.0000	2.0000	1.00
2	mill	3.0000	2.0000	1.00
3	turn	3.0000	2.0000	1.00
4	check	3.0000	2.0000	1.00
5	dispatch	3.0000	2.0000	1.00

Die Tabelle *masterdataL[1,1]*

Schließlich wird die Anzahl der gewünschten FTF in den Dialog des *AGVS_managers* eingetragen. Klicken Sie auf die Schaltflächen Reset und Start des *EreignisVerwalters*, um die FTF in Bewegung zu setzen und die Aufträge abzuarbeiten.

Der Erfassung von statistischen Werten der Fahrzeuge dient der Baustein *VehUtil* (Fahrzeugauslastung). Durch Aktivieren des Chartbausteins in *VehUtil* wird die momentane Auslastung der Fahrzeuge angezeigt.

Das oben beschriebene Beispielmodell *AGVScurves* wird durch die Methode *.Models.Transport.buildDEMO.AGVScurves* der Beispielmodellsammlung automatisch aufgebaut.

Fahrzeugdisposition

Wenn ein Teil von einer Station zu einer anderen transportiert werden soll, stellt es beim *AGVS_manager* einen entsprechenden Antrag. Für das Teil wird noch in der Station ein neues Ziel bestimmt. Anschließend wird es zur Andockstation weitergereicht. In der Andockstation angekommen, wird es beim *AGVS_manager* in die Anforderungsliste *TransportOrderL* eingetragen und versucht, einen Transportauftrag zu starten (Methode *tryToStartTransOrder*).

Ist kein FTF frei, so bleibt der Auftrag in der Liste. Meldet sich nun ein FTF frei, so überprüft es zunächst, ob sich ein Auftrag in der Liste befindet. Ist dies der Fall, so wird erneut versucht, eine Transportauftrag zu starten. Die Fahrzeuge ohne Auftrag fahren zum nächsten freien Bahnhof. Dafür wird für diese Fahrzeuge ein Platz in einem Bahnhof reserviert. Bekommt dieses Fahrzeug einen Auftrag, so wird die Reservierung wieder aufgehoben.

Die Auswahl eines frei gemeldeten FTF erfolgt nach folgenden Dispositionsregeln:

- Längste Wartezeit
- Kürzeste Wartezeit
- Kürzester Weg
- Zufällige Auswahl
- Freie Abarbeitungsfolge

Längste Wartezeit

Bei dieser Regel werden die FTF ausgewählt, die sich zuerst bei dem *AGVS_manager* frei gemeldet haben und somit am längsten keinen Auftrag hatten. Die Fahrzeuge werden also nach dem FIFO-Prinzip (First In First Out) zugeteilt.

Kürzeste Wartezeit

Hier werden die Fahrzeuge nach dem LIFO-Prinzip (Last In First Out) zugeteilt, d.h. das Fahrzeug, das sich zuletzt bei dem *AGVS_manager* freigemeldet hat, wird zuerst wieder mit einem Auftrag versehen.

Kürzester Weg

Bei Anwahl dieser Strategie wird immer das FTF, das dem neuen Ziel am nächsten steht und frei ist, mit dem Auftrag versehen.

Zufällige Auswahl

Bei der zufälligen Auswahl wird mittels einer Gleichverteilung aus allen freigemeldeten FTF das FTF ermitteln, das den Auftrag durchführen soll.

Freie Abfertigungsfolge

Für die freie Abfertigungsfolge stehen drei Möglichkeiten zur Verfügung:

1.Minimale Auslastung

- Das Fahrzeug mit der niedrigsten Auslastung wird ausgewählt.

2.Maximale Geschwindigkeit

- Das Fahrzeug mit der höchsten Geschwindigkeit wird ausgewählt.

3.Maximale Batterieladung

- Das Fahrzeug mit der höchsten Batterieladung wird ausgewählt.

Auftragsdisposition

Meldet sich ein Fahrzeug frei und stehen mehrere Fahraufträge an, so kann auch hier anhand der eingestellten Dispositionsregeln eine entsprechende Zuteilung erfolgen.

Längste Wartezeit

Der Auftrag mit der längsten Wartezeit wird dem FTF zugewiesen. Diese Art der Auftragsdisposition entspricht dem FIFO-Prinzip.

Kürzester Weg noch nicht implementiert

Es wird der Auftrag zugewiesen, der die kürzeste Entfernung zum Fahrzeugstandort hat, d.h. zu dem das Fahrzeug am nächsten steht.

Zufällige Auswahl

Aus der Liste der Aufträge wird mittels Gleichverteilung ein Auftrag ausgewählt.

Beschreibung der Bausteine

Bei der Beschreibung der einzelnen Bausteine wird kurz die Funktionalität mit der Benutzerschnittstelle erläutert. Bausteine, die sich in der Funktionalität nicht oder nur geringfügig unterscheiden, sind zusammengefaßt.

Zu jedem Element existiert für die Änderung von Parametern ein Dialogfeld. Alle längenorientierte Elemente enthalten ein Eingabefeld für die Länge des entsprechenden Bausteins.

Wegbausteine

Wegbausteine werden von FTF befahren. Zu jedem zusammenhängenden Wegsysteme muß ein *AGVS_manager* eingesetzt werden. Der Dialog der Wegbausteine enthält deshalb ein Eingabeelement für den verantwortlichen AGVS manager. Dieser Manager verwaltet die Fahrzeuge, die das zugehörige Wegsystem befahren können.

Der Baustein AGVS_track

Symbol: 

Funktionalität:

Sie können einen Weg für die Fahrzeuge modellieren, der aus Strecken und Kreisbögen besteht.

Benutzerschnittstelle:

Dialog der Objekts *AGVS_track*

Die Bausteine AGVS_straight und AGVS_bend

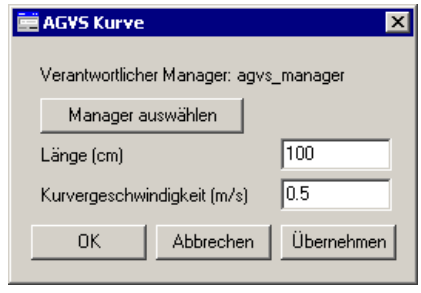
Symbole: 

Mit diesen Bausteinen werden Geraden bzw. 90°-Kurven im System abgebildet.

Funktionalität:

Ein ankommendes FTF wird entsprechend dem Grundverhalten umgelagert.

Benutzerschnittstelle:



Dialog des Bausteins *AGVS_bend*

Die Verzweigungen AGVS_br2 und AGVS_br3

Symbole: 

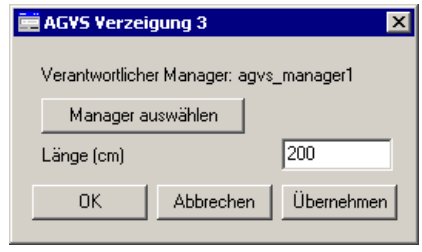
Funktionalität:

Mit diesem Bausteinen werden Verzweigungen in zwei oder drei Richtungen abgebildet. In den Verzweigungen wird mit Hilfe der Wegweiserlisten *rgSign** erkannt, in welche Richtung das FTF umgelagert werden soll.

Die Wegweiserlisten müssen entweder manuell oder unter Verwendung des Assistenten ausgefüllt werden.

Ein ankommendes FTF durchsucht die Wegweiserlisten nach seinem Ziel. Ist das Ziel gefunden, wird das FTF in die entsprechende Richtung umgelagert. Wird das Ziel nicht gefunden, so fährt das FTF geradeaus weiter.

Benutzerschnittstelle:



Dialog der Bausteine *AGVS_br**

Bei der Eingabe der Länge des Bausteins wird davon ausgegangen, daß die Verzweigung symmetrisch aufgebaut ist.

Die Zusammenführungen AGVS_junction2 und AGVS_junction3

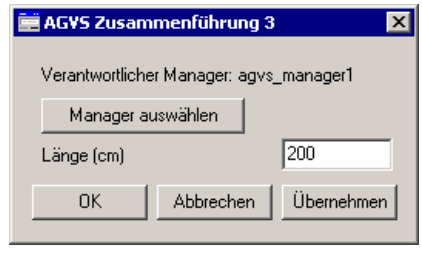
Symbole: 

Funktionalität:

Diese Bausteine dienen zur Nachbildung von Zusammenführungen aus zwei oder drei Richtungen.

Das ankommende FTF wird entsprechend dem Grundverhalten umgelagert. Priorität hat dabei immer das FTF, das zuerst in den Baustein eintritt.

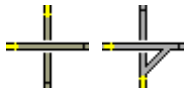
Benutzerschnittstelle:



Dialog der Bausteine *AGVS_junction**

Bei der Eingabe der Länge des Bausteins wird davon ausgegangen, daß die Zusammenführungen symmetrisch aufgebaut sind.

Die Kreuzungen AGVS_crossing und AGVS_cross

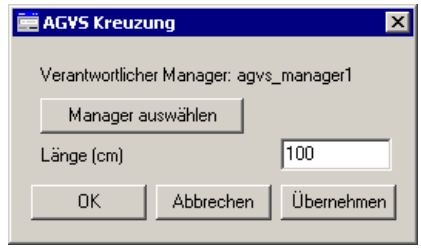
Symbole: 

Funktionalität:

Die Bausteine dienen der Modellierung von Kreuzungen. In einem Baustein *AGVS_crossing* kann ein ankommendes FTF immer nur geradeaus umgelagert werden, d.h. es ist nicht möglich innerhalb einer Kreuzung zusätzlich eine Kurve nachzubilden. Der Baustein *AGVS_cross* ermöglicht eine Rechtskurve in einer Fahrrichtung.

Ein eintreffendes Fahrzeug blockiert die Kreuzung, so daß diese für die andere Richtung gesperrt ist. Fahrzeuge können also nicht zusammenstoßen.

Benutzerschnittstelle:



Dialog des Bausteins *AGVS_crossing*

Bei der Eingabe der Länge des Bausteins wird davon ausgegangen, daß die Kreuzung symmetrisch aufgebaut ist.

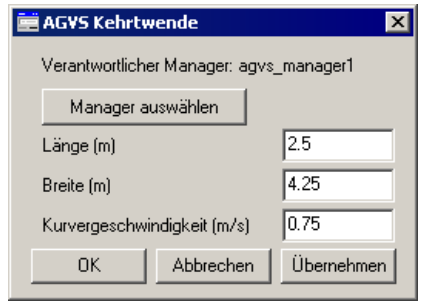
Die Kehrtwende AGVS_U_turn

Symbol: 

Funktionalität:

Der Baustein dient der Modellierung von Kehrtwenden. Der Baustein darf nicht in einer Stichstecke vorkommen, d.h. die FTF befahren den Baustein nur in den auf dem Icon angezeigten Richtungen. Es kann nur ein FTF auf dem Baustein *AGVS_U_turn* umkehren. Aus den oberen und unteren Wegen können sich die FTF nicht berühren, so daß sich auf diesen Wegen mehrere FTF befinden können.

Benutzerschnittstelle:



Dialog des Bausteins *AGVS_U_turn*

Schnittstelle zu anderen Transportsystemen

AGVS_interface

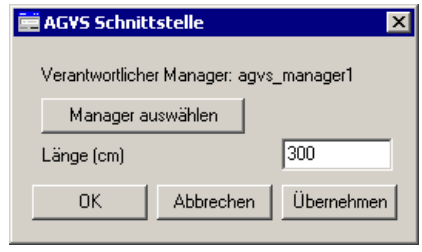
Symbol: 

Funktionalität:

Mittels einer Schnittstelle kann ein FTS mit anderen FTS oder Materialflußsystemen, z. B. Elektrohängebahn, Stetigförderer, etc. gekoppelt werden. Auch eine Schnittstelle stellt für ein FTF ein Ziel dar. Die Schnittstelle darf nicht als Ersatz für eine Andockstation eingesetzt werden. Auch wenn sich beide ähnlich sind, so sind die Funktionalitäten grundsätzlich verschieden.

Es wird überprüft, ob das Ziel eines ankommenden FTF die Schnittstelle ist. Ist das der Fall, so wird das FTF entweder be- oder entladen und anschließend zum Nachfolger umgelagert. Ein abgeladenes Teil wird zur Partnerschnittstelle weitergereicht und löst dort einen Fahrauftrag aus.

Benutzerschnittstelle:



Dialog des Bausteins *AGVS_interface*

Andere Stationen

Bahnhof AGVS_station

Symbol: 

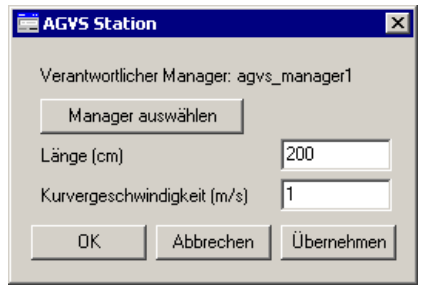
Funktionalität:

Der Baustein AGVS_station dient zur Abbildung von Bahnhöfen, wobei mehrere Bahnhöfe in einem System verwendet werden können. In einem System darf die Anzahl der verwendeten FTF (vgl. Dialog des AGVS_managers) nicht größer als die Gesamtkapazität aller Bahnhöfe sein.

Hat ein Fahrzeug als Ziel einen Bahnhof, so wird das FTF aus dem Fahrkurs ausgeschleust, in ein Wartegleis umgelagert und als frei gemeldet. An FTF, die hinter einem anderen FTF auf dem Wartegleis stehen, kann kein Transportauftrag vergeben werden. Zusätzlich werden eventuell gesperrte Blockstrecken wieder freigegeben.

Hat ein FTF im Bahnhof einen Auftrag zugeteilt bekommen, so fährt es aus dem Wartegleis auf die Strecke, ohne zu überprüfen, ob eine eventuell gesperrte Blockstrecke befahren wird bzw. ob der Platz nicht durch ein anderes FTF belegt ist.

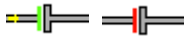
Benutzerschnittstelle:



Dialog des Bausteins *AGVS_station*

Blockstreckensteuerung AGVS_blocklineCtrl

Symbole:



Funktionalität:

Werden Blockstrecken in einem Teilsystem eingesetzt, so wird dieses Wegsystem in Bereiche (Regionen) unterteilt. In jedem Bereich darf sich nur ein FTF aufhalten.

Hinweis: Blockstrecken dürfen nicht zusammen mit Stichstrecken verwendet werden.

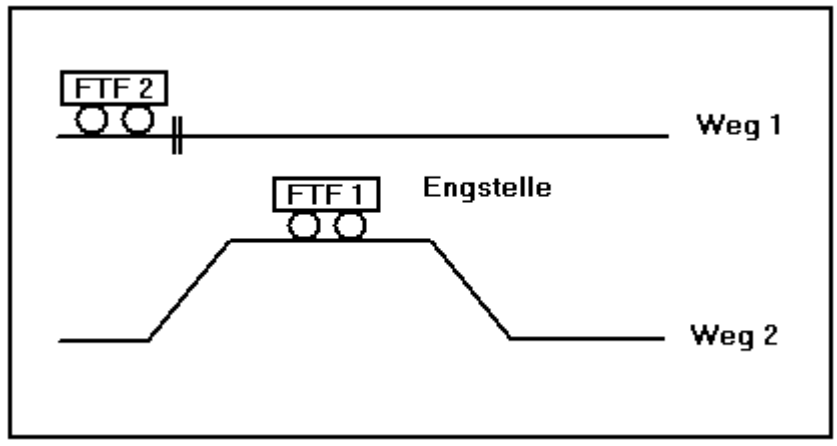
Jeder Bereich kann über mehrere Blockstrecken befahren werden. Wird ein solcher Baustein überfahren, so werden alle Eingänge des Bereiches gesperrt (rot dargestellt) und die gesperrten Eingänge des zurückliegenden Bereiches werden geöffnet (grün dargestellt). Die Erkennung der Regionen kann mit Hilfe des Assistenten *ASSI* erfolgen.

Benutzerschnittstelle:

Dialog des Bausteins *AGVS_blocklineCtrl*

Beim Einsatz von Blockstrecken kann es durch ungünstige Verteilung im System zu sogenannten Deadlocks (Blokkierungen) kommen, d.h. die Fahrzeuge behindern sich gegenseitig, bis schließlich keine Bewegung mehr möglich ist. Dies ist z.B. der Fall, wenn die Anzahl der FTF größer oder gleich der Anzahl der Blockstreckenelemente ist.

Es ist auch möglich, Engstellen innerhalb eines Systems zu definieren. Dazu müssen jedoch im System Blockstrekken eingesetzt sein.



Erklärung einer Engstelle

Unter einer Engstelle ist eine Stelle zu verstehen, an der der Abstand zweier Wege so gering ist, daß zwei FTF einander nicht passieren können. Es muß immer eine der beiden Strecken gesperrt sein.

Verzweigung in Stichstrecke AGVS_tl_br

Symbol:

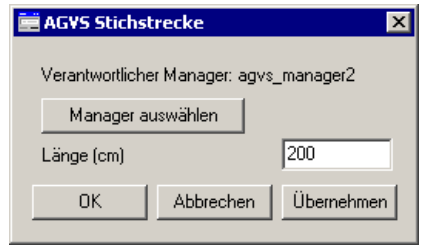
Funktionalität:

Um Stichstrecken im System abzubilden, ist es notwendig, einen Baustein anzulegen, der den Anfang einer solchen Stichstrecke darstellt. An diesem Baustein wird der verzweigende Pfad automatisch als Stichstrecke definiert. Die Wegelemente einer Stichstrecke werden durch den Assistent ASSI automatisch erkannt (siehe benutzerdefiniertes Attribute *termLineBr* in jedem Wegelement).

Als äußeres Zeichen dafür, daß diese Meldelinie richtig gelaufen ist, werden alle Wegelemente der Stichstrecke rot markiert.

Ein ankommendes FTF wird auf sein Ziel hin überprüft. Liegt das Ziel nicht in der Stichstrecke, so wird das FTF geradeaus umgelagert. Anderenfalls wird geprüft, ob ein Eintritt in die Stichstrecke möglich ist oder ob sich schon ein FTF in Stichstrecke aufhält. Ist die Stichstrecke belegt, so fährt das FTF geradeaus weiter.

Benutzerschnittstelle:



Dialog des Bausteins *AGVS_tl_br*

Bei der Eingabe der Länge des Bausteins wird davon ausgegangen, daß die Verzweigung symmetrisch aufgebaut ist.

Andockstation AGVS_dock_station

Symbol:

Funktionalität:

Die Andockstation und die Ausschleusestelle *agvs_PutOut* sind anfahrbare Ziele für ein FTF.

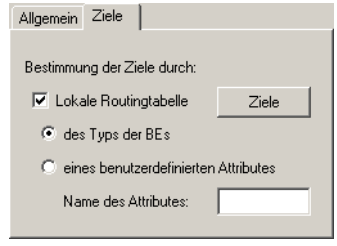
Die Andockstation kann ein Anschlußbaustein für eine *Station* eines Produktionssystems darstellen. Eine Station kann ein Netzwerk oder eine Folge von Grundbausteinen sein. In der Station muss sich ein Grundbausteine befinden, der eine Bearbeitung der Teile ausführen kann. Der Baustein darf nur mit einer Bearbeitungsstation, die Teile aufnimmt und nach der Bearbeitung wieder abgibt, verbunden werden. Das sind die Bausteine Einzelstation, Parallelstation, Senke, Assembly und Disassembly. Die Andockstation ist mit einer Station verbunden, die das Teil als Ziel hat. Die Zuordnung Andockstation - Station wird im *AGVS_manager* realisiert. Dort erfolgt die Umsetzung von Teileziel zu Fahrzeugziel. Die Andockstation kann für normale Strecken und für Stichstrecken eingesetzt werden.

Es wird überprüft, ob das Ziel eines ankommenden FTF die Andockstation ist. Ist dies der Fall, so wird das FTF entweder be- oder entladen und anschließend zum Nachfolger umgelagert. Ein abgeladenes Teil wird zur angeschlossenen Station weitergereicht und dort bearbeitet.

Benutzerschnittstelle:

Die Registerkarte Allgemein des Dialogs des Bausteins *AGVS_dock_station*

Wenn das nächste Ziel nicht durch ein Produktionssystem dem Teil zugeordnet wird, so können Sie die nächsten Ziele durch lokale Routingtabellen definieren. Auf der Registerkarte Ziele könne Sie festlegen, ob das nächste Ziel vom Teiletyp oder von den Wert eines beliebigen freien Attributes abhängt. Der Typ ist durch den Wert des freien Attributes *entityType* (string) bestimmt. Hat das Teil nicht ein solches Attribut, so wird der Typ des Teils aus dem Namen bestimmt.



Die Registerkarte Ziele des Dialogs des Bausteins *AGVS_dock_station*

Durch die Schaltfläche Ziele öffnen Sie die Routingtabelle, in der Sie die Zuordnung der Teile zu den Zielen festlegen können.

Einschleusestelle AGVS_PutIn

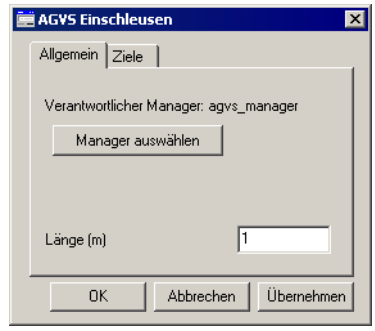
Symbol:

Funktionalität:

Zu bearbeitende Teile (Objekt *part*) werden über diesen Baustein an das Transportsystem übergeben. Das Teil löst einen Transportauftrag aus. Dieses Objekt kann z.B. mit einer Quelle verbunden werden.

Bei einem ankommendes FTF wird überprüft, ob der Transportauftrag mit dem Teil überein. Wenn ja, so wird das Teil auf das FTF aufgeladen. Die Funktionsweise der lokalen Routingtabellen ist im Abschnitt über die Andockstation beschrieben.

Benutzerschnittstelle:



Dialog des Bausteins *AGVS_PutIn*

Ausschleusestelle AGVS_PutOut

Symbol:

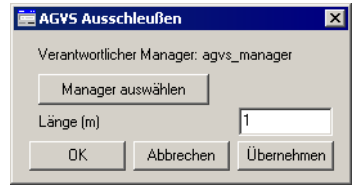
Funktionalität:

Wie die Andockstation ist die Ausschleusestelle *agvs_PutOut* ein anfahrbare Ziele für ein FTF.

Dieses Objekt kann z.B. mit einer Senke verbunden werden.

Ein ankommendes und beladenes FTF wird auf sein Ziel hin überprüft und gegebenenfalls entladen. Die Ausschleusestelle ist mit einem Objekt verbunden, das das Teil als Ziel hat. Die Zuordnung Ausschleusestelle - Teileziel wird im *AGVS_manager* realisiert.

Benutzerschnittstelle:



Dialog des Bausteins *AGVS_PutOut*

Transportmanager

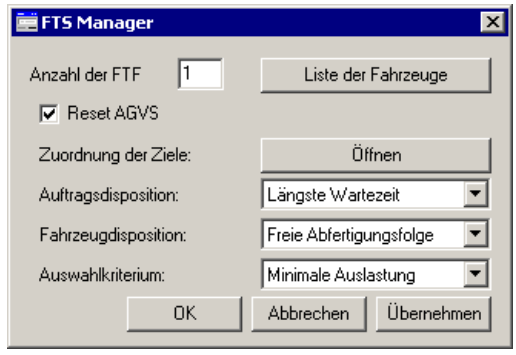
Das Objekt AGVS_manager

Symbol: 

Funktionalität:

Der *AGVS_manager* verwaltet alle Wegelemente eines zusammenhängenden Teilsystems eines Modells und die FTF, die dieses Teilsystem befahren können. Die Kartei *rgAnnounce* enthält diese Wegelemente. Die Liste *rgAssignL* enthält die Zuweisungen von Zielen (Andockstationen, Einschleusestationen, Schnittstellenbausteine) der FTF zu den (Bearbeitungs-) Stationen. Diese Liste kann durch eine entsprechende Schaltfläche geöffnet werden. Weiterhin sind alle im System verwendeten Bahnhöfe registriert. Die genannten Registrierungen können automatisch mit Hilfe des Assistenten *ASSI* durchgeführt werden.

Benutzerschnittstelle:



Dialog des Bausteins *AGVS_manager*

Der Dialog des *AGVS_manager* ermöglicht die gewünschte Anzahl der FTF einzustellen und eine Liste der verwendeten FTF anzuzeigen. Ferner kann definiert werden, was bei einem Reset passieren soll. Ist das Kontrollkästchen aktiviert, so werden bei einem Reset sämtliche FTF auf allen Elementen gelöscht. Dies sollte die Standardeinstellung sein. Nicht wünschenswert ist dies, wenn z.B. der Wiederanlauf nach einem Systemausfall simuliert werden soll. Dann sollten die FTF dort sein, wo sie vor dem Ausfall waren. Hierzu muß das Kontrollkästchen deaktiviert werden.

Statistischen Auswertungen

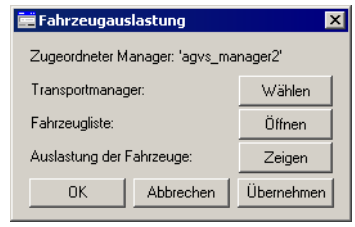
Das Objekt VehUtil

Symbol: 

Funktionalität:

Die Berechnung und Darstellung von statistischen Werten der Fahrzeuge wird mit diesem Baustein durchgeführt. Der Baustein muß im selben Netzwerk wie der Manager *AGVS_manager* eingesetzt sein.

Benutzerschnittstelle:



Dialog des Bausteins *VehUtil*

Bewegliche Elemente

Fahrzeug agv

Symbole:

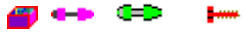
Das Objekt *agv* bildet das Fahrerlose Transportfahrzeug (FTF) im System ab.

Beim Öffnen eines FTF erscheint ein Dialogfenster, in dem die Geschwindigkeit eingetragen wird.

- Dem FTF sind folgende Attribute beigefügt:
  - *vd_property*: Dieses Attribut wird für die Fahrzeugdisposition im Bahnhof benötigt. Ist der Wert "inQueue", so bedeutet das, daß das Fahrzeug in den Bahnhof eingefahren ist, aber nicht an erster Stelle steht, d.h. ihm kann kein Auftrag zugewiesen werden, bis er an erster Stelle steht.
  - *transportOrder*: Das zu transportierende Teil part wird gespeichert. Dadurch ist sichergestellt, da jedes FTF den richtigen Fahrauftrag transportiert. Datentyp object.
  - *origSpeed* wird verwendet, um nach einer Kurve *agvs_bend* oder *agvs_track* die Geschwindigkeit des Fahrzeug auf den ursprünglichen Wert zu setzen Datentyp†speed.

Teil part

Symbole:



Der Baustein *part* wird bearbeitet und von den FTF Bausteinen transportiert.

- Der Baustein part ist mit folgenden Attributen ausgestattet:
  - *globDestination*: Dieses Attribut enthält das Ziel des Teils, wie z.B. eine Bearbeitungsstation (Datentyp object).
  - *locDestination*: Der Pfad zum Zieles des Fahrzeuges, wie z.B. eine Andockstation. (Datentyp object).
  - *EntityType*: Dieses optionale Attribut enthält den Teiletyp. Anhand dieses Teiletyps wird das nächste Ziel bestimmt (Datentyp string).

3

Conveyor

Einführung

Mit der Objektbibliothek Plant Simulation Conveyor können Stetigfördersysteme, wie Transportbänder oder Rollenbahnen, abgebildet werden. Um eine Fertigung mit einem Stetigfördersystem zu modellieren, muß mit den bereitgestellten Wegelementen das Layout nachgebildet werden. Notwendige technische Parameter und Steuerungsregeln können einfach in Dialogen festgelegt werden.

Die zu befördernden Teile werden von dem Transportsystem übernommen und an ein Bestimmungsort transportiert. Der Bestimmungsort kann durch

- Auftragslisten und Arbeitsplänen oder durch
- lokale Routingtabellen

erfolgen. Zur Modellierung eines Produktionssystems mit Auftragslisten und Arbeitsplänen können Sie die Objektbibliothek Plant Simulation Shop Light verwenden. Die zu befördernden Teile werden auf dem kürzesten Weg an den Bestimmungsort befördert. Die Objektbibliothek Plant Simulation Conveyor kann mit anderen Bausteinkästen zur Modellierung von Materialflußsystemen, wie zum Beispiel dem Fahrerlosen Transportsystem Plant Simulation AGVS, kombiniert werden.

Ein geschlossenes Wegesystem ist einem Transportmanager *Conveyor Manager* zugeordnet. In einem Simulationsmodell können mehrere Wegesysteme abgebildet werden.

Zur Modellierung des Wegesystems stehen folgende Bausteine zur Verfügung:

- *Conveyor Andockstation* (Übergang zu einer Bearbeitungsstation eines Produktionssystems)
- *Conveyor Weg* (Modellierung durch einen Polygonzug)
- *Conveyor Gerade* (standardisiertes gerades Wegestück)
- *Conveyor Kurve* (standardisiertes 90°-Kurvenstück)
- *Conveyor Verzweigung* (Verzweigungen in 2 oder 3 Richtungen)
- *Conveyor Zusammenführung* (Zusammenführung aus 2 oder 3 Richtungen)
- *Conveyor Kreuzung* (Kreuzung von 2 Wegestücken)
- *Conveyor Pulkstrecke* (Bildung von Teilegruppen)
- *Conveyor Einschleußen* (Übernahme von Teilen)
- *Conveyor Ausschleußen* (Abgabe von Teilen)
- *Conveyor Schnittstelle* (Schnittstelle zu anderen Transportsystemen)

Zur automatischen Registrierung aller Bausteine steht der Assistenzbaustein *Assistant* zur Verfügung.

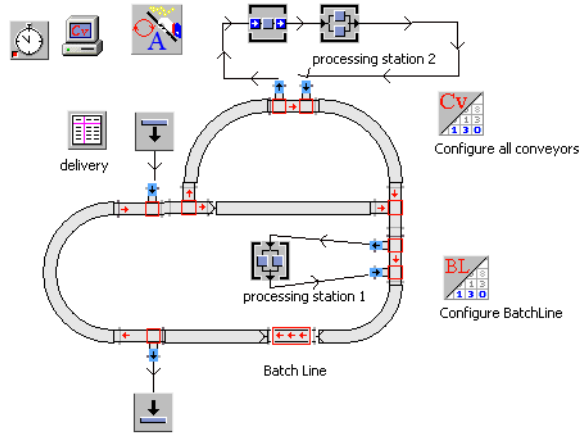
Alle Bausteine sind offen für individuelle Anpassungen. Beispiele zur Anwendung der Objektbibliothek finden Sie in der Kategorie *Freie Bibliotheken* der Beispielkollektion.

Beispiele

Die Kategorie *Freie Bibliotheken* der Beispielmodellsammlung enthält die beschriebenen Beispielmodelle.

Einfache Anwendung

Das Modell *ConvSys1* zeigt eine Werkstatt, bei der die Reihenfolge der Bearbeitungsschritte in den Bearbeitungsstationen *ProcStation* fest eingestellt ist. Der nächste Bestimmungsort eines Teils wird in dem Attribut *globDestination* festgelegt. Die Erstellung dieses Modells wird hier kurz beschrieben.



Einfache Anwendung der Objektbibliothek Plant Simulation Conveyor

Legen Sie zuerst Sie ein leeres Netzwerk an. Ein Stetigförderersystem besteht immer aus einer Zentrale *cv_Manager* und mehreren Wegelementen. Die Wegelemente werden entsprechend dem Layout angeordnet. Es ist für die Laufzeit der Modelle vorteilhaft, das Wegesystem mit einer möglichst geringen Anzahl von Bausteinen zu realisieren.

In dem Quellenbaustein *Source* ist im Pop-up Menü Erzeugungszeitpunkte *Lieferliste* eingestellt.

delivery time	MU	number	name	attributes
1:00.0000	.MUs.part	3	wave	attr
1:10.0000	.MUs.part	4	screw	attr

Quelle mit Lieferliste *delivery*

In der Lieferliste wird in der Attributliste (5. Spalte der Lieferliste) das Attribut *globDestination* angelegt und gesetzt.

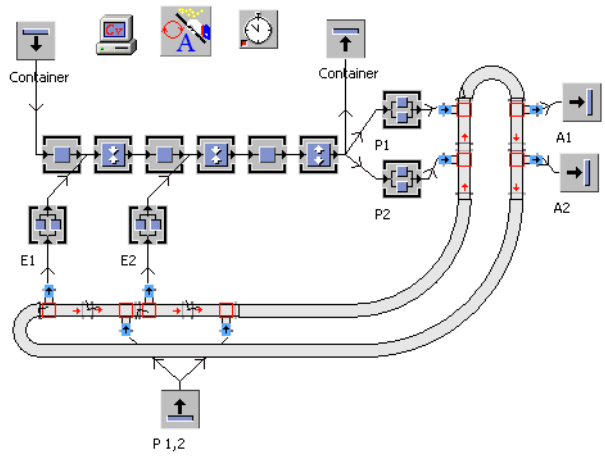
Die Bearbeitung der Teile erfolgt auf den Parallelstationen. Diese Parallelstatioen besitzen eine Ausgangssteuerung als freies Attribut, in der das nächste Ziel dem Teil übergeben wird:

```
is
do
      @.globDestination := drain;
      @.move;
end;
```

Nun können Sie die Simulation starten. Achten Sie beim Passieren des Verzweigungsbausteins *cv_br2* darauf, dass die Teile tatsächlich den kürzesten Weg zu den Zielen nehmen. Nach dem ersten Bearbeitungsschritt gelangen die Teile in den Baustein *cv_batchline*. Dort werden Gruppen von Teilen gebildet.

Modell mit lokalen Routingtabellen

Das Modell *AssemblyLineWithConv* zeigt die Anwendung der Routingtabellen in den Bausteinen *cv_dockingStation* und *cv_PutIn*. Mit Routingtabellen können Sie den Materialfluss ohne eine Produktionssteuerung und ohne Methodenprogrammierung realisieren.



Das Modell *AssemblyLineWithConv*

Wird ein Teil in das Conveyorsystem eingeschleust, so wird der nächste Bestimmungsort in Abhängigkeit von dem BE-Typ oder eines freien Attributes in einer Zieletabelle festgelegt.

	string 0	object 1
string	Typ der BEs	Nächste Bearbeitungsstation
1	T1	root.E1
2	T2	root.E2
3		

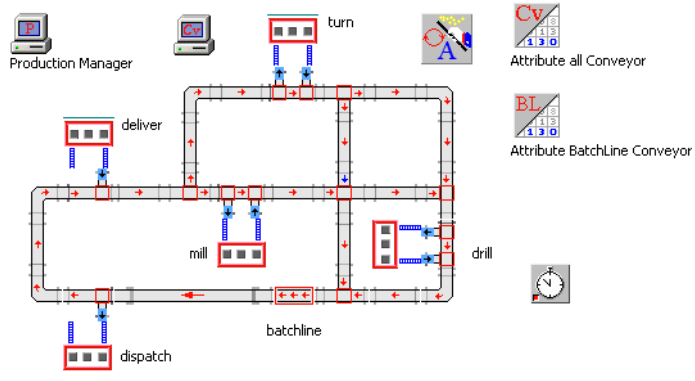
Die Routingtabelle legt die nächsten Ziele der Teile fest.

Öffnen Sie auf der Registerkarte Ziele die Routingtabellen in den Bausteinen *cv_PutIn*.

Modell einer Fertigung mit Stückgutcharakter

Werden in einer Produktion verschiedene Teile mit individuellen Arbeitsplänen und Auftragslisten hergestellt, so muß der Materialfluß flexibel gestaltet werden. Die Zuweisung der Ziele der Teile kann nicht wie in den beiden ersten Beispielen fest in das Modell programmiert werden. In diesem Abschnitt werden wir von einer konkreten Produktionsanlage zur Herstellung von mehreren Teilen ausgehen und das Modell schrittweise entwickeln.

Stellen Sie sich vor, Sie sollen für eine Firma ein Stetigfördersystem entwickeln. Die Fertigung besteht aus dem Wareneingang, einer Dreh-, Fräs- und Bohrmaschine und dem Versand. Dazu wird ein Layout erstellt, das folgendes Aussehen hat:



Layout der Fertigung mit Stetigfördersystem

Zuerst wird ein leeres Netzwerk angelegt. Die Zentralen *ProductionManager* und *cv_Manager* werden eingesetzt. Beim Einsetzen einer Station öffnet sich ein Dialog. Dort wird direkt nach dem Stationstyp gefragt. Der Typ der Station muss eingetragen werden. Dann werden für jeden Stationstyp Operationen definiert, die diese Stationen ausführen können. Doppelklicken Sie auf die Station und klicken im sich öffnenden Dialog auf die Schaltfläche Öffnen neben Ausführbare Operationen.

Nach dem Aufbau des Kurses soll geprüft werden, ob alle Elemente miteinander verbunden sind. Nun sind die zu fertigenden Teile zu definieren und deren Arbeitspläne zu erstellen. Hierzu wird der Baustein *ProductionManager* der Fertigungssteuerung geöffnet. Nach einem Doppelklick auf den Baustein erscheint ein Dialog. Ein Klick auf die Schaltfläche Stammdaten der Teile öffnet die folgende Tabelle:

	string 0	table 1
string	Teiletyp	Arbeitsplan
1	screw	Plan
2	wave	Plan

Die Tabelle *masterdataL* des Fertigungssteuerung *ProductionManager*

Als nächstes wird der Typ des zu erzeugenden Teiles (in unserem Beispiel *Part*) eingetragen. Durch einen Doppelklick auf den Eintrag in der Spalte *Arbeitsplan* öffnet sich eine weitere Tabelle, in die ein Arbeitsplan einzugeben ist. Dieser Arbeitsplan legt die Reihenfolge der Operationen der Maschinen fest, in der das Teil durchgeschleust werden soll.

	string 1	time 2	time 3	real 4
string	Operation	Ruestzeit	Bearbeitungszeit	Faktor
1	deliver	5:00.0000	1:00.0000	1.00
2	drill	5:00.0000	1:00.0000	1.00
3	turn	5:00.0000	1:00.0000	1.00
4	mill	5:00.0000	1:00.0000	1.00
5	dispatch	5:00.0000	1:00.0000	1.00

Der Arbeitsplan *masterdataL[1,1]*

Die Anzahl der zu erzeugenden Teile wird ebenfalls in der Fertigungssteuerung durch Anklicken der Schaltfläche Aufträge definiert. Es öffnet sich eine Tabelle, in die die entsprechenden Daten eingetragen werden. Unbedingt eingetragen werden muss der *Teiletyp* (hier: *screw* und *wave*) sowie die Anzahl der zu fertigenden Teile.

Die Modellierung der beschriebenen Fertigung ohne Transportsystem ist im Netzwerk *workshop* enthalten, das durch die Methode *buildDEMOS.buildWorkshop* aufgebaut wird. Die Funktionstüchtigkeit der Fertigung kann separat getestet werden.

Beschreibung der Bausteine

Bei der Beschreibung der einzelnen Bausteine werden die Funktionalität und die Benutzerschnittstelle erläutert. Bausteine, die sich in der Funktionalität nicht oder nur geringfügig unterscheiden, sind zusammengefaßt. Zu jedem Baustein existiert für die Änderung von Parametern ein Dialog. Mit den Schaltflächen OK oder Anwenden werden die Einstellungen übernommen. Wird eine Länge eines Wegbausteins verändert, wird eine Registrierung mit dem Assistenten ist notwendig. Über den Kontextmenüeintrag Öffnen wird das Netzwerk des jeweiligen Bausteins geöffnet.

Wegbausteine

Die folgenden Bausteine dienen zur Abbildung von Wegelementen, die *nicht* mit einem Hub- oder Drehtisch ausgerüstet sind.

Baustein cv_Track

Symbol: 

Funktionalität:

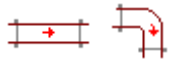
Dieses Objekt beschreibt einen Weg des Stetigförders. Jede Komponente des Wegesystems muß einem Transportmanager *cv_manager* zugeordnet sein. Deshalb enthält der Dialog der Wegebausteine ein Auswahlelement für den zuständigen Transportmanager, der die zu transportierenden Teile verwaltet.

Benutzerschnittstelle:

Sie können ein Wegstück modellieren, das aus Strecken und Kreisbögen besteht.

Der Dialog des Bausteins *cv_track*

Die Bausteine cv_straight und cv_bend

Symbole: 

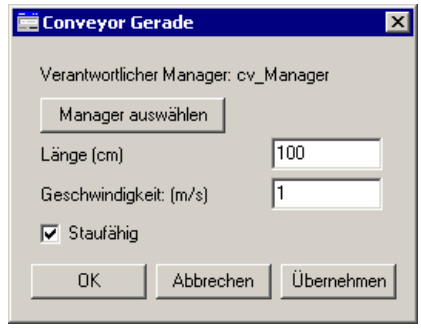
Funktionalität:

Die beiden Bausteine stellen ein gerades bzw. gebogenes Teilstück des Materialflußsystems dar.

Sie sollten aus Laufzeitgründen vermeiden, mehrere Bausteine *cv_straight* aufeinander folgen zu lassen. Besser ist es, das Bild des Bausteins mit der linken Maustaste und gedrückter Umschalt-Strg-Taste in die Länge zu ziehen und die Länge des Bausteins über das Dialogfenster entsprechend einzustellen.

Benutzerschnittstelle:

Bei den Bausteinen kann die Länge und die Transportgeschwindigkeit in Eingabefeldern festgelegt werden. Über ein Pop-up Menü kann eingestellt werden, ob die Bausteine staufähig sind. Der Entfernungsberechnung dient die eingebaute Meldelinie. Wird z.B. die Länge verändert, so wird bei der nächsten Initialisierung eine neue Entfernungsberechnung angestoßen.

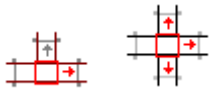


Dialog des Bausteins *cv_straight*

Wegbausteine mit Hub- oder Drehtisch

Der Hub- oder Drehtisch ist in verschiedenen Wegbausteinen eingebaut, z. B. in den Bausteinen *cv_dockingStation* (Verbindungsbaustein zu einer Bearbeitungsstation *station*) und *cv_interface* (Schnittstelle verschiedener Transportsysteme). Ist ein Teil vollständig auf den Tisch gefahren, so wird der Tisch bewegt. Dieser Vorgang benötigt die *Positionierzeit*. Nach der Bewegung kann das Teil vom Tisch fahren. Die Geschwindigkeit beim Ausfahren wird durch die eingestellte Geschwindigkeit des Bausteins, auf den das Teil umgelagert wird, bestimmt. Ist das Teil vollständig vom Tisch gefahren, wird der Tisch zurückbewegt. Dieser Vorgang benötigt die *Rückstellzeit*. Nach der *Rückstellzeit* kann das nächste Teil auf den Tisch auffahren. Der Tisch funktioniert nur dann korrekt, wenn die Länge des Tisches mindestens so groß ist wie die Länge des Teils und die mit dem Anwenderbaustein verbundenen Nachfolger Grundbausteine vom Typ *Förderstrecke* sind!

Die Verzweigungen cv_br2 und cv_br3

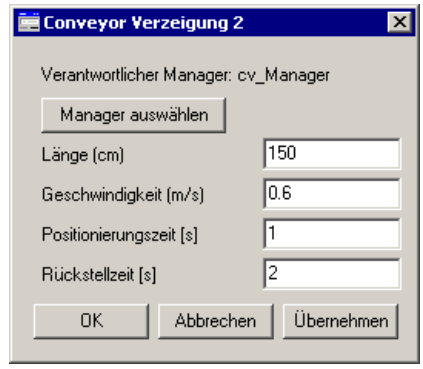
Symbole: 

Funktionalität:

Mit der Verzweigung kann ein Hub- oder Drehtisch abgebildet werden, der ein Teil auf zwei Förderstrecken verteilen kann. Ein Teil kann in Bewegungsrichtung oder zur Seite befördert werden. Soll ein Teil zur Seite befördert werden, wird der Tisch gedreht. Nachdem das Teil heruntergefahren ist, wird der Tisch automatisch zurückgestellt. An die Verzweigung kann keine Station angeschlossen werden. Benutzen Sie hierfür die Bausteine *cv_PutOut* und *cv_dockingStation*.

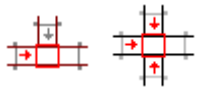
Benutzerschnittstelle:

Die Länge, die Geschwindigkeit beim Einfahren auf den Tisch, die Positionierzeit und die Rückstellzeit des Tisches können eingestellt werden.



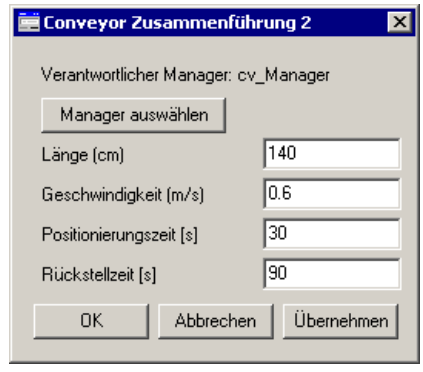
Dialog des Bausteins *cv_br2*

Die Zusammenführungen cv_junction2 und cv_junction3

Symbole: 

Funktionalität:

Der Baustein stellt einen Hub- oder Drehtisch dar, der Teile aus einem Seitenstrang einschleust. Die Zusammenführung kann zwei Förderstrecken auf eine Förderstrecke zusammenführen. Eine Förderstrecke ist in der Bewegungsrichtung und die andere seitlich versetzt angeordnet. Soll ein Teil von der Seite weiterbefördert werden, wird der Tisch zuerst gedreht. Dann fährt das Teil auf den Tisch und der Tisch dreht zurück. Das Teil kann nach der Drehung von dem Tisch herunterfahren.



Dialog des Bausteins *cv_junction2*

Benutzerschnittstelle:

Die Länge, die Geschwindigkeit beim Einfahren auf den Tisch, die Positionierzeit und die Rückstellzeit des Tisches sind einstellbar. Der Dialog dieses Bausteins hat die gleichen Eingabemöglichkeiten wie *cv_Br2*. An die Zusammenführung kann keine Station angeschlossen werden. Für diesen Zweck gibt es den *cv_PutIn*- bzw. den Baustein *cv_dockingStation*.

Die Kreuzung cv_crossing

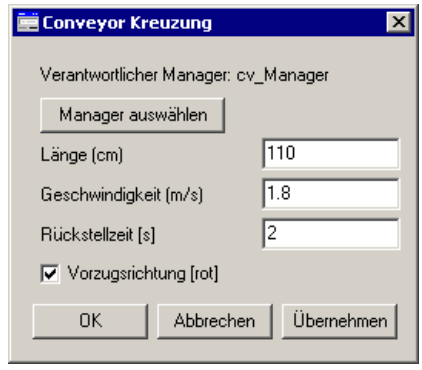
Symbol: 

Funktionalität:

Der Baustein stellt einen Hub- oder Drehtisch dar, der als Kreuzungselement zweier Förderstrecken benutzt werden kann. Ein Abbiegen auf der Kreuzung ist nicht möglich. Will ein Teil in den Tisch eintreten, entscheidet die Ruheposition des Tisches *(Bevorzugte Richtung)* und die Position des Teiles, ob der Tisch bewegt werden muß. Sind beide Positionen identisch, muß sich der Tisch nicht bewegen. Sind die Positionen verschieden, bewegt sich der Tisch zur Seite. Danach kann das Teil auf den Tisch einfahren. Kommt ein Teil aus der nicht bevorzugten Richtung, muß der Tisch zuerst bewegt werden. Das Teil passiert die Kreuzung und der Tisch bewegt sich wieder in die Ruheposition.

Benutzerschnittstelle

Die Länge, die Geschwindigkeit beim Einfahren auf den Tisch, die *Positionierzeit* und die *Rückstellzeit* des Tisches sind einstellbar. Zusätzlich kann eine *Bevorzugte Richtung* eingestellt werden, die als Ruheposition für den Tisch dient.



Dialog des Bausteins *cv_crossing*

Die Zeit für das Ausfahren des Teils wird durch die Länge und Geschwindigkeit des folgenden Bausteins abgebildet.

Die Pulkstrecke cv_batchline

Symbol: 

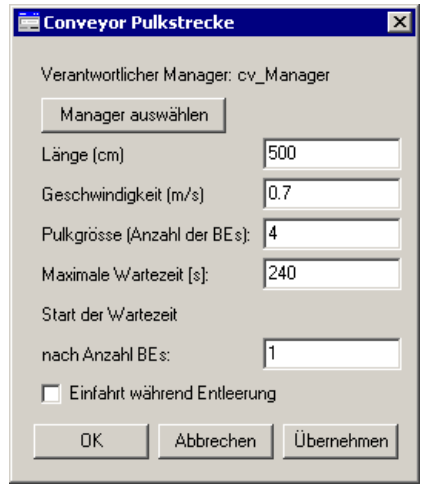
Funktionalität:

Dieser Baustein bildet aus unregelmäßig ankommenden Teilen sogenannte Pulks. Er sammelt eine einstellbare Anzahl Teile an und gibt diese ohne Abstand an den nachfolgenden Baustein weiter. Er besteht aus einem nicht staufähigen Band. Trifft ein Teil bei der Pulkstrecke ein, läuft das Band an und stoppt wieder, wenn das Teil komplett auf dem Band ist. Trifft ein weiteres Teil bei der Pulkstrecke ein, wird das Band erneut gestartet. Dieser Vorgang wird solange wiederholt, bis *Pulkgrösse* erreicht ist. Dann läuft das Band, bis alle Teile dieses Pulks von dem Band befördert wurden. Wenn nicht unbedingt alle Teile eines Pulks angesammelt werden müssen, gibt es die Möglichkeit, eine *Wartezeit* festzulegen. Weiterhin kann angegeben werden, ab wann diese *Wartezeit* aktiviert wird. Man kann z. B. festlegen, daß bei einer Pulkgröße von 5 nach dem dritten Teil die *Wartezeit* gestartet wird. Trifft während der *Wartezeit* kein weiteres Teil ein, werden die angesammelten Teile weitergegeben. Trifft ein weiteres Teil ein, wird die Wartezeit erneut gestartet. Damit wird festgelegt, daß ein Pulk eine minimale Anzahl von Teilen hat. Bitte beachten Sie, daß dieser Baustein niemals entleert wird, wenn weniger als drei Teile ankommen. Über *Einfahren während Entleerung* kann bestimmt werden, ob bereits während der

Entleerung des Bandes neue Teile einfahren dürfen. Dadurch können Lücken auf dem Band auftreten. Vereinzelte Teile dürfen dann auch einzeln aus der Pulkstrecke austreten, um Blockierungen zu vermeiden.

Benutzerschnittstelle

In diesem Baustein kann die Länge und Geschwindigkeit der Förderstrecke eingestellt werden. Durch die Länge der Pulkstrecke und die Länge der Teile wird festgelegt, wieviele Teile sich höchstens auf der Pulkstrecke befinden können. Diese Anzahl muß mindestens der *Pulkgrösse* entsprechen. Weiterhin kann die oben beschriebene Wartezeit und die Betriebsart festgelegt werden.



Dialog des Bausteins *cv_batchline*

Der Baustein cv_dockingStation

Symbol:

Funktionalität:

Mit dem Baustein kann ein Hub- oder Drehtisch abgebildet werden, der einen Anschlußbaustein für eine *Station* eines Produktionssystems darstellt. Eine Station kann ein Netzwerk oder eine Folge von Grundbausteinen sein. In der Station muss sich ein Grundbausteine befinden, der eine Bearbeitung der Teile ausführen kann. Der Baustein darf nur mit einer Bearbeitungsstation, die Teile aufnimmt und nach der Bearbeitung wieder abgibt, verbunden werden. Das sind die Bausteine Einzelstation, Parallelstation, Senke, Assembly und Disassembly.

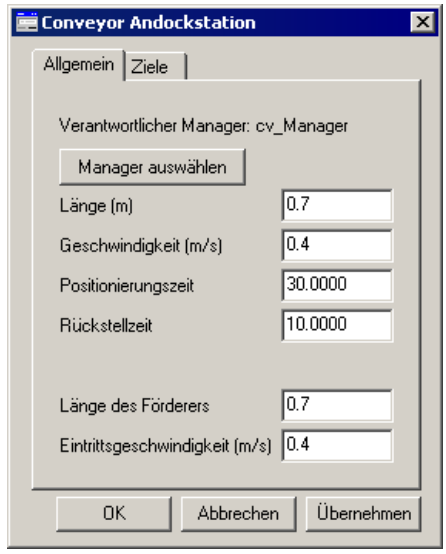
Der Baustein *cv_dockingStation* enthält einen Ausschleus- und eine Einschleustisch. Die Tische arbeiten voneinander unabhängig. Sie sind so angeordnet, dass gleichzeitig ein Teil eingeschleust und ein Teil ausgeschleust werden kann. Der Ausschleusetisch befindet sich vor dem Einschleusetisch. Unabhängig davon, ob das Teil in der angeschlossenen Station bearbeitet werden muß oder nicht, fährt das Teil vollständig auf den Tisch. Die Zeit für das Einfahren des Teils wird durch die Geschwindigkeit und die Länge des Grundbausteins *LiftTableLeft* abgebildet.

Muß der Tisch bewegt werden, so wird die *Positionierzeit* des Tisches berücksichtigt. Die Zeit für das Ausfahren des Teils wird durch die Länge und Geschwindigkeit von *ConveyorBelt_PartOut* bzw. *ConveyorBelt_PartIn* abgebildet. Ist das Teil komplett vom Tisch heruntergefahren, wird die *Rückstellzeit* des Tisches berücksichtigt. Danach kann ein weiteres Teil auf den Tisch fahren.

Wird ein Teil geradeaus über den Tisch befördert, wird keine Positionier- und Rückstellzeit verbraucht. Die Zeit für das Durchfahren des Teils wird durch die in *LiftTableRight, LiftTableLeft* und *conveyorBelt_Center* eingestellte Länge und Geschwindigkeit abgebildet.

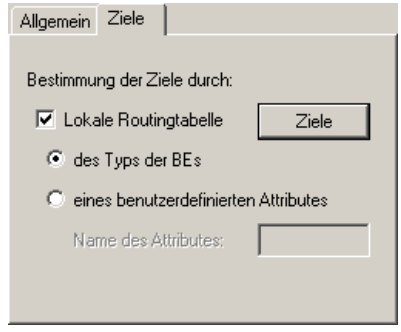
Benutzerschnittstelle:

Zwischen den beiden Tischen befindet sich eine Förderstrecke. Für die Förderstrecke sind Länge, Geschwindigkeit und Staufähigkeit einstellbar, für die Tische die Länge, die Geschwindigkeit beim Einfahren, die *Positionierzeit* und die *Rückstellzeit*.



Die Registerkarte Allgemein des Dialogs des Bausteins *cv_dockingStation*

Wenn das nächste Ziel nicht durch ein Produktionssystem dem Teil zugeordnet wird, so können Sie die nächsten Ziele durch lokale Routingtabellen definieren. Auf der Registerkarte Ziele könne Sie festlegen, ob das nächste Ziel vom Teiletyp oder von den Wert eines beliebigen freien Attributes abhängt. Der Typ ist durch den Wert des freien Attributes *entityType* (string) bestimmt. Hat das Teil nicht ein solches Attribut, so wird der Typ des Teils aus dem Namen bestimmt.



Die Registerkarte Ziele des Dialogs des Bausteins *cv_dockingStation*

Durch die Schaltfläche Ziele öffnen Sie die Routingtabelle, in der Sie die Zuordnung der Teile zu den Zielen festlegen können.

Einschleusestelle cv_PutIn

Symbol:

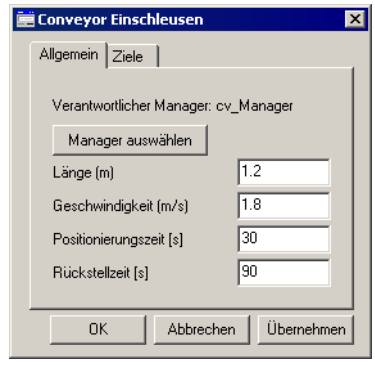
Funktionalität:

Mit dem Baustein kann ein Hub- oder Drehtisch abgebildet werden, der einen Anschlußbaustein für eine Station darstellt, die nur Teile abgibt z. B. eine Quelle.

Will ein Teil von der Seite in den Tisch eintreten, bewegt sich der Tisch zur Seite. Diese *Rückstellzeit* wird berücksichtigt. Nach der *Rückstellzeit* fährt das Teil vollständig auf den Tisch.

Nun muß der Tisch bewegt werden. Diese Bewegung benötigt die *Positionierzeit*. Danach fährt das Teil auf die Ausfahrstrecke, die sich außerhalb des Bausteins befindet. Ist das Teil komplett vom Tisch heruntergefahren kann ein weiteres Teil auf den Tisch fahren.

Benutzerschnittstelle:



Die Registerkarte Allgemein des Dialogs des Bausteins *cv_PutIn*

Auf der Registerkarte Ziele können Sie wie bei dem Baustein *cv_dockingStation* die nächsten Ziele über eine lokale Routingtabelle definieren.

Ausschleusestelle cv_PutOut

Symbol:

Funktionalität:

Mit dem Baustein kann ein Hub- oder Drehtisch abgebildet werden, der einen Anschlußbaustein für eine Station darstellt, die nur Teile aufnimmt z. B. eine Senke.

Soll ein Teil ausgeschleust werden, fährt das Teil auf den Tisch. Dieser wird gedreht. Dann fährt das Teil vom Tisch herunter und der Tisch dreht zurück. Der Unterschied zum Baustein *cv_br2* besteht darin, daß der Baustein *cv_PutOut* ein Ziel für die Teile ist. Wenn der Baustein als lokales Ziel in einem Teil eingetragen ist, fährt das Teil den Baustein auf dem kürzesten Weg automatisch an.

Benutzerschnittstelle:

Die Länge, die Geschwindigkeit beim Einfahren auf den Tisch, die *Positionierzeit* und die *Rückstellzeit* des Tisches sind einstellbar.

Die Gestaltung des Dialogs entspricht der des Bausteins *cv_PutIn*.

Die Schnittstelle zu anderen Transportsystemen

Der Baustein cv_interface

Symbol:

Der Baustein stellt einen Anschlußbaustein für die Schnittstelle eines weiteren Materialflußsystems dar. Das angeschlossene Materialflußsystem kann wieder ein Netzwerk aus der Objektbibliothek Plant Simulation Conveyor oder aber auch ein Netzwerk aus dem Objektbibliothek Fahrerloses Transportsystem Plant Simulation AGVS sein.

Funktionalität:

Die Teile werden über Tische ein- und ausgeschleust. Der Ausschleusetisch befindet sich vor dem Einschleusetisch. Die Tische arbeiten voneinander unabhängig. Dadurch können gleichzeitig Teile ein- und ausgeschleust werden. Zwischen den beiden Tischen befindet sich eine Förderstrecke. Für die Förderstrecke sind Länge, Geschwindigkeit und Staufähigkeit einstellbar, für die Tische die Länge, die Geschwindigkeit beim Einfahren auf den Tisch, die *Positionierzeit* und die *Rückstellzeit*.

Über den Ausschleusetisch kann ein Teil zu der Schnittstelle auf der anderen Seite ausgeschleust werden und über den Einschleusetisch kann ein Teil von der gegenüberliegenden Schnittstelle eingeschleust werden.

Der Unterschied zum Baustein *cv_dockingStation* besteht darin, daß hier die Ziele des angeschlossenen Materialflußsystems verwaltet werden. Eine Schnittstelle kann nur mit einer weiteren Schnittstelle verbunden werden. Bei der Verbindung ist auf die Richtung zu achten.

Benutzerschnittstelle:

Bedienung und Aufbau dieses Bausteins ähneln dem Baustein *cv_dockingStation*.

Die zentrale Steuerung

Der Baustein cv_Manager

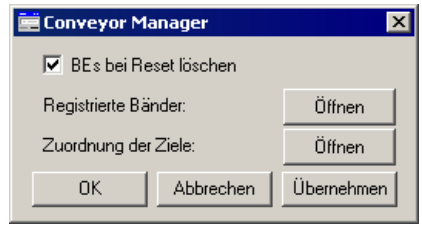
Symbol:

Funktionalität:

Die Zentrale hat verschiedene Funktionen. Zum einen stellt sie eine aktuelle Liste aller verfügbaren Bausteine, der Schnittstellen und aller erreichbaren Ziele zur Verfügung. Diese Listen werden bei jeder Änderung aktualisiert. Die Zentrale reagiert hierzu auf Meldungen der Bausteine z. B. *erzeugen, vernichten* oder *umbenennen*.

Benutzerschnittstelle:

In diesem Baustein kann eingestellt werden, ob das System neu initialisiert werden soll. Bei großen Modellen mit vielen Verzweigungen und Schnittstellen, benötigt die Initialisierung längere Laufzeiten. Wurden nur kleine Änderungen vorgenommen, so kann man die Initialisierung an dieser Stelle auch verhindern. Weiter kann eingestellt werden, ob bei einem *Reset* alle Teile gelöscht werden sollen. Wird der Baustein *ProductionManager* verwendet, so müssen die Teile immer gelöscht werden.



Dialog des Bausteins *cv_Manager*

Das bewegliche Element part

Symbol:

Mit diesen Bausteinen können die zu bearbeitenden Werkstücke dargestellt werden. Jedes Teil, das transportiert werden soll, muss die freien Attribute *globDestination* und *locDestination* haben. Die Zuordnungen der lokalen und globalen Ziele kann in dem Transportmanager *cv_Manager* eingesehen werden.

globDestination:

In diesem Attribut steht nach der Zielzuweisung des Teils der Pfad der nächsten Bearbeitungsstation. - Datentyp ist *object*.

locDestination:

Dieses Attribut enthält das lokale Ziel in einen Transportsystem, wie z.B. eine Andockstation. - Datentyp ist *object*.

EntityType:

Dieses optionale Attribut bestimmt, um welche Art Werkstück es sich bei dem Teil handelt - Datentyp ist *string*.

4

EOM

Einführung

Die Objektbibliothek Plant Simulation EOM (Electrical Overhead Monorail) ist für die Nachbildung eines Unstetigfördersystems mittels einer Elektrohängebahn (EHB) entwickelt worden. Um ein Transportsystem zu modellieren, muß mit den bereitgestellten Wegelementen das Layout nachgebildet werden. Die einzelnen Bausteine, wie z.B. Gerade, Kurve, Auf- und Abgabestation, Puffer, Weiche bilden das reale Verhalten einer *Elektrohängebahn* ab. Anlagenspezifische Parameter, die zur Steuerung des Systems erforderlich sind, müssen vom Anwender in Tabellen und globalen Variablen der eingesetzten Bausteine beschrieben werden. Die Anzahl der gewünschten Fahrwerke, die zur Erledigung der Transportaufträge eingesetzt werden sollen, sowie deren Verteilung im System, sind ebenfalls vom Anwender vorzugeben. Eine weitere komfortable Möglichkeit zur Simulation des Anlagendurchsatzes bietet die Tranportmatrix, über die der Materialfluß von beliebig vielen Aufnahmestationen an beliebig viele Abgabestationen vorgegeben werden kann.

Diese Objektbibliothek soll die jeweilige Modellierungsaufgabe möglichst vollständig abdecken. Aufgrund seiner Flexibilität und Allgemeinheit sind die Bausteine und die Beziehung der Bausteine untereinander oft komplex. Die Anpassung verlangt ein gutes Verständnis der Funktionalität und ist daher nicht immer einfach. Deshalb ist vor jedem Einsatz zu prüfen, ob die vorliegende Funktionalität die Anforderungen des einzelnen Einsatzfalles abdeckt. Sind Anpassungen zu erwarten, sollte das Nutzen-Aufwand-Verhältnis für den aktuellen Einsatz sorgfältig geprüft werden. Jedenfalls eignet sich diese Objektbibliothek für die schnelle Erstellung von Prototypen und als Referenz für eigene Bausteinkästen.

Die Objektbibliothek *Elektrohängebahn* ist auf der Basis von Plant Simulation entwickelt worden und setzt zu seiner Nutzung eine Plant Simulation Entwicklungslizenz voraus. Für die Benutzung der Objektbibliothek werden Kenntnisse in Plant Simulation vorausgesetzt.

Beispielmodelle

Das Modell *EOMsystem* zeigt beispielhaft das Layout einer Fertigungskette, mit der Aufgabe, zu fertigende Teile anhand eines vorgegebenen Arbeitsplanes durch einzelne Bearbeitungsstationen zu schleusen. Das Materialflußsystem ist mit den Elementen der *Elektrohängebahn*-Objektbibliothek realisiert. Die Fertigungssteuerung ermittelt dabei jeweils das nächste Ziel für die eingesetzten Fahrwerke. Der Zeitverbrauch für den Transport zwischen den Stationen wird realistisch mit Fahrgeschwindigkeiten und Blockierungen abgebildet.

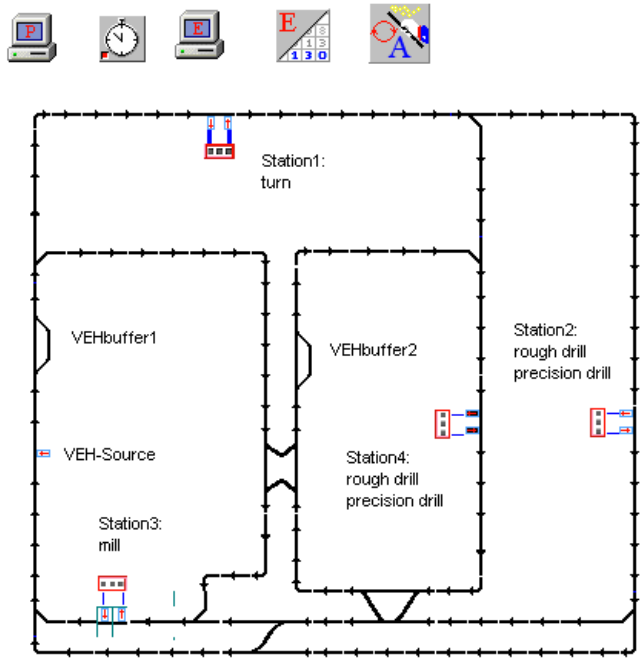


Abbildung 1:Layout einer Fertigungskette mit Elektrohängebahn

Das Modell besteht aus folgenden Komponenten:

- Fertigungssteuerung *ProdMgr* (Kundenaufträge, Arbeitspläne der Teile, Auftragseinlastung)
- Station zur Bearbeitung der Teile
- Bausteine der *Elektrohängebahn* zur Modellierung des Transportweges
- EHB-Zentrale zur Verwaltung der Fahrwerke und Teile (*EOMcontrol*)

Am Ende dieses Abschnittes wird der prinzipieller Ablauf kurz erläutert:

In der Fertigungssteuerung *ProdMgr* wird mit einem Arbeitsplan und einer Auftragsliste vorgegeben, welche Werkstücke wann und wo eingelastet und wie bearbeitet werden sollen. Alle Bearbeitungsstationen operieren zusammen mit der Fertigungssteuerung.

Die Einspeisung der leeren Fahrwerke in das System wird von einer Fahrwerksquelle vorgenommen. Es werden sowohl kreisende als auch wartenden Fahrwerke in den Puffern, Be- und Entladestationen erzeugt.

Die Schnittstellen zwischen den Bearbeitungsstationen und dem Fördersystem bilden die Be- und Entladestationen der Objektbibliotheks Lediglich für die Entladestationen ist eine Zuordnung zwischen der Bearbeitungsstation und der Andockstation erforderlich, da die bearbeiteten Teile von der Fertigungssteuerung entsprechend dem Arbeitsplan als nächstes Ziel eine Bearbeitungsstation erhalten. Die Zuordnung der Stationen zu den Andockstationen wird in einer Tabelle in der EHB-Zentrale *EOMcontrol* durch den Modellierer vorgenommen.

Wie ein solches Modell erstellt wird und welche Parameter für die einzelnen Bausteine gesetzt werden müssen, ist nachfolgend beschrieben. Die genaue Beschreibung aller Bausteine erfolgt in den späteren Kapiteln.

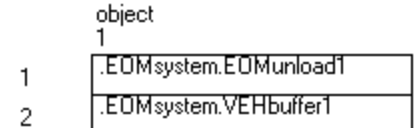
Im nächsten Absatz werden die Schritte zur Erstellung des im Layout einer Fertigungskette mit Elektrohängebahn dargestellten Modells beschrieben. Das fertige Modell *EOMexample* ist in der Bausteinbibliothek enthalten.

EHB-Bausteine

Vor dem Einsetzen der Bausteine werden oft verwendete Wegelemente in der Bausteinbibliothek parametrisiert. Hierzu gehören insbesondere die Länge, die konstant auf den Wert 5 m gesetzt wurde, sowie die Bausteinkapazität, die (mit Ausnahme des Pufferweges in den Leerfahrwerkspuffern) immer den Wert 1 hat. Die Länge der Bausteine ist identisch mit dem Blockungsweg. Alle weiteren Eingaben sind vom Anwender in den Listen, Tabellen und globalen Variablen, die im jeweiligen Baustein in grüner Farbe dargestellt sind, vorzunehmen.

Beladestationen

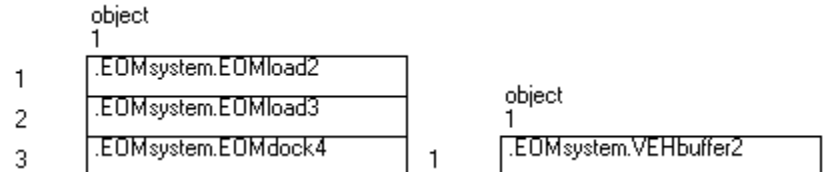
Die Eingangspuffer der Beladestationen werden mit dem Ausgangspuffer der zugehörigen Bearbeitungsstation über eine Kante verbunden. Die globale Variable *loadtime* beschreibt die Zeit zum Beladen des Fahrwerkes. Die Liste *StationsL* erhält die Einträge von vorgelagerten Netzwerken, aus denen verfügbare Leerfahrwerke von bearbeiteten Teilen angefordert werden. Sollen freie Fahrwerke ohne Fahrauftrag beim Durchfahren der Station angehalten und in Wartestellung versetzt werden, muß die globale Variable *veh_wait* auf TRUE gesetzt werden.



Liste *StationsL* von der Beladestation 1

Entladestationen

Die Ausgangspuffer der Entladestationen werden mit den Eingangspuffern der Bearbeitungsstationen über eine Kante verbunden. Die logische Zuordnung zwischen den Entlade- und den Bearbeitungsstationen erfolgt in der EHB-Zentrale. Die Variable *unloadtime* beschreibt die Zeit zum Entladen eines Fahrwerkes. Damit das Fahrwerk nach dem Entladen vor der Station stehen bleibt, wird die Variable *veh_wait* in der Entladestation auf TRUE gesetzt. Die Liste *StationsL* enthält die Einträge von nachfolgenden Beladestationen, in denen weiterfahrende Fahrwerke nach dem Entladen nach abholbereiten Werkstücken suchen. In die Pufferliste *BufferL* wird der nächstgelegene Puffer eingetragen.



Listen *StationsL* und *BufferL* von Entladestation 2

Dockstation

An die Bearbeitungsstation 4 wurde die kombinierte Be- und Entladestation *EOMdock4* angeschlossen. Sie besitzt die gleiche Funktionalität wie die beiden einzelnen Stationen. Die Variablen *loadtime* und *unloadtime* müssen gesetzt werden. Weiterhin müssen die Listen *VEHStationsL* für die Anforderung nach verfügbaren Leerfahrwerken und *CARGOStationsL* für die Suche nach einer wartenden Teilen ausgefüllt werden. Die Variable *veh_wait* steht in diesem Baustein auf FALSE, so daß die in dieser Station entladenen Fahrwerke weiterfahren. Ankommende Leerfahrwerke, die nicht beladen werden, fahren dann an dieser Station vorbei. Entladene Fahrwerke suchen vor der Weiterfahrt in der Pufferliste *BufferL* nach einem anzufahrenden Leerfahrwerkspuffer.

	object 1
1	.EOMsystem.VEHbuffer2
2	.EOMsystem.VEHbuffer1
3	.EOMsystem.EOMunload1

	object 1
1	.EOMsystem.EOMload3
2	.EOMsystem.EOMload1
3	.EOMsystem.EOMload2

Listen *VEHStationsL* und *CARGOStationsL* von *Dockstation4*

EHB-Zentrale

Die EHB-Zentrale *EOMcontrol* enthält die Tabelle *STATattachL*, in der die Zuordnung zwischen den Bearbeitungsstationen und den Entladestationen der *Elektrohängebahn* vorgenommen werden. Dies ist erforderlich, da die Fertigungssteuerung den zu bearbeitenden Teilen als nächstes Ziel den Namen *GlobDestName* der Station zuweist, die Fahrwerke der Objektbibliothek jedoch mit den Namen der Entladestationen *LocDestNameUnload* operieren. In diesem Baustein befindet sich auch das Netzwerk *CARGOinput* zur Einspeisung der Frachten (Teile) über eine Transportmatrix, welches an dieser Stelle nicht beschrieben werden soll, da es im Beispielmodell nicht verwendet wird.

string 0	object 0 GlobDestName	object 1 LocDestNameUnload
1	.EOMsystem.Station1	.EOMsystem.EOMunload1
2	.EOMsystem.Station2	.EOMsystem.EOMunload2
3	.EOMsystem.Station3	.EOMsystem.EOMunload3
4	.EOMsystem.Station4	.EOMsystem.EOMdock4

Tabelle *STATattachL* in der EHB-Zentrale

Geraden und Kurven

Alle Parameter der Bausteine *EOMcurve, EOMstraight* wurden schon in Bausteinbibliothek gesetzt, so daß nach dem Einsetzen keine Eingaben erforderlich sind. Abweichende Längen, die gleichzeitig auch die Blockstrecken repräsentieren, können im Modell natürlich jederzeit umgesetzt werden. Die Geschwindigkeit der Fahrwerke ist in Kurven geringer als auf Geraden. Die Eingabe der Kurvengeschwindigkeit wird über das freie Attribut *vCurve* des Fahrwerkes vorgegeben.

Verzweigungen

In den Verzweigungen müssen die Zielliste *DestL* sowie die Wegetabelle *WayDataL* ausgefüllt werden. Die Zielliste enthält alle lokalen Ziele, die über die entsprechende Verzweigung erreichbar sind. In der Wegetabelle sind für die vorgegebenen Routen die Streckenlängen, sowie die Blockungszeiten für die vorzeitige Streckenfreigabe einzutragen. Nach Ablauf der Blockungszeit wird der Eintrittsbaustein einer Verzweigung schon wieder zur Einfahrt freigegeben, bevor das Fahrwerk das Ende seiner Blockstrecke (das Ende des angestrebten Ausgangsbausteines) erreicht hat. Die Blockstreckensteuerung ist in Kapitel 3 beschrieben.

	object 1
1	.EOMsystem.EOMdock4
2	.EOMsystem.EOMload3
3	.EOMsystem.EOMunload3
4	.EOMsystem.VEHbuffer2
5	.EOMsystem.EOMload1
6	.EOMsystem.EOMunload1

Zielliste *DestL* der Verzweigung *EOMbranch2*

string 0	string 0	real 1	time 2
	Route	distance	blockingtime
1	way_in-way_out1	5	0.0000
2	way_in-way_out2	6	0.0000
3			

Wegetabelle *WayDataL* der Verzweigung *EOMbranch2*

Zusammenführungen

In Zusammenführungen muß ebenfalls die Tabelle *WayDataL* ausgefüllt werden. Neben den schon von der Verzweigung bekannten Werten, gibt es einen zusätzlichen Parameter *priority* um die Vorfahrt zu regeln. Standardmäßig sind die Prioritäten für beide Routen auf die höchste Priorität 1 gesetzt, wodurch das FIFO-Prinzip (first-in-first-out) realisiert wird. Um bei gleichzeitiger Belegung der Eingangswege ein bestimmtes Fahrwerk bevorzugt einfahren zu lassen, muß die Priorität der anderen Route durch Änderung auf einen Wert größer als 1 erniedrigt werden.

string 0	string 0	real 1	time 2	integer 3
	Route	distance	blockingtime	priority
1	way_in1-way_out	5	0.0000	1
2	way_in2-way_out	6	0.0000	1

Wegetabelle *WayDataL* der Zusammenführung *EOMjoin23*

Weichen

Da Weichen eine Kombination aus Zusammenführungen und Verzweigungen sind, müssen in diesen Bausteinen die Ziellisten *DestL* und die Wegetabellen *WayDataL* ausgefüllt werden.

	object 1
1	.EOMsystem.VEHbuffer2
2	.EOMsystem.EOMdock4

Zielliste *DestL* der Weiche *EOMswitch2*

string 0	string 0	real 1	time 2	integer 3
	Route	distance	blockingtime	priority
1	way_in1-way_out1	8	0.0000	1
2	way_in1-way_out2	10	0.0000	1
3	way_in2-way_out1	10	0.0000	1
4	way_in2-way_straight	5	0.0000	0
5	way_straight-way_out2	5	0.0000	1

Wegetabelle *WayDataL* der Weiche *EOMswitch2*

Puffer

Der Pufferbaustein *way_buff* unterscheidet sich von allen anderen Wegbausteinen durch seine unendliche Kapazität (beschrieben durch -1). Die Anzahl der Fahrwerke, die vom Puffer aufgenommen werden können, ist von der Länge des

Puffers und der Stapellänge der Fahrwerke (freies Attribut *stackLength*) abhängig. Da im Beispiel die Pufferlänge jeweils 10 m beträgt, wird zum Längenausgleich neben *way_straight* noch der Wegbaustein *way* mit ebenfalls 5 m Länge eingesetzt.

In den Puffer-Bausteinen sind immer zwei Listen auszufüllen: Die Liste *BufferL* enthält jeweils den Eintrag eines nachfolgenden Puffers, der angefahren werden soll, wenn der aktuelle Puffer belegt ist. In die Liste *StationsL* wurden die Einträge von den jeweils nachfolgenden Beladestationen eingetragen. Sobald ein Fahrwerk in die vorderste Pufferposition gelangt, wird geprüft, ob in einer dieser Stationen ein abholbereites Werkstück auf seinen Weitertransport wartet. Im Baustein *way_straight* wurde das Attribut *prio2way_out* auf 2 zurückgesetzt, so daß ein Fahrwerk aus dem Puffer immer bevorzugt losfahren darf, da diesem Fahrwerk ein Fahrauftrag zugewiesen wurde.

	object 1
1	.EOMsystem.VEHbuffer2

	object 1
1	.EOMsystem.EOMload1
2	.EOMsystem.EOMload2
3	.EOMsystem.EOMdock4

BufferL und *StationsL* des Leerfahrwerkpuffers *VEHbuffer1*

Fahrwerksquelle

In der Fahrwerksquelle wurde die Tabelle *VEHtableL* mit den einzuspeisenden Fahrwerken belegt. Neben der Erzeugung einer unterschiedlichen Anzahl von Fahrwerken in den beiden Puffern, werden auch in der Quelle selbst im 15 s Rhythmus 6 kreisende Fahrwerke in Umlauf gebracht. Vor Be- und Entladestationen kann selbstverständlich immer nur ein Fahrwerk in Warteposition sein.

string 0	object 1 v_source	object 2 v_typ	integer 3 v_num
1	.EOMsystem.VEHinput	.veh	6
2	.EOMsystem.VEHbuffer1	.veh	2
3	.EOMsystem.VEHbuffer2	.veh	3
4	.EOMsystem.EOMload1	.veh	1
5	.EOMsystem.EOMdock4	.veh	1
6			

Fahrwerktabelle *VEHtableL* im Baustein *VEHinput*

Fahrwerkparameter

In der Objektbibliothek befindet sich das Fahrwerk, das über die Quelle in beliebiger Anzahl abgeleitet und in Umlauf gebracht wird. Vor Simulationsbeginn wurden folgende Parameter gesetzt:

- Fahrwerklänge: 1 m
- Geschwindigkeit des Fahrwerkes auf der Geraden: 1 m/s
- Stapellänge *stackLength* der Fahrwerke in Puffern: 3 m
- Geschwindigkeit *vCurve* der Fahrwerke in Kurven: 0.5 m/s

Einsetzen des Protokollbausteines

Sollten während eines Simulationslaufes Fehler auftreten, so werden diese in Meldungs-, Warnungs- und Fehlerlisten registriert. Im Protokollbaustein können diese Listen aufgerufen werden.

Nach erbrachter Vorarbeit kann das Modell nun über den Ereignisverwalter gestartet werden.

Hinweis: Um die Komplexität des Modells in Grenzen zu halten, wurden nicht alle zur Verfügung stehenden Bausteine verwendet.

Allgemeine Abläufe

In den folgenden Abschnitten wird die prinzipielle Funktionsweise der *Elektrohängebahn* beschrieben. Diese Kapitel nehmen zum besseren Verständnis gelegentlich konkreten Bezug auf bestimmte Anwenderbausteine, die parallel zur Lektüre dieser Dokumentation betrachtet werden sollten.

Disposition der Fahrwerke

Das Zusammenspiel von verfügbaren EHB-Fahrwerken und den zu transportierenden Frachten ist bei dem Materialflußsystem *Elektrohängebahn* das zentrale Thema. Es ist hierbei von sekundärer Bedeutung, ob die durchzuführenden Transportaufträge von einer Fertigungssteuerung *ProdMgr* oder über eine Transportmatrix vorgegeben werden. Der Unterschied hierbei besteht lediglich darin, daß beim Einsatz einer Fertigungssteuerung die Teile nacheinander zu mehreren Bearbeitungsstationen befördert werden können, während beim Einsatz einer Transportmatrix die Teile jeweils nur von einer Quellstation zu einer Zielstation befördert werden, und dort nach dem Abladen sofort vernichtet werden. Sowohl die Anforderung von Fahrwerken, als auch die Suche nach wartenden Frachten, geschieht über Listen, die schon in den betreffenden Bausteinen angelegt und vom Modellierer auszufüllen sind.

Ein EHB-Fahrwerk kann grundsätzlich folgende Belade- und Dispositionszustände aufweisen, die durch Farben gekennzeichnet sind:

- a)beladen

  mit Fahrauftrag auf dem Weg zum Ziel (Entladestation) - grau
- b)unbeladen
  - kreisend in der Anlage ohne Ziel - grün
  - kreisend in der Anlage mit Ziel “nächster freier Puffer“ - grün
  - in Warteposition in einer Be- oder Entladestation - blau
  - austrittsbereit aus Puffer - rot
  - eingereiht in Puffer - rot
  - mit Fahrauftrag auf dem Weg zum Ziel (Beladestation) - gelb

Um die anstehenden Fahraufträge jeweils bestmöglich erledigen zu können, wurden in der Objektbibliothek *Elektrohängebahn* prinzipiell folgende Strategien realisiert:

- Ein abholbereites Teil wird in den Eingangspuffer einer Beladestation überstellt und fordert von dort ein verfügbares Leerfahrwerk an. Alle verfügbaren Leerfahrwerke sind in einer Fahrwerksliste in der EHB- Zentrale registriert, aus der sie nach erfolgter Anforderung wieder gelöscht werden. Kann kein Leerfahrwerk gefunden werden, wird das bearbeitete Teil seinerseits in der EHB-Zentrale in einer Teileliste registriert. Um die Suche nach Leerfahrwerken räumlich einzugrenzen, wurde in den Beladestationen eine zusätzliche Liste eingeführt, die vom Anwender mit den Einträgen der benachbarten Stationen versehen werden kann. Die Suche wird zunächst nur in diesen Stationen durchgeführt und erst wenn dort kein Fahrwerk gefunden werden kann, wird automatisch in allen übrigen Stationen gesucht.
- Jene Leerfahrwerke, die in eine Warteposition gelangen, suchen ihrerseits nach bearbeiteten Teilen, die ebenfalls in einer Teileliste der EHB-Zentrale registriert wurden. Wird ein solches Teil gefunden, wird dessen Registrierung aus der Liste gelöscht und das Fahrwerk fährt zu der betreffenden Beladestation. Bei erfolgloser Suche wird das Fahrwerk in der Fahrwerksliste der EHB-Zentrale registriert. Wie schon in den Beladestationen, wurden zur Eingrenzung der Suche nach abholbereiten Teilen auch in diesen Bausteinen eine Stationsliste eingeführt, die vom Anwender jeweils mit den Einträgen der vorrangig zu durchsuchenden Beladestationen versehen werden muß.

- Die dritte Möglichkeit der Lastübernahme geschieht durch kreisende Leerfahrwerke, die leer und ohne Fahrauftrag zufällig an einer Beladestation vorbeikommen. Das abholbereite Teil wird übergeben und vom Fahrwerk zu dessen nächstem Ziel gebracht. Eine evtl. schon bestehende, von dem ankommenden Teil ausgelöste Fahrwerksanforderung, wird gelöscht.
- Beladene Fahrwerke bringen die Teile zu der entsprechenden Entladestation und übergeben diese an den Ausgangspuffer der Station. Abhängig von den getroffenen Benutzereinstellungen, bleibt ein Fahrwerk nach dem Entladen entweder stehen oder sucht sich ein abholbereites Teil. War die Suche erfolglos, fährt es bei ausgefüllter Pufferliste in einen Leerfahrwerkspuffer oder kreist in der Anlage und fährt selbständig in den nächsten freien Puffer. Kreisende Fahrwerke können durch Setzen einer Variablen vor den Beladestationen ebenfalls in den Wartezustand versetzt werden.
- Bei allen Ladevorgängen können unterschiedliche Be- und Entladezeiten vorgegeben werden.

Ereignis: Teil sucht Fahrwerk

Nach abgeschlossener Bearbeitung in einer Bearbeitungsstation, erhält ein Teil entsprechend dem Arbeitsplan ein neues Ziel und wird in den Eingangspuffer *part_avail* der angedockten Beladestation *EOMloadx/EOMdockx* überstellt. Dort angekommen, werden die in der Liste *StationsL/VEHStationsL* eingetragenen Bausteine (*VEHbuffer, EOMunload, EOMload, oder EOMdock*) sequentiell nach einem verfügbaren Leerfahrwerk durchsucht. Konnte in den eingetragenen Stationen kein Fahrwerk gefunden werden oder enthält die Stationsliste *StationsL* keinen Eintrag, so wird die Anforderung über die Liste *EOMcontrol.VEHavailableL* in der EHB-Zentrale auf die gesamte Anlage ausgedehnt. Bei erfolgreicher Suche, wird das Fahrwerk angefordert und seine Registrierung aus dieser Liste *EOMcontrol.VEHavailableL*, die alle verfügbaren Fahrwerke enthält, gelöscht. Die Variable im Beladebaustein *veh_required* wird zu Prüfzwecken auf das angeforderte Fahrwerk gesetzt. Sollte die wartende Fracht schon vor dessen Eintreffen von einem kreisenden Fahrwerk übernommen worden sein, so wird die Anforderung über diese Variable rückgängig gemacht und das Fahrwerk wieder frei gesetzt. Beim Eintreffen des gerufenen Fahrwerks wird die Variable ebenfalls zurückgesetzt. Vor der Weiterfahrt wird die benutzerdefinierte Beladezeit *loadtime* berücksichtigt.

Da die Liste *StationsL* sequentiell abgearbeitet wird, werden sinnvollerweise zuerst die der Beladestation direkt vorgelagerten Puffer und Entladestationen eingetragen, damit die Fahrwege möglichst kurz gehalten werden.

Enthält *StationsL* keinen Eintrag, oder wurde kein Leerfahrwerk gefunden, wird das wartende Teil in der Liste *EOMcontrol.CARGOwaitingL* registriert.

	object 1
1	.EOMsystem.EOMunload2
2	.EOMsystem.EOMunload1
3	.EOMsystem.VEHbuffer1

Liste *StationsL* einer Beladestation

Ereignis: Fahrwerk sucht Auftrag

Fahrwerke, die in die erste Pufferpostion in *VEHbufferx.way_buff* gelangen, oder solche, die in einer Entladestation *EOMunload/EOMdock* soeben entladen wurden und dort nicht aufgrund der benutzerdefinierten, globalen Variablen *veh_wait* in Warteposition verbleiben sollen, suchen selbständig nach einem neuen Fahrauftrag. In diesen Bausteinen befindet sich ebenfalls eine Liste *StationsL/CARGOStationsL*, in die vom Benutzer die umliegenden Beladestationen (vom Typ *EOMload* oder *EOMdock*) eingetragen werden. Entsprechend der eingetragenen Reihenfolge werden die Stationen sequentiell nach abholbereiten Teilen durchsucht. Bei erfolgreicher Suche wird im Zielbaustein die globale Variable *veh_required* auf das suchende Fahrwerk gesetzt, das auf den Weg zur Übernahme ist. Sollte die wartende Fracht schon vor dessen Eintreffen von einem kreisenden Fahrwerk übernommen werden, so wird die Anforderung über diese Variable rückgängig gemacht und das Fahrwerk wieder freigesetzt. Nach dem Eintreffen und Beladen des angeforderten Fahrwerks wird diese Variable ebenfalls zurückgesetzt.

Bei erfolgloser Auftragssuche in den eingetragenen Stationen, wird auch hier über die Liste *EOMcontrol.CARGOwaitingL* automatisch in der gesamten Anlage nach den am längsten wartenden Teilen (FIFO-Prinzip, d.h. unabhängig von deren derzeitigem Standort) gesucht. Die Registrierung gefundener Teile wird dann gelöscht. Um kurze Fahrwege zu realisieren, werden in *StationsL/CARGOStationsL* die unmittelbar nachfolgenden Beladestationen eingetragen.

Wurde in der gesamten Anlage keine wartendes Teil gefunden, so wird ein Fahrwerk, das sich in einem Puffer *VEHbufferx* befindet, in der Fahrwerksliste *EOMcontrol.VEHavailableL* registriert. Ein Fahrwerk in einer Entladestation sucht in einem weiteren Schritt in der Liste *BufferL* nach einem anzufahrenden Puffer. Bei eingetragenem Puffer wird dieser zum neuen Zielort des Fahrwerkes. Anderenfalls geht das Fahrwerk in den Zustand des Kreisens über und sucht sich selbständig den nächsten freien Puffer. Kommt ein kreisendes Fahrwerk an einer Belade- oder Dockstation vorbei, in der die Variable *veh_wait* auf TRUE gesetzt ist, verbleibt das Fahrwerk vor dieser Station in Wartezustand und wird wiederum in der EHB-Zentrale in der Liste aller verfügbaren Fahrwerke *EOMcontrol.VEHavailableL* registriert. Es verbleibt vor der Station, bis es dort entweder beladen, von einem Teil in einer anderen Station angefordert oder von einem Nachfolger vertrieben wird

Listen *StationsL* und *BufferL* eines Leerfahrwerkpuffers

Ereignis: Kreisendes Fahrwerk übernimmt wartendes Teil

Bei jeder Durchfahrt an einer Beladestation prüft ein kreisendes Leerfahrwerk oder ein Fahrwerk auf dem Weg zu einem Puffer, ob in dem Eingangspuffer der Beladestation *part_avail* ein wartendes Teil ist. Ein solches Teil wird vom Fahrwerk übernommen. Es wird über die Variable *veh_required* geprüft, ob die anstehende Fracht zuvor schon ein Leerfahrwerk aus einem Puffer oder einer anderen Station angefordert hat. Ist dies der Fall, so wird dieses Fahrwerk wieder frei gesetzt. Konnte von dem wartenden Teil kein Fahrwerk angefordert werden, erfolgt deren Registrierung in *EOMcontrol.CARGOwaitingL*.

Ereignis: Beladenes Fahrwerk übergibt Fracht

Beladene Fahrwerke vergleichen an Entladestationen *EOMunload/EOMdock* ihren Zielort mit dem aktuellen Standort, und übergeben bei Übereinstimmung ihre Fracht an den Ausgangspuffer *part_unload* der Station. Die für den Entladevorgang notwendige Zeit ist in der Variable *unloadtime* gespeichert, die vom Benutzer gesetzt wird. Wurde vom Benutzer die Variable *veh_wait* auf TRUE gesetzt, verbleibt das entladene Fahrwerk im Wartezustand und wird gleichzeitig in *EOMcontrol.VEHavailable* registriert. In diesem Zustand kann es wiederum von abholbereiten Teilen angefordert oder durch nachfolgend eintreffende Fahrwerke vertrieben werden. Soll das Fahrwerk nach dem Entladen nicht stehenbleiben, sucht es einen neuen Fahrauftrag, fährt in einen Puffer oder kreist in der Anlage.

Blockstrecken

Um das Auffahren von Fahrwerken zu vermeiden, wird bei der *Elektrohängebahn* im Regelfall das Prinzip der Eisenbahn-Blockung angewendet. Unter einer *Blockstrecke* versteht man einen Bereich, in dem nur ein Fahrwerk befinden darf. Zu diesem Zweck wird die Wegstrecke der gesamten Anlage aus einzelnen Wegbausteinen aufgebaut, von denen jeder für sich immer genau eine Blockstrecke repräsentiert. Die Länge des Grundbausteins *Weg* stimmt mit der Länge der Blockstrecke überein. Bei der Eisenbahn-Blockung wird aus Sicherheitsgründen zwischen zwei in Fahrt befindlichen Fahrwerken immer mindestens eine Blockstrecke freigehalten. Dies wird dadurch realisiert, daß von einem Fahrwerk die vorhergehende Blockstrecke immer erst dann zum Einfahren freigegeben wird, wenn es seinerseits das Ende seiner eigenen Blockstrecke erreicht hat und bereit zur Einfahrt in die nachfolgende Strecke ist. Unabhängig davon, ob dieses Fahrwerk sofort weiterfahren kann, oder seinerseits durch ein vorausfahrendes Fahrwerk blockiert wird, kann der Nachfolger keinesfalls in dessen Blockstrecke nachrücken, da sich in jeder Blockstrecke immer nur ein Fahrwerk befinden kann. Aus-

löser der Freigabe- und Sperraktionen ist jeweils das Fahrwerk, welches das Ende einer Blockstrecke erreicht hat. In diesem Moment wird der Vorgängerbaustein freigegeben.

Diese Blockungsstrategie wurde mit Ausnahme der Fahrwerkstapelung in Puffern, sowie bei der Fahrt durch Verzweigungen, Weichen und Zusammenführungen in allen anderen Bausteinen zur Anwendung gebracht. In diesen speziellen Bausteinen repräsentieren die Eingangswege zwar ebenfalls jeweils eine Blockstrecke, es können im weiteren Verlauf jedoch unterschiedliche Weglängen, Blockungszeiten und Prioritäten zur Vorfahrtsregelung gesetzt werden.

Räumstrategie

Wie schon erwähnt, können Fahrwerke entsprechend der Variablen *veh_wait* in Entladestationen *EOMunload*, Beladestationen *EOMload* oder kombinierten Stationen *EOMdock* solange verweilen, bis sie von einen Teil angefordert werden. Da ein wartendes Fahrwerk jedoch alle nachkommenden Fahrwerke blockieren würde, wurde die globale Räumstrategie *veh_clear* in *EOMcontrol* eingeführt. Hierzu prüft ein blockiertes Fahrwerk bei jeder auftretenden Blockierung den Status des blockierenden Fahrwerkes. Handelt es sich bei diesem seinerseits um ein blockiertes Fahrwerk, liegt der normale Blockungsfall oder eine Be-/Entladung vor. Befindet sich dieses Fahrwerk jedoch im Status *waiting*, tritt die Räumstrategie in Kraft, schickt das wartende Fahrzeug weiter und löscht dessen Registrierung in *EOMcontrol.VEHavailableL*.

Verhalten in Puffern

Kreisende Fahrwerke und Fahrwerke mit dem Ziel eines beliebigen Puffers *VEHbuffer*, prüfen bei Erreichen des Puffereingangsbausteines *way_in* die Belegung des eigentlichen Pufferweges *way_buff*, in dem die leeren Fahrwerke entsprechend ihrem Stapelabstand aufgereiht werden. Der Stapelabstand *stackLength* ist ein freies Attribut des Fahrwerkes, das vom Benutzer bei der Verwendung von Puffern unbedingt gesetzt werden muß. Die Überprüfung der belegten Pufferlänge in diesen Bausteinen wird über die globalen Variable *occuLength* realisiert. Die Länge des gesamten Puffers muß im Wegbaustein von *way_buff* gesetzt werden.

Alle Fahrwerke, die an der 1. Position eines Puffers stehen, und beim dortigen Eintreffen nicht gleich einen Fahrauftrag gefunden und angetreten haben, sind in der Liste *EOMcontrol.VEHavailableL* registriert. Sie verlassen ihre Position erst, wenn sie von einer anstehenden Fracht angefordert werden. Sollte das Fahrwerk nach einer Anforderung nicht sofort austreten können, da der Ausgangsbaustein *way_out* in diesem Moment von einem durchfahrenden Fahrwerk belegt ist oder inaktiviert wurde, wartet das Fahrwerk auf die Freigabe dieses Bausteines. Sollte der Fahrauftrag während dieser Wartezeit gelöscht werden, da ein kreisendes Fahrwerk das anfordernde Teil inzwischen mitgenommen hat, verbleibt das Fahrwerk im Puffer und wird wieder registriert.

Kann ein Fahrwerk nicht in einen Puffer eintreten, da dieser schon belegt ist, wird in der Liste *BufferL* in den Bausteinen *VEHbuffer* nach einem neuen Puffer gesucht. Erhält die Liste einen alternativen Puffereintrag, wird dieser zum neuen Zielort des Fahrwerkes, andernfalls kreist es und sucht sich selbst einen leeren Puffer.

Die gerade Pufferdurchfahrt besteht aus den Bausteinen *way* und *way_straight*, in denen auch die standardmäßige Blockung greift. Sollten bei Verlängerung des eigentlichen Pufferweges *way_buff* in den geraden Teil noch weitere Bausteine (also Blockstrecken) eingeführt werden, so sind diese unbedingt zwischen die 2 schon existierenden Bausteine *way* und *way_straight* einzusetzen.

Verhalten in Weichen und Zusammenführungen

Weichen und Zusammenführungen gleichen sich dahingehend, daß aus mehreren Richtungen kommende Fahrwerke gleichzeitig auf demselben Baustein einfahren wollen, bzw. diesen als Kreuzungspunkt bis zum Ende ihrer gesamten Blockstrecke benötigen. Das gleichzeitige Einfahren wird durch die Anwendung der FIFO-Regel vermieden. Das zuerst eintreffende Fahrwerk reserviert die für seine Weiterfahrt erforderlichen Bausteine. Kann das Fahrwerk wegen einer Reservierung oder Blockierung nicht sofort einfahren, so wird die Ausgangssteuerung *way_inx_oCtrl* des durchfahrenen Eingangsbausteines *way_in*x in die Tabelle *WaitL* des angestrebten Ausgangsbausteines *way_out* eingetragen. Bei jeder neuerlichen Aktivierung eines Ausgangsbausteines, die wiederum durch das Eintreffen eines Fahrwerkes am Ende der

nachfolgenden Blockstrecke ausgelöst wird, führt das System die in der zugehörigen Warteliste *WaitL* am längsten anstehende Steuerung aus, so daß nacheinander alle wartenden Fahrwerke durch die Weichen geschleust werden.

Zur Realisierung von Vorfahrtsregelungen kann über die Wegetabelle *WayDataL* bei Weichen und Zusammenführungen für jede mögliche Strecke ein Prioritätsattribut gesetzt werden, das standardmäßig mit dem Wert 1 hat. Bei gleichen Prioritäten wird nach den Prinzip FIFO verfahren. Bei unterschiedlichen Prioritäten werden die anstehenden Steuerungen entsprechend ihrer Priorität in die Tabelle *WaitL* eingereiht und nach der Aktivierung des Ausgangsbausteins nacheinander abgearbeitet. Der Wert 1 steht für die höchste Priorität.

Neben den Attributen für die Prioritätenregelung muß in Weichen und Zusammenführungen die Weglänge aller zu durchfahrenden Strecken angegeben werden. Während des Simulationslaufes wird die Bausteinlänge des anzufahrenden Ausgangs *way_out* mit den entsprechenden Werten gesetzt. Die Umschaltung auf Kurvengeschwindigkeit wird automatisch vorgenommen. In der Spalte *blockingtime* wird für jede Strecke eine Zeit einzutragen, nach der die Freigabe dieses Bausteines erfolgen soll. Diese Zeit beginnt ab dem Austritt des Fahrwerkes aus seinem Eingangsbaustein. Ein evtl. nachfolgendes Fahrwerk kann schon eher wieder in den Eingangsbaustein einfahren. Es kann ein aus einer anderen Richtung kommendes Fahrwerk kann schon eher wieder über einen Kreuzungspunkt zu einem anderen Ausgang fahren. Wird die Blockungszeit auf den Standardwert 0 gesetzt, liegt das normale Blockungsverhalten vor. Der Eingangsbaustein wird dann erst freigegeben, wenn das Fahrwerk am Ende seiner Blockstrecke angekommen ist. Die erwähnten Eingaben werden in der Tabelle *WayDataL* vorgenommen, die Weglänge der Eingänge wird in den Bausteinen direkt gesetzt.

	string 0	real 1	time 2	integer 3
string	route	distance	blockingtime	priority
1	way_in1-way_out1	6	0.0000	0
2	way_in1-way_out2	8	0.0000	1
3	way_in2-way_out2	6	0.0000	1
4				

Tabelle *WayDataL* eines Weichenbausteines

Pulkbildung vor Zusammenführungen

Um für zeitkritische Aufgaben den Durchsatz beim Überfahren von Zusammenführungen zu erhöhen, werden diese von hintereinander aus derselben Richtung kommenden Fahrwerken gruppenweise durchfahren. Damit wird ein ständiges Umschalten der Weiche im Falle von abwechselnd eintreffenden Fahrwerken vermieden. Dieses Verhalten kann mit dem Baustein *EOMjoinG* abgebildet werden, der die hintereinander aufgereihten Fahrwerke solange im Pulk durchfahren läßt, bis entweder eine Lücke auftritt, oder eine vorgegebene Gesamtanzahl erreicht wird. Auslöser einer Pulkfahrt ist ein blokkiertes Fahrwerk auf einem der Eingangswege *way_in1* oder *way_in2*, deren Steuerung nach Aktivierung des Austrittsbausteines *way_out* durchlaufen wird. Hier wird nun geprüft, ob hinter dem blockierten Fahrwerk inzwischen ein weiteres Fahrwerk eingetroffen ist. Im weiteren Verlauf werden die von diesen Fahrwerken durchfahrenen Eingangsbausteine, entsprechend der Freigabezeit *freeifgroup* aus der Tabelle *FreeifgroupL* schon vor Erreichen des Blockstreckenendes wieder freigegeben, so wie es auch schon bei der normalen Zusammenführung praktiziert wird. Im Gegensatz zu dieser, wird in dem Moment der Freigabe nun wiederum der Vorgänger auf ein austrittsbereites Fahrwerk überprüft, und dieses dann seinerseits dem Fahrwerkspulk zugeordnet, so daß sich die Pulkbildung nach hinten fortsetzt. Wenn die Fahrgeschwindigkeit über die Weiche bei der Kurvenfahrt stark reduziert wird, oder die Weglänge über die Weiche wesentlich länger ist als in den Eingangsbausteinen ist, könnten somit ständig neue Fahrwerke nachrücken. Um dies zu vermeiden, kann die Pulkgröße über die Variable *group_max* begrenzt werden. Die Freigabezeit bei Pulkfahrt kann für jeden Wegbaustein separat gesetzt werden. Beim Überfahren der Ausgangsbausteine wird der Gruppenstatus wieder aufgehoben.

Wenn keine Pulkfahrt vorliegt, arbeitet dieser Baustein wie eine normale Zusammenführung mit jeweils 2 zusätzlichen Blockstrecken vor den Eingängen und hinter dem Ausgang. Die Wegstrecken über die Weiche (die Länge des Weges *way_out*), die Blockungszeiten für das normale Blockungsverhalten und die Prioritäten werden in der Tabelle *WayDataL* vorgegeben. Die Weglängen aller anderen Wege werden direkt in den Bausteinen gesetzt.

string 0	string 0	time 1
	Building block	freeifgroup
1	way_in1	3.0000
2	way_in1_gp1	3.0000
3	way_in1_gp2	3.0000
4	way_in2	3.0000
5	way_in2_gp1	3.0000
6	way_in2_gp2	3.0000
7	way_out	3.0000
8	way_out_gp1	3.0000
9	way_out_gp2	3.0000

Tabelle *FreeifgroupL* in einem Pulkbaustein

Verhalten in Verzweigungen

In der vorliegenden Objektbibliothek sind Streckenverzweigungen in zwei und drei unterschiedliche Richtungen realisiert. Die auf den jeweiligen Abzweigestrecken zu erreichenden Ziele müssen in die zugehörige Zielliste *DestL* eingetragen werden. Die Länge der Ausgänge und die Blockungszeiten werden über die Tabelle *WayDataL* gesetzt, die des Einganges im Baustein selbst. Der Eingangsbaustein *way* wird nach dem Austritt des Fahrwerkes bis zum Erreichen des angestrebten Ausgangsbausteines *way_outx* oder bis zum Ablauf der Blockungszeit deaktiviert und anschließend wieder aktiviert.

Verhalten in Hubstationen

Hubstationen sind prinzipiell Verzweigungen in vertikale Richtung, nachdem sich die Fahrwerke zuvor den Heber zum Liften teilen müssen. Ankommende Fahrwerke benutzen die Hubschiene entweder zum Durchfahren als eigenständige Blockstrecke, oder zum Heben auf eine andere Etage. Während des Hubvorganges werden die entsprechenden anderen Einfahrwege blockiert. Dementsprechend wird die Ausgangssteuerung des blockierten Einfahrweges in die Warteliste *WaitL* eingetragen. Sobald der Lift wieder zur Verfügung steht, wird er vom blockierten Fahrwerk angefordert und reserviert. Sollten während des Hubvorgangs Fahrwerke auf beiden Eingängen eintreffen, wird über die benutzerdefinierte Vorfahrtregelung aus der Tabelle *RightofWayL* entschieden, wer zuerst Anspruch auf den Hubbalken hat. In dieser Tabelle werden die angestrebten Strecken der wartenden Fahrwerke paarweise miteinander verglichen, und abhängig von der aktuellen Position des Hubbalkens und der Priorität entschieden, wer bevorzugt anfordern darf. Bei gleicher Priorität wird entsprechend FIFO verfahren, ansonsten steht der Wert 1 für die höchste Priorität Die Länge sämtlicher Wege wird direkt in den Bausteinen gesetzt. Die Hubzeit *lift_time*, sowie eine evtl. anzufahrende Warteposition des Hebers wird über die globalen Variable *lift_waitPos* gesetzt.

string 0	string 0	integer 1	integer 2
	Route	lift_in_bottomPos	lift_in_topPos
1	way_inB-way_outB	1	1
2	way_inT-way_outT	1	1
3	----------		
4	way_inB-way_outB	1	1
5	way_inT-way_outB	1	1
6	----------		
7	way_inB-way_outT	1	1
8	way_inT-way_outT	1	1
9	----------		
10	way_inB-way_outT	1	1
11	way_inT-way_outB	1	1

Tabelle *RightofWayL* in einer Hubstation

Fahrwerkseinspeisung

Der Anwenderbaustein *VEHinput* dient zum Einbringen von leeren Fahrwerken in den Förderkreislauf. Leerfahrwerke können entweder in den Pufferbausteinen oder vor Be- und Entladestationen bzw. vor der kombinierten Dockstation in Warteposition erzeugt werden, oder werden direkt in der Quelle *VEHinput* als kreisende Vehikel ohne eigentliche Zielangabe in das Fördersystem gebracht. Die Fahrwerke im Puffer werden entsprechend der vorgegebenen Anzahl im jeweiligen Stapelabstand *stackLength* aufgereiht, während vor den Übergabestationen jeweils nur 1 wartendes Fahrwerk erzeugt wird.

Auftragseinlastung

Alternativ zur Kombination Fertigungssteuerung/Bearbeitungsstation können auch über den Baustein *CARGOinput* die auszuführenden Transporte vorgegeben werden. In einer Transportmatrix bilden die Beladestationen die Quellen, in denen die Aufträge eingelastet werden, die Senken werden durch die Entladestationen repräsentiert, in denen die Frachten nach dem Abladen vernichtet werden. Die Quellen sind der Zeilenindex und die Senken der Spaltenindex. Soll eine erstellte Matrix nicht verwendet werden, da zum Beispiel die Fertigungssteuerung verwendet wird, so muß die globale Variable *matrix_flag* auf FALSE gesetzt werden. Der einzulastende Teiletyp wird über die Variable *cargo* definiert. Die einzugebenden Transporteinheiten verstehen sich als Transporte/Stunde.

Die Transporte werden in der Beladestation über die Bausteine *cargo_source* und *cargo_enter* unter Berücksichtigung der Verweilzeiten in *ProcTimeL* nach *part_avail* überstellt, von wo sie von den Fahrwerken übernommen werden. Der Baustein *CARGOinput* kann entweder in der EHB-Zentrale oder auch direkt im Modell eingesetzt werden.

obje 0	object 0	integer 1	integer 2
		.EOMtest.EOMdock1	.EOMtest.EOMdock2
1	.EOMtest.EOMdock1		6
2	.EOMtest.EOMdock2	5	
3	.EOMtest.EOMdock3	12	
4	.EOMtest.EOMdock4		6

Transportmatrix *TransportM* in *CARGOinput* zur Auftragseinlastung

Beschreibung der Bausteine

Die Objektbibliothek Plant Simulation EOM ermöglicht es dem Benutzer, eine beliebige Fördersystem komfortabel in Plant Simulation nachzubilden.

Ziel der Entwicklung dieser Bausteine war es, eine möglichst flexible Grundlage für die Modellierung eines Transportweges zur Verfügung zu stellen. Die Bausteine sind daher so allgemein und einfach wie möglich gehalten, was ihre eigentliche Funktionalität betrifft. Die grundsätzlichen, für alle Problemstellungen gleichen Abläufe, sind bereits implementiert, so daß der Anwender nur anlagenspezifischen Details modellieren muß.

Die Bausteine *ProdMgr* und *Station* ermöglichen die Modellierung eine Fertigung nach Auftragslisten und Arbeitsplänen. Die Werkstücke in dieser Fertigung werden mit dem Baustein *part* abgebildet. Dieser Baustein gehört sowohl zum Transportsystem als auch zur Fertigung. Er wird im Kapitel 5 beschrieben.

Beschreibung der befahrbaren Bausteine

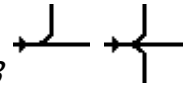
Die nachfolgend beschriebenen Bausteine werden von den Fahrwerken befahren. Für alle gelten die schon oben beschriebenen Verhaltensweisen bzgl. Blockungsverhalten, Räumstrategie, etc., so daß darauf im weiteren Verlauf nicht mehr explizit eingegangen wird. An dieser Stelle werden bevorzugt die vom Benutzer zu treffenden Aktionen beim Einsetzen in das Modell behandelt. In allen nachfolgenden Bausteinen muß vom Benutzer in den einzelnen Weg-elementen jeweils die richtige Länge eingetragen werden. Bei manchen Bausteinen (Weichen, Zusammenführungen und Verzweigungen) wird die Länge der Ausgänge über die Wegetabelle *WayDataL* gesetzt, da zu diesen Elementen noch weitere Eingaben erforderlich sind. Die Kapazität der Weg-Bausteine ist mit Ausnahme der Pufferstrecke *way_buff* immer mit 1 belegt und darf nicht verändert werden.

Gerade - *EOMstraight*

Mit diesem Baustein werden gerade Streckenelemente modelliert. Die im Weg eingetragene Länge entspricht immer genau einer Blockstrecke, da lediglich ein Fahrwerk aufgenommen werden kann. Dieser Baustein steht in vier Fahrtrichtungen zur Verfügung, wobei der Ursprungsbaustein mit einem Punkt im Icon versehen wurde.

Kurve - *EOMcurve*

Mit diesem Baustein werden die Kurven eines Fahrkurses abgebildet. Da in der Regel Kurven mit geringerer Geschwindigkeit durchfahren werden, wird in der Eingangssteuerung *way_iCtrl* die Kurvengeschwindigkeit über das freie Fahrwerkattribut *vCurve* verringert und diese in der Ausgangssteuerung *way_oCtrl* wieder auf den ursprünglichen Wert *vOri* zurückgesetzt. Wie bei der Geraden stehen auch hier mehrere Richtungen zur Verfügung, der Ursprungsbaustein ist markiert.



Verzweigungen - *EOMbranch2 / EOMbranch3*

Mit diesem Bausteinen können Verzweigungen in zwei oder drei Richtungen realisiert werden. Zur Erkennung aller Ziele, die von den Fahrwerken über die jeweiligen Ausgänge angefahren werden sollen, müssen diese über ihren absoluten Pfad in die zugehörigen Ziellisten *DestL* eingetragen werden. Über die Wegetabelle *WayDataL* werden die Längen der jeweils

zu durchfahrenden Blockstrecken, sowie die Blockungszeiten zur vorzeitigen Freigabe des Eingangsbausteines *way_in* eingetragen.

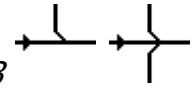
	object 1
1	.EOMsystem.EOMdock4
2	.EOMsystem.EOMload3
3	.EOMsystem.EOMunload3
4	.EOMsystem.VEHbuffer2
5	.EOMsystem.EOMload1
6	.EOMsystem.EOMunload1

Zielliste *DestL* in Verzweigungen

string 0	string 0	real 1	time 2
	Route	distance	blockingtime
1	way_in-way_out1	5	3.0000
2	way_in-way_out2	6	4.0000

Wegetabelle *WayDataL* in einer Verzweigung in 2 Richtungen

Zusammenführungen - *EOMjoin2 / EOMjoin3*



Diese Bausteine dienen zur Abbildung von Zusammenführungen aus zwei oder drei Richtungen. Beim Eintreffen mehrerer Fahrwerke aus unterschiedlichen Richtungen, hat standardmäßig das zuerst ankommende Fahrwerk Vorrang (FIFO-Prinzip). Soll nach einer Blockierung aus einer bestimmten Richtung jedoch bevorzugt eingefahren werden, greift die Prioritätenregelung (vgl. Absatz 3.5). Die gültigen Parameter werden in die Wegetabelle *WayDataL* eingetragen.

string 0	string 0	real 1	time 2	integer 3
	Route	distance	blockingtime	priority
1	way_in1-way_out	9	6.0000	1
2	way_in2-way_out	12	6.0000	2

Wegetabelle *WayDataL* bei einer Zusammenführung aus zwei Richtungen

Pulkbaustein - *EOMjoinG*

Im Pulkbaustein werden Gruppen von Fahrwerken (Pulks) gebildet. Dadurch wird ständiges Umschalten der Durchfahrtrichtung in Zusammenführung umgangen. Für die normale Durchfahrt ohne Gruppenbildung muß die Wegetabelle *WayDataL* ausgefüllt werden. Zusätzlich müssen die Freigabezeiten der einzelnen Bausteine in die Tabelle *FreeifgroupL* für Pulkfahrten eingetragen werden. Dadurch ist ein schnelles Durchfahren des gesamten Bausteines möglich. Weiterhin wird die Wartezeit des Nachfolgers bis zur Einfahrt verkürzt. Für jeden Wegbaustein kann eine unterschiedliche Freigabezeit gesetzt werden.

string 0	Building block	time 1 freeifgroup
0	way_in1	3.0000
1	way_in1_gp1	3.0000
2	way_in1_gp2	3.0000
3	way_in2	3.0000
4	way_in2_gp1	3.0000
5	way_in2_gp2	3.0000
6	way_out	3.0000
7	way_out_gp1	3.0000
8	way_out_gp2	3.0000
9		

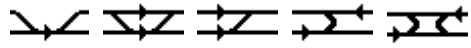
Tabelle *FreeifgroupL* in einem Pulkbaustein

Hubstation - *EOMlift*

In einer Hubstation eintreffende Fahrwerke fahren bei vorhandenem und freiem Hubbalken sofort ein. Entsprechend ihrem Zielort, der mit den Einträgen in der Zielliste *DestL* verglichen wird, fahren sie geradeaus oder werden auf das andere Niveau gehoben. Die während der Belegung des Hubbalkens ankommende Fahrwerke werden blokkiert und entsprechend ihrer Priorität in die Warteliste *WaitL* eingereiht. Nach erfolgter Liftfreigabe wird der Hubbalken vom höherwertigen Eingang angefordert und reserviert. Die Einträge in der Vorfahrttabelle *RightofWayL* sind vom Benutzer auszufüllen (abhängig von der aktuellen Liftposition und jeweils zwei möglichen Streckenkombinationen). Weitere Eingabeparameter sind die Hubzeit *lift_time*, sowie eine evtl. zu definierende Warteposition des Hubbalkens *lift_waitPos*, die im Ruhezustand eingenommen werden soll. Ist diese Variable nicht definiert, verbleibt der Balken nach der Fahrwerksausfahrt in seiner aktuellen Position.

string 0	Route	integer 1 lift_in_bottomPos	integer 2 lift_in_topPos
0	way_inB-way_outB	1	2
1	way_inT-way_outT	2	1
2	----------		
3	way_inB-way_outB	1	2
4	way_inT-way_outB	2	1
5	----------		
6	way_inB-way_outT	1	2
7	way_inT-way_outT	2	1
8	----------		
9	way_inB-way_outT	1	2
10	way_inT-way_outB	2	1
11			

Tabelle *RightofWayL* in einer Hubstation



Weichen - *EOMswitch1 ... EOMswitch5*

Für die Weichenbausteine gelten im Wesentlichen dieselben Vorschriften wie für die Zusammenführungen, mit dem Unterschied, daß die Vorrangfolge bei der Einfahrt im Falle von zwei Ausgangsbausteinen unterschiedlich gesetzt werden kann. Der Benutzer hat demzufolge die Möglichkeit, für jeden Eingangsbaustein *way_in* die Priorität zu jedem Ausgangsbaustein *way_out* über die Tabelle *WayDataL* zu setzen.

Über die Blockungsvariable *blockingtime* vom Datentyp *time*, hat der Benutzer die Möglichkeit, den blockierten Eingangsbaustein vor der Ankunft des Fahrwerkes am Blockstreckenende (Ausgangsbaustein) freizugeben. Bei Verwendung einer Blockungszeit muß natürlich beachtet werden, daß die eingetragene Zeit kürzer als die erforderliche Fahrzeit ist. Alle Ziele, die von den Fahrwerken über die jeweiligen Ausgänge angefahren werden sollen, müssen analog zu den Verzweigungen in die zugehörigen Ziellisten *DestL* eingetragen werden.

Puffer - *VEHbuffer* 

Im Pufferbaustein werden überzählige Leerfahrwerke zur weiteren Verwen-dung in Bereitschaft gehalten. Alle ankommenden Leerfahrwerke ohne Zielangabe oder mit dem Zielort des lokalen Puffers, fahren bei ausreichender Pufferkapazität in *way_buff* ein und werden dort entsprechend ihrem Stapelabstand aufgereiht. Jedes Fahrwerk, das an vorderster Pufferposition ankommt, durchsucht die vom Benutzer in der Liste *StationsL* eingetragenen Beladestationen nach abholbereiten Frachten. Da die Pufferausfahrt wie eine Zusammenführung aus zwei Richtungen betrachtet werden kann, ist auch in diesem Baustein die Vergabe von Prioritäten möglich. Die freien Attribute *prio2way_out* in den Weg-Grundbausteinen *way_buff* und *way_straight* müssen hierzu entsprechend ihrem Vorrang wie bei den Zusammenführungen belegt werden.

Fahrwerke, die aus Platzgründen nicht mehr in den Puffer einfahren können, suchen in der Pufferliste *BufferL* nach einem alternativen Puffer. Wird ein Puffer gefunden, so wird der Puffer zum neuen Zielort des Fahrwerkes. Ist in der Liste kein Eintrag vorhanden, kreist das Fahrwerk und sucht selbständig einen anderen Puffer.

Stationenliste *StationsL* und Pufferliste *BufferL*

Fahrwerkquelle - *VEHinput* 

Im Baustein *VEHinput* ist insbesondere die Fahrwerkstabelle *VEHtableL* von Bedeutung, da über deren Einträge die gewünschte Anzahl Fahrwerke erzeugt wird. Die Variable *v_typ* steht für den einzusetzenden Fahrwerkstyp, und *v_num* gibt die Anzahl vor. Die Erzeugung von kreisenden Fahrwerken erfolgt in der Quelle, wenn in der Spalte *v_source* der Pfad dieses Bausteins *VEHinput* eingegeben wurde. Neben der Erzeugung kreisender Fahrwerke, kann in einer beliebigen Anzahl von Puffern vom Typ *VEHbuffer* eine beliebige Anzahl von gestapelten leeren Fahrwerken erzeugt werden. Ein weitere Möglichkeit besteht in der Erzeugung von jeweils einem wartenden Fahrwerk vor einer Be-, Entlade oder Dockstation. In *v_source* muß stets der absolute Pfad eingetragen werden.

string 0	object 1 v_source	object 2 v_typ	integer 3 v_num
1	.EOMsystem.VEHinput	.veh	6
2	.EOMsystem.VEHbuffer1	.veh	2
3	.EOMsystem.VEHbuffer2	.veh	3
4	.EOMsystem.EOMload1	.veh	1
5	.EOMsystem.EOMdock4	.veh	1
6			

Tabelle *VEHtableL* zur Fahrwerkerzeugung

Beladestation - *EOMload*

Nach abgeschlossener Bearbeitung in der angegliederten Station oder bei Verwendung einer Transportmatrix wird die Fracht in den Eingangspuffer *part_avail* der Beladestation übergeben. Die Beladestation fordert über die Liste *StationsL* ein verfügbares Fahrwerk an. In diese Liste trägt der Benutzer die Pfade der vorgelagerten Puffer *VEHbufferx* oder Be-, Entlade- und Dockstationen *EOMload, EOMunload, EOMdock* ein. Dadurch hat ein von dort angefordertes Fahrwerk nur eine möglichst kurze Fahrstrecke.

Sollte dieser Gesichtspunkt keine Rolle spielen, kann die Liste auch leer bleiben, da vom System in diesem Fall über die Liste *VEHavailable* in der EHB-Zentrale automatisch in der gesamten Anlage nach einem verfügbaren Fahrwerk gesucht wird. Die Einträge in *StationsL* werden von der Methode *part_avail_oCtrl* sequentiell durchsucht.

	object 1
1	.EOMsystem.EOMunload2
2	.EOMsystem.EOMunload1
3	.EOMsystem.VEHbuffer1

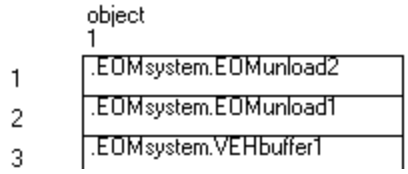
Liste *StationsL* für die Fahrwerksuche

Die erforderliche Zeit für die Beladung kann vom Benutzer in jeder Station über die globale Variable *loadtime* gesetzt werden. Angeforderte Fahrwerke werden in der Variablen *veh_required* registriert.

Entladestation - *EOMunload*

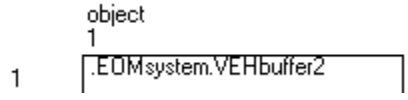
Beladene Fahrwerke übergeben nach Erreichen des Zielortes ihre Fracht an den Ausgangspuffer *part_unload* der Station. Hierfür wird jeweils die benutzerdefinierte Zeit *unloadtime* berücksichtigt. Wird die globale Variable *veh_wait* auf den Wert TRUE gesetzt, verbleibt das Fahrwerk nach dem Entladen vor der Station und wird in *EOMcontrol.VEHavailableL* registriert.

Ein weiterfahrendes Fahrwerk sucht über die Liste *StationsL* zunächst nach einer wartenden Fracht. Diese Liste wird vom Benutzer mit den Pfaden der nachfolgenden Beladestationen ausgefüllt. Dadurch wird bei der Leerfahrt ein möglichst kurzer Weg gewählt. Ist in diesen Stationen keine abholbereite Fracht, so wird die Suche automatisch über die Liste *CARGOwaiting* in der EHB-Zentrale auf alle Stationen der Anlage ausgedehnt.



Liste Stations*L* einer Entladestation für die Frachtsuche

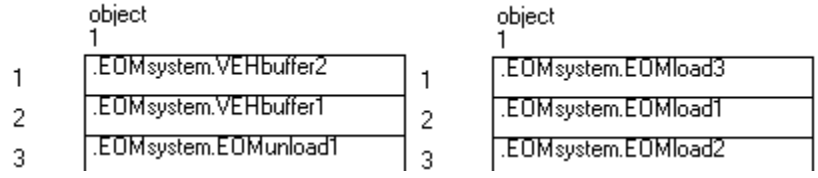
Ist diese Suche auch erfolglos, so sucht das leere Fahrwerk in der Liste *BufferL* nach einem Leerfahrwerkspuffer. In dieser Liste kann vom Benutzer der nächste anzufahrende Puffer bestimmt werden. Ist die Liste leer, kreist das Fahrwerk mit dem Zielort des obersten Netzwerkes.



Liste *BufferL* für Pufferzuweisung

Kombinierte Be- und Entladestation - *EOMdock*

Die kombinierte Station *EOMdock* vereinigt die Funktionalität einer Be- und einer Entladestation. In dieser Station greifen dieselben Dispositionsstrategien wie in den Einzelstationen. Dadurch kann dieser Baustein an eine Bearbeitungsstation angeschlossen werden. Weiterhin kann der Baustein als Quelle bzw. Senke zur Betreibung über die Transportmatrix verwendet werden. In diesem Baustein suchen abholbereite Frachten nach verfügbaren Fahrwerken und entladene Fahrwerke nach wartenden Frachten. Die Liste *StationsL* wurde durch die Listen *VEHStationsL* und *CARGOStationsL* ersetzt, in denen der Benutzer die zu durchsuchenden Stationen einträgt. Für die Fahrt eines entladenen Fahrwerkes zum nächsten Leerfahrwerkpuffer dient wiederum die Liste *BufferL*, die zu berücksichtigenden Be- und Entladezeiten werden über die globalen Variablen *loadtime* und *unloadtime* vorgegeben.



Listen *VEHStationsL* und *CARGOStationsL* einer Dockstation

Werkbankbetrieb - *EOMwork*

Beim Werkbankbetrieb wird ein Werkstück entweder am Fahrwerk hängend bearbeitet, oder es wird abgeladen, bearbeitet und auf dasselbe wartende Fahrwerk wieder aufgeladen. Nachrückende Fahrwerke werden solange blokkiert. Die Handlingzeiten werden über die globalen Variablen *unloadtime*, *worktime* und *loadtime* eingestellt. Werden die Werkstücke am Fahrwerk hängend bearbeitet, müssen die Ladezeiten auf Null gesetzt werden. In der Werkbank können keine Werkstücke vernichtet werden. In der Variablen *nexLocDest* muß das nächste Ziel eingetragen werden. Das Werkstück wird solange an eine Werkbank weitergereicht bis in einer Werkbank als nächstes Ziel eine Entlade- oder Dockstation angegeben ist. In einem solchen Baustein kann das Teil abgeladen und vernichtet werden.

Der Baustein *EOMwork* kann nicht in Verbindung mit einer Fertigungssteuerung verwendet werden.

Verwaltung der Fahrwerke

Die EHB-Zentrale *EOMcontrol* übernimmt die Aufgabe der übergeordneten Verwaltung von verfügbaren Fahrwerken und der zum Transport anstehenden Frachten, die nicht lokal in den einzelnen Stationen zugeordnet werden können.

EHB-Zentrale - *EOMcontrol* 

Hierfür werden Listen gehalten, deren Bearbeitung von den Schnittstellenmethoden übernommen wird. In der Tabelle *STATattachL* wird die Zuordnung der Bearbeitungs- und den Entlade- bzw. Dockstationen vorgenommen.

Die Methode *veh_setParams* in der EHB-Zentrale wird immer dann aufgerufen, wenn sich der Zustand eines Fahrwerkes ändert. Parameter wie Zielort, Dispositionsstatus und aktuelles Icon müssen neu gesetzt werden.

Die Räummethode *veh_clear* in der EHB-Zentrale wird von jedem Fahrwerk aktiviert, sobald es auf einen blockierten Baustein trifft. Wird die Blockierung durch ein vor einer Station in Warteposition befindliches Fahrwerke verursacht, so wird das wartende Fahrwerk weitergeschickt.

- *VEHavailableL* Liste der verfügbaren Fahrwerken, die einen Transportauftrag übernehmen können. Diese wartenden Fahrwerke stehen entweder austrittsbereit an erster Pufferposition oder sind im Wartezustand im Baustein *way* in einer Belade, Entlade- oder Dockstation. Die Methoden *veh_require*, *veh_register* und *veh_unregister* verwenden diese Liste.
- *CARGOwaitingL* Hier sind alle Frachten registriert, die auf ihren Weitertransport warten. Diese Liste wird von den Methoden cargo_*require*, cargo_*register* und cargo_*unregister* verwendet.
- *STATattachL*In dieser Tabelle findet die Zuordnung der Bearbeitungsstationen *GlobDestName* und der Übergabestationen *LocDestNameUnload* über die Methode *stat_glob2loc* statt. Jede eingesetzte Bearbeitungsstation muß zugeordnet werden.

string	object 0 GlobDestName	object 1 LocDestNameUnload
0		
1	.EOMsystem.Station1	.EOMsystem.EOMunload1
2	.EOMsystem.Station2	.EOMsystem.EOMunload2
3	.EOMsystem.Station3	.EOMsystem.EOMunload3
4	.EOMsystem.Station4	.EOMsystem.EOMdock4

Tabelle *STATattachL* zur Zuordnung der Stationen

- *CARGOinput*Über dieses Netzwerk werden alternativ zur Fertigungssteuerung die Aufträge über eine Transportmatrix vorgegeben. Die Beladestationen bilden die Quellen und die Entladestationen die Senken, wo die Fracht nach dem Abladen vernichtet wird. Zur Kennzeichnung des Teiles beim Vernichten, dient das freie Attribut *place*, das beim Einlasten über die Fertigungssteuerung immer 0, beim Einlasten über die Transportmatrix immer ungleich 0 ist. Die einzulastenden Aufträge werden in Transporten pro Stunde vorgegeben.

obje	object 0	integer 1	integer 2
0		.EOMsystem.EOMunload1	.EOMsystem.EOMunload2
1	.EOMsystem.EOMload1		12
2	.EOMsystem.EOMload2	20	
3	.EOMsystem.EOMload3	10	10
4	.EOMsystem.EOMdock4		12

Matrix *TransportM* in *CARGOinput* zur Auftragseinlastung

Fahrwerk – *veh*

Das Fahrwerk *veh* übernimmt entsprechend den anstehenden Fahraufträgen die Transportaufgaben des Fördersystems und durchfährt (unter Beachtung der Blockungs- und Räumungsalgorithmen) die Wegstrecken. Länge und Fahr-geschwindigkeit des Fahrwerkes sind vom Benutzer zu setzen. Zur weiteren Steuerung und Verwaltung der Fahrwerke, wurden folgende freie Attribute eingeführt:

vOri:

In diesem Attribut wird die ursprüngliche Fahrgeschwindigkeit gemerkt, auf die das Fahrwerk nach einer Kurvendurchfahrt für die Geradeausfahrt wieder zurückgesetzt wird. - Typ *speed;*

vCurve: (Benutzereingabe)

Kurvengeschwindigkeit, die bei Eintritt in die Kurve gesetzt wird. - Typ *speed;*

stackLength: (Benutzereingabe)

Dieses Attribut enthält die Stapellänge zur Stapelung der Fahrwerke in Puffern. Bei der Einfahrt in einen Puffer wird die Länge dieses Fahrwerks in der Eingangssteuerung *way_buff_iCtrl* auf diesen Wert und beim Verlassen in *way_out_iCtrl* wieder auf den ursprünglichen Wert gesetzt - Typ *length;*

fd_dispoStatus:

Entsprechend seinem aktuellen Fahr- und Beladezustand, befindet sich ein Fahrwerk in einem der folgenden Dispositions-Zustände - Typ *string;*

“free“	unbeladen im System unterwegs
“hasOrder“	unterwegs um eine Fracht zu übernehmen
“loaded“	beladen unterwegs zu einer Entladestation
“loading“	während eines Ladevorganges
“waiting“	unbeladen wartend vor einer Entladestation
“working“	wartend während der Bearbeitung in einer Werkbank
“inBuffer“	unbeladen wartend in einem Leerfahrwerkspuffer

lLoc:

Dieses Attribut registriert den durchfahrenen Eintrittsbaustein *way_in* in Weichen und Zusammenführungen. Der Eintrittsbaustein kann dadurch bei der Ausfahrt oder nach der Blockungszeit wieder freigeben werden. - Typ *object; clearRef:*

In diesem Attribut steht der Verweis auf die Räummethode *veh_clear* in der EHB-Zentrale *EOMcontrol*. In dieser Methode werden vor Entladestationen wartende Fahrwerke erkannt und bei Bedarf weitergeschickt - Typ *object;*

group:

Dieses Attribut dient als Flag zur Markierung eines Fahrwerkes im Falle der Gruppenzugehörigkeit bei einer Pulkfahrt. - Typ *boolean;*

Bausteinübergreifender Ablauf

Im folgenden sind die bausteinübergreifenden Vorgänge beschrieben, die eine wesentliche Funktionalität der Objektbibliothek in Verbindung mit der Fertigungssteuerung beinhalten.

Reset und Init

Durch Aktivieren von Reset im Ereignisverwalter wird das zugehörige Modell in einen definierten Ausgangszustand zurückgesetzt. In allen Bausteinen der Objektbibliothek werden hierbei sämtliche BEs gelöscht. Weiterhin werden alle dynamisch während des Simulationslaufs erzeugten Informationen in Listen, Tabellen und globalen Variablen entfernt.

In den Bausteinen der *Elektrohängebahn* sind davon alle während der Simulation vorgenommenen Einträge in den Listen der verfügbaren Fahrwerke, sowie der wartenden Frachten betroffen. Weiterhin werden alle Variablen die zur Steuerung benötigt werden zurückgesetzt.

In der Fertigungssteuerung *ProdMgr* betrifft dies Einträge über freie Kapazitäten und Zustände von Stationen, die Listen verfügbarer Stationen und die Listen von Teilen, die auf eine Operation warten. Ebenso wird die Tabelle mit den Fertigungsaufträgen gelöscht und neu aus der Vorgabe in der Tabelle *OrderL* erzeugt. In der Station sind die Inhalte der *NwDatas* und der Ergebnistabellen zu löschen.

Bei der Initialisierung wird das Modell aus dem Ausgangszustand in einen lauffähigen Zustand versetzt. In den Elementen der Objektbibliothek werden die Attribute der Bausteine mit den Eingabewerten der Benutzer initialisiert und die Erzeugung von Fahrwerken und Aufträgen angestoßen. In der Fertigungssteuerung *ProdMgr* wird hier die Methode *wc_startOrder* zum ersten Mal eingeplant, womit das Einlasten von Teilen nach der Auftragsliste angestoßen wird. Die Stationen melden sich beim Initialisieren bei der zugehörigen Fertigungssteuerung *ProdMgr* mit der freien Kapazität und dem Zustand *avail*.

Zielvergabe

Sobald ein Werkstück fertig bearbeitet ist, gelangt es in den Ausgangspuffer der Station. Dort wird in der Eingangssteuerung *st_PartReady* die Methode *wc_nextDestination* aufgerufen, um ein neues Ziel für das Teil zu bestimmen. Die Fertigungssteuerung bestimmt anhand des Arbeitsplans zu dem Teiletyp und der Arbeitsplanposition die nächste auszuführende Operation.

Zu der Operation muß nun eine Station bestimmt werden, die diese ausführen kann und die verfügbar ist. Aus der Tabelle *OperationL* werden alle Stationstypen bestimmt, welche die geforderte Operation ausführen können. In der Tabelle *StatTypeL* sind zu allen Stationstypen die jeweils verfügbaren Stationen aufgeführt. Anhand dieser Information wird eine Liste aller Stationen, die in Frage kommen, erstellt. Mit dieser wird die Methode *MD_StatChoose* aufgerufen, in der eine der möglichen Stationen ausgewählt wird.

Die ausgewählte Station wird von der Fertigungssteuerung belegt (Methode *ws_occupyStation*). Dabei wird die freie Kapazität der Station um 1 verringert und die Station ggf. aus der Liste der verfügbaren Stationen in *StatTypeL* entfernt. Die Adresse der Station wird dann an die aufrufende Station als neues Ziel für das Teil zurückgegeben.

War keine Station für die geforderte Operation verfügbar, so wird das Teil in der Tabelle *OperationL* als auf die jeweilige Operation wartend vorgemerkt. Als Rückgabewert wird *non_avail* an die anfragende Station zurückgemeldet, so daß das Teil dort festgehalten wird, bis ein Ziel ermittelt ist.

Die Liste der vorgemerkten Teile wird abgearbeitet, sobald eine Station sich bei der Fertigungssteuerung frei meldet (Methode *wc_StationFree*). Dies ist der Fall, wenn ein Teil in der Station fertig bearbeitet wurde. Falls Teile auf eine Operation der sich frei gemeldenden Station warten, wird eine Liste der wartenden Teile zusammengestellt und ein Teil ausgewählt.

5

HBW

Einleitung

Plant Simulation HBW ist eine Objektbibliothek zur Modellierung von Puffer- und Hochregallagern (HBW High Bay Warehouse) mit Plant Simulation.

Plant Simulation HBW stellt drei Varianten zur Verfügung, die sich im Detaillierungsgrad der Abbildung eines Lagers unterscheiden.

Variante 1:

Modellierung des Lagers als Black-Box, wobei keine Unterscheidung einzelner Lagergassen oder Artikel erfolgt.

WarehouseUT

- Lagerkapazität
- Ein- und Auslagerzeiten
- Verhalten bei Überfüllung und bei nicht erfüllbarer Auslageranforderung

Variante 2:

Modellierung des Lagers mit Lagerverwaltung (warehouse manager *WhMgr*) und Lagergassen (*Aisle*) einschließlich Regalbediengeräten (*RSU* Rack Serving Unit), jedoch ohne Berücksichtigung einzelner Lagerplätze. Zur Berechnung der Spielzeiten des *RSU* werden Lagerplätze mittels einer Zufallsverteilung ermittelt.

warehouse manager

- Liste der eingelagerten Teile
- Statistische Auswertungen
- Gassenbelegung

aisle

- Lagergasse:
  - Anzahl der Ebenen und Spalten
  - Ebenen- und Spaltenraster
  - Ein- und Auslagerposition in x/y Koordinaten
- Regalbediengerät (RSU):
  - Geschwindigkeit in x- und y-Richtung
  - Beschleunigung in x- und y-Richtung
  - Positionier- und Übergabezeit

Über den *WhMgr* sind Informationen über die eingelagerten Teile und die Belegung der einzelnen Lagergassen zugänglich. Zeiten für Last- und Leerfahrten sowie Warte- und Übergabezeiten des *RSU* werden statistisch erfasst. Dazu werden die technischen Parameter des *RSU* (Geschwindigkeit und Beschleunigung in x- und y-Richtung, Positionier- und Übergabezeit) und die geometrischen Daten des Hochregals (Anzahl Ebenen und Spalten der Regalgasse, Ebenen- und Spaltenraster und Aus- und Einlagerposition in x- und y-Richtung) durch den Benutzer eingestellt.

Variante 3:

Modellierung des Lagers mit Lagergassen samt Regalbediengeräten und Lagerverwaltung unter Berücksichtigung einzelner Lagerplätze innerhalb der Lagergassen und Lagerbereiche für A-, B- und C-Artikel.

Warehouse Management für ABC Artikel

- Liste der eingelagerten Teile (A, B, C)
- Statistische Auswertungen
- Gassenbelegung (A, B, C)

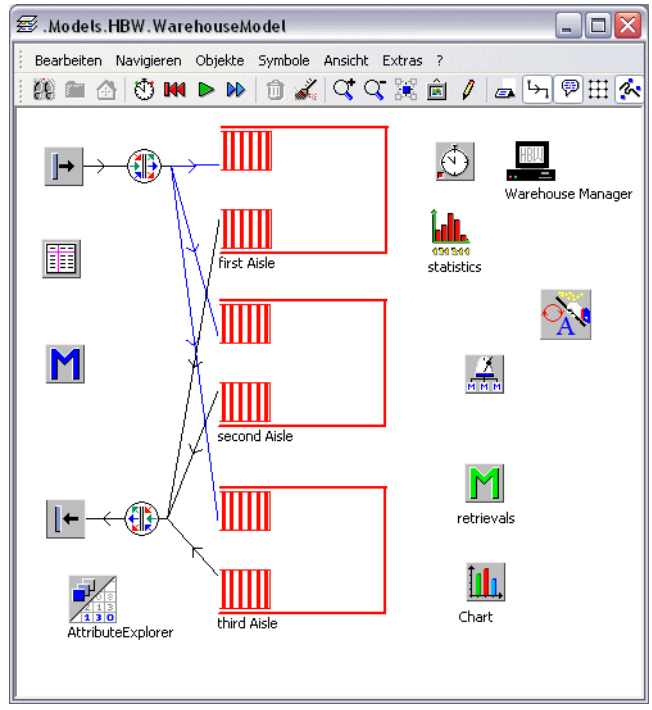
Lagergasse für ABC Artikel

- Lagergasse:
  - Anzahl der Ebenen und Spalten
  - Ebenen- und Spaltenraster
  - Ein- und Auslagerposition in x/y Koordinaten
  - Lagergebiete für ABC Artikel
- Regalbediengerät (RSU):
  - Geschwindigkeit in x- und y-Richtung
  - Beschleunigung in x- und y-Richtung
  - Positionier- und Übergabezeit

Alle Bausteine bieten eine Benutzerschnittstelle in Form von Dialogboxen, Popup-Menüs und Listen zur Eingabe von Daten. Die Funktionalität pro Baustein und das Zusammenwirken der Bausteine ist beschrieben. Hilfefunktionen erleichtern die Bedienung. Alle Bausteine sind offen für individuelle Anpassungen. Beispiele zur Anwendung der Objektbibliothek werden mitgeliefert.

Beispiele

Um die Funktions- und Arbeitsweise eines Lagers mit drei Hochregalen, die jeweils mit einem Regalbediengerät ausgerüstet sind, zu demonstrieren, ist ein Simulationsmodell zu erstellen. In verschiedenen Simulationsläufen soll der Einfluß der Anzahl der verschiedenen Lagerteile, der Häufigkeit von Ein- und Auslageranforderungen sowie der geometrischen Eigenschaften der drei Lagergassen untersucht werden.



Das entwickelte Modell

In diesem Kapitel werden die Schritte zur Modellerstellung zu der hier beschriebenen Simulationsstudie beschrieben. Das fertige Modell ist in der Bausteinbibliothek unter dem Namen *WarehouseModel* abgelegt. Ein mögliches Layout des Modells ist in Das entwickelte Modell gezeigt.

object 1	real 2	string 3	table 4
MU	Frequencies	Name	Attributes
.node	6.00	Part1	attr
.node	2.00	Part2	attr
.node	7.00	Part3	attr
.node	9.00	Part4	attr
.node	4.00	Part5	attr
.node	5.00	Part6	attr

string 1	integer 2	boolean 3	string 4
Name of Attribute			
type	1		
EntityType			part
currIcon			Typ2

Die Häufigkeitstabelle *Objecttypes* für den Grundbaustein *source*

Zuerst wird in der Randleiste ein leeres Netzwerk geschaffen und (zum Beispiel) in *WarehouseModel* umbenannt. Nach dem Öffnen des Netzwerkes wird ein Ereignisverwalter und ein Warehousemanager *WhMgr* und drei Bausteine *Aisle,* eine Quelle *Source*, eine Senke *Drain* und zwei Flußsteuerungsbausteine eingesetzt. Zur Registrierung aller Lagergassen setzen Sie den Assistenten *ASSI* ein und starten ihn.

Um die Anzahl und die Häufigkeit (englisch: frequences) der Lagerteile festzulegen, öffnen Sie den Quellenbausteine *Source*. In der Dialogbox stellen Sie in dem Popup-Menü Erzeugungszeitpunkte Abstand einstellbar und im Popup-Menü BE-Auswahl Zufaellig ein. Die Liste Objecttypes der zu erzeugenden Teile mit den gewünschten Häufigkeiten ist in Die Häufigkeitstabelle Objecttypes für den Grundbaustein source dargestellt. Wollen Sie die Einträge nicht manuell vornehmen, dann verwenden Sie die im Modell WarehouseModel enthaltene Methode fill.

Eine Simulation, in der nur Teile eingelagert werden, ist nun schon möglich. Um das Lager vollständig zu testen, müssen regelmäßig Auslagerungen angestoßen werden. Dazu wird ein *Generator* eingesetzt und eine Methode *retrievals* erstellt. Öffnen Sie den Generator und tragen Sie in die Abstandssteuerung die Methode *retrievals* ein. Soll die erste Auslagerung nach zehn Minuten erfolgen, so tragen Sie bei Start *konst* 10:00.00 ein. Geben Sie nun den Abstand zwischen zwei Auslagerungen ein, z.B. *konst* 1:00.00 (also 1 Minute). Drücken Sie nun die Schaltflächen Anwenden und OK.

```
is
     type : integer;
do
  type := floor(z_uniform(1,1,61));
  WhMgr.RetrievalDemand(„part"+to_str(type));
end;
```

Wollen Sie in regelmäßigen Abständen Informationen über die Fahr-, Warte- und Übergabezeiten des Regalbediengerätes *RSU* haben, so setzen Sie das Netzwerk *IntStat* in Ihr Modell ein. Aller zehn Minuten werden nun in den Statistiktabellen der Lagergassen und des Warehousemanagers die Zustandszeiten des *RSU* erfaßt. Die entsprechenden Tabellen können aus den Dialogboxen geöffnet werden.

Die Lagerkapazität kann über die Anzahl der Ebenen und Spalten eingestellt werden. Öffnen Sie dazu das Menü Daten der Regalgasse. In der 3. und 4. Zeile dieser Dialogbox werden die Abstände zwischen den Ebenen und Spalten in Meter eingegeben. Schließlich können die x- und y-Koordinaten der Ein- und Auslagerpositionen eingegeben werden. In dem Menü Daten des Regalbediengerätes werden die x- und y-Komponenten der maximalen Geschwindigkeit und der Beschleunigung des RSU festgelegt. Die x-Richtung ist die waagerechte und die y-Richtung die senkrechte Bewegungsrichtung des RSU. Beachten Sie, dass alle physikalischen Größen in den Einheiten einzugeben sind, die in eckigen Klammern in den Menüs vermerkt sind.

Beschreibung der Bausteine

Alle Bausteine besitzen eine Dialogbox, über die der Benutzer alle Einstellungen vornehmen kann, die für den Einsatz des Bausteins maßgebend sind. Tabellen mit statistischen Daten über das *RSU* sowie Daten zu eingelagerten Teilen (zweite und dritte Detaillierungsstufe) sind über diese Dialogboxen zugänglich. Die Dialogboxen öffnen sich nach einem Doppelklick auf dem Baustein in einem Netzwerk. In diesen Dialogboxen finden Sie einen Knopf Hilfe. Bei Betätigung wird dem Benutzer eine kleine Hilfe zu dem jeweiligen Baustein angeboten sowie Hilfen zu Eintragungen in der Dialogbox gegeben.

WarehouseUT

Symbol:

Der Baustein *WarehouseUT* besitzt je einen Ein- und Ausgang. Die Kapazität des Lagers, das Ein- bzw. Auslagerverhalten sowie die Verweilzeiten für den Ein- und Auslagerplatz sind einstellbar.

Funktionalität:

Die am Einlagerplatz ankommenden Teile werden bei der Umlagerung in das Lager in der Tabelle *Objectlist* verbucht. Die *Objectlist* enthält für jeden im Lager vorhandenen Typ eine Warteschlange mit den Lagerobjekten dieses Typs.

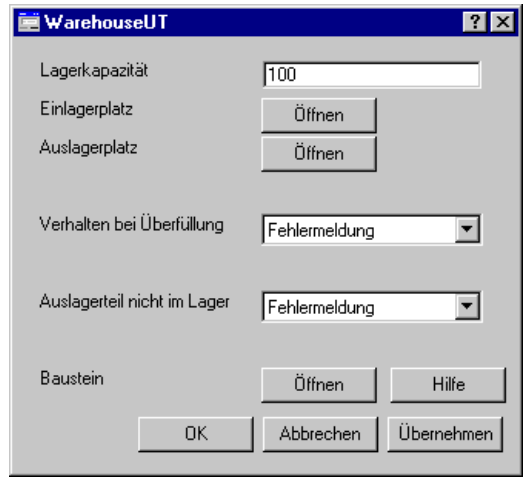
Sollte eine Einlagerung nicht möglich sein, so wird entweder eine Fehlermeldung ausgegeben oder aber die Lagerkapazität erhöht. Das Verhalten ist abhängig von den Einstellungen, die der Benutzer in der Dialogbox getroffen hat. Ist Dimension erhoehen gewählt, so wird die Lagerdimension um eins erhöht und zwar solange, bis der Benutzer in der Dialogbox auf Fehlermeldung umschaltet.

Benutzerschnittstelle:

Zur Auslagerung eines Objektes kann der Benutzer die im Baustein eingebaute Methode *RetrievalDemand* verwenden. Als Parameter wird der Typ des auszulagernden Teiles als String übergeben.

Über die Dialogbox lassen sich die Parameter des Einlagerplatzes, des Auslagerplatzes sowie die Kapazität des Lagers definieren.

Ebenfalls einstellen läßt sich das Verhalten des Bausteins, wenn das Lager voll ist und eine Einlageranforderung ansteht, bestimmen. Hier kann der Benutzer wählen, ob die Dimension des Lagers erhöht werden soll oder ob eine Fehlermeldung ausgegeben wird. Weiterhin kann der Benutzer über die Dialogbox einstellen, was geschehen soll, wenn eine Auslageranforderung ansteht, ein Teil dieses Typs aber nicht im Lager vorhanden ist. Hier hat der Benutzer die Möglichkeit der Auswahl zwischen Teil erzeugen und Fehlermeldung.



Dialogbox des Bausteins WarehouseUT

Bei einer Auslageranforderung wird zuerst geprüft, ob der Auslagerplatz *retrievalPlace* belegt ist. Wenn dies der Fall ist, wird der angeforderte Typ in die Auslagerliste *retrieval* angehängt. Anderenfalls wird geprüft, ob der Typ in der Objektliste vermerkt und damit im Lager ist.

Befindet sich der geforderte Typ im Lager, so wird das erste Teil aus der Warteschlange des entsprechenden Typs in der *Objectlist* entfernt und auf den Auslagerplatz *retrievalPlace* umgelagert. Es wird also immer das am längsten in Lager befindliche Teil ausgelagert. Ist die Warteschlange leer geworden, wird der Typ aus der *Objectlist* gestrichen.

Ist der geforderte Typ nicht im Lager, wird entweder ein Teil mit dem entsprechenden Typ auf dem Auslagerplatz erzeugt oder eine Fehlermeldung ausgegeben. Dies ist von den Einstellungen abhängig, die der Benutzer in der Dialogbox getroffen hat.

Verläßt ein Teil den Auslagerplatz *retrievalPlace* und befindet sich in der Auslagerliste noch ein Auftrag, so wird der erste Auftrag entnommen und das entsprechende Teil analog wie oben beschrieben entweder aus dem Lager entnommen oder neu erzeugt.

Lagermanager WhMgr

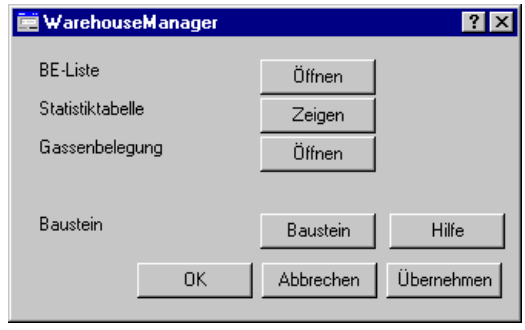
Symbol:

Der Baustein *WhMgr* (Warehousemanager) verwaltet die zu einem Lager gehörenden Lagergassen vom Typ *Aisle*. Durch einen *WhMgr* und mindestens einen Baustein *Aisle* kann ein Hochregallager dargestellt werden.

Funktionalität:

Im Warehousemanager *WhMgr* werden die zu dem Lager gehörenden Lagergassen sowie die eingelagerten Teile verwaltet. Weiterhin werden hier Auslageraufträge entgegengenommen und an die Gassen weitergeleitet.

In der Tabelle *MU_List* werden die eingelagerten Objekte verwaltet. Jeder Objekttyp im Lager hat einen Tabelleneintrag, der neben dem Typ einen Verweis auf eine Tabelle enthält, in der die Objekte dieses Typs in der Reihenfolge der Einlagerung mit Angaben über den Einlagerzeitpunkt und der Lagergasse enthalten sind. Beispiele werden im Kapitel 4 erläutert. Über die vom Anwender aufzurufende Methode *RetrievalDemand* werden Auslagerungen angestoßen. Ist der angegebene Typ im Lager, wird er in den Auftragspuffer *OrderBuffer* der Lagergasse vermerkt.



Dialogbox des Lagerverwaltungsrechners WhMgr

Ist das angeforderte Teil noch nicht im Lager, so wird es entsprechend der Einstellung in der Dialogbox entweder neu erzeugt (die Lagergasse, aus der das neu erzeugte Teil ausgelagert wird, wird durch Zufall bestimmt) oder aber es wird eine entsprechende Fehlermeldung ausgegeben.

Benutzerschnittstelle:

Zur Auslagerung eines Objektes kann der Benutzer die im Baustein eingebaute Methode *RetrievalDemand* verwenden. Als Parameter wird der Typ des auszulagernden Teiles als String übergeben.

Die Methode *MinUsedAisle* liefert die Nummer der Lagergasse, in der sich zur Zeit die wenigsten Teile befinden.

Die Methode *MaxUsedAisle* liefert die Nummer der Lagergasse, in der sich zur Zeit die meisten Teile befinden.

Die Methode *MaxEntityinWarehouse* gibt den Teiletyp zurück, der zur Zeit am häufigsten eingelagert ist.

Über die Dialogbox kann der Benutzer neben den Tabellen über die Zustandszeiten des *RSU* der Lagergassen und über die eingelagerten Teile diese 3 Werte durch Betätigung des Knopfes Gassenbelegung abfragen.

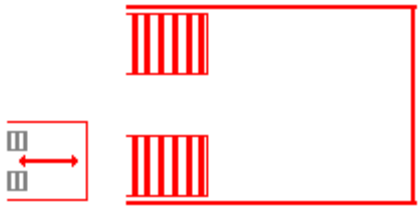
Es ist nicht zulässig, zwei Lagermanager innerhalb der gleichen Modellebene einzusetzen. Der zweite Baustein wird als zu löschen markiert, es wird eine Warnung ausgegeben.

Es ist nicht möglich, den Lagermanager interaktiv umzubenennen. (Die Lagergassen benötigen einen festen Namen für den Lagermanager, um diesen zu finden.)

Lagergasse Aisle

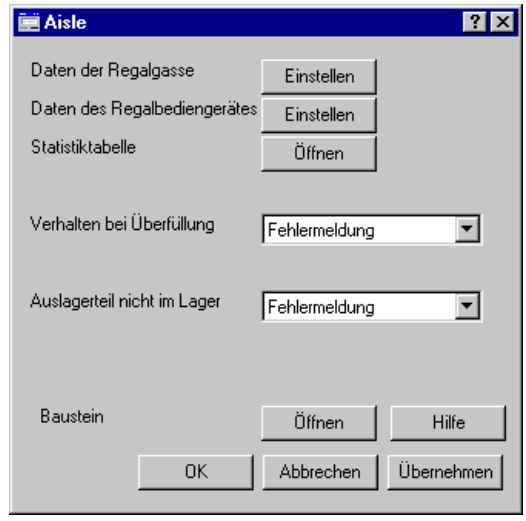
Der Baustein *Aisle* stellt eine Lagergasse dar, in der keine einzelnen Plätze verwaltet werden.

Symbole:



Benutzerschnittstelle

Über die Dialogbox kann der Benutzer das Ein- und Auslagerverhalten festlegen und die Tabelle mit den statistischen Daten des RSU anzeigen.



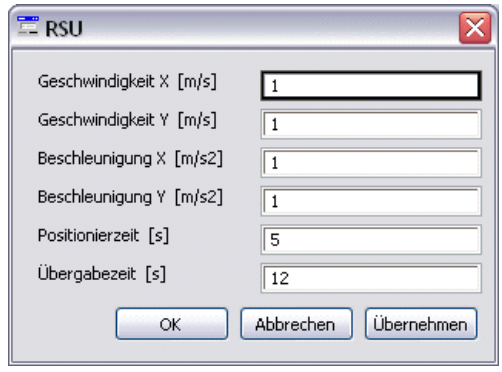
Dialogbox des Bausteins *Aisle*

Durch Drücken des Knopfes Daten der Regalgasse öffnet sich eine weitere Dialogbox, in der die geometrischen Daten der Regale (Anzahlen und Rasterungen der Ebenen und Spalten) und die x- und y-Positionen des Ein- und Auslagerplatzes eingestellt werden.

Hinter dem Knopf Daten des Regalbediengerätes verbirgt sich eine Dialogbox zur Einstellung der x- und y-Komponenten der Geschwindigkeit und Beschleunigung des RSU und der Positionier- und Übergabezeit.

Dialogbox der Regalgasse

Um die Daten des Regalbediengerätes zu ändern, muß man in der Dialogbox *Gasse* den Knopf einstellen hinter Daten des Regalbediengerätes betätigen, worauf folgende Dialogbox erscheint:



Dialogbox des Regalbediengerätes RSU

Funktionalität

Der Gassenbaustein übernimmt die eingehenden Teile auf dem Einlagerplatz *storingPlace*. Von dort wird das Teil vom *RSU* (sobald es frei ist) übernommen und zu dem Lagerplatz befördert. Zur Berechnung der entsprechenden Fahrzeiten des *RSU* wird ein Lagerplatz mittels einer Dreiecksverteilung zufällig bestimmt.

Auslageranforderungen werden nur dann bearbeitet, wenn der Auslagerplatz frei ist. Ist dies der Fall, fährt das *RSU* zu dem wiederum zufällig bestimmten Lagerplatz, übernimmt das Teil, fährt zum Auslagerplatz und gibt dort das Teil ab.

Bei der Ermittlung der Fahrzeit wird davon ausgegangen, dass das *RSU* immer Diagonalfahrten macht, d. h. es wird die längere der Fahrzeiten in x- und y-Richtung als Fahrzeit genommen. Hinzugerechnet wird noch die Positionierzeit.

Lagermanager WhMgr_ABC

Symbol: 

Benutzerschnittstelle

Zur Auslagerung eines Objektes kann der Benutzer die im Baustein eingebaute Methode *RetrievalDemand* verwenden. Als Parameter wird der Typ des auszulagernden Teiles als String übergeben.

Die Methode *MinUsedAisle* liefert die Nummer der Lagergasse, in der sich zur Zeit die wenigsten Teile befinden.

Die Methode *MaxUsedAisle* liefert die Nummer der Lagergasse, in der sich zur Zeit die meisten Teile befinden.

Die Methode *Art_distribMin* liefert die Nummer der Lagergasse, in der sich zur Zeit die wenigsten Teile eines als Parameter zu übergebenden Typs befinden. Gibt es mehrere Lagergassen mit der gleichen minimalen Anzahl, wird die Lagergasse zufällig unter den Lagergassen mit minimaler Anzahl bestimmt.

Die Methode *Art_distribMax* liefert die Nummer der Lagergasse, in der sich zur Zeit die meisten Teile eines bestimmten Typs befinden. Gibt es mehrere Lagergassen mit der gleichen maximalen Anzahl, wird die Lagergasse zufällig bestimmt.

Durch die Methoden *Is_A_article* und *Is_B_article* wird die Zugehörigkeit der Artikel zu den Artikelklassen A, B bzw. C festgelegt.

Die Dialogbox des Lagermanagers WhMgr_ABC hat das gleiche Aussehen wie die des WhMgr.

Funktionalität

Im Warehousemanagerbaustein werden die zu dem Lager gehörenden Lagergassen sowie die eingelagerten Teile verwaltet. Weiterhin werden hier Auslageraufträge entgegengenommen und an die Lagergassen weitergeleitet.

In der Tabelle *MU_List* werden die eingelagerten Objekte verwaltet. Jeder Objekttyp im Lager hat als Tabelleneintrag in dieser Liste eine Warteschlange, in der die Objekte dieses Typs in der Reihenfolge der Einlagerung mit Angaben über den Einlagerzeitpunkt und der Lagergasse enthalten sind.

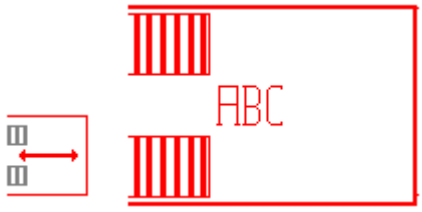
Über die vom Anwender aufzurufende Methode *RetrievalDemand* werden Auslagerungen angestoßen. Ist der als Parameter der Methode angegebene Typ im Lager, wird er in den *OrderBuffer* der Lagergasse eingetragen.

Ist das angeforderte Teil noch nicht im Lager, so wird der Typ in die Liste der unerledigten Auslageraufträge eingefügt.

Lagergasse Aisle _ABC

Der Baustein *Aisle_ABC* stellt eine Erweiterung der Lagergasse *Aisle* dar, in der die Lagerbelegung entsprechend dreier verschiedener Artikel A, B und C verwaltet wird. Bekanntlich werden Lagerteile nach bestimmten Eigenschaften klassifiziert, z. B. der Haltbarkeit in einem Lager. Teile, die häufiger ein- und ausgelagert werden, faßt man allgemein zu der Artikelklasse A zusammen. Um das Ein- und Auslagern von A-Artikeln schnell zu realisieren, werden deren Lagerplätze in der Nähe des Ein- bzw. Ausgangs des Lagers gewählt. Andererseits wird im Allgemeinen die Erledigung einer Einlageranforderung eines C-Artikels die meiste Zeit benötigen.

Symbole:



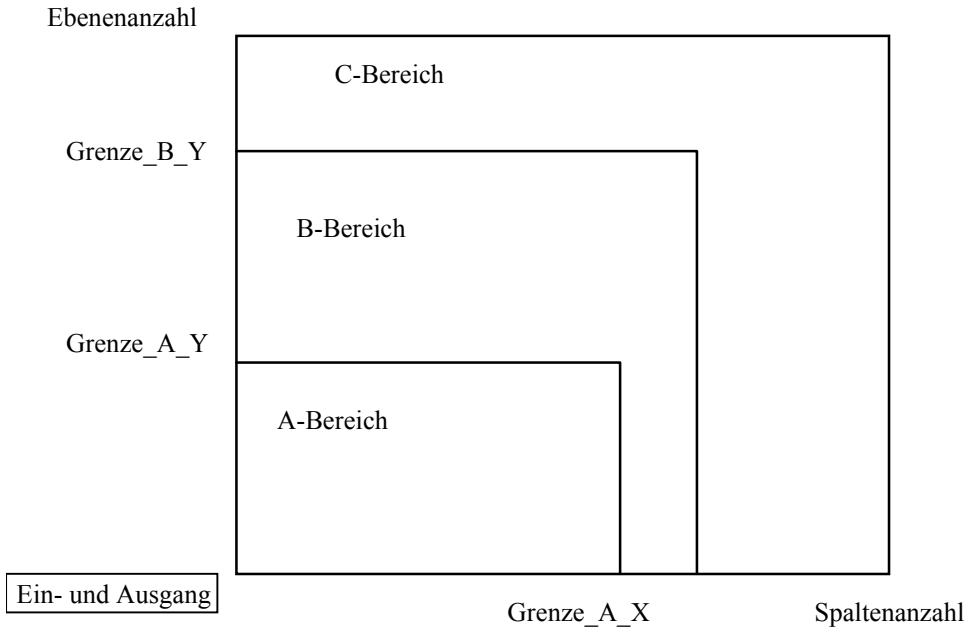
Benutzerschnittstelle

Über die Dialogbox kann der Benutzer die Daten der Regalgasse festlegen. Hierzu gehören die geometrischen Daten der Regale, die Positionen des Ein- und Auslagerplatzes sowie die Fahrwerte des RSU (Einzelheiten wie in 4.3.1).

Weiterhin können die Grenzen des Bereiches für die Lagerung von A- und B-Artikeln festgelegt werden. Die C-Artikel werden vorrangig in dem verbleibenden Teil gelagert (Dialogbox der Regalgasse).

Der Benutzer muß selbst auf die Einhaltung der folgender Ungleichungen achten:

0	<	limit_A_X	<	limit_B_X	<	NumberColumns (Anzahl der Spalten),
0	<	limit_A_Y	<	limit_B_Y	<	NumberFloors (Anzahl der Ebenen)



Verteilung der Lagerbereiche für A, B und C Artikel

Da die Gestaltung der Dialogbox zum Baustein *Aisle_ABC* der zum Baustein *Aisle* gleicht Dialogbox des Bausteins Aisle), wird auf deren Abbildung an dieser Stelle verzichtet.

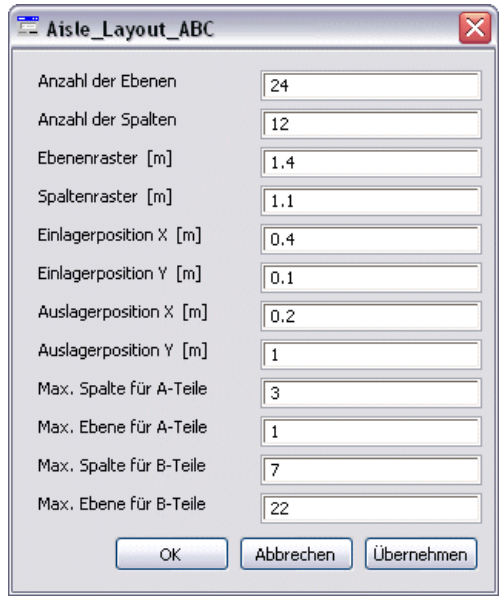
Zum Ändern der Daten der Regalgasse muß man den Knopf einstellen hinter Daten der Regalgasse betätigen. Dann öffnet sich die in Dialogbox Daten der Regalgasse für ABC-Artikel gezeigte Dialogbox, in der die Geometrie der Regalgasse eingegeben werden kann:

Um die Daten des Regalbediengerätes zu ändern, muß man in der Dialogbox *Aisle_ABC* den Knopf einstellen hinter Daten des Regalbediengerätes betätigen, worauf die gleiche Dialogbox erscheint, wie zum RSU von Aisle (Dialogbox des Regalbediengerätes RSU).

Funktionalität

Der Baustein übernimmt die eingehenden Teile auf den Einlagerplatz. Existiert in der Lagerverwaltung für diesen Typ eine unerledigte Anforderung, so wird das Teil vom *RSU* aufgenommen und direkt zum Auslagerplatz gebracht. Andernfalls wird das Teil vom *RSU* übernommen, sobald es frei ist und zu dem Lagerplatz befördert.

Der Lagerplatz wird entsprechend der vorgegebenen ABC-Einteilung ausgewählt. Dabei wird abwechselnd in den beiden Regalen der Lagergasse spaltenweise von unten nach oben der erste freie Platz innerhalb des vorgegebenen Bereiches gesucht.



Dialogbox Daten der Regalgasse für ABC-Artikel

Sollte ein Teil in dem für ihn vorgesehenen Bereich keinen Platz finden, kann er auch in einem anderen Bereich gelagert werden. Hierbei gilt die folgende Reihenfolge:

	für A-Artikel	für B-Artikel	für C-Artikel
erst	A-Bereich	B-Bereich	C-Bereich
dann	B-Bereich	C-Bereich	B-Bereich
zuletzt	C-Bereich	A-Bereich	A-Bereich

Auslageranforderungen werden nur dann bearbeitet, wenn der Auslagerplatz *retrievalPlace* frei ist. Ist dies der Fall, fährt das *RSU* zu dem Lagerplatz des auszulagernden Teils, übernimmt das Teil, fährt zum Auslagerplatz und gibt dort das Teil ab. Bei der Ermittlung der Fahrzeit wird davon ausgegangen, dass das *RSU* immer Diagonalfahrten macht, d. h. es wird die längere der Fahrzeiten in x- und y-Richtung als Fahrzeit genommen. Hinzugerechnet wird noch die Positionierzeit.

Anwendungen und Testmodelle

Erweiterungen und Einschränkungen

Für die Kombination der Lagerbausteine mit anderen Bausteinen gibt es aus der Sicht der Lagerbausteine keine Einschränkungen. Zu beachten ist lediglich, dass die eingelagerten Teile ein Attribut vom Typ String mit dem Namen *TeileTyp* besitzen müssen.

Bei nicht erfüllbaren Auslageranforderungen werden Teile vom Typ Objekt erzeugt, sofern der Benutzer dies in der Dialogbox so eingestellt hat. Die auf diese Weise erzeugten BEs sind bei der Animation rot markiert. Sollen im Einzelfall andere BEs im Lager gehalten werden, so sind Anpassungen der Methode *RetrievalDemand* in den entsprechenden Bausteinen vorzunehmen, damit die richtigen Teile erzeugt werden.

Testmodelle

Die Methoden *BuildUT, BuildAisle* und *BuildAisle_ABC* erzeugen die Modelle *Test_UT, Test_Aisle* und *Test_Aisle_ABC*. Diese Modelle verbinden ein Lager mit einer Quelle und einer Senke. Zum Auslagern eines Teiles ist die Methode *Retrieval* entsprechend zu verändern und zu starten. Es ist interessant, während der Simulation die Tabelle *MU_List* im Lagermanager mit den entsprechenden Warteschlangen und den *OrderBuffer* in dem Lager zu beobachten.

Demonstrationsmodelle

Im Modell *Demo* sind 3 Lagergassen eingebaut. Zum Auslagern wird die Methode *retrievals* verwendet, die über den Generator in Abständen von 10 Sekunden ausgeführt wird.

Die Tabelle, in der die Zustandszeiten des *RSU* gespeichert werden, ist in folgender Weise aufgebaut:

	string 0	time 1	time 2	time 3	time 4	time 5
string		Letzte Ereigniszeit	Warterzeit	Uebergabezeit	Lastfahrt	Leerfahrt
1	Gesamt	31:17.0000	11:24.000	10:12.0000	5:51.0000	3:50.0000
2	Uebertrag	31:05.0000	50.0000	12.0000	14.0000	5.0000
3	Zwischendaten	0.0000	0.0000			
4		10:00.0000	3:44.0000	3:12.0000	1:53.0000	1:11.0000
5		20:00.0000	7:06.0000	6:31.0000	3:44.0000	2:39.0000
6		30:00.0000	10:34.000	9:50.0000	5:51.0000	3:45.0000
7						

Die Statistiktabellen der Gassen

In den Spalten 2-5 sind die Zustandszeiten des *RSU* eingetragen. Der Zeitpunkt der letzten Veränderung der Tabelle ist an der Stelle (1,1) eingetragen. Darunter steht die Zeit der vorherigen Veränderung. Die Summe der Spalten 2-5 der 1. Zeile ergibt die letzte Ereigniszeit. Gleiche Eigenschaften haben die Zeilen 4,5,... . In diesen Zeilen werden in Zeitabständen von 10 Minuten die Zustandszeiten des *RSU* der jeweiligen Lagergasse ermittelt. Aus der in Die Statistiktabellen der Gassen gezeigten Tabelle geht hervor, dass zwischen 25:32 und 25:44 das *RSU* eine Übergabezeit von 12 Sekunden hatte.

Zu einem bestimmten Simulationszeitpunkt hat die *MU_List* im Baustein *WhMgr* die Einträge:

Teile Typ	Gassenliste
yellow	table21
green	table22
black	table23

Die *MU_List* enthält alle eingelagerten Teile.

Eine der (in der zweiten Spalte aufgeführten) Warteschlangen von BEs ist in der folgenden Abbildung dargestellt:

aisle	storing time	stored part
.Models.WarehouseModel.Aisle1	2:32.0000	.MUs.part:724
.Models.WarehouseModel.Aisle2	3:08.7000	.MUs.part:728
.Models.WarehouseModel.Aisle1	15:39.0000	.MUs.part:764
.Models.WarehouseModel.Aisle3	17:14.0000	.MUs.part:769
.Models.WarehouseModel.Aisle3	20:18.6000	.MUs.part:781
.Models.WarehouseModel.Aisle3	22:13.1000	.MUs.part:784
.Models.WarehouseModel.Aisle1	25:44.0000	.MUs.part:795
.Models.WarehouseModel.Aisle3	27:11.0000	.MUs.part:801

Eine Warteschlange aus der *MU_List*

Der *Diagram* Baustein stellt die Anzahlen der eingelagerten Teile in den einzelnen Lagergassen dar. Natürlich können auch andere Parameter der Simulation graphisch dargestellt werden.

Die Statistiktabelle des *WhMgr* faßt die Daten über die Zustandszeiten der *RSU* der Lagergassen in Zeitabständen von 10 Minuten zusammen:

Zeit	Gasse	Wartezeit	Uebergabezeit	Lastfahrt	Leerfahrt
10:00.0000	Aisle1	8:02.0000	1:12.0000	27.0000	19.0000
10:00.0000	Aisle2	9:14.0000	0.0000	27.5000	18.5000
10:00.0000	Aisle3	6:42.4000	0.0000	2:12.6000	1:05.0000
20:00.0000	Aisle1	16:05.0000	2:24.0000	49.0000	42.0000
20:00.0000	Aisle2	18:11.6000	0.0000	59.1000	49.3000
20:00.0000	Aisle3	13:09.3000	0.0000	4:34.6000	2:16.1000

Die Statistiktabelle des Lagermanagers

Die Daten der ersten Lagergasse sind in Die Statistiktabellen der Gassen im einzelnen dargestellt.

6

Shop Light

Die Bibliothek dient der Modellierung einer einfachen Produktionssteuerung.

Der Produktionsmanager ProductionManager

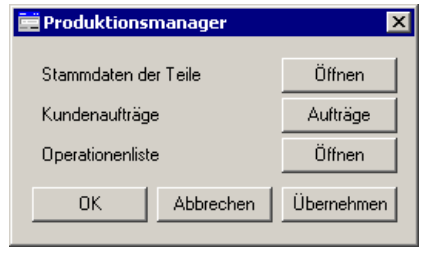
Symbol:

Funktionalität:

Der Produktionsmanager ist für den richtigen Ablauf der Bearbeitung der Teile verantwortlich. Hier ist für jedes Teil ein Arbeitsplan hinterlegt, ebenso die Operationen, die die einzelnen Stationen ausführen können.

Weiterhin können Sie hier die Menge der zu fertigenden Teile mit einem eventuellen frühesten Starttermin eingeben.

Benutzerschnittstelle:



Dialog des Bausteins ProductionManager

In der Fertigungssteuerung müssen vom Benutzer die Anzahl der Aufträge, die zu bearbeitenden Teile sowie deren Anzahl definiert werden. Ein Klick auf die entsprechende Schaltfläche öffnet folgende Tabelle:

string 0	string 0	table 1
	TeileTyp	ArbeitsplanL
1	part	Plan
2		

Die Tabelle der Stammdaten masterdataL

In die erste Spalte werden die zu erzeugenden Teiletypen eingetragen, in die zweite Spalte die Namen der entsprechenden Arbeitspläne. Durch einen Doppelklick auf den Eintrag öffnet sich eine weitere Tabelle, in der Sie die durchzuführenden Operationen in der Reihenfolge ihrer Abarbeitung eintragen.

Bei Aktivierung der Schaltfläche Aufträge öffnet sich eine Tabelle, in der die zu erzeugenden Teiletypen sowie die zu erzeugende Anzahl einzutragen ist.

AuftragsNr	TeileTyp	Anzahl	Freigabetermin	Faelligkeit
4711	teil	10		

Die Tabelle *OrderL*

Station

Symbol:

Der Baustein *Station* dient zur Nachbildung von Bearbeitungsstationen verschiedener Arten. Bearbeitungs- und Rüstzeiten können teile-, stations- oder faktorspezifisch festgelegt werden. Es kann nur ein Teil bearbeitet werden, wobei ein Eingangs- und Ausgangsplatz in der Station integriert sind.

Funktionalität:

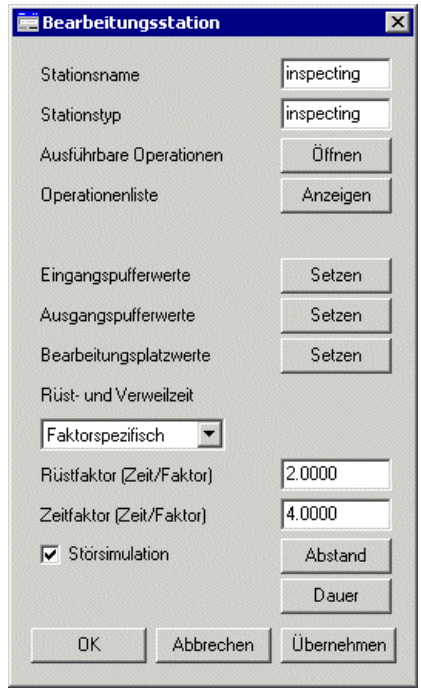
Kommt ein Teil in der Station an, so bleibt es dort eine definierte Zeit auf dem Rüstplatz (die Rüstzeit) liegen. Auf dem anschließenden Einzelplatz *Work* bleibt das Teil für die Bearbeitungszeit liegen.

Nach Ablauf dieser Zeit wird für das Teil ein neues Ziel bestimmt und es wird an das angeschlossene Transportsystem weitergereicht. Ohne Transportsystem wird das Teil direkt an die nächste Station geschickt.

Benutzerschnittstelle:

Beim Einsetzen des Bausteins öffnet sich ein Dialog, in den der Stationstyp eingetragen werden muß. Danach müssen die Operationen definiert werden, die die entsprechende Maschine ausführen kann.

Ein Klick auf die Schaltfläche Anzeigen neben Operationenliste öffnet eine Tabelle, in die sämtliche Operationen eingetragen werden, die von der Maschine durchgeführt werden können. Die hier eingetragenen Operationen müssen sich dann in den Arbeitsplänen wiederfinden. Beachten Sie bitte, daß Stationen gleichen Typs die gleichen Operationen ausführen. Die Operationslisten von Stationen gleichen Typs werden automatisch angepaßt.



Dialog des Bausteins *Station*

Die Werte (Kapazität, Verweilzeiten) des Eingangs- und Ausgangsplatzes können verändert werden. Hierzu stehen entsprechende Schaltflächen zur Verfügung, die selbsterklärend sind.

Weiterhin besteht die Möglichkeit, Störsimulationen zu aktivieren. Wenn das entsprechende Pop-up Menü auf ja gesetzt ist, können Dauer und Abstand der Störungen eingetragen werden. Diese Zeiten können natürlich auch zufällig gewählt werden.

Rüst- und Bearbeitungszeiten

Durch Auftreten von Rüstzeiten ist die Verwendung eines separaten Rüstplatzes innerhalb der Station notwendig. Rüstzeiten entstehen dadurch, daß sich entweder der Typ des Teiles, die Auftragsparameter oder der Arbeitsgang ändern. Die Rüstzeiten sind entweder im Arbeitsplan eingetragen oder stationsspezifisch festgelegt.

Der Benutzer hat drei Möglichkeiten, die Rüst- und Bearbeitungszeiten zu setzen:

- Eingabe im Arbeitsplan (über Fertigungssteuerung).
- Eingabe im Dialogfenster der Station.
- Eingabe im Dialogfenster der Einzelstation.

Im Dialogfenster der Station kann der Benutzer entscheiden, welche Zeiten gesetzt werden sollen, die des Arbeitsplanes (*teilespez.*), die der Station (*stationsspez.*) oder faktorabhängig *(faktorspez.)*.

Fällt die Wahl auf *stationsspez.*, so werden zwei Eingabefelder für die Rüst- und Bearbeitungszeiten sichtbar. Diese Zeiten sind unabhängig von dem aktuellen Teiletyp und der auszuführenden Operation.

Wird *faktorspez.* ausgewählt, werden ebenfalls zwei Eingabefelder sichtbar. Für den Rüst- und Bearbeitungsvorgang ist die Grundzeit zu hinterlegen. Diese Zeit entspricht dem Faktor 1. Sie kann z. B. die notwendige Zeit für eine Schweißnaht der Länge 1 m beschreiben.

Nach erneutem Aufruf des Dialogs werden die zuletzt eingegebenen Werte und die Art der Bestimmung von Rüst- und Bearbeitungszeiten angezeigt.

Störsimulation

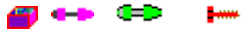
Maschinenausfälle, z.B. Störungen, Wartungen, etc. können durch eine Störung der Einzelstation *Work* in der Station abgebildet werden. Ist eine Station gestört, so wird dies der Fertigungssteuerung mitgeteilt. Diese wiederum schickt solange kein wartendes Teil an die Station, bis diese wieder entstört ist.

Um den Eingriff des Benutzers in den Baustein *Station* so gering wie möglich zu halten, können die Störzeiten im Dialogfenster der *Station* direkt eingegeben werden.

Im Dialog der *Station* kann gesetzt werden, ob Störungen simuliert werden sollen oder nicht. Ist das entsprechende Kontrollkästen selektiert, so können mit den zwei Schaltflächen für *Störabstand* und *Stördauer* die entsprechenden Verteilungen eingegeben werden.

Sie haben die Möglichkeit, Störbilder für die einzelnen Stationen zu setzen. Sollen diese Bilder bei einem Störfall angezeigt werden, so muß an den Bildnamen der ungestörten Station die Endung *_st* angefügt werden.

Das Teil part

Symbole: 

Der Baustein *part* wird bearbeitet und von den FTF Bausteinen transportiert. Der Baustein *part* ist mit folgenden Attributen ausgestattet:

- *globDestination*: Dieses Attribut enthält den Pfad zum Ziel des Teils, z.B. eine Bearbeitungsstation. Datentyp object.
- *Local*: Hier ist die Station eingetragen, an der das FTF letztmalig be- oder entladen wurde, Datentyp object.
- *OperationsPlanPos*: Die aktuell Position im Arbeitsplan ist hier eingetragen, Datentyp integer.
- *OperationsPlan*: Jedes Teil führt seinen Arbeitsplan mit, um unnötige Suchzeiten zu unterbinden, Datentyp table.

- *EntityType*: Der Teiletyp ist eingetragen. Anhand dieses Teiletyps wird aus der Tabelle masterdataL der Arbeitsplan bestimmt, Datentyp string.
- *DestDetermined*: Hier ist eingetragen, ob ein Ziel bereits ermittelt wurde oder nicht, Datentyp boolean.
- *LocDestination*: Der Pfad zum Ziel des Fahrzeugs, z. B. eine Andockstation. Datentyp object.

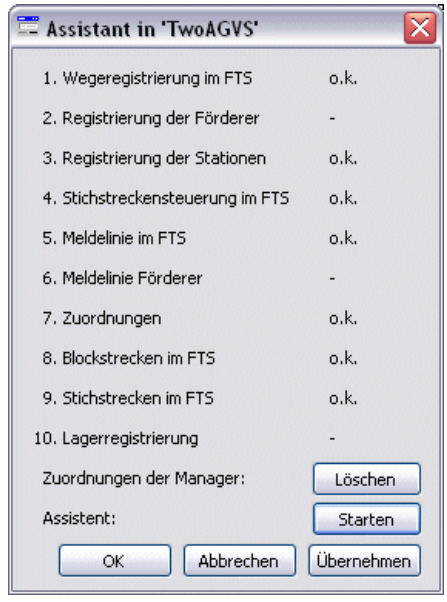
7

Modellierungshilfen

Der Assistent

Symbol: 

Wenn ein neues System aufgebaut wurde, müssen die Ziellisten aufgebaut werden. Dies ist wichtig, damit die FTF in den Verzweigungen entscheiden können, wohin sie fahren sollen. Ferner müssen die Entfernungen zwischen Verzweigungen und Zielen ermittelt werden, da diese Informationen für die Dispositionsregeln benötigt werden.



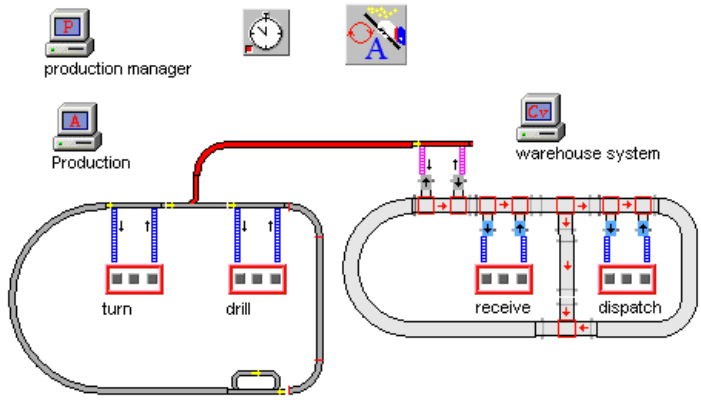
Der Assistent *ASSI*

- Mit Hilfe des Assistenten können folgende Funktionen automatisiert werden:
  - Gegenseitiges Anmelden der Wegelemente und Manager
  - Registrierung aller Bearbeitungsstationen beim Produktionsmanager
  - Erkennen der Stichstrecken (terminal lines)
  - Erzeugen der Wegweiserlisten in Verzweigungen (register line)
  - Erkennen der Verbindungen zwischen Transport- und Produktionssystem
  - Erkennen der Bereiche der Blockstrecken und deren Eingänge
  - Erzeugen der Wegweiserlisten in Stichstrecken (terminal lines)

Wenn sich am Layout oder an einer Weglänge etwas ändert, so müssen diese Entfernungen und Ziellisten neu erstellt werden. Dies wird vom System erkannt und bei einem Neustart werden die Meldelinien neu gestartet.

Schnittstellen zwischen verschiedenen Transportsystemen

Mit den Bausteinkästen AGVS und Conveyor können komplexe Transportsysteme mit mehreren Fahrkursen unterschiedlichen Typs modelliert werden. Der Materialaustausch zwischen den Fahrkursen erfolgt über die Schnittstellenbausteine. Um ein Transportsystem mit zwei FTS-Kursen zu modellieren, wird für jeden Kurs ein eigener *AGVS_manager* benötigt. Der Produktionsmanager *ProductionManager* muß sich über diesen Netzwerken befinden. Der Modellaufbau ist aus der folgenden Abbildung ersichtlich:



FTS-System mit zwei Fahrkursen

Um die Zuordnung der beiden Manager zu den Wegsystemen gezielt vornehmen zu können, setzen Sie mit der Schaltfläche Reset den Assistent *ASSI* zurück. Tragen Sie in ein beliebiges Wegelement den gewünschten Manager ein. Der Assistent ordnet alle verbundenen Wegelemente diesem Manager zu.

Die abgebildete Tabelle *rgAssignL* zeigt die Zuordnungen der Bearbeitungsstationen zu den anfahrbaren Zielen in den jeweiligen Wegeteilsystem. Die lokalen Ziele des zu *AGVS_manager* gehörende Wegesystems sind die Bausteine, die ein FTF in diesem Wegesystem anfahren kann. Soll z.B. ein Teil zu der Bearbeitungsstation *station_A_T* transportiert werden, so kann direkt die angeschlossene Andockstation *Dock_A_T* angefahren werden.

.Models.Integration.agvs_manager.rgAssignL

.Models.Integration.station_A_T

	destinations	local destination
1	.Models.Integration.station_A_T	.Models.Integration.Dock_A_T
2	.Models.Integration.station_A_D	.Models.Integration.Dock_A_D
3	.Models.Integration.station_Cv_R	.Models.Integration.agvs_interface
4	.Models.Integration.station_Cv_D	.Models.Integration.agvs_interface

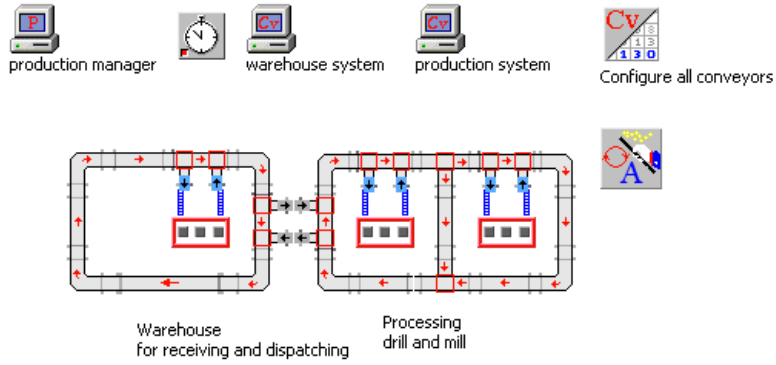
OK | Abbrechen | Übernehmen

Die Tabelle *rgAssignL* von *AGVS_manager*

Das Ziel eines Teils *station_Cv_D* kann nur über die Schnittstelle *agvs_interface* erreicht werden.

Schnittstellen zwischen Transportsystemen

Mit den Bausteinkästen Plant Simulation Conveyor und Plant Simulation AGVS können komplexe Transportsysteme unterschiedlichen Typs modelliert werden. Der Materialaustausch zwischen den Fahrkursen erfolgt über die Schnittstellenbausteine *cv_interface*. Die Fertigungssteuerung *ProductionManager* muß sich über diesen Netzwerken befinden. Der Modellaufbau ist aus der folgenden Abbildung ersichtlich. Das Modell *interface* wird durch die Methode *.buildDEMOS.buildInterface* aufgebaut.



Netzwerke des Stetigfördersystems mit zwei Kursen

Um die Zuordnung der beiden Manager zu den Wegsystemen gezielt vornehmen zu können, setzen Sie mit der Schaltfläche Reset den Assistent *ASSI* zurück. Tragen Sie in ein beliebiges Wegelement den gewünschten Manager ein. Der Assistent ordnet alle verbundenen Wegelemente diesem Manager zu.

Die in der Abbildung Die Tabelle rgAssignL von cv_Manager2 gezeigte Tabelle *rgAssignL* zeigt die Zuordnungen der Bearbeitungsstationen zu den anfahrbaren Zielen in den jeweiligen Wegteilsystem. Die lokalen Ziele des zu *cv_Manager2* gehörende Wegsystems sind die Bausteine, die ein Teil in diesem Wegsystem erreichen kann. Soll z.B. ein Teil zu der Bearbeitungsstation *station21* transportiert werden, so kann direkt die angeschlossene Andockstation *cv_dockingStation21* angefahren werden.

	object 0	object 1
string 0	destinations	local destination
1	.interface.station11	.interface.cv_interface1
2	.interface.station21	.interface.cv_dockingStation21
3	.interface.station22	.interface.cv_dockingStation22

Die Tabelle *rgAssignL* von *cv_Manager2*

Das Ziel eines Teils *station11* kann nur über die Schnittstelle *cv_interface1* erreicht werden.

About Siemens PLM Software

Siemens PLM Software, a division of Siemens Automation and Drives (A&D), is a leading global provider of product lifecycle management (PLM) software and services with 4.6 million licensed seats and 51,000 customers worldwide. Headquartered in Plano, Texas, Siemens PLM Software's open enterprise solutions enable a world where organizations and their partners collaborate through Global Innovation Networks to deliver world-class products and services. For more information on Siemens PLM Software products and services, visit www.siemens.com/plm.

SIEMENS

Division headquarters

United States
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
972 987 3000
Fax 972 987 3398

Regions

Americas
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
800 498 5351
Fax 972 987 3398

Europe
Norwich House Knoll Road
Camberley, Surrey
GU15 3SY
United Kingdom
44 1276 702000
Fax 44 1276 705150

Asia-Pacific
Suites 6804-8, 68/F., Central Plaza
18 Harbour Road, WanChai
Hong Kong
852 2230 3333
Fax 852 2230 3210

10/07