3 Anforderungsbeschreibung und -analyse

Dieses Kapitel stellt die aufgenommenen Anforderungen und die Ergebnisse der Anforderungsanalyse dar. Ausgehend von der Darstellung der Anforderungen soll aufgezeigt werden, welche Komponenten des realen Systems[1], Prozesse und Schnittstellen in einem späteren Design zu beachten sind.

Ein Simulator unterstützt den Anwender bei der Modellbildung und der Ausführung des Modells. Die Spanne der Simulatoren reicht von einer Simulationssprache über eine Modellerstellungs- und Experimentierumgebung bis hin zu einer Simulatorenentwicklungsumgebung [VDI96, S. 16]. Die Ansätze differieren deutlich bezogen auf den Umfang ihres Anwendungsbereiches und ihre Benutzerfreundlichkeit (vgl. Abschnitt 2.2.2). Daher stellt der erste Abschnitt heraus, welche Funktionen der zu entwickelnde Simulator erfüllen muss. Hierbei wird insbesondere auf den späteren Verwendungszweck des Simulators eingegangen. Ausgehend von den gestellten Anforderungen wird eine Architektur des Programms ausgearbeitet, die einzelne Bereiche des Simulators abgrenzt und somit eine Arbeitsteilung bei der Entwicklung erlaubt (Abschnitt 3.2).

Im Rahmen meiner Diplomarbeit habe ich mich auf die Entwicklung des Simulatorkerns konzentriert. Dieser Programmteil befasst sich vorwiegend mit den Modellelementen, die zur Abbildung des realen Systems zur Verfügung stehen und der zeitlichen Fortschreibung der eintretenden Ereignisse [VDI96, S. 16]. Somit werden im Abschnitt 3.3, als ein Bestandteil der Anforderungsanalyse, die spezifischen Modellelemente dezentral gesteuerter schienengebundener Transportsysteme erläutert. Neben den statischen Beziehungen zwischen den einzelnen Systemkomponenten werden zusätzlich Sequenzdiagramme angegeben, die dynamische Prozesse im System veranschaulichen. Des Weiteren wird auf die notwendige Kommunikation der einzelnen Ebenen des Simulators eingegangen.

Besonders schwierig stellte sich zu Beginn des Projektes die Behandlung einzelner Fachbegriffe heraus, die im Zusammenhang mit den Komponenten schienengebundener Transportsysteme stehen. So wurde z. B. die Vorrichtung zur Identifikation der Fahrzeuge des Transportsystems zumeist als „*Transponder*“ bezeichnet. Nach [Geh99, S. 143] ist ein Transponder aber nur eine Art einer Identifikationseinrichtung. Weitere sind Barcodeleser und der Einsatz einer induktiven Codierung. Im Zusammenhang mit der Ermittlung der Modellelemente erscheint mir eine Festlegung der Terminologie wichtig. In Abschnitt 3.3.1 werden im Rahmen der Analyse der Systemkomponenten die entsprechenden Fachbegriffe erläutert.

[1] Da der ein Simulator für dezentral gesteuerte Schienensysteme entwickelt werden soll, wird das reale System als ein Materialflusssystem angesehen.

3.1 Anforderungsbeschreibung

Die Anforderungen, die der Simulator erfüllen muss, sind vorwiegend von den Mitarbeitern der FASTEC gestellt worden, mit denen während dieser Diplomarbeit zusammengearbeitet wurde. Weiterhin werden Forderungen aus dem ISILEIT-Projekt berücksichtigt, so dass es möglich ist, den entwickelten Simulator, speziell den in dieser Arbeit erstellten Teil, in diesem Projekt zu verwenden.

Die FASTEC veräußert Steuerungen für Materialflussanlagen, die dezentral gesteuert und schienengebunden sind. Das Unternehmen ist überwiegend an einem Instrument interessiert, das zur Kontrolle der entworfenen Steuerung bereits vor dem Aufbau der Anlage eingesetzt werden kann. Gerade in diesem Bereich besteht ein enger Zusammenhang zu einem Einsatz des Simulators im ISILEIT-Projekt (vgl. Abschnitt 2.1.2). Während der in FUJABA integrierte Simulator bereits eine schematische Darstellung enthält, soll eine anschauliche Virtual Reality Darstellung das Verständnis der Simulation verbessern. Weiterhin möchte die FASTEC ihren Kunden die Möglichkeit geben, benutzerfreundlich und kostengünstig eine Simulation der jeweiligen Anlage durchzuführen.

Bisher wurden die Rahmenbedingungen aufgezeigt, die vor der Entwicklung des Simulators bestanden. Zur genaueren Analyse der Anforderungen ist es jedoch notwendig, die allgemeinen Beschreibungen *benutzerfreundlich* und *Einsatz zur Kontrolle der Steuerung* detaillierter zu betrachten. Während die Benutzerfreundlichkeit sich eher auf die Softwareergonomie (insbesondere die Gestaltung der Benutzungsoberfläche) bezieht, fordert die Steuerungskontrolle Aspekte ein, die unmittelbar mit der Modellierung des realen Systems zusammenhängen. Zumal ich mich in meiner Diplomarbeit vorwiegend mit der Modellierung der Systemkomponenten und den Prozessen beschäftige, werden hier nur grundlegende Hinweise auf eine benutzerfreundliche Gestaltung gegeben.

Benutzerfreundliche Gestaltung

Die Benutzungsoberfläche des Simulators soll es dem Anwender erlauben, seine Anlage einfach zu konfigurieren. Hierzu soll er vorgefertigte Streckenmodule, die in seinem Materialflusssystem eingesetzt werden, auswählen und miteinander zu einer entsprechenden Streckentopologie zusammensetzen können. Neben der Topologie soll der Benutzer in der Lage sein, Steuerungsknoten zu platzieren und eine Zuordnung der Anwendungsprogramme zu den Knoten vornehmen zu können. Damit sich der Entwickler der Simulationsstudie schnell mit den Modellelementen zurechtfinden kann, ist eine ansprechende Darstellung zu wählen. Im Falle einer Kundenpräsentation ist hierbei eine VR-Darstellung angebracht.

Kontrolle der programmierten Steuerung

Die Steuerung der Materialflusssysteme besteht in dezentral gesteuerten Systemen aus einzelnen Steuerungsknoten, die über ein Netzwerk miteinander verbunden

sind. Jeder Knoten erhält die lokale Information von seinem Prozessinterface. Die Reaktion auf die Information ist abhängig von dem Anwendungsprogramm, das auf dem Steuerungsknoten implementiert wurde. Zur Kontrolle der Steuerung soll es möglich sein, den Einfluss des Anwendungsprogramms auf den Materialfluss abzubilden. D. h., dass Objekte, die nach Aufgabenstellung in Java implementiert sind, Informationen über das System erhalten und der Einfluss dieser Objekte auf das modellierte Transportsystem erlaubt wird.

Weitere Anforderungen

Neben den bereits angeführten Forderungen bestehen noch weitere Punkte, die zu beachten sind. Wesentlich ist hier vor allem die Laufzeit des Simulationsmodells. Es ist zu berücksichtigen, dass ein Simulationslauf schneller abläuft, als das System in der Realität, um eine Darstellung in Echtzeit zu erlauben.

Der Simulator soll auf Grundlage des in Abschnitt 3.3 zu analysierenden Transportsystems entwickelt werden. Neben der Modellierung spezifischer Eigenschaften ist darauf zu achten, dass er um weitere Systemkomponenten erweitert werden kann.

3.2 Anwendungsarchitektur des Simulators

Zumal der Simulatorkern[2] auch Anwendung im ISILEIT-Projekt finden soll, ist eine Trennung des Kerns von der grafischen Darstellung und den Steuerungsobjekten sinnvoll. Die Klassen des Simulatorkerns können so als Klassenbibliothek bereitgestellt werden. Hierdurch erhöht sich die *Wiederverwendbarkeit* dieses Porgrammteils. In Bild 3-1 wird ein grober Überblick über die Architektur[3] des Simulators gegeben.

Die angegebene Struktur berücksichtigt eine Unterteilung der gegebenen Forderungen. Die Aufgaben einer benutzerfreundlichen Gestaltung werden von der Benutzungsoberfläche bearbeitet, während auf der Ebene der Steuerungsobjekte die Steuerungsknoten modelliert werden. Die Architektur verdeutlicht, dass der Simulatorkern nur auf Daten der Modellumgebung arbeitet. Diese sind von den anderen beiden Teilbereichen zu erstellen. Auf der Ebene der Benutzungsoberfläche konfiguriert der Entwickler die zu simulierende Anlage. Die Konfiguration bezieht sich sowohl auf die Streckenführung als auch auf technische Daten, wie bspw. die Geschwindigkeiten der Fahrzeuge. Die Steuerungsobjekte beeinflussen die Modelldaten des Kerns abhängig von den zu bearbeitenden Aufträgen und

---

[2] *„Der Simulatorkern bezeichnet den Programmteil, der die Modellwelt mit ihren Modellelementen bereitstellt und die automatische, chronologische Erzeugung der Ereignisse, die zur korrekten Abbildung eines Prozeßablaufes im Modell benötigt werden, verwaltet. …“ [VDI96, S.16]*

[3] Architektur: Spezifikation der grundlegenden Struktur eines Systems [Oes98, S. 341]

Steuerungsregeln. Auf die Entwicklung des Simulatorkerns und die sich aus der Architektur ergebenen Schnittstellen wird in den folgenden Abschnitten eingegangen. Zuvor wird jedoch im nächsten Abschnitt behandelt, mit welchen Werkzeugen der Simulator entwickelt wird. Die bei der Entwicklung des Simulator eingesetzten Werkzeuge wirken sich unmittelbar auf das weitere Vorgehen aus, zumal diese die Flexibilität in der Modellierung und den Programmieraufwand vorgeben (vgl. Abschnitt 2.2.2).

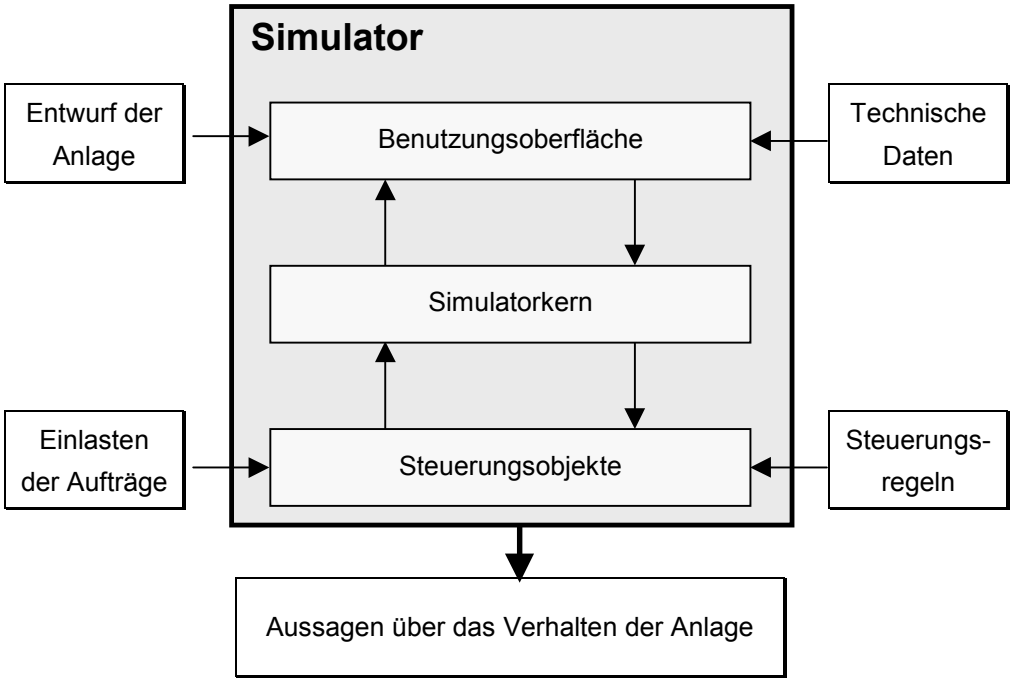


Bild 3-1: Die Anwendungsarchitektur des Simulators trennt die Entwicklung des Simulators in die drei Teilbereiche der Benutzungsoberfläche, den Simulatorkern und der Steuerungsobjekte auf.

3.2.1 Auswahl der Entwicklungsumgebung

Wie im Abschnitt 2.2.2 bereits diskutiert gibt es zur Unterstützung bei der Durchführung einer Simulationsstudie eine breite Spanne an Werkzeugen. Die Anzahl an kommerziell verfügbaren Simulationsumgebungen erhöht sich vor allem im Bereich der Materialflusssimulation zunehmend. Ansätze zur Realisierung einer dreidimensionalen Darstellung sind in einigen Tools bereits vorhanden (WitnessVR, Simple++). Nachteile dieser Entwicklung sind die hohen Kosten der Software und eine einseitige Richtung in der Konfiguration des Simulationsmodells. Zumeist ist es erforderlich, ein Simulationsmodell zu erzeugen und hieraus eine VR-Darstellung zu generieren. Aufgrund des hohen Funktionsumfangs und der allgemeinen Verwendbarkeit der Werkzeuge entsteht ein hoher Lernaufwand.

Zusätzlich ist bei der Modellierung oft eine Abstraktion der Systemkomponenten erforderlich, die Kenntnisse in Simulationstechniken bedingt.

Die Integration externer Routinen wird in den bekannten Simulationsumgebungen über eine C/C++-Schnittstelle realisiert. Eine Anbindung an Java-Objekte liegt noch nicht vor. Dieser Aspekt ist im Hinblick auf die Anforderungen, die an den zu entwickelnden Simulator gestellt werden, wesentlich. Hier muss die Integration der Steuerungsknoten, die durch Java-Objekte modelliert werden, erlaubt werden. Zumal die Darstellung grundsätzlich durch das Java3D-API erfolgen soll, empfiehlt sich Java zur Implementierung des Simulatorkerns. Vorteilhaft erweist sich ferner, dass Java als objektorientierte Programmiersprache die objektorientierte Softwareentwicklung unterstützt. Auch die Anbindung des Kerns an FUJABA untermauert die Entscheidung, Java einzusetzen. Die Entwicklung der Modellelemente kann sehr individuell erfolgen und ist nicht wie in Simulationsumgebungen durch Bausteine der Entwickler vorgegeben. Weiterhin ist die Entwicklungszeit von Java-Applikationen häufig kürzer im Vergleich zu einer Implementierung in C++.

Problematisch an diesem Ansatz erscheinen zunächst zwei Punkte:

- das Laufzeitverhalten von Java-Applikationen
- eine fehlende Unterstützung von Simulationsklassen

Java wird vornehmlich als plattformunabhängige Sprache verwendet, die während der Laufzeit den zuvor erzeugten Bytecode interpretiert. Die Programme werden in einer *Java Virtual Machine* (JVM) gestartet und ausgeführt. Die Ausführungszeit von Programmen, die interpretiert werden, ist aber im allgemeinen länger als von Anwendungen, die zuvor in einem Maschinencode compiliert wurden. Das Laufzeitverhalten von Java hat sich aber in den letzten Jahren durchgängig verbessert. Der Einsatz von *Just In Time*-Compilern (JIT) verringert die Laufzeit drastisch. JIT-Compiler compilieren den Bytecode, bevor er ausgeführt wird. Zur Zeit weist Java ein 3-10 mal langsameres Laufzeitverhalten im Vergleich zu C auf [vgl. Fla98, S. 8].

Die fehlende Unterstützung von Simulationsmodellen kann durch Klassenbibliotheken ausgeglichen werden, die zunehmend auch in Java entwickelt werden.

3.2.2 Anforderungen an den Simulatorkern

Die Aufgabe bei der Entwicklung des Simulatorkerns ist unter Beachtung der in [VDI96, S. 16] gegebenen Definition, Komponenten des realen Systems und deren Eigenschaften zu finden, die zur Nachbildung der realen Anlage erforderlich sind. Neben den Komponenten sind zur automatischen, chronologischen Erzeugung der Ereignisse, die dynamischen Prozesse zu beachten. Dabei müssen die

Beziehungen, die zwischen den einzelnen Komponenten bestehen, nachgebildet werden. D. h., nachdem ein statisches System mit entsprechenden Attributen vorliegt (z. B. aus der Benutzereingabe), muss eine Veränderung der Attribute einzelner Objekte in der Modellwelt möglich sein. Hierbei werden die aktuellen Werte der Objekte als *Zustand* bezeichnet[4]. Die Summe der Zustände aller Objekte im System wird als *Systemzustand* bezeichnet [VDI96, S. 20]. Der Übergang der Systemzustände ist so zu ermitteln, dass die Realität so weit angenähert wird, wie es erforderlich ist, um innerhalb eines Toleranzbereiches, der von der Aufgabenstellung der Simulation abhängig ist, zu bleiben. Wesentliche Aufgabe in diesem Zusammenhang ist festzulegen, ob Zustandsübergänge kontinuierlich oder sprunghaft diskret vorgenommen werden. Weiterhin muss festgelegt werden, wie ein Fortschalten der virtuellen Uhr realisiert wird. Ansätze aus der Literatur und der Praxis wurden in Kapitel 2.2.3 zur Untergliederung der Simulationsmethoden gegeben.

Bei der Modellbildung ist auf einen geeigneten Detaillierungsgrad zu achten [Lie95, S. 226]. Bei Modellen, die das reale System zu detailliert abbilden, kann die Entwicklungszeit so groß sein, dass der Aufwand nicht mehr mit den Nutzen aufzuwiegen ist. Weiterhin wird der Zeitaufwand für Simulationsläufe größer. In Hinblick auf die Wahl der abzubildenden Systembestandteile ist somit immer die Aufgabenstellung zu berücksichtigen [Lie95, S. 118]. Der Detaillierungsgrad des Simulators sollte nach der Anforderungsanalyse so gehalten werden, dass eine benutzerfreundliche VR-Darstellung zur Visualisierung verwendet werden kann. Somit ist es nicht das vorrangige Ziel, Schaltzeiten von speziellen Signalen zu erfassen. Im Vordergrund steht die Validierung der Ablaufsteuerung auf Grund der Applikationssoftware des Steuerungsknotens.

Der Simulatorkern muss zur grafischen Animation die errechneten Werte der Benutzungsoberfläche zur Verfügung stellen. Da der Simulatorkern ferner mit der Steuerung kommuniziert, sind Schnittstellen zu beiden Teilen zu ermitteln. Das Kernproblem liegt hier neben dem eigentlichen Datenfluss besonders in der *zeitlichen Synchronisation* der Kommunikation.

3.3 Analyse des Transportsystems

In diesem Abschnitt soll eine Analyse der Komponenten des Materialflusssystems erfolgen. Da der Simulatorkern auch die chronologisch geordnete Erzeugung der Ereignisse leisten muss, werden Prozessabläufe analysiert, die in der späteren Si-

[4] Im Folgenden werden die Bestandteile des realen Systems als „Komponenten“, die Nachbildungen innerhalb der Modellwelt als „Objekte“ bezeichnet. Im Sinne der UML werden Komponenten als ausführbare Softwaremodule mit eigener Identität und wohldefinierten Schnittstellen verwendet, vgl. [Oes98, S. 348]. Werden Komponenten in diesem Sinne verwendet, wird im Rahmen dieser Arbeit ausdrücklich darauf hingewiesen.

mulation zu berücksichtigen sind. Im Rahmen einer objektorientierten Analyse geht es in diesem Abschnitt um die Identifikation der Klassen und Prozesse.

3.3.1 Systemkomponenten

Die in diesem Abschnitt vorgenommene Analyse bezieht sich auf das in [GF98] beschriebene System der Firma Montech, das auch als Fallstudie im ISILEIT-Projekt dient. Der Materialtransport erfolgt durch selbstfahrende Palettenträger (Shuttle) auf einer vorgegebenen Streckentopologie. Die Streckentopologie des Transportsystems besteht aus mehreren *Streckenmodulen*[5] (LayoutElement)[6], die miteinander verbunden sind. Innerhalb der Streckenelemente werden ungeregelte Elemente wie Geraden und Bögen sowie geregelte Elemente wie Weichen, Kreuzungen und Lifte unterschieden [GKM00a]. Anhand der aufgezählten Elemente, die nach den Forderungen in Abschnitt 3.1 erweiterbar sein sollen, ist erkennbar, dass es Streckenelemente mit statischen und dynamischen Charakter gibt (Bild 3-2). Als statische Streckenelemente werden Elemente bezeichnet, welche eine feste Strecke vorgeben, die während des Einsatzes des Systems nicht verändert werden kann. Hierzu zählen die ungeregelten Elemente. Die variablen Streckenmodule erlauben unter bestimmten Bedingungen eine Veränderung des initialen Wegesystems. Sie führen zu Verzweigungen, die einen Palettenträger mit unterschiedlichen Streckenteilen verbinden können.

Variable Streckenmodule setzen sich aus statischen und variablen Streckenabschnitten[7] (AttachedTrack) zusammen. Die aktuell eingestellte Streckenführung ergibt sich aus der Verbindung der statischen Teile mit den dynamischen. Jeder variable Teilstreckenabschnitt eines Streckenmoduls wird einem Antrieb (InnerActor) zugeordnet. Bei der Modellierung ist zu beachten, dass ein Antrieb nicht nur einen Abschnitt kontrollieren kann, sondern dass durch das Auslösen eines Antriebs auch mehrere Abschnitte ihre Streckenführung ändern können. Dieser Aspekt wird in der Modellierung einer Querverschiebung bzw. eines Liftes deutlich. In der Anforderungsaufnahme zur Einstellung der Streckenführung wurde gefordert, dass die Einstellzeiten der Position durch konstante Zeiten festgelegt werden. Die Berechnung der mechanischen Bewegungen innerhalb eines Streckenmoduls im Simulatorkern entfällt daher. Da in der realen Anlage unterschiedliche Einstellzeiten auftreten können, wird dem lokalen Steuerungsobjekt ein Signal über das vollständige Einstellen der Streckenführung gegeben. In Bild 3-3 ist ein Klassendiagramm angegeben, das den Einfluss eines Antriebs auf die Streckenabschnitte verdeutlicht.

---

[5] Im Folgenden auch Streckenelement

[6] Die Bezeichnung einer Klasse, die Objekte einer Systemkomponente beschreibt, wird der jeweiligen Komponente in Klammern angehangen.

[7] Auch Streckenteilabschnitt

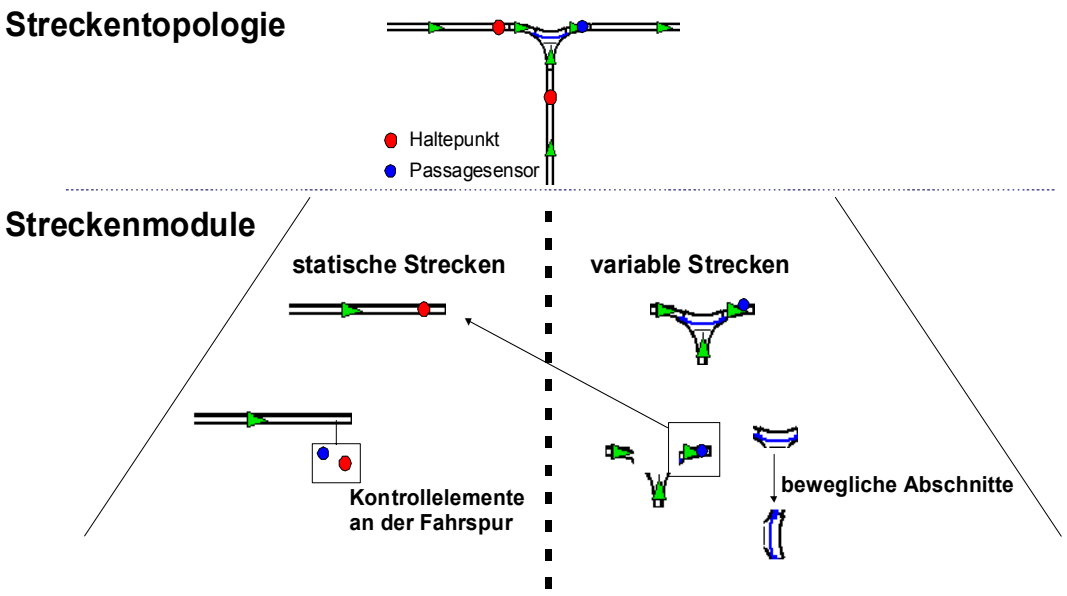


Bild 3-2: Analyse statischer und dynamischer Streckenmodule. Das Beispiel zeigt, dass eine Weiche aus vier Streckenabschnitten besteht. Drei Abschnitte verändern ihre Position nicht, nur der bewegliche Abschnitt führt zu einer anderen Streckenführung.

Im Zusammenhang mit den Streckenelementen wurde wiederholt auf exklusive Streckenteile hingewiesen. Exklusive Streckenteile können sich auf einzelne Streckenelemente sowie auf mehrere zusammenhängende Streckenelemente beziehen. Sie dürfen gleichzeitig nur von einem Shuttle befahren werden.

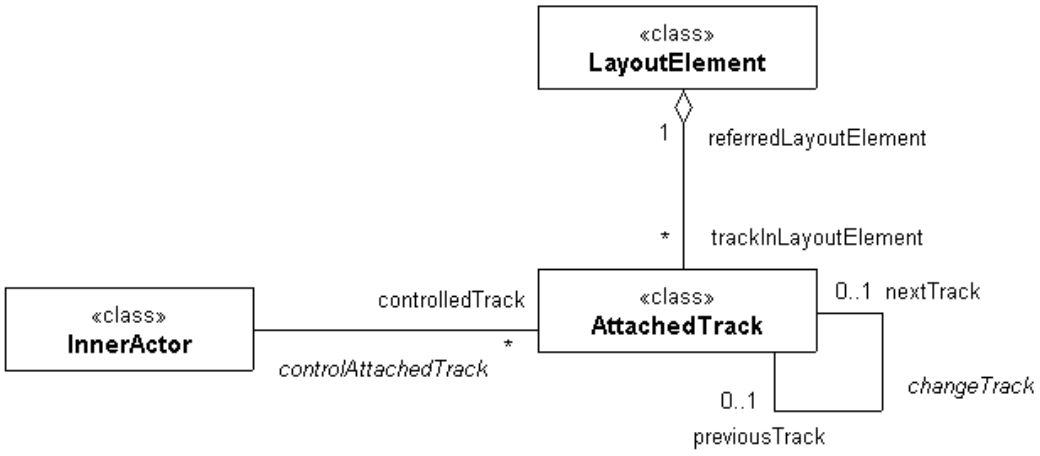


Bild 3-3: Klassendiagramm zur Analyse eines Streckenelements. Das vollständige Diagramm ist in Anhang A.1 (Bild 3) wiedergeben.

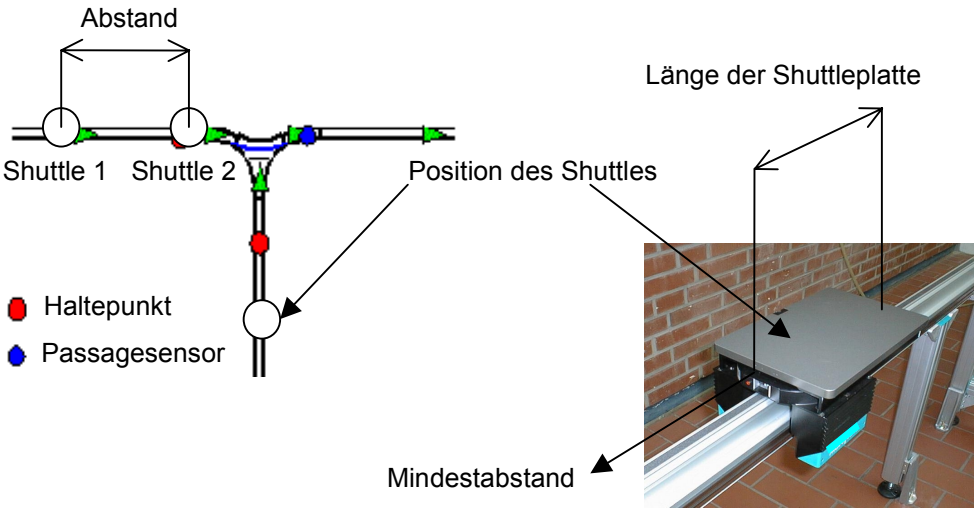
Zur Kontrolle des Systems wird eine Steuerung eingesetzt, die u. a. Weichen zu einer Änderung ihres aktuellen Zustandes veranlasst bzw. auf die Geschwindigkeit der Shuttles Einfluss nimmt. Die Steuerung erhält Signale von *Kontrollelementen* (ControlElement) und kann Signale an die Streckenelemente bzw. an ein Shuttle über andere Kontrollelemente weitergeben. Es werden Sensoren und Aktoren unterschieden. *Sensoren* (Sensor) haben die Aufgabe, durch Signale die Steuerung über einen veränderten Zustand des Systems zu informieren. Innerhalb der Modellierung sind zwei Arten von Sensoren zu differenzieren: *Identifikationseinheiten* (IdentificationUnit) und *Passagesensoren* (Sensor). Die Identifikationseinheiten liefern der Steuerung eine Zeichenkette, die ein passierendes Shuttle identifiziert. Passagesensoren übertragen an die Steuerung ein binäres Signal, sobald sie ein Shuttle registrieren. Die Passagesensoren reagieren auf Metallplatten, die sich am Shuttle befinden. Die Platten müssen nicht am vordersten Bereich des Shuttles angebracht sein. Bei der Modellierung der Shuttles ist dies durch die Angabe eines besonderen Punktes zu berücksichtigen. Mit Hilfe von Aktoren kann die Steuerung Veränderungen am System vornehmen. Aktoren bewirken den Start eines Shuttles. Eine weitere Funktion der Aktoren ist die Einstellung von dynamischen Streckenelementen. Neben Aktoren und Sensoren können sich noch weitere Kontrollelemente an einem Streckenelement befinden, um die Geschwindigkeit eines Shuttles zu verringern bzw. ein Shuttle abzustoppen. In den Gesprächen mit den Mitarbeitern der FASTEC wurden für diese Elemente die Bezeichnungen *AB-Nocke* (HSPoint[8]) bzw. *A-Nocke* (StopPoint) verwendet, die im Folgenden beibehalten werden sollen. Fährt ein Shuttle an einer AB-Nocke vorüber, wird es angewiesen, seine Geschwindigkeit auf die Hälfte zu reduzieren. Eine A-Nocke stoppt das Fahrzeug. Die Kontrollelemente können auch an einzelnen Einheiten zusammen auftreten. *Start-/Stoppeinheiten* verfügen neben einer A-Nocke zum Abbremsen des Shuttles über einen Aktor, der einen späteren Start des Shuttles auslöst und einen Sensor, der an die Steuerung das Signal gibt, dass ein Shuttle sich an diesem Element befindet.

Wesentlicher Bestandteil dezentral gesteuerter schienengebundener Transportsysteme sind die *Fahrzeuge* (ShuttleType), auf denen Produktteile zu den einzelnen Arbeitsstationen transportiert werden. Die Fahrzeuge bilden eigenständige Komponenten des zu modellierenden Systems. Sie fahren auf den fest vorgegebenen Strecken. Zur Modellierung eines Shuttles[9] im Simulatorkern ist es daher nicht erforderlich, die räumlichen Koordinaten einer dreidimensionalen Darstellung zu ermitteln. Die Position der Fahrzeuge während des Simulationslaufes wird durch ein Attribut bestimmt, das die Entfernung angibt, die ein Shuttle bereits auf einem Streckenabschnitt zurückgelegt hat. Jedes Fahrzeug ist in dieser Fallstudie mit Vorrichtungen ausgestattet, die den Abstand zu einem anderen Fahrzeug

[8] Die Buchstaben *HS* stehen in diesem Zusammenhang für *Half Speed* und verdeutlichen somit die Funktion des Elementes.

[9] Im Folgenden werden die Begriffe *Fahrzeuge* und *Shuttle* synonym verwendet.

ermitteln. Zur Berechnung des Abstandes zwischen zwei Shuttles müssen die Lage des ausgezeichneten Punktes (siehe auch Passagesensoren) vom Beginn der jeweiligen *Shuttleplatte* und die Länge des Shuttles geben sein. Des Weiteren sind zur Beschreibung eines Shuttles Werte für die aktuelle und die maximale Geschwindigkeit vorzusehen. In Bild 3-4 werden die Werte verdeutlicht. Das Klassendiagramm zur Klasse *ShuttleType* ist dem Anhang A.1 (Bild 1) zu entnehmen.



Pos(x) = Position des Shuttles x
Point(x) = Abstand des ausgezeichneten Punktes zum vordersten Punkt von Shuttle x
MinDis = Minimaler Abstand zum vorausfahrenden Shuttle
Length(x) = Länge der Shuttleplatte von Shuttle x

Pos (2) – Pos (1) = Abstand >= Point(1) + MinDis + Lentgh(2) – Point(2)

Bild 3-4: Modellierung der Eigenschaften eines Shuttles im einem Simulationslauf. Die Einhaltung des Abstands ist in der Simulation als Invariante anzusehen.

Ein Fahrzeug verfährt auf dem gegebenen Schienensystem nach einem Plan, der die Reihenfolge vorgibt, in welcher das Shuttle die einzelnen Arbeitsstationen anfahren muss. Die Pläne werden in einem Steuerungsknoten abgelegt, der eine Zuordnung zwischen einem zu bearbeitenden Plan und einem Shuttle herstellt. An den *Arbeitsstationen* (Station) werden von den Fahrzeugen transportierte Teile bearbeitet. Die Arbeitsstationen sind nicht unmittelbar als Bestandteil des Transportsystems anzusehen, da sie aus Sicht des Materialflusses einen Stopp- mit anschließenden Startprozess bilden. Ein Shuttle erreicht eine Station und stoppt an einem Haltepunkt. Der lokale Steuerungsknoten übernimmt die Kontrolle über den weiteren Ablauf, indem er der Arbeitsstation ein Startsignal zur Bearbeitung

gibt. Nach Beenden des Arbeitsvorgangs erhält dieser Knoten von der Station eine Bestätigung. Daraufhin kann das Transportshuttle wieder gestartet werden. Da das Transportsystem eine feste Systemgrenze hat, wirkt die Arbeitsstation von außen auf das System ein. Eine detaillierte Modellierung der Arbeitsstationen soll daher innerhalb des Simulatorkerns nicht erfolgen. Damit die Trennung zwischen der Ebene der Steuerungsobjekte und der Benutzungsschnittstelle erhalten bleibt, muss im Simulatorkern eine Klasse zur Repräsentation der Arbeitsstationen vorgesehen werden.

In den vorherigen Abschnitten wurden die Systemkomponenten, die bei der Modellierung berücksichtigt werden müssen, beschrieben. Das Bild 3-5 zeigt ein Klassendiagramm mit den vorgestellten Systemkomponenten. Da die Analyse der Systemkomponenten unabhängig von der im Simulatorkern realisierten Simulationsmethode ist, werden die Klassen in den folgenden Abschnitten weiter spezifiziert. Zumeist werden nur Teilaspekte des Systems berücksichtigt, somit dient das in Bild 3-5 dargestellte Diagramm zur Orientierung. Die vollständigen Beziehungen zwischen den Klassen sind dem Anhang A.1 (Bild 6) zu entnehmen.

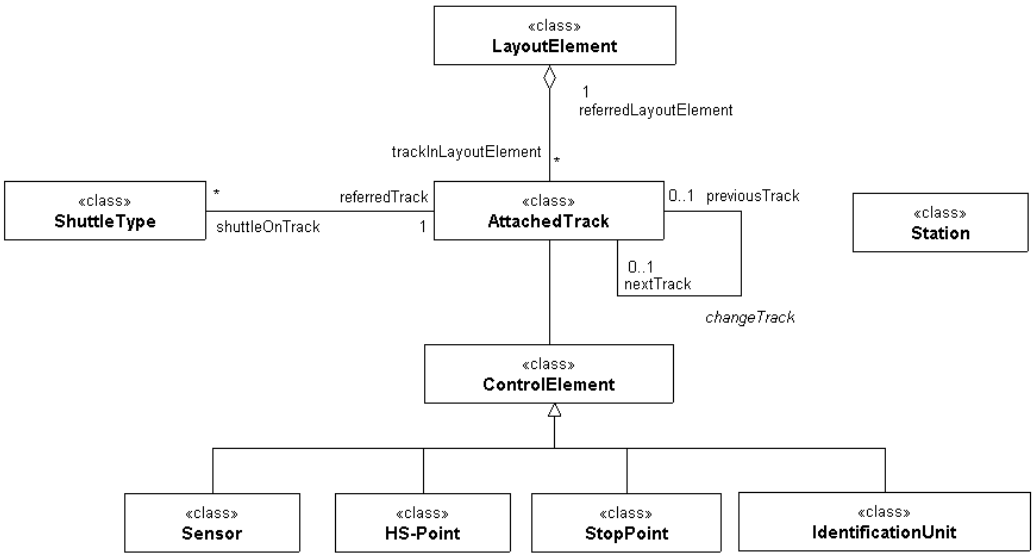


Bild 3-5: Klassendiagramm der statischen Modellelemente. Die Klasse „Inner-Actor“ wurde zur besseren Übersicht nicht abgebildet.

Das in Bild 3-5 dargestellte Klassendiagramm zeigt neben den Klassen zur Repräsentation der Systemkomponenten auch deren Beziehungen untereinander. Es ist erlaubt, dass mehrere Shuttles gleichzeitig auf einem Streckenabschnitt fahren. Die Einhaltung des Abstands ist von den Objekten der Klasse *ShuttleType* zu gewährleisten. Die Behandlung exklusiver Streckenabschnitte wird von der Steuerungsebene kontrolliert. Der enge Bezug der Kontrollelemente zu den Streckenteilen wird durch die Assoziation zwischen den beiden Klassen modelliert. Weil

jedes Kontrollelement an der Strecke befestigt ist, jedoch unterschiedliche Aufgaben erfüllt, wird die Oberklasse *ControlElement* in Unterklassen spezialisiert.

3.3.2 Dynamische Prozesse

Nachdem die Komponenten genannt wurden, wird nachfolgend ihr Verhalten genauer analysiert. Hierbei sollen die Zustandsänderungen der Elemente im Zusammenhang mit den dynamischen Prozessen im System erläutert werden.

Fahrzeuge

Bei der Analyse der Shuttles ist aufgefallen, dass sie bis auf eine Sensorik, die den Abstand zum vorausfahrenden Shuttle kontrolliert, keine eigene Funktion aufweisen. Die Zustandsübergänge (Bild 3-6) der Shuttles werden vorwiegend von der Steuerung, der Streckentopologie und den Kontrollelementen bestimmt.

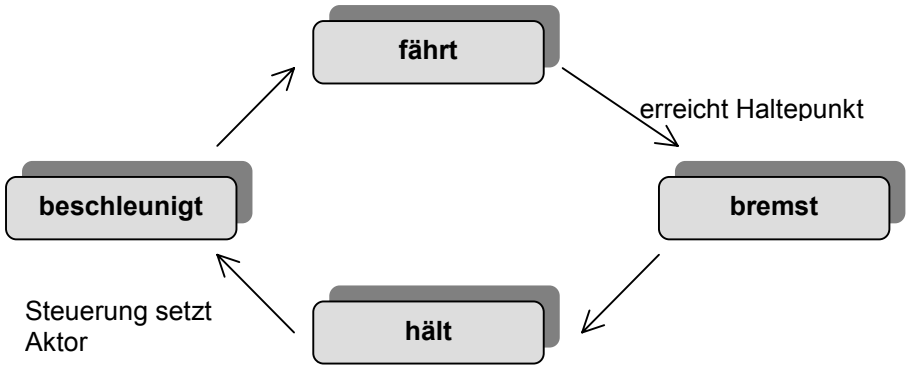


Bild 3-6: Zustandsübergänge eines Shuttles an einem Haltepunkt

Erreicht ein Shuttle einen Haltepunkt, wechselt es in den Zustand „bremst“ und darauf in den Zustand „hält“. Erst nachdem über die Steuerung einem Aktor das Signal zum Starten des Shuttles gegeben wurde, beschleunigt das Shuttle und setzt seinen Weg fort, der durch die Streckentopologie vorgegeben ist. Das angegebene Zustandsdiagramm ist nur beispielhaft. Ähnliche Übergänge treten auf, wenn das Shuttle eine AB-Nocke passiert oder wenn der Abstand zum vorausfahrenden Shuttle zu gering wird und die Geschwindigkeit daraufhin angepasst werden muss.

Streckenmodule mit variabler Streckenführung

Neben den Shuttles verändern auch die dynamischen Streckenelemente ihren Zustand, während das Transportsystem in Betrieb ist. Der Ablauf an einer Weiche wird in Bild 3-7 dargestellt. Beim angegebenen Beispiel wird das Verhalten an einer zusammenführenden Weiche (Joiner) beschrieben. Die Steuerung kontrol-

liert, dass sich nur ein Fahrzeug auf der Weiche befindet und die Weichenführung richtig eingestellt ist.

Damit die Steuerung die Kontrolle über den Ablauf an der Weiche ausüben kann, benötigt sie Informationen darüber, wo sich die Shuttles befinden und welcher Streckenverlauf an der Weiche aktuell eingestellt ist. Die Position der Shuttles wird über die Sensoren an die Steuerung geliefert. Um die Streckenführung an der Weiche verändern zu können, muss der variable Teil des Streckenmoduls veranlasst werden, seine Stellung zu ändern. Dafür verfügen die Systemkomponenten über Antriebe, die auf Anforderung der Steuerung eine gewünschte Position einstellen.

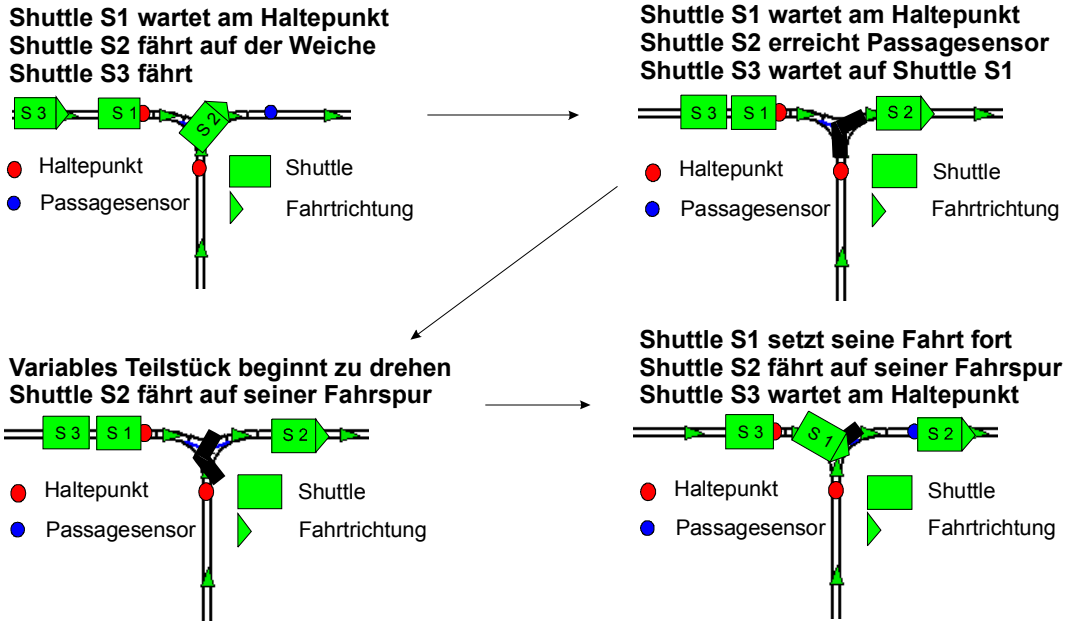


Bild 3-7: Ablauf an einer Weiche: Das Shuttle S2 passiert die Weiche. Nach der Abmeldung durch den Passagesensor wird die Streckenführung eingestellt und die wartenden Shuttles gestartet.

Wie aus dem Beispielablauf (Bild 3-7) hervorgeht, ist es für die Modellierung der Prozesse eines variablen Streckenabschnittes noch nicht ausreichend, die Streckenführung zu verändern. Vielmehr ist darauf zu achten, dass ein wartendes Shuttle nach erfolgreichem Einstellen des Streckenmoduls wieder gestartet wird. Hierzu muss der Steuerungsknoten den Aktor an einem Start-/Stoppelement ansprechen (Bild 3-8). Dieser löst durch einen Mechanismus das Starten des Shuttles aus.

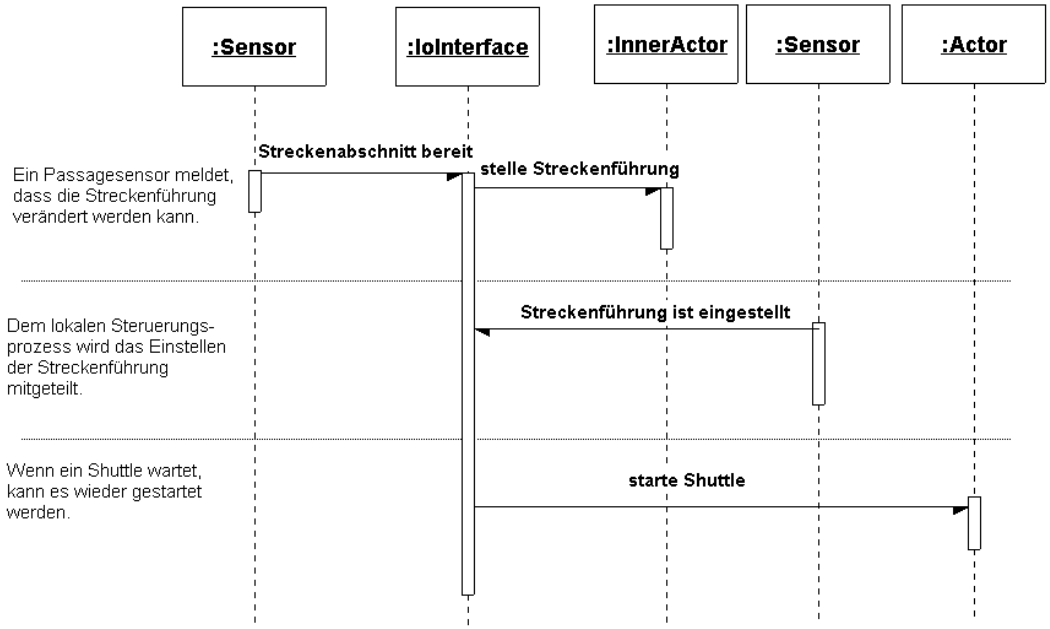


Bild 3-8: Schematischer Ablauf beim Einstellen variabler Streckenmodule. Der Ablauf setzt voraus, dass der Abschnitt von einem Shuttle befahren wird, ein Shuttle vor dem Modul wartet und eine neue Streckenführung eingestellt werden muss.

Arbeitsstationen

An den Arbeitsstationen werden die transportierten Teile bearbeitet, neue Teile in das System eingebracht oder fertige Produkte entnommen. Zur Bearbeitung halten die Shuttles an und werden nach der Bearbeitung wieder gestartet. Da der Simulator eine Simulation des Transportsystems unterstützen soll, ist es nicht unmittelbare Aufgabe, die Bearbeitung an den Stationen zu simulieren bzw. zu animieren. Allerdings hat die Bearbeitungszeit an den Arbeitsstationen Einfluss auf den Ablauf im Transportsystem. Je länger ein Produkt an einer Arbeitsstation bearbeitet wird, desto länger ist diese für andere Produkte gesperrt. So können Stauungen im Transportsystem entstehen, die eventuell auch den Transport zu anderen Stationen blockieren können. Die Produktivität[10] der Anlage wird stark beeinträchtigt.

Der Ablauf an einer Arbeitsstation aus Sicht des Transportsystems ist in Bild 3-9 erkennbar. Aus dieser Abbildung geht hervor, dass es bei der Modellierung einer Station für den Simulatorkern nicht entscheidend ist, welche Art Station sich an der Strecke befindet. Stattdessen ist die Zeit ausschlaggebend, die bis zur Weiterfahrt des Shuttles vergeht. Die Bearbeitungszeit an einer Station ist abhängig von dem Produkt, das gefertigt werden soll. Ein Shuttle hat aber keine Information über das Produkt, das es transportiert. Diese Information liegt in einem Knoten

[10] Die Produktivität ist das Verhältnis zwischen dem Faktoreinsatz und dem Faktorertrag.

der Steuerung, in dem die Arbeitsaufträge abgelegt sind, nach denen die Shuttles arbeiten. An dieser Stelle zeigt sich, dass die Modellierung eng mit der Steuerungsebene zusammenhängt, deren Schnittstelle zum Kern ein Gegenstandsbereich des nächsten Kapitels sein wird.

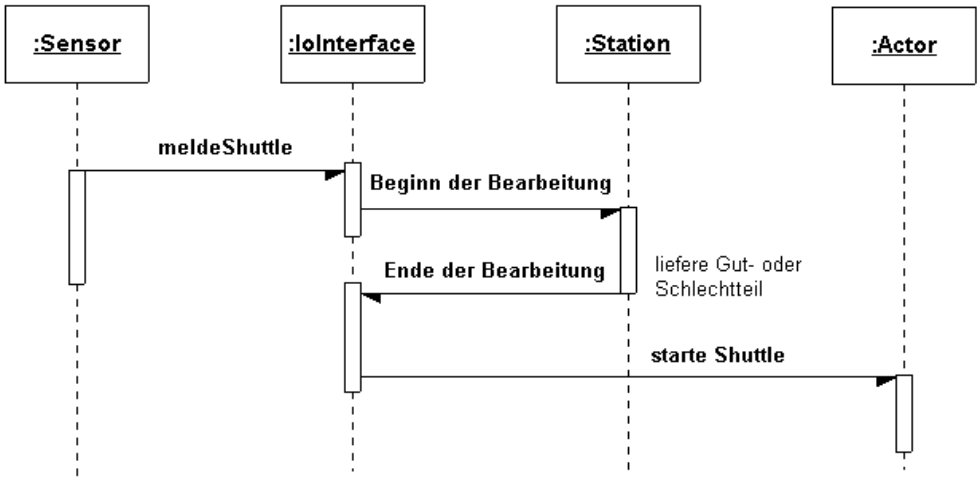


Bild 3-9: Schematischer Ablauf an einer Arbeitsstation. Ein Sensor meldet die Ankunft eines Shuttles. Nach der Bearbeitung wird das Shuttle wieder gestartet.

3.4 Analyse der Schnittstellen

Die Architektur des zu entwickelnden Simulators besteht aus drei Schichten. Neben der Benutzungsoberfläche werden die Ebenen des Simulatorkerns und der Steuerungsobjekte unterschieden. Diese Aufteilung soll unterstreichen, dass der Simulator vor allem zur Validierung der Spezifikation der Steuerung eingesetzt werden soll. Ziel ist es, das Verhalten der Steuerung von den Objekten des Simulatorkerns zu trennen. Die Steuerungsobjekte können so ein unterschiedliches Verhalten der Materialflussanlage simulieren, ohne dass der Simulatorkern verändert werden muss. Damit eine eindeutige Trennung der Schichten eingehalten werden kann, sind Schnittstellen zu definieren, an die sich die Objekte der einzelnen Ebenen halten müssen. In diesem Abschnitt sollen die Schnittstelle zwischen der Benutzungsoberfläche und den Simulatorkern sowie die Schnittstelle des Kerns und der Steuerungsebene inhaltlich bearbeitet werden.

3.4.1 Schnittstelle zur Benutzungsoberfläche

In der Aufgabenstellung zur Diplomarbeit wird darauf hingewiesen, dass der Simulationskern auf einer vorhandenen Java3D-Visualisierung aufbauen kann und dass die Konfigurationsdaten der Anlage in einer objektorientierten Datenbank gespeichert sind. Diese Aufgaben werden im Rahmen einer anderen Diplomarbeit bearbeitet. Da sich die Realisierung der Aufgaben zeitlich verzögert, sind die Anforderungen, die an die analysierte Schnittstelle gestellt werden, im Verlauf der Arbeit geändert worden. Die Änderungen beziehen sich zumeist auf die Anbindung an die grafische Darstellung der Ergebnisse: Zum einen soll durch die Visualisierung der korrekte Ablauf des Kerns überwacht werden, zum anderen erfordert die Fortführung des gesamten Projektes eine prototypische Darstellung. Da in der weiteren Entwicklung des Simulators eine Darstellung auf Basis des Java3D-API vorgesehen ist, werden die Anforderungen an beide Alternativen der Darstellung analysiert.

Der ursprünglich geplante Ansatz der Anbindung an Java3D stellt durch die Realisierung der Geometrie als Java-Objekte einen Weg dar, die Modellelemente des Kerns unmittelbar mit den grafischen Objekten zu verbinden. Hierzu delegiert ein Modellelement die Nachrichten seiner Veränderung an die grafische Oberfläche, die unmittelbar auf die Daten reagieren kann (Bild 3-10).

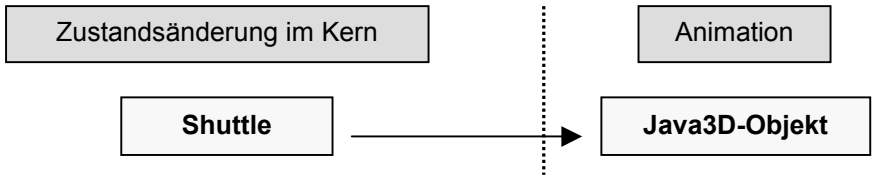


Bild 3-10: Schematische Darstellung einer Delegation der Anweisungen zur Animation der Zustandsveränderungen.

Die Aufgabe der Schnittstellenspezifikation ist es zum einen festzulegen, welche Daten weitergeleitet werden und zum anderen die Synchronisation der beiden Bereiche vorzunehmen. Die Synchronisation der beiden Teilbereiche soll über eine einheitliche Zeit durchgeführt werden. Hierzu ist es notwendig, dass die Objekte der Animation eine Darstellung in einer vorgegebenen Zeitspanne garantieren. Der Simulatorkern muss die erforderlichen Daten bereitstellen. Unterschreitet die gelieferte Datenmenge die geforderte, ist eine Warnung auszugeben. Ändert sich der Zustand eines Modellelementes im Kern, ist der Start-, der Endzeitpunkt und der Umfang der Bewegung anzugeben. Der Bewegungsumfang bezieht sich bei den Shuttles auf die zurückgelegte Strecke, bei variablen Streckenelementen auf die einzustellende Position und bei einer Arbeitsstation auf den Grad der Vollständigkeit der Bearbeitung innerhalb der angegebenen Zeitspanne.

Die anderen Ansätze zur Animation der Zustandsänderungen entziehen sich einer möglichen interaktiven Einflussnahme durch den Benutzer, indem eine Visualisierung nach dem Simulationslauf erfolgt. Eine Variante baut auf der Animation der Simulationsereignisse mit einem kommerziellen Werkzeug, dem CASUS-Presenter[11], auf. Die andere basiert auf einer Darstellung in einem VRML-Viewer. Da der CASUS-Presenter bei der FASTEC noch nicht einsatzfähig gewesen ist, wird auf eine VRML-Darstellung zurückgegriffen.

Die Anforderung an die Schnittstelle besteht hier im Erzeugen einer Datei, deren Inhalt dann in der Darstellung verwendet werden kann. In der Tabelle 3-1 werden die Ereignisse aufgeführt.

Tabelle3-1: Satzformat der Datei zum Austausch der Simulationsereignisse

Zeilenaufbau	**Auswirkung im VRML-Viewer**
Zeit *changeTrack* Shuttle LayoutElement AttachedTrack	Wechsel eines Shuttles auf das angegebene Streckenmodul
Zeit *stopShuttle* Shuttle	Hält das Shuttle an
Zeit *startShuttle* Shuttle	Startet das Shuttle
Zeit *moveElement* LayoutElement AttachedTrack Position	Stellt das angegebene Streckenmodul auf die entsprechende Position ein

3.4.2 Schnittstelle zur Steuerungsebene

In diesem Abschnitt bearbeite ich thematisch die Festsetzung der Lage der Schnittstelle zwischen dem Simulatorkern und der Steuerung. Darauf aufbauend wird dargelegt, welche Information auf beiden Seiten der Schnittstelle vorhanden sein muss, um eine Kommunikation zu ermöglichen und gleichzeitig die Teilbereiche autonom zu halten.

Die Steuerung der realen Anlage besteht aus Steuerungsknoten, die über ein Kommunikationsnetz miteinander verbunden sind. Jeder Steuerungsknoten erhält von seinem Prozessinterface lokale Information über die von ihm verwalteten Bereiche (Abschnitt 2.1.1). Hiervon ausgehend wird die Lage der Schnittstelle zwischen den Simulatorkern und den Steuerungsobjekten an diesem Interface angesiedelt (Bild 3-11). Die Modellelemente der Hardware der Förderungstechnik

[11] Der CASUS-Presenter erlaubt die Visualisierung einer ereignisgesteuerten Simulation. Er wird am Frauenhofer Institut für grafische Datenverarbeitung in Darmstadt entwickelt.

liefern notwendige Informationen an das zugeordnete Interface. Die Objekte der Steuerung erhalten diese Information, verarbeiten sie und leiten Nachrichten an das Prozessinterface weiter, damit die Anlage die erforderlichen Einstellungen vornehmen kann.

Eine weitere Frage bezieht sich auf die Zuordnung der Hardware zu den Steuerungsknoten. D. h. ist ein Hardwareobjekt einem Knoten zugeordnet, oder ist es erlaubt, dass mehrere Knoten Informationen von einem Element erhalten. Nach Rücksprache mit einem Mitarbeiter der FASTEC wurde festgelegt, dass die entsprechenden Komponenten des Systems wie Sensoren und Aktoren jeweils genau einem Knoten zugeordnet sind. Ein Knoten kann allerdings von mehreren Elementen Informationen erhalten. Benötigt ein Knoten Information von einem Kontrollelement, das ihm nicht zugeordnet ist, muss die Applikationssoftware eine Kommunikation des entsprechenden Knotens mit dem Steuerungsknoten vorsehen, der die gewünschte Information erhält. Dieser Datenaustausch erfolgt über den Kommunikationsbus und wird im Rahmen der Entwicklung des Simulatorkerns nicht weiter berücksichtigt.

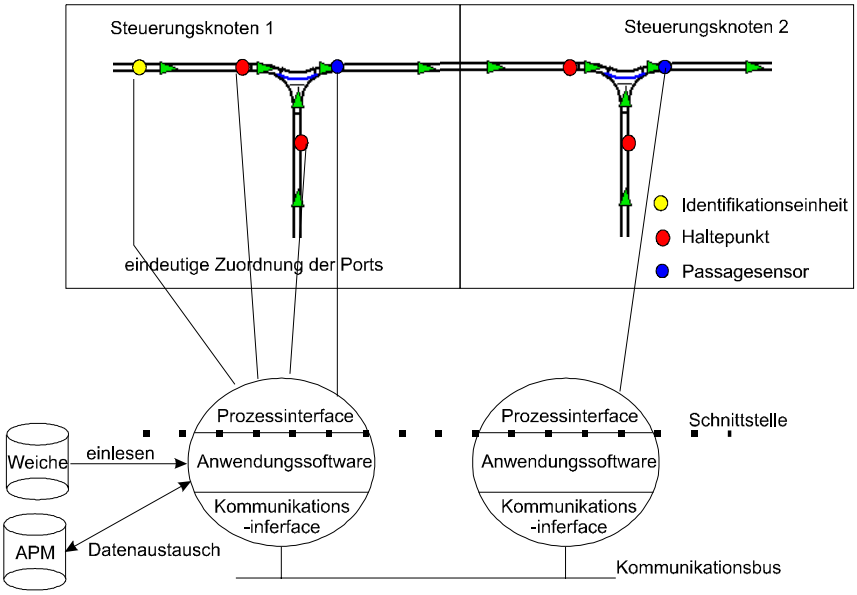


Bild 3-11: Schematische Darstellung der Kommunikation zwischen Hard- und Software. Der zu bearbeitende Auftragsbestand wird über das Auftragsplanungsmodul (APM) in die Simulation eingebracht. Die Applikationssoftware der Weiche steuert den Verlauf in diesem Streckenbereich.

Obschon die Applikationssoftware nicht Gegenstand des Simulatorkerns ist, reagiert die Software auf Informationen, die von den Elementen des Simulatorkerns zu berechnen sind. Das Prozessinterface stellt *Ports* zur Verfügung, an denen die

Sensoren und Aktoren angeschlossen werden können. Die Applikationsschicht des Steuerungsknotens muss eine eindeutige Zuordnung der *Ports* zu den Kontrollelementen haben, um die Information über den Zustand des Systems interpretieren zu können. Da die Software vom Benutzer des Simulators erstellt wird, muss die Zuordnung der Kontrollelemte zu den *Ports* des Steuerungsknotens auch durch den Benutzer vorgenommen werden. Die erforderlichen Eingaben sind über die Benutzungsoberfläche vorzunehmen. Neben der Portzuordnung muss der Benutzer auch die Applikationssoftware den Steuerungsknoten zuordnen.

Die Ereignisse im Simulatorkern werden chronologisch geordnet erzeugt. Zumal sich der Systemzustand der Modellelemente durch den Einfluss der Steuerungsobjekte ändert, ist auf eine zeitliche Synchronisation der beiden Architekturebenen zu achten.