KONZEPTION UND OBJEKTORIENTIERTE MODELLIERUNG EINES SIMULATIONSVERFAHRENS AUF DER BASIS ZELLULARER AUTOMATEN FÜR DEN VERKEHRSABLAUF AUF BINNENWASSERSTRASSEN

CONCEPTION AND OBJECT ORIENTED MODELLING OF A SIMULATION METHOD BASED ON CELLULAR AUTOMATA TECHNIQUES FOR THE TRAFFIC FLOW ON INLAND WATERWAYS

von

Michael BERNARD

A B S T R A C T

This concept describes a system for the microscopic simulation of ship movements on differently constituted waterways, locks, crossings and ports. The ships are simulated within the program sytem with their specific characteristics and they are directly identifiable objects.

Simulation methods on waterways usually use a heuristic basis. Averaged speed, probability and distribution models are used to describe the simulation area. Since from these models no precise ship positions and states could be determined at a certain time, for this microscopic simulation of ship movements a new concept has been developed. The cellular automata concept is based on the theory, which is already used for simulation of vehicles (cars) in cities and on motorways. With this technique the units are moved on the basis of determined rules on a cell grid in defined time intervals. To gain flexibility for extensions and special adjustments, the developed system uses the methodology of the object-oriented analysis, resulting in a hierarchical modular structure. Based hereon, rules and characteristics can be assigned to categories (groups of objects) using inheritance mechanisms. By this a maximal flexibility is achieved.

Z U S A M M E N F A S S U N G

Das vorgestellte Konzept beschreibt ein System zur mikroskopischen Simulation von Schiffsbewegungen auf unterschiedlich gearteten Wasserstraßen, Schleusen, Kreuzungen und Häfen. Die Schiffe werden mit deren Spezifika im Programmsystem simuliert und sind direkt identifizierbare Objekte.

Simulationsverfahren auf Wasserstraßen benutzen in der Regel heuristische Ansätze. Es kommen neben Wahrscheinlichkeits- und Verteilungsmodellen beispielsweise gemittelte Fahrtgeschwindigkeiten zum Einsatz. Da aus diesen Modellen keine konkreten Schiffspositionen und -zustände zu bestimmten Zeitpunkten ermittelt werden können, wurde für die mikroskopische Simulation von Schiffsbewegungen ein neues Konzept entwickelt. Das Zellularautomaten-Konzept basiert auf der Theorie, die auch für die Simulation von Fahrzeugen (Autos) in Städten und auf Autobahnen angewandt wird. Hierbei werden die Einheiten anhand festgelegter Regeln auf einem Zellraster in definierten Zeitintervallen bewegt. Um das entwickelte System für Erweiterungen und spezielle Anpassungen flexibel zu halten, wurde die Methodik der objektorientierten Analyse verwendet, so dass sich eine hierarchische modulare Struktur ergibt. Hierauf aufbauend können Regeln und Eigenschaften von Elementen des Simulationsprogramms mit Hilfe der Vererbungsmethodik Kategorien (Objektgruppen) zugewiesen werden, womit eine größtmögliche Flexibilität erreicht wird.

I N H A L T

A B B I L D U N G S V E R Z E I C H N I S

S Y M B O L V E R Z E I C H N I S

l	=	Länge [m]
p	=	Wahrscheinlichkeit [-]
q	=	Verkehrsfluss [Einheiten/s]
t	=	Zeit [s]
v	=	Geschwindigkeit [m/s]
x	=	Ortskoordinate [m]
A	=	Beschleunigung [m/s²]
T	=	Zeitintervall [s]
ρ	=	Verkehrsdichte [Einheiten/m²]

1 Einleitung

In dieser Arbeit wurde ein Simulationsverfahren für den Verkehrsablauf auf Binnenwasserstraßen konzipiert und objektorientiert modelliert. Die Basis für das Simulationsmodell bildet die Theorie der zellularen Automaten.

Ziel der Arbeit ist es eine problemorientierte Herangehensweise aufzuzeigen, aus der eine pragmatische Methodik resultiert, die zu einem klar strukturierten und einfach erweiterbaren Programmsystem führt. Das entwickelte Programm stellt mächtige Funktionen zur Verfügung, mit denen auf einfachste Weise Modellierungsgebiete abgebildet und simuliert werden können.

Das System ist intern in Schichten gegliedert, die aufeinander aufbauen. Erweiterungen des Systems können auf diese Weise punktuell vorgenommen werden und integrieren sich sofort in das Gesamtkonzept.

Der Datenaustausch zwischen Modell und Benutzer bzw. einer Datenschnittstelle wird durch ein klar definiertes Datenkonzept geregelt, das auch Raum-Zeit-Relationen durch ein leicht erfassbares Konzept abbildet und verarbeitet.

Das Datenkonzept ist für die geplante Anbindung an Echtzeit-Schiffspositionen nötig, die von Schiffen, die mit einem GPS-System ausgestattet sind, an eine Zentrale geschickt werden. Diese Daten können auf einfache Weise mit dem hier vorgestellten Konzept verarbeitet werden. Das Programm kann mit beliebigen Systemzuständen starten und auf der Grundlage dieser Daten gute Prognosen erstellen.

Es werden Methoden und Konzepte vorgestellt, die in dieser Form bisher kaum untersucht wurden und in den meisten Punkten in keiner Umsetzung zu finden sind:

- Mikroskopische Simulation von Schiffsbewegungen auf Basis zellularer Automaten,
- Kombination von Zellularautomaten und Zustandsautomaten,
- spezielles Puffersystem zur Entkopplung einzelner Elemente und gleichzeitig als Verbindungsmöglichkeit unterschiedlicher Technologien,
- objektorientierte Modellierung zur leichten Erweiterbarkeit bei gleichzeitig sehr hoher Bearbeitungsgeschwindigkeit,
- Umsetzung aller Basiselemente (Kanal, Kreuzung, Schleuse, Hafen) in einem System, die sich beliebig kombinieren lassen und über Parameter anpassbar sind,
- Positions- und Zeitdatenkonzept, das Realdaten und Simulationsdaten in gleichen Strukturen speichert und direkt vergleichen kann,
- Simulationsmöglichkeiten von allgemeinen Fällen (beispielsweise mehrspuriger Verkehr mit Überholen und Gegenverkehr) mit realitätsnahem Verhalten durch Pseudo-Intelligenz,
- Routing-Algorithmen, die eine freie Wahl von Fahrtrouten erlauben.

2 Grundlagen

2.1 Prinzipien der Objektorientierung

Es können in den Grundlagen nicht alle Aspekte der Objektorientierung behandelt werden. Die hier aufgeführten Prinzipien beschreiben die Methodik nur in ihren Kernpunkten.

Ein großes Problem bei der Softwareentwicklung ist die Komplexität der Aufgabenstellung. Es wird versucht, das Problem durch Reduzierung auf das Wesentliche (Abstraktion), strukturierte Einordnung (Hierarchisierung) und durch Zerlegung in überschaubare Teilaufgaben (Modularisierung) in den Griff zu bekommen. Frühere Ansätze in der Informatik konnten dieses Vorgehen aber nur teilweise durchgängig unterstützen. Größtes Problem war bisher der Übergang vom Entwurf eines Softwaresystems hin zu seiner Implementierung. Die Programmiersprachen und deren Konzepte, allen voran der prozedurale Ansatz, verlangen oft ein Umdenken, da Funktionalität und Datenstrukturen einer starken Trennung unterworfen werden. Zudem sind bereits beim Entwurf die Möglichkeiten der Implementierungsumgebung mit einzubeziehen, um überhaupt zu einem Ergebnis kommen zu können. Dieses führt aber zwangsläufig zu einem Entwurf, der die Implementierung bereits vorwegnimmt und nur schwer nachvollziehbar ist.

Wesentlicher Aspekt der Objektorientierung ist es deshalb, eine dem menschlichen Denken nachempfundene und vor allem für alle Phasen der Softwareentwicklung durchgängige Methode zu bieten. Die Objektorientierung schafft die Möglichkeit, der menschlichen Denkweise entsprechende Konzepte nahezu 1:1 abzubilden. Neben Abstraktion, Hierarchisierung und Modularisierung wird die Trennung von Funktionalität und Daten durch Kapselung aufgehoben (Eschen, 1998).

2.1.1 Objekte

Die grundlegenden Konstruktionen bei objektorientiertem Softwareengineering ist das Objekt, welches Eigenschaften und Verhalten in einer Einheit

zusammenfasst. Die Eigenschaften werden von den Attributen beschieben. Die Attributwerte bestimmen den Zustand eines Objekts. Ein Zustand kann dynamisch durch Operationen verändert werden. Eine Menge von definierten Methoden beschreibt das mögliche Verhalten eines Objekts (Damrath, 1998).

2.1.2 Abstraktion

Abstraktion ist ein grundlegendes Prinzip des Menschen, um mit Komplexität umzugehen. Hierunter wird verstanden nur die wesentlichen Aspekte zu behandeln, Gemeinsamkeiten zwischen verschiedenen Objekten zu erkennen.

Viele Sachverhalte, die im Computer abgebildet werden sollen, sind sehr komplex. Um diese zu erfassen, wird einen Teil der zur Verfügung stehenden Information ausgewählt. Ist dieser Teil gut gewählt, stehen genügend Informationen zur Verfügung, um ein pseudo-reales Bild zu erhalten. Abstraktion hilft, ein Problem zu beschreiben, ohne alle zur Verfügung stehenden Informationen (und deren Komplexität) auswerten zu müssen.

Bei Anwendung von Abstraktion wird ein Modell der Realität entwickelt. Dabei werden Objekte identifiziert und deren Verhalten analysiert. Wichtig ist es, vorhandene Abhängigkeiten zwischen den Objekten zu erkennen, um später eine geeignete Klassenhierarchie entwickeln zu können. Größtes Problem ist die Abwägung des Grads der Abstraktion für das jeweilige Anwendungsgebiet (Eschen, 1998). Diese Problematik wird in der Systemanalyse behandelt (Natke, 1998).

2.1.3 Hierarchisierung

Das Konzept der Abstraktion ist ein effizientes Hilfsmittel, jedoch werden sich in komplexeren Anwendungsgebieten eine Vielzahl von Abstraktionen finden lassen. Diese bilden oftmals eine Hierarchie, deren Analyse zu einem viel genaueren Problemverständnis führt.

Innerhalb der Hierarchie finden sich verschiedene Abstraktionsebenen. Dabei generalisieren höhere Ebenen, wo hingegen niedrigere Ebenen spezialisieren (Eschen, 1998).

2.1.4 Modularisierung

Modularisierung ist ein Prinzip, das bereits bei der strukturierten Programmierung eingesetzt wird. Es ermöglicht eine getrennte Betrachtung von Problemen eines Anwendungsbereichs. Logisch zusammengehörige Teile werden als Einheit (Modul) verwaltet. Dabei stehen Module untereinander durch eine möglichst lose Kopplung in Verbindung.

Module sind programmtechnisch physische Behälter, in denen man Klassen des logischen Entwurfs deklariert. Sie werden mit einer Schnittstelle versehen, über die sie mit ihrer "Außenwelt" kommunizieren können. Ihr Inneres bleibt für andere verborgen. Bei der Entwicklung erleichtert die Modularisierung das Coding. Bereits fertiggestellte Module werden einmal übersetzt (compiliert) und stehen dann in übersetzter Form zur Verfügung. Bei weiterer Verwendung der Module entfällt der Übersetzungsvorgang, was bei komplexen Programmen eine Menge Zeit einspart (Eschen, 1998).

2.1.5 Kapselung

Das Prinzip der Kapselung sorgt dafür, dass ein Objekt zu einer abgeschlossenen Einheit wird, die über definierte Schnittstellen nach außen hin kommuniziert, aber alle Details der Realisierung verbirgt (information hiding). Hierdurch werden Attribute und Methoden eines Objekts zusammengefasst, also Daten und Funktionalität an gleicher Stelle definiert.

Der Zugriff auf Attribute des Objekts geschieht grundsätzlich über Methoden, sog. getAccessors und setAccessors. Dieses Vorgehen mag auf den ersten Blick etwas umständlich aussehen, wenn für jeden auszulesenden Wert eine eigene Methode definiert werden muss. Der entscheidende Vorteil ist allerdings, dass aufrufende Objekte nicht geändert werden müssen, falls sich die Interna des aufgerufenen Objekts ändern. Hierbei ist die Protokollebene (Methoden-Interface) als Zwischenschicht zu verstehen, die ein stabiles Interface nach außen darstellt. Nachträgliche Änderungen, die ausschließlich unterhalb der Schicht durchgeführt werden, also im Objektkern, führen zu keiner Veränderung des Interfaces. Somit haben die Änderungen auch keine Auswirkung nach außen (Eschen, 1998).

2.1.6 Polymorphismus

Polymorphismus bedeutet, dass dieselbe Methode Objekten verschiedener Klassen zugeordnet werden kann. Für jede Klasse wird eine spezielle Version einer polymorphen Methode definiert. Jedes Objekt kennt seine Klasse und daher wird die richtige Version der Methode ausgewählt. Unabhängig von bestehenden Versionen können für andere Klassen neue Versionen der polymorphen hinzugefügt werden. Das Konzept des Poymorphismus wird auch Überladen genannt. Für objektorientierte Software spielt Polymorphismus eine entscheidende Rolle im Zusammenhang mit Vererbung (Damrath, 1998).

2.1.7 Vererbung

Vererbung bedeutet, dass Attribute und Methoden nach dem Konzept der „Eltern – Kind“ Beziehung zwischen Klassen geteilt werden. Eine Klasse kann als generalisierte Oberklasse definiert und anschließend als stärker spezialisierte Unterklasse verfeinert werden. Jede Unterklasse erbt alle Attribute und Methoden zu seiner Spezialisierung hinzu. Eine Unterklasse

kann als eine spezielle Version seiner Oberklasse angesehen werden (Damrath, 1998).

2.2 Objektorientierte Analyse

2.2.1 Methoden zur Klassenfindung

Der Entwicklungsprozess für objektorientierte Anwendungssysteme lässt sich grob in drei Schritte gliedern. Im einzelnen sind dies:

- objektorientierte Analyse
- objektorientiertes Design
- Realisierung

Im Rahmen der objektorientierten Analyse wird derjenige Ausschnitt der realen Welt, der softwaretechnisch unterstützt werden soll, hinsichtlich existierender Objekte, Attribute, Operationen, Klassen und Beziehungen untersucht. In der Designphase erfolgt eine evolutionäre Weiterentwicklung des Analysemodells um implementationsspezifische Informationen (zum Beispiel Programmiersprachenspezifika). Zudem sind Fragen bezüglich der Systemarchitektur Gegenstand des objektorientierten Designs. In der Realisierung wird schließlich objektorientierter Programmcode erzeugt.

Aus der Vorgehensweise bei der Realisierung objektorientierter Softwaresysteme wird deutlich, dass das objektorientierte Modell der Analysephase und hier insbesondere das Klassendiagramm, dessen Erstellung der erste Schritt auf dem Weg zu einem objektorientierten Modell darstellt, eine grundlegende Bedeutung für den gesamten Softwareentwicklungsprozess besitzt. Das Kernproblem liegt somit in der Ermittlung der Klassen.

Eine methodische Vorgehensweise bei der Ableitung von Klassendiagrammen gewinnt insbesondere unter den Gesichtspunkten der Nachvollziehbarkeit und Transparenz der angestellten Überlegungen immer stärker an Bedeutung. Dies ist vor allem bei der Entwicklung komplexer Softwareprodukte sehr wichtig, da hierbei in der Regel mehrere Personen über einen langen Zeitraum beteiligt sein können (Eversheim et al., 1998).

2.2.2 Identifikationsmethoden zur Klassenbildung

Bei der Klassenfindung innerhalb der objektorientierten Analyse und des objektorientierten Designs wurden die Methoden der heuristischen und die der iterativen Vorgehensweise verwendet. Deren Methodik sei an dieser Stelle ansatzweise erläutert:

Heuristische Klassenfindung

Bei der Entwicklung objektorientierter Softwareprodukte mit geringer Komplexität wird häufig ein pragmatischer Ansatz gewählt. Dieser Ansatz kann dann gewählt werden, wenn nur eine Person bzw. Personengruppe beteiligt ist. Die Klassenfindung erfolgt in einem inkrementellen iterativen Prozess: Die zunächst gefundenen Klassen werden sukzessive mit Attributen und Operationen versehen und in Beziehung zu anderen Klassen gesetzt.

Iterative Vorgehensweise

Booch empfiehlt eine spiralförmige, iterative Vorgehensweise, in der immer wieder zwischen objektorientierter Analyse und objektorientiertem Design gependelt wird. Diese Vorgehensweise wird damit begründet, dass die Güte der Klassifizierung erst in späteren Phasen des Designs bewertet werden kann. In diesen späteren Phasen kann entschieden werden,

- aus einer bestehenden Klasse Unterklassen zu erzeugen (Ableitung durch Einrichtung einer Vererbungsbeziehung);
- eine große Klasse in mehrere kleine zu unterteilen (Faktorisierung durch Einrichtung einer Utility-Klasse);
- eine große Klasse durch Zusammenfassung mehrerer kleiner Klassen in einer neu einzuführenden Klasse abzubilden (Abstraktion durch Einführung einer abstrakten Klassen, von der die betroffenen Klassen erben) (Eversheim et al., 1998).

2.3 Objektmodellierung

Ein objektorientiertes Modell ist eine Menge von kooperierenden Objekten, die man aus drei verschiedenen Sichten betrachten kann. Diese Modellsichten führen zu den drei Teilmodellen: Objektmodell, dynamisches Modell und funktionales Modell. Die Teilmodelle werden einheitlich als ungerichtete oder gerichtete Graphen dargestellt. In der Literatur über objektorientierte Programmierung werden verschiedene graphische Notationen beschrieben. Die graphische Notation von Rumbaugh et al. wird weit verbreitet angewendet und für objektorientierte Softwareentwicklung angenommen (Damrath, 1998). Diese Notation wird in dieser Arbeit verwendet. Im Folgenden sollen nur das Objektmodell und das zugehörige Klassenmodell knapp definiert werden. Es sei weiterführend auf Priestley (2000) verwiesen.

2.3.1 Abstraktion

Ein *Objektmodell* ist eine strukturierte Menge von Objekten. Die Struktur wird durch Relationen zwischen den Objekten spezifiziert. Im Allgemeinen werden nur binäre Relationen betrachtet, wobei jede binäre Relation aus einem Paar von Objekten besteht.

Ein Objektmodell wird als Graph in einem *Objektdiagramm* dargestellt. Die Objekte sind die Knoten des Graphen. Die binären Relationen zwischen den Objekten sind die Kanten des Graphen. Abhängig von den Eigenschaften und Relationen können die Kanten gerichtet oder ungerichtet sein.

Die Objekte eines Objektmodells sind klassifiziert. Jede Klasse stellt eine Menge von äquivalenten Objekten dar, die eine Untermenge der Objektmenge ist. Eine Klasse ist eine Abstraktion von äquivalenten Objekten. Ein Objekt ist eine Instanz einer Klasse.

Ein *Klassenmodell* ist eine *Abstraktion* von äquivalenten Objektmodellen. Es ist eine strukturierte Menge von Klassen. Die Struktur wird durch eine Menge von binären Relationen zwischen den Klassen aufgebaut. Jede Relation zwischen zwei Klassen repräsentiert die Relation zwischen korrespondierenden Mengen äquivalenter Objekte in einem Objektmodell.

Ein Klassenmodell wird als ein Graph in einem *Klassendiagramm* dargestellt. Die Klassen sind die Knoten des Graphen. Die Relationen zwischen zwei Klassen sind die Kanten des Graphen. Die Kanten können gerichtet oder ungerichtet sein (Damrath, 1998).

2.3.2 Klassen und Objekte

Eine *Klasse* ist spezifiziert durch einen Klassennamen. Sie beinhaltet sowohl eine Reihe von Attributen mit ihren Namen, ihrem Typ und ihren Initialwerten, als auch eine Reihe von Operationen mit ihren Parametern und ihren Ergebnissen. Eine Klasse wird im Klassendiagramm als Rechteck dargestellt. Virtuelle Klassen, d. h. Klassen die nicht als Objekt instanziiert werden können, werden im Diagramm durch einen kursiv geschriebenen Namen gekennzeichnet.

Ein *Objekt* ist spezifiziert durch seine Klasse und einen Objektnamen. Jedes Objekt kennt seine Klasse. Der Zustand eines Objekts wird beschrieben durch die Werte seiner Attribute. Ein Objekt wird im Objektdiagramm durch ein Rechteck mit abgerundeten Ekken dargestellt.

Eine *Relation* zwischen einem Objekt mit seiner Klasse nennt man eine Instanzierungsrelation. Sie wird durch einen gestrichelten Pfeil wie in Abb. 2-1 dargestellt (Damrath, 1998).

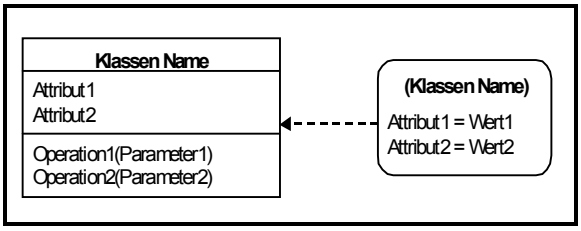

Abb. 2-1: Relation zwischen einem Objekt und seiner Klasse (Damrath, 1998)

2.3.3 Assoziationen

Eine *Assoziation* repräsentiert eine binäre Relation zwischen zwei Klassen (Abb. 2-2). Diese Relation ist symmetrisch. Sie wird im Graph des Klassendiagramms als ungerichtete Kante dargestellt. Eine Assoziation einer Klasse mit sich selbst ist eine reflexive Relation und repräsentiert binäre Relationen zwischen korrespondierenden Objekten mit einem Objektmodell.

Eine Assoziation in einer Klasse wird spezifiziert durch Eigenschaften der Objektrelation in dem korrespondierenden Objektmodell. Die Eigenschaften beschreiben die Vielfalt, die Möglichkeiten und die Ordnung der Objektrelationen. Sie werden hier im Klassendiagramm durch Angabe der Relation (beispielsweise 1-1, 1-n, n-m etc.) gekennzeichnet. Eine Klasse kann einer Relation zwischen zwei Klassen zugeordnet sein. Man nennt das eine Assoziationsklasse. Sie beinhaltet Attribute und Operationen, die nicht einem der assoziierten Objekte zugeordnet werden kann (Damrath, 1998).

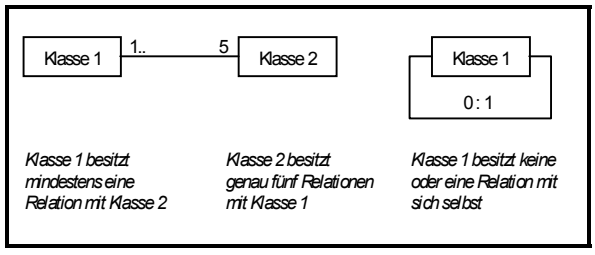

Abb. 2-2: Binäre Relationen zwischen Klassen (Damrath, 1998)

2.3.4 Aggregationen

Eine Aggregation repräsentiert eine „ist Teil von“ Relation zwischen zwei Klassen. Die Relation ist asymmetrisch und transitiv. Sie wird im Graphen des Klassenmodells als gerichtete Kante mit einem Diamanten (einer Raute) als Symbol für die Richtung dargestellt.

Symmetrie: $(A \in B) \Rightarrow (B \in A)$

Transitivität: $(A \subset B) \wedge (B \subset C) \Rightarrow (A \subset C)$

Eine Aggregation in einem Klassenmodell wird spezifiziert durch *Eigenschaften* von Objektrelationen in dem korrespondierenden Objektmodell. Diese Eigenschaften beschreiben die Vielfalt, die Optionalität und die Ordnung der Objektrelationen. Sie werden auf die gleiche Weise gekennzeichnet wie die Assoziationen (Abb. 2-3; Damrath, 1998).

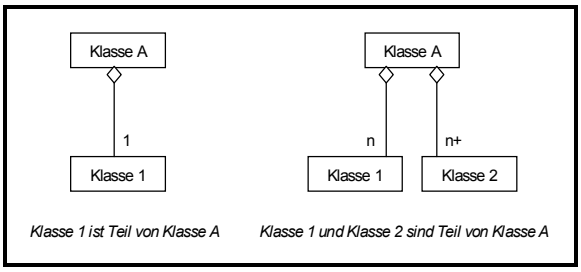

Abb. 2-3: Aggregation: "Teil von..." Beziehungen (Damrath, 1998)

2.3.5 Generalisierung

Eine *Generalisierung* ist eine Relation zwischen einer generellen und einer speziellen Klasse. Eine spezielle Klasse ist eine Version einer generellen Klasse. Die Relation „Version von“ ist asymmetrisch und transitiv. Generelle und spezielle Klassen werden auch Super- bzw. Unterklasse genannt. Eine Generalisierung wird als eine gerichtete Kante in Graphen des Klassendiagramms mit einem Dreieck als Symbol für die Richtung dargestellt (Abb. 2-4).

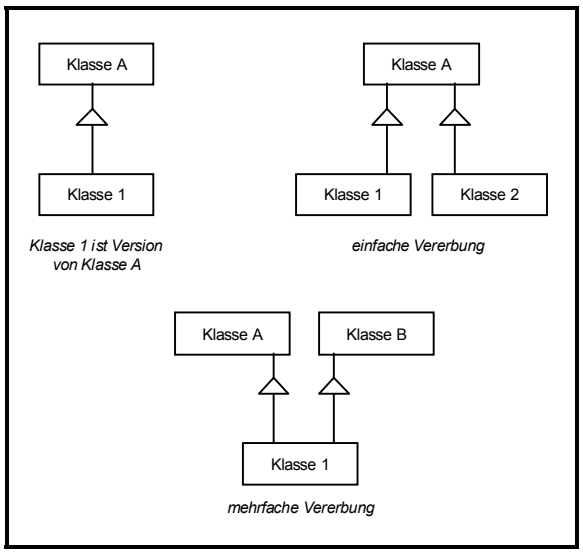

Abb. 2-4: Generalisierung/Vererbung (Damrath, 1998)

Spezielle Klassen werden von einer allgemeineren Klasse abgeleitet. Jede spezielle Klasse erbt alle Attribute und Methoden von der generellen Klasse. Eigene Attribute und Methoden können den geerbten Attributen und Methoden hinzugefügt werden. Geerbte Methoden können für die spezielle Klasse redefiniert werden. Die *Vererbung* ist einfach, wenn die spezielle Klasse von nur einer generellen Klasse abgeleitet wird oder mehrfach, wenn die spezielle Klasse von mehr als einer generellen Klasse abgeleitet wird.

Ein Objekt ist gleichzeitig eine Instanz einer speziellen Klasse und aller generellen Klassen, von der die spezielle Klasse abgeleitet wurde (Damrath, 1998).

2.4 Simulationsmodelle

2.4.1 Eigenschaften

In diesem Abschnitt werden die Eigenschaften einer Auswahl von Simulationsmodellen beschrieben. Zu den wesentlichen Eigenschaften dieser Modelle gehören die Fähigkeiten folgende Funktionen darzustellen:

Parallele Prozesse

Die Modelle enthalten im Allgemeinen mehrere Elemente, die selbständig und nebeneinander über eine gewisse Zeit in Funktion stehen, sich gegenseitig beeinflussen können und somit parallel auf der Zeitachse ablaufen. Diesen Modellen ist insbesondere eigen, dass sie selber in Zeitrichtung ablaufen, wobei die Aktion nicht nur bei einem Element liegt (wie es zum Beispiel bei einem klassischen Computerprogramm der Fall wäre).

Standardisierte Funktionen

Gewisse mathematische Funktionen darunter oft solche, die den Zufall darstellen können, treten im Modellaufbau häufig auf, bzw. sind sie zur modellmäßigen Darstellung einer Wirklichkeit geeignet. Dazu gehören auch Funktionen, die Schwingungen, Wachstumsprozesse oder Abklingverhalten ausdrükken. Auch hier ist die Zeitabhängigkeit gegeben.

Beschr. der Beziehungen zwischen den Elementen

Diese Beziehungen können auf sehr unterschiedliche Arten beschrieben werden. In sehr mathematischen deterministischen Modellen sind das Gleichungen und Ungleichungen, in anderen können es wahrscheinlichkeits- oder allgemein bedingungsgesteuerte Einflüsse sein.

Zweckmäßige Ausgabe der Ergebnisse

Der Ablauf des Modellgeschehens soll auf eine illustrative Weise sichtbar werden. So ist für eine sinnvolle Datenausgabe mit entsprechenden Kenngrößen zu sorgen (Bauknecht et al., 1976).

2.4.2 Klassifizierung

Obwohl jede Modellbildung grundsätzlich von ihrer Aufgabenstellung ausgehen sollte und nicht vorgefasste Konzepte anderer Modelle kopieren darf, ist dennoch eine grobe Klassifizierung möglich. Sie bietet zudem eine Übersicht von vorhandenen Grundstrukturen.

2.4.2.1 Kontinuierliche Systeme

Die Natur und ein großer Teil der Technik sind auf Kontinuität und Stetigkeit ihrer Eigenschaften ausgerichtet, wobei plötzliche Änderungen unnatürlich sind. Die Übergänge von einem Zustand in einen anderen sind ohne Bruch. Daher sind auch Modelle dieser

Wirklichkeit auf Stetigkeit ausgerichtet. Die die Realität beschreibenden Funktionen ändern sich in beliebig kleinen Zeitschritten nur um infinitesimale Beträge und zu ihrer Beschreibung ist die Differentialrechung das geeignete Instrument. Entsprechende Modelle wären dann Systeme von Differentialgleichungen. Typische Fragestellungen an solche Systeme befassen sich mit stationären Zuständen, Schwingungsverhalten und Belastungsgrenzen (Bauknecht et al., 1976).

2.4.2.2 Systeme diskreter Ereignisse

In vielen beispielsweise organisatorischen Bereichen ist die natürliche Eigenschaft der stetigen Veränderung zwar verständlich, jedoch besteht hier kein Interesse an diesem Detail, sondern an der nächsthöheren Stufe des Systems, in der nur der jeweilige Anfangs- und Abschlusszeitpunkt einer Tätigkeit berücksichtigt wird. Ein System dieser Stufe verzichtet auf die Kenntnis aller Zwischenzustände. Von Bedeutung sind nur die Hauptzustände und der Zeitpunkt ihrer Gültigkeit. Durch die Systematisierungsmaßnahme des Verzichts auf Zwischenzustände wird das natürliche Untersystem zu einem Element in dem organisatorischen System (Bauknecht et al., 1976).

2.4.2.3 Hybride Systeme

In der Praxis treten Fälle auf, die sich nicht in eine strikte Klassifizierung einordnen lassen, da sie sowohl mit kontinuierlichen als auch mit diskreten Ereignissen arbeiten. Diese Systeme werden als hybride Systeme bezeichnet, wenn nach beiden Klassifizierungen (diskret und kontinuierlich) arbeitende Elemente vorhanden sind (Bauknecht et al., 1976).

2.4.2.4 Deterministische oder stochastische Systeme

Bei Simulationssystemen wird zwischen deterministischen und stochastischen Systemen unterschieden. So haben deterministische Planungsverfahren große Bedeutung und Vorteile, da sie im Allgemeinen stabiler und ihre Ergebnisse manuell besser steuerbar sind. Wohingegen Monte-Carlo-Simulationen (bei stochastischen Systemen) diese häufig erwünschten Eigenschaften nicht bieten (Bauknecht et al., 1976).

2.5 Verkehrsflussmodellierung

Die Modellierung von Verkehrsflüssen stellt ein komplexes Problem dar: Es gibt einzelne Wasserfahrzeuge, deren Bewegung sowohl von deren Typ, Funktion und Ladungszustand sowie auch von äußeren Einflüssen (Fahrwasserbreite und –tiefe, Geschwindigkeitsbeschränkungen, Wetterlage etc.) abhängen. Durch die Wechselwirkungen der Fahrzeuge untereinander ergeben sich die Beziehungen der makroskopischen Größen auf dem Wasserweg: Fahrzeugdichte, Fluss (d. h. die Zahl der Fahrzeuge, die in einem bestimmten Zeitintervall einen Querschnitt passieren) und mittlere Geschwindigkeit. Erweitert man die Problemstellung auf Wasserstraßennetze, müssen zusätzlich die Wechselwirkungen an Kreuzungen, Schleusen und Häfen bzw. Stellplätzen berücksichtigt werden.

Für die Verkehrsflussmodellierung gibt es grundsätzlich zwei verschiedene Ansatzpunkte: Zum einen kann man den Verkehrsfluss durch Beziehungen makroskopischer Größen – in der Regel mittlere Geschwindigkeit, Fahrzeugdichte und Gesamtfluss des betrachteten Streckenabschnitts – beschreiben (*makroskopische Modelle*). Andererseits ist es möglich die Bewegungen einzelner Fahrzeuge zu modellieren (*mikroskopische Modellierung*).

Es sollen typische Modelle für die beiden Ansätze angesprochen werden sowie beispielhaft ein oft zur Modellierung von Autostraßenverkehren verwendetes Zellularautomaten-Modell von Nagel und Schrenkenberg. Anhand dieses einfachen Modells soll das hier umgesetzte Prinzip des zellularen Automaten erläutert werden. Bei im Rahmen dieser Arbeit durchgeführte Recherche kam zu dem Ergebnis, dass der Ansatz eines zellularen Automaten bisher noch nicht für Schiffsverkehre angewendet wurde. Es ergeben sich hierbei neue komplexe Problemfälle, so dass die Kenntnis von vergleichbaren einfachen Grundzusammenhängen hilfreich ist (Esser, 1997).

2.5.1 Makroskopisch

Die makroskopischen Modelle werden in *kontinuierliche* und *diskrete* Ansätze unterteilt. Ausgangspunkt der kontinuierlichen Modellierungsansätze ist die Erhaltung der Fahrzeuganzahl in Form der Kontinuitätsgleichung

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial q(x,t)}{\partial x} = 0 \qquad (1)$$

Hierbei bezeichnet $\rho(x,t)$ die orts- und zeitabhängige Fahrzeugdichte und $q(x,t)$ den Fluss. Die Beziehung $q(x,t)$ muss hierbei vorgegeben werden. Mit dem Ansatz $q(x,t)=q(\rho(x,t))$ erhält man die Lighthill-Whitham Gleichung (Lighthill et al., 1964), die unter anderem die M der Zeit durch nichtkontinuierliche Dichtesprünge gekennzeichnete Schockwellen bilden können (Helbing, 1997). Aus diesem Grund wurde sie durch einen Diffusionsterm erweitert; außerdem wurde ein Rauschterm eingeführt, um den nicht-deterministischen Aspekt des Verkehrsflusses zu berücksichtigen.

In einigen Modellansätzen wird neben der Kontinuitätsgleichung auch eine dynamische Geschwindigkeitsregelung berücksichtigt. Beispielsweise

wird in Kerner et al. (1995) ein Ansatz behandelt, bei dem eine Geschwindigkeits-Dichte Relation entsprechend der Fermi-Dirac-Verteilung angenommen wird. Mit diesem Modell ist es möglich, die Entstehung von Dichteclustern durch lokale Störungen des homogenen Verkehrsflusses zu beschreiben.

Die Anpassungen dieses Modells auf reale Begebenheiten beziehen sich durchweg auf Autostraßen sowie deren Netze (Esser, 1997).

2.5.2 Mikroskopisch

In vielen Fällen ist die makroskopische Beschreibung des Verkehrsflusses nicht ausreichend, so dass die einzelnen Fahrzeugbewegungen berücksichtigt werden müssen. Besonders hohe Ansprüche an eine realitätsnahe Modellierung der Wasserfahrzeugbewegungen stellen Untersuchungen von Informations- und Lokalisationssystemen. Aber auch in den Gebieten des Kraftstoffverbrauchs und der Schadstoffemissionen ist die Frage noch nicht geklärt, inwieweit über Standardwerte für bestimmte Fälle hinaus, die durch die Verkehrssituation und –zusammensetzung charakterisiert sind, nicht auch eine mikroskopische Auflösung insbesondere des Beschleunigungsvorgangs erforderlich ist. Die unterschiedlichen Anforderungen an mikroskopischen Modellierungsansätzen haben zu einer weiteren Unterteilung geführt. Während bei *high fidelity* Modellen Beschleunigungs- und Bremsvorgänge realitätsnah nachgestellt werden, enthalten *low fidelity* Ansätze mikroskopische Fehler in der Beschreibung der Fahrzeugbewegung. Diese Unterteilung ist jedoch im Gegensatz zur Unterscheidung zwischen mikroskopischer und makroskopischer Modellierung nicht immer eindeutig.

Die Grundidee des sogenannten *Fahrzeug-Folge-Ansatzes* ist, dass sich die Bewegung eines Fahrzeugs ausschließlich an dem vorausfahrenden Fahrzeug orientiert. Die Grundgleichung

$$a_{n+1}(t+T) = \alpha(v_n(t) - v_{n+1}(t)) \qquad (2)$$

beruht auf dem Ansatz, dass sich die Reaktion auf das Verhalten des vorausfahrenden Fahrzeugs durch *Aktion=Sensitivität x Stimulus* beschreiben lässt. Die Fahrzeuge sind hierbei entgegen der Fahrtrichtung aufsteigend durchnummeriert. Aus den Gleichungen können Geschwindigkeits-Dichte sowie Fluss-Dichte Beziehungen abgeleitet werden, die allerdings mit realen Dichten nicht sehr gut übereinstimmen. Probleme bereitet hierbei auch die Parameteridentifikation, da die Werte m und l nicht als direkt messbare charakteristische Parameter der Fahrzeuge oder des Fahrerverhaltens interpretiert werden können.

Eine Erweiterung der klassischen Fahrzeug-Folge Modelle erfolgte durch Unterscheidung verschiedener Fahrzustände (Folgen, Fahren mit Wunschgeschwindigkeiten und Bremsen), die durch verschiedene Parametersätze charakterisiert sind, in einem pseudophysischen Abstandsmodell nach Wiedemann (Wiedemann, 1974).

Die auf dem Fahrzeug-Folge-Ansatz basierende Modellierung der Fahrzeugbewegungen erfordert einen hohen numerischen Aufwand zur Lösung der entsprechenden Gleichungen für jedes einzelne Fahrzeug. Aus diesem Grund sind Echtzeit-Simulationen mit derart hochaufgelösten Modellen für große Straßennetze derzeit nicht möglich. Grundlegendes Bestreben bei der mikroskopischen Simulation von Straßennetzen ist daher die Fahrzeugbewegungen durch möglichst wenige Regeln zu beschreiben, so dass jedoch die wesentlichen Aspekte des Verkehrsflusses berücksichtigt werden. Der Grundgedanke der diskreten Verkehrsflussmodellierung ist bereits seit längerem bekannt. In Cremer et al. (1986) wurde ein Ansatz vorgestellt, bei dem Fahrzeuge durch Bits dargestellt und mit Hilfe von logischen Operationen bewegt werden.

Der Einsatz zellularer Automaten hat sich auf vielen Gebieten der Physik und Biologie bewährt. Dies liegt darin begründet, dass die Dynamik der betrachteten Systeme durch wenige räumlich und zeitlich diskrete Regeln beschrieben wird und damit Simulationen in den erforderlichen Größenordnungen überhaupt erst möglich werden (Esser, 1997).

2.5.3 Basismodell: zellularer Automat

Das Zellularautomaten-Modell, im folgenden ZA-Modell genannt, das hier exemplarisch aufgezeigt wird, ist die Basis der meisten Modelle zur Simulation von Autos auf Straßen. Die in diesem Abschnitt vorgestellten Regeln können nicht in dieser einfachen Form auch auf Wasserstraßen bezogen werden. Es soll hier nur das Grundprinzip erläutert werden, auf dem später aufgebaut wird.

2.5.3.1 Grundmodell

Die Grundlagen des ZA-Modells sind auf einem linearen Gitter wie in Abb. 2-5 definiert. Hierfür wird die Straße in gleichgroße Zellen (Index i) unterteilt, die entweder ein Fahrzeug enthalten oder leer sind. Die aktuellen Geschwindigkeiten der Fahrzeuge werden durch ganze Zahlen angegeben.

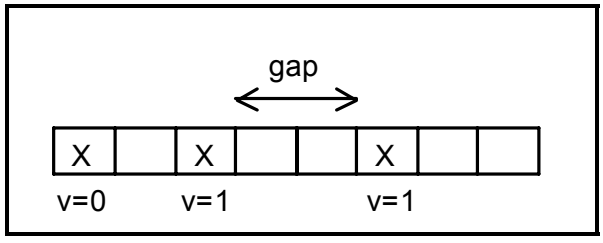

Abb. 2-5: Unterteilung einer Straße in Zellen entsprechend dem ZA-Modell

In jedem Zeitschritt werden die Geschwindigkeiten nach folgenden Regeln bestimmt:

- Beschleunigung

  $v_i \leftarrow \min\{v_i+1, v_{max}\}$
- Abstand halten

  $v_i \leftarrow \min\{v, gap\}$
- zufälliges Abbremsen

  falls rand() < p : $v_i \leftarrow \max\{v_i-1, 0\}$
- Fahrzeugbewegung

  $x_i \leftarrow x_i + v_i$

Hierbei bezeichnet die Größe „gap“ den Abstand zum vorausfahrenden Fahrzeug und „rand()“ eine Zufallszahl im Intervall [0; 1]. Durch die zweite Regel (Abstand halten) werden Unfälle vermieden. Die Fahrzeugbewegungen sind individuell fahrer- und fahrzeugabhängig und werden zusätzlich durch eine Vielzahl äußerer Faktoren beeinflusst (in diesem speziellen Fall Auto: Straßenbeschaffenheit, Wetter, Ablenkung des Fahrers etc.), deren Auswirkungen nicht genau modelliert werden können und daher durch die nichtdeterministische Verzögerung mit einer Wahrscheinlichkeit von „p“ (zufälliges Abbremsen) berücksichtigt werden (Esser, 1997).

2.5.3.2 Parameter

Für die Zelllänge wird üblicherweise l_z=7,5 m gewählt. Dies entspricht der Länge, die ein Fahrzeug (Auto) im dichtesten Stau einnimmt einschließlich des Abstands zum nächsten Fahrzeug. Die mittlere Zahl von Zellen, die ein mit maximaler Geschwindigkeit fahrendes Fahrzeug in einem Zeitschritt zurücklegt, beträgt v^{ZA}_{max}-p. Hiermit gilt für die zeitliche Auflösung

$$\Delta t = \frac{l_z (v^{ZA}_{\max} - p)}{v^{real}_{\max}} = 1{,}0125 \text{ s} \qquad (3)$$

mit v^{real}_{max}=120km/h, v^{ZA}_{max}=5 und p=0,5. Die Auflösung von ≈ 1 s liegt somit in der Größenordnung der Reaktionszeit. Für die Schrittweite der Geschwindigkeiten gilt mit diesen Parametern eine Auflösung von 26,67 km/h.

Gerade in städtischen Straßennetzen liegen die Geschwindigkeitsbegrenzungen deutlich niedriger als der für Autobahnen angenommene Wert. Für jede Straße werden die maximalen Geschwindigkeiten entsprechend gesetzt:

$$v^{ZA}_{\max} = (v^{real}_{\max} / l_z) + 0{,}5 + p \qquad (4)$$

2.5.3.3 Modell-Erweiterungen

Das Grundmodell ist den *low fidelity* Modellen zuzuordnen, da es mikroskopische Fehler enthält. Fahrzeuge können beispielsweise unabhängig von ihrer aktuellen Geschwindigkeit innerhalb einer Sekunde bis zum Stehen abbremsen. Manche Untersuchungen (z. B. fahrzeugfeine Berechnungen von Emissionen) erfordern jedoch die Berücksichtigung realistischer Brems- und Beschleunigungsvorgänge. Im ersten Schritt wurde eine kontinuierliche Version des Grundmodells entwickelt (Krauß et al., 1996), bei der die Fahrzeugbewegungen entsprechen folgender Regeln durchgeführt werden:

- $v_i \leftarrow \min\{v_i+a_{max}, v_{max}, gap\}$
- $v_i \leftarrow \max\{0, v_i\text{-rand() s}\}$
- $x_i \leftarrow x_i + v_i$

Hierbei bezeichnet a_{max} die maximale Beschleunigung und s das maximale Abbremsen aufgrund des statistischen Rauschens.

Zwar vereinfacht die Anwendung des kontinuierlichen Regelsatzes die Kalibrierung des Modells, es ist jedoch hiermit allein noch nicht möglich alle makroskopischen Zustände, die im realen Verkehrsfluss auftreten, nachzustellen. Beispielsweise gibt es bei dieser Form des Ansatzes metastabile Zustände, die dadurch charakterisiert sind, dass Fahrzeuge die Bewegung des vorausfahrenden Fahrzeuges im nächsten Zeitschritt antipizieren. Es hat sich gezeigt, dass bereits die Erweiterung des ZA-Modells um realistische Beschleunigungs- und Bremsverhalten in einfacher Form ausreicht, um derartige Zustände in der Simulation zu erzeugen (Esser, 1997).

2.5.3.4 Spurwechsel

In der Realität stellen Spurwechsel komplizierte, von der jeweiligen Verkehrssituation abhängige Vorgänge dar, die durch möglichst einfache Regeln in der Simulation so zu berücksichtigen sind, dass die makroskopischen Merkmale realistisch nachgestellt werden. In Chordhury et al. (1997) wurden Regeln für das Spurwechselverhalten eingeführt, die in jedem Updateschritt zusätzlich zu den Regeln der Fahrzeugbewegung durchgeführt werden. Hierfür werden der Abstand zu Vordermann auf der aktuellen Spur (gap_1), zum Vordermann auf der Zielspur (gap_2) und zum Nachfolger auf der Zielspur (gap_3) berücksichtigt. Für einen Spurwechsel müssen folgende Bedingungen erfüllt sein:

- $v_i^d > gap_1$; aktuelle Spur ungünstig
- $v_i^d \leq gap_2$; Zielspur günstig
- $gap_3 > v_{max}$; genügend Abstand zum Nachfolger

Hierbei bezeichnet $v_i^d = \min\{v_i+1, v_{max}\}$ die Wunschgeschwindigkeit des Fahrzeugs für den aktuellen Zeitschritt. Diese Regeln beschreiben symmetrische Spurwechsel, wie sie für zweispurige Fahrwege in eine Richtung charakteristisch sind, d. h. es wird nicht zwischen Wechseln von rechts nach links und umgekehrt unterschieden. Zur Berücksichtigung von asymmetrischen Spurwechseln bei beispielsweise Rechtsfahrgebot müssen weitere Bedingungen für den Wechsel von links nach rechts erfüllt sein:

- $gap_1 > 2\ v_i^d$
- $v_i^d \leq gap_2$
- $gap_3 > v_{max}$

Ein Fahrzeug wechselt demnach erst zurück auf die rechte Spur, wenn der Vordermann auf der linken Spur weit entfernt ist und es auf der rechten Spur nicht behindert wird.

Die Spurwechsel werden in jedem Zeitschritt vor den Fahrzeugbewegungen ausgeführt. Hierbei werden die entsprechenden Bedingungen für die Fahrzeuge entgegen der Fahrtrichtung auf den Straßen überprüft, da sonst ein direkt vor einem Hindernis stehendes Fahrzeug aufgrund der nachfolgenden Fahrzeuge, die in diesem Fall zuerst auf die freie Spur wechseln würden, unrealistisch lange warten müsste (Esser, 1997).

3 Umsetzung

3.1 Methodik

In dieser Arbeit wurde eine für die Simulation von Schiffsbewegungen neue Methodik entwickelt. Zum Zeitpunkt dieser Anfertigung wurden von mir keine Hinweise darauf gefunden, dass bisher der Versuch unternommen wurde Schiffsbewegungen mikroskopisch (in diesem Fall mittels zellularer Automaten) aufzulösen. Bisherige Untersuchungen haben mikroskopische Effekte wie beispielsweise Überholvorgänge in der Regel durch Wahrscheinlichkeitsannahmen wie bereits in Koehler (1968) aufgezeigt oder werden gar unter Annahme einer genügend geringen Verkehrsdichte völlig unbetrachtet gelassen. In Zimmermann et al. (1998) wird jedoch darauf hingewiesen, dass diese Effekte bei unterschiedlich schnellen Schiffen nicht zu vernachlässigen sind.

Es konnte ein strukturiertes, modular aufgebautes und leicht erweiterbares Konzept für die mikroskopische Simulation von Schiffsbewegungen entwickelt werden und in einem Simulationsprogramm umgesetzt werden. Hierbei kamen Techniken aus mehreren Bereichen zur Anwendung:

- Graphentheorie
- Objektorientierte Analyse/Modellierung
- Techniken der Zellularautomaten
- Techniken der Algorithmenoptimierung
- Fuzzy-Logik im Zusammenhang mit Pseudo-Intelligenz

3.2 Abbildung des Simulationsgebiets

3.2.1 Graphensicht

Das Simulationsgebiet kann im allgemeinen Fall ein beliebiger Teil eines von Wasserfahrzeugen befahrbaren Gebiets sein. Hierbei sind nur die Wasserwege bzw. befahrbare Gewässer von Interesse, so dass andere Elemente nicht mitmodelliert werden müssen. In die Abbildung werden nur die zu betrachtenden Wasserwege und diese beeinflussende bauliche oder geographische Elemente übernommen. Beispielsweise muss ein Berg in aller Regel nicht modelliert werden, eine Brücke, die die Durchfahrtshöhe von Schiffen einschränkt jedoch berücksichtigt werden.

Ferner muss das Modell die Funktionalität bieten, Fahrtrouten zu ermitteln und anzugeben, wenn nicht nur zufällige Schiffsbewegungen simuliert werden sollen und keine konkreten Situationen betrachtet werden sollen.

Fahrwegsfunktionalitäten können mit Hilfe eines hinterlegten Wegegraphen geboten werden. Ein beliebiges Netz aus Fahrwegen wird über einen Graphen durch *Knoten* und *Kanten*, die diese Knoten verbinden, abgebildet.

3.2.2 Definition von Routen

Die Knoten sind in diesem Graphen (Abb. 3-1) mögliche Zielpunkte einer Fahrtroute. Wird den Kanten eine Gewichtung hinzugefügt, kann über einen so aufgebauten Graphen der günstigste Weg zwischen zwei Knotenpunkten ermittelt werden, wobei nach unterschiedlichen Gesichtspunkte wie beispielsweise Fahrzeit oder Kosten unterschieden werden kann.

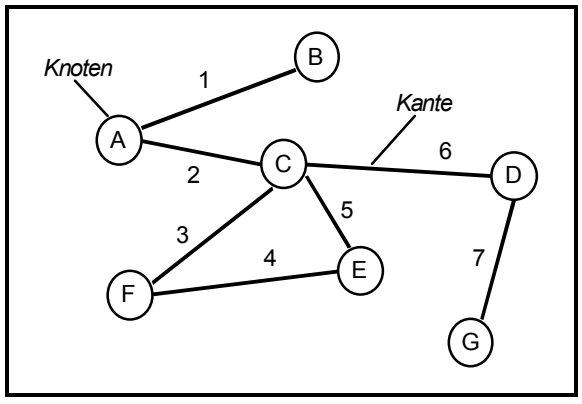

Abb. 3-1: Allgemeines Netz aus Knoten und Kanten

In dieser Abstraktion bewegen sich Schiffe über Kanten von einem Knoten zu einem anderen. Ein Schiff in

Knoten A mit der Fahrtroute {C, E, F} fährt somit über die Kanten 2, 5 und 4; in dieser Reihenfolge. Es lässt sich durch Angabe einer Folge von Zielknoten eindeutig eine Fahrtroute bestimmen.

Sobald ein Schiff oder allgemein ein Vehikel den ersten Zielknoten erreicht, wird dieser Knoten aus der Fahrtroutenliste entnommen. Käme das Schiff in diesem Beispiel über die Kante 2 in Knoten C an, wird dieser aus der Liste entfernt, so dass die daraus resultierende Fahrtroute durch {E, F} gekennzeichnet wäre.

3.2.3 Fahrwegwahl: Routing

Das in dem Beispiel angesprochene Schiff befinde sich nun im Knoten C und die folgenden Fahrzielknoten sind E und F. Mit der bisher beschriebenen Methode ist es nicht ohne Erweiterungen möglich in einem Knoten zu entscheiden welche Kante, die dem Fahrweg entspricht, zu wählen ist. Jeder Knoten muss daher als *Router* agieren und seine benachbarten Knoten mit zugehörigen Verbindungskanten kennen.

Anhand von *Routingtabellen* (Abb. 3-2) kann in jedem Knoten entschieden werden, welches der zu wählende Fahrweg ist, der zum nächsten Zielknoten aus der Liste der Fahrtroute führt.

Knoten C:
Routingtabelle:

Kante	Nachbarknoten
2	A
3	F
5	E
6	D

Abb. 3-2: Routingtabelle von Knoten C

3.2.4 Erweiterung der Graphensicht zur Objektsicht

Die in diesem Abschnitt besprochenen Objekte sind innerhalb der Sichtenaufteilung nicht als instanziierte Klassen im Sinne der Objektorientierung zu verstehen sondern als Objekt im Sinne eines allgemeinen Gegenstands oder einer Einheit.

Die besprochenen Knoten und Kanten der klassischen Graphentheorie müssen um Eigenschaften erweitert werden, um Positionsangaben eines Schiffes in einem Simulationsnetz wiederzugeben und verarbeiten zu können. Aus Graphensicht sind Kanten nur Relationen zwischen Knoten. Folglich kann sich kein Element „auf einer Relation befinden“. Den Kanten werden Funktionalitäten zur Lokalitätsbestimmung hinzugefügt und als Objekte anstelle von Relationen angesehen, um Simulationsalgorithmen auf ihnen durchführen zu können. Im Folgenden werden diese Objektkanten *Sektionen* genannt. Innerhalb dieser Sektionen können sich Einheiten (beispielsweise Schiffe) auf bestimmten Punkten befinden. Auch die Eigenschaften der Knoten im Netzverbund müssen neu definiert werden, da sie innerhalb der Simulation spezielle Aufgaben übernehmen sollen. Die Knoten stellen in dem Simulationsnetz Punkte besonderer Aktionen dar. Sie sind beispielsweise als Wasserstraßenkreuz, Schleuse, Liegestelle, Hafen etc. zu verstehen. Ein Knoten umfasst nach dieser Vorstellung mehr als nur einen Wegpunkt in einer Fahrtstrecke. Knoten, die um Objekteigenschaften erweitert sind, werden hier als *Aktionspunkte* bezeichnet.

Sektionen und Aktionspunkte sollen weiterhin alle Eigenschaften einer Kante bzw. eines Knotens haben. Es handelt sich nur um eine Erweiterung der Grundelemente, die auf der Graphensicht aufbaut.

3.2.5 Gruppieren von Eigenschaften der Objektsicht

Wie bereits beschrieben, enthalten Sektionen und Aktionspunkte eine Anzahl von Funktionalitäten. Innerhalb dieser funktionellen Eigenschaften lassen sich Gruppen erkennen, die für gleiche Anwendungsfälle immer vorhanden sein müssen. Wasserstraßenkreuze besitzen in jeder Form Kreuzungscharakter, analog Schleusen Häfen etc.. Genauso sind beispielsweise bei Kanälen und sonstigen Wasserstraßen eine Reihe von ähnlichen Ausprägungen aufzuzählen. Die einzelnen Schnittmengen der Attribute und Funktionen können in einer neuen Schicht gruppiert werden. Diese Schicht wird im Folgenden Metaschicht genannt.

3.2.6 Spezialisierte Objekte

Ein spezialisiertes Objekt geht aus seiner Gruppe der Metaschicht hervor. Es hat spezielle Ausprägungen und Eigenschaften, die jedoch parametrisierbar gehalten sind. Beispielsweise kann dies ein spezieller Schleusentyp sein, der je nach Standort unterschiedliche Schleusungszeiten, Kapazitäten etc. besitzt. Die Funktion ist aber fest definiert.

3.2.7 Strukturierung durch Schichtenmodell

Die Abbildung des Simulationsgebietes wurden in mehrere Teile bzw. Sichten geteilt. Zusammenfassend lässt sich hieraus ein Schichtenmodell bilden mit den Schichten:

- Graphenschicht
- Objektschicht
- Metagruppenschicht
- Spezialisierungsschicht

Die Einteilung in vier Schichten erlaubt eine gute strukturelle Erfassung der Problematik, ohne die Überschaubarkeit einzuschränken, was bei einer zu großen Zahl von Schichten der Fall wäre. Dieses Modell ist im nächsten Schritt, der objektorientierten Analyse, sehr hilfreich bei der Identifizierung und Eingliederung von Klassen.

3.2.8 Objektorientierte Analyse des Gebiets

Mit den Methoden der objektorientierten Analyse lassen sich unter Berücksichtigung der Vorüberlegungen und definierten Schichten die Klassen in Abb. 3-3 finden.

Die Klassennamen sind so gewählt, dass anhand ihres Namens auf ihre Funktionalitäten geschlossen werden kann. Jede Klasse ist eindeutig einer Schicht zugewiesen, die die Klassenmethodik festlegt.

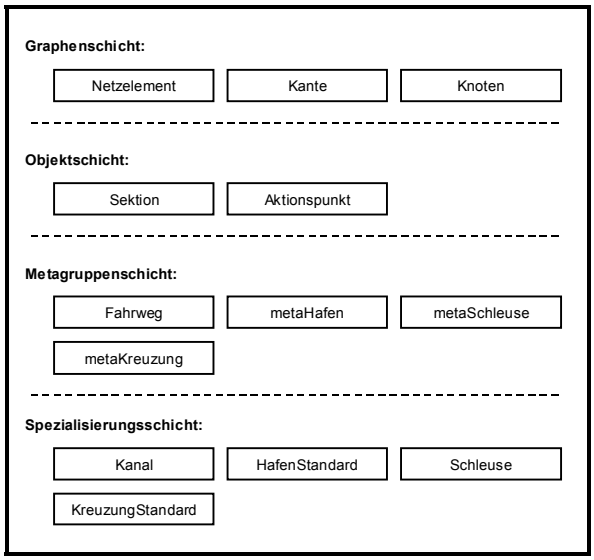

Abb. 3-3: Identifizierte Klassen im Schichtenmodell

3.2.9 Objektorientierte Modellierung des Gebiets

Die gefundenen Klassen werden zur Illustrierung der Abhängigkeiten in einem Klassenmodell (Abb. 3-4) dargestellt. Hierbei wird die in den Grundlagen beschriebene Notation von Rumbaugh verwendet.

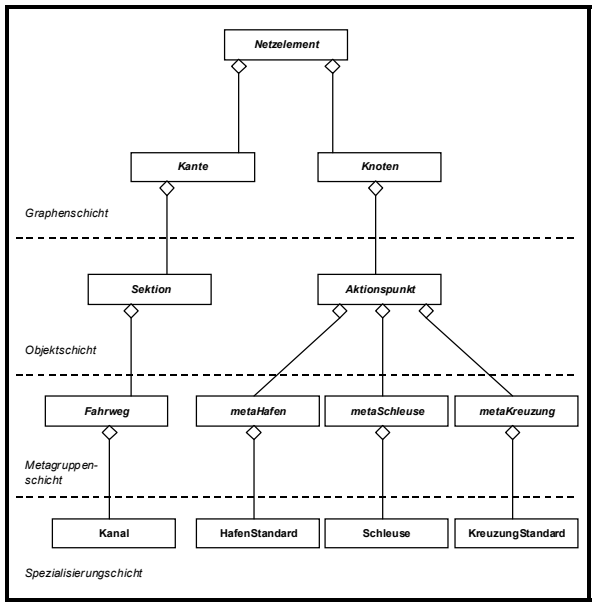

Abb. 3-4: Klassenmodell des Simulationsgebiets

3.3 Datenerfassung

Die für eine Simulation nötigen Randbedingungen werden zum größten Teil durch die analysierte Abbildung des Simulationsgebiets gegeben. Auch bei den Anfangsbedingungen ist eine Systemanalyse durchzuführen, um für die Simulation nötigen Parameter zu bestimmen.

Die Parameter können in statische und dynamische Daten unterschieden werden. Statische Daten bleiben während eines gesamten Simulationsdurchlaufs konstant, wohingegen dynamische Werte variieren können, jedoch nicht müssen. Randbedingungen sind durch ihre Definition statische Daten und Anfangsbedingungen häufig dynamische. Beispielsweise sind die Abmessungen eines Kanals als statisch anzusehen – der Zustand einer Schleuse als dynamisch.

Die Wahl der Parameter sollte pragmatisch nach „so wenige wie möglich, so viele wie nötig“ erfolgen. Ziel einer Simulation sollte es sein, mit minimalem Aufwand ein Maximum an Realitätsnähe wiederzugeben. Bei der Parameteridentifikation muss in diesem Fall sowohl die Seite der vorhandenen oder ermittelbaren Parametern bearbeitet werden als auch die Seite der gewünschten Parameter, die auch nicht feststellbare Werte haben können.

Wenn ein Simulationsprogramm erweiterbar gehalten werden soll, was bei dieser Arbeit der Fall ist, muss selbst dieser Aspekt bei der Parameterwahl berücksichtigt werden. Dies kann an manchen Stellen dazu führen, dass Attribute eingeführt werden, die mit Standardwerten fest belegt sind.

Bei der Entwicklung der Klassen und der Programmstruktur wurde an sehr vielen Stellen die iterative Modellierungsmethode verwendet. Aus diesem

Grund ist kein umfangreicher und zeitaufwändiger Parameterkatalog erstellt worden, aus dem Attribute und eventuell ersichtliche Eigenschaften direkt ausgewählt oder abgeleitet werden. Die Beschreibung der gewählten Parameter erfolgt innerhalb der Beschreibungen der Klassen.

3.3.1 Zeitkomponente von dynamischen Daten: „Timestamps“

Dynamische Daten können sich im Zeitverlauf verändern. Zur Illustrierung sei die Koordinatenposition eines Schiffs genannt. Nicht nur die Werte der Koordinaten sind von Interesse sondern auch der Zeitpunkt der Wertnahme. Zwei Schiffe können zu unterschiedlichen Zeiten die gleiche Koordinatenposition haben. In einem Simulationsmodell, das zu einem Zeitpunkt gestartet werden soll, dürfen sich jedoch nicht zwei Schiffe auf dem selben Punkt befinden. Auch von anderen dynamischen Vorgängen ist der Zeitpunkt des Eintretens von Bedeutung.

Ein weiteres Problem ergibt sich aus der Simulation heraus: Es existieren zu bestimmten Zeitpunkten real gemessene Koordinatenpositionen von Schiffen. In der Simulation werden zu neuen Zeitpunkten (die im Vergleich zur realen Zeit nicht unbedingt in der Zukunft liegen müssen) theoretische Koordinatenpositionen ermittelt. Es reicht nicht aus, nur Zeitpunkte zu unterscheiden, sondern es muss auch zwischen Realzeit und Simulationszeit unterschieden werden.

Eine andere Problematik sind zeitabhängige Ereignisse oder Eigenschaften. Wenn beispielsweise eine Schleuse zu bestimmten Uhrzeiten oder Tagen nicht in Betrieb ist, muss die reale Zeit in die Simulation einfließen, um auf diese Effekte reagieren zu können.

Um die angesprochenen Fälle in der Simulation korrekt behandeln zu können, wurde das Zeitstempelkonzept (*Timestamp*) eingeführt. Ein Timestamp ist ein Zeitstempel in Realzeit. Dieser Stempel gibt innerhalb der Simulation eine korrekte, real existierende Zeitangabe mit Datum wieder. So können bei Langzeitprognosen Sonderregelungen (beispielsweise kein Schleusungsbetrieb an Feiertagen) realisiert werden. Da es sich um eine Datums- und Uhrzeitangabe handelt, können Simulationsergebnisse mit Messergebnissen direkt verglichen werden und als Rückkopplung in das Modell einfließen.

3.3.2 Eigenschaftsstrukturierung bei Simulationsfahrzeugen

Die Eigenschaften und Attribute der in der Simulation vorkommenden Einheiten (i. d. R. Schiffe) wurden ähnlich wie die Abbildung des Simulationsgebietes in ein vierschichtiges Schema eingegliedert (Abb. 3-5). Dieses Schema führt auf folgendes Klassenmodell:

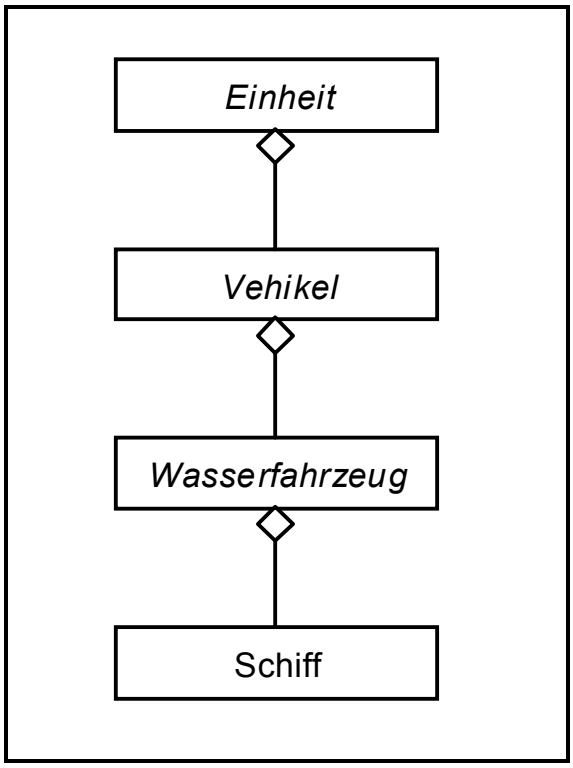

Abb. 3-5: Klassenmodell der Simulationseinheiten

Eine Einheit ist ein allgemeiner Körper, der sich innerhalb des Fahrwassers befinden kann. Unter Vehikel werden bewegliche Einheiten zusammengefasst. Sie können eine Geschwindigkeit besitzen. Wasserfahrzeuge sind spezielle Vehikel bzw. allgemeine Schiffe. In dieser Klasse sind allen Wasserfahrzeugen gemeine Eigenschaften zusammengefasst wie beispielsweise Breite, Tiefgang etc.. In der Klasse Schiff sind Schiffe enthalten, die sich aus eigener Kraft fortbewegen können. Sie besitzen zudem Parameter über deren Beschleunigungs- und Bremseigenschaften.

3.3.3 Georeferenzierung

Die mikroskopische Simulation von Fahrzeugbewegungen ist sehr rechenintensiv. Es werden einfache Strukturen und Berechnungsoperationen gewählt, um die Rechenzeit für einen Simulationsdurchlauf möglichst gering zu halten. Für die Simulation werden daher einfache Koordinatenangaben verwendet. Angaben über den Verlauf von Wasserstraßen und genaue Positionen von beispielsweise Schleusen und Häfen werden separat behandelt und können hoch genau in jedem gewünschten Format angegeben werden.

Wasserstraßen werden durch einen Polygonzug von Geraden approximiert. Auf diese Weise können längere gerade Strecken durch nur zwei Punkte beschrieben werden, da kein Raster vorgegeben ist, auf das die Streckenpunkte angepasst werden müssen. Gleichzeitig wird die Datenmenge reduziert, wenn nur

in Bereichen von Krümmungen zusätzliche Referenzpunkte angegeben werden.

Die Georeferenzpunkte werden in einer speziell entwickelten Datenstruktur, dem *GeoPunkt*, gespeichert. Eine Wasserstraße erhält eine Menge von Georeferenzpunkten, anhand derer die Lage bestimmt wird. Die Klasse GeoPunkt stellt Methoden zur Interpolation und Abstandsbestimmung zur Verfügung. Mit Hilfe dieser Funktionen können Koordinatenwerte zwischen Simulationskoordinaten und Realkoordinaten in dem angegebenen Format umgerechnet werden. Alle Georeferenzierungen werden über die Klasse GeoPunkt wiedergegeben. Auf diese Weise können durch Anpassung dieser Klasse beliebige Koordinatenformate gewählt werden, ohne weitere Eingriffe in das bestehende Programm vornehmen zu müssen.

3.3.4 Zellulare Container für gleichartige Abschnitte

Innerhalb von Sektionen (allgemeine Fahrwege) werden Bereiche mit gleichartigen Eigenschaften gruppiert. Die Gruppenattribute und –funktionen werden in der Klasse *Abschnitt* zusammengeführt. Ein Abschnitt ist eine lineare Verbindung zwischen zwei Georeferenzpunkten (GeoPunkten). In einem Abschnitt treten keine die Fahrt beeinflussenden Veränderungen auf. Die minimale Fahrwasserbreite und –tiefe ist als nahezu konstant anzusehen.

Für Zellularautomatensysteme stellen Abschnitte Bereiche mit einheitlichen Regeln dar. Für die Verwendung in einem solchen System umfassen Abschnitte Container für Einheiten (im Sinne von Behältnis, Platzhalter). Wobei eine Einheit in dieser Arbeit eine oder mehrere Zellen belegen kann.

Zellularautomaten bestehen im herkömmlichen Sinne aus einer Vielzahl von Zellen, die jeweils einer Speicherregion im Computer entsprechen. Zellen können gefüllt oder leer sein. In dieser Arbeit wird dieser Ansatz leicht neu interpretiert. Es werden keine Zellen modelliert, die mit Inhalt gefüllt werden können oder leer sind, sondern Container für Verweise (intern: Pointer, Zeiger) auf Einheiten im System implementiert. Diese Verweise nehmen sehr wenig Speicher in Anspruch und haben den Rechenzeitvorteil, dass Zellen nicht mit Daten gefüllt oder geleert werden müssen, sondern nur Zeiger angepasst werden. Der größte Vorteil ist die Neuerung, dass auf diese Weise Einheiten (Fahrzeuge) im System direkt identifiziert werden können und ihr Status sofort abfragbar ist. Es gibt keine „anonymen“ Fahrzeuge, deren Herkunft, Ziel und Eigenschaften unbestimmbar sind.

3.3.5 Fahrwegmodellierung: Sektion

Sektionen sind allgemeine Fahrwege und werden als Zellularautomat modelliert. Eine Sektion besitzt als Abbild eines polygonalen Streckenzuges, der einen Fahrweg kennzeichnet, mindestens eine Relation zu einem Abschnitt-Objekt (zellularer Container).

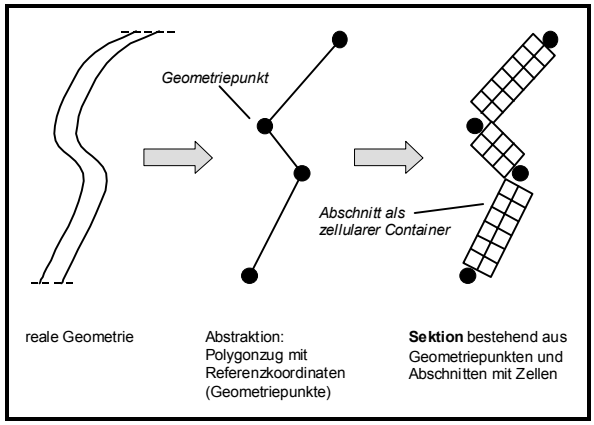

Abb. 3-6: Modellierung einer Sektion

Der Verlauf einer Sektion wird durch mindesten zwei Geometriepunkte der Klasse GeoPunkt beschrieben. Die äußeren Punkte der Sektion sind deckungsgleich mit denen der angeschlossenen Aktionspunkte (Knoten). Durch diese Methodik kann ein krummliniger Verlauf von Wasserstraßenbereichen modelliert werden, ohne Hilfsknoten einführen zu müssen (Abb. 3-6). Es können sehr lange Bereiche als Ganzes modelliert und simuliert werden, wobei die berechneten Positionskoordinaten von Schiffen im System sehr gut den real gegebenen Topologien entsprechen.

Die Sektion fasst alle Abschnitte als einen großen zellularen Container zusammen. Die internen Methoden erlauben einen fließenden Übergang zwischen Zellen verschiedener Abschnitte. Auf diese Weise ist es auch möglich, in einer Sektion über unterschiedliche Abschnitte verschiedene Fahrwasserbreiten und unterschiedliche Anzahlen von Fahrspuren zu verwenden (Abb. 3-7).

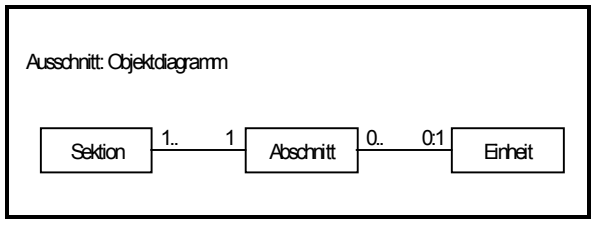

Abb. 3-7: Objektdiagramm zwischen Sektion, Abschnitt und Einheit

3.4 Zellularautomatenkonzept

3.4.1 Adaptive Zellbelegung

Die Umsetzung des Zellularautomatenkonzepts für Schiffe erfordert einige logische Erweiterungen der in den Grundlagen vorgestellten Methodiken, die für Autos anwendbar sind. Diese Methodiken schlagen eine Zellenlänge vor, die der eines typischen Autos

plus minimalem Sicherheitsabstand entspricht. Hier werden Längenunterschiede der Fahrzeuge vernachlässigt. Bei Schiffen sind die genauen Längen und deren Differenzen beispielsweise bei Überholvorgängen und in Schleusen von großer Bedeutung. Aus diesen Gründen wurde eine adaptive Zellbelegung entwickelt. Es wird hierbei davon ausgegangen, dass ein Schiff mehr als eine Zelle belegen kann, um die räumliche Ausdehnung auch im Simulationsprogramm möglichst exakt wiederzuspiegeln. Als *Position* wird die Spitze der Einheitsfront (Schiffsbug) angegeben. Alle Koordinatenangaben einer Einheit beziehen sich auf diesen Punkt.

Alle in der Simulation vorkommenden Einheiten (im speziellen Schiffe) werden mit ihren exakten Größen angegeben. Aus ihrer *Länge* wird die *Anzahl der Zellen* berechnet (Abb. 3-8), die die Einheit in ihrer Länge belegt. Die Zelllänge ist auch bei dieser neuen Methodik konstant mit einer vorgegebenen Länge, um einheitliche Regeln verwenden zu können.

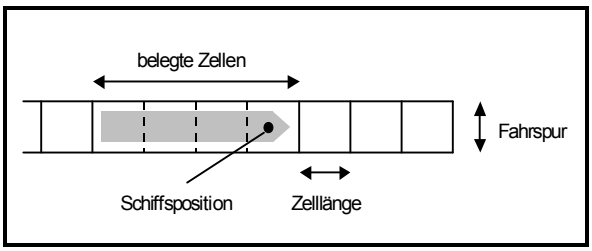

Abb. 3-8: Ein Schiff belegt adaptiv mehrere Zellen in der Länge

Die Breite der Einheiten sowie die der Zellen wird gesondert abgebildet. Das neu entwickelte Konzept sieht Zellen ohne räumliche Ausdehnung in der Breite. Die *Breite* ist als *Fahrspur* zu verstehen und die von der Einheit benötigte Breite wird über die Parameter der Einheit direkt ermittelt. Sind in einem Abschnitt beispielsweise drei Fahrspuren vorhanden (also drei Zellen in der Breite nebeneinander), bedeutet dies, dass dieser Abschnitt für die theoretische Fahrt von drei Schiffen nebeneinander vorgesehen ist. Die Durchführbarkeit eines solchen Manövers muss jedoch überprüft werden. Im einfachsten Fall werden die beteiligten Schiffsbreiten und Sicherheitsabstände mit der Breite an der betreffenden Strecke in Relation gesetzt.

Die unterschiedliche Behandlung von Länge und Breite ermöglicht eine extreme Vereinfachung der verwendeten Algorithmen und Regeln bei größtmöglicher Realitätstreue. Durch die neu entwickelte Methodik der adaptiven Zellbelegung können Längenunterschiede der Simulationseinheiten wiedergegeben werden, ohne die Berechnungsfunktion in starkem Maße zu verkomplizieren und somit die Simulationsgeschwindigkeit herabzusetzen.

3.4.2 Größenbestimmung von Zellen und Geschwindigkeitsstufen

Bei der Fahrt von Wasserfahrzeugen werden sie in der Simulation in diskreten Zeitschritten abhängig von der Geschwindigkeit eine ganzzahlige Anzahl von Zellen (die auch null sein kann) weitergesetzt. Ein Vehikel kann auf keinen Fall zum Beispiel um eine halbe Zelle versetzt werden. Es wird immer um die Anzahl von Zellen weitergesetzt, die der Wert der Geschwindigkeitsstufe angibt. Beträgt die Stufe beispielsweise 20 so wird diese Einheit in dem nächsten Zeittakt um 20 Zellen in der Fahrtrichtung vorangesetzt. Selbstverständlich gilt dieser Wert nur wenn die Geschwindigkeit innerhalb des Zeitschritts konstant bleibt.

Die Länge der Zellen hat einen definierten Wert und auch die Zeitschritte haben eine konstante Größe. Über die Geschwindigkeit ergibt sich ein direkter Zusammenhang zwischen Weg und Zeit (v = s * t), so dass auch die abbildbaren Geschwindigkeiten in Stufen angegeben werden müssen. Die drei Parameter zur Bestimmung der Stufen, Geschwindigkeit (v), Weg/Zellgröße (s) und Zeit (t) hängen voneinander ab, weshalb nur zwei von ihnen frei wählbar sind und der jeweils dritte sich ergibt.

Gewünscht ist für alle Parameter eine möglichst kleine und somit genaue Abstufung der Größen. Diese Abstufung darf nicht beliebig klein gewählt werden, da mit kürzer werdender Zelllänge die Anzahl der Zellen für einen Streckenabschnitt und damit auch die benötigte Rechenleistung und Speicherbedarf für die Simulation steigt.

In einen iterativen Prozess wurden folgende Abstufungen gefunden:

- Zelllängen: 2 m
- Geschwindigkeitsstufen: 0,125 m/s
- Zeittakt: 16 s

Die Geschwindigkeitsstufen von 0,125 m/s erlauben eine hinreichend genaue Abbildung der real gefahrenen Geschwindigkeiten. Der Maximale Fehler ist kleiner als 1/16 m/s. Bei einer angenommenen Fahrt von 100 km mit konstant 10 km/h (10 h Fahrzeit) ergäbe sich durch die Abbildung auf Geschwindigkeitsstufen ein ETA-Fehler (Estimated Time of Arrival) von nur ca. 6 min oder etwa 1%. Das Modell kann damit als sehr hochauflösend bezeichnet werden, da Längen auf 2 m genau und die Simulationszeit auf 16 s genau angegeben werden können.

Durch das Konzept der Zellularen Container ist selbst der Speicherbedarf bei einer Zelllänge von nur 2 m nicht unangemessen groß. Selbst auf heutigen Standardcomputern können sehr große Gebiete simuliert werden.

3.4.3 Übergang: Kante – Knoten

Sektionen (spezielle Kanten) und Aktionspunkte (spezielle Knoten) sind in der Regel stark unterschiedlich modelliert. Auf Sektionen wird mit dem Zellularautomatenkonzept operiert, wobei in Aktionspunkten sehr häufig Zustandsautomaten eingesetzt werden, um ihre Funktionalität nachzustellen (Abb. 3-9). Hieraus resultieren Probleme beim Übergang zwischen Kante und Knoten, weil unterschiedliche Systeme sich gegenüberstehen. Ein gleitender Übergang ist aber auch aus Simulationssicht aus zwei Gründen nicht erstrebenswert:

1. Es muss eine Bearbeitungsreihenfolge bei der Ermittlung des Systemzustands im nächsten Zeitschritt vorgegeben werden, um einfache Regeln verwenden zu können, die ohne Plausibilitäts- und Fehlerkontrollmechanismen auskommen. Beispielsweise kann ein Vehikel, das hinter einem vorausfahrenden fährt, durchaus im nächsten Zeitschritt an die Position des Vorausfahrenden gelangen, wenn das Vordere sich auch vorwärts bewegt. Wird das Hintere jedoch in der Bearbeitung dem Vorderen vorgezogen, so kommt es zu einer Konfliktsituation, da sich in einem Zwischenschritt zwei Vehikel an der selben Position befinden.

   Ist ein gleitender Übergang innerhalb eines zyklischen Wegenetzes vorhanden, kann kein Bearbeitungsanfang gefunden werden. In Zyklen (Ringen) muss eine Routine zur Ermittlung des nächsten Zeitschritts (Update-Routine) mindestens zwei Mal durchgeführt werden, um Konflikte aufzudecken und eventuell zu vermeiden. Im schlechtesten Fall muss die Update-Routine (V+1) Mal aufgerufen werden, wenn V die höchste Geschwindigkeitsstufe aller Schiffe in dem betreffenden Ring darstellt. Sind gekoppelte Zyklen vorhanden potenziert sich die Aufrufzahl. Diese hohe Ordnung eines Algorithmus zur Ermittlung eines Folgezeitpunktes ist nicht akzeptabel, zumal davon ausgegangen wurde, dass die Ringe erkannt und gesondert behandelt werden müssen.

2. Bei sehr großen Simulationssystemen werden häufig zur schnelleren Berechnung mehrere Prozessoren oder Computer eingesetzt. Die einzelnen Prozessoren führen unabhängige Berechnungen durch. Diese erfordern Programmteile die unabhängig von anderen berechnet werden können. Ist ein kontinuierlicher Übergang zwischen allen Systembereichen vorhanden, können keine unabhängigen Bereiche gefunden werden, da jeder Bereich mit einem anderen in Verbindung steht.

In dieser Arbeit wurden Kanten und Knoten voneinander entkoppelt. Hierdurch entstehen leichte Simulationsstörungen an den Übergängen, die jedoch mit natürlichen Ungleichmäßigkeiten verglichen werden können. Beispielsweise ergeben sich in Realität häufig kleine Störungen bei der Einfahrt eines Schiffes in einen Hafen. In der Simulation entstehen die Störungen auch nur bei Ein- und Ausfahrten in gesonderte Bereiche wie Häfen, Schleusen und Wasserstraßenkreuze. Die Übergänge werden durch sogenannte *Puffer* realisiert. Jeder Aktionspunkt besitzt Eingangs- und Ausgangspuffer. Diese Puffer entsprechen etwa den Ein- und Ausfahrtsbereichen von Aktionspunkten in der Realität. Die Behandlung der Puffer erfolgt nach dem FIFO-Prinzip (First In First Out), was einem Überholverbot in diesen Bereichen entspräche. Von den Kanten werden Vehikel, die die Kante verlassen in die zugehörigen Eingangspuffer der Aktionspunkte übergeben, sofern diese noch freien Platz für ein Vehikel haben. Die Knoten arbeiten nach dem Inbox/Outbox-Prinzip, das auch Mail-System genannt wird. Es werden die Vehikel aus den Eingangspuffern genommen und je nach Funktion des Knotens bearbeitet. Nach Bearbeitung können die Vehikel in Ausgangspuffer gesetzt werden, falls es der Funktionalität des Aktionspunktes entspricht. Nachdem eine Kante komplett bearbeitet wurde, werden vorhandene Vehikel von den Ausgangspuffern der Knoten auf die Kante gesetzt, falls dies möglich ist.

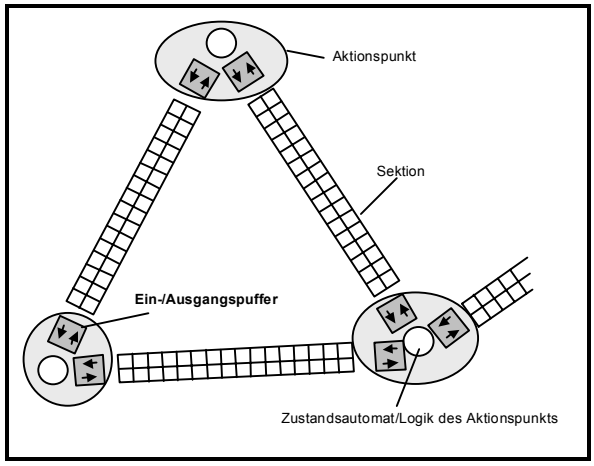

Abb. 3-9: Ein-/Ausgangspuffer-System

3.4.4 Zellulare Eigenschaften von Schiffen

Einheiten oder speziell Schiffe haben bestimmte Eigenschaften, die für die Verarbeitung mit zellularen Automaten von Bedeutung sind. Wie bereits besprochen, besitzen sie eine Größe, wodurch sie mehrere Zellen in der Länge belegen können. Neben allgemeinen Informationen über beispielsweise der Typ und die Bezeichnung der Einheit besitzen sie noch eine Koordinatenposition und Netzreferenzen, die ständig vom System ermittelt werden. Vehikel (bewegliche

Einheiten) haben zusätzlich Angaben zu deren Brems- bzw. Beschleunigungsverhalten, Sicherheitsabstände zu anderen Vehikeln und aktuelle Werte für die Momentangeschwindigkeit als auch für die gewünschte Fahrtgeschwindigkeit.

Die Wunschgeschwindigkeit wird vom System versucht zu erreichen und zu halten, sofern sie nicht über der maximal erlaubten Geschwindigkeit in dem befahrenen Bereich liegt. Die Wunschgeschwindigkeit kann auch online zur Simulationszeit angepasst werden, so dass direkt äußere Ereignisse in die Simulation einfließen können.

Beschleunigungen, Abbremsen und das Halten von einer Geschwindigkeit wird nicht durch Funktionalitäten der Vehikel bestimmt, sondern durch die Logik der Sektionen. Dies ist eine Verfolgung des Zellularautomatenkonzepts, das den Vorteil hat, dass auf unterschiedlichen Sektionen diverse Verhaltensregeln implementiert werden können, die Einheiten und davon abgeleitete Klassen jedoch unberührt bleiben. Vehikel (allgemeine Schiffe) stellen nur Inhalte für Zellen dar, die über keine Funktionalität hinsichtlich der Zellularautomaten-Simulation besitzen.

3.4.5 Update-Regeln

Das Simulationsnetz ist in ungekoppelte Aktionspunkte und Sektionen eingeteilt. Diese Elemente können unabhängig voneinander bearbeitet werden. In jedem Berechnungsschritt, der den Systemzustand der nächsten Zeitstufe ermittelt (*Update*), müssen alle Netzelemente behandelt werden. Damit zwei unterschiedliche Positionen im Netz mit gleichen Eigenschaften die gleichen zeitlichen und physikalischen Effekte innerhalb eines Updateschrittes besitzen, wird eine Updateordnung vorgeschlagen, die jedoch nicht zwingend notwendig ist:

1. neuen Zeitschritt ermitteln
2. alle Aktionspunkte updaten
3. alle Sektionen updaten

   ... weiter mit 1.

Wird diese Updatereihenfolge benutzt, ist das Verhalten an den Übergängen einheitlich und Störungen des Verkehrsflusses werden vermieden.

3.4.5.1 Aktionspunkt

In Aktionspunkten sollen komplexe Vorgänge sehr stark abstrahiert werden. Sie werden dazu als *Zustandsautomaten* abgebildet. Diese Automaten können in jedem Zeitschritt, was einem Update entspricht, ihren Zustand ändern. Je nach Zustand werden die Eingangs- bzw. Ausgangspuffer bedient. Zudem können sie noch interne Speicher und Funktionalitäten besitzen, in denen Vehikel bearbeitet werden. Es werden nur Zustandsübergänge behandelt, die in der Regel zeit- und vehikelabhängig sind. Diese Beschränkung erlaubt es, durch sehr kompakte Regeln komplexe Begebenheiten für den Simulationsablauf auf wesentliche Details zu reduzieren (Abb. 3-10).

Um das Prinzip zu verdeutlichen, sei exemplarisch ein Hafen als Aktionspunkt genannt. Das Hafengeschehen kann auf keinen Fall in allen Einzelheiten abgebildet werden, ohne enorme Rechenkapazitäten zu benötigen. Für eine Simulation von Schiffsankünften reicht häufig eine Angabe über Abfahrts- und Ankunftszeiten aus. In diesem Fall besteht nicht die Notwendigkeit beispielsweise einen Ladevorgang eines Schiffes in das Programm zu integrieren. Stattdessen kann ein Zustandsautomat nach Ablauf der Ladezeit das Schiff als beladen deklarieren und für die Ausfahrt freigeben.

Theoretisch können Aktionspunkte auch als Zellularautomatensysteme oder als Kombinationssysteme operieren. Da an diesen Punkten sehr viel mehr als die Fahrt von Fahrzeugen betrachtet werden muss, macht dieser Ansatz aus Implementierungs- und Rechenzeitgründen in den meisten Fällen keinen Sinn.

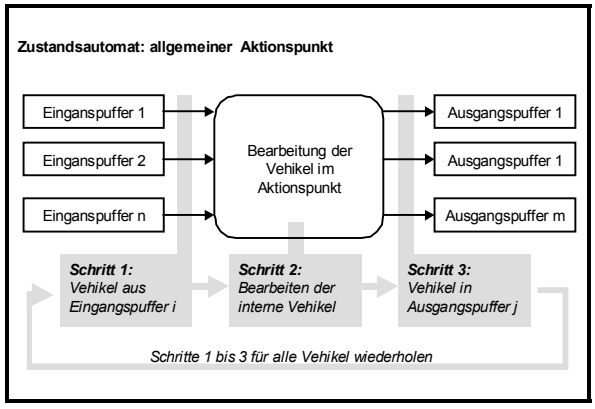

Abb. 3-10: Allgemeine Arbeitsweise der Zustandsautomaten in Aktionspunkten

Soll ein Vehikel von einer Sektion in einen Aktionspunkt gesetzt werden, wird automatisch geprüft, ob der zugehörige Eingangspuffer noch über Platz für das neue Vehikel verfügt. Jeder Aktionspunkt enthält eine Tabelle aller an ihn angeschlossenen Sektionen. Nur so können die korrekten korrespondierenden Puffer ermittelt werden (Abb. 3-11). Ist kein Platz in einem Eingangspuffer vorhanden, muss das Schiff in der Sektion verbleiben. Die Geschwindigkeit wird dabei den Gegebenheiten angepasst. Für die Eingangspuffer kann eine Einfahrtsgeschwindigkeit definiert werden. Dies findet beispielsweise bei Hafeneinfahrten Einsatz, wo Schiffe höchstens eine bestimmte Geschwindigkeit fahren dürfen. Auch in diesem Fall wird schon auf der Sektion gegebenenfalls die Geschwindigkeit angepasst.

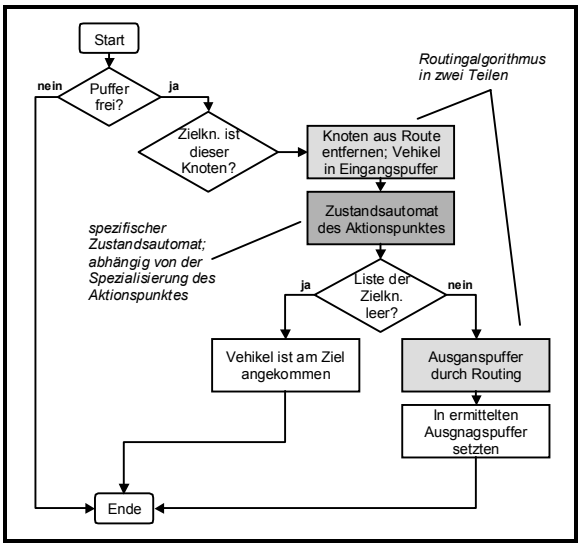

Abb. 3-11: Allgemeines Ablaufschema eines Aktionspunktes

Fährt eine Schiffseinheit in einen Aktionspunkt ein (der Eingangspuffer sei nicht voll), so wird sie von der Sektion entfernt und dem Knoten hinzugefügt. Sie wird auch dann komplett entfernt, wenn sie nur teilweise in den Aktionspunkt einfahren würde und in dem betreffenden Zeitschritt eine Anzahl von Zellen der Sektion vom hinteren Teil des Schiffes belegt blieben (eine Einheit kann mehrere Zellen in der Länge belegen). Die Position des Vehikels, die durch den vorderen Teil angegeben wird, ist in diesem Fall schon im Aktionspunktbereich, so dass diese Methodik keine Systemunstetigkeit darstellt.

Beim Setzen in den Eingangspuffer eines Aktionspunktes wird der erste Teil des Routingalgorithmus ausgeführt. Der erste Zielknoten in der Liste des Fahrtweges sollte der Aktionspunkt sein, in den eingefahren wurde. Dieser Zielknoten wird aus der Liste der zu fahrenden Knotenfolge entfernt, da das Vehikel in diesem Knoten nun angekommen ist. Sollte ein Fehler beim manuellen Setzten des Vehikels auf eine Sektion oder bei der manuellen Definition der Fahrtroute aufgetreten sein, wird dies vom Routingalgorithmus erkannt und dem Benutzer gemeldet.

Nach Bearbeitung der speziellen Zustandsautomatenroutinen wird der zweite Teil des Routingalgorithmus zur Bestimmung der Ausgangspuffer ausgeführt. Hierbei werden die Routingtabellen benutzt, um den Puffer zu bestimmen, der auf die Sektion führt, die mit dem nächsten Zielknoten verbunden ist. Auch in diesem Teil werden Fehler, die durch manuelle Eingaben entstehen können, erkannt und gemeldet.

Spezielle Aktionspunkte (wie Hafen, Schleuse, Kreuzung) implementieren hauptsächlich einen spezifischen Zustandsautomaten. Dieser Zustandsautomat kann sehr komplex sein, so dass Elemente in der Metaschicht gruppiert werden und einer feineren Spezialisierung zur Verfügung gestellt werden. Auf diese Weise müssen immer nur elementare Spezifika in einer Klasse hinzugefügt werden, da alle Grundfunktionalitäten von der nächsthöheren Schicht geerbt werden.

Aktionspunkt Hafen

Die Implementierung eines einfachen Hafens (Klasse: *HafenStandard*) besitzt neben den Ein- und Ausgangspuffern noch *Stellplätze* als interne Speicher für alle Schiffe, die sich im Hafen befinden. Auch Schiffe die be- oder entladen werden werden einem Stellplatz zugeordnet. Jeder Stellplatz besitzt eine *Ausfahrtzeit*, zu der das diesen Platz belegende Schiff den Hafen verlassen soll. Über diese Ausfahrtzeit können beispielsweise Ladevorgänge oder termingerechte Abfahrten nachgebildet werden. Ist die Ausfahrtzeit kleiner oder gleich der momentanen Simulationszeit, so verlässt das entsprechende Schiff den Zustandsautomaten und wird zur Ausfahrt in einen Ausgangspuffer gesetzt. In die Hafenaktivitäten kann jederzeit eingegriffen werden, um die Simulation an reale Angaben über beispielsweise Ausfahrtzeiten zu koppeln. Dazu stellt die Klasse HafenStandard Methoden zur Verfügung, die über den Zustand aller relevanten Hafeninformationen Auskunft geben.

Aktionspunkt Schleuse

Der Zustandsautomat der Schleuse (Klasse: *Schleuse*) soll anhand eines Ablaufdiagramms verdeutlicht werden (siehe Abbildung). Die Klasse Schleuse vereint die Eigenschaften:

- Die Schleusungszeit muss für jede Instanz dieser Klasse frei wählbar sein.
- Die Größe der Schleuse ist über Parameter zu setzten.
- Es werden bei Bedarf so viele Schiffe gleichzeitig geschleust, wie es die Größe der Schleusenkammer zulässt.
- Es soll nur dann geschleust werden, wenn Schiffe vor der Schleuse auf Einfahrt warten, d. h. es befinden sich Vehikel in mindestens einem der Eingangspuffer.
- Befinden sich Schiffe auf der geschlossenen Seite der Schleuse, auf der offenen Seite aber nicht, wird eine definierbare Zeit (0 möglich) gewartet, um die Anzahl der Leerschleusungen zu vermindern. Erst nach Ablauf dieser Zeitspanne wird leer geschleust.

Das in Abb. 3-12 dargestellte Ablaufdiagramm der Schleuse wird in jedem Updateschritt von „Start“ bis „Ende“ durchlaufen. Jeder Schritt enthält nach dem

Timestamp-Konzept die Simulationszeit. Das Ende des Schleusungsvorgangs wird ebenfalls durch einen Zeitstempel gekennzeichnet. Eine offene Schleuse ist gleichbedeutend mit einer Zeitangabe des Schleusungsendes, die kleiner ist als die momentane Simulationszeit.

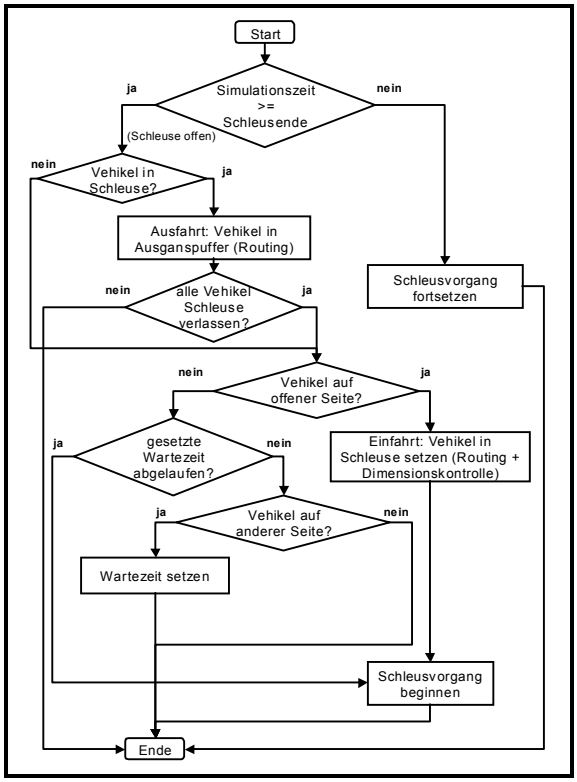

Abb. 3-12: Zustandsautomat der Schleuse

Ist die Schleuse offen, muss zunächst die Ausfahrt der eventuell in der Schleuse befindlichen Schiffe geregelt werden. Dieser Vorgang kann unter Umständen mehrere Zeitschritte (Updates) dauern, falls der zugehörige Ausgangspuffer aufgrund eines Staus nicht frei ist.

Nachdem alle Einheiten aus der Schleuse (hiermit sind die internen Speicher gemeint, nicht der Ausgangspuffer) entfernt sind, fahren an der offenen Seite befindliche Schiffe ein, bis die Schleuse voll ist, oder sich keine Schiffe mehr im Einganspuffer (Einfahrtbereich) befinden. Befinden sich keine Schiffe auf der offenen Seite, jedoch auf der anderen, wird eine gegebene Zeit gewartet, nach deren Ablauf eine Leerschleusung vorgenommen werden muss, wenn keine Schiffe auf der offenen Seite eingetroffen sind.

Der Schleusungsvorgang wird im Zustandsautomat durch Warten auf das Schleusungsende simuliert. Hierfür wird nur eine vernachlässigbar geringe Rechenkapazität benötigt. Wenn der Vorgang beendet ist, wird auf der geöffneten Seite die beschriebene Ausfahrt eingeleitet.

Aktionspunkt Kreuzung

Die Kreuzung (Klasse: *KreuzungStandard*) ist als parametrischer Router umgesetzt. Es werden hierbei keine internen Speicher zur Aufnahme von Vehikel benötigt. Die alleinige Aufgabe dieser Klasse ist es Funktionalitäten zur Verfügung zu stellen, durch die automatisch Vehikel von den Eingangspuffern in die richtigen Ausgangspuffer gesetzt werden, die über Sektionen mit den nächsten Zielknoten verbunden sind. Es wird ein adaptiver FIFO-Algorithmus (First In First Out) verwendet, um Vehikel an hinteren Positionen im Einganspuffer auf freie Ausgangspuffer setzen zu können, während weiter vorne platzierte aufgrund eines Staus nicht bedient werden können. Die Klasse KreuzungStandard ist so ausgelegt, dass sie mit einer beliebigen Anzahl von verbundenen Sektionen arbeitet.

3.4.5.2 Sektion

Eine Sektion nutzt die Klasse Abschnitt zur Erstellung eines Zellgitters über den geometrischen Verlauf. Das erstellte Zellgitter ist stetig über die gesamte Sektion, da eine automatische Umrechnung zwischen den Zellpositionen im einzelnen Abschnitt auf die in der Sektion erfolgt. Alle Zellen können durch Angabe der globalen Sektionsposition oder über einen Parameter im Bereich von [0;1] in der Länge angesprochen werden. In der Breite werden die Zellen über die Fahrspurnummer abhängig von der Fahrtrichtung ermittelt. In der Simulation wird an den meisten Stellen angenommen, dass ein Schiff bei freier Fahrt rechts fährt, auch wenn es sich in Realität in der Mitte oder am linken Rand der Fahrspur befindet. Für den Simulationsverlauf bei freier Fahrt spielt der Abstand des Schiffes zum Fahrwegrand keine Rolle, so dass diese Annahme keine Einschränkung darstellt. Die Klasse Sektion stellt elementare Funktionen für zellulare Automaten zur Verfügung:

- Einheiten können an gegebenen Positionen gesetzt und entfernt werden.
- Funktionen zur Umgebungsanalyse wie die Ermittlung freier Zellen um eine zu testende Zelle herum (auf einer gegebenen Spur mit Fahrtrichtung).
- Informationen über die Sektion mit ihren Abschnitten und allen enthaltenen Einheiten sind abrufbar.
- Spezielle Abstandsanalysen zur Projektion von gegebenen realen Schiffskoordinaten auf die Sektion: Diese *Mapping*-Funktionen werden zur Initialisierung des Systems mit einer Startkonfiguration benötigt, wenn die Eingangswerte aus realen Messungen stammen.

Sektion Kanal

Ein Kanal (Klasse: *Kanal*) ist ein spezieller Fahrweg (Klasse: *Fahrweg*, Metaschicht) mit zwei Fahrspuren, der mit einem Überholverbot belegt werden kann. Zwei Spuren bedeuten, dass maximal zwei Schiffe nebeneinander fahren oder sich begegnen können. Die Logikregeln des Kanals sind so ausgelegt, dass diese Restriktionen eingehalten werden und Konfliktfälle in jedem Fall mit einem naturähnlichen Verhalten vermieden werden. Diese Ereignisse treten in aller Regel nur bei Begegnungen mehrerer Schiffe auf. Die eingebauten Regeln erlauben den Vehikeln in der Simulation ein pseudo-intelligentes Verhalten. Beispielsweise reagieren die einzelnen Vehikel angemessen auch auf äußere unnatürliche Eingriffe wie das Setzten eines neuen Schiffes direkt vor ein fahrendes, das sich bereits in der Simulation befindet. Eine Reaktion kann je nach Situation ein Bremsmanöver und/oder Ausweichen auf eine andere Spur sein.

Pseudo-Intelligenz

Die *Pseudo-Intelligenz* ist bei allen Fahrwegen nötig, bei denen Begegnungsfahrt nicht ausgeschlossen werden kann. Die in den Grundlagen beschriebenen Regeln beziehen sich auf Straßen ohne Begegnungen und sind daher simpel im Vergleich zu den in dieser Arbeit verwendeten. Die Pseudo-Intelligenz basiert auf einem Algorithmus zur Wahl der höchstmöglichen Fahrtgeschwindigkeit auf der am weitesten rechts liegenden Spur unter Berücksichtigung von definierten Beschränkungen. Der Algorithmus kann vereinfachend mit folgenden Schritten beschrieben werden:

1. Ermittlung einer theoretisch bei freier Fahrt möglichen neuen Geschwindigkeitsstufe unter Berücksichtigung der Brems- und Beschleunigungseigenschaften dieses Vehikels und der Geschwindigkeitsbeschränkungen dieses Bereichs.
2. Ermittlung der theoretisch, im nächsten Zeitschritt zurückgelegten Fahrtstrecke unter Berücksichtigung eines eventuellen Beschleunigungs- bzw. Bremsvorgangs.
3. Ermittlung der maximal fahrbaren Geschwindigkeit für alle Spuren, wobei der Wertebereich bei $[-\infty; v_{max,\ theoretisch}]$ liegt. Kommt auf einer Spur ein Vehikel entgegen, so kann die auf dieser Spur fahrbare Geschwindigkeit im negativen Bereich liegen. Höhere Geschwindigkeiten als die ermittelte theoretische maximale Geschwindigkeit müssen nicht betrachtet werden.
   a) Für jede Spur wird die Anzahl der in Fahrtrichtung freien Zellen ermittelt, woraus sich der freie Weg berechnen lässt. Die Anzahl der freien Zellen wird durch das Ende der Sektion oder durch eine Einheit beschränkt.
   b) Befindet sich am Ende der freien Zellen ein Aktionspunkt, wird die Einfahrtsgeschwindigkeit des zugehörigen Eingangspuffers ermittelt. Ist die Anzahl der freien Zellen durch eine Einheit beschränkt, so wird diese Geschwindigkeit ermittelt, die ungleich null sein kann, falls es sich um ein Vehikel oder eine davon abgeleitete Instanz handelt.
   c) Mit der freien Strecke und der ermittelten Geschwindigkeit am Ende dieser Strecke wird iterativ eine Geschwindigkeit ermittelt, die im nächsten Zeitschritt gefahren werden kann, ohne dass das Vehikel unter Berücksichtigung seiner Bremsfähigkeiten und Sicherheitsabstände in eine Kollision verwickelt wird. Dies ist ein sehr aufwendiger Prozess, der nur iterativ lösbar ist, da die Ermittlung des Bremsweges im allgemeinen nicht direkt lösbar ist.
4. Von den für jede Spur ermittelten maximalen Fahrgeschwindigkeiten wird die am weitesten rechts liegende Spur (geringste Fahrspurnummer) unter folgenden Einschränkungen gewählt:
   - Ist eine weiter rechts liegende Spur als die aktuelle ermittelt worden, muss geprüft werden, ob durch die Länge des Vehikel beim Zurückwechseln überholte Einheiten beeinflusst werden. Ist dies der Fall kann der Spurwechsel nicht vollzogen werden und es ist die nächst linke Spur zu prüfen bzw. muss für diesen Zeitschritt auf der aktuellen Spur verblieben werden.
   - Sonderfall: Überholmanöver bei nahem und/oder schnellen Gegenverkehr. Ist die freie Fahrt durch eine Einheit behindert, kann auf die linke Spur gewechselt werden, um diese zu Überholen. Dieses Manöver wird jedoch nicht ausgeführt, wenn auf der linken Spur auch die Geschwindigkeit reduziert werden muss, obwohl die ermittelte Geschwindigkeit dort höher ist als die auf der aktuellen rechten Fahrspur. Im Fall eines Spurwechsels müsste das Vehikel aufgrund von Gegenverkehr, Geschwindigkeitsbeschränkungen oder einer Aktionspunkteinfahrt selbst auf der linken Spur die Geschwindigkeit reduzieren. Es darf kein Überholmanöver gestartet werden, weil es im nächsten Zeitschritt abgebrochen werden müsste, da die freien Abstände im folgenden Schritt noch kleiner würden. Es wird auf der aktuellen Spur abgebremst, bis ein Überho-

len möglich ist. Andernfalls muss die Geschwindigkeit der der vorderen Einheit angepasst werden.

Mit Hilfe dieser komplexen Regel werden auf einer Wasserstraße die Geschwindigkeit und die Fahrspur für den nächsten Zeitschritt ermittelt. Geschwindigkeit und Fahrspurwahl können nicht unabhängig voneinander ermittelt werden, so dass es einer Regel dieses Umfangs bedarf.

Es werden zur einfachen Simulation von Verkehrsvorgängen auf Kanälen nahezu keine zusätzlichen Regeln benötigt, da selbst Überholmanöver durch die integrierte Pseudo-Intelligenz abgewickelt werden können.

Die Pseudo-Intelligenz beinhaltet vorausschauendes Fahren nach einem Peilungsverfahren. Das bedeutet, dass nur direkt ermittelbare Objekte Einfluss auf die Fahrweise haben können. Soll beispielsweise ein Überholvorgang eingeleitet werden, könnte eine Überprüfung der Überholspur ergeben, dass ein entgegenkommendes Schiff mit sehr geringer Geschwindigkeit fährt. Ein hinter diesem Schiff fahrendes und auch zum Überholen ansetzendes Schiff mit hoher Geschwindigkeit wird nicht ermittelt. Es könnte in diesem Beispiel passieren, dass ein eingeleiteter Überholvorgang abgebrochen werden muss. Diesen Abbruchvorgang geben die Regeln zur Geschwindigkeits- und Fahrspurermittlung her. Bei genauer Analyse können sich unter Umständen sprunghafte Fahrspurwechsel der Schiffe ergeben, die jedoch keinen Einfluss auf das Gesamtergebnis der Simulation haben, da die Fahrspuren keinen physikalischen Positionen entsprechen, sondern nur logische Spuren sind. Dieses Beispiel macht deutlich, weshalb auf eine sehr aufwändige Gesamtsystemzustands-Analyse verzichtet werden kann, wenn Überholmanöver zugelassen sind. Es müssten bei andersartiger Implementierung unter anderem Kolonnenbildungen von langsam fahrenden Schiffen erkannt und behandelt werden, um die Durchführbarkeit eines Überholmanövers in jedem Fall vorab sicherstellen zu können. Dies erfordert einen sehr großen Rechenaufwand, der die Ausführung der Simulation unakzeptabel verlangsamen würde. Die beschriebene Pseudo-Intelligenz stellt in diesem Problemumfeld eine sehr gute Lösung dar.

Update-Routine

Es werden nur Vehikel und davon abgeleitete Klassen in einer Sektion bewegt. Einheiten sind nicht beweglich und werden in der Regel als Hindernisse im Fahrwasser behandelt. Ein Kanal arbeitet intern mit Richtungen, die nach außen nicht sichtbar sind. Für die Update-Routine ist die Richtung, in die ein Schiff fährt von Bedeutung, da bei einem Überholmanöver nicht von der Fahrspur auf die Fahrtrichtung geschlossen werden kann. Die Richtungen werden mit 0 (Hinrichtung) und 1 (Rückrichtung) bezeichnet. Jede Richtung besitzt zwei Spuren: rechts und links, die intern Spur 0 bzw. Spur 1 genannt werden. Ein Updatedurchlauf kann in zwei grobe Bearbeitungsschritte eingeteilt werden:

1. Bewegen aller Vehikel im Kanal und eventuelle Übergabe an einen Aktionspunkt, falls das betreffende Vehikel das Ende des Kanals in diesem Zeitschritt erreicht.
2. Einfügen von vorhandenen Vehikeln aus den Ausgangspuffern der angrenzenden Aktionspunkte.

Die beiden Bearbeitungsschritte werden in der angegebenen Reihenfolge ausgeführt, damit keine Konflikte beim Setzten der Vehikel entstehen. Im ersten Bearbeitungsschritt werden folgende Unterschritte ausgeführt:

- Ermittlung aller Einheiten, die in Richtung 0 (Hinrichtung) auf Spur 1 (linke Spur) fahren in der Reihenfolge entgegen ihrer Fahrtrichtung, also vom Ende des Kanals zum Anfang hin. Für die ermittelten Vehikel wird die neue Geschwindigkeit und Fahrspur nach den angegebenen Regeln ermittelt, und sie werden daraufhin weitergesetzt, wenn die Geschwindigkeit nicht konstant null beträgt.
- Ermittlung aller Einheiten der Richtung 0 (Hinrichtung) auf der rechten Spur (Spur 0) und Weitersetzten dieser.
- Analoges Vorgehen der obigen zwei Punkte für Richtung 1 (Rückrichtung), wobei die entgegengesetzte Fahrtrichtung vom Kanalanfang zum Ende hin verläuft.
- Einfügen von eventuell vorhandenen Einheiten am Kanalanfang und am Kanalende aus den zugehörigen Ausgangspuffern der angrenzenden Aktionspunkte. Kann ein Vehikel nicht gesetzt werden, weil es mit seiner aktuellen Fahrtgeschwindigkeit in diesem Zeittakt nicht komplett in den Kanal einfahren würde, so wird die Anzahl der Wartetakte für die Einfahrt der entsprechenden Ausgangspufferposition erhöht. Im folgenden Zeittakt wird die Schiffsgeschwindigkeit mit den Wartetakten multipliziert, woraus sich eine Position ergibt, die das Schiff nach der Anzahl von Takten erreicht hätte. Sobald das Schiff komplett gesetzt werden kann, wird es in den Kanal aus dem Ausgangspuffer über-

tragen. Weitere Schiffe können folgen, wenn genügend Bremsweg und Sicherheitsabstand vorhanden ist.

Ist nicht genügend Platz für die Einfahrt eines Schiffs vorhanden, da sich ein Stau auf dem Kanal direkt vor der Einfahrt gebildet hat, so wird die Anzahl der Wartetakte nicht erhöht, da das für die Einfahrt bestimmte Vehikel auch in Realität abbremsen muss und nicht die gewünschte Strecke zurücklegen kann.

Mit diesem Update-Schema lassen sich Szenarien realitätsnah simulieren. Der Übergang zwischen Sektion und Aktionspunkt wird allein durch das Puffersystem realisiert. Auf diese Weise sind die einzelnen Elemente voneinander entkoppelt berechenbar. Ein weiterer Vorteil ist die hierdurch erreichte Geschwindigkeit und Flexibilität für Erweiterungen, da durch einfache Operationen auf eine definierte Schnittstellen an den Übergängen zugegriffen werden kann.

4 Zusammenfassung

Das vorgestellte Modellierungskonzept lässt eine flexible objektorientierte Modellierung von Wasserstraßen zu, deren spezifische Eigenschaften sich durch den modularen Aufbau leicht an gegebene Bedürfnisse angepasst werden können. Durch die mikroskopische Simulation von identifizierbaren Wasserfahrzeugen können realitätsnahe Daten berechnet werden, die mit anderen Verfahren nicht ermittelt werden könne bzw. sich nur in aufwändigen Realmessungen erheben lassen.

Wird ein anhand realer Wasserstraßendaten und Schiffseigenschaften geeichtes System vorausgesetzt, bietet sich die Möglichkeit, das Wasserstraßengeschehen für zukünftige veränderte Bedingungen wie beispielsweise Fahrwasserverbreiterungen oder gemischten Verkehr zu simulieren. Die resultierenden Effekte lassen durch die mikroskopische Technik im Detail analysieren.

Die für Wasserstraßen angepasste Technik zur Simulation von Fahrzeugbewegungen stellt ein nützliches Analyseinstrument zur Verfügung, das bereits im Auto-Straßenverkehr angewandt wird.

5 Schrifttum

Bauknecht, K., Kohlas, J. & Zehnder, C. A.: Simulationstechnik - Entwurf und Simulation von Systemen auf digitalen Rechenautomaten. Hamburg, 1976

Chowdhury, D., Wolf, D. E. & Schreckenberg, M.: Particle hopping models for two-lane traffic with two kinds of vehicles - Effects of lane-changing rules. Physica, 1997

Cremer, M. & Ludwig, J.: A fast simulation model for traffic flow on the basis of boolean operations. Mathematics and Computers in Simulation, 1986

Damrath, R.: Objektorientierte Methoden - Vorlesungsskript Institut für Bauinformatik. Hannover, 1998

Eschen, R.: Objektorientierung - Wiederverwendbarkeit als Regel bei der Softwareentwicklung. www.oase-schareware.org, V 2.1, 1998

Esser, J.: Simulation von Stadtverkehr auf der Basis zellularer Automaten. Hamburg, 1997

Eversheim, W. & Kees, A.: Objektorientierte Analyse - Methoden zur Klassenfindung. Objekt Fokus 5/6, 1998

Helbing, D.: Verkehrsdynamik - Neue physikalische Modellierungskonzepte. Hamburg, 1997

Kerner, B. S., Konhäuser, P. & Schilke, M.: Deterministic spontaneous appearance of traffic jams in slightly inhomogenous traffic flow. Physical Review E 51, 1995

Koehler, R.: Verkehrsablauf auf Binnenwasserstraßen - Untersuchungen zur Leistungsfähigkeitsberechnung und Reisezeitverkürzung. Schriftenreiche des Instituts für Verkehrswesen, Universität Karlsruhe, 1968

Krauß, S., Wagner, P. & Gawron, C.: Continous limit of the Nagel Schreckenberg model. Physical Review E 54, 1996

Lighthill, M. J. & Whitham, G. B.: On kimematic waves II., A theory of traffic flow on long crowed roads. Highway Research Board, Special Report 79, 1964

Natke, H. G.: Strutural safety evaluation based on systems identification approaches. Hannover, 1998

Priestley, M.: Practical object-orientated design with UML. The McGrow-Hill Companies, London, 2000

Wiedemann, R.: Simulation des Straßenverkehrsflusses. Schriftenreihe des Instituts für Verkehrswesen, Universität Karlsruhe, Karlsruhe, 1974

Zimmermann, C., Hüsig, A. & Linke, T.: Schnelle Schiffe für Kombiverkehre und Containertransport: Auswirkungen auf Betrieb, Unterhaltung, Sicherheit und Ökologie von Binnenwasserstraßen. Bericht, Franzius-Institut, Hannover, 1998