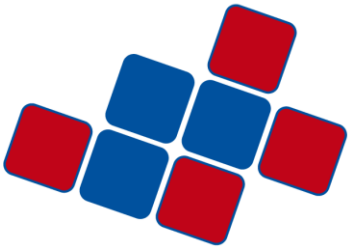


SYSTEMANALYSE UND -OPTIMIERUNG

Carl von Ossietzky Universität Oldenburg

ISS

Intelligente Schiffs Simulation

Abschlussdokumentation

Projektgruppe Intelligente Schiffssimulation

Betreuer:	Prof. Dr.-Ing. Axel Hahn Dipl. Inform. Sören Schweigert Dr.Ing. Volker Gollücke Dr.Ing. Christian Denker M.Sc. Matthias Steidel M.Sc. Leon Siegel
Gruppenmitglieder:	Hendrik Heeren Markus Höge Sören-Martin Köhler Julian Look Michel Mammes Mario Mühleck Marcel Obist Tjark Sauer Martin Steinbach Tilman Teusch Jannes Wemken
vorgelegt am:	05.04.2018

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AIS	Automatic Identification System
BSH	Bundesamt für Seeschifffahrt und Hydrographie
COLREG	Conventions on the International Regulations for Preventing Collisions at Sea
eMIR	eMaritime Integrated Reference Platform
EmsSchO	Schifffahrtsordnung Emsmündung
ENC	Electronic Navigational Chart
GML	Geography Markup Languague
HLA	High Level Architecture
IHO	International Hydrographic Organization
ISS	Intelligente Schiffssimulation
KVR	Kollisionsverhütungsregeln
MTS	Maritime Traffic Simulation

MTSS	Maritime Transportation System Simulation
POM	Project Object Model
SeeSch-StrO	Seeschifffahrtsstraßen-Ordnung

Tabelle 1: Abkürzungsverzeichnis

1 Einleitung

Seit einigen Jahren forscht die Carl von Ossietzky Universität in Zusammenarbeit mit dem OFFIS in Oldenburg an der Optimierung des Schiffsverkehrs und dessen angrenzenden Bereichen. Aktuell ist einer der Schwerpunkte des OFFIS die intelligente Schiffssimulation. Dabei handelt es sich um die Simulation von fahrerlosen Wasserfahrzeugen. Bekannt ist dieses Forschungsgebiet aus dem Straßenverkehr, bei dem es darum geht, Kraftfahrzeuge völlig autonom am Verkehr teilnehmen zu lassen.

1.1 Motivation

Vor der Einführung neuer Systeme sind ausgiebige Tests unabdingbar. Um einen großen Nutzen ohne Risiko und bei geringen Kosten zu erlangen, soll dafür zunächst eine computergestützte Simulation verwendet werden. Eine solche computergestützte Simulation eignet sich als Testumgebung für bestehende Systeme, wie beispielsweise MTCAS (Maritime Traffic Alert and Collision Avoidance System). MTCAS ist ein Alarmsystem zur Kollisionsverhütung, das ähnlich wie TCAS, welches an Bord von Flugzeugen eingesetzt wird, funktionieren soll. Damit die Funktionalität dieses Systems getestet werden kann, muss eine realitätsnahe Schiffsumgebung bereitgestellt werden.

Schiffssimulationen gibt es bereits auf dem Markt, um zum Beispiel zukünftige Schiffskapitäne auszubilden. Hier kann die entwickelte Simulationsumgebung nach Anpassung Anwendung finden, da die Testumgebung das Verhalten der anderen Schiffe realitätsgetreu simuliert.

1.2 Problembeschreibung

Es gibt bereits Ansätze, welche das Verhalten von Schiffen simulieren. Diese handeln jedoch strikt nach vorgegebenen Routen und reagieren beispielsweise nicht auf Routenüberschneidungen oder fehlerhaftes Verhalten anderer Schiffe. Des Weiteren werden in bestehenden Simulationen die Kollisionsverhütungsregeln (KVR) und weitere regional zulässige Schifffahrtsordnungen nicht berücksichtigt. Aktuelle Simulationsumgebungen können ohne intelligentes Verhalten und integrierte Schifffahrtsregeln nicht für Tests von Kollisionsverhütungssystemen, wie z.B. MTCAS, verwendet werden.

Derzeit ist für Schiffe ab 300 Tonnen die Ausstattung mit Kollisionswarnungssystemen verpflichtend. Diese Systeme agieren unterstützend und sollen dem Kapitän eine Hilfestellung geben, sobald auf der angestrebten Route eine Kollision entstehen könnte. Für die Beteiligten auf dem Schiff muss auch dieser Umgang mit den Systemen geübt sein.

1.3 Zielsetzung

Das Ziel dieser Projektgruppe ist es, die bestehende Simulationsumgebung MTS (Maritime Traffic Simulation) der Projektgruppe MTSS, um eine intelligente Komponente zur Steuerung von Schiffen zu erweitern. Darunter wird verstanden, dass die einzelnen Schiffe bzw. Agenten realitätsgetreu, das heißt wie echte Kapitäne, handeln. Dies soll dazu dienen, eine Testumgebung für andere Systeme, wie das zuvor genannte MTCAS, bereitzustellen. Weiterführend soll das System nicht als Routenfindung fungieren, sondern eine vorausschauende Planung zur Vermeidung von Kollisionen darstellen. Daran anknüpfend soll die Simulation mit Gefahrensituationen umgehen können. Als Beispiel kann hier die Reaktion auf eine drohende Kollision angeführt werden, in der ein kreuzendes Schiff kein Ausweichmanöver einleitet, wie es eigentlich in den KVR vorgegeben wäre. Zudem soll mithilfe der Trajektorienplanung die technische Ausführung zur Steuerung der Schiffe umgesetzt werden. Anknüpfend an die Ausnahmeregelungen soll es Schiffe bzw. Kapitäne in der Simulation geben, die ein unterschiedliches Verhalten an den Tag legen, damit auch unerfahrene Kapitäne oder beispielsweise rücksichtslose Kapitäne dargestellt werden können.

Da das MTCAS vorrangig im kommerziellen Schiffsverkehr zum Einsatz kommen wird, werden lediglich solche Schiffe betrachtet, die sich für den kommerziellen Einsatz anbieten, der Fokus liegt hier auf Frachtschiffen. Eine weitere Einschränkung ergibt sich aus dem Testgebiet. Als primäres Testgebiet dient die Deutsche Bucht, da dort ein reiches Portfolio unterschiedlichster Situationen simuliert werden kann sowie die rechtliche Betrachtung auf deutsche Gesetze, die die KVR ergänzen oder genauer ausführen, beschränkt werden kann. Primäres Ziel ist es dabei, dass realistischer Verkehr auf See simuliert wird. Standardvorgänge wie das Anlegen im Hafen oder Ankern sollen zunächst nicht betrachtet werden.

1.4 Grobe Lösungsidee

Als Grundlage der Architektur soll das Drei-Schichten-Modell dienen. Mithilfe dieses Modells soll eine autonome Schiffssimulation ermöglicht werden. Schiffe sollen gefährliche Situationen

selbst erkennen und entsprechend der geltenden Schifffahrtsregeln (KVR, SeeSchStrO, EmsSchO) auf Gefahrensituationen reagieren können und somit Kollisionen vermeiden können. Die Lösung bzw. die Simulation soll am Ende in verschiedensten Situationen auf See richtige bzw. menschenähnliche Entscheidungen treffen können, welche als Grundlage die regionalen Schifffahrtsregeln betrachtet. Innerhalb des Drei-Schichten-Modells soll die Simulation in Form eines Observers prüfen, ob sich Schiffe in der Nähe befinden und weitere Maßnahmen eingeleitet werden müssen. Neben dem regelbasierten Ansatz, welchem die Schifffahrtsregeln zugrunde liegen, soll unter Berücksichtigung historischer Daten, statistischer Auswertungen und Methoden des Data Minings aus realen Daten gelernt und somit das Verhalten in die Simulation übertragen werden.

2 Technologien

In den folgenden Absätzen wollen wir kurz die jeweiligen Technologien vorstellen, die wir während unserer Zeit innerhalb der Projektgruppe verwendet haben.

2.1 Atlassian Confluence

Confluence ist eine Content Collaboration-Software, die das Arbeiten im Team ermöglicht. Es gibt den Teammitgliedern die Möglichkeit, das Projekt an einem zentralen Ort voranzubringen und gemeinsam an diesem zu arbeiten. Inhalte können schnell angelegt werden und durch die verwendete Baumstruktur, werden die einzelnen Inhalte übersichtlich dargestellt. Auch das Umstrukturieren geht mittels Drag & Drop einfach von der Hand. Mittels der E-Mail-Funktion werden die Gruppenmitglieder stets darüber informiert, sobald Änderungen vorgenommen wurden. Zudem informiert das Übersichtsdashboard die Teammitglieder über die aktuellen Aktivitäten der übrigen Teilnehmer. Für unsere Zwecke besonders förderlich ist die Integration von Office-Dokumenten, die mittels der Funktion des Dokumentenimports simpel in das Confluence importiert werden können. Zudem bietet Confluence die Möglichkeit, weitere Multimedia-Dateien wie Videos oder Präsentationen zu importieren, sodass Präsentationen, denen ihr zugehörigen Word-Dateien, angehängt werden können. Unserer Gruppe diente das Confluence, um die Betreuer über unsere wöchentlichen Ausarbeitungen auf dem Laufenden zu halten. Zudem galt es der Gruppe als Grundlage, um Informationen zentral und übersichtlich darzustellen, sodass sich das Nachschauen nach Inhalten einfach gestaltete. Deshalb wurden Protokolle im Confluence hochgeladen, damit die Gruppenmitglieder ihre Aufgaben für die

kommende Woche hier nochmal nachschauen konnten und keine Aufgabe vernachlässigt wurde. Als zentraler Inhalt und Nachschlagewerk fungierte auch das im Confluence angelegte Kapitel der Seminararbeiten sowie die Projektziele und die Meilensteinplanung. Neben der User-Story-Planung fand im Confluence zudem unsere Urlaubsplanung und das Erstellen der Dokumentation statt. (Vgl. https:/de.atlassian.com/software/confluence)

2.2 Eclipse

Eclipse ist eine Open Source Community für eine leistungsstarke integrierte Entwicklungsumgebung, die Werkzeuge zur Softwareentwicklung erstellt, plattformunabhängig benutzt werden kann und sich der Entwicklung einer robusten Entwicklungsplattform verschrieben hat, die für den kommerziellen Gebrauch eingesetzt werden kann. Hierbei stellt Eclipse eine erweiterbare Plattform zur Entwicklung von Werkzeugen bereit und dient als Framework für integrierte Entwicklungsumgebungen. Eclipse ist erweiterbar durch Plug-ins. Das Grundgerüst des Projektes bildet die Eclipse Plattform, die eine erweiterbare und auf Java basierende DIE darstellt. Neben den Features, die Eclipse selbst zur Verfügung stellt, wird ein großer Wert auf Integration gelegt, was den Einsatz externer Tools vereinfacht. Während unserer Projektgruppenzeit kamen die beiden Eclipse-Versionen Neon und Oxygen zum Einsatz.

2.3 Cloud-Storage

Der IT-Dienst, der von der Universität Oldenburg allen Studenten zur Verfügung gestellt wird, ermöglicht es den Studenten, das lokale Verzeichnis auf das Datenhaltungssystem der Universität zu spiegeln. Nicht nur zwischen diesen beiden Verzeichnissen liegt eine Synchronität vor, sondern auch auf den unterschiedlichen Endgeräten des Nutzers. Das synchronisierte Verzeichnis kann von beliebig vielen ausgewählten weiteren Nutzern genutzt werden. Verwaltet wird das Cloud-Storage-Verzeichnis vom Universitätspersonal. Der Zugriff auf das Cloud-Storage ist von allen gängigen Plattformen möglich. Mithilfe des Sync-Clients wird es dem Nutzer ermöglicht, Teile seines Datenbestandes automatisch mit dem CloudStorage zu synchronisieren. Falls ein Nutzer aus Versehen eine Datei löscht, dann kann er diese selbstständig wiederherstellen. Innerhalb des CloudStorage ist ebenfalls das kollaborative Schreiben möglich, sodass Dokumente, die bereits in der Cloud gespeichert sind, mit dem integriertem ONLYOFFICE bearbeitet werden können. Dies erlaubt es dem Nutzer gleichzeitig mit anderen Nutzern an ein und demselben Dokument zu arbeiten. Innerhalb der Projektgruppe wurde das CloudStorage

benutzt, um dort sämtliche Dateien abzuspeichern. Somit hatten wir hiermit eine eigene Datenbank, die mittels einer Ordnerstruktur strukturiert wurde, sodass die Suche nach Dateien vereinfacht wurde und jede Teilgruppe in ihrer Ordnerstruktur arbeiten konnte und sich die weiteren Projektgruppenmitglieder in dem Ordner der Gruppe über den aktuellen Stand informieren konnten. (Vgl. https://www.uni-oldenburg.de/itdienste/services/datenhaltung/cloudstorage/)

2.4 Git

Git fällt unter den Bereich der Versionskontrollsysteme, die es ermöglichen, dass mehrere Nutzer zusammen an einer Datei arbeiten. Hierfür werden die Dateien auf einem zentralen Server verwaltet, dessen gesamte verwaltete Sammlung von Arbeitskopien als Repository bezeichnet wird. Jedes Teammitglied verbindet sich in der Folge mit dem Repository. Somit wird gewährleistet, dass auch in größeren Entwicklerteams der Überblick über die Verantwortung eingehalten wird. Das Versionskontrollsystem bietet eine Übersicht, welcher Entwickler zuletzt welche Änderungen in welcher Methode vorgenommen hat. Zudem ist jeder Entwickler dazu gezwungen, in jedem Änderungsvorgang mittels einer sogenannten Commit-Message zu beschreiben, welche Änderungen er vorgenommen hat. Damit die Verantwortungszuweisung erfolgen kann, meldet sich jeder Benutzer mit seiner persönlichen Benutzerkennung und Passwort an. Durch das Protokollieren und die Versionierung aller Änderungen ist es möglich, zu früheren Zuständen zurückzukehren. Jeder Benutzer hat ein eigenständiges und vollständiges Repository. Git besteht aus den drei Hauptbestandteilen Arbeitskopie, Index und Repository. So werden Änderungen in der Arbeitskopie mittels eines „adds“ zuerst in den Index übertragen, danach werden Änderungen im Index mit Hilfe eines „commits“ in das Repository eingepflegt. Das lokale Repository wird mittels SSH in ein Remote-Repository gepusht. Öffentliche Commits sollten hierbei niemals geändert werden, so sollten keine Commits mittels reset gelöscht werden, sondern mit Hilfe eines reverts zurückgesetzt werden. Eigene Änderungen sollten in ein Remote-Repository gepusht werden und sobald dies geschehen ist, sollten die anderen Entwickler darüber informiert werden und nachgeschaut werden, dass keine Konflikte entstehen. (Vgl. https://wr.informatik.uni-hamburg.de/_media/teaching/wintersemester_2010_2011/git-workshop-2010.pdf)

2.5 KNIME

KNIME ist eine Open-Source-Software, mit der lokale Dateien oder Datenbanken als Informationsquelle eingelesen werden können. Die hieraus entstandenen Informationen werden als Tabelle dargestellt und können in weiteren Schritten mittels sogenannter Knoten weiterverarbeitet werden. Hierbei können Filterungen, Löschungen oder Berechnungen vorgenommen werden. Mittels Ports werden die eingehenden und ausgehenden Datenströmungen der jeweiligen Knoten beachtet. Das Tool kommt insbesondere in der Datenanalyse zum Einsatz. Wichtig für unsere Zwecke war die Entwicklung von KNIME in der Java-Umgebung. Da KNIME für Eclipse ein Plugin bereitstellt, bietet sich Eclipse auch als Entwicklungsumgebung an. Die Bedienung in KNIME erfolgt via Drag-&-Drop, mit dessen Methode die einzelnen Knoten in das Bearbeitungsfenster verschoben werden. Durch die Abfolge der einzelnen Aktivitäten wird der sogenannte Workflow kreiert. Dieser Workflow entsteht, indem die einzelnen Knoten mittels Assoziationen bzw. Kanten verbunden werden. Im Node Repository sind alle Knoten aufgelistet, die in KNIME eingesetzt werden können. Hierbei wird die Suche mittels Gliederung in IO, Data Manipulation oder Statistics vereinfacht. Ebenso gibt es die Möglichkeit, die Knoten über einen direkten Filter anzuwählen. Die Knoten sind unterteilt in die jeweiligen Kategorien Einlese-, Umwandlungs-, Analyse-, Visualisierungs- und Ausgabeknoten. Die Einleseknoten definieren sich über das Einlesen und Integrieren von Daten, welche als Tabellen mit einer Kopfzeile und einer ID-Spalte ausgestattet sind. Umwandlungsknoten helfen dabei, Daten in andere Formate umzuwandeln oder aber Gruppierungen bzw. Filterungen vorzunehmen. Analyse- und Data Mining-Knoten unterstützen den User bei der Analyse. Mittels Visualisierungsknoten und Ausgabeknoten können Bilder oder Diagramme erstellt werden und was für unseren Zweck von gehobener Bedeutung ist, das Erstellen von Dateien wie CSV. Der WorkflowManager ist nicht nur für die Verbindung zwischen den einzelnen Knoten verantwortlich, er versieht die Knoten auch mit unterschiedlichen Ampelfarben. Ein rot gekennzeichneter Knoten steht hierbei dafür, dass dieser nicht konfiguriert ist, ein gelb gekennzeichneter für einen konfigurierten und ein grüner für einen ausgeführten, wobei eine grüne Kennzeichnung nur auftritt, wenn die Ausführung des Knotens erfolgreich war. Einer Ausführung eines Knotens bedarf einer erfolgreichen Ausführung der vorangestellten Knoten.

Innerhalb unserer Projektgruppe fand KNIME in der Evaluation Anwendung. Wir benutzten das Tool dafür, um aus einer Menge an AIS-Daten Schiffstracks herauszufiltern, die sich für

unsere Evaluation als geeignet herausstellen sollten. Hierbei kamen Filter zum Einsatz, die unter anderem fehlerhafte Daten herausfiltern sollten, aber auch um das Hauptaugenmerk auf Schiffe zu legen, die von ihrer Konstruktion den Schiffen aus unserer Simulation ähnlich sind. Um einzelne Tracks herauszufiltern, erstellten wir Java-Snippets, die aus allen AIS-Daten eines Schiffes nur einen bestimmten Zeitraum betrachteten, sodass wir nur den Track zwischen zwei Häfen angezeigt bekommen haben. Mittels eines CSV Writers gelang es uns, die AIS-Daten zu exportieren und so die Grundlage für den Vergleich mit unseren Agenten in HAGGIS zu schaffen.

2.6 S-100

S-100 ist das Dokument, das erklärt, wie IHO benutzt wird und erweitert die internationale ISO 1900 Serie um geographische Standards, zu denen hydrographische und maritime Ausgaben zählen. Hierbei erweitert der S-100 Standard den Umfang des S-57 Standards. So ist der S-100 Standard flexibler und ermöglicht u.a. auch die Verwendung von bildlichen- und Rasterdatentypen, sowie von erweiterten Metadaten und mehreren Kodierungsformaten. Der S-100 basiert auf der ISO 19100 Serie als geographischer Standard, sodass die S-100 basierten Daten kompatibel mit denen des ISO-Standards sind. (Vgl. https://www.iho.int/iho_pubs/standard/S-100/S-100_Info.htm)

3 Anforderungen

Im Folgenden werden die Anforderungen aufgeführt. Neben nicht funktionalen Anforderungen wird hierzu auf die Ziele und die erstellten User-Stories eingegangen.

3.1 Nicht funktionale Anforderungen

Für die Software wurden zudem folgende nicht funktionale Anforderungen erhoben: Erweiterbarkeit, Wartbarkeit, Funktionalität, Benutzbarkeit, Übertragbarkeit und Zuverlässigkeit.

3.1.1 Erweiterbarkeit

Die Software soll eine Erweiterung um folgende Punkte ermöglichen:

- weitere Seegebiete neben der deutschen Bucht
- verschiedene Schiffstypen
- weitere Verkehrsteilnehmer

- Konfiguration von Fahrverhalten eines Agenten

3.1.2 Wartbarkeit

- Die Dokumentation muss sowohl für Entwickler als auch für Anwender verständlich sein
- Es muss möglich sein, die Software anhand verschiedener Szenarien zu testen

3.1.3 Funktionalität

Das zugrundeliegende Datenmodell muss S-57/S-63/S-100 konform sein, damit es in die bestehenden Forschungsprojekte MTS, LABSKAUS und HAGGIS integrierbar ist.

3.1.4 Benutzbarkeit

Alle Anwenderfunktionen müssen von der Benutzeroberfläche gesteuert werden können. Dies gilt auch für Einstellungsmöglichkeiten. Dazu wird die bestehende Benutzeroberfläche der eMIR-Software erweitert.

3.1.5 Übertragbarkeit

Die Software funktioniert nur im Zusammenhang mit den eMIR-Projekten des OFFIS unter Java in der Version 1.8. Durch den Einsatz von Java ist die Software plattformunabhängig.

3.1.6 Zuverlässigkeit

Die Software muss eine geringe Fehlertoleranz aufweisen. Tonnen können zum Beispiel nicht überfahren werden. Außerdem soll die Software zuverlässig alle COLREGs berücksichtigen und umsetzen. Die Software muss auch bei mehreren intelligenten Schiffsagenten lauffähig bleiben.

3.2 Ziele

Im Rahmen der Planungsphase wurden Ziele für die Software erarbeitet. Diese werden in Projektrahmenziele und Projektergebnisziele differenziert. Das Hauptziel ist die Entwicklung eines realitätsgetreuen, nicht-deterministischen (intelligente) Agenten in einer maritimen Simulationsumgebung.

3.2.1 Projektrahmenziele

R1	Es dürfen keine Funktionalitäten entwickelt werden, die bereits von HAGGIS bereitgestellt werden. Bestehende Komponenten müssen verwendet und ggf. angepasst werden.
R2	Die MTS muss als grundlegende Basis der maritimen Simulationsumgebung genutzt werden.

R3	Jedes Schiff muss ein Agent sein, dessen Verhalten durch Steuerungskomponenten umgesetzt wird.
R4	Die MTS muss um Komponenten erweitert werden, die das Verhalten abbilden.
R5	Die Deutsche Bucht muss als Anwendungsgebiet für die MTS verwendet werden.
R6	Anker- und Anlegevorgänge sollen nicht betrachtet werden.
R7	Das Verhalten muss so gestaltet werden, dass Umwelteinflüsse, wie Strömung, Wind und Änderungen am Dynamikmodell des Schiffes in die MTS integriert werden können, und die entwickelten Verhalten damit weiterhin funktionieren.
R8	Sämtliches Verhalten der Agenten soll nicht-deterministisch erfolgen.
R9	Die entwickelte Software muss bis zum 05.04.2018 geliefert werden.
R10	Die Softwaredokumentation muss bis zum 05.04.2018 geliefert werden.
R11	Der PG-Abschlussbericht muss bis zum 05.04.2018 geliefert werden.
R12	Die Abschlusspräsentation muss bis zum 05.04.2018 stattgefunden haben.
R13	Ein erster Prototyp muss bis zum 01.11.2017 entwickelt und präsentiert werden.
R13.1	Schiffe müssen Routen verfolgen.
R13.1.1	Die Route kann durch Hindernisse führen.
R13.2	Schiffe müssen dynamischen Hindernissen aus Seekarten (z.B. Bojen, Windparks, …) ausweichen/vermeiden können.
R13.3	Schiffe müssen dynamischen Hindernissen ausweichen/vermeiden können.
R13.3.1	Beim Ausweichen von dynamischen Hindernissen können hier bereits KVR-konforme Manöver ausgeführt werden.
R14	Es müssen überprüfbare Meilensteine erstellt werden.
R14.1	Jeweils zum nächsten Meilenstein müssen die umzusetzenden Anforderungen und Stories definiert sein.
R14.1.1	Die Anforderungen und Stories müssen von den Betreuern bestätigt werden.
R14.2	Die Meilensteine müssen von den Betreuern bestätigt werden.

Tabelle 2:Projektrahmenziele

3.2.2 Projektergebnisziele

E1	In der Simulation muss, durch mehrere Agenten in einem bestimmten Seegebiet, Verkehr nachgebildet werden.
E1.1	Der Gesamtverkehr muss auf Realitätstreue geprüft werden.
E1.2	Das Verhalten eines einzelnen Agenten muss auf Realitätstreue geprüft werden.

E2	Ein Agent muss Input über die jeweilige Schiffssensorik erhalten (Ausschluss des „God mode" - dem Agenten stehen nicht alle Informationen der gesamten Simulation zur Verfügung).
E3	Ein Agent muss mindestens ein Standardmodell für dessen Verhalten haben.
E4	Das Befahren von Routen mittels Bahnen muss durch das Verhalten beeinflusst werden.
E5	Das Verhalten muss durch Toleranzwerte in der MTS (HAGGISControl) individualisierbar sein.
E5.1	Passierabstände müssen variieren können.
E5.2	Der Zeitpunkt der Manövereinleitung muss variieren können.
E5.3	Die Geschwindigkeitstypen der Agenten müssen variieren können.
E5.4	Die Beschleunigungstypen der Agenten müssen variieren können.
E5.5	Die Art und Weise der Verfolgung von Routenkanten muss variieren können.
E5.6	Die Befolgung von KVR-Regeln muss variieren können.
E5.7	Allen variierenden Parametern soll jeweils ein Fehlermodell unterliegen.
E6	Das Verhalten eines Agenten muss einem Fehlermodell unterliegen.
E7	In der MTS muss ein Agent statischen Hindernissen ausweichen können.
E7.1	Ein Agent muss NoGo-Areas ausweichen können. (Nutzung des NoGo-Solvers)
E7.2	Ein Agent muss Seekartenobjekten ausweichen können.
E8	In der MTS muss eine Begegnungsbehandlung für dynamische Objekte stattfinden.
E8.1	In der MTS müssen Agenten entsprechend der KVR auf andere kreuzende Schiffe reagieren können und selbst andere Schiffe KVR-konform kreuzen können.
E8.2	In der MTS müssen Agenten entsprechend der KVR Überholmanöver durchführen und darauf reagieren können.
E8.3	In der MTS müssen Agenten entsprechend der KVR Abstand zu anderen Schiffen halten können.
E9	Der Agent muss ortsabhängig Regeln befolgen.
E9.1	Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.
E10	Es muss eine Startkonfiguration aller Schiffe im Verkehrsgebiet erzeugt werden.
E10.1	Jeder Agent muss sich eine Route aus dem Routengraphen generieren können.
E10.1.1	Es muss ein einfacher Routengraph genutzt werden, der in HAGGIS vorhanden ist.
E10.1.2	Es kann eine Routenvalidierung stattfinden, aber nur sofern dies für die Einhaltung des Gesamtziels - der Realitätstreue - notwendig ist.

Tabelle 3: Projektergebnisziele

3.3 User Stories

Für jeden Meilenstein wurde in der Planungsphase eine User-Story erstellt. Die einzelnen User-Stories werden folgend aufgeführt.

3.3.1 User Story 1: Benötigte Objekte können aus Seekarten ausgelesen werden

Meilenstein: Auslesen von Objekten aus Seekarten

Nummer	1
Name	Identifizierung der Features und Überführung in Objekte
Akteure	Initiiert von System.
Beschreibung	Küstenlinien Tiefgang Sperrgebiete usw.
Ergebnisziele	
Anfangsbedingung	
Abschlussbedingung	Features sind als Java-Objekte verfügbar.

Tabelle 4: User-Story 1

Nummer	1.1
Name	GML-Dateien vom Chart-Server ziehen
Akteure	Initiiert von Userstory 1.
Beschreibung	GML-Datei wird vom Chart-Server geladen und lokal gespeichert. Durch eine Bounding Box wird angegeben, welcher Bereich extrahiert werden soll.
Ergebnisziele	
Anfangsbe-dingung	Es besteht eine dauerhafte Verbindung zum Chart-Server.
Abschlussbe-dingung	GML-Dateien liegen lokal vor.

Tabelle 5: User-Story 1.1

Nummer	1.2
Name	GML-Dateien liegen lokal vor
Akteure	Initiiert von Userstory 1.1.
Beschreibung	Das System filtert die erforderlichen Daten aus den GML-Dateien und überführt diese mittels des FileFeatureProviders in Java-Objekte. Erforderliche Daten sind z.B.: Buoy Cardinal Installation Isolated danger Lateral Safe water Special purpose/general Traffic separation Line Scheme boundary Scheme crossing Scheme lane part zone signal station, traffic signal station, warning Recommended traffic lane part Inshore traffic zone Free port area Fairway Canal
Ergebnisziele	
Anfangsbedingung	Die GML-Datei liegt lokal vor und die Simulation ist gestartet.
Abschlussbedingung	Die oben genannten GML-Daten wurden in Java Objekte überführt.

Tabelle 6: User-Story 1.1.1

Nummer	1.2.1
Name	Lesezugriff auf GML-Datei herstellen
Akteure	Initiiert von Userstory 1.2.

Beschreibung	GML-Dateien können geöffnet und deren Inhalt mittels des FileFeatureProviders gelesen werden.
Ergebnisziele	
Anfangsbedingung	GML-Dateien sind lokal vorhanden.
Abschlussbedingung	Daten können aus GML-Dateien gelesen werden.

Tabelle 7: User-Story 1.2.1

Nummer	1.3
Name	Ablage der Features in Form von Java Objekten in einer Datenstruktur
Akteure	Initiiert von Userstory 1.
Beschreibung	Java-Objekte müssen in einer performanten Datenstruktur abgelegt werden, die auch größere Datenmengen gut skaliert, das heißt es soll eine logarithmische Laufzeitkomplexität statt einer linearen erreicht werden. Zusätzlich soll es möglich sein, auf der Datenstruktur eine Suche innerhalb von <= 5ms zu vollziehen. Zudem muss die Datenstruktur eine effiziente Filterung nach Feature-Kategorien ermöglichen. Das Vorgehen in deshalb notwendig, damit zur Laufzeit eine schnelle Verarbeitung von Seekarten-Informationen stattfinden kann.
Ergebnisziele	
Anfangsbedingung	Userstory 1.2 ist abgeschlossen.
Abschlussbedingung	Aktueller Datenstand, über die in Userstory 1.2 genannten Features.

Tabelle 8: User-Story 1.3

3.3.2 User Story 2: Auswahl eines Agenten in einem Szenario

Meilenstein: Agentendesign erstellen

Nummer	2
Name	Einem Agenten kann ein Szenario hinzugefügt werden
Akteure	Nutzer der Simulation.

Beschreibung	In der Oberfläche von HAGGIS Control kann ein Agent erstellt und einem Szenario hinzugefügt werden. Dessen Parameter bezüglich seines Fahrverhaltens können über Schieberegler verändert und spezifiziert werden.
Ergebnisziele	
Anfangsbedingung	HAGGIS Control wurde gestartet und ein Szenario erstellt.
Abschlussbedingung	Userstory 2.1 und 2.2 sind erfüllt.

Tabelle 9: User-Story 2

Nummer	2.1
Name	Agentendesign
Akteure	Initiiert von Userstory 2.
Beschreibung	Es wird ein Modell entwickelt, das die Architektur eines Agenten zur Umsetzung seines Gesamtverhaltens abbildet. Das Gesamtverhalten setzt sich dabei aus mehreren Teilverhalten zusammen. Ein Agent unterscheidet sich von einem Schiff (Vessel) durch mehrere Parameter bezüglich seines Fahrverhaltens.
Ergebnisziele	
Anfangsbedingung	Ein Agent baut auf der Grundlage eines Vessels auf.
Abschlussbedingung	Es existiert ein Modell, auf dessen Grundlage die Implementierung eines Agenten stattfindet.

Tabelle 10: User-Story 2.1

Nummer	2.2
Name	Anpassung der grafischen Oberfläche in HAGGIS Control bei der Auswahl eines Agenten in einem Szenario
Akteure	Initiiert von Userstory 2.
Beschreibung	Bei der Auswahl eines Agenten in einem Szenario kann zwischen einem Standard- und einem Individualverhalten unterschieden werden. Das Individualverhalten unterscheidet sich vom Standardverhalten insofern, dass feste

	Stereotypen von Kapitänen ausgewählt werden können, deren Fahrverhalten durch Festlegung ihrer Parameter vorbestimmt ist. Bei Auswahl des Standardverhaltens hingegen, können diese Parameter über Schieberegler durch den Nutzer verändert und angepasst werden. Zusätzlich ist es möglich, die Sichtweite eines Agenten in der Oberfläche zu beschränken oder zu erweitern. Die Sichtweite beschreibt quasi das Auge des Kapitäns, das je nach Beschränkung Informationen über Seekartendaten in einem angegebenen Bereich erhält.
Ergebnisziele	
Anfangsbedingung	Generische Anpassbarkeit der Oberfläche über Attribute in HAGGIS Control.
Abschlussbedingung	Bei Auswahl eines Agenten in einem Szenario in HAGGIS Control kann zwischen Standard- und Individualverhalten gewählt werden und es existieren entsprechende Schieberegler.

Tabelle 11: User-Story 2.2

3.3.3 User Story 3: Routenkanten verfolgen durch Agenten

Meilenstein: Routenkanten verfolgen durch Agenten

Nummer	3
Name	Routenkanten verfolgen durch Agenten
Akteure	Nutzer der Simulation.
Beschreibung	Ein Agent muss sich eine Route generieren können. Diese Route soll er selbstständig abfahren können. Das Befahren von Routenkanten/Bahnen durch den Agenten kann durch die Manipulation einiger Parameter, die auf Basis eines Standardmodells dargestellt werden, variieren.
Ergebnisziele	E4: Das Befahren von Routen mittels Bahnen muss durch das Verhalten beeinflusst werden E5.2 Der Zeitpunkt der Manövereinleitung muss variieren können E5.3 Die Geschwindigkeiten der Agenten müssen variieren können E5.4 Die Beschleunigungstypen der Agenten müssen variieren können. E5.5 Die Art und Weise der Verfolgung von Routenkanten muss variieren können. E10.1 Jeder Agent muss sich eine Route aus dem Routengraphen generieren können E10.1.1 Es muss ein einfacher Routengraph genutzt werden, der in HAGGIS vorhanden ist
Anfangsbedingung	Es muss einen Start- und Zielpunkt geben; Es muss ein Schiff vorhanden sein.

Abschlussbedingung	In der Simulation ist zu sehen, dass zwei Agenten des gleichen Schifftyps auf einer Testroute, durch unser entwickeltes Verhalten mit unterschiedlich hohen Geschwindigkeiten auf den Bahnen unterwegs sind und Kurven in unterschiedlich großen Winkeln abfahren werden. Zudem ist in der Simulation erkennbar, dass die Agenten unterschiedlich schnell beschleunigen und bremsen können.

Tabelle 12: User-Story 3

Nummer	3.1
Name	Integration der Agentenarchitektur in das bestehende System.
Akteure	Projektgruppe.
Beschreibung	Die Implementierung unseres Standardverhaltens muss in die bestehende Architektur von HAGGIS eingepflegt werden. Dazu muss eine Einarbeitung in das bestehende System erfolgen.
Ergebnisziele	R2 Die MTS muss als grundlegende Basis der maritimen Simulationsumgebung genutzt werden.
Anfangsbedingung	Es gibt eine Agentenarchitektur, die in ein bestehendes System integriert werden kann.
Abschlussbedingung	Es kann eine Instanz unseres Standardverhaltens einem Schiff in der Simulation zugewiesen werden und die Simulation kann gestartet werden. In der GUI werden Parameter angezeigt, die die Basis des Standardverhaltens darstellen.

Tabelle 13: User-Story 3.1

Nummer	3.2
Name	Parametrisierung des Fahrverhaltens beim Abfahren von Routenkanten/Bahnen durch einen Agenten
Akteure	Nutzer der Simulation.
Beschreibung	Das Befahren von Routenkanten/Bahnen durch den Agenten kann durch die Manipulation einiger Parameter, die auf Basis eines Standardmodells dargestellt werden, variieren.
Ergebnisziele	E4: Das Befahren von Routen mittels Bahnen muss durch das Verhalten beeinflusst werden E5.2 Der Zeitpunkt der Manövereinleitung muss variieren können E5.3 Die Geschwindigkeiten der Agenten müssen variieren können E5.4 Die Beschleunigungstypen der Agenten müssen variieren können.

	E5.5 Die Art und Weise der Verfolgung von Routenkanten muss variieren können
Anfangs-bedingung	Es gibt Parameter, die das Fahrverhalten beeinflussen können; Es gibt Routenkanten/Bahnen; Es gibt einen Agenten, dem ein Verhalten hinzugefügt werden kann.
Abschluss-bedingung	Der Nutzer der Simulation kann in der GUI von HAGGIS Control ein Schiff auswählen, diesem ein Verhalten zuweisen und die Parameter zur Individualisierung des Fahrverhaltens einstellen. Der Agent fährt Routenkanten/Bahnen entsprechend der gesetzten Parameter ab.

Tabelle 14: User-Story 3.2

Nummer	3.2.1
Name	Kursanpassungen erfolgen zu variablen Zeitpunkten
Akteure	Agent.
Beschreibung	Der Agent leitet abhängig von seiner gesetzten Parametrisierung ein Manöver ein.
Ergebnisziele	E5.2 Der Zeitpunkt der Manövereinleitung muss variieren können.
Anfangsbedingung	Es gibt einen Parameter für den Zeitpunkt der Manövereinleitung, der das Fahrverhalten beeinflusst; Es gibt Routenkanten/Bahnen; Es gibt einen Agenten, dem ein Verhalten hinzugefügt werden kann.
Abschlussbedingung	Zwei Agenten des gleichen Schifftyps werden beim Abfahren der gleichen Route Kursanpassungen, abhängig von ihrer Parametrisierung, zu einem unterschiedlichen Zeitpunkt, vornehmen.

Tabelle 15: User-Story 3.2.1

Nummer	3.2.2
Name	Geschwindigkeitsregulierung beim Abfahren von Routenkanten/Bahnen
Akteure	Agent.
Beschreibung	Der Agent passt, abhängig seiner gesetzten Parameter, die Geschwindigkeit an den zu befahrenden Streckenabschnitt an.
Ergebnisziele	E5.3 Die Geschwindigkeiten der Agenten müssen variieren können E5.4 Die Beschleunigungstypen der Agenten müssen variieren können.

Anfangsbedingung	Es gibt einen Parameter, der die Geschwindigkeit und die Beschleunigung des Agenten regelt und so das Fahrverhalten beeinflusst; Es gibt Routenkanten/Bahnen; Es gibt einen Agenten, dem ein Verhalten hinzugefügt werden kann.
Abschlussbedingung	Zwei Agenten des gleichen Schifftyps werden beim Abfahren der gleichen Route, abhängig von ihrer Parametrisierung, mit unterschiedlichen Geschwindigkeiten unterwegs sein.

Tabelle 16: User-Story 3.2.2

Nummer	3.2.3
Name	Erkennung und Korrektur von Kursabweichungen
Akteure	Agent.
Beschreibung	Der Agent nimmt abhängig von seiner Parametrisierung Kursanpassungen beim Verlassen seiner Route vor.
Ergebnisziele	Der Agent nimmt abhängig von seiner Parametrisierung Kursanpassungen beim Verlassen seiner Route vor.
Anfangsbedingung	Es gibt einen Parameter, der das Verhalten des Agenten beim Verlassen seiner Route regelt. Der Agent muss von seiner Route abgewichen sein; Es gibt Routenkanten/Bahnen; Es gibt einen Agenten, dem ein Verhalten hinzugefügt werden kann.
Abschlussbedingung	Zwei Agenten des gleichen Schifftyps werden beim Abfahren der gleichen Route, abhängig von ihrer Parametrisierung, unterschiedliche Korrekturen bei Kursabweichungen vornehmen.

Tabelle 17: User-Story 3.2.3

Nummer	3.2.4
Name	Beeinflussung des Verhaltens beim Befahren von Kurven
Akteure	Agent.
Beschreibung	Der Kurvenradius beim Befahren einer Kurve durch den Agenten ist von seiner Parametrisierung abhängig.
Ergebnisziele	E4: Das Befahren von Routen mittels Bahnen muss durch das Verhalten beeinflusst werden
Anfangsbedingung	Es gibt Routenkanten/Bahnen und Kurven auf der vorgegebenen Route; Es gibt einen Agenten, dem ein Verhalten hinzugefügt werden kann.

Abschluss-bedingung	Zwei Agenten des gleichen Schifftyps werden beim Abfahren der gleichen Route, abhängig von ihrer Parametrisierung, in der gleichen Kurve unterschiedlich große Bögen fahren.

Tabelle 18: User-Story 3.2.4

Nummer	3.3
Name	Routengenerierung
Akteure	Agent.
Beschreibung	Aus dem Routengraphen von HAGGIS Control generiert sich der Agent unter Angabe von Koordinaten eine Route. Dazu ist eine Einarbeitung in das bestehende System notwendig.
Ergebnisziele	E10.1 Jeder Agent muss sich eine Route aus dem Routengraphen generieren können E10.1.1 Es muss ein einfacher Routengraph genutzt werden, der in HAGGIS vorhanden ist.
Anfangsbedingung	Es gibt einen Routengraphen.
Abschluss-bedingung	Dem Agenten werden in der GUI von HAGGIS Control zwei Koordinaten übergeben und die generierte Route wird angezeigt.

Tabelle 19: User-Story 3.3

3.3.4 User Story 4: Statische Hindernisse werden erkannt und automatisch umfahren

Meilenstein: Statische Hindernisse werden erkannt und automatisch umfahren

Nummer	4
Name	Statische Hindernisse werden erkannt und automatisch umfahren.
Akteure	Agent.
Beschreibung	Ein Agent muss statische Hindernisse erkennen und diesen je nach Parametrisierung, die sich am Verhalten des Agenten orientiert, ausweichen können.
Ergebnisziele	E 5.2: Der Zeitpunkt der Manövereinleitung muss variieren können. E 5.3: Die Geschwindigkeitstypen der Agenten müssen variieren können. E 5.4: Die Beschleunigungstypen der Agenten müssen variieren können. E 7: In der MTS muss ein Agent statischen Hindernissen ausweichen können. E 7.1: Ein Agent muss NoGo-Areas ausweichen können. (Nutzung des

	NoGo-Solvers) E7.2: Ein Agent muss Seekartenobjekten ausweichen können.
Anfangsbedingung	Es existiert eine Route; Es muss ein Schiff vorhanden sein; Auf der gegebenen Route existiert ein statisches Hindernis.
Abschlussbedingung	In der Simulation ist zu sehen, dass ein Agent auf einer Testroute, mit einem dort existierenden statischen Hindernis, dieses erkennt und diesem ausweicht.

Tabelle 20: User-Story 4

Nummer	4.1
Name	Erkennung statischer Hindernisse, die sich auf der Route des Agenten befinden.
Akteure	Agent.
Beschreibung	Alle in der Nähe unseres Agenten befindlichen statischen Hindernisse können über den NoGo-Solver und über die Auswertung von Seekartenobjekten erkannt werden. Ein statisches Hindernis definiert sich als ein Objekt auf See, dem ein Schiff ausweichen muss, damit die Weiterfahrt uneingeschränkt fortgeführt werden kann.
Ergebnisziele	
Anfangsbedingung	Es existiert eine Route; Es muss ein Agent vorhanden sein; Auf der gegebenen Route existiert ein statisches Hindernis.
Abschlussbedingung	Der Agent kann ein statisches Hindernis auf seiner Route erkennen und in einem beispielhaften Szenario vorerst vor dem erkannten Hindernis zum Stehen kommen.

Tabelle 21: User-Story 4.1

Nummer	4.1.1
Name	Anfrage über die hinterlegten Objekte.
Akteure	Agent.
Beschreibung	Es werden Anfragen an den NoGo-Solver und an die Seekarten erstellt, aus denen die erforderlichen Daten für die Verfolgung der Route gezogen werden können.
Ergebnisziele	

Anfangsbedingung	NoGo-Solver und Seekarten-Server müssen aktiv sein, damit die benötigten Objekte ausgelesen werden können.
Abschlussbedingung	4.1.1.1 und 4.1.1.2 sind erfüllt.

Tabelle 22: User-Story 4.1.1

Nummer	4.1.1.1
Name	Anfrage über die hinterlegten Objekte des NoGo-Solvers.
Akteure	Agent.
Beschreibung	Es werden Anfragen an den NoGo-Solver erstellt, aus denen die erforderlichen Daten für die Verfolgung der Route gezogen werden können. Danger-Areas definieren sich als Seekartenausschnitte, bei denen ein erhöhtes Gefahrenpotenzial besteht, aber eine Passage physikalisch möglich ist. NoGo-Areas beschreiben Seekartenausschnitte, für die eine Passage physikalisch unmöglich ist.
Ergebnisziele	
Anfangsbedingung	NoGo-Solver muss aktiv sein, damit die benötigten Objekte ausgelesen werden können.
Abschlussbedingung	Anfrage an die Objekte des NoGo-Solvers wurden erfolgreich gespeichert.

Tabelle 23: User-Story 4.1.1.1

Nummer	4.1.1.2
Name	Anfrage über die hinterlegten Objekte in den Seekarten.
Akteure	Agent.
Beschreibung	Es werden Anfragen an die Seekarten erstellt, aus denen die erforderlichen Daten für die Verfolgung der Route gezogen werden können. In Seekartenobjekten muss ausgewertet werden, ob das jeweilige Objekt für unseren Schiffstyp als statisches Hindernis angesehen werden kann.
Ergebnisziele	
Anfangsbedingung	Seekarten-Server muss aktiv sein, damit die benötigten Objekte ausgelesen werden können.

Abschluss-bedingung	Anfrage an die Objekte der Seekarte wurden erfolgreich gespeichert.

Tabelle 24: User-Story 4.1.1.2

Nummer	4.1.2
Name	Filterung der Daten.
Akteure	Agent.
Beschrei-bung	Bevor die Daten in die Struktur des Quadtrees eingefügt werden, wird die Filterung für eine vorgeschriebene Route vorgenommen.
Ergebnis-ziele	
Anfangsbe-dingung	Auf der Route eines Agenten liegen statische Hindernisse vor, die durch die vorangestellte Anfrage (4.1.1) erkannt wurden.
Abschluss-bedingung	In einem dargestellten Seekartenausschnitt werden nur noch die statischen Hindernisse erkannt, welche sich direkt auf unserer gegebenen Routen-kante befinden, ausgegeben.

Tabelle 25: User-Story 4.1.2

Nummer	4.1.3
Name	Antworten des NoGo-Solvers und aus den Seekarten werden im Quadt-ree abgelegt.
Akteure	Agent.
Beschreibung	Die Daten, die vom NoGo-Solver oder aus den Seekarten zurückgegeben werden, werden anschließend in die Struktur des Quadtrees eingefügt.
Ergebnisziele	
Anfangsbe-dingung	Die Objekte auf einer existierenden Route werden erkannt und liegen ge-filtert vor (4.1.1 und 4.1.2 sind erfüllt).
Abschlussbe-dingung	Objekte aus den Seekarten und aus dem NoGo-Solver liegen performant in der Datenstruktur des Quadtrees vor.

Tabelle 26: User-Story 4.1.3

Nummer	4.2
Name	Der Agent kann statischen Hindernissen, die sich auf seiner Route befinden, ausweichen.
Akteure	Agent.
Beschreibung	Der Agent berechnet seinen Kurs und seine Geschwindigkeit und leitet ein Ausweichmanöver ein.
Ergebnisziele	E 5.2: Der Zeitpunkt der Manövereinleitung muss variieren können. E 5.3: Die Geschwindigkeitstypen der Agenten müssen variieren können. E 5.4: Die Beschleunigungstypen der Agenten müssen variieren können. E 7: In der MTS muss ein Agent statischen Hindernissen ausweichen können. E 7.1: Ein Agent muss NoGo-Areas ausweichen können. E 7.2: Ein Agent muss Seekartenobjekten ausweichen können.
Anfangsbedingung	Es gibt eine vorgegebene Route, auf der ein statisches Hindernis erkannt wurde; Es gibt einen Agenten, dem ein Verhalten hinzugefügt werden kann.
Abschlussbedingung	Zwei Agenten des gleichen Schifftyps, aber mit unterschiedlicher Parametrisierung, können in der Simulation einem, auf ihrer Route befindlichen und bereits erkannten statischen Hindernis ausweichen.

Tabelle 27: User-Story 4.2

Nummer	4.2.1
Name	Entscheidung über die Ausweichrichtung.
Akteure	Agent.
Beschreibung	Der Agent muss auf Basis seiner Parametrisierung entscheiden, ob das, auf der Route befindliche, Hindernis auf Steuerbord- oder Backbordseite umfahren wird oder der Agent gar zum Stehen kommen sollte.
Ergebnisziele	E 7: In der MTS muss ein Agent statischen Hindernissen ausweichen können. E 7.1: Ein Agent muss NoGo-Areas ausweichen können. E 7.2: Ein Agent muss Seekartenobjekten ausweichen können.
Anfangsbedingung	Es gibt eine vorgegebene Route, auf der ein statisches Hindernis erkannt wurde; Es gibt einen Agenten, dem ein Verhalten hinzugefügt werden kann.
Abschlussbedingung	Zwei Agenten des gleichen Schifftyps, aber mit unterschiedlicher Parametrisierung, können in der Simulation einem, auf ihrer Route befindlichen und bereits erkannten statischen Hindernis ausweichen. Die Art des Ausweichens

	unterscheidet sich zwischen den beiden Agenten in der Art, dass dem statischen Hindernis in unterschiedlichen Richtungen ausgewichen wird.

Tabelle 28: User-Story 4.2.1

Nummer	4.2.2
Name	Entscheidung mit welchem Abstand das statische Hindernis umfahren werden soll.
Akteure	Agent.
Beschreibung	Der Agent muss auf Basis seiner Parametrisierung entscheiden, mit welchem Sicherheitsabstand er das, auf seiner Route befindliche, Hindernis umfährt.
Ergebnisziele	E 5.2: Der Zeitpunkt der Manövereinleitung muss variieren können. E 7: In der MTS muss ein Agent statischen Hindernissen ausweichen können. E 7.1: Ein Agent muss NoGo-Areas ausweichen können. E 7.2: Ein Agent muss Seekartenobjekten ausweichen können.
Anfangsbedingung	Es gibt eine vorgegebene Route, auf der ein statisches Hindernis erkannt wurde; Es gibt einen Agenten, dem ein Verhalten hinzugefügt werden kann.
Abschlussbedingung	Ein Agent, der eine risikofreudige Parametrisierung zugewiesen bekommt, weicht dem statischen Hindernis mit dem gleichen Schiffstyp mit einem geringeren Sicherheitsabstand aus, als ein Agent der ein risikoaverses Verhalten aufweist.

Tabelle 29: User-Story 4.2.2

Nummer	4.2.3
Name	Geschwindigkeitsregulierung beim Ausweichen eines statischen Hindernisses.
Akteure	Agent.
Beschreibung	Der Agent muss auf Basis seiner Parametrisierung entscheiden, wie er die Geschwindigkeit beim Ausweichen eines statischen Hindernisses reguliert oder ob er gar zum Stehen kommen sollte/muss.
Ergebnisziele	E 5.3: Die Geschwindigkeitstypen der Agenten müssen variieren können. E 5.4: Die Beschleunigungstypen der Agenten müssen variieren können. E 7: In der MTS muss ein Agent statischen Hindernissen ausweichen können. E 7.1: Ein Agent muss NoGo-Areas ausweichen können. E 7.2: Ein Agent muss Seekartenobjekten ausweichen können.

Anfangs-bedingung	Es gibt eine vorgegebene Route, auf der ein statisches Hindernis erkannt wurde; Es gibt einen Agenten, dem ein Verhalten hinzugefügt werden kann.
Ab-schlussbe-dingung	Der Agent kann, sobald ein statisches Hindernis auf seiner Route erkannt wurde, seine Geschwindigkeit unter Berücksichtigung seiner Parametrisie-rung und seines Schiffstyps anpassen. Sollte ein Hindernis nicht umfahren werden können, aufgrund der Gegebenheiten des Schiffes oder dem Risiko-verhalten des Agenten, so bleibt dieser vor dem Hindernis stehen.

Tabelle 30: User-Story 4.2.3

3.3.5 User Story 5: Begegnungsbehandlung von dynamischen Hindernissen

Meilenstein: Begegnungsbehandlung von dynamischen Hindernissen

Nummer	5
Name	Begegnungsbehandlung von dynamischen Hindernissen
Akteure	Agent
Beschrei-bung	Ein Agent muss dynamische Hindernisse erkennen und diesen je nach Para-metrisierung ausweichen können. Hierbei muss mit einer sicheren Ge-schwindigkeit gefahren werden, sodass Maßnahmen getroffen werden kön-nen, um einen Zusammenstoß zu vermeiden und innerhalb einer, den gege-benen Umständen und Bedingungen, entsprechenden Entfernung, zum Ste-hen zu kommen. Für eine sichere Geschwindigkeit ausschlaggebend sind die Verkehrsdichte, die Manövrierfähigkeit und der Tiefgang im Verhältnis zur vorhandenen Wassertiefe. (KVR-Regel 6)
Ergebnis-ziele	E5.1: Passierabstände müssen variieren können. E5.2: Der Zeitpunkt der Manövereinleitung muss variieren können. E5.3: Die Geschwindigkeitstypen der Agenten müssen variieren können. E5.4: Die Beschleunigungstypen der Agenten müssen variieren können. E5.6: Die Befolgung von KVR-Regeln muss variieren können. E8: In der MTS muss eine Begegnungsbehandlung für dynamische Objekte stattfinden. E8.1: In der MTS müssen Agenten entsprechend der KVR auf andere kreu-zende Schiffe reagieren können und selbst andere Schiffe KVR-konform kreuzen können. E8.2: In der MTS müssen Agenten entsprechend der KVR Überholmanöver durchführen und darauf reagieren können. E8.3: In der MTS müssen Agenten entsprechend der KVR Abstand zu ande-ren Schiffen halten können.
Anfangs-bedingung	Es existiert eine Route; Es muss ein Schiff vorhanden sein; Auf der gegebe-nen Route existiert ein dynamisches Hindernis.

Abschlussbedingung	In der Simulation ist zu sehen, dass ein Agent auf einer Testroute, mit einem dort existierenden dynamischen Hindernis, dieses erkennt und diesem entsprechend der KVR ausweicht.

Tabelle 31: User-Story 5

Nummer	5.1
Name	Erkennung dynamischer Hindernisse, die sich auf der Route des Agenten befinden.
Akteure	Agent
Beschreibung	Der Agent muss, mit allen ihm zur Verfügung stehenden Mitteln (AIS, Sicht), feststellen, ob bei gegebenen Parametern, die Gefahr eines Zusammenstoßes mit einem dynamischen Hindernis besteht. Im Zweifelsfall ist diese Möglichkeit anzunehmen. Ein dynamisches Hindernis definiert sich als ein bewegliches Objekt auf See. (KVR-Regel 7)
Ergebnisziele	E8: In der MTS muss eine Begegnungsbehandlung für dynamische Objekte stattfinden.
Anfangsbedingung	Es existiert eine Route; Es muss ein Agent vorhanden sein; Auf der gegebenen Route existiert ein dynamisches Hindernis.
Abschlussbedingung	Sobald der Agent ein, auf seiner Route befindliches, dynamisches Hindernis erkannt hat, passt er seine Bahn gegebenenfalls an.

Tabelle 32: User-Story 5.1

Nummer	5.1.1
Name	Erkennung eines dynamischen Hindernisses mithilfe von AIS-Daten
Akteure	Agent
Beschreibung	Durch das Auslesen der AIS-Daten werden andere, sich auf der Route befindliche Schiffe, erkannt.
Ergebnisziele	
Anfangsbedingung	Es existiert eine Route; Es muss ein Agent vorhanden sein; Auf der gegebenen Route existiert ein dynamisches Hindernis.
Abschlussbedingung	Wenn ein dynamisches Hindernis erkannt wird, liegen die AIS-Daten des jeweils anderen Schiffs vor. Die Ergebnisse der Anfrage werden korrekt zurückgegeben.

Tabelle 33: User-Story 5.1.1

Nummer	5.1.2
Name	Erkennung eines dynamischen Hindernisses, welches sich in Sichtweite befindet
Akteure	Agent
Beschreibung	Sämtliche Objekte, die sich im Sichtfeld des Agenten befinden, werden erkannt.
Ergebnisziele	
Anfangsbedingung	Es existiert eine Route; Es muss ein Agent vorhanden sein; Auf der gegebenen Route existiert ein dynamisches Hindernis.
Abschlussbedingung	In einem beispielhaften Szenario erkennt der Agent alle dynamischen Hindernisse, die sich in einem Sichtfeld mit einer Reichweite von 100 Metern befinden.

Tabelle 34: User-Story 5.1.2

Nummer	5.2
Name	Passieren eines dynamischen Hindernisses
Akteure	Agent
Beschreibung	Der Agent reagiert auf die Begegnung mit einem dynamischen Hindernis mittels parameterbasiertem Manöver zur Vermeidung von Zusammenstößen. Des Weiteren muss jede Änderung der Geschwindigkeit und/oder des Kurses so groß sein, dass ein anderes Schiff diese optisch erkennen kann und kein Nahbereich eines dritten dynamischen Hindernisses tangiert. Ein Manöver muss einen sicheren Passierabstand gewährleisten und endet, sobald das dynamische Hindernis endgültig passiert wurde. Bei akuter Gefahr eines Zusammenstoßes muss ein Agent die Fahrt mindern oder durch stoppen oder rückwärtsgehen jegliche Fahrt wegnehmen.
Ergebnisziele	E5.1: Passierabstände müssen variieren können. E5.2: Der Zeitpunkt der Manövereinleitung muss variieren können. E5.3: Die Geschwindigkeitstypen der Agenten müssen variieren können. E5.4: Die Beschleunigungstypen der Agenten müssen variieren können. E5.6: Die Befolgung von KVR-Regeln muss variieren können. E8: In der MTS muss eine Begegnungsbehandlung für dynamische Objekte stattfinden. E8.1: In der MTS müssen Agenten entsprechend der KVR auf andere kreuzende Schiffe reagieren können und selbst andere Schiffe KVR-konform kreuzen können. E8.2: In der MTS müssen Agenten entsprechend der KVR Überholmanöver

	durchführen und darauf reagieren können. E8.3: In der MTS müssen Agenten entsprechend der KVR Abstand zu anderen Schiffen halten können.
Anfangsbedingung	5.1 ist erfüllt; Es existiert eine Route; Es muss ein Agent vorhanden sein; Auf der gegebenen Route existiert ein dynamisches Hindernis.
Abschlussbedingung	Ein, sich auf der Route befindliches, dynamisches Hindernis kann passiert werden.

Tabelle 35: User-Story 5.2

Nummer	5.2.1
Name	Kreuzen eines dynamischen Hindernisses
Akteure	Agent
Beschreibung	Kreuzt der Agent ein dynamisches Hindernis oder wird gekreuzt, und besteht hierbei die Gefahr eines Zusammenstoßes, muss dasjenige parameterbasiert ausweichen (Ausweichpflichtig), welches das andere (Kurshalter) an seiner Steuerbordseite hat; wenn die Umstände es zulassen, muss es vermeiden, den Bug des anderen Fahrzeugs zu kreuzen.
Ergebnisziele	E8.1: In der MTS müssen Agenten entsprechend der KVR auf andere kreuzende Schiffe reagieren können und selbst andere Schiffe KVR-konform kreuzen können.
Anfangsbedingung	5.1 ist erfüllt; Es existiert eine Route; Es muss ein Agent vorhanden sein; Auf der gegebenen Route existiert ein dynamisches Hindernis, welches unsere Route kreuzt.
Abschlussbedingung	Einem kreuzenden dynamischen Hindernis weicht der Agent unter parameterbasierter Anwendung der KVR aus.

Tabelle 36: User-Story 5.2.1

Nummer	5.2.2
Name	Überholen eines dynamischen Hindernisses
Akteure	Agent
Beschreibung	Von einem Überholmanöver spricht man, wenn sich ein Schiff einem anderen Schiff aus einer Richtung von mehr als 22,5 Grad achterlich als querab nähert. Ist nicht klar ersichtlich, ob ein anderes Schiff den Agenten überholen möchte, so ist dies anzunehmen. Ein Schiff, das überholt wird, gilt hierbei als Kurshalter.

Ergebnis-ziele	E 8.2: In der MTS müssen Agenten entsprechend der KVR Überholmanöver durchführen und darauf reagieren können.
Anfangs-bedingung	5.1 ist erfüllt; Es existiert eine Route; Es muss ein Agent vorhanden sein; Auf der gegebenen Route existiert ein dynamisches Hindernis, welches uns überholt oder überholt werden soll.
Ab-schlussbe-dingung	Ein Agent führt ein Überholmanöver unter parameterbasierter Anwendung der KVR durch.

Tabelle 37: User-Story 5.2.2

Nummer	5.2.3
Name	Entgegengesetzte Kurse zweier Schiffe
Akteure	Agent
Beschrei-bung	Bei entgegengesetzten Kursen zweier Schiffe müssen beide Schiffe ihren Kurs nach Steuerbord so ändern, dass sie einander an Backbordseite passieren. Ein entgegengesetzter Kurs liegt vor, wenn ein Agent ein anderes Schiff frontal voraus sieht. Ist nicht erkennbar, ob eine solche Lage besteht, so muss dies angenommen werden.
Ergebnis-ziele	E8: In der MTS muss eine Begegnungsbehandlung für dynamische Objekte stattfinden.
Anfangs-bedingung	5.1 ist erfüllt; Es existiert eine Route; Es muss ein Agent vorhanden sein; Auf der gegebenen Route existiert ein Schiff, welches frontal auf den Agenten zufährt.
Ab-schlussbe-dingung	Ein Agent reagiert auf ein frontal auf ihn zufahrendes Schiff unter parameterbasierter Anwendung der KVR.

Tabelle 38: User-Story 5.2.3

Nummer	5.2.4
Name	Maßnahmen des Kurshalters
Akteure	Agent
Beschrei-bung	Als Kurshalter wird ein Schiff bezeichnet, welches nicht ausweichpflichtig ist, hierbei muss es Kurs und Geschwindigkeit beibehalten. Sobald die Abwendung eines Zusammenstoßes bei beibehaltenem Kurs nicht mehr möglich ist, muss der Kurshalter selbst ein Ausweichmanöver einleiten.

Ergebnisziele	E8: In der MTS muss eine Begegnungsbehandlung für dynamische Objekte stattfinden.
Anfangsbedingung	5.1, 5.2.1, 5.2.2
Abschlussbedingung	Ein Schiff, das überholt oder gekreuzt wird, hält seinen Kurs und seine Geschwindigkeit und reagiert im Notfall mit einem Manöver des letzten Augenblicks.

Tabelle 39: User-Story 5.2.4

3.3.6 User Story 6: Einbau von Fehlermodellen

Meilenstein: Einbau von Fehlermodellen

Nummer	6
Name	Einbau von Fehlermodellen
Akteure	Agent
Beschreibung	Um zu verhindern, dass in der Simulation zwei oder mehr Agenten mit gleichen Parametern, die vorgegebene Route exakt im selben Muster befahren, wird ein Fehlermodell dafür sorgen, dass auch bei gleichen Agenten eine Varianz gegeben ist. Es gibt unterschiedliche Fehlermodelle. Das Fehlermodell ist deaktivierbar. Das Fehlermodell soll auf einer passenden mathematischen Grundlage basieren. Auch das Fehlermodell soll parametrisierbar sein. Das Fehlermodell soll erweiterbar sein.
Ergebnisziele	E 5.7: Allen variierenden Parametern soll jeweils ein Fehlermodell unterliegen. E6: Das Verhalten eines Agenten muss einem Fehlermodell unterliegen.
Anfangsbedingung	Einem Agenten muss ein Schiff zugewiesen sein. Das Schiff nimmt aktiv am Verkehr teil.
Abschlussbedingung	In der Simulation ist zu sehen, dass zwei gleiche Agenten (gleiches Verhalten, gleiche Parameter) beim strikten Abfahren einer Testroute und der deaktivierten Einstellung zur Verhinderung des Aufperleffektes, aufgrund des Fehlermodells, Unterschiede beim Befahren der Route aufzeigen.

Tabelle 40: User-Story 6

Nummer	6.1

Name	Wahl eines mathematischen Modells für den Fehlerkatalog
Akteure	PG
Beschreibung	Bestimmung einer mathematischen Grundlage für das Fehlermodell dient als Triggermechanismus, mit dem der Injektionszeitpunkt für ein Fehlverhalten bestimmt wird.
Ergebnisziele	
Anfangsbedingung	
Abschlussbedingung	Häufigkeit des Auftretens eines Fehlers sowie die jeweilige Fehlerart können über ein passendes mathematisches Modell bestimmt werden.

Tabelle 41: User-Story 6.1

Nummer	6.2
Name	Fehlermodelle anlegen
Akteure	PG
Beschreibung	Es werden mehrere unterschiedliche Fehlermodelle angelegt, die später in HAGGIS ausgewählt werden können.
Ergebnisziele	E6: Das Verhalten eines Agenten muss einem Fehlermodell unterliegen.
Anfangsbedingung	Es existiert ein Fehlerkatalog.
Abschlussbedingung	Einem Agenten lassen sich Fehlermodelle zuweisen.

Tabelle 42: User-Story 6.2

Nummer	6.2.1
Name	Erstellung eines Fehlerkataloges
Akteure	PG
Beschreibung	Damit einem Agenten verschiedene Fehler zugewiesen werden können, müssen Fehlverhalten definiert werden.

Ergebnisziele	E 5.7: Allen variierenden Parametern soll jeweils ein Fehlermodell unterliegen. E6: Das Verhalten eines Agenten muss einem Fehlermodell unterliegen.
Anfangsbedingung	Für eine realistische Darstellung von Fehlverhalten, ist eine Analyse von real auftretenden Unfällen im maritimen Bereich notwendig.
Abschlussbedingung	Ein Fehlerkatalog mit verschiedenen Fehlverhalten wurde erstellt.

Tabelle 43: User-Story 6.2.1

3.3.6.1 Fehlerkatalog Schiffssimulation

Fehler	**Auswirkung**
F1. Mangelnde Sicht	Späte Reaktion, evtl. Manöver des letzten Augenblicks
F2. Falsche Wahrnehmung des OoW	F2.1 Falsche Situationserkennung (Crossing wird als entgegengesetzter Kurs erkannt oder umgekehrt) F2.2. Distanz zum Hindernis wird nicht korrekt eingeschätzt.
F3. Unaufmerksamkeit, Ermüdung, Drogeneinfluss	F3.1. Späte Reaktion, evtl. Manöver des letzten Augenblicks F3.2. Verlassen des geplanten Kurses
F4. Unzureichende Kenntnisse	F4.1. Vorsichtige Fahrweise F4.2. Keine vorausschauende Fahrweise
F5. Kommunikationsfehler zwischen zwei Schiffen	Ausweichpflichtiger hält Kurs oder umgekehrt
F6. Regelverstöße	F6.1. Ausweichpflichtiger hält Kurs F6.2. Falsche Position im Fahrwasser F6.3. Missachtung von Speergebieten. F6.4. Überschreiten der vorgeschriebenen Geschwindigkeit.

Tabelle 44: Fehlerkatalog

Nummer	6.2.2
Name	Fehlermodell ist parametrisierbar
Akteure	Anwender
Beschreibung	Das aktivierte Fehlermodell, lässt sich in seiner Häufigkeit und Intensität konfigurieren. Das bedeutet, dass der Fehler des Fehlermodells entweder selten oder

	häufig auftritt. Die Intensität entscheidet, wie stark die Auswirkungen des Fehlers die Simulation beeinflussen.
Ergebnisziele	E 5.7: Allen variierenden Parametern soll jeweils ein Fehlermodell unterliegen.
Anfangsbedingung	Einem Agenten muss ein Schiff zugewiesen sein. Das Schiff nimmt aktiv am Verkehr teil. Dem Agenten sind ein Verhalten und ein Fehlermodell zugewiesen.
Abschlussbedingung	Zwei Agenten weisen Unterschiede in Häufigkeit und Intensität eines Fehlers auf.

Tabelle 45: User-Story 6.2.2

Nummer	6.3
Name	Fehlermodell für das Verhalten
Akteure	Agent
Beschreibung	Das Verhalten des Agenten wird durch ein Fehlermodell beeinflusst.
Ergebnisziele	E6: Das Verhalten eines Agenten muss einem Fehlermodell unterliegen.
Anfangsbedingung	Einem Agenten muss ein Schiff zugewiesen sein. Das Schiff nimmt aktiv am Verkehr teil.
Abschlussbedingung	In der Simulation ist zu sehen, dass zwei gleiche Agenten (gleiches Verhalten, gleiche Parameter) beim strikten Abfahren einer Testroute und der deaktivierten Einstellung zur Verhinderung des Aufperleffektes, aufgrund des Fehlermodells, Unterschiede beim Befahren der Route aufzeigen.

Tabelle 46: User-Story 6.3,

Nummer	6.3.1
Name	Fehlermodell beeinflusst die Bahn des Agenten
Akteure	Agent
Beschreibung	Der Agent verwendet nicht zwingend die optimale Bahn.
Ergebnisziele	E6: Das Verhalten eines Agenten muss einem Fehlermodell unterliegen.
Anfangsbedingung	Einem Agenten muss ein Schiff zugewiesen sein. Das Schiff nimmt aktiv am Verkehr teil. Der Agent wählt immer die optimale Route.

Abschlussbedingung	Der Agent verfolgt nicht immer die schnellste Bahn, sondern wählt eine alternative Bahn.

Tabelle 47: User-Story 6.3.1

Nummer	6.3.2
Name	Fehlermodell beeinflusst die gewählte Charakteristik des Verhaltens.
Akteure	Agent
Beschreibung	Das zuvor ausgewählte Verhalten für einen Agenten, kann durch das Fehlermodell derart stark beeinflusst werden, als dass kurzzeitig ein Verhalten auftritt, welches sich außerhalb der Norm des zuvor ausgewählten Verhaltens befindet.
Ergebnisziele	E6: Das Verhalten eines Agenten muss einem Fehlermodell unterliegen.
Anfangsbedingung	Einem Agenten muss ein Schiff zugewiesen sein. Das Schiff nimmt aktiv am Verkehr teil. Dem Agenten ist ein Verhalten zugewiesen.
Abschlussbedingung	Ein Agent, welcher beispielsweise ein vorsichtiges Verhalten zugewiesen bekommen hat, kann trotzdem eine Gefahr übersehen.

Tabelle 48: User-Story 6.3.2

Nummer	6.4
Name	Fehlermodell ist aktivierbar
Akteure	Agent
Beschreibung	Das Fehlermodell kann je nach Situation aktiviert bzw. deaktiviert werden.
Ergebnisziele	E 5.7: Allen variierenden Parametern soll jeweils ein Fehlermodell unterliegen. E6: Das Verhalten eines Agenten muss einem Fehlermodell unterliegen.
Anfangsbedingung	Einem Agenten muss ein Schiff zugewiesen sein. Das Schiff nimmt aktiv am Verkehr teil.
Abschlussbedingung	Am Verhalten des Agenten kann festgestellt werden, ob dieser mit oder ohne Fehlermodell fährt.

Tabelle 49: User-Story 6.4

Nummer	6.4.1
Name	Fehlermodell ist nicht permanent
Akteure	Agent
Beschreibung	Das Fehlermodell weist einem Agenten zufällig Fehler zu. So handeln diese bei gleichen Situationen unterschiedlich.
Ergebnisziele	E 5.7: Allen variierenden Parametern soll jeweils ein Fehlermodell unterliegen. E6: Das Verhalten eines Agenten muss einem Fehlermodell unterliegen.
Anfangsbedingung	Einem Agenten muss ein Schiff zugewiesen sein. Das Schiff nimmt aktiv am Verkehr teil. Der Agent besitzt ein aktiviertes Fehlermodell.
Abschlussbedingung	Der Agent handelt zufällig regelkonform oder abweichend. Das Handeln des Agenten kann in gleichen Situationen verschiedene Vorgehensweisen aufweisen.

Tabelle 50: User-Story 6.4.1

3.3.7 User Story 7: Verhalten definieren (Charaktereigenschaften)

Meilenstein: Verhalten definieren (Charaktereigenschaften)

Nummer	7
Name	Verhalten definieren (Charaktereigenschaften)
Akteure	PG
Beschreibung	Die Agenten sollen, unabhängig vom Fehlermodell, unterschiedliche Charakterausprägungen aufweisen. Hierzu ist es notwendig, dass dem Agenten unterschiedliche Verhalten zugewiesen werden können.
Ergebnisziele	E4 Das Befahren von Routen mittels Bahnen muss durch das Verhalten beeinflusst werden. E5 Das Verhalten muss durch Toleranzwerte in der MTS (HAGGIS Control) individualisierbar sein.
Anfangsbedingung	Ein Agent muss ausgewählt sein.
Abschlussbedingung	Die Verhaltensausprägung für den Agenten lässt sich individuell einstellen. Der Agent weist, abhängig von seinem Verhalten, Unterschiede bei dem Befahren von Routen und Bahnen auf.

Tabelle 51: User-Story 7

Nummer	7.1.
Name	Anlegen von Reglern
Akteure	PG
Beschreibung	Es werden Schieberegler in HAGGIS integriert, die es dem Nutzer ermöglichen, die Stärke der Verhaltensausprägung festzulegen. Die Regler lassen sich mittels einer Skala von niedrig bis hoch steuern. Die exakte Skalierung wird im Hintergrund, unter Berücksichtigung der jeweiligen Parameter, festgelegt.
Ergebnisziele	E5 Das Verhalten muss durch Toleranzwerte in der MTS (HAGGIS Control) individualisierbar sein.
Anfangsbedingung	Ein Agent muss ausgewählt sein.
Abschlussbedingung	Die Verhaltensausprägung für den Agenten lässt sich individuell einstellen.

Tabelle 52: User-Story 7.1

Nummer	7.1.1
Name	Einflussgrößen auf das Verhalten
Akteure	Anwender
Beschreibung	Folgende Parameter nehmen Einfluss auf das Verhalten des Kapitäns: 1. sightDistance (Sichtweite des Schiffes/Agenten) 2. headOnCollisionAngle (Bereich einer Head-On Situation) 3. thresholdHeading (Reaktionswinkel für Manövereinleitung) 4. overtakeAngle (Ab wann wird ein Überholmanöver erkannt) 5. boxLength (mit welcher Distanz werden statische Hindernisse erkannt) 6. Buffer (Distanz ergänzend zum Sicherheitsabstand) 7. turningAngle (Gibt an, ab wann eine Kursabweichung eine Kursänderung ist) 8. passGap (legt den Passierabstand zu Drittschiffen fest) 9. velocity (Geschwindigkeit des Agenten in Relation zur maximalen Geschwindigkeit des Schiffes)
Ergebnisziele	E4 Das Befahren von Routen mittels Bahnen muss durch das Verhalten beeinflusst werden. E5 Das Verhalten muss durch Toleranzwerte in der MTS (HAGGIS Control) individualisierbar sein.

Anfangsbedingung	Ein Agent muss ausgewählt sein.
Abschlussbedingung	Der Anwender kann in der Simulation, mittels der oben genannten Regler, dem Agenten ein individuelles Verhalten zuweisen.

Tabelle 53: User-Story 7.1.1

Nummer	7.1.2
Name	Vordefinierte Charaktere
Akteure	HAGGIS
Beschreibung	In HAGGIS werden vordefinierte Charaktere angelegt. Diese verfügen über festgelegte Voreinstellungen der Regler aus User Story 7.1.2. Die Voreinstellungen können nach Auswahl des Charakters durch den Nutzer angepasst werden. Ein beispielhaft voreingestellter Charakter ist der „rücksichtslose Kapitän". Dieser hätte folgende Einstellungen: 1. sightDistance: mittel 2. headOnCollisionAngle: mittel 3. thresholdHeading: niedrig 4. overtakeAngle: mittel 5. boxLength: mittel 6. Buffer: niedrig 7. turningAngle: hoch 8. passGap: niedrig 9. velocity: hoch
Ergebnisziele	E4 Das Befahren von Routen mittels Bahnen muss durch das Verhalten beeinflusst werden. E5 Das Verhalten muss durch Toleranzwerte in der MTS (HAGGIS Control) individualisierbar sein.
Anfangsbedingung	User Story 7.1.2 ist abgeschlossen. Der Katalog mit den vordefinierten Charakteren muss in HAGGIS angelegt sein. Ein Agent muss ausgewählt sein.
Abschlussbedingung	Der Anwender kann in der Simulation vordefinierte Charaktere auswählen und diese bei Bedarf im Nachhinein anpassen. Der Agent weist, abhängig von seinem Verhalten, Unterschiede bei dem Befahren von Routen und Bahnen auf.

Tabelle 54: User-Story 7.1.2

3.3.8 User Story 8: Ortsabhängig Regeln befolgen

Meilenstein: Ortsabhängig Regeln befolgen

Nummer	8
Name	Ortsabhängig Regeln befolgen
Akteure	Agent
Beschreibung	Der Agent erkennt seine Umgebung und weiß entsprechend welche Verkehrsregeln er berücksichtigen muss. Der Agent handelt entsprechend. Die berücksichtigten Regelwerke sind erweiterbar.
Ergebnisziele	E9: Der Agent muss ortsabhängig Regeln befolgen. E9.1: Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.
Anfangsbedingung	Es existiert eine Route, es ist ein Schiff vorhanden. Das Fehlermodell ist ausgeschaltet und der Standard-Kapitän ist ausgewählt.
Abschlussbedingung	In der Simulation ist zu sehen, dass das Schiff die Ortsabhängigen Regeln vorrangig gegenüber den KVR berücksichtigt.

Tabelle 55: User-Story 8

Nummer	8.1
Name	Erkennung der vor Ort geltenden Regeln anhand der Position des Agenten
Akteure	Agent
Beschreibung	Der Agent kann anhand seiner Position bestimmen, welche Verkehrsregeln er zu berücksichtigen hat. Gelangt ein Agent in ein Gebiet, für das eine oder mehrere Regeln bestehen, wird die ortsabhängig vorrangige Regel als zu befolgende Regel erkannt.
Ergebnisziele	E9: Der Agent muss ortsabhängig Regeln befolgen. E9.1: Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.
Anfangsbedingung	Es existiert eine Route, es ist ein Schiff vorhanden, das Schiff befindet oder bewegt sich in einem Gebiet mit ortsabhängigen Verkehrsregeln.
Abschlussbedingung	Ausgabe in der Konsole über aktuelles Verkehrsregelwerk sowie die anzuwendende Verkehrsregel (Beispiel: SeeSchStrO, Begegnung).

Tabelle 56: User-Story 8.1

Nummer	8.2
Name	Einhaltung von ortsabhängigen Regeln
Akteure	Agent
Beschreibung	Der Agent handelt entsprechend der bereits erkannten Verkehrsregeln für den Bereich, in dem er sich aktuell befindet.
Ergebnisziele	E9: Der Agent muss ortsabhängig Regeln befolgen. E9.1: Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.
Anfangsbedingung	Es existiert eine Route, es ist ein Schiff vorhanden, das Schiff befindet oder bewegt sich in einem Gebiet mit ortsabhängigen Verkehrsregeln. Die anzuwendenden Regeln wurden vom Agenten erkannt.
Abschlussbedingung	In der Simulation ist zu erkennen, dass der Agent ortsabhängig spezifizierte Regeln gegenüber allgemeingültigen Regeln vorrangig befolgt.

Tabelle 57: User-Story 8.2

Nummer	8.2.1
Name	Rechtsfahrgebot
Akteure	Agent
Beschreibung	Der Agent wird vom Rechtsfahrgebot befreit. (In Anlehnung an §22 SeeSchStrO)
Ergebnisziele	E9: Der Agent muss ortsabhängig Regeln befolgen. E9.1: Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.
Anfangsbedingung	Es existiert eine Route, es ist ein Schiff vorhanden, das Schiff befindet oder bewegt sich in einem Gebiet mit ortsabhängigen Verkehrsregeln, in dem das Rechtsfahrgebot nicht gilt. Diese Regel wurde vom Agenten erkannt.
Abschlussbedingung	In der Simulation ist zu erkennen, dass der Agent das Rechtsfahrgebot nicht zwingend anwendet.

Tabelle 58: User-Story 8.2.1

Nummer	8.2.2
Name	Überholverbot
Akteure	Agent

Beschreibung	Der Agent befolgt Überholverbote. (In Anlehnung an §23 SeeSchStrO)
Ergebnisziele	E9: Der Agent muss ortsabhängig Regeln befolgen. E9.1: Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.
Anfangsbedingung	Es existiert eine Route, es ist ein Schiff vorhanden, das Schiff befindet oder bewegt sich in einem Gebiet mit ortsabhängigen Verkehrsregeln. Die Regel des Überholverbots wurde vom Agenten erkannt.
Abschlussbedingung	In der Simulation ist zu erkennen, dass der Agent sich an Überholverbote hält.

Tabelle 59: User-Story 8.2.2

Nummer	8.2.3
Name	Überholvorgang rechts erlaubt
Akteure	Agent
Beschreibung	Der Agent hat die Möglichkeit auch rechts zu überholen. (In Anlehnung an §23 SeeSchStrO)
Ergebnisziele	E9: Der Agent muss ortsabhängig Regeln befolgen. E9.1: Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.
Anfangsbedingung	Es existiert eine Route, es ist ein Schiff vorhanden, das Schiff befindet oder bewegt sich in einem Gebiet mit ortsabhängigen Verkehrsregeln. Besagte Regel wurde vom Agenten erkannt.
Abschlussbedingung	In der Simulation ist zu erkennen, dass der Agent auch rechts überholen kann sofern es die Regeln zulassen.

Tabelle 60: User-Story 8.2.3

Nummer	8.2.4
Name	Begegnen
Akteure	Agent
Beschreibung	Begegnen sich im Geltungsbereich der SeeSchStrO zwei Schiffe darf gemäß §24 Abs. 3 ausnahmsweise nach Backbord ausgewichen werden.

Ergebnis-ziele	E9: Der Agent muss ortsabhängig Regeln befolgen. E9.1: Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.
Anfangs-bedingung	Es existiert eine Route, es begegnen sich zwei Schiffe in einem Gebiet mit ortsabhängigen Verkehrsregeln. Besagte Regel wurde vom Agenten erkannt. §24 Abs. 3 findet unter Berücksichtigung von §22 Abs. 1 (Aufhebung des Rechtsfahrgebots) Anwendung.
Abschluss-bedingung	In der Simulation ist zu erkennen, dass der Agent in einer Begegnungssituation, in der die Regel Anwendung findet, nicht zwangsläufig nach rechts ausweicht.

Tabelle 61: User-Story 8.2.4

Nummer	8.2.5
Name	Vorfahrt der Schiffe im Fahrwasser
Akteure	Agent
Beschrei-bung	Der Agent berücksichtigt die Vorfahrtsregeln des Schiffsverkehrs im Fahrwasser. (In Anlehnung an §25 SeeSchStrO) Fahrzeuge, die sich im Fahrwasser befinden, haben stets Vorfahrt vor Fahrzeugen außerhalb des Fahrwassers.
Ergebnis-ziele	E9: Der Agent muss ortsabhängig Regeln befolgen. E9.1: Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.
Anfangsbe-dingung	Es existiert eine Route, es ist ein Schiff vorhanden, das Schiff befindet oder bewegt sich in einem Fahrwasser oder will in dieses einfahren bzw. dieses kreuzen. Die Situation wurde vom Agenten erkannt.
Abschluss-bedingung	In der Simulation ist zu erkennen, dass der Agent seine Fahrweise gemäß dieser Regel anpasst.

Tabelle 62: User-Story 8.2.5

Nummer	8.2.6
Name	Fahrgeschwindigkeit
Akteure	Agent
Beschrei-bung	Der Agent fährt mit einer sicheren Geschwindigkeit und berücksichtigt Geschwindigkeitsbeschränkungen. (In Anlehnung an §26 SeeSchStrO) Eine sichere Geschwindigkeit ergibt sich aus der erkannten Situation sowie der Dynamik und Physik des Schiffes.

Ergebnis-ziele	E9: Der Agent muss ortsabhängig Regeln befolgen. E9.1: Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.
Anfangs-bedingung	Es existiert eine Route, es ist ein Schiff vorhanden, das Schiff befindet oder bewegt sich in einem Gebiet in dem die Geschwindigkeit angepasst werden muss. Die Situation wurde vom Agenten erkannt.
Abschluss-bedingung	In der Simulation ist zu erkennen, dass der Agent seine Fahrweise gemäß dieser Regel anpasst.

Tabelle 63: User-Story 8.2.6

Nummer	8.2.7
Name	Durchfahrt von Brücken
Akteure	Agent
Beschrei-bung	Wenn ein Agent im Bereich einer Brücke auf ein weiteres Fahrzeug trifft, passt er seine Geschwindigkeit an, sofern der Platz nicht ausreicht, um aneinander vorbeizufahren. (In Anlehnung an §28 SeeSchStrO)
Ergebnis-ziele	E9: Der Agent muss ortsabhängig Regeln befolgen. E9.1: Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.
Anfangsbe-dingung	Es existiert eine Route, es ist ein Schiff vorhanden, das Schiff befindet oder bewegt sich in einem Gebiet in dem diese Regel nach §28 Anwendung findet. Die Situation wurde vom Agenten erkannt.
Abschluss-bedingung	In der Simulation ist zu erkennen, dass der Agent seine Geschwindigkeit gemäß dieser Regel anpasst.

Tabelle 64: User-Story 8.2.7

3.3.9 User Story 9: Evaluation einzelner Agenten

Meilenstein: Evaluation einzelner Agenten

Nummer	9
Name	Evaluation eines einzelnen Agenten
Akteure	Agent
Beschrei-bung	Die Bahnen eines Agenten, die durch sein Verhalten bestimmt werden, werden auf Abweichungen gegenüber realen Tracks, die aus historischen Daten rekonstruiert werden, überprüft. Abweichungen definieren sich durch die

	räumliche und zeitliche Distanz. Die Tracks, die für den Abgleich herangezogen werden, müssen frei von Verkehrseinflüssen sein. Für die Evaluation dieser Abweichungen werden die erzielten Ergebnisse analysiert, um einen Indikator zu bestimmen, der Realitätstreue definiert. In der GUI von HAGGIS werden die historischen Tracks dargestellt, sodass auch eine Okularinspektion vorgenommen werden kann.
Ergebnisziele	E1.2: Das Verhalten eines einzelnen Agenten muss auf Realitätstreue geprüft werden.
Anfangsbedingung	Zum Abgleich der Bahnen muss ein Track aus historischen Daten gewählt werden, der frei von Störungsfällen ist. Es existiert eine Route, die auf Startpunkt und Endpunkt des ausgewählten Tracks beruht.
Abschlussbedingung	Es existiert ein Indikator anhand dessen realitätsnahes Verhalten eines einzelnen Agenten geprüft werden kann. In der Oberfläche von HAGGIS ist eine Okularinspektion möglich.

Tabelle 65: User-Story 9

Nummer	9.1
Name	Bildung historischer Tracks
Akteure	KNIME
Beschreibung	Mit Hilfe der Daten aus der AIS-Datenbank werden Tracks frei von Verkehrseinflüssen rekonstruiert.
Ergebnisziele	-
Anfangsbedingung	Es gibt einen Zugang zu historischen Daten.
Abschlussbedingung	Es existieren historische und störungsfreie Tracks.

Tabelle 66: User-Story 9.1

Nummer	9.2
Name	Evaluation Stufe 0 Okularinspektion
Akteure	User

Beschreibung	Ein historischer Track wird in der Simulation dargestellt. Der Agent fährt eine Route ab, die ausschließlich den Start- und Zielpunkt dieses Tracks besitzt. Auf diese Weise können die räumlichen und zeitlichen Abweichungen durch Betrachtung der Simulation erkannt werden.
Ergebnisziele	E1.2: Das Verhalten eines einzelnen Agenten muss auf Realitätstreue geprüft werden.
Anfangsbedingung	Es existieren historische störungsfreie Tracks.
Abschlussbedingung	Tracks können in HAGGIS abgebildet werden. Routen für den Agenten können unter Berücksichtigung des Start- und Endpunktes dieses Tracks generiert werden.

Tabelle 67: User-Story 9.2

Nummer	9.3
Name	Evaluation Stufe 1 -> Nutzung von Metriken
Akteure	User
Beschreibung	Mit Hilfe der Daten aus der AIS-Datenbank werden störungsfreie Tracks rekonstruiert. Aus der Simulation werden beim Abfahren der Route durch den Agenten Daten über die Bahnen entnommen, die mit den Daten der historischen Tracks abgeglichen werden. Durch die Analyse dieses Abgleichs werden Metriken erstellt, die einen Indikator für realitätsnahes Verhalten definieren. Bei den Metriken handelt es sich um Werte, die räumliche und zeitliche Abweichungen abbilden.
Ergebnisziele	E1.2: Das Verhalten eines einzelnen Agenten muss auf Realitätstreue geprüft werden.
Anfangsbedingung	Es existieren historische störungsfreie Tracks.
Abschlussbedingung	Es existieren Indikatoren zur Definition realitätsnahen Verhaltens auf Basis von Metriken.

Tabelle 68: User-Story 9.3

4 Systemarchitektur

HAGGIS ist eine virtuelle Simulations- und Modellierungsumgebung, die es ermöglicht, neue e-Maritime Technologien zu testen, ohne den Einsatz von physischem Equipment auf hoher See. Um dieses Ziel zu erreichen, wurde HAGGIS aus mehreren Modulen zusammengesetzt,

die beliebig in verschiedenen Applikationen orchestriert werden können. (vgl. Schweigert et. al., 2014)

Die Komponenten von HAGGIS basieren auf dem Verteilungsansatz des verteilten gemeinsamen Speichers, wodurch die Interoperabilität der einzelnen Komponenten untereinander sichergestellt wird. Darauf aufbauend basiert die daraus resultierende Architektur der Co-Simulation auf dem High Level Architecture (HLA) Standard für Co-Simulationen (“IEEE 1516,” 2000). HLA sieht vor, dass die gesamte Simulation (Federation) in einzelne Teile (Federates) zerlegt wird. Um die Kommunikation zwischen den einzelnen Federates zu ermöglichen, sind diese über eine Laufzeitinfrastruktur miteinander verbunden.

In Abbildung 1 ist die Gesamtarchitektur von HAGGIS abgebildet. Zu sehen sind dort die einzelnen Komponenten, die in den folgenden Abschnitten näher erläutert werden. Zudem ist der Unterschied zwischen Design-Zeit und Laufzeit zu sehen. Zur Design-Zeit werden das Scenario und das zu untersuchende Experiment definiert. Während der Laufzeit wird das Experiment mithilfe verschiedener Simulationen, die beliebig zusammen orchestriert werden können, durchgeführt. Eine gleichzeitige Analyse sowie Beobachtung ist während der Laufzeit möglich. Die Architektur unterliegt der HLA. Als Datengrundlage wird das S-100 Datenmodell verwendet. Das Datenmodell S-100 wird an dieser Stelle nicht weiter erläutert, es wird auf die spezifische Seminararbeit verwiesen.

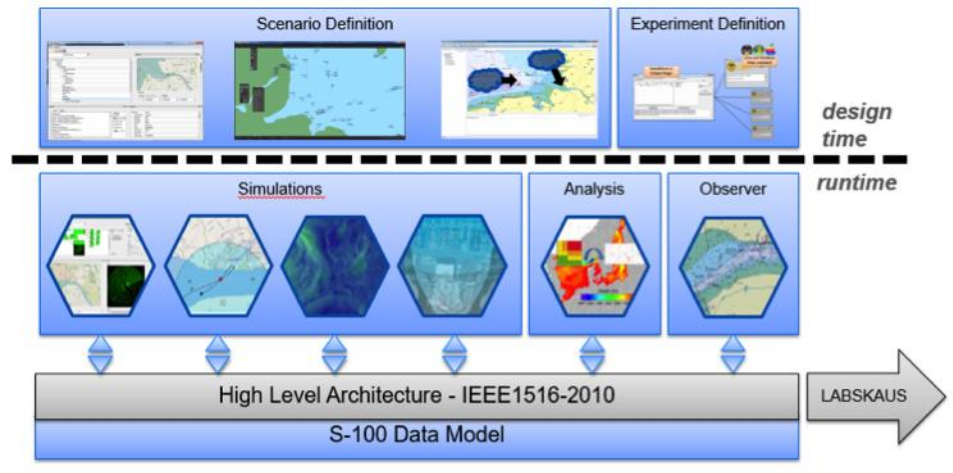


Abbildung 1: Gesamtarchitektur des HAGGIS Frameworks (Gollücke, 2017)

Die folgenden Abschnitte sind nach den Komponenten des HAGGIS Frameworks gegliedert und beschreiben diese näher.

4.1 Szenario Definition

Während der Szenario Definition, die während der Design-Zeit stattfindet, werden grundsätzliche Parameter, des zu untersuchenden Anwendungsfalls, definiert. Dazu zählt die Anzahl der Schiffe sowie deren individuelle Ausgestaltung. Spezifische Attribute sind die MMSI (Maritime Mobile Service Identity), IMO-Nummer, Rufzeichen und Schiffsname. Weitere Merkmale definieren den globalen Standort und die Größe des Schiffes. Sensoren können den Schiffen optional zugeordnet werden. (vgl. Schweigert et. al., 2014)

4.2 Szenario Konfiguration

Die Konfiguration der Co-Simulation besteht aus der Verteilung der HLA Laufzeit-Infrastruktur und der Simulationskomponenten. Das bedeutet, dass die Komponenten benannt werden müssen, die für die Simulation des Szenarios erforderlich sind.

Während der Konfiguration benutzt das Werkzeug DistriCT die Informationen über die Simulationskomponenten, das Testsystem und die Kommunikation der Laufzeitinfrastruktur, um die beteiligten Softwarekomponenten auf verschiedenen Plattformen zu verteilen. (vgl. Schweigert et. al., 2014)

DistriCT verfügt dafür über eine zentrale Kommunikationskomponente, in der alle aktuell kommunizierten Daten der Simulatoren hinterlegt sind. Dieses dient der Risikodistanzberechnung, auf die explizit im Abschnitt 3.5, der automatischen Auswertung, eingegangen wird. Darüber hinaus besitzt DistriCT Kontrollkomponenten, mit denen Simulator-Programme auf unterschiedlichen Systemen gestartet und gestoppt werden können oder Zustände gespeichert und geladen werden können. DistriCT ermöglicht auch die Parameterkonfiguration während der Laufzeit der Simulation, um auf eingehende Analyseergebnisse der Kontrollkomponenten reagieren zu können. Somit trägt DistriCT zu einer deutlich erhöhten Flexibilität von HAGGIS bei. (vgl. Gollücke, 2017)

Nachdem die HLA Laufzeit-Infrastruktur und die Simulatoren spezifiziert und die Parameterkonfiguration abgeschlossen wurde, wird der Simulationsplan generiert. Der Simulationsplan ist ein konfigurierbarer Sequenzfluss, der die Simulationsdurchläufe beschreibt.

Nachdem das Szenario modelliert, der Simulationsplan erstellt und die Laufzeitkomponenten auf verschiedene Systeme verteilt wurden, kann die Simulation gestartet werden. (vgl. Schweigert et. al., 2014)

4.3 Maritime Traffic Simulation

Die Maritime Traffic Simulation (deutsch: Maritime Verkehrssimulation, kurz: MTS) benötigt für die Initialisierung Daten aus der Szenario Definition, insbesondere die Daten der modellierten Schiffe. Dafür nutzt es die Datei, die zur Beschreibung des Szenarios von DistriCT erstellt wurde. Die Vorbereitung umfasst eine Wegeplanung für die Schiffe, denen kein exakter Weg vorgegeben wurde. Die MTS nutzt für die Wegplanung einen A* Algorithmus über ein 50x50m Gitter, das auf Basis einer Seekarte generiert wird. Die Projektgruppe MTSS hat sich dafür entschieden, den A*-Algorithmus ohne ein fest definiertes Raster zu implementieren, da dies in bestimmten Szenarien zu unbefriedigenden Ergebnissen führen könnte (vgl. MTSS, 2016).

Der A*-Algorithmus gehört zur Gruppe der informierten Suchalgorithmen. Der Algorithmus basiert auf dem Dijkstras Algorithmus zur Suche nach dem kürzesten Weg innerhalb eines Graphen. Mithilfe von Heuristiken kann eine gute Performanz erreicht werden (vgl. Reddy, 2013), die dem Gesamtsystem HAGGIS zuträglich ist. Die Wegplanung berechnet nicht nur Wegpunkte auf dem vorgenannten Gitternetz, sondern bezieht auch den Tiefgang des konfigurierten Schiffes, die Wassertiefe und Informationen auf der Seekarte mit ein. Der eingesetzte Algorithmus ist auch in der Lage, Wasserstraßen und Kanäle abzufahren und somit die kürzeste Route zwischen zwei Wegpunkten unter realistischen Bedingungen zu errechnen.

Abbildung 2: Beispiel einer Route mit dem A-Algorithmus (MTSS, 2016)*

Abbildung 2 visualisiert den vorher beschriebenen Sachverhalt an dem praktischen Beispiel der Emsmündung und dreier Schiffe mit deren Wegberechnung. Weitere Informationen sind in der Projektdokumentation (MTSS, 2016) des Projekts „Maritime Transportation Systems Simulation“ (MTSS) enthalten.

Diese Details ließen sich von der Projektgruppe ISS wiederverwenden. Darauf aufbauend könnte eine noch dynamischere Wegplanung realisiert werden, die weitere Gefahren, wie z. B. Kollisionsgefahren mit anderen Schiffen einbezieht und darauf reagiert. Dafür müssen die COLREGs bei der Implementation intelligenten Verhaltens berücksichtigt werden, damit möglichen Gefahren dynamisch begegnet werden kann. Zudem müssen andere Faktoren dynamischer Natur berücksichtigt werden, z. B. Gezeiten, Wetter, sonstige Hindernisse.

4.4 Sensor Simulation

Die Sensor Simulation bietet die Möglichkeit Sensormessungen, wie sie in der echten Welt vorkommen, darzustellen. Dies wird über einen nur lesenden Zugriff auf die Simulationsdaten und die daraus abgeleitete Transformierung der Sensordaten bewerkstelligt. Während der Transformierung werden die Simulationsdaten so angepasst, z. B. werden AIS Daten nur für einen gewissen Raum ausgegeben, sodass sie der Darstellung in der echten Welt bestmöglich nachempfunden werden. Ein wichtiger Teil der Sensor Simulation ist der AIS Emitter. AIS ist gemäß den Vorschriften des SOLAS-Übereinkommens, einer Sonderorganisation der Vereinten Nationen, der Internationalen Seeschifffahrts-Organisation (IMO), auf Schiffen ab einer bestimmten Größe (Bruttoraumanzahl >300) zwingend mitzuführen (vgl. SOLAS Kapitel V,

2002). Der AIS Emitter bezieht unterschiedliche Sensordaten mit ein. So ist beispielsweise der Kurs über Grund ein Teil einer AIS Nachricht, diese Information wird aus der Berechnung der zurückgelegten Strecke und der dafür benötigten Zeit ermittelt. Dies geschieht aus den Daten der Simulation oder, falls entsprechende Sensoren dem Schiff während der Design Zeit zugeordnet wurden, aus den Daten der entsprechenden Sensoren. Ist ein GPS Sensor einem Schiff zugeordnet, so nimmt dieser die Position des Schiffes, die durch die Simulation errechnet wurde und ordnet dieser Position einen statistischen Fehler zu, um eine realistischere Darstellung zu ermöglichen.

Weitere Elemente der Sensor Simulation simulieren die Funkübertragung, über die AIS-Daten übertragen werden. (vgl. Schweigert, et al., 2014)

4.5 Automatische Auswertung

Für die automatische Auswertung eines Simulationsdurchlaufes wird auf das Werkzeug DistriCT zurückgegriffen. Die Updates der einzelnen Simulatoren werden auf die korrespondierenden Datenobjekte der Analyseinstanz abgebildet. Dies erlaubt dem Nutzer eine Berechnung der Distanz zu einem beobachteten Risiko. Für die Berechnung der Distanz wurde eine Risiko Monitor Komponente entwickelt. DistriCT wird in der automatischen Auswertung dafür benutzt, um bei Überschreiten eines gewissen Schwellwertes innerhalb der Distanzfunktion den aktuellen Status der Simulatoren zu speichern. Dieser gespeicherte Status wird genutzt, um die kritischen Situation zu analysieren. (vgl. Schweigert, et al., 2014)

5 Komponenten

In diesem Kapitel soll einmal näher auf die Meilensteine eingegangen werden, die wir auf dem Weg zur intelligenten Schiffssimulation als wichtig und nötig angesehen haben. Hierbei wurde sich an den aufgestellten User-Stories orientiert, sodass mit Bearbeitung der User-Stories auch die Meilensteine als abgeschlossen angesehen werden konnten. Nachfolgend werden die einzelnen Meilensteine einmal einzeln vorgestellt.

5.1 Seekartenobjekte

Meilenstein: Benötigte Objekte können aus den Seekarten ausgelesen werden

Der erste Meilenstein behandelte das Auslesen von Objekten aus Seekartendaten. Dabei sollte vor allem Wert auf die Identifizierung der Features/Objekte und auf den eigentlichen Zugriff auf die Seekarten gelegt werden. Wichtig hierbei ist, sich von vornherein Gedanken darüber zu machen, welche Objekte ausgelesen werden sollen.

- Was sind überhaupt Features/Objekte in den Seekartendaten?

Features und/oder Objekte in Seekartendaten sind alle möglichen Daten, die in einer Karte zu finden sind. So sind neben Küstenlinien, Bahnstrecken und Sperrgebiete auch Daten wie Tiefgang und jede einzelne Boje vorhanden. Diese dienen den Kapitänen einerseits zur Orientierung, dazu gehört natürlich auch ein Leuchtturm, der auch in der Karte gespeichert ist, und andererseits zur Einhaltung von Regeln. Nun sollte herausgefiltert werden, welche dieser Daten wichtig sind, um eine reibungslose Fahrt zu gewährleisten. Dabei muss natürlich hauptsächlich auf die Hindernisse, weil auch Wracks und Ölplattformen angegeben sind, und auf Objekte auf See geachtet werden. Dabei muss dann speziell nach Objekten gefiltert werden, welche physikalisch überfahrbar sind, welche nicht physikalisch überfahrbar sind und welche für das Einhalten der Regeln wichtig sind. Sollte diese Unterteilung getätigt sein, muss natürlich auf die Seekartendaten zugegriffen werden.

- Wie wird auf die Seekarten zugegriffen?

Da die Seekartendaten in digitaler Form noch nicht öffentlich abrufbar sind, wird der Zugriff über einen Chartserver, welcher im OFFIS bereitgestellt wird, ermöglicht. Auf diesem Chartserver sind Seekartendaten zu der deutschen Bucht vorhanden. Dies muss über einen bestimmten Zugriff angepeilt werden. Über einen implementierten GMLLoader und einen GMLReader wird auf den Chartserver und die GML-Datei, also eine Datei, die in der Geography Markup Language verfasst wurde, zugegriffen. Diese GML-Datei beinhaltet hauptsächlich raumbezogene Objekte zu einem, durch Koordinaten angegebenen, Gebiet. Nachdem die Datei vom Chartserver gezogen wurde, wird sie lokal angelegt, sodass darauf zugegriffen werden kann, sobald der Chartserver ausfallen und ausgeschaltet werden sollte. Lokal angelegt werden nun speziell über den GMLReader die GML-Datei ausgelesen.

Ein Beispiel wird in der Abbildung 3 dargestellt.

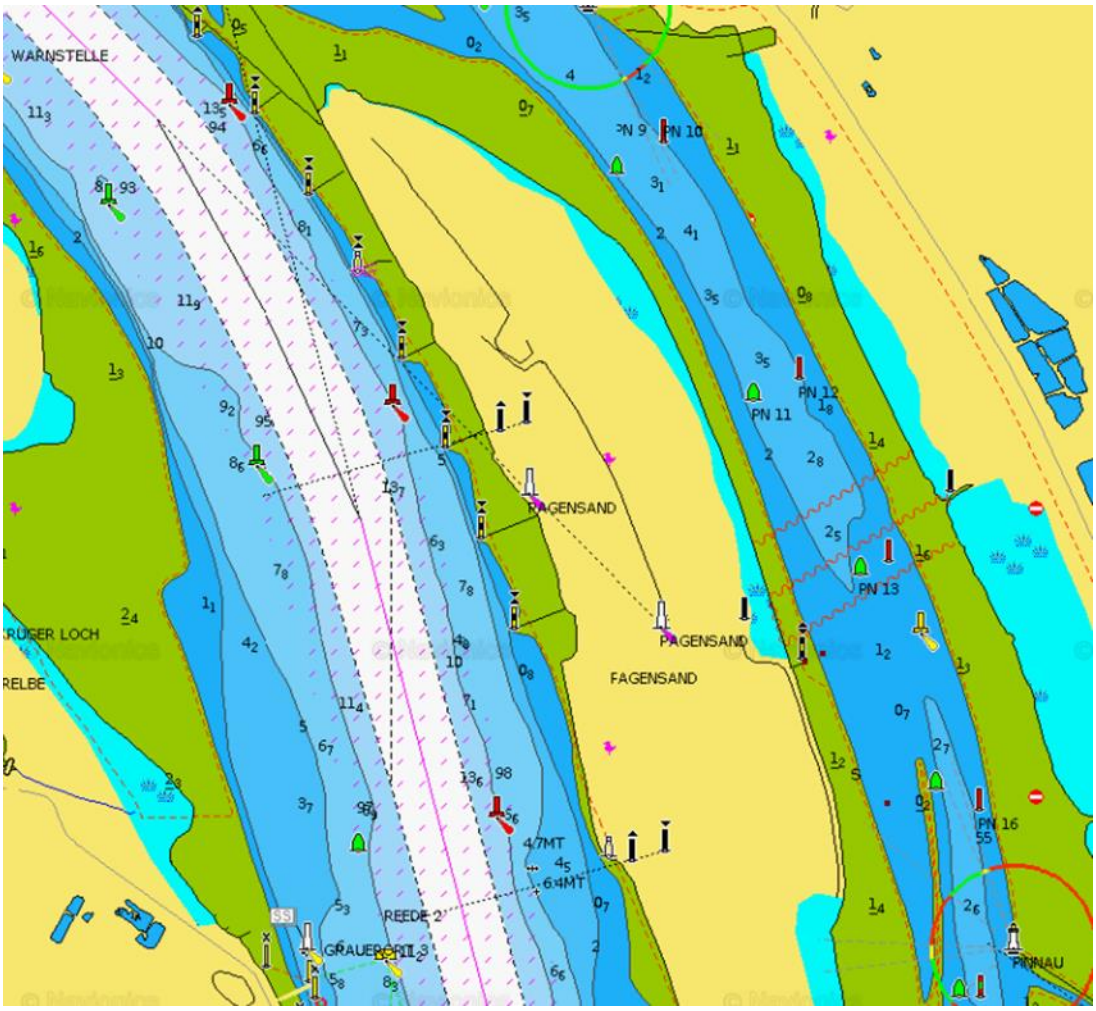


Abbildung 3: Seekarte von Helgoland (http://www.openseamap.org/index.php?id=openseamap&no_cache=1)*

- Was wird genau ausgelesen?

Nachdem die GML-Datei lokal vorliegt, wird diese nach bestimmten Objekten gefiltert. Darunter fallen z.B. Bojen, Offshore Gebiete, Wracks, Leuchtfeuer und Erhebungen im Wasser. Diese Objekte, die in Abbildung 4 dargestellt werden, sind überwiegend so genannte Nogos, welche physikalisch von einem Schiff nicht überfahrbar sind. Folgend werden die bis dato erkannten und ausgefilterten Objekte dargestellt, welche an das erstellte Grid übergeben werden.

```
staticObstacleCharacteristics.add(BeaconFeature.class);
staticObstacleCharacteristics.add(BridgeFeature.class);
staticObstacleCharacteristics.add(BuoyFeature.class);
staticObstacleCharacteristics.add(HulkFeature.class);
staticObstacleCharacteristics.add(LightFloatFeature.class);
staticObstacleCharacteristics.add(LightVesselFeature.class);
staticObstacleCharacteristics.add(ObstructionFeature.class);
staticObstacleCharacteristics.add(OffshorePlatformFeature.class);
staticObstacleCharacteristics.add(PileFeature.class);
staticObstacleCharacteristics.add(SingleBuildingFeature.class);
staticObstacleCharacteristics.add(WreckFeature.class);

staticObstacleFeatures.add(CoastLineFeature.class);
staticObstacleFeatures.add(DepthAreaFeature.class);
staticObstacleFeatures.add(DepthContourFeature.class);
staticObstacleFeatures.add(DumpingGroundFeature.class);
staticObstacleFeatures.add(FortifiedStructureFeature.class);
staticObstacleFeatures.add(IceAreaFeature.class);
staticObstacleFeatures.add(MilitaryPracticeAreaFeature.class);
staticObstacleFeatures.add(NewObjectFeature.class);
staticObstacleFeatures.add(OilBarrierFeature.class);
staticObstacleFeatures.add(PontoonFeature.class);
staticObstacleFeatures.add(PylonBridgeSupportFeature.class);
staticObstacleFeatures.add(RescueStationFeature.class);
staticObstacleFeatures.add(RiverBankFeature.class);
staticObstacleFeatures.add(SoundingFeature.class);
```

Abbildung 4: Liste der ausgelesenen Nogos aus den Seekarten

Die nun ausgelesenen Daten werden an den entwickelten Quadtree zur Speicherung und Auslagerung übergeben. Auf diesen Quadtree können alle erstellten Provider zugreifen, sodass jeder die nötigen Daten holen und diese dann verarbeiten kann.

- Was ist ein Quadtree?

Ein Quadtree, der in Abbildung 5 gezeigt wird, ist in der Regel eine baumartige Datenstruktur, welche zur effizienten Speicherung von Objekten und/oder Rasterdaten dient. Diese werden dann in einer räumlichen Datenbank gespeichert. Der Quadtree legt ein homogenes Feld über die angegebene GML-Datei und teilt somit die Datei in zusammenhängende, homogene Rasterzellen. Der Quadtree heißt Quadtree, da die aufgelegten Felder immer geviertelt werden. Somit bekommt jedes Viertel wieder die Option sich zu vierteln, sodass versucht wird, dass alle Quadranten möglichst gleich viele Attribute/Objekte enthalten. Dementsprechend wird der

Quadtree geladen und abhängig von der Anzahl der Objekte unterteilt, bzw. angelegt. Zusätzlich kann auf diese Weise eine einfache und übersichtliche Baumstruktur erstellt werden. Im Folgenden wird eine einfache Unterteilung dargestellt.

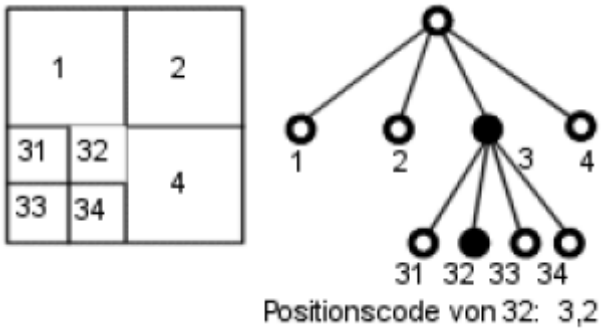


Abbildung 5: Quadtree (http://www.geoinformatik.uni-rostock.de/einzel.asp?ID=1410)

Zusätzlich ermöglicht die räumliche Aufteilung eine Indizierung in der Baumstruktur, die eine räumliche Anfrage ermöglicht, sodass Objekte mit ihren Koordinaten angegeben werden können.

5.2 Erstellen der Agentenarchitektur

In der Simulation HAGGIS wird die graphische Oberfläche durch eine, in dem Projekt integrierte, „Generate Java Code“ Funktion erstellt. Diese jgc wird durch eine TUML-Datei erstellt.

Eine Erweiterung der TUML-Datei für eine Verbesserung der graphischen Oberfläche von HAGGIS kann über JPanels erstellt werden, in welchen definiert wird, wie ein Panel und somit die Oberfläche gestaltet wird.

In einer TUML-Datei werden eigene Strukturen generiert, die auf Basis der in der TUML-Datei implementierten Struktur erstellt werden. In dieser werden die Pakete des benötigten Projekts und die benötigten Klassen benannt und schließlich generiert. Zusätzlich werden Abhängigkeiten in der TUML-Datei dargestellt, die dann für die Klassen/Interfaces übernommen werden. Nach Anpassung der TUML-Datei, wird aus der TUML-Datei eine Java Generate Code- Datei generiert. In dieser können Anpassungen bezüglich der Interfaces, der Implementierung und der Ordneranpassung getätigt werden. So kann z.B. ein Prefix oder Suffix bei den Interfaces eingefügt werden. Nach Angabe dieser können durch Rechtsklick-> eMir-> Generate Java Code die Pakete und Klassen erstellt werden. Diese haben automatisch Abhängigkeiten zu ihren Interfaces und zu den benötigten Klassen, welche in der TUML-Datei angegeben wurde.

Beim Erstellen der TUML-Datei für den ISS-Ordner wurde wie folgt vorgegangen:

Manuelle Erstellung einer TUML-Datei anhand der MTSS.tuml. Dabei enthält die ISS.tuml Imports aus den erforderlichen schon erstellten X .tuml. Zusätzlich müssen Pakete, Klassen und Abstrakte Klassen definiert werden. Des Weiteren wird die Paketstruktur durch die ISS.tuml bestimmt und erstellt somit auch Interfaces zu den erstellten Klassen. Danach wird aus der ISS.tuml eine Java Generate Code ISS.jgc generiert. Dabei entsteht folgende Ordnerstruktur: In dem angegebenen Paket wird ein Ordner erstellt (impl) in dem die class Datei liegt, eine Ebene darüber wird ein Interface zu der Klasse erstellt. Diese Struktur ist vorgegeben und kann nicht geändert werden, wodurch neue Klassen über die ISS.tuml generiert werden müssen.

Danach muss die ISS.tuml in HAGGIS integriert werden. Dies muss manuell über die HAGGISEXTENSION.java in dem Projekt eMIR-EPD in dem HAGGISExtensions Ordner gemacht werden. Dort muss die ISSModel.java initiiert werden, über „ISSModel.init();“. Als letztes muss in der pom.xml die Dependancy angepasst werden. Hinzuzufügen ist noch, dass das ISS-Projekt ab der Integrierung in das HAGGIS Projekt ausschließlich das Erzeugen von Klassen und Paketen über die ISS.tuml erlaubt und zulässt.

Bei der Integrierung des Projektes in HAGGIS sind mehrere Probleme aufgetreten. Diese Probleme sind allerdings hauptsächlich manuell gelöst worden und sind auf die schon vorher festgelegte Struktur des ISS-Projektes zurückzuführen.

So wurden z.B. nicht alle „extends“ und „implements“ beim Generieren der Klassen übernommen und es musste eine manuelle Integrierung vor allem der Abhängigkeiten zu den UObjects vorgenommen werden. Da ohne die Vererbung auf die UObjects die Klassen nicht erstellt wurden, fehlten damit auch die Verweise und das manuelle Einfügen hat somit zwei Probleme gleichzeitig gelöst.

Ein weiteres Problem bestand bei den Interfaces. Da diese durch die ISS.tuml automatisch erstellt wurden, werden sie auch direkt mit in die Ordnerstruktur mit aufgenommen. Dies führt allerdings dazu, dass die Pakete und auch das Gesamtprojekt sehr umfangreich werden, auch wenn die einzelnen Interfaces an sich leer sind.

Ein abschließendes Problem besteht in der Oberfläche von HAGGIS. Dort wird die graphische Oberfläche aller in der TUML-Datei angegebenen Elemente über Panels dargestellt. Allerdings

werden diese Panels mittels der TUML-Datei nicht erstellt, sodass diese manuell für jeden Provider noch erstellt werden müssen. Dies dient dann neben der Auswählbarkeit in HAGGIS auch der Darstellung und Abänderbarkeit im gesamten ISSProjekt. Somit können darüber Parameter eingestellt und angezeigt werden, was unter anderem dem Nutzer der Simulation dienlich ist.

5.2.1 Anlegen eines Agenten

Das Ziel der Projektgruppe für diesen Meilenstein war es, dass der Nutzer innerhalb der Simulation einen Agenten anlegen kann und diesem ein eigenes Verhalten zuweisen kann. Deshalb sollte es dem Nutzer ermöglicht werden, in der Oberfläche einen Agenten zu erstellen und diesem ein Szenario zuzuordnen. Das Verhalten sollte innerhalb der Simulation auswählbar sein und mittels Schieberegler angepasst werden können. Innerhalb dieses Meilensteins beschloss die Projektgruppe, auch die Architektur hinter den Agenten festzulegen. Hierfür soll ein Modell entwickelt werden, welches das Gesamtverhalten eines Agenten abdeckt. Hierbei ist die Architektur so aufgebaut, dass sich das Gesamtverhalten aus einzelnen Teilverhalten zusammensetzt. Innerhalb des Individualverhaltens der Agenten können feste Stereotypen von Kapitänen ausgewählt werden, deren Fahrverhalten durch Festlegung ihrer einzelnen Parameter von der Projektgruppe vorbestimmt ist. Bei den Stereotypen kann hierbei z.B. zwischen dem Standardkapitän, dem erfahrenen oder dem aggressiven Kapitän ausgewählt werden. Des Weiteren hat der Nutzer die Möglichkeit über die eingebauten Schieberegler das Verhalten über mehrere vorgegebene Regler anzupassen.

Im folgenden Abschnitt soll einmal auf die einzelnen Komponenten, die das Gesamtverhalten eines Agenten bestimmen, eingegangen werden und wie das Zusammenspiel zwischen diesen Komponenten erfolgt. Zum einem wird das Klassendiagramm in Abbildung 6 gezeigt und zum anderen das entsprechende Sequenzdiagramm in Abbildung 7 dargestellt. Das MainBehaviour enthält von dem CruiseProvider Informationen über die Route, die für den Agenten ausgewählt wurde und über den aktuellen Wegpunkt, der angesteuert wird. Zudem wurde die Klasse BehaviourComponent angelegt, die das Verhalten der einzelnen Agenten steuert. Innerhalb der BehaviourComponent werden die Kommandos zur Steuerung des Schiffes berechnet. Hierbei unterstützen die drei Provider: UtilityScoreProvider, HeadingProvider sowie der VelocityProvider. Im VelocityProvider wird die Geschwindigkeit vorgegeben, die der Agent beim Abfahren einer Route wählt. Innerhalb des HeadingProviders wird das Heading des Agenten vorgegeben.

Die Struktur eines Agenten basiert auf der Idee der „Utility-based Artifical Intelligence“. Innerhalb der Struktur des Agenten wird jeder Komponente mittels des UtilityScoreProviders ein Utility Score zugewiesen, der die Handlungsrelevanz im aktuellen Moment widerspiegelt. Diese Idee der „Utility-based Artifical Intelligence“ wurde sich aus der Computerspiel-Branche abgeschaut. Hierbei wird nicht zwischen endgültigen Zuständen gewechselt, sondern stattdessen laufend alle möglichen Aktionen innerhalb einer Situation betrachtet. Anschließend wird für jede mögliche Aktion der Nutzen berechnet, die diese dem Agenten einbringt und der Agent entscheidet sich infolgedessen für die Handlung, die ihm den größten Nutzen einbringt. Zum Beweis, dass dies ein realistisches Szenario darstellt, welches unter rationalen Aspekten vorkommt, kann ein Fall aus der Medizin angeführt werden. So wird ein Patient, der gesund ist, sich mit einer Wahrscheinlichkeit von 0% ein Medikament kaufen, sodass unsererseits angenommen werden kann, dass die Handlungsalternative eines Kaufes ausgeschlossen werden kann. Diese Denkweise haben wir uns zu Nutzen gemacht und auf den maritimen Verkehr übertragen. In unserem Fall bedeutet dies, dass wir drei Situationen auf See haben können. Entweder wir können ungehindert der Route folgen oder es kommt zu einer Ausweichsituation mit einem statischen oder dynamischen Hindernis. Hierfür wurde unter dem HeadingProvider der RouteFollowHeadingProvider, der StaticObstacleAvoidanceHeadingProvider und der DynamicObstacleAvoidanceHeadingProvider angelegt. Für die einzelnen Provider wird in der Folge jeweils laufend ein UtilityScore berechnet. Für den RouteFollowHeadingProvider wurde hierfür ein konstanter Wert von 10 angegeben. Solange nämlich kein Hindernis auf unserer Route auftaucht, sollte der Agent strikt seiner Route folgen. Solange in der Umgebung des Agenten keine Hindernisse auftauchen, sind die UtilityScores für den StaticObstacleAvoidanceHeadingOrovider sowie den DynamicObstacleAvoidanceHeadingProvider bei 0 angesiedelt. Umso näher der Agent einem Hindernis kommt, desto mehr steigt der UtilityScore für die jeweiligen Ausweichsituationen an. Für das dynamische Ausweichen nehmen wir für den Moment des letzten Augenblicks einen UtilityScore von 100 an und stufen diesen dann linear ab. Sobald also der UtilityScore für das dynamische Ausweichen über den konstanten Wert des RouteFollowHeadingProviders angestiegen ist, wird in den DynamicObstacleAvoidanceHeadingProvider übergegangen und dem Hindernis ausgewichen. Ebenso verhält es sich für das statische Ausweichen, bei dem der UtilityScore auch linear ansteigt, umso näher man dem statischen Hindernis kommt.

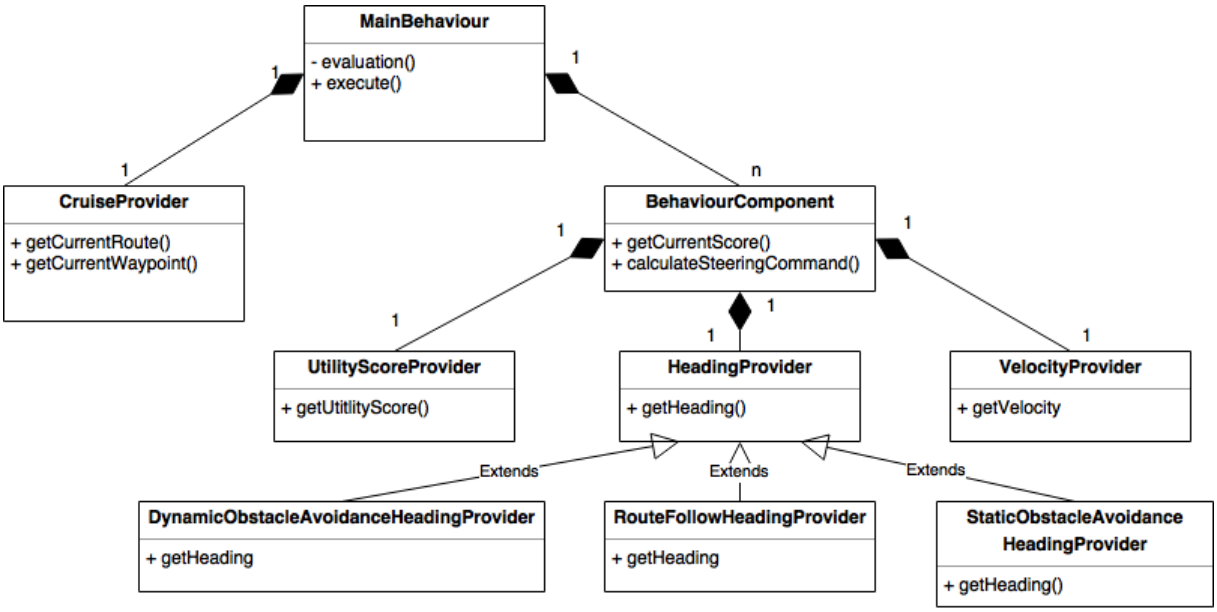


Abbildung 6: Klassendiagramm Utility-AI

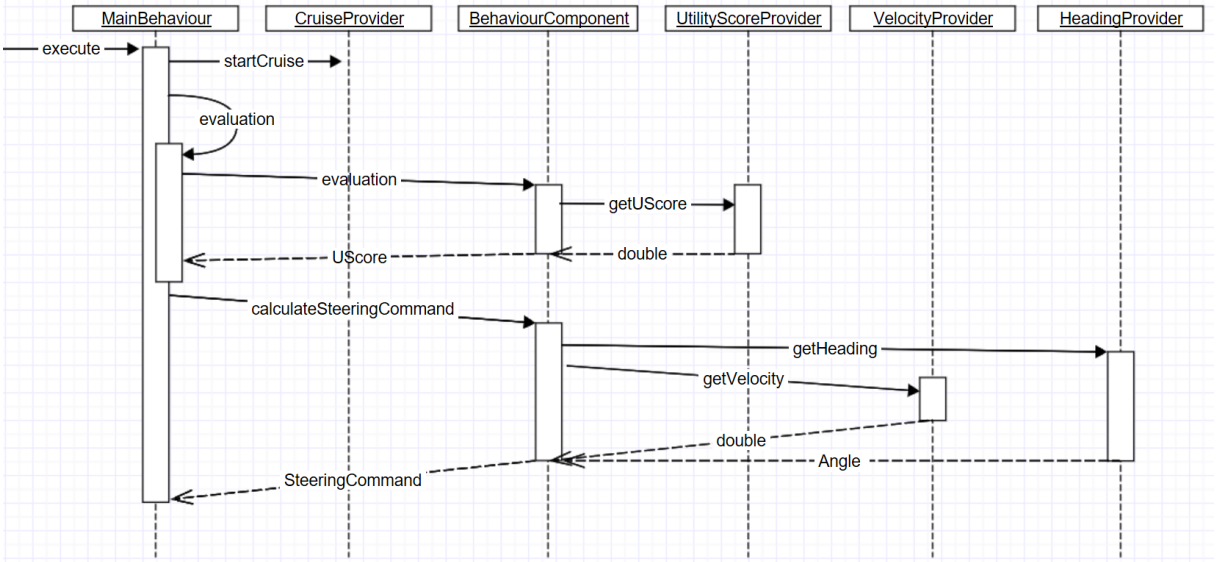


Abbildung 7: Sequenzdiagramm Utility-AI

Für unsere Agentenarchitektur ist die Utility-AI daher von Relevanz, weil mittels des UtilityScores festgelegt werden kann, ab wann ein bestimmter Stereotyp an Kapitän vom RouteFollowHeadingProvider in einen Ausweich-Provider wechselt. So kann die Reichweite, ab wann die Berechnung des UtilityScores für das Ausweichen beginnt je nach Stereotyp variiert werden. Dies würde dann gewährleisten, dass der aggressive Kapitän später zum Ausweich-Manöver ansetzt.

5.3 Routenkanten verfolgen

Ziel dieser Teilaufgabe ist es, es den Agenten zu ermöglichen, sich eine Route zu generieren. Dem Agenten sollen dafür lediglich ein Start- und ein Zielpunkt mitgegeben werden, die benötigte Route sucht er sich anschließend selbstständig. Das ist auf Basis der Funktionen von HAGGIS bereits möglich. Die Schiffe fahren die Routen exakt und ohne Abweichung ab. Für eine realitätsnahe Simulation soll eine Funktion eingebaut werden, die es ermöglicht, dass die Schiffe einen *Aufperleffekt* umgehen und in einem gewissen Korridor von der vorgegebenen Route abweichen. Außerdem muss berücksichtigt werden, dass später implementierte Verhaltensmuster der Agenten die Routen variabel abfahren können.

Für das Generieren der Routen wird der in HAGGIS integrierte Routengraph verwendet. Dieser berechnet wie gewünscht eine Route zwischen Start- und Zielpunkt. Alternativ kann einem bestehenden Schiff auch nur ein Zielpunkt zugewiesen werden. Der Routengraph errechnet anschließend automatisch eine Route von der Position des Schiffes zum Zielpunkt. Die generierten Routen werden als valide angenommen und nicht näher geprüft.

Ergänzend wurde eine Funktion implementiert, die analysiert, welcher Routenpunkt dem Schiff am nächsten ist. Wurde eine Route berechnet, deren Startpunkt in einer Position liegt, zu deren Erreichung das Schiff eine Kehrtwende oder ähnliches durchführen muss, so wird der optimale, alternative Routenpunkt gesucht und angefahren, dieses wird in der Abbildung 8 dargestellt.

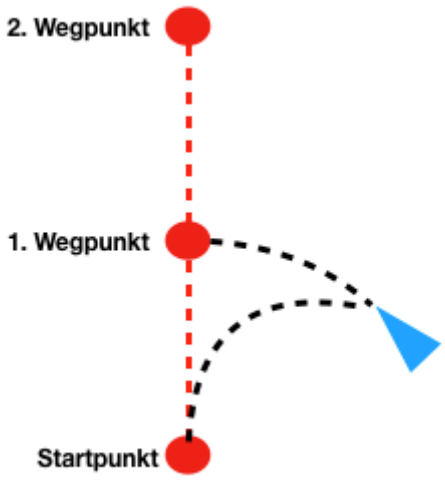


Abbildung 8:Anfahren des ersten Routenpunktes

Um Realitätsnähe und eine Varianz bei dem Abfahren der Routen zu gewährleisten, wird jedem Agenten ein Verhalten zugewiesen. Die Verhalten beeinflussen die Art und Weise wie ein Agent/Schiff die Route abfährt. Nähere Informationen zu den Verhaltensmustern der Agenten

sind unter Kapitel 7, der Definition der Kapitänstypen, zu finden. Außerdem wurden Parameter geschaffen, mit deren Hilfe das Abfahren der Bahnen und Routen mit einer gewissen Varianz ermöglicht wird. So können zum Beispiel für jeden Agenten die Geschwindigkeit, der Ruderwinkel, der Abstand, bis der Agent auf einen kommenden Wegpunkt reagiert oder ähnliches variabel gesetzt werden. Ebenfalls kann zu Beginn festgelegt werden, wie sich das Schiff verhält, wenn es den Zielpunkt seiner Route erreicht hat. So kann es beispielsweise ankern oder zum Startpunkt zurückkehren.

Um den *Aufperleffekt* zu vermeiden wurde eine Funktion namens "Deviation" entwickelt. Diese Funktion sorgt dafür, dass, sollten mehrere Schiffe auf der Route hintereinanderfahren, diese nicht alle exakt auf der Route fahren, sondern durchaus auch leicht versetzt zur Route unterwegs sein können.

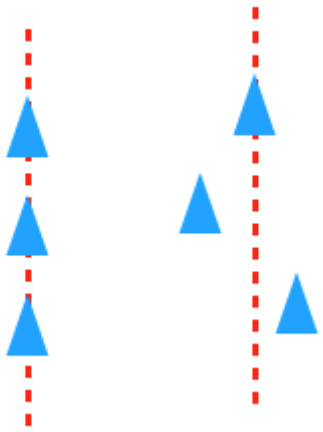


Abbildung 9: Verhinderung des Aufperleffektes

Ein wichtiger Aspekt ist, dass der Agent, sollte er aufgrund seiner Parametrisierung oder eines Ausweichmanövers, seine vorgegebene Route verlassen, immer wieder auf diese zurückkommt. Der Agent muss also aktiv erkennen, dass er die Route verlässt und entsprechende Gegenmaßnahmen einleiten. Dieses wird in der Abbildung 9 abgebildet. Das Erreichen des Zielpunktes muss gewährleistet sein. Eine Ausnahme besteht natürlich, sollte das Ziel auf Land oder in einem unzugänglichen Gebiet liegen.

6 Hinderniserkennung

Im Folgenden wird die Hinderniserkennung und Behandlung beschrieben. Dabei wird zunächst auf den zugrundeliegenden NoGo-Solver eingegangen und später auf die statischen und dynamischen Hindernisse.

6.1 NoGo-Solver

Der NoGo-Solver Server ist ein vom ICD MTCAS bereitgestellter Dienst, welcher in der Version 0.1 vorliegt und am OFFIS e.V. – Institute for IT entwickelt wurde. Der NoGo-Solver wird entweder lokal über die Konsole gestartet oder liegt als eigentlicher Server im OFFIS an. Als Zugriffsport wird der Port 8025 verwendet und es muss ein Websocket implementiert werden, welcher den Austausch von Nachrichten mit dem Server regelt. Der NoGo-Solver verarbeitet Angaben von Koordinaten und Schiffsdaten. Diese müssen wie in der Abbildung 10 zu sehen, eingestellt werden.


```
{
   "action":"GET",
   "contentType":"NoGoInformation",
   "content":{
      "draft":4.6,
      "width":1.1,
      "length":7.6,
      "height":0,
      "minLat":53.889312,
      "maxLat":53.973399,
      "minLon":8.718779,
      "maxLon":8.878081
   }
}
```


Abbildung 10: Verarbeitung der Koordinaten vom NoGo-Solver

Dabei werden zuerst die Daten über das Schiff festgelegt und danach die Koordinaten. Dabei wird darauf geachtet, dass „minLat" kleiner/gleich „maxLat" ist und „minLon" kleiner/gleich „maxLon". Ansonsten kann der NoGo-Solver diese Daten nicht verarbeiten. Dies stellt auch eine Bereichsanfrage dar, in welchen Daten ausgelesen werden. Als Antwort bekommt der Nutzer, in der Abbildung 11, visualisierte Rückmeldung.

```
{
   "action":"POST",
   "contentType":"NoGoSolverInformation",
   "content":{
      "dangers":[
         "POINT (53.90247704 8.72951302)",
         "POINT (53.927982 8.764883981)",
         "LINESTRING ((53.99999998 8.684618038, 53.99999998 8.692982004, 53
.99999998 8.692982004, 53.99999998 8.698713968, 53.99999998 8.698713968, 53
.99999998 8.766823972))"
      ],
      "nogos":[
         "POINT (53.90247704 8.72951302)",
         "POINT (53.927982 8.764883981)",
         "POLYGON ((53.99999998 8.684618038, 53.99999998 8.692982004, 53.99
999998 8.692982004, 53.99999998 8.698713968, 53.99999998 8.698713968, 53.99
999998 8.766823972))"
      ],
      "missingData":[
         "POINT (53.90247704 8.72951302)",
         "POINT (53.927982 8.764883981)",
         "POLYGON ((53.99999998 8.684618038, 53.99999998 8.692982004, 53.99
999998 8.692982004, 53.99999998 8.698713968, 53.99999998 8.698713968, 53.99
999998 8.766823972)"
      ]
   }
}
```


Abbildung 11: Antwort des NoGo-Solvers

Hierbei wird beim NoGo-Solver in drei Bereiche unterschieden. Die „dangers", die „nogos" und die „missingData". Dabei stehen die „dangers" für die Flächen oder Objekte, die prinzipiell gefährlich sind, bei denen das Schiff aber physikalisch in der Lage ist weiter zu fahren. Anders dagegen bei den „nogos". Diese stellen Passagen, Flächen, Linien oder Punkte dar, bei denen das Schiff auf gar keinen Fall weiterfahren bzw. die das Schiff nicht überqueren darf. Die „missingData" sind genau das, was vermutet wird. Hierbei fehlen einfach grundlegende Daten in den Seekarten und können somit nicht vom NoGo-Solver verwertet werden.

Zur Erkennung der statischen Hindernisse wird der NoGo-Solver-Service in der Version 0.2 verwendet, da dieser stabiler läuft als die Version 0.3. In der Klasse NoGoConnector wird zunächst ein WebsocketClientEndpoint erstellt, mit dem der Server über eine Adresse erreicht werden kann. Daraufhin wird geschaut, ob sich in dem angegebenen Bereich Polygone, Punkte oder LineStrings befinden. Falls es sich dabei um NoGo-Objekte handelt, werden diese zu einer Liste hinzugefügt. Falls in dem StaticObstacleAvoidanceBehaviourComponent angegeben ist,

dass der NoGoSolver verwendet werden soll, wird in dem StaticObstacleAvoidanceNogoSolverProvider die updateNoGoSolver() Methode ausgeführt. In dieser wird geprüft, ob sich der zu betrachtende Bereich verändert hat. Falls dies der Fall ist, wird eine neue Anfrage an den NoGoSolver gesendet.

6.2 Statische Hindernisse

Das Ausweichen von statischen Hindernissen wird durch das Zusammenwirken von mehreren Komponenten erreicht. Als Basis für das Ausweichen werden Daten aus dem NoGoSolver gezogen. Mithilfe dieser Daten wird für jedes statische Hindernis ein UtilityScore berechnet, der die Priorität für das Behandeln der statischen Hindernisse bestimmt. Anschließend wird mit einem A*-Algorithmus der Weg um das statische Hindernis berechnet.

6.2.1 UtilityScore

Damit in dem A* auf den UtilityScore für das jeweilige statische Hindernis zugegriffen werden kann, wurde ein eigener Datentyp definiert. Der Datentyp besteht zum einen aus einem UObject, das sowohl eine S57 Characteristic als auch ein EFeature sein kann, und zum anderen aus einem Integer Wert für den UtilityScore. StaticObstacles werden absteigend sortiert.

In der Methode getUtilityScore() wird der UtilityScore für statische Hindernisse berechnet. Zunächst wird für das aktuelle Schiff eine Geometrie angelegt. Mithilfe der sightRange() Methode in den GeometryUtils wird über die Parameter mCorridor und rangeOfVision die zugehörige Geometrie berechnet. Diese Geometrie ist ein Dreieck, welches in Richtung des Headings vor dem Schiff angegeben wird. Zusammen mit der berechneten Geometrie werden in dem StaticObstacleAvoidanceBehaviourComponent die statischen Objekte für den Sichtbereich in einer Liste gespeichert. Anschließend wird für jedes Hindernis bzw. jede Geometrie geschaut, ob sich diese mit der Geometrie für die Sichtweite des Schiffes überschneidet. Falls ja, wird geschaut, ob die Distanz zu dem Hindernis größer ist, als zu der bisher weitesten Distanz in einem StaticObstacle. Der UtilityScore berechnet sich dann aus dem Produkt von dem Multiplikator m= (-100 / Sichtdistanz in Metern) und der Distanz. Auf dieses Produkt werden zudem 100 addiert.

$$\left(\left(\frac{-100}{sightRangeInMeter}\right) * distance\right) + 100$$

6.2.2 A*-Algorithmus

Um die statischen Hindernisse zu erkennen, werden zunächst alle, sich in der Seekarte befindlichen, Objekte in einem Umkreis ermittelt. Diese Objekte werden dann, anhand ihrer Geoinformationen aus dem Quadtree, in eine Liste übergeben. Die Zellen für die Wegfindung des A* werden nur erzeugt, wenn diese auch benötigt werden. Dieses On-the-fly-Verfahren ermöglicht Speicheroptimierungen. In diesem Schritt wird das aktuelle Umfeld der Schiffe mit einem Grid überzogen. Zellen des Grids, die Objekte aus der Liste der Hindernisse beinhalten, werden für das Befahren gesperrt. Diese gesperrten Flächen sind in der Abbildung 11 rot schattiert. Im Folgenden wird mittels eines angepassten A*-Algorithmus, der die mit Hindernissen belegten Zellen berücksichtigt, eine mögliche Bahn ermittelt. Der Zielpunkt des A* in dem Grid wird mit dem Schnittpunkt der Route und dem Grid berechnet. Dem A* liegen zum einen eine Heuristik und zum anderen eine Neigbourhood-Funktion zugrunde. Diese werden im Folgenden näher erläutert:

Mit der zugrundeliegenden Heuristik wird ein kurzer fahrbarer Weg berechnet, der nicht optimal sein muss, da die kürzesten Entfernungen überschätzt werden. Das heißt, dass bei der Heuristik nicht der direkte kürzeste Weg errechnet wird. So setzt sich die Heuristik nämlich aus der Summe von zwei Entfernungen zur Zielzelle zusammen. Die eine Entfernung ist die Targetline, welche das Heading Richtung Zielzelle darstellt und die andere Entfernung ist die Orthogonale zur Targetline. Zellen, die sich hinter dem Schiff befinden, werden bestraft, indem die Heuristik künstlich mit einer Variable hochgesetzt wird.

Ein weiterer zugrundeliegender Teil des A* ist die Neigbourhood-Funktion. Mit dieser Funktion werden die Zellen evaluiert, die ausgehend von der aktuellen Zelle des A-Sterns erreicht werden können. Mithilfe der Angabe eines Umkreises über eine Variable werden alle Zellen unter Berücksichtigung des Dynamikmodells zur Evaluation in Betracht gezogen. Weiterführend wird geprüft ob zwischen der Ausgangszelle und der Zielzelle ein Hindernis liegt. Durch die angewandte Selektion der Zellen wurde der A* optimiert.

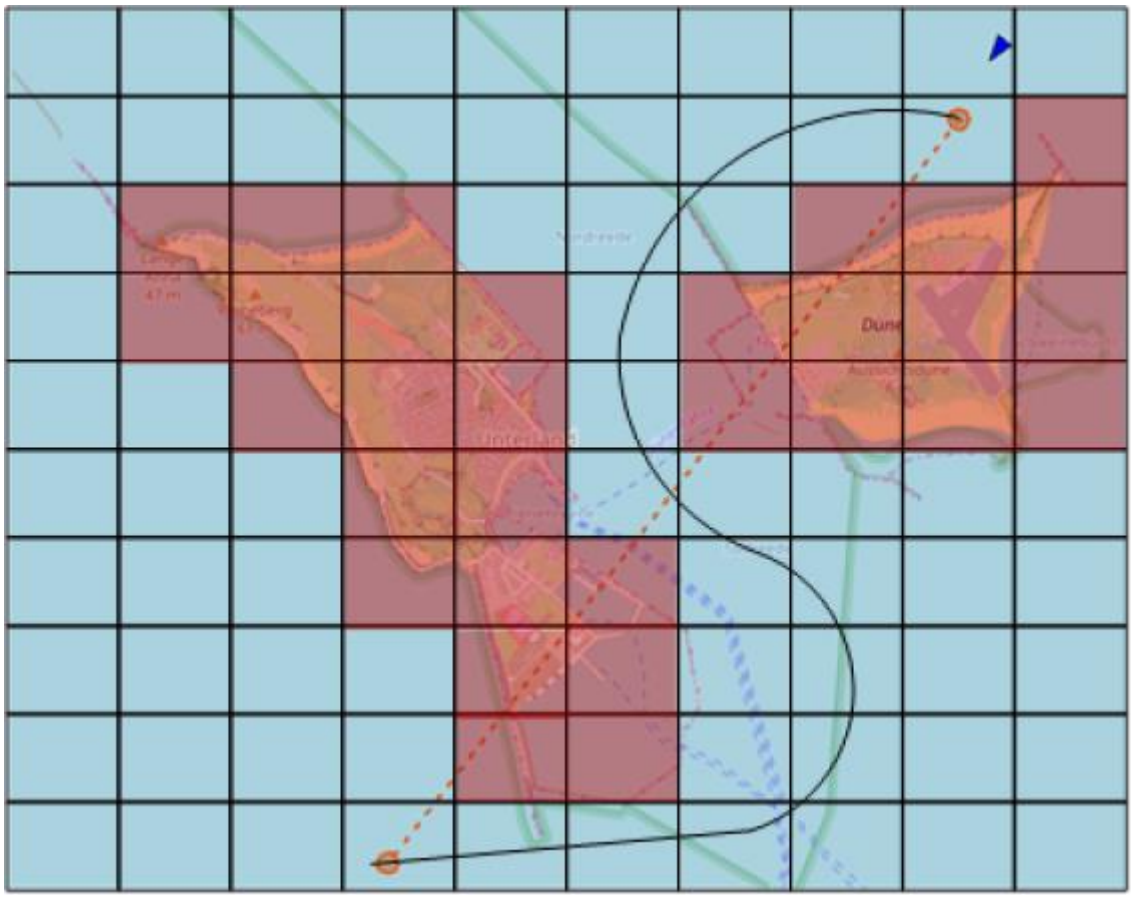


Abbildung 12: A-Grid*

In der projizierte Abbildung 12 sind die blockierten Zellen, die rot eingefärbt sind, zu sehen.

6.3 Anzeige des Grids auf der Oberfläche von HAGGIS zur Verdeutlichung von statistischen Hindernissen

Nachdem die Behandlung von statischen Hindernissen abgearbeitet wurde, gab es bezüglich der bildlichen Darstellung in HAGGIS einige Unstimmigkeiten. Da wir bis dato nur die Schiffe, die Routen und unseren berechneten Kurs angeben konnten, aber nicht die statischen Hindernisse, um die wir herumfahren wollten, haben wir uns entschlossen, auch diese Lücke anzugreifen. Dazu haben wir folgende Überlegungen angestrebt:

- Was soll dargestellt werden?
- Wie sollte es dargestellt werden?
- Wie setzen wir die Darstellung um?

Die erste Überlegung wurde relativ schnell abgehandelt. Da wir schon Schiffe, Routen und berechnete Routen anzeigen lassen konnten, sollten jetzt statische Hindernisse angezeigt werden. Zusätzlich sollte das integrierte Grid, in dem die statischen Hindernisse als Envelopes angegeben sind, dargestellt werden. Dies führte zu der zweiten Frage, wie dies dargestellt werden soll.

Diese Frage war etwas schwieriger zu beantworten, da noch niemand eine Idee hatte, wie die schon dargestellten Oberflächenobjekte implementiert wurden. Daher war erstmal grundlegend die Idee, dass das Grid, sollte es angezeigt werden, rote Umrandungen bekommt und dass jede einzelne Zelle im Grid bemalt wird. Zusätzlich sollten die Zellen, welche ein statisches Hindernis enthalten, geblockt und somit rot ausgefüllt werden.

Die dritte Überlegung beinhaltete den programmatischen Teil der Aufgabe. Hierbei lief es erstmal darauf hinaus, die Stellen im Code zu finden, an denen die Oberflächenelemente gezeichnet werden. Schließlich wurden wir im Paket eMIR-EPD. GenericMapView im Ordner de.emir.rcp.views.map und dem beiliegenden de.emir.ui.mv fündig. Dabei entpuppte sich die MapView.class als Hauptklasse der gezeichneten Oberflächenelemente. Hierüber konnten wir dann Provider erstellen, welche uns das Grid erstellten. Im Folgenden werde die Klassen erwähnt, welche uns das Anzeigen des Grids als Oberflächenobjekt erlauben:

- MVGridShape
  Im MVGridShape gibt es eine Methode, welche update heißt. Die update-Methode stellt fest, ob das Schiff ein Grid besitzt. Ist dieses null bricht die Methode ab. Sollte dies nicht der Fall sein und das Grid ist befüllt, wird jede einzelne Zelle an die Graphics übergeben.

- MVGridShapeProvider
  Der MVGridSchapeProvider ist der zugehörige Provider zum MVGridShape

- MVMainBehaviourIP
  Der MVMainBehaviourIP ist ein Itemprovider für unser MainBehaviour. Hierbei wird ein neues Item erstellt, welches übergeben wird.

- VesselGridShape

Das VesselGridShape beinhaltet die eigentliche Zeichenmethode. Hierbei wird in der update-Methode alles was obstacle ist im Grid eingefärbt. Also somit die Umrandung der einzelnen Zellen. Zusätzlich werden alle besetzten Zellen rot markiert bzw. befüllt, dies ist in Abbildung 10 dargestellt.

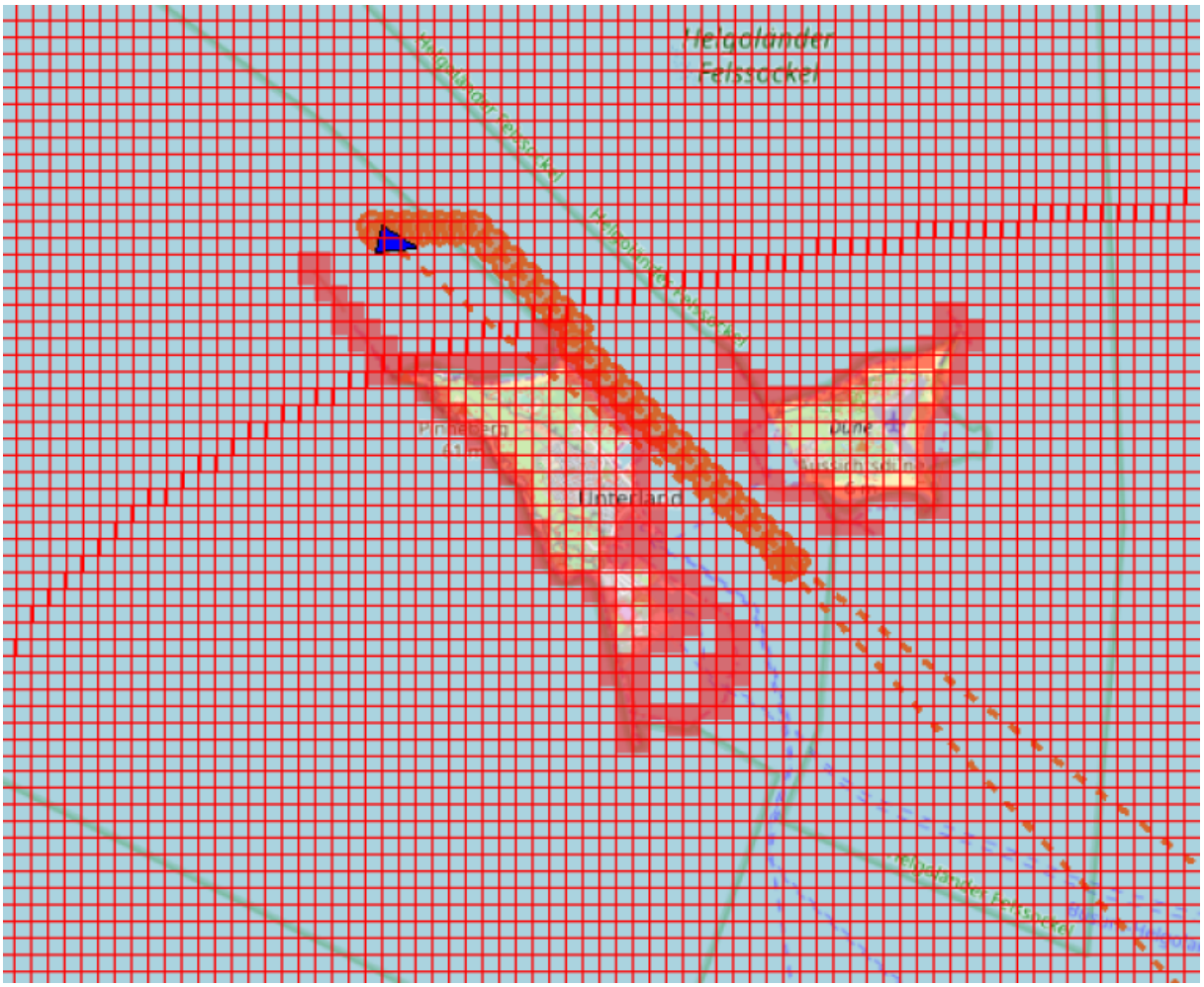


Abbildung 13: Eingefärbtes Grid

6.4 Dynamische Hindernisse

Auch das Ausweichen von dynamischen Hindernissen wird durch das Zusammenwirken von mehreren Komponenten erreicht. Zum einen ist das der UtilityScore und zum anderen der A*-Algorithmus für dynamische Hindernisse. Der UtilityScore betrachtet die umliegende Situation des Schiffes und bewertet die dynamischen Hindernisse. Der A* berechnet daraufhin den bestmöglichen Weg unter Betrachtung der Kollisionsverhütungsregeln (KVR).

6.4.1 Theoretische Grundlagen

Um auf Situationen mit anderen Schiffen reagieren zu können, müssen die Zustände zunächst erkannt werden. Dabei spielen die Bereiche des Schiffes eine entscheidende Rolle. In der Abbildung 14 wird die Einteilung der Bereiche dargestellt. Die Bereiche lauten Backbord, Steuerbord, Stern und Dead Ahead und benötigen eine unterschiedliche Erkennung und Behandlung.

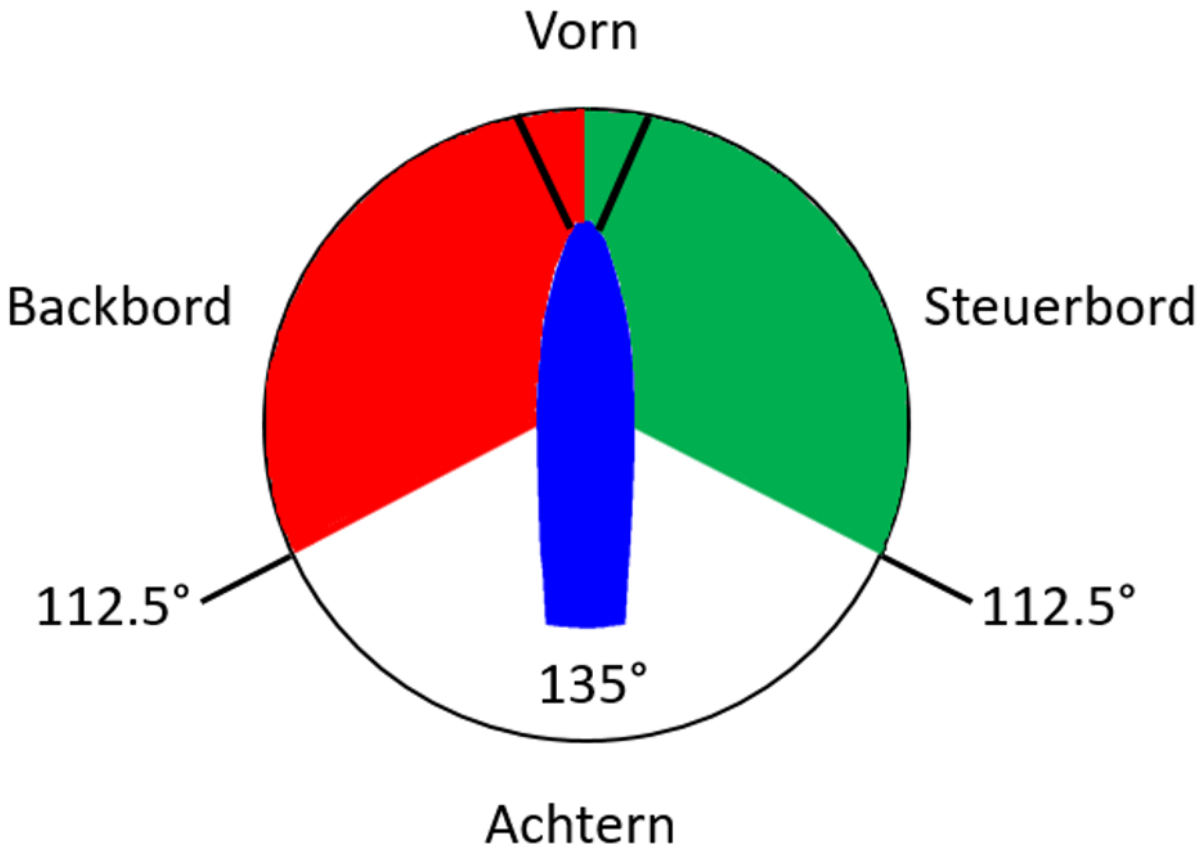


Abbildung 14: Einteilung der Situationserkennung (Vgl. https://www.boatsmartexam.com/knowledge-base/article/right-of-way-rules-boating/)

6.4.1.1 Backbord

Wenn sich ein anderes Schiff von der Backbordseite her nähert, ist das andere Schiff kurshaltepflichtig. Hier tritt dann die Regel 17 der KVR ein. Der Verkehrsteilnehmer, der kurshaltepflichtig ist, hat den Kurs und die Geschwindigkeit beizubehalten. Jedoch darf der Kurshalter ausweichen, wenn es zur Abwendung eines Zusammenstoßes dient. Dieses tritt in Kraft, wenn der andere Verkehrsteilnehmer sich nicht ordnungsgemäß an die Ausweichpflicht hält. Zudem darf das kreuzende Schiff, sofern es die Umstände zulassen, gegenüber dem anderen Maschinenfahrzeug nicht nach Backbord ändern.

6.4.1.2 Steuerbord

Nähert sich ein Maschinenfahrzeug auf der Steuerbordseite, so muss das Schiff ausweichen, welches das andere Schiff auf der Steuerbordseite erkennt. Es ist zu vermeiden, den Bug des anderen Teilnehmers zu kreuzen. Dies ist in Regel 15 der KVR niedergeschrieben.

6.4.1.3 Dead Ahead

Durch zwei entgegenkommende Schiffe oder auch zwei fast entgegenkommende Maschinenfahrzeugen, die sich entsprechend so annähern, dass die Gefahr besteht, dass sie kollidieren könnten, muss jedes Schiff seinen Kurs zur Steuerbordseite so ändern, dass sie auf der jeweiligen Backbordseite passieren können. Dieses wird in der der Regel 14 der KVR festgelegt.

6.4.2 UtilityScore

Im Folgenden wird der Aufbau für die Berechnung des UtilityScores beschrieben. In der getUtilityScore() Methode wird zunächst die Methode checkVesselForCollision() aufgerufen. Falls die Geschwindigkeit unseres Schiffes nicht null und größer 0 ist, wird mit der Iteration über alle in der Umgebung liegenden Schiffe geschaut, welche Verkehrssituation vorliegt. Die getNearbyVessels() Methode gibt eine Liste mit fremden Schiffen zurück, die sich in unserem Sichtbereich befinden. Zum Prüfen der richtigen Verkehrssituation gibt es folgende vier Methoden: overtakeAhead(), frontalAhead(), crossAhead(), standOn().

Eine OvertakeAhead Situation liegt vor, wenn die Geschwindigkeit des fremden Schiffes nicht null und größer 0 ist. Des Weiteren muss sich das fremde Schiff in dem Sichtbereich von unserem Schiff befinden und die Kurse der beiden Schiffe müssen sich aus unserer Sicht überschneiden. Des Weiteren wird geprüft, ob die Geschwindigkeit unseres Schiffes größer ist als die des Fremden. Der entscheidende Faktor ist, ob unser Schiff das fremde Schiff in den linken oder rechten 112,5° aus Fahrtrichtung treffen würde. Falls diese Situation vorliegt, wird das Enum in DynamicAvoidanceSituation entweder auf OVERTAKE_LEFT oder OVERTAKE_RIGHT gesetzt. Anschließend wird in der calculateScore() Methode der UtilityScore für das Schiff in einer Overtake Situation berechnet. Falls dieser der höchste unter den bestehenden UtilityScores ist, überschreibt der neue den alten Wert. Abschließend wird dem DynamicObstacleAvoidanceBehaviourComponent ein DynamicObstacle mit den zugehörigen Werten hinzugefügt.

Eine FrontalAhead bzw. HeadOn Situation liegt vor, wenn die Geschwindigkeit des fremden Schiffes nicht null und größer 0 ist. Hinzu kommt, dass sich das fremde Schiff in dem Sichtbereich von unserem Schiff befinden muss und die Kurse sich so überschneiden müssen, dass diese sich in einem Schwellbereich aus der Sicht unseres Schiffes befinden. Anschließend wird in der calculateScore() Methode der UtilityScore für das Schiff in einer HeadOn Situation berechnet. Falls dieser der höchste der bestehenden UtilityScores ist, überschreibt der neue Wert den alten Wert. Abschließend wird dem DynamicObstacleAvoidanceBehaviourComponent ein DynamicObstacle mit den zugehörigen Variablen hinzugefügt.

Die CrossAhead Situation liegt vor, wenn die Geschwindigkeit des fremden Schiffes nicht null ist und größer 0 ist. Anschließend wird sowohl für unser als auch für das fremde Schiff eine fiktionale Linie in Fahrtrichtung in Form eines LineStrings berechnet. Falls sich das fremde Schiff nicht mit 10° Abweichung in der Fahrtrichtung unseres Schiffes befindet und sich die Geometrien mit den Sichtweiten schneiden, dann wird mit der Methode calculateTCPA() in den GeometryUtils der Zeitpunkt einer Kollision berechnet. Falls die berechneten Zeiten bis zum möglichen Kollisionspunkt innerhalb eines definierten Intervalls sind, besteht eine CrossAhead Situation. Anschließend wird in der giveWay() Methode geprüft, ob unser Schiff oder das fremde Schiff ausweichpflichtig ist. Falls unser Schiff ausweichpflichtig ist, wird mit der calculateCrossingScore() Methode der UtilityScore für das fremde Schiff berechnet. Im anderen Fall wird der UtilityScore des anderen Schiffes auf 20 gesetzt. Abschließend wird dem DynamicObstacleAvoidanceBehaviourComponent ein DynamicObstacle mit den zugehörigen Variablen hinzugefügt. Der Wert des Enums wird in diesem Objekt entweder auf CROSSING oder STANDON_CROSSING gesetzt.

Es handelt sich um eine StandOn Situation, falls sich unser Schiff in dem Sichtbereich des fremden Schiffs befindet. Falls sich diese beiden Geometrien schneiden, wird anschließend geprüft, ob die Kurse unter Berücksichtigung eines Schwellwertes übereinstimmen. Abschließend wird dem DynamicObstacleAvoidanceBehaviourComponent ein DynamicObstacle mit unserem Schiff, dem UtilityScore von 20 und der DynamicAvoidanceSituation STANDON_OVERTAKE hinzugefügt.

Bevor der UtilityScore in der Methode calculateScore() berechnet wird, werden Hilfsvariablen berechnet. Für sowohl das eigene als auch das fremde Schiff wird die Vektorgeschwindigkeit

für die X- und die Y-Achse berechnet. Die Vektorgeschwindigkeit für die X-Achse ergibt sich aus dem Produkt vom Cosinus des Kurses und der Geschwindigkeit des Schiffes. Die Vektorgeschwindigkeit für die Y-Achse ergibt sich aus dem Produkt vom Sinus des Kurses und der Geschwindigkeit des Schiffes. Des Weiteren wird die Distanz zwischen den Schiffen auf der X- und Y-Achse berechnet. Der Quotient aus der Distanz auf den Achsen und der Differenz aus den Vektorgeschwindigkeiten gibt die Zeit für die jeweilige Achse an. Mithilfe der zuvor berechneten Variablen und der meterToCoordinate() Methoden in den RouteUtils wird die Distanz zum Kollisionspunkt auf der X- und Y-Achse berechnet. Anschließend wird eine Koordinate mit den Punkten auf der X- und Y-Achse erstellt. Der UtilityScore wird mit den Parametern ownCourseCorrection und maxTurn berechnet. Die ownCourseCorrection ist der Asinus aus dem Produkt des SafePassingDistanceFactors und der eigenen Schiffslänge dividiert durch die Distanz zum Kollisionspunkt. Die Variable maxTurn ist das Produkt aus dem rateOfTurn, welcher in der Simulation gesetzt wird, und der Zeit bis zum Kollisionspunkt, der Variable tcpaOwnVessel. Der UtilityScore ergibt sich aus dem Quotienten von der courseCorrection und dem maxTurn multipliziert mit 100. Falls das Produkt größer als 100 ist, wird der UtilityScore auf 100 gesetzt.

$$\frac{ownCourseCorrection}{maxTurn} * 100 = \text{UtilityScore}$$

In der Methode calculateCrossingScore() wird in einem ersten Schritt mit der Methode calculateCPA() die Geometrie berechnet, in der die beiden Schiffe kollidieren werden. Falls diese Geometrie weder leer noch null ist, wird die Distanz von unserem Schiff zu dem Kollisionspunkt berechnet. Falls die Distanz zum Kollisionspunkt null oder 0 ist, wird ein UtilityScore von 0.0 zurückgegeben. Anderenfalls wird der UtilityScore mit den Parametern courseCorrection und maxTurn berechnet. Die courseCorrection ist der Asinus aus dem Produkt des SafePassingDistanceFactors und der eigenen Schiffslänge dividiert durch die Distanz zum Kollisionspunkt. Die Variable maxTurn ist das Produkt aus dem rateOfTurn, welcher in der Simulation gesetzt wird, und der Zeit bis zum Kollisionspunkt, der Variable tcpaOwnVessel. Der UtilityScore ergibt sich aus dem Quotienten der courseCorrection und dem maxTurn multipliziert mit 100. Falls das Produkt größer als 100 ist, wird der UtilityScore auf 100 gesetzt.

$$\frac{courseCorrection}{maxTurn} * 100 = \text{UtilityScore}$$

6.4.3 A*-Algorithmus

Nachdem die Erkennung der Situation beim dynamischen Ausweichen vollendet wurde, muss die Situation behandelt werden. Dies wird mithilfe eines angepassten A*-Algorithmus ermöglicht. Dem A* liegen zum einen eine Wahrscheinlichkeitsfunktion und zum anderen eine Heuristik zu Grunde.

Bei der Wahrscheinlichkeitsverteilung wird davon ausgegangen, dass das fremde Schiff nicht ausweichpflichtig ist und seinen Kurs beibehält. Diese Annahme gilt nicht bei HeadOn Situationen. Basierend auf der Annahme wird eine Wahrscheinlichkeitsverteilung für den Kurs des gegnerischen Schiffes erstellt. Diese Verteilung gibt an, mit welcher Wahrscheinlichkeit sich das gegnerische Schiff bei welcher Koordinate befindet. Gleichzeitig werden Zeitschlitze für das gegnerische Schiff berechnet, die angeben wann sich das Schiff, unter Berücksichtigung der aktuellen Geschwindigkeit und dem Abstand zur Koordinate, dort befindet. Diese Zeitschlitze sind Intervalle mit einer Puffer-Variable von aktuell +10% und -10%. In der Abbildung 15 lässt sich die Wahrscheinlichkeitsverteilung für das fremde Schiff, hier als schwarzes Dreieck dargestellt, erkennen. Je dunkler der Rot-Ton ist, desto höher ist die Wahrscheinlichkeit, dass sich das gegnerische Schiff dort befindet.

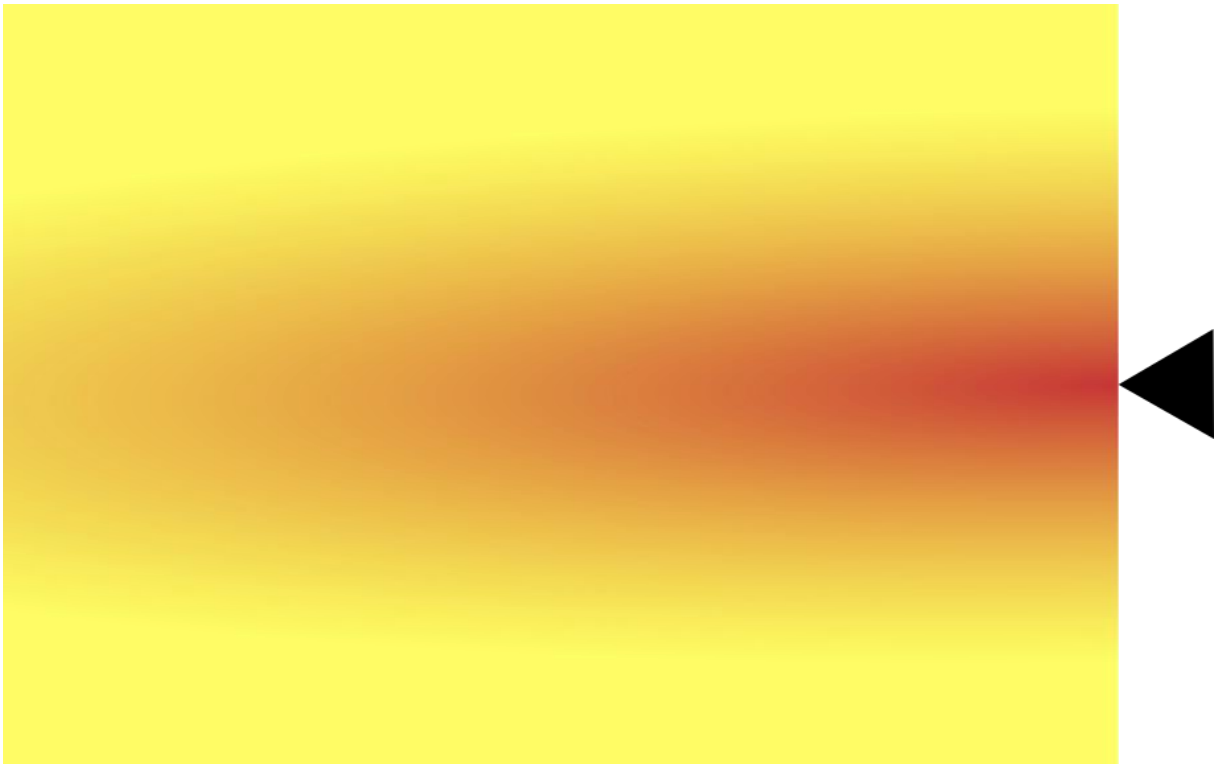


Abbildung 15: A-Algorithmus Wahrscheinlichkeitsverteilung des gegnerischen Schiffes*

Die Heuristik wird immer in Zeit abgebildet und betrachtet die Entfernung durch Geschwindigkeit. Ähnliche wie bei dem statischen Ausweichen werden Punkte hinter dem Schiff künstlich durch die Heuristik bestraft. Die unterschiedliche Behandlung von Situationen durch die Heuristik wird im Folgenden beschrieben:

Falls es sich bei der Situation um das Kreuzen mit einem Schiff handelt, so wird zunächst ein Punkt der Kollision berechnet. Anschließend werden links und rechts vom Kollisionspunkt die ersten Punkte berechnet, bei denen es auf Basis der aktuellen Zeitschlitze sicher zu keiner Kollision kommt. Einer dieser berechneten Koordinaten ist der nächste Zielpunkt im A*-Algorithmus. Der vorherige Zielpunkt wird hinten angereiht. Beim Kreuzen von mehreren Schiffen wird die Heuristik lediglich über die Distanz zum Zielpunkt dargestellt. In Überholen und HeadOn Situationen wird links und rechts eine Passgap angelegt. Diese Passgap gibt den Passierabstand in diesen Situationen an und befindet sich in HeadOn Situationen immer rechts. Basierend auf der Passgap werden die Koordinaten evaluiert. Koordinaten auf der falschen Seite der Passgap werden mit einem Multiplikator bestraft. Basierend auf dem Zielpunkt im betrachteten Envelope wird eine Orthogonale gebildet mit der die Heuristik berechnet wird.

Ähnlich wie bei der Behandlung von statischen Hindernissen gibt es auch bei dynamischen Hindernissen eine Neighbourhood-Funktion. Diese lautet Prediction Neighbourhood und ist ein Kreis von Koordinaten, der auf Basis der aktuellen Koordinate berechnet wird. Dabei werden nur Koordinaten berücksichtigt, die auf Basis des Dynamikmodells erreicht werden können.

Anschließend werden die Koordinaten, die aus der Neighbourhood-Funktion zurückgeben werden, mit einem Zeitintervall belegt. Dieser wiederum gibt an wann das gegnerische Schiff sich bei der Koordinate befinden könnte. Daraufhin wird im A* die berechnete Wahrscheinlichkeit mit unserer parametrisierbaren minProbability verglichen. Diese gibt an, welche Koordinaten, auf Basis der Wahrscheinlichkeiten, befahrbar sind. Wenn die Wahrscheinlichkeit der Koordinate höher ist, dann wird diese nicht evaluiert. Ausgehend von der zuletzt berechneten Koordinate und der geringsten Heuristik werden diese Schritte iterativ wiederholt.

6.5 Fehlermodell und Nicht-Determinismus

Bei der Simulation eines realen Schiffsverkehrs muss beachtet werden, dass nicht jeder Kapitän gleich handelt. Haben zwei Kapitäne denselben Ausgangs- und Zielhafen, bedeutet dies nicht,

dass Sie auch denselben Weg fahren. Dieses Problem wurde mit Hilfe von Fehlermodellen umgesetzt, mit dem Ziel das Verhalten der Agenten menschenähnlich zu gestalten. Diese sollen sicherstellen, dass zwei Agenten mit denselben Parametern eine vorgegebene Route nicht im exakt selben Muster befahren. Hierzu wurde ein erweiterbarer Katalog mit verschiedenen Fehlern erstellt. Bei der Verteilung der Fehler hat der Anwender neben einer Gaußschen Normalverteilung, die Wahl zwischen einer logistischen und linearen Verteilung. Um zu verhindern, dass ein Fehler mehrmals hintereinander auftritt, wird das Prinzip des Critical Hit aus der Spieleindustrie verwendet. Dieses setzt die Wahrscheinlichkeit eines Fehlers nach dessen Auftreten herab, sowie nach längerem nicht-auftreten nach oben, da Menschen in der Regel aus Ihren Fehlern lernen und denselben Fehler selten ein zweites Mal in Folge begehen. In der nachfolgenden Tabelle 69 werden sämtliche Fehler, sowie die daraus resultierende Auswirkung auf das Verhalten des Schiffes, aufgelistet.

Fehler	Auswirkung
Regel übersehen	Regel wird nicht erkannt
Schiff übersehen	Schiff wird nicht erkannt
Statisches Hindernis übersehen	Hindernis wird nicht erkannt

Tabelle 69: Fehlermodell

Nicht-Determinismus	Auswirkung
Statische Hindernisse	später oder eher erkennen
Dynamische Hindernisse	später oder eher erkennen
Routenpunkte	später oder eher erkennen
Falsches Behaviour auswählen	Bei nahezu gleichen UtilityScores wird per Zufall entschieden, welches Behaviour ausgewählt wird

Tabelle 70: Nicht-Determinismus

Tabelle 70 zeigt die Parameterabweichungen der Nicht-Determinismus-Komponente. Die Nicht-Determinismus-Komponente soll eine realitätsnahe Abweichung von der eigentlich berechneten Handlung ermöglichen. Dies geschieht mit Hilfe eines Seeds, welcher es ermöglicht den eingestellten Parameter variabel zu gestalten. Der Seed bestimmt dabei die Varianz der Zufallsfunktion, sodass die Parameter bei einem gleichen Seed nicht voneinander abweichen,

dabei sollen die KVRs weiter beachtet werden. Ergebnis der Nicht-Determinismus-Komponente ist, dass mehrere Agenten bei identischen Routen unterschiedliche Bahnen fahren.

6.6 Kapitänstypen

Die KVRs geben sämtliche Regeln vor, wie sich ein Kapitän im Schiffsverkehr verhalten muss, diese Regeln können jedoch auf unterschiedliche Weise interpretiert werden. Um die unterschiedlichen Interpretationen und Verhaltensweisen von Kapitänen abzubilden, wurden in dem System mehrere Schieberegler, wie in der Abbildung 16 abgebildet, umgesetzt. Diese Schieberegler sind als Einflussgrößen für das Verhalten zu verstehen. So können Kapitäne mit unterschiedlichen Charakterausprägungen erstellt werden, zur Vereinfachung wurden vier Kapitänstypen vordefiniert.

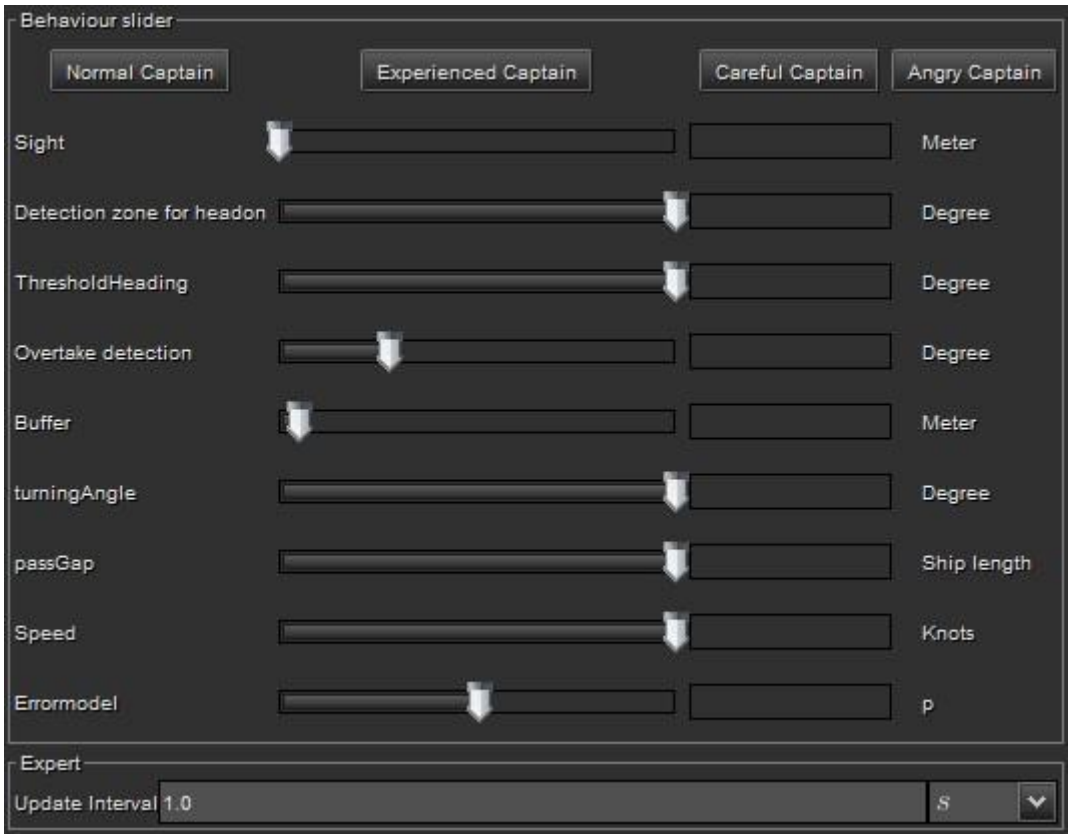


Abbildung 16: Schieberegler und Kapitänstypen

In Tabelle 71 ist aufgelistet, welchen Einfluss die einzelnen Schieberegler auf das Verhalten haben, sowie deren Maximum und Minimum Werte in.

Verhaltensänderung	
Sight	Hier wird festgelegt, in welchem Umkreis der Agent Hindernisse erkennt. (1.000-20.000m)
Detection zone for Headon	Dieser Wert bestimmt, in welchem Winkel entgegenkommende Schiffe erkannt werden. (0°-20°)
ThresholdHeadings	Mittels des ThresholdHeadings wird ein Wert festgelegt, um welchen sich das Heading eines Schiffes vom Heading eines 2. Schiffes unterscheiden kann und es weiterhin als „gleiches“ Heading bewertet wird. (0°-20°)
Overtake Detection	Dieser Wert bestimmt, in welchem Winkel überholende Schiffe erkannt werden. (0°-180°)
Buffer	Der Buffer legt fest, wie viel Abstand ein Schiff beim Passieren eines statischen Hindernisses einhält, hierzu wird der eingegebene Wert zur Größe des Schiffes addiert. (0m-1.000m)
turningAngle	Der TurningAngle gibt an, ab welcher Kursabweichung eine Kursänderung vorliegt. (0°-45°)
PassGap	Der PassGap gibt den Passierabstand zu einem anderen Schiff in Schiffslängen an. (0 -5)
Speed	Speed legt die Geschwindigkeit des Schiffes in Knoten fest. (0-25)
Errormodel	Anhand des gewählten Fehlermodells, bestimmt der Wert p eine gemittelte Wahrscheinlichkeit aller auftretenden Fehler. (0-100)

Tabelle 71: Änderungsparameter der Kapitänstypen

Den vordefinierten Kapitänstypen wurden, wie in Tabelle 72 zu sehen, Charakterausprägungen zugewiesen:

Rücksichtsloser Kapitän:	Normaler Kapitän:	Vorsichtiger Kapitän:	Erfahrener Kapitän:
Sight: 10.000 m Detection zone for Headon: 0° thresholdHeading: 0° Overtake Detection: 0° Buffer: 0m turningAngle: 3° passGap: 0 Schiffslängen Speed: 25 kn Errormodell: 75p	Sight: 15.000m Detection zone for Headon: 5° thresholdHeading: 5° Overtake Detection: 135° Buffer: 20m turningAngle: 20° passGap: 2 Schiffslängen Speed: 20 kn Errormodell: 40p	Sight: 20.000m Detection zone for Headon: 15° thresholdHeading: 15° Overtake Detection: 180° Buffer: 50m turningAngle: 30° passGap: 4 Schiffslängen Speed: 15kn Errormodell: 20p	Sight: 20.000m Detection zone for Headon: 5° thresholdHeading: 5° Overtake Detection: 90° Buffer: 10m turningAngle: 10° passGap: 2 Schiffslängen Speed: 25kn Errormodell: 15p

Tabelle 72: Kapitänstypen

6.7 Ortsabhängig Regeln erkennen

Neben den COLREGS gibt es weitere Ortsabhängige Regeln wie Beispielsweise die Seeschiff-fahrtsstraßen-Ordnung. Diese Regeln werden von den jeweils zuständigen Behörden erlassen und haben nur regionale Bedeutung. Da im Rahmen der Projektgruppe nicht alle Regeln be- und verarbeitet werden können, soll eine Funktion eingeführt werden, diese Regeln individuell anlegen zu können.

Wir haben eine Möglichkeit geschaffen, geografische Abschnitte zu bestimmen, in denen orts-abhängige Regeln gelten. Dies schafft Flexibilität und gewährleistet die Erweiterbarkeit. Die Regeln lassen sich abspeichern und anschließend beliebigen Agenten/Schiffen zuweisen. Es wird ein Polygon erstellt und an HAGGIS übergeben. Innerhalb dieses Polygons gelten die entsprechend ausgewählten Regeln. Der Utility-Score wird entsprechend angepasst.

Für die Regeln (Rules) wird ein eigener Container in HAGGIS angelegt. In diesem Container können dann beliebige Regeln erstellt und abgespeichert werden.

Folgendes wird übergeben:

- Ort
- Regel
- Typ
- Zeit
- Priorisierung

Um die Möglichkeit zu gewährleisten, die unterschiedlichsten Regeltypen zu implementieren und zu ergänzen, wurde zuvor ein Modell mit dem Namen „LocationDependentRules" in der ISS.tuml-Datei implementiert. Darin befindet sich die Klasse „RulesStore", in der alle Regeln mit zugehöriger Position und Priorität hinterlegt sind. Außerdem beinhaltet das Package ein Interface „Rule".

Innerhalb des Packages besteht ein weiteres Package „RuleTypes", welches folgende Klassen beinhaltet:

In der Klasse „DrivingCommanded" wird die Aufhebung des Rechtsfahrgebotes behandelt.

In der Klasse „HeadonSite" wird die Begegnung zweier Schiffe innerhalb eines Gebietes, in dem das Rechtsfahrgebot aufgehoben wurde, behandelt.

In der Klasse „OvertakeRestriction" werden Überholverbote behandelt.

In der Klasse „TakeoverSite" wird das Überholen auf der rechten Seite behandelt.

In der Klasse „Speedrestriction" werden Geschwindigkeitsbegrenzungen behandelt.

In der Klasse „LockArea" werden Schleusen behandelt.

Diese Regeltypen greifen über die Reflection auf die Methoden und die Variablen von der Behaviour Komponente zu, um diese und auch die Provider zu manipulieren. Nun können in HAGGIS Regeln erstellt werden. Der „RuleComplianceProvider" ist im Mainbehaviour aufrufbar. Die Regel wird mit einem Namen, einer geographischen Lage und der entsprechenden „Regel" aus dem Seeverkehrsrecht zugeordnet. Die Regel wird in Sichtweite des Schiffes vor Beginn der geltenden „Regelzone" erkannt, damit es dem Agenten ermöglicht wird, frühzeitig zu reagieren.

Die Regeltypen sind stets erweiterbar, wie es in den Anforderungen vorgegeben ist. Dabei wurde darauf geachtet, dass die Regeln von dem Behaviour getrennt sind. Das hat den entscheidenden Vorteil, dass bei der Erweiterung von Regeln keine Anpassung des ursprünglich erstellten Quellcodes erfolgen muss.

Der Zugriff auf die Schiffe erfolgt über den Application-Tree, also auf die Instanzen der Behaviours. Hierzu wurde eine Regeltypklasse, die eine übergeordnete Klasse darstellt, angelegt. Von dieser Klasse erben die diversen anderen Regelklassen, damit die neu erstellten Regeln dem Regel-Store hinzugefügt werden können. In dieser übergeordneten Klasse sind zudem generische Komponenten implementiert, die den zeitlichen und räumlichen Definitionen zugeordnet werden.

Tabelle 73 zeigt die bisher umgesetzten Regeln.

Regel	Umsetzung	Auswirkung
Sperrzone	Gebiet wird als Polygon angegeben	Gebiet wird als statisches Hindernis erkannt und umfahren
Rechtsfahrgebot	Blockingarea angeben, welche die linke Spur in einer Seestraße anzeigt	Gebiet wird je nach Fahrtrichtung des Schiffes als statisches Hindernis erkannt
Überholverbot	Gebiet als Polygon angeben	In diesem Gebiet werden keine Schiffe überholt und Geschwindigkeiten an den Vorfahrenden angepasst

Geschwindig-keitsbegrenzung	Gebiet muss als Polygon angege-ben werden, mit maximaler Ge-schwindigkeit	Geschwindigkeit wird auf maximal erlaubte Geschwindigkeit angepasst.

Tabelle 73: Ortsabhängige Regeln

Jeder Regel kann ein Zeitfenster zugeordnet werden, in dem die Regel gültig ist.

7 Evaluation

Um die Daten der Simulation mit den historischen Daten vergleichen zu können, haben wir eine Methode geschrieben, die die gefahrenen Routen und Bahnen der Schiffe speichert. Dies funktioniert, indem man dem Schiff eine TrackCharacteristic zuweist. Hat das Schiff eine TrackCharacteristic, so speichert es in einem wählbaren Zeitraum (Beispielsweise alle 30 Sekunden) die MSI, die Position, das Heading und die Geschwindigkeit des Schiffes. Diese Daten können mit Hilfe eines CSV-Writers exportiert und gespeichert werden. Die CSV Dateien sind zum Vergleich vorerst so formatiert, wie auch die via KNIME aus den vorliegenden AIS-Daten extrahierten Daten.

So lassen sich Position des Schiffes und die entsprechende Zeit mit realen Daten der AIS-Aufzeichnung vergleichen.

Außerdem können valide externe Routen, beispielsweise gefilterte AIS Daten, über einen CSV-Reader eingelesen werden. Aus diesen Daten können mit dem Routen-Generator Routen in HAGGIS dargestellt werden.

7.1 Filterung von historischen Daten

Auswahl eines spezifischen Tracks:

Bei der Auswahl eines spezifischen Tracks wurden zunächst sämtliche Tracks innerhalb der abgerufenen Daten angezeigt. Im Folgenden wurde mittels Okularinspektion eine Route ausgewählt, welche möglichst vollständige AIS Informationen über einen Track besitzt. Nach der Auswahl eines Tracks, wurde dieser auf Störungsfreiheit untersucht. Hierzu wurden die Daten wie folgt gefiltert:

Filterung der historischen Daten:

Filter: Database Reader: Hier werden aus der Datenbank die ersten 5.000.000 AIS Daten abgerufen. Aufgrund der schlechten Verbindung zum Server wurde sich auf die ersten 5.000.000 Daten beschränkt, diese ließen sich fehlerfrei und innerhalb weniger Stunden abrufen.

Delegating: Innerhalb dieser Methode soll zum einen eine Bereinigung und Filterung der Daten, und zum anderen die Auftrennung von „position" in Latitude und Longitude erfolgen.

Rule-based Row Filter: Hier wurden fehlerhafte Daten mit einem heading > 360° bzw. einem negativen heading herausgefiltert. Des Weiteren werden Daten mit einem course over ground > 360° und < 0° herausgefiltert. Am Ende wurde noch festgelegt, dass nur Schiffe mit einer Geschwindigkeit > 1 Knoten angezeigt werden, so konnten im Hafen liegende Schiffe herausgefiltert werden.

Java Snippet: Hier wird lediglich die Differenz zwischen course over ground und heading berechnet, damit Daten mit zu großen Abweichungen später herausgefiltert werden können.

Rule-based Row Filter: Hier werden mit dem Ergebnis aus dem Java Snippet, sämtliche Daten mit einer Abweichung > 15 zwischen curse over ground und heading, herausgefiltert.

Java Snippet: Hier wird die Zelle „position", welche die Koordinaten der einzelnen AIS Datensätze enthält, in die Zellen „latitude" und „longitude" getrennt.

Column Filter: Hier werden ausgewählte Zellen mit in die Betrachtung aufgenommen oder aus dieser ausgeschlossen.

GroupBy: Diese Notation filtert alle gleichen Datensätze aus der Ausgabe.

Sorter: Der Sorter sortiert in erster Instanz nach der MMSI und in zweiter Instanz nach dem timestamp

Extract Time Window: Hier kann die Ausgabe auf einen bestimmten Zeitraum begrenzt werden. So können die angezeigten Schiffe auf den Zeitslot unseres Beispielschiffes begrenzt werden.

Geo-Coordinate Row Filter: Diese Notation ermöglicht die Eingrenzung der Koordinaten auf einen bestimmten Bereich, d.h. es werden nur Datensätze angezeigt, welche Koordinaten innerhalb des abgesteckten Gebietes haben. Dadurch wird die Menge der Daten weiter eingegrenzt.

Color Manager: Weist jedem Datensatz eine Farbe zu, um diese auf der Karte besser voneinander unterscheiden zu können.

OSM Map View: Hier wird die Ausgabe auf einer Karte visualisiert, so kann eine okulare Evaluation der Störungsfreiheit vorgenommen werden.

7.2 Einzelevaluation eines Agenten

Bei der Evaluation wurden 5 Millionen AIS-Daten in KNIME eingelesen. Die erste Idee für die Evaluation bestand darin, einen Schlauch aus den AIS-Daten darstellen zu lassen, von dem aus die räumliche Abweichung zu unserem Agenten ausgelesen werden kann. Bei der Auswertung der Evaluation sollte darauf Acht gegeben werden, dass durch Geschwindigkeiten sowie Strömungen und Winde Abweichungen entstehen können. Mittels der Filterung nach dem Schiffstyp, der Geschwindigkeit und der Zeit konnte in KNIME der Track eines Containerschiffes von Bremerhaven nach Hamburg extrahiert werden. Mittels des Colormarkers in KNIME gibt es die Möglichkeit, unterschiedliche Tracks in unterschiedlichen Farben darzustellen. Zudem kann mit dem Color Manager die Sättigung der Farben eingestellt werden, sodass diese transparenter dargestellt werden können.

In HAGGIS wird für die Generierung unserer Vergleichsroute der Pathplanner dafür benutzt, eine Route zwischen dem Start- und dem Zielpunkt des historischen Tracks zu generieren. So können in dem Cruiseprovider der Startpunkt und die Destination angegeben werden und wenn eine Route vorhanden ist, muss nichts weiter gemacht werden, ansonsten wird mittels des Pathplanners eine Route erzeugt. Hierfür wurde in HAGGIS ein Initialtask angelegt, mit dem man eine exportierte csv-Datei aus KNIME als Track anlegen kann. Anschließend werden dem Schiff auf den Koordinaten der AIS-Signale jeweils die Wegpunkte seine Route hinzugefügt, sodass unser Schiff in HAGGIS den extrahierten Track nachfahren kann. Die Tracks der abgefahrenen Routen können anschließend in der Simulation abgespeichert werden. Außerdem wurde ein Schiff generiert, dass sich, mit gleichen Start- und Zielkoordinaten des Schiffes aus dem extrahierten Track, aus dem Routengraphen eine Route erzeugt.

Mittels der Funktion, die es uns ermöglicht, CSV-Dateien in HAGGIS einzulesen, können die historischen Tracks in HAGGIS abgebildet werden. Das finishBehaviour erlaubt es uns, die abgefahrene Route zu exportieren, sodass der Import nicht nur von KNIME nach HAGGIS erfolgen kann, sondern auch ein Export von KNIME nach HAGGIS möglich ist.

Um sämtliche Störungen von unserem herausgefilterten Track von Bremerhaven nach Hamburg zu erhalten, wurden unterschiedliche Filter eingebaut. Als ein hinreichender Abstand, für

den wir davon ausgehen können, dass unser Schiff ohne Behinderung, sprich ohne dem Einleiten eines Manövers, unterwegs ist, wurde die zweifache Länge unseres Schiffes bzw. die zweifache Länge eines Containerschiffes angenommen. Störungen wurden deshalb als Begegnungen mit anderen Schiffen definiert, die einen Abstand von weniger als 250 Metern aufweisen und dessen Begegnung innerhalb eines 10-sekündigem Zeitintervalls stattgefunden haben. Mittels des Colormarkers konnten wir die Tracks der anderen Schiffe soweit farblich herausfiltern, dass wir mit Hilfe farblicher Markierungen der Tracks zwischen entgegenkommenden Schiffen und Überholsituationen unterscheiden konnten. Durch eine Okularinspektion konnte innerhalb der Gruppe festgestellt werden, dass es auf unserem Track von Bremerhaven nach Hamburg zwar zu mehreren Begegnungssituationen mit anderen Schiffen kommt, unser Schiff jedoch keine sichtbaren Manöver fährt, sodass der Track als störungsfrei angesehen werden kann.

Anfangs war zu erkennen, dass unser Schiff in der Simulation eine „Abkürzung" genommen hatte, die, auf diese Erkenntnis gelangte die Gruppe, aufgrund des Tiefgangs des Schiffes von dem historischen Schiff in der Realität nicht gefahren wurde.

Für unsere Evaluation wurde die minimale und die maximale Abweichung unserer Tracks mit dem historischen Track betrachtet. Für die Generierung mehrerer Tracks wurde unsere Simulation mehrfach durchlaufen, sodass man eine Vielzahl an simulierten Tracks erhalten hatte. Anschließend wurden eine Tabelle sowie eine graphische Darstellung für die räumliche Abweichung über die Zeit verwendet. Hieraus konnte abgelesen werden, dass im Durchschnitt eine Abweichung von 500 Metern zu notieren war, was unsererseits als realistisch eingestuft wurde, da aufgrund der Abkürzung hohe Abweichungen zustande gekommen sind. Ansonsten konnten hohe Übereinstimmungen festgestellt werden, sodass unsere Tracks an allen anderen Streckenabschnitten nur geringe räumliche Abweichungen aufwiesen. Zudem konnte nachgewiesen werden, dass der Nicht-Determinismus funktioniert, da innerhalb der generierten Tracks Unterschiede im Befahren der Route aufgezeigt werden konnten. Einzelne Ausreißer, die zwischen unseren Tracks und dem historischen Track auftauchten, konnten darüber erklärt werden, dass keine validen Seekarten-Daten vorlagen. Allerdings ist es auch eine Eigenschaft von intelligentem Verhalten, wenn sich nicht immer regelkonform verhalten wird. Beispielhaft kann ein Auto im Straßenverkehr angeführt werden, dass nicht als intelligent angesehen wird, wenn es sich exakt an die Geschwindigkeitsbegrenzungen hält, da es dann den laufenden Verkehr behindern würde.

7.3 Ergebnis der Evaluation des Einzelverkehrs

Tabelle 74 zeigt das Ergebnis der Einzelevaluation. Hier wurde ein Mittelwert aller Minimal-Abweichungen, sowie ein Mittelwert aller Maximal-Abweichungen berechnet. Des Weiteren zeigt die Tabelle den Mittelwert sämtlicher Abweichungen. Die Größe der Stichprobe zeigt die Anzahl der verglichenen Datensätze zwischen den simulierten und dem historischen Track. Abweichungen zwischen den simulierten Tracks entstehen durch den implementierten Nicht-Determinismus, welcher bewirkt, dass eine Route nicht von jedem Agenten auf dieselbe Art und Weise befahren wird. So wird ein realitätsnahes Agentenverhalten simuliert. Des Weiteren weichen die Routen aus dem Routengraphen in der Simulation von den Routen der historischen Schiffe ab. Dies geschieht, aufgrund einer fehlenden Routenvalidierung, welche im Rahmen der Projektgruppe nicht vorgesehen war. Im Rahmen der Einzelevaluation wurde aktuell lediglich ein Track mit 10 simulierten Routen verglichen, hierdurch können Abweichungen durch Umwelteinflüsse, welche bei simulierten Tracks nicht betrachtet wurden, entstehen. Das Package „Evaluation“ im System, beinhaltet sämtliche Klassen zur Berechnung der Distanzen zwischen einem „Main-Track“ und beliebig vielen simulierten Tracks.

Größe der Stichprobe	3626
Mittelwert aller Minimal-Abweichungen	85,36 m
Mittelwert aller Maximal-Abweichungen	210,73 m
Mittelwert aller Abweichungen	142,85 m

Tabelle 74: Ergebnis der Einzelevaluation

Die nachfolgende Abbildung 17 stellt die Mittelwerte der minimalen, maximalen und durchschnittlichen Abweichungen dar.

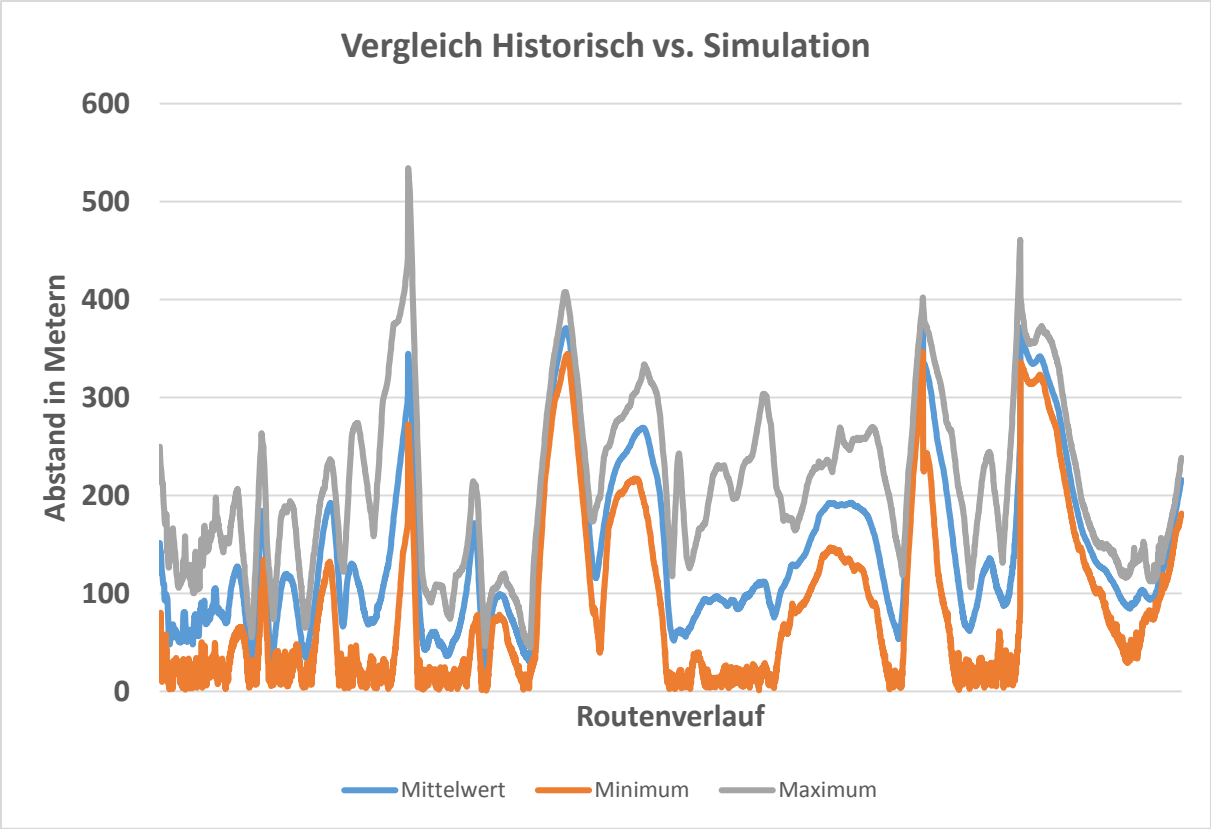


Abbildung 17: Kursabweichungen zum historischen Track

7.4 Evaluation des Gesamtverkehrs

Anfangs kam innerhalb der Gruppe die Idee auf, dass man für mehrere Bereiche einen Korridor mit Maximum- und Minimum-Werte bildet, um anschließend zu überprüfen, ob unser Schiff sich in diesem Korridor bewegt. Dann könnten mehrere Schiffe generiert werden, die diese Route abfahren, damit die Maximum- und Minimum-Werte unserer Schiffe mit den Maximum- und Minimum-Werten aus den historischen Tracks verglichen werden können.

Um für unsere Tracks eine Annäherung zum historischen Track herzustellen, wurde das Szenario um Sealanes erweitert und dementsprechend Sealanes als Regel angelegt.

Da das Tool KNIME nicht dafür ausgelegt ist, über 10 Millionen Daten abzubilden, und es somit auch mit der vorhandenen Rechenleistung nicht möglich ist, diese Daten abzufragen, musste die deutsche Bucht in kleinere Bereiche unterteilt werden, um so einzeln Vergleiche ziehen zu können.

Nach dem Versuch die „Verkehrsschläuche" abzubilden, um so einen Korridor erstellen zu können, in dem sich der reale Schiffsverkehr abgespielt hat, musste festgestellt werden, dass der Aufwand, um die Positionsdaten zu extrahieren, zu groß erscheint. Zudem kamen wir zu dem Entschluss, dass die Ergebnisse zu ungenau seien. Stattdessen kam innerhalb der Gruppe die Idee auf, die Positionsdaten durch eine transparente Darstellung in KNIME abzubilden. Somit hätte man eine Art Heat-Map geschaffen, die Verkehrsgebiete, die höher frequentiert befahren werden, stärker hervorhebt, als Seegebiete, die weniger stark befahren sind. Dies würde einen guten grafischen Vergleich zwischen den historischen und unseren Tracks darstellen.

Nachdem man die generierten Positionssignale in KNIME mit Daten von dem realen Verkehr vergleichen wollte, um Abweichungen der simulierten Agenten von realen Routen optisch mittels einer Heatmap identifizieren zu können, musste festgestellt werden, dass die Performanz bei der Darstellung von großen Datenmengen in der Mapview von KNIME nicht ausreicht.

Um trotzdem zu aussagekräftigen Ergebnissen zu kommen, entschloss sich die Gruppe dazu, eine Lösung zur Darstellung der historischen Daten in HAGGIS zu implementieren. Hierfür wurde das dichteverbundenes Clusterverfahren des DBSCANs verwendet.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise - Dichtebasierte räumliche Clusteranalyse mit Rauschen) ist ein dichtebasierender Algorithmus zur Clusteranalyse. Der Algorithmus benötigt zwei Eingaben ε und minPts. Ein Punkt ist von einem anderen Punkt erreichbar, wenn der Abstand der beiden Punkte kleiner als ε ist. Dabei wird als Abstand wie bei dem k-nächste-Nachbarn-Algorithmus für gewöhnlich die euklidische Distanz genommen. Ein Punkt ist dicht, wenn er minPts erreichbare Nachbarn hat. Es werden drei Arten von Punkten unterschieden:

- Kernpunkte sind Punkte, welche dicht sind.
- Dichte-erreichbare Punkte sind Punkte, welche selbst nicht dicht sind, aber von einem Kernpunkt aus erreichbar sind.
- Rauschpunkte sind die übrigen Punkte, die also weder dicht, noch dichte-erreichbar sind.

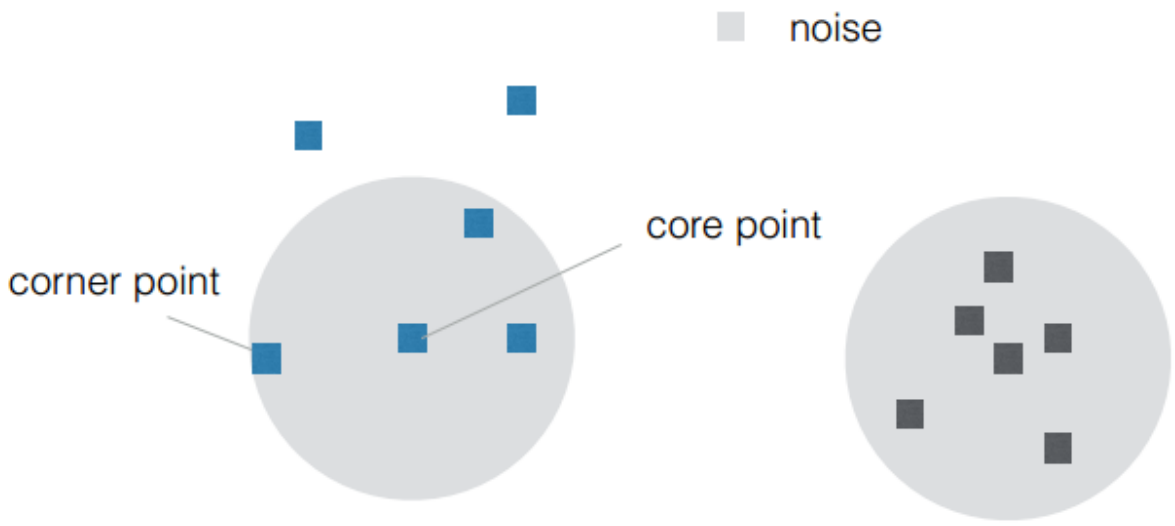


Abbildung 18: DBSCAN (vgl. Kramer 2009)

Der Algorithmus arbeitet dann wie folgt: Solange es noch einen nicht besuchten Punkt gibt, zähle seine erreichbaren Nachbarn. Handelt es sich um einen Dichtepunkt ist das Zentrum eines neuen Clusters gefunden, ansonsten wird er vorläufig als Rauschpunkt markiert. Wenn ein neues Cluster gefunden wurde, werden alle Punkte, die vom Dichtepunkt aus erreichbar sind, dem Cluster zugeordnet. Die Punkte, die dem Cluster zugehören, aber bereits besucht wurden, können keine Dichtepunkte sein, sondern sind nur dichte-erreichbare Punkte und werden entsprechend markiert. Die Punkte, die dem Cluster angehören und noch nicht besucht wurden, werden als nächstes besucht. Wenn sie ebenfalls Dichtepunkte sind, werden mit ihren erreichbaren Punkten auf gleiche Art und Weise fortgefahren, wenn sie nur dichte-erreichbar sind, werden sie nur entsprechend markiert und dem Cluster zugeordnet. Wenn es keine weiteren nicht besuchten Punkte, die vom einen Dichtepunkt des Clusters aus erreichbar sind, mehr gibt, wurde das komplette Cluster gefunden und es wird mit dem nächsten nicht besuchten Punkt auf normale Art fortgefahren.

Auf diese Weise wird jeder Punkt nur genau einmal besucht und der Algorithmus ist von seiner Komplexität linear, wenn die Berechnung der erreichbaren Punkte in konstanter Zeit erfolgt, ansonsten quadratisch. Ein dichte-erreichbarer Punkt kann von mehreren Dichtepunkten aus erreichbar sein. Der Algorithmus ordnet ihm nicht-deterministisch einem Cluster zu. Der Rest geschieht deterministisch.

Ein Vorteil gegenüber anderen Clustering-Algorithmen, wie zum Beispiel k-Means besteht darin, dass man nicht vorher die Anzahl der Cluster angeben muss. Das Problem besteht wieder darin, die passenden Werte für die Parameter ε und minPts zu wählen.

Um den DBSCAN auf unsere Problemstellung übertragen zu können, wurden die historischen Daten in einen eigens programmierten Quadtree geladen. Hierbei wurde eine Anpassung vorgenommen, sodass dort bereits die Filterung der Daten vorgenommen wird. Mittels der Nachbarschaftsbeziehung sollen keine Punkte eingefügt werden, die innerhalb eines parametrisierbaren Radius eines anderen Punktes liegen. Dies gewährleistet, dass die Datenmenge überschaubar bleibt. Anfangs haben wir Punkte als benachbart angesehen, die eine Entfernung von unter 25 Metern aufweisen. Bei ersten Tests der Projektgruppe konnten drei Millionen Tupel verarbeitet werden, aus denen knapp 8000 Cluster in der Deutschen Bucht identifiziert und visuell abgebildet werden konnten. Aus diesen Clustern galt es dann, eine konkave Hülle über diese zu legen, sodass eine bessere Visualisierung gewährleistet ist.

Der Grund, weshalb man sich für das Clustering entschieden hat, war, dass es möglich ist, Ausreißer in unserer Evaluation eliminieren zu können. Mittels der externen Bibliothek OpenTripPlanner wird eine konkave Hülle berechnet, die über das gesamte Cluster gelegt wird. Der Sinn hinter diesem Vorgehen bestand darin, dass die konkave Hülle in HAGGIS angezeigt werden und analysiert werden kann, ob sich die Schiffe in der Simulation innerhalb dieser Hülle bewegen.

Da das Clustering in HAGGIS bis zu 25 Minuten beanspruchen konnte, entschied man sich dafür, das Clustering und das Anzeigen in HAGGIS separat vorzunehmen. Auf der einen Seite wurde ein eigenständiges Tool entwickelt, welches aus den AIS-Daten WKT-Dateien mit den konkaven Hüllen in Formen von Polygonen zurückgibt. Andererseits existiert ein initialer Task, der diese Dateien einliest und die enthaltenen Polygone in HAGGIS anzeigt. Weiterhin kann der Prozess beschleunigt werden, indem das Gebiet in kleinere Datenmengen aufgeteilt und die entstandenen Polygone anschließend zusammengeführt werden. Dies war für unsere Zwecke ebenso hilfreich, da für das Verkehrstrennungsgebiet auf einer Seite nur vereinzelt Daten vorhanden waren und diese durch das Clustern wegfielen. So war das aufgeteilte Clustern hilfreich, damit das übergreifende Clustern die Daten im Verkehrstrennungsgebiet nicht als Ausreißer ansieht und verhindert werden kann, dass diese dann nicht mit geclustert werden.

Anfangs sind bei der Evaluation mittels der Cluster noch Probleme aufgetreten, da die konkave Hülle teilweise Nogoareas ausgewiesen hat, die das Schiff komplett umgaben und es somit

keinen Weg finden konnte und zum Stehen gekommen ist. Die Schlussfolgerung, die die Gruppe aus dieser Tatsache zog, war, dass fehlerhafte Seekarteninformationen vorliegen. Zum Beispiel könnte es sein, dass die Seekarten auf Ebbe ausgelegt sind und so ein Durchqueren der Elbmündung für unsere Schiffe nicht möglich ist, da der Nogosolver eine Nogoarea zurückgibt. Hierfür wurde zuerst der Lösungsansatz verfolgt, Schiffe mit einem geringeren Tiefgang auszuwählen.

Um nun einen Vergleich zwischen dem realen Verkehr und den simulierten Agenten ziehen zu können, wurden sich anfangs zwei Ideen überlegt: Die erste bestand darin, dass man möglichst viele Schiffe generiert und diese im gleichen Gebiet wie die historischen Tracks fahren lässt. Hierbei würde man sich dann ebenfalls die AIS-Signale ausgeben lassen, um ebenso Cluster und konkave Hüllen berechnen zu lassen. Anschließend könnte ein Vergleich zu den historischen konkaven Hüllen gezogen werden, indem eine Okularinspektion vorgenommen wird oder aber Distanzen zwischen den konkaven Hüllen berechnet und ausgewertet werden. Der zweite Vorschlag beschäftigt sich mit einer Variante, die sich mit einer Bildverarbeitung auseinandersetzt. So könnte das historische Polygon beispielsweise blau eingefärbt werden und das Polygon aus dem Simulationsdaten gelb eingefärbt werden und die beiden eingefärbten Polygone anschließend übereinander gelegt werden. Die Bereiche, die anschließend eine grüne Einfärbung aufweisen, würden dann Übereinstimmungen im Gesamtverkehr darstellen, während der blaue bzw. der gelbe Bereich Gegenden im Gesamtverkehr darstellt, in denen der historische vom simulierten Verkehr abweicht.

7.5 Ergebnis der Evaluation des Gesamtverkehrs

Die Auswertung des Gesamtverkehrs wurde wie oben beschrieben mit Hilfe von eingefärbten Polygonen durchgeführt. Die gebildeten konkaven Hüllen der historischen Tracks wurden blau, die der simulierten Tracks rot eingefärbt (siehe Abbildung 19). Beide Hüllen wurden übereinandergelegt und anschließend die farbigen Pixel gezählt.

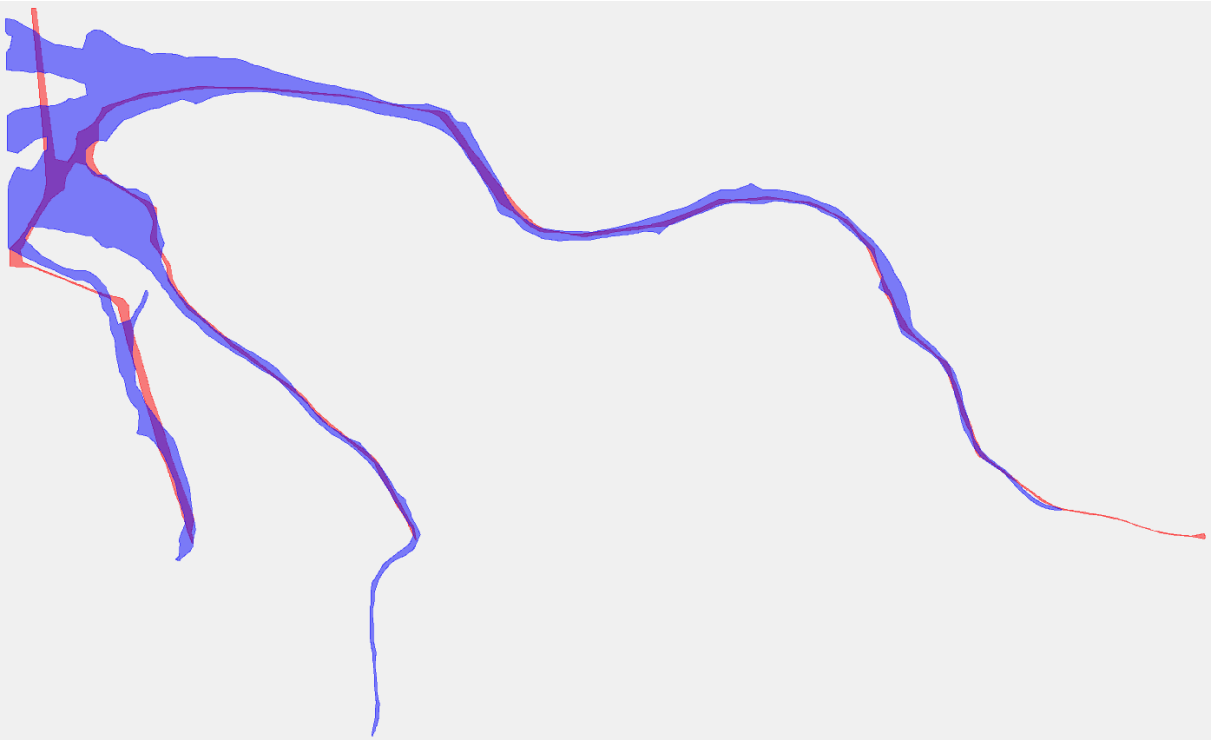


Abbildung 19: Evaluation Gesamtverkehr Polygon

Der Abgleich der Daten wurde mit Hilfe der „Precision and Recall“ Sichtweise beurteilt. Dabei sagt der Recall aus, wie viele der simulierten Daten die historischen Daten abdecken (Vollständigkeitsquote). Precision sagt aus, wie genau die simulierten Daten in den historischen liegen (Genauigkeitsquote). Folgende Werte wurden ermittelt:

Historische Daten ohne Übereinstimmung mit simulierten Daten: 50.316 Pixel

Simulierte Daten ohne Übereinstimmung mit historischen Daten: 3.536 Pixel

Übereinstimmung von simulierten und historischen Daten: 10.654 Pixel

Dadurch lassen sich Precision und Recall wie folgt ermitteln:

Precision: 10.654/(10.654+50.316)*100 = 75,08

Recall: 10.654/(10.654+3.536)*100 = 17,47

Der Recall ist hierbei zweitrangig, da es darum geht, ob alle Strecken aus der Realität abgedeckt wurden. Diese Aussage wird dann interessant, wenn eine Aussage über realitätsnahen Gesamtverkehr in einem speziellen Szenario getroffen werden soll.

Wichtiger ist die Precision, welche aussagt ob ein Szenario valide ist. Die Precision zeigt also, ob die simulierten Tracks auch in der Realität gefahren werden.

Dabei ist zu beachten, dass in der Simulation die Verbindung Wilhelmshaven - Helgoland angegeben wurde. Diese Fährverbindung existiert in der Realität jedoch nicht mehr, was dazu führte, dass die simulierten Daten von den historischen Daten abweichen (siehe Abbildung 20).

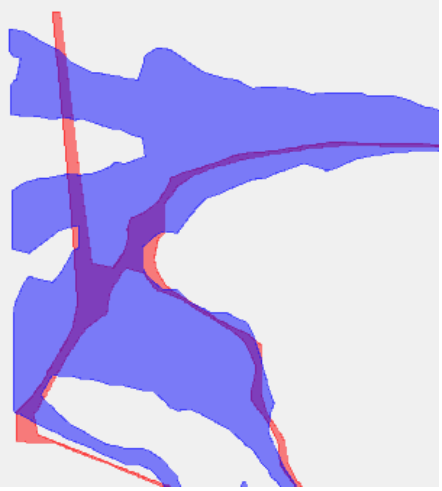


Abbildung 20: Abweichung des Gesamtverkehrs

Unter Berücksichtigung der Abweichung können die simulierten Tracks mit einer Precision von rund 75 schließlich als realitätsnah eingestuft werden.

8 Test

JUnit-Test sind nicht durchgeführt worden, da bei neu hinzukommenden Funktionalitäten neue Tests implementiert werden müssten. In diesem Projekt wurde dieser Aufwand vermieden und stattdessen im fortlaufenden Projekt, zum Abschluss einzelner Userstories, Szenarien erstellt. Die Szenarien wurden dann auf die gewünschten Funktionalitäten überprüft. Über die grafische Oberfläche wurde der Ablauf verfolgt und ebenfalls über die Konsolenausgabe der Entwicklungsumgebung auf Richtigkeit überprüft.

Tabelle 75 zeigt die Werte, welche beim Testen des frontalen Manövers verwendet wurden.

StandardFrontal	
UpdateRate	1
TrackControl Rudder	
SightDistance	10.000m
RPMAcc	6.000 rounds/min^2
RouteFollowe	Deviation ausschalten
RadaSetSpeed	15 rpm
Physics	
Pcontroler	3.0
MaxSpeed	25m/s
MinSpeed	15m/s
MaxRPM	6.000 rpm
MaximumRateOfTurn	2 deg/min
IntendendVelocityProvider	
Increase	10.0
EnemyShip	
BehaviourCompnent	

Tabelle 75: Parametrisierung des Szenarios StandardFrontal

Die Schiffe stoppten nach kurzem Anfahren. Die Berechnung eines Weges erfolgte zwar, die Schiffe fuhren aber nicht weiter. Der UScore und Geschwindigkeit änderten sich fortlaufend trotz Stillstand des Schiffes.

Nach 1-2 min wird ein neuer Weg berechnet und die Schiffe fuhren weiter, jedoch nicht die geplante Route. Die Schiffe machten große Sprünge.

Nach Abänderung folgender Parameter:

- Physics: MaximumRateOfTurn=2 rpm

wurde das Problem gelöst. Die Schiffe berechneten nun einen sinnvollen Weg und fuhren diesen auch ab.

Nach Abänderung der Zellgröße auf 500x500 wurde ein effizienter Weg berechnet. Das Schiff konnte jedoch keine Kurve mit engem Radius fahren und weichte so stark vom berechneten Weg ab. Zudem ist aufgefallen, dass das Schiff einen großen Bogen macht, um zurück auf die Route zu gelangen.

Tabelle 76 zeigt die Werte, welche beim Testen des Kreuzungsmanövers verwendet wurden.

StandardKreuzen	
IntendendVelocityProvider:	
UpdateRate	1
Increase	10.0
Physics:	
MaximumRateOfTurn	2 deg/min
RadaSetSpeed	15 rpm
RPMAcc	6.000 rounds/min2
MaxRPM	6.000 rpm
MaxSpeed	25m/s
TrackControl Rudder	
Pcontroler	3.0
BehaviourCompnent	
SightDistance	10.000m
EnemyShip	
RouteFollow:	Deviation ausschalten
Physics:	
MaxSpeed	15m/s

Tabelle 76: Parametrisierung des Szenarios StandardKreuzen

Zwei Schiffe kreuzten, jedoch erfolgte keinerlei Wegberechnung oder jedwedes Ausweichmanöver.

Nach der Abänderung der Physics, der MaximumRateOfTurn auf2 rpm wurden erfolgreich verschiedene Wege berechnet und der zuletzt berechnete Weg wurde abgefahren. Das Dynamikmodell des Schiffes lässt das Befahren der zuletzt berechneten Route nicht zu. Die Schiffe fuhren ineinander.

Nach der zusätzlichen Abänderung der Zellgröße auf 500x500 wurde das Problem erfolgreich gelöst.

Tabelle 77 zeigt die Werte, welche beim Testen des Überholmanövers verwendet wurden.

StandardÜberholen	
IntendendVelocityProvider	
UpdateRate	1
Increase	10.0
Physics:	
MaximumRateOfTurn	2 deg/min
RadaSetSpeed	15 rpm
RPMAcc	6.000 rounds/min^2
MaxRPM	6.000 rpm
MaxSpeed	25m/s
TrackControl Rudder	
Pcontroler	3.0
BehaviourComponent	
SightDistance	10.000m
EnemyShip:	
RouteFollow	Deviation ausschalten
Physics:	
MaxSpeed	15m/s

Tabelle 77: Parametrisierung des Szenarios StandardUeberholen

Bei einer SightDistance von 10.000m erkennt OurShip das EnemyShip bevor es die Höchstgeschwindigkeit erreicht hat. Dadurch ist der Geschwindigkeitsunterschied nicht groß genug und das OurShip passt seine Geschwindigkeit der des EnemyShips an und folgt diesem.

Nach der Abänderung der SightDistance auf 3.000m hält das Schiff mit dem RouteFollow den Kurs und wird von dem OurShip rechts überholt ohne Alternativ- oder Ausweichweg, trotz Utilityscore von 100. Hierbei wird die Geschwindigkeit bei Annäherung an das EnemyShip gedrosselt.

Das Schiff passt sich trotz erreichter Höchstgeschwindigkeit und einem Geschwindigkeitsunterschied von 11 m/s der Geschwindigkeit des Vordermannes an.

Zudem fuhren die Schiffe zu dicht aneinander vorbei.

Wiederum wurde der MaximumRateOfTurn auf 2 rpm abgeändert.

Diese Veränderung löste das Problem, worauf die Schiffe den berechneten Weg, abfuhren.

Verschiedene Routen wurden berechnet, dabei steht das Schiff für einen kurzen Moment. Nach abgeschlossener Berechnung springt das Schiff weiter.

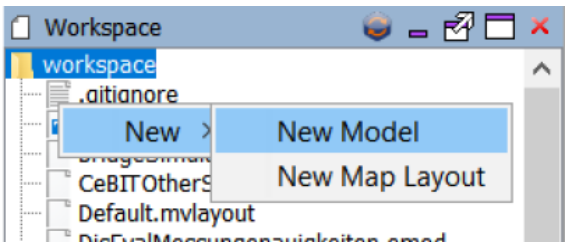
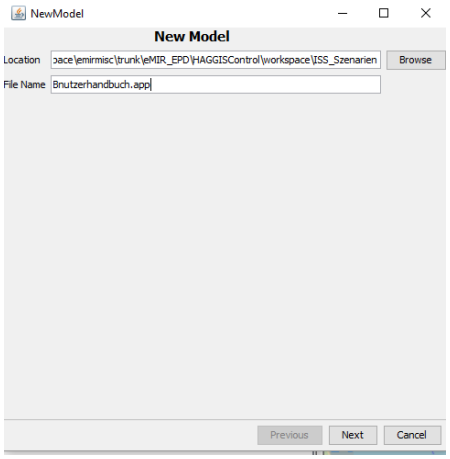
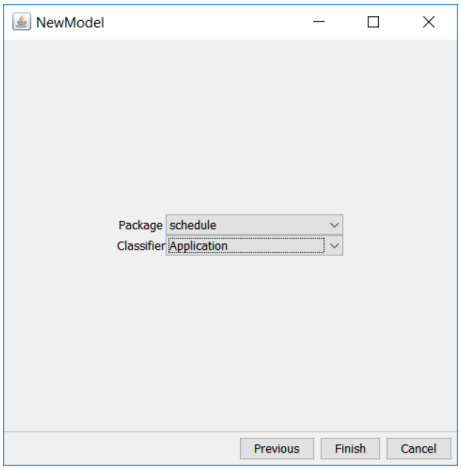
Nach der erneuten Änderung der Zellgröße auf 500x500 wurde der erste berechnete Weg befahren, ohne dass ein Schiff, dem ausgewichen werden muss, vorhanden war.

Diese Tests sind ebenfalls mit mehreren Schiffen durchgeführt worden. Die Fehler, die durch eine Veränderung in der Parametrisierung nicht behoben werden konnten, sind im Laufe des Projektes durch eine Verbesserung der Implementation behoben worden. Auch die Optimierung der Konzepte führte zu einer Verbesserung. Gerade durch die Optimierung des A-Sterns konnten Schwierigkeiten des Ablaufes behoben werden.

9 Benutzerhandbuch (Anlegen eines Szenarios)

Dieses Benutzerhandbuch beschreibt in den folgenden Tabellen, das Erstellen von Szenarien, beginnend von dem Anlegen einer App über verschiedene Zuweisung des unterschiedlichsten Verhaltens. Ebenfalls sind Parametervorschläge zu finden. Darüber hinaus sind Zuweisungen zu beschrieben, indem eine Evaluation des Verkehres in HAGGIS möglich ist.

9.1 Erstellen einer App

	Über das Kontextmenü des Workspace wird New Model gewählt, um eine neue App zu erstellen.
	Anschließend wird ein entsprechender Name der App festgelegt.
	Um ein Szenario anzulegen, wird das Package auf „schedule“ und der Classifier auf „Application“ festgelegt.

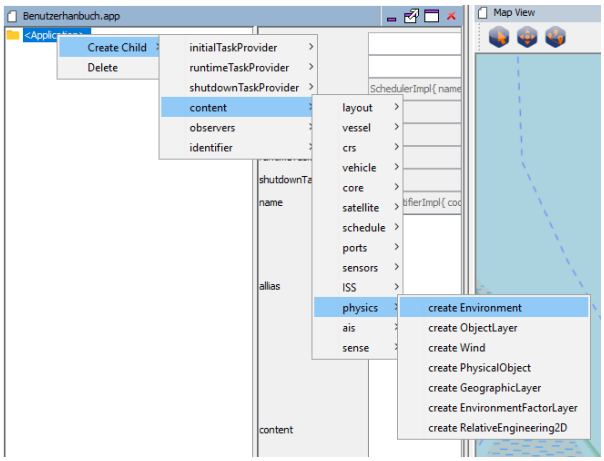
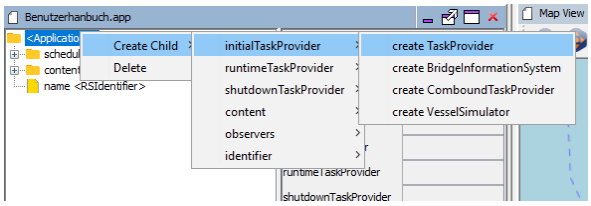
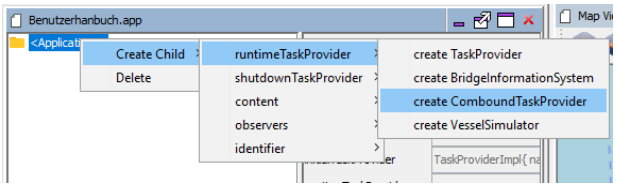
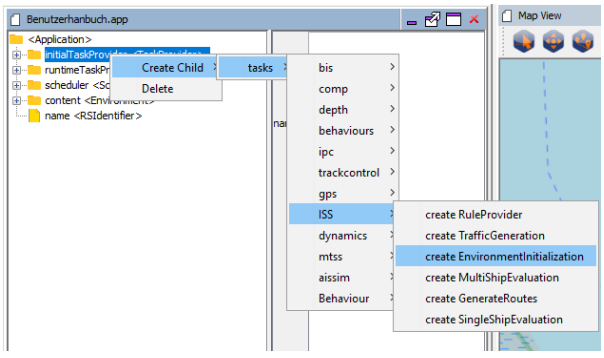
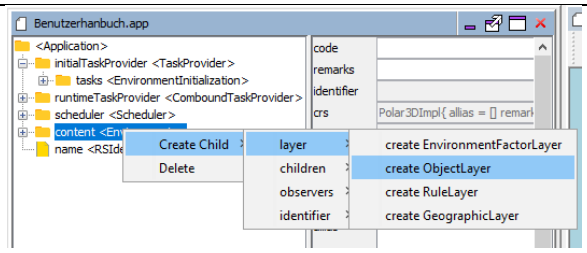
 	Nachdem das New Model angelegt wurde, werden aus dem neuen Bereich weitere Funktionen hinzugefügt. Zum einen muss über das Kontextmenü content<enviroment> und der runtime-TaskProvider<TaskProvider> hinzugefügt werden. Um Schiffe in einem Szenario anzulegen, wird über einen Rechtsklick auf der <Application> das Kontextmenü create-Child -> content -> physics -> create Enviroment die dafür benötigte Umgebung erstellt.
	Als Nächstes wird der CombundTask-Provider erstellt, um die Objekte, in diesem Fall die Schiffe, anzulegen
	Zusätzlich wird noch die EnviromentIni-tialization benötigt
	Über die zuvor erstellte Umgebung wird daraufhin der ObjectLayer hinzugefügt, um die eigentlichen Schiffe zu erzeugen

Tabelle 78: Anlegen einer App

9.2 Erstellen von Schiffen

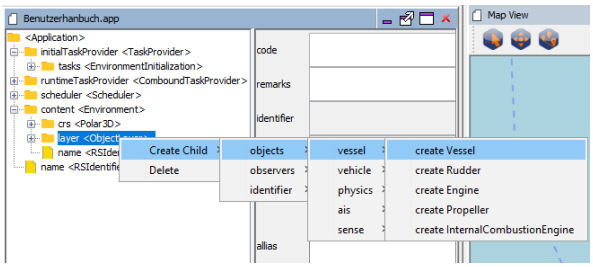
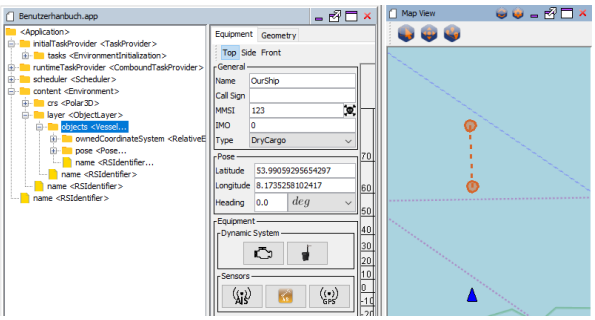
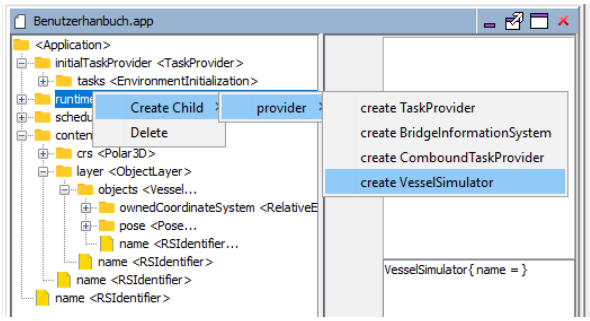
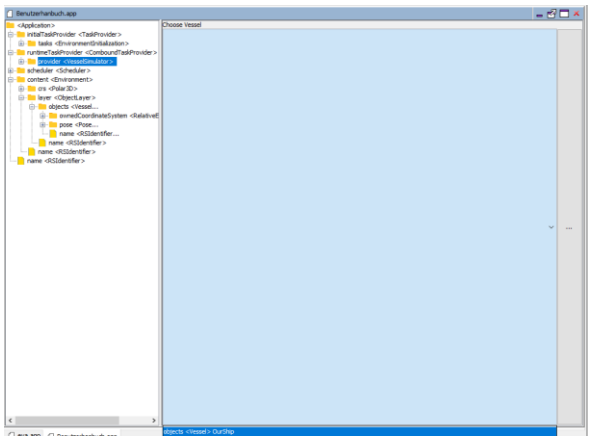
	Um die Schiffe in einem Szenario anzulegen, wird über das Kontextmenü das zuvor angelegte content <Enviroment> -> layer<Objectlayer> über den Reiter create Vessel, die Umgebung des Schiffes angelegt.
	Im Bereich General des Vessels werden hier Name, der MMSI-Wert, sowie die Position über Latitude und Longitude eingegeben. Durch Aktualisierung der Karte wird das Schiff im Kartenbereich angezeigt.

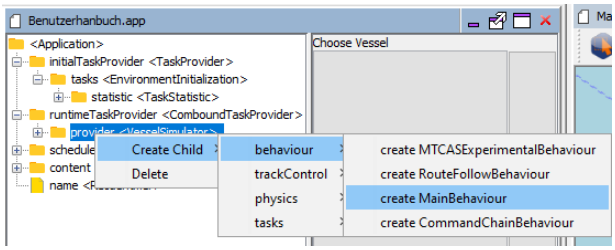
Tabelle 79: Das Erstellen von Schiffen_1



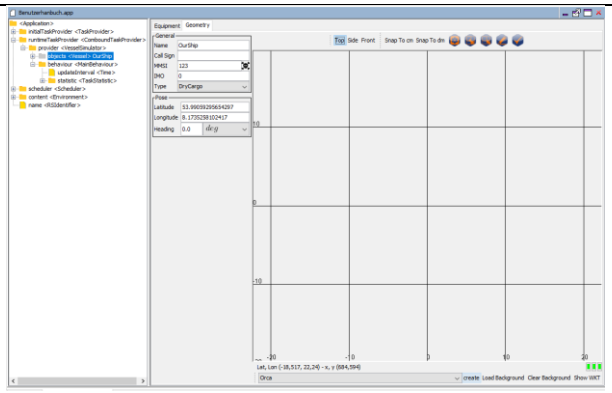
Über das Kontextmenü des runtimeTask-Providers wird der entsprechende Vessel-Simulator erstellt und das Objekt hinzugefügt.



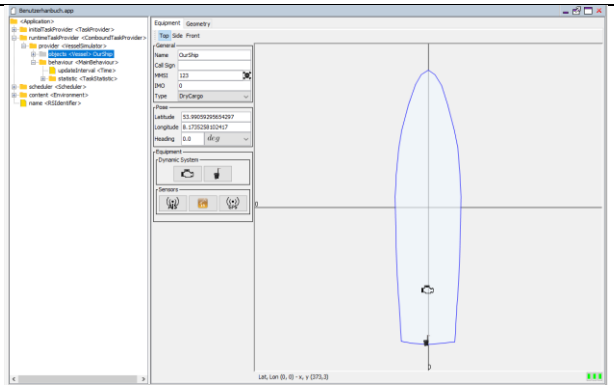
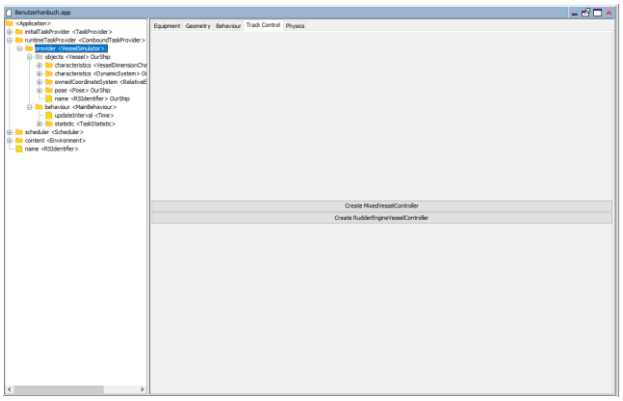
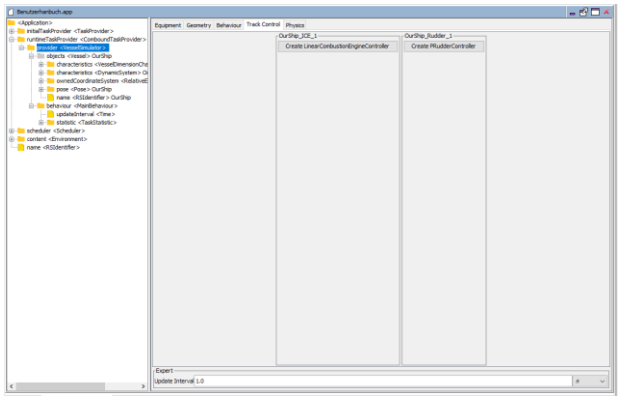
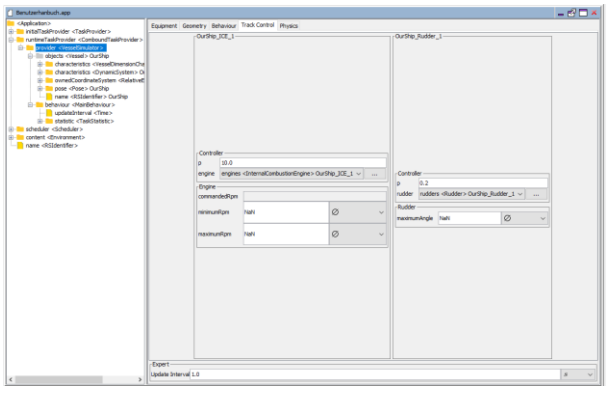
9.2.1 Main Behaviour



Sobald das Objekt hinzugefügt wurde, wird das MainBehavior erstellt.



Jetzt kann über den Reiter „Geometry", Top, Side und Front über den createButton erstellt werden. Im Anschluss muss gespeichert werden.

	Danach besteht die Möglichkeit, über den Reiter „Equipment“ den Motor, sowie das Ruder hinzuzufügen. Dieses geschieht über drag und drop.
	Jetzt können die Parameter der Controller des Ruders und des Motors gesetzt werden. Hierfür müssen zunächst im Provider über den Track Control die entsprechenden Buttons gesetzt werden.
 	Nach dem Betätigen des Create RudderEngineVesselController wird die nebenstehende Maske mit Ausgangswerten erstellt.

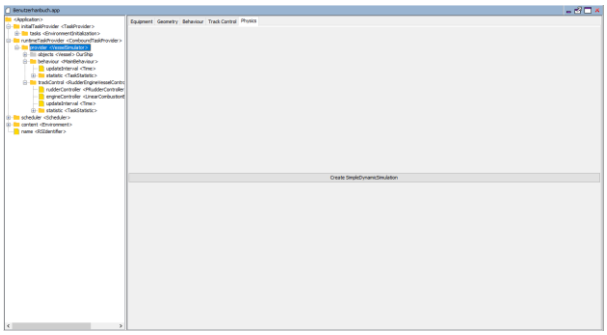
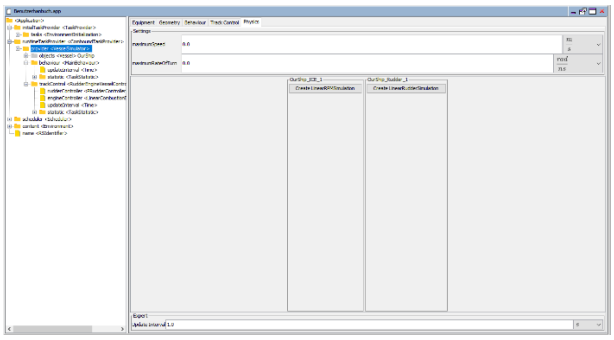
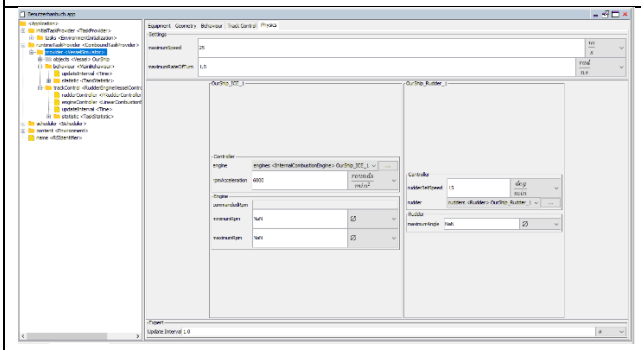
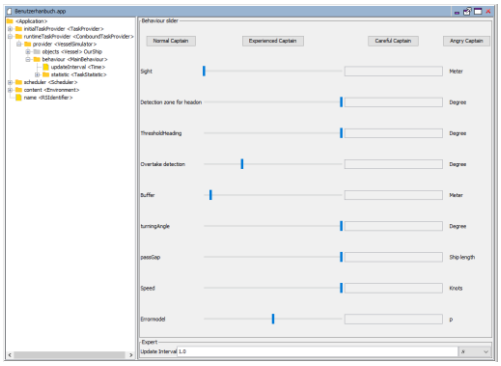
 	Über den Reiter "physics" wird der Bereich angelegt, um die Eigenschaften des physikalischen Objektes, in diesem Fall das Schiff, einzustellen.
	Als Beispielwerte können gewählt werden: • maximumSpeed 25 m/s • maximumRateofTurn 1.0 rad / ns • rpmAcceleration auf 6000 rounds /min² • rudderSetSpeed auf 15 deg/min

Tabelle 80: Erstellen von Schiffen_2

9.3 Verhalten

	In der Ansicht vom <MainBehaviour> wird eine Auswahl verschiedener Kapitänstypen vorgeschlagen. Über die angezeigten Buttons können diese ausgewählt werden. Um eine individuelle Einstellung vorzunehmen, werden die Regler für eine manuelle Anpassung verwendet.

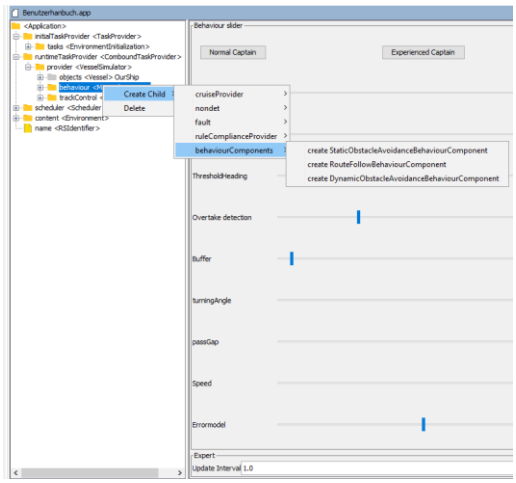
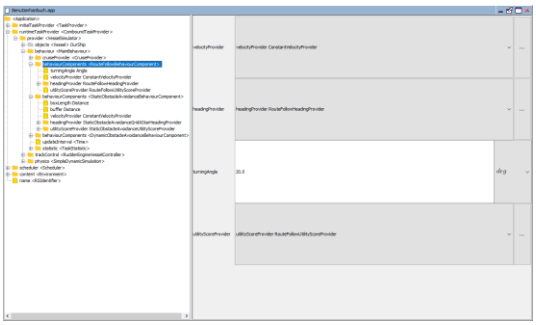
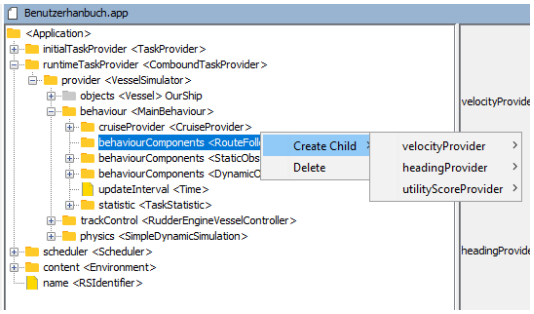
	Über das MainBehaviour wird nicht nur der Cruiseprovider, der im späteren noch näher erläutert wird, erstellt, auch die behaviourComponets werden hierüber hinzugefügt. Die nebenstehende Abbildung zeigt die Auswahlmöglichkeiten zwischen der RouteFollowBehaviourComponent, der DynamicObstacleAvoidanceBehaviourComponent und der StaticObstacleAvoidanceBehaviourComponent.

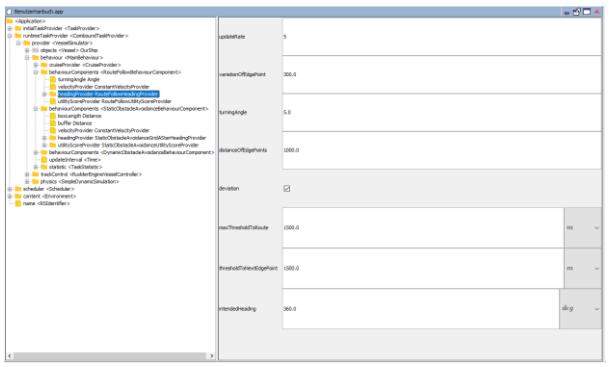
Tabelle 81: Charaktereigenschaften und Verhalten

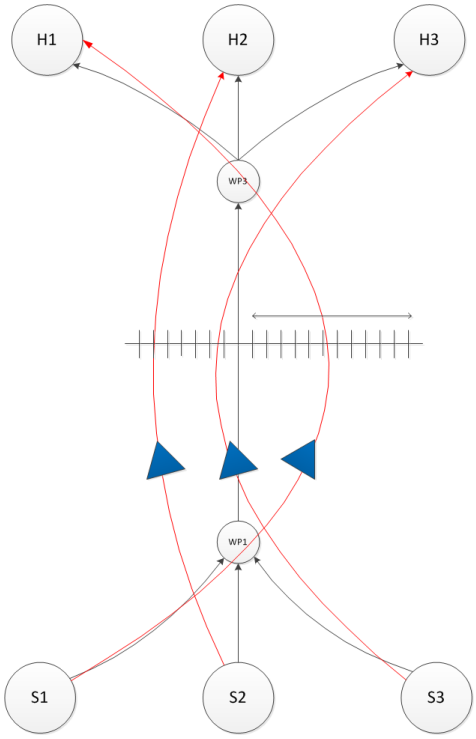
9.4 Routen verfolgen

### 9.4.1 RouteFollowBehaviourComponent 	Zusätzlich hat die RouteFollowBehaviourComponent auch noch ein turningAngle, welcher angibt, ab welcher Kursabweichung eine Kursänderung vorliegt.
	Die möglichen Provider werden durch einen Rechtsklick auf die erstellte behaviourComponent generiert. Hierbei kann zwischen utilityScoreProvider, velocityProvider und headingProvider entschieden werden, welches in der Abbildung zu sehen

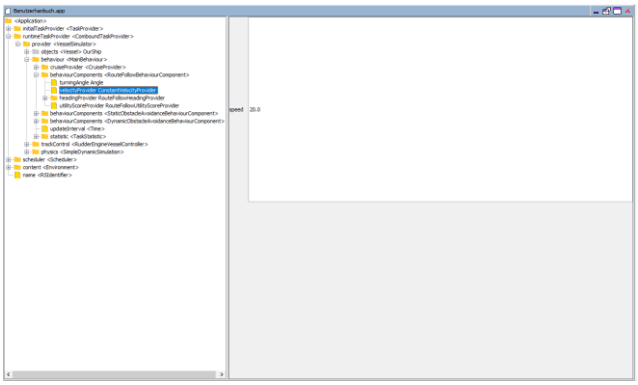
	ist. Diese haben jeweils noch unterschiedliche Ausprägungen, die im Folgenden noch näher beschrieben werden.

9.4.2 *RouteFollowHeadingProvider*

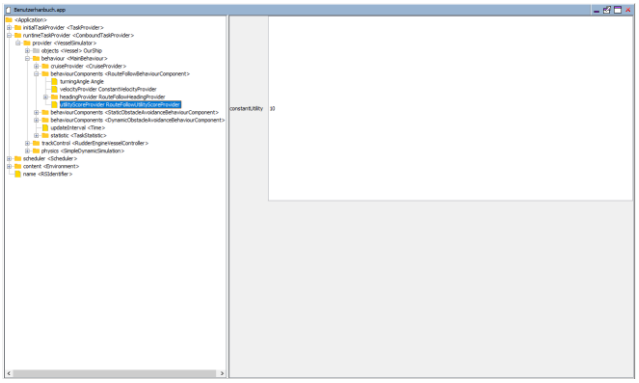




Der *RouteFollowHeadingProvider* beinhaltet eine Reihe von Parametern, welche eingestellt werden müssen. Hier können neben der *updaterate*, dem *turningAngle*, die Einstellungen zu bestimmten Thresholds und zu den EdgePoints getroffen werden.
Der *turningAngle* gibt die maximale Winkeländerung pro Update an. Die *updaterate* gibt an, nach wie vielen Ticks ein Update durchgeführt wird. Die schon erwähnten Thresholds werden durch das *maxThresholdToRoute* und das *thresholdToNextEdgePoint* dargestellt. Diese Werte geben einmal an, wie weit das Schiff von seinem Kurs abweichen darf, bevor es zurücksteuern muss und zusätzlich noch, wann das Schiff den nächsten EdgePoint anfahren soll. *maxThresholdToRoute* gibt an, wie weit die EdgePoints von der Route entfernt sein dürfen.
EdgePoints sind auf der Routenkante mehrere äquidistante Punkte innerhalb der Strecke zwischen zwei Routenpunkten, die angeben, wo das Schiff langfährt. Diese *EdgePoints* werden benötigt, damit ein „Aufperleffekt" nicht auftritt und die Schiffe eine Varianz in ihrer Fahrweise haben. Wie die Abbildung zeigt, kann jedes Schiff individuell die Routenpunkte abfahren, ohne direkt die Routenkanten abzufahren. Der Abstand der EdgePoints

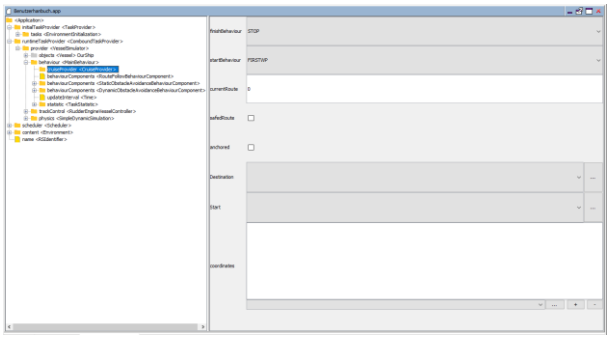
	zur Routenkante wird mit Hilfe einer Normalverteilung innerhalb der angeben Varianz (*variationOfEdgePoint*) vom letzten EdgePoint berechnet. Der Abstand zwischen den EdgePoints in Richtung der Routenkante wird über den Parameter *distanceOfEdgePoints* eingestellt.
### 9.4.3 *ConstantVelocityProvider* 	Der *ConstantVelocityProvider* stellt eine konstante Geschwindigkeit für das jeweilige Verhalten dar. Dies ist nur eine mögliche Variante. Im weiteren Verlauf gibt es noch andere VelocityProvider, wie zum Beispiel den IntentVelocityProvider und den RouteFollowVelocityProvider, um verhaltensangepasste Geschwindigkeiten zu realisieren; wie z.B. einem Überholvorgang.

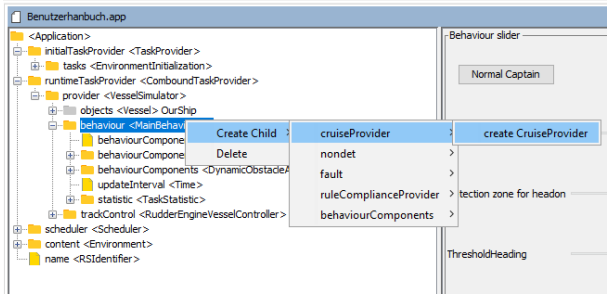
9.4.4 *RouteFollowUtilityScoreProvider*



Beim RouteFollowUtilityScoreProvider wird ein konstanter UtilityScore festgelegt, welcher aussagt, bei welchem UtilityScore-Wert das *RouteFollowBehaviour* angewandt wird.

9.4.5 Cruise Provider





Der CruiseProvider wird über einen Rechtsklick auf das MainBehaviour erstellt und entsprechend der Abbildung ausgewählt. Dieser ermöglicht die Option, aus einem vorhandenen Routengraphen eine Route auszuwählen und dem jeweiligen Schiff zuzuordnen. Dabei beinhaltet der CruiseProvider noch spezielle Funktionen. Im CruiseProvider wird festgelegt, was am Anfang bzw. am Ende einer Route passieren soll. Die grafische Oberfläche wird in der nebenstehenden Abbildung dargestellt.

Der CruiseProvider bietet einen Cruise an. Dieser Cruise besteht aus mehreren Routen.

Zusätzlich können noch Einstellungen zur Route getroffen werden. Durch aktivieren von *safedRoute* wird die Route aus der VoyageCharacteristic genommen. Wenn diese nicht aktiviert ist, wird die Route aus den getroffenen Koordinaten berechnet.

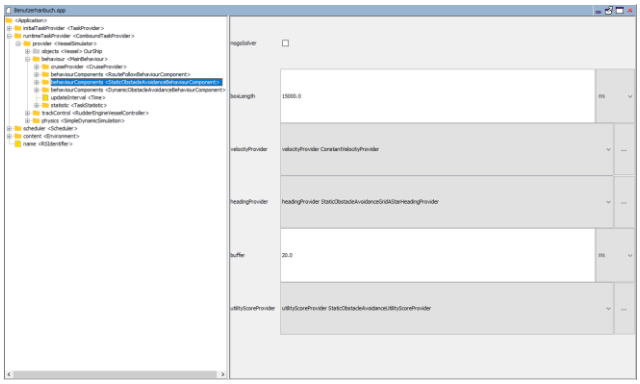
In dem Bereich *Coordinate* kann eine Liste angegeben werden, aus der die Route berechnet wird.

Durch Anchored kann bestimmt werden, ob das entsprechende Schiff vor Anker liegt.

Tabelle 82: Das Verfolgen von Routen

9.5 Statische Hindernisse ausweichen

9.5.1 *StaticObstacleAvoidanceBehaviourComponent*



Die StaticObstacleAvoidanceBehaviour-Component *(statische Hindernisse ausweichen)* ist ähnlich aufgebaut wie die dynamische Komponente. Bei der statischen Komponente gibt es zusätzlich noch die Möglichkeit, den *nogoSolver* zu etablieren.

Die *boxLength* gibt einen Bereich um das Schiff an. Hierbei wird unterschieden zwischen einer festen inneren Box und einer äußeren Box um das Schiff. Wenn der Bereich der inneren Box durch ein Schiff überfahren wird, wird eine erneute Abfrage durchgeführt, um statische Hindernisse in der äußeren Box zu erkennen. Der *buffer* addiert zur Größe des Schiffes den eingegeben Wert. Dieser Puffer wird in den NogoSolver integriert, um eine mögliche Sicherheit zu gewährleisten, da dieser nur mit exakten Werten des Schiffes arbeitet.

Tabelle 83: Ausweichen von statischen Hindernissen_1

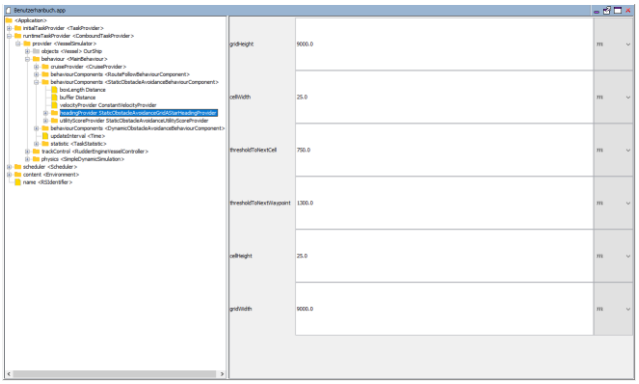
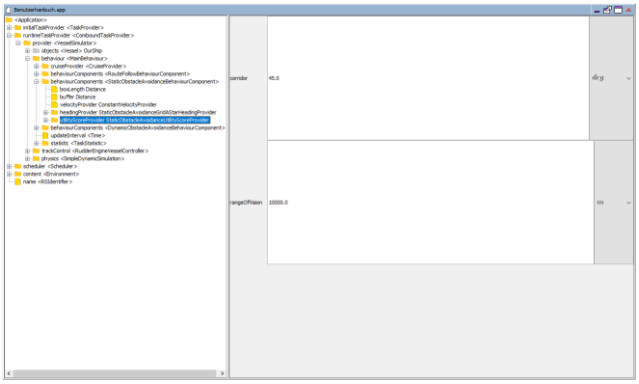
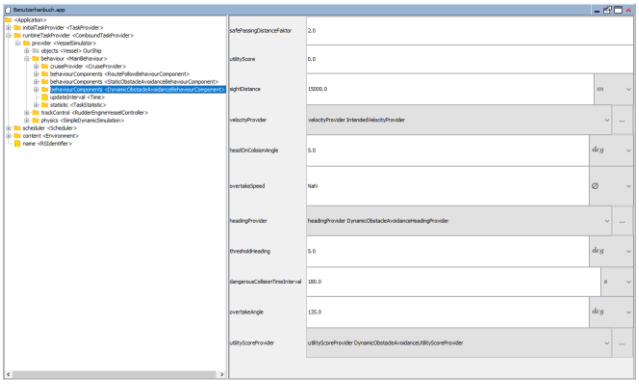
9.5.2 StaticObstacleAvoidanceGridAStarHeadingProvider 	Bei dem *StaticObstacleAvoidanceGridAStarHeadingProvider* wird anders als beim *RouteFollowHeadingProvider* ein Grid erstellt. In diesem Grid werden die Zellen markiert, die mit einem Hindernis auf der Karte kollidieren. Dafür wird die Grid-Größe mit einer Höhe und einer Breite angeben. Die Parametrisierung ist in der Abbildung dargestellt. Durch den *thresholdToNextCell* wird bestimmt, wann die nächste Zelle angesteuert wird.
9.5.3 StaticleObstacleAvoidanceUtilityScoreProvider 	Der *StaticObstacleAvoidanceUtilityScoreProvider* beinhaltet die Angabe zu einem Sichtfeld für das Schiff. Dabei stellt der *corridor* die Winkelgröße des Sichtfeldes ein und die *rangeOfVision* die Länge des Sichtfeldes.

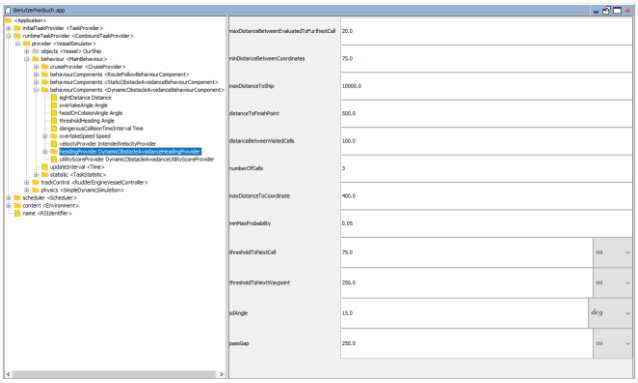
Tabelle 84:Ausweichen von statischen Hindernissen_2

9.6 Dynamische Hindernisse ausweichen

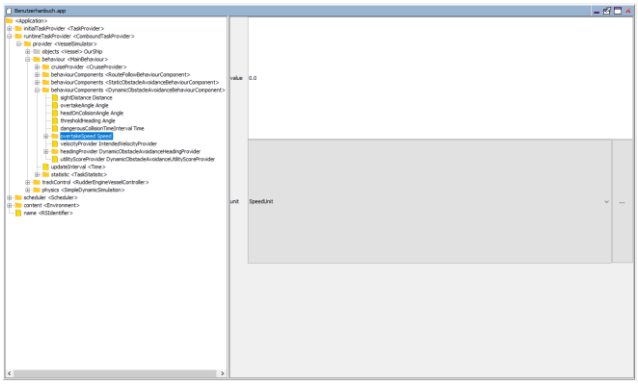
9.6.1 DynamicObstacleAvoidanceBehaviour-Component



Die *DynamicObstacleAvoidanceBehaviourComponent*, die in der Abbildung gezeigt ist, dient zum Ausweichen von dynamischen Hindernissen.
Hier werden folgende Eigenschaften behandelt und parametrisiert:
Um die Reaktionen des Schiffes zu verwalten, sind die *sightDistance*, die *headOnCollisionAngle*, das *thresholdHeading* und die *overtakeAngle* parametrisierbar.
Die *sightDistance* gibt die Sichtweite des Schiffes, also die Größe der Distanz an, in welcher die Schiffe erkannt werden.
Der *headOnCollisionAngle* gibt an, in welchem Bereich eine HeadOn Situation, also zwei auf sich zu fahrende Schiffe, zu behandeln sind. Hierfür wird ein Winkel eingegeben, um ein HeadOn Bereich zu definieren. Dabei handelt es sich bei der Eingabe um den halben Winkel. Der HeadOn Bereich beträgt also 0° +/- eingegeben Winkel.
Das *thresholdHeading* regelt die Winkelgröße, in der sich das Heading des Schiffes verändern kann, sodass diese Situation als eine betrachtet wird, die eine weitere Maßnahme erfordert, wie beispielsweise das Einleiten eines Manövers.
Zuletzt gibt der *overtakeAngle* den Winkel an, in dem ein Überholmanöver er-

	kannt wird. Ist also ein Schiff in dem *overtakeAngle* eines anderen, wird dieses als Überholvorgang gewertet.
9.6.2 DynamicObstacleAvoidanceHeadingProvider 	Die Abbildung zeigt die Parametrisierbarkeit des DynamicObstacleAvoidanceHeadingProvider dar. Hierbei können folgende Einstellungen vorgenommen werden: Die Gridgröße kann festgelegt werden, wobei sich das Schiff im Mittelpunkt des Grids befindet. Die Zellengröße kann bestimmt werden. Die minProbability wird hier definiert und beschreibt, ab welcher Wahrscheinlichkeit ein Schiff sich nicht mehr in einer Zelle befindet. Auch hier ist eine Eingabe für die thresholdToNextCell möglich. Der einzugebende sdAngle regelt die vorhergesagte mögliche Abweichung des anderen Schiffes, dass von seiner vorhergesagten Bahn unter Angaben eines Winkels berechnet wird. Diese Berechnung erfolgt lediglich, wenn das Schiff geradeaus fährt. Der passGap gibt den Passierabstand zum anderen Schiff an.

9.6.3 overtakeSpeed



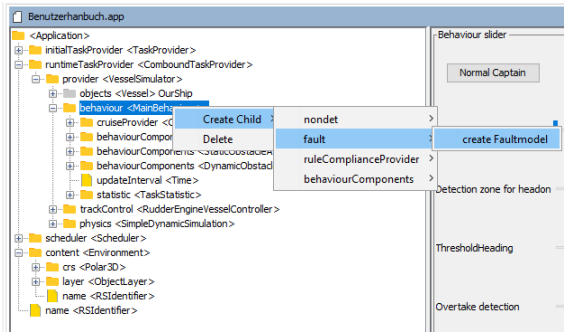
Die Geschwindigkeit wird über einen eigenen Bereich variiert.

9.6.4 DynamicObstacleAvoidanceUtilityScoreProvider

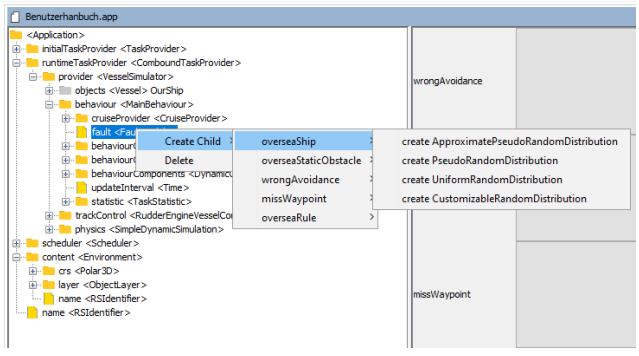
Der DynamicObstacleAvoidanceUtilityScoreProvider wird benötigt, um dynamischen Hindernissen auszuweichen. Dieser benötigt keine weiteren Parametrisierungen und hat deswegen keine grafische Oberfläche.

Tabelle 85: Ausweichen von dynamischen Hindernissen

9.7 Fehlermodell

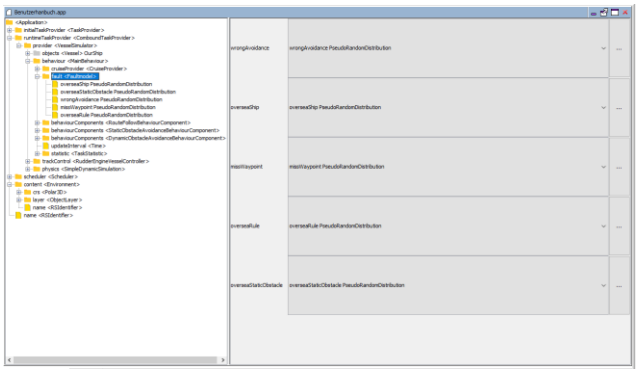


Um das Fehlermodell zu erstellen, wird über das Content-Menü des <MainBehaviour> der entsprechende Menüpunkt ausgewählt.



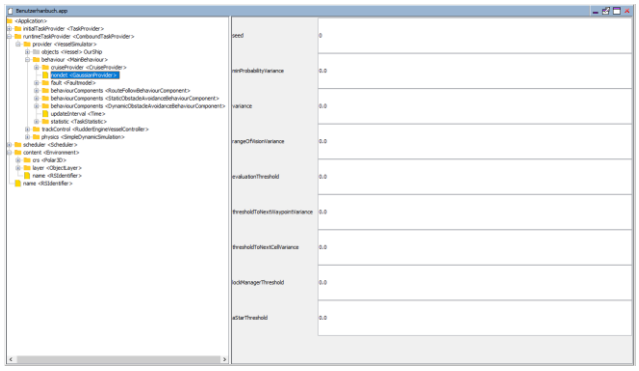
Anschließend kann zwischen verschiedenen Kategorien ein Fehlermodell eingestellt werden.

- overseaShip
- overseaStaticObstacle
- wrongAvoidance
- missWaypoint
- overseaRule



Die zugeordneten Fehler der einzelnen Kategorien werden im <Faultmodel> angezeigt.

9.7.1 Nicht Determinismus



Um den Nicht-Determinismus zu generieren, werden folgende Werte parametrisiert:

- Seed = Startwert der Zufallszahl (0 - unendlich)
- rangeOfVisionVariance = Dieser Wertebereich bezieht sich auf die Gauß-Verteilung. Der einstellbare Wert zwischen 0 - 200 wird mit einer Wahrscheinlichkeit von 68% realisiert
- evaluationThreshold = Dieser Wertebereich liegt zwischen 0.0 und 1.0. Der Wert 1.0 interpretiert die immer zufällige Auswahl. Der Wert 0.0 immer den höchsten Wert. 0 - 1= Wie weit entfernt vom utilityScore kann er noch genutzt werden.
- thresholdToNextWaypointVariance = hier wird der Wert eingestellt, ab wann der nächste Wegpunkt als erreicht angesehen wird. Geeignete Werte liegen hier zwischen 0 und 300.
- thresholfToNextCellVariance = Die einzustellende Varianz liegt zwischen 0 und 150 und definiert, ab wann die nächste Zelle anvisiert werden soll
- lockmanagerThreshold= Die Werte zwischen 0 und 1 bestimmen in diesem Feld, ab wann von einem Verhalten, wie das

	Ausweichen von statischen Hindernissen, auf beispielweise das Ausweichen von dynamischen Hindernissen gewechselt wird • aStarThreshold = Da die Zellen im Grid immer mit dem geringsten Wert zuerst evaluiert werden, kann hier von dieser Thematik abgewichen werden. Da der geringste Wert g aus f (bisherige Weg zur Zelle) + h (Heuristik) berechnet wird, darf hier g um den Threshold abweichen

Tabelle 86: Das Fehlermodell

9.8 Ortsabhängige Regeln befolgen

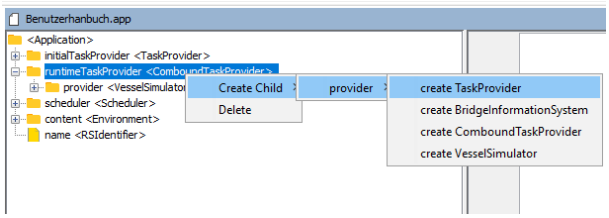
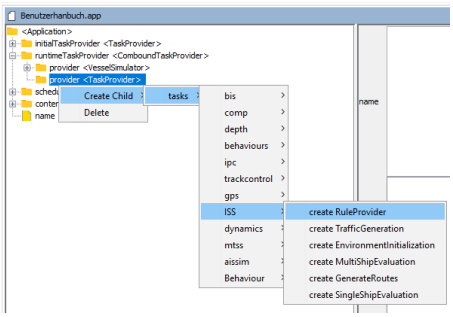
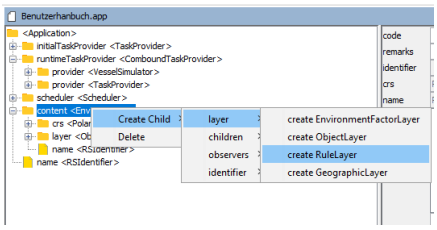
	Um die Funktion der ortsabhängigen Regeln befolgen zu können, muss ein TaskProvider im runtimeTaskProvider erstellt werden.
 	Anschließend wird der RuleProvider benötigt, um im nächsten Schritt den RuleLayer erstellen zu können.

Tabelle 87: Erstellen von Regeln_1

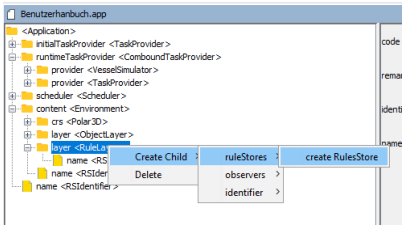
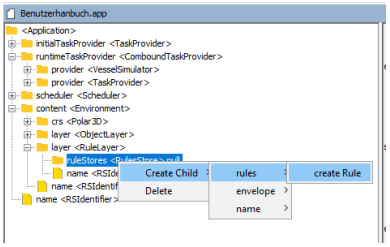
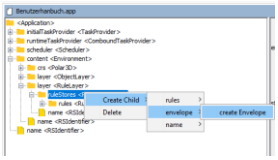
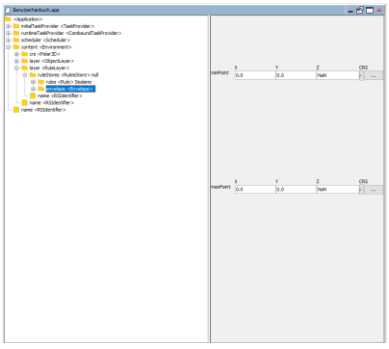
 	Der RuleStore muss anschließend erstellt werden, um auf die einzelnen Regeltypen zugreifen zu können.
 	Aus Performancegründen ist es von Vorteil, ein Envelope anzulegen. Dieses ermöglicht, einen Bereich über XY-Koordinaten anzulegen, um der zu erstellenden Regel einen gültigen Bereich zuzuordnen.

Tabelle 88: Erstellen von Regeln_2

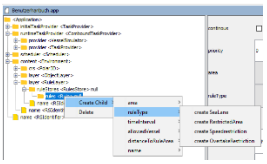
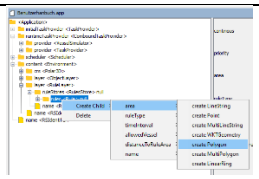
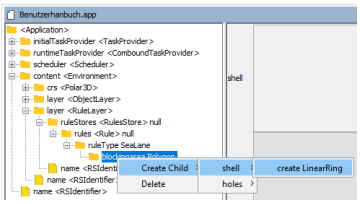
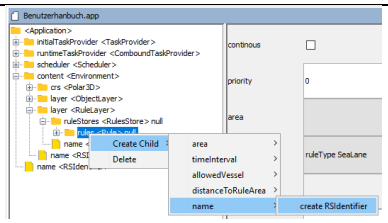
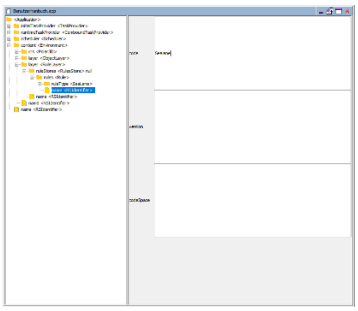
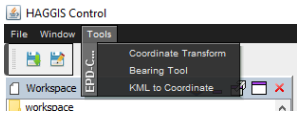
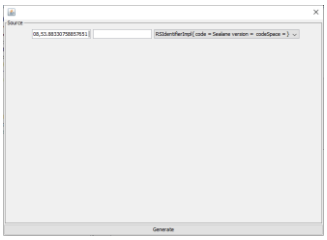
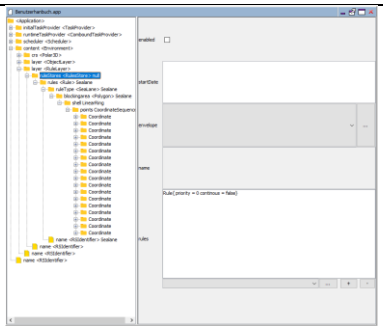
	Vier Regeltypen sind vorimplementiert. SeaLane, RestrictedArea, Speedretriction und die OvertakeRestriction. Alle Regeln werden über die manuelle Eingabe von Koordinaten erstellt. Außer die SeaLane, diese kann über eine Funktion KML to Coordinate erstellt werden, die im späteren noch vorgestellt wird. Als Beispiel wird in diesem Handbuch eine SeaLane mit Hilfe der Funktion KML to Coordinate angelegt.
	Nachdem der RuleType festgelegt wurde, wird innerhalb der jetzigen SeaLane eine Area als Polygon angelegt.
	Jetzt muss zwischen „shell“ und „holes“ entschieden werden. In diesem Beispiel wird ein „shell“ als LineRing angelegt. In diesem LineRing werden später die Koordinaten erstellt. Manuell wird dieses über das entsprechende Kontextmenü imitiert. Bei einem Polygon müssen die ersten wie auch die letzten Koordinaten dieselben Werte haben, damit das Polygon geschlossen wird. Da hier die Koordinaten automatisch erstellt werden sollen, muss die „Rule“ mit einem Namen definiert werden.

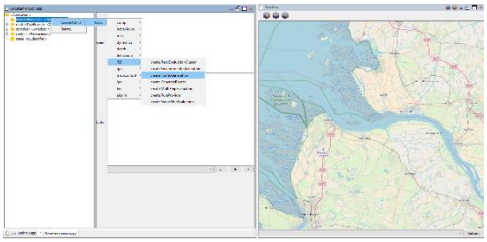
Tabelle 89: Erstellen von Regeln_3

 	RSIdentifier wird angelegt, um anschließend über das Feld „Code“ einen Namen zuzuordnen.
 	Über die HAGGIS Menüreiter kann die Funktion KML to Coordinate ausgewählt werden. Diese ist über Tools ausführbar. Das anschließende Pop-up Fenster ermöglicht Koordinaten von einem Bereich einzufügen. Dieser Bereich kann über OpenSeamap erstellt werden und als eine KML-Datei abgespeichert werden. Die Koordinaten können aus der Datei 1:1 kopiert und unverändert im linken Feld des KML to Coordinate eingefügt werden. Zusätzlich muss der entsprechende RSIdentifier ausgewählt werden. Danach werden über den Botton „Generate“ die Koordinaten als Polygon eingetragen.

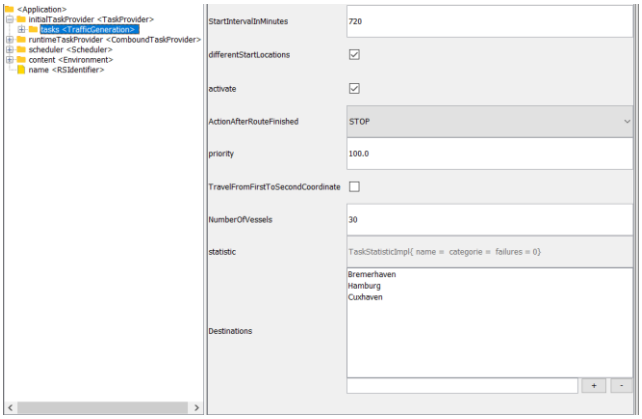


Nach dem aktualisieren der Ordnerstruktur sind die Koordinaten in der Menüstruktur unter dem Ordner „points CoordinateSequence“ zu erkennen.

9.8.1 TrafficGenerator



Der TrafficGenerator erstellt durch vorherige Parameter ein Schiff mit gewünschter Route zwischen zwei ausgewählten Häfen. Dazu wird im initialTaskProvider die TrafficGeneration generiert.



In dem erstellten Task wird der Start und Zielhafen über das Eingabefeld „Destinations“ eingegeben und mit dem „+“ Button hinzugefügt.
Im Feld NumberOfVessel werden die Anzahl der Schiffe eingegeben.

Beim Aktivieren des Feldes DifferentStartLocations werden erzeugte Schiffe am selben Hafen nicht an derselben Stelle, sondern in einem Umkreis von 100m zufällig verteilt.

Das Feld TravelFromFirstToSecondCoordinate sorgt dafür, dass ein erzeugtes Schiff von der ersten zur zweiten Destination fährt, was dafür genutzt werden kann um ein einzelnes Schiff

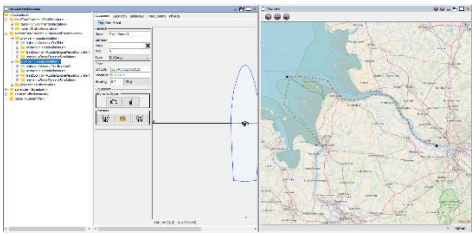
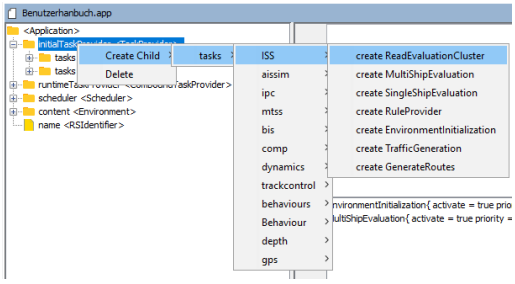
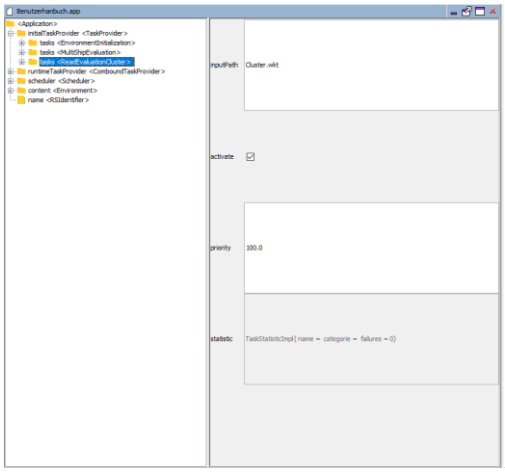
	zwischen zwei bestimmten Häfen fahren zu lassen. Die Auswahl ActionAfterRouteFinished sagt aus, was das Schiff nach Erreichen des Ziels tun soll. Nach dem starten der Simulation und dem refreshen der Map, werden die Schiffe und die Route erstellt.
	Das Schiff ist durch Starten der Simulation im runtimeTaskProvider eingefügt und die Route in der Karte angelegt worden.

Tabelle 90: Erstellen von Regeln und Nutzen des Traffic Generators

9.9 Evaluation

9.9.1 ReadEvaluationCluster





Die ReadEvaluationCluster ist die task, die es ermöglicht. geclusterte, historische Daten in einer konkaven Hülle anzuzeigen. Für die grafische Anzeige wird ein Polygon erstellt.

Diese .wkt-Datei wird über den inputPath angegeben.

Diese Datei wird in folgender Ordnerstruktur hinterlegt:

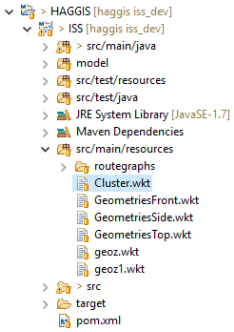


Tabelle 91: Einfügen der Evaluation

10 Vorgehen

Nach dem durchgeführten Kick-off Meeting, sowie die Einarbeitung und Vorstellung der Seminarthemen, wurde mit der Aufnahme des Ist-Zustandes des Projektes begonnen. Hierbei wurde eine gezielte Einarbeitung in den Quellcode sowie das zurechtfinden in der entsprechenden Umgebung vorgenommen. Des Weiteren sind Anforderungen definiert worden.

Zu Beginn wurde die Idee verfolgt, in Ausbaustufen zu arbeiten und diese als Meilensteine anzusehen. Die Ausbaustufen wurden kontinuierlich verbessert und letztendlich durch Userstories ersetzt. Innerhalb dieser Stories wurden die Ergebnisziele definiert.

Nachdem der Projektplan erstellt, die Anforderungen definiert und abgenommen wurden, ist der erste Meilenstein einer Userstory verfasst worden.

Anschließend ist innerhalb der Projektgruppe in Teilgruppen gearbeitet worden. Die Teilergebnisse wurden nach einem festgelegten Zeitraum zusammengetragen und auf eventuelle Änderungen, Verbesserungen oder auch notwendige Anpassungen geprüft worden.

10.1 Themeneinführung

Bei der Auswahl der Projektgruppe wurde zu Anfang ein Paper ausgegeben, die die Projektgruppe Intelligente Schiffssimulation (ISS) als einen Methodeneinsatz beschreibt, der sich mit dem maschinellen Lernen und der künstlichen Intelligenz, die für die agentenbasierte Simulation, von autonomen Schiffen in der Simulationsumgebung relevant ist, auseinandersetzt.

In der Kooperation mit der Projektgruppe Mate wurden Ansätze und Anwendungen vorgeschlagen, die für ein maschinelles Lernen möglich sein sollten. Unter anderem waren hier das Lernen des Verkehrssystems als Gesamtes und das automatische/intelligente Fahren des Gesamtverkehrs unter der Berücksichtigung der einzelnen Schiffsmodelle beschrieben (vgl. Projektgruppe Intelligente Schiffssimulation ISS).

Unter der Verwendung von HAGGIS, das ein Bestandteil von eMIR ist, ist eine Simulationsumgebung für maritimen Verkehrssituation gegeben. In dieser Umgebung soll ein Verhalten implementiert werden, welches um eine intelligente Komponente erweitert wird. Dabei ist der Begriff so zu definieren, dass das Schiff bzw. der Agent einem realitätsgetreuen Verhalten gleichkommt.

Dabei sind verschiedene Kapitänstypen zu berücksichtigen, die zum einem Regeln befolgen können, aber auch in der Large sind, die gegeben Regeln zu missachten. Des Weiteren muss das Verhalten das Ausweichen von statischen und dynamischen Hindernissen gewährleisten. Hierfür ist die Funktionsweise des MTCAS und die KVR‘s berücksichtigt worden.

Diese Projektgruppe hat auf Basis des aktuellen Standes der MTSS-Projektgruppe ihre Arbeit begonnen.

10.2 Gruppen und Aufgabenverteilung

Zu Beginn wurden die wichtigsten Rollen, wie die des Projektleiters vergeben. Die Rolle des Moderators, wie auch die des Protokollanten wurde nach Alphabet der Reihe nach verteilt. Ein angemessener Strafenkatalog festigte das Verhalten der einzelnen Teilnehmer. Fehlverhalten des Protokollanten wurde durch das wiederholte Schreiben des nächsten Protokolls vereinbart.

Die Rollen von Entwickler und Tester sind zum Teil ineinander übergegangen. Nach dem ersten Release sind durch festgelegte Tester gezielte Überprüfungen der zu erwarteten Funktionen durchgeführt worden.

10.3 Projektmanagement

Unter dem Vergleich verschiedener Vorgehensmodelle ist das Scrum-Modell die Grundlage des Projektes. Dieses agile, iterative, inkrementelle Verfahren erforderte ein Backlog, dass die Kernaufgabe erfasst. Hier wurde ein Projektplan mit der Zeitplanung hinterlegt. In dieser Planung waren entsprechende Anforderungen definiert. Weiterhin wurden Projektziele festgehalten. Vor jedem Sprint, welcher auch mal länger als eine Woche ging, wurden die benötigten User-Stories verfasst, sodass eine anschließende Gruppeneinteilung zu den einzelnen Themengebiete, erfolgten konnte.

Tabelle 92 zeigt die erste Version des Projektplans.

Vorgangsname	**Dauer**	**Anfang**	**Ende**
Seminarphase	17 Tage	06.04.17 08:00	28.04.17 17:00

Seminarphase abgeschlossen	0 Tage	28.04.17 17:00	28.04.17 17:00
Projektplanung	20 Tage	01.05.17 08:00	26.05.17 17:00
Projektplanung abgeschlossen	0 Tage	26.05.17 17:00	26.05.17 17:00
Anforderung	20 Tage	29.05.17 08:00	23.06.17 17:00
Anforderungsdokument abgesegnet	0 Tage	23.06.17 17:00	23.06.17 17:00
Auslesen von Objekten aus Seekarten	27 Tage	26.06.17 08:00	01.08.17 17:00
Stufe 1: Benötigte Objekten können aus Seekarten ausgelesen werden	0 Tage	01.08.17 17:00	01.08.17 17:00
Anpassung des Routengraphen	20 Tage	02.08.17 08:00	29.08.17 17:00
Stufe 2: Der Routengraph enthält Kanteninformationen und kann aus historischen Daten erweitert werden	0 Tage	29.08.17 17:00	29.08.17 17:00
Routenvalidierung	22 Tage	30.08.17 08:00	28.09.17 17:00
Stufe 3: Invalide Routen werden erkannt und können ausgeschlossen werden.	0 Tage	28.09.17 17:00	28.09.17 17:00
Trajektorienoptimierung	42 Tage	30.08.17 08:00	26.10.17 17:00
Stufe 4: Trajektorien werden zur Laufzeit optimiert	0 Tage	26.10.17 17:00	26.10.17 17:00
Prototyp: Vorstellung der Routengraphen mit Kanteninformationen und einer Routenvalidierung anhand eines Szenarios	0 Tage	28.09.17 17:00	28.09.17 17:00
Behandlung statischer Hindernisse	20 Tage	27.10.17 08:00	23.11.17 17:00
Stufe 5: Statische Hindernisse werden erkannt und automatisch Umfahren	0 Tage	23.11.17 17:00	23.11.17 17:00
Behandlung anderer Schiffe	42 Tage	27.10.17 08:00	02.01.18 17:00

Stufe 6: Andere Schiffe werden erkannt und automatisch nach KVRs behandelt	0 Tage	02.01.18 17:00	02.01.18 17:00
Ausnahmesituationen	40 Tage	03.01.18 08:00	27.02.18 17:00
Stufe 7: Ausnahmesituationen werden erkannt und automatisch behandelt	0 Tage	27.02.18 17:00	27.02.18 17:00
Dokumentation	10 Tage	28.02.18 08:00	13.03.18 17:00
Endpräsentation erstellen	10 Tage	28.02.18 08:00	13.03.18 17:00
Endpräsentation vorstellen	1 Tag	14.03.18 08:00	14.03.18 17:00
Endabgabe	1 Tag	14.03.18 08:00	14.03.18 17:00
PGISS Beendet	0 Tage	14.03.18 17:00	14.03.18 17:00

Tabelle 92:Erste Version des Projektplanes

Der zu Beginn des Projektes erstellte Projektplan, dargestellt in der Tabelle 92, konnte durch auftretende Fehler innerhalb des Projektes, so wie Änderungen der Projektanforderungen ausgehend der Betreuer nicht eingehalten werden. Dieser Projektplan wurde im Verlauf des Projektes angepasst.

Im ersten Projektplan wurden Ausbaustufen erarbeitet, die in den weiteren Versionen ersetzt worden waren.

Der folgende Projektplan, dargestellt in der Tabelle 93, ist die finale Version. Nicht nur die Ausbaustufen wurden ausgetauscht, unter anderem sind auch die Refactor-Wochen hinzugefügt.

Name	Dauer	Anfang	Ende
Seminarphase	17 Tage	06.04.17 08:00	28.04.17 17:00
Seminarphase abgeschlossen	0 Tage	28.04.17 17:00	28.04.17 17:00

Projektplanung	**60 Tage**	**01.05.17 08:00**	**21.07.17 17:00**
Projektplanung abgeschlossen	0 Tage	21.07.17 17:00	21.07.17 17:00
Auslesen von Objekten aus Seekarten	15 Tage	24.07.17 08:00	11.08.17 17:00
Benötigte Objekten können aus Seekarten ausgelesen werden	0 Tage	11.08.17 17:00	11.08.17 17:00
Agentendesign erstellen	10 Tage	24.07.17 08:00	04.08.17 17:00
Verschiedene Agentenarchitekturen wurden erstellt	0 Tage	04.08.17 17:00	04.08.17 17:00
Routenkanten verfolgen durch Agenten	10 Tage	07.08.17 08:00	18.08.17 17:00
Agenten können Routen verfolgen	0 Tage	18.08.17 17:00	18.08.17 17:00
Behandlung statischer Hindernisse	20 Tage	14.08.17 08:00	08.09.17 17:00
Statische Hindernisse werden erkannt und automatisch Umfahren	0 Tage	08.09.17 17:00	08.09.17 17:00
Sensordaten werden vom Agenten Empfangen	10 Tage	21.08.17 08:00	01.09.17 17:00
Begegnungsbehandlung von dynamischen Hindernissen	**25 Tage**	**11.09.17 08:00**	**13.10.17 17:00**
Dynamische Hindernisse können nach KVRs behandelt werden	0 Tage	13.10.17 17:00	13.10.17 17:00
Präsentationsvorbereitung des Prototypen	10 Tage	16.10.17 08:00	27.10.17 17:00
Vorstellung des Prototypen	0 Tage	27.10.17 17:00	27.10.17 17:00
Refactor Week #1	14 Tage	30.10.17 08:00	16.11.17 17:00
Beta Release	0 Tage	16.11.17 17:00	16.11.17 17:00
Fehlermodel einbauen und Toleranzparameter hinzufügen	15 Tage	17.11.17 08:00	07.12.17 17:00

Agenten machen Fehler und verhalten sich innerhalb von Toleranzen	0 Tage	07.12.17 17:00	07.12.17 17:00
Verhalten definieren (Charaktereigenschaften)	10 Tage	30.11.17 08:00	13.12.17 17:00
Oberfläche zur Definition von Verhalten ist vorhanden	0 Tage	13.12.17 17:00	13.12.17 17:00
Ortsabhängig Regeln befolgen	19 Tage	08.12.17 08:00	11.01.18 17:00
Agenten können Ortsabhängige Regeln befolgen	0 Tage	11.01.18 17:00	11.01.18 17:00
Herausfinden von Trafficpatterns aus historischen Daten	15 Tage	12.01.18 08:00	01.02.18 17:00
Muster von Trajektorien aus historischen Daten erstellt	0 Tage	01.02.18 17:00	01.02.18 17:00
Evaluation des Verhaltens eines einzelnen Agenten	15 Tage	12.01.18 08:00	01.02.18 17:00
Die Trajektorien der Agenten wurden auf realitätstreue überprüft	0 Tage	01.02.18 17:00	01.02.18 17:00
Evaluation des Gesamtverkehrs	15 Tage	02.02.18 08:00	22.02.18 17:00
Historische und simulierte Daten wurden verglichen	0 Tage	22.02.18 17:00	22.02.18 17:00
Refactor Week #2	10 Tage	23.02.18 08:00	08.03.18 17:00
Software Endabgabe	0 Tage	08.03.18 17:00	08.03.18 17:00
Dokumentation	15 Tage	09.03.18 08:00	29.03.18 17:00
Dokumentation ist fertiggestellt	0 Tage	29.03.18 17:00	29.03.18 17:00
Endpräsentation erstellen	15 Tage	09.03.18 08:00	29.03.18 17:00
Endpräsentation vorgestellt	0 Tage	29.03.18 17:00	29.03.18 17:00
Endabgabe	1 Tag	30.03.18 08:00	30.03.18 17:00

PGISS Beendet	0 Tage	30.03.18 17:00	30.03.18 17:00

Tabelle 93: Finale Version des Projektplanes

Die Projektplanung ist auf Grundlage der einzelnen Userstories entwickelt worden, die mit entsprechenden Meilensteinen abgeschlossen wurden.

Allerdings hatten auch unerwartet Performanceschwierigkeiten einen negativen Einfluss auf die Zeitplanung. Eine immer wieder Überarbeitung des A-Stern-Algorithmus war erforderlich, um den Anforderungen gerecht zu werden.

10.4 Gruppentreffen

Die Gruppentreffen mit den Betreuern fand zuerst freitags im regelmäßigen Zyklus statt, welches im Laufe des Projektes auf den Donnerstag verlegt wurde.

Die Teilgruppen haben sich je nach Themengebiet zusammengefunden. Die Gruppenstärke variierte hierbei zwischen 2 und 5 Teilnehmern. Je nach Bedarf sind diese Treffen individuell vereinbart und gestaltet worden. Projekttreffen fanden in den Teilgruppen überwiegend von Montag bis Mittwoch von 9:00 Uhr bis 17:00 Uhr statt.

In den Teilgruppen wurden an Konzepten und den Ideen gemeinsam gearbeitet. Die Implementierung ist dann über den an den Beamer angeschlossenen Laptop erfolgt, wobei eine schnelle Fehlersuche und Verbesserungsvorschläge erfolgen konnten, ähnlich wie bei dem Vorgehensmodell von „Paarprogrammierung“. Nebenher sind hierbei die Konzepte verschriftlicht worden, die jedoch im Laufe der Arbeit immer mal wieder überarbeitet bzw. generell verworfen und noch mal neu erstellt worden sind. Dieses war eine Konsequenz aus den gesammelten Erfahrungen und Tests, die während der Implementierung gesammelt worden sind. Hieraus sind effizientere und effektivere Methoden gewählt worden.

10.5 Fazit

Rückblickend auf das Jahr wurde am Anfang des Projektes die Methode gewählt, nach Scrum/Kanban zu arbeiten. Im Laufe der Arbeit ist das Scrum Modell in einer vereinfachten Variante durchgeführt worden. Dabei wurde sich auf die wesentlichen agilen Aspekte konzentriert, sodass auf die Wünsche und Änderung der Productowner eingegangen werden konnte.

Der zeitliche Verlauf des Projektes ist im Wesentlichen aufgegangen. Vereinzelt mussten jedoch Anpassungen in den Zeiten, wie auch im Ablauf des Projektes vorgenommen werden.

Die Zielsetzung der Vermeidung von Kollisionen haben wir erfüllt. Die Gefahrenerkennung von potenziellen Kollisionen wie mit statischen und auch dynamischen Hindernisse sind erfolgreich implementiert worden. Beispielregeln der KVR‘s sind erstellt worden und können um weitere ortsabhängigen Regeln ergänzt werden. Bei dem Ausweichen von dynamischen Hindernissen ist die Trajektorienplanung erfolgreich umgesetzt worden. Selbstverständlich wurden auch unterschiedliche Charaktereigenschaften von verschiedenen Kapitänstypen berücksichtigt. Nicht nur vorkonfigurierte Typen, sondern auch individuelle Anpassungen können über eine entsprechende Nutzeroberfläche angepasst werden.

In der folgenden Auflistung sind die Projektrahmenziele und die Projektergebnisziele aufgelistet und mittels einer Checkbox ist dargestellt, ob die jeweiligen Ziele erfüllt wurden:

10.5.1 Projektrahmenziele

☒ R1 Es dürfen keine Funktionalitäten entwickelt werden, die bereits von HAGGIS bereitgestellt werden. Bestehende Komponenten müssen verwendet und ggf. angepasst werden.

☒ R2 Die MTS muss als grundlegende Basis der maritimen Simulationsumgebung genutzt werden.

☒ R3 Jedes Schiff muss ein Agent sein, dessen Verhalten durch Steuerungskomponenten umgesetzt wird.

☒ R4 Die MTS muss um Komponenten erweitert werden, die das Verhalten abbilden.

☒ R5 Die Deutsche Bucht muss als Anwendungsgebiet für die MTS verwendet werden.

☒ R6 Anker- und Anlegevorgänge sollen nicht betrachtet werden.

☒ R7 Das Verhalten muss so gestaltet werden, dass Umwelteinflüsse, wie Strömung, Wind und Änderungen am Dynamikmodell des Schiffes in die MTS integriert werden können, und die entwickelten Verhalten damit weiterhin funktionieren.

☒ R8 Sämtliches Verhalten der Agenten soll nicht-deterministisch erfolgen.

☒ R9 Die entwickelte Software muss bis zum 05.04.2018 geliefert werden.

☒ R10 Die Softwaredokumentation muss bis zum 05.04.2018 geliefert werden.

☒ R11 Der PG-Abschlussbericht muss bis zum 05.04.2018 geliefert werden.

☒ R12 Die Abschlusspräsentation muss bis zum 05.04.2018 stattgefunden haben.

☒ R13 Ein erster Prototyp muss bis zum 01.11.2017 entwickelt und präsentiert werden.

☒ R13.1 Schiffe müssen Routen verfolgen.

☒ R13.1.1 Die Route kann durch Hindernisse führen.

☒ R13.2 Schiffe müssen dynamischen Hindernissen aus Seekarten (z.B. Bojen, Windparks, …) ausweichen/vermeiden können.

☒ R13.3 Schiffe müssen dynamischen Hindernissen ausweichen/vermeiden können.

☒ R13.3.1 Beim Ausweichen von dynamischen Hindernissen können hier bereits KVR-konforme Manöver ausgeführt werden.

☒ R14 Es müssen überprüfbare Meilensteine erstellt werden.

☒ R14.1 Jeweils zum nächsten Meilenstein müssen die umzusetzenden Anforderungen und Stories definiert sein.

☒ R14.1.1 Die Anforderungen und Stories müssen von den Betreuern bestätigt werden.

☒ R14.2 Die Meilensteine müssen von den Betreuern bestätigt werden.

10.5.2 Projektergebnisziele

☒ E1 In der Simulation muss, durch mehrere Agenten in einem bestimmten Seegebiet, Verkehr nachgebildet werden.

☒ E1.1 Der Gesamtverkehr muss auf Realitätstreue geprüft werden.

☒ E1.2 Das Verhalten eines einzelnen Agenten muss auf Realitätstreue geprüft werden.

☒ E2 Ein Agent muss Input über die jeweilige Schiffssensorik erhalten (Ausschluss des „God mode" - dem Agenten stehen nicht alle Informationen der gesamten Simulation zur Verfügung).

☒ E3 Ein Agent muss mindestens ein Standardmodell für dessen Verhalten haben.

☒ E4 Das Befahren von Routen mittels Bahnen muss durch das Verhalten beeinflusst werden.

☒ E5 Das Verhalten muss durch Toleranzwerte in der MTS (HAGGISControl) individualisierbar sein.

☒ E5.1 Passierabstände müssen variieren können.

☒ E5.2 Der Zeitpunkt der Manövereinleitung muss variieren können.

☐ E5.3 Die Geschwindigkeitstypen der Agenten müssen variieren können.

☐ E5.4 Die Beschleunigungstypen der Agenten müssen variieren können.

☒ E5.5 Die Art und Weise der Verfolgung von Routenkanten muss variieren können.

☒ E5.6 Die Befolgung von KVR-Regeln muss variieren können.

☒ E5.7 Allen variierenden Parametern soll jeweils ein Fehlermodell unterliegen.

☒ E6 Das Verhalten eines Agenten muss einem Fehlermodell unterliegen.

☒ E7 In der MTS muss ein Agent statischen Hindernissen ausweichen können.

☒ E7.1 Ein Agent muss NoGo-Areas ausweichen können. (Nutzung des NoGo-Solvers)

☒ E7.2 Ein Agent muss Seekartenobjekten ausweichen können.

☒ E8 In der MTS muss eine Begegnungsbehandlung für dynamische Objekte stattfinden.

☒ E8.1 In der MTS müssen Agenten entsprechend der KVR auf andere kreuzende Schiffe reagieren können und selbst andere Schiffe KVR-konform kreuzen können.

☒ E8.2 In der MTS müssen Agenten entsprechend der KVR Überholmanöver durchführen und darauf reagieren können.

☒ E8.3 In der MTS müssen Agenten entsprechend der KVR Abstand zu anderen Schiffen halten können.

☒ E9 Der Agent muss ortsabhängig Regeln befolgen.

☒ E9.1 Es muss ein erweiterbares Modell für Seeschifffahrtsregeln entwickelt werden, in dessen Rahmen die KVR umgesetzt werden.

☒ E10 Es muss eine Startkonfiguration aller Schiffe im Verkehrsgebiet erzeugt werden.

☒ E10.1 Jeder Agent muss sich eine Route aus dem Routengraphen generieren können.

☒ E10.1.1 Es muss ein einfacher Routengraph genutzt werden, der in HAGGIS vorhanden ist.

☐ E10.1.2 Es kann eine Routenvalidierung stattfinden, aber nur sofern dies für die Einhaltung des Gesamtziels - der Realitätstreue - notwendig ist.

Aus den Listen ist erkennbar, dass alle Projektrahmenziele erfüllt worden sind. Bei den Ergebniszielen sind drei Ziele nicht erfüllt worden. Unter anderem das Ergebnisziel E5.3. Dieses ist nicht zu 100% erfüllt, da die maximale Geschwindigkeit zwar einstellbar ist und im statischen Modell auch berücksichtigt wird, jedoch nicht im dynamischen Modell. Der Kapitän hat nicht die Möglichkeit, bei einem Ausweichmanöver seine Geschwindigkeit zu variieren. Das Ergebnisziel E5.4 wurde während unserer Arbeit nicht weiter verfolgt.

Das Ergebnisziel E10.1.2, das eine „kann“ – Anforderung ist, wurde nicht weiter berücksichtigt.

Die Gruppendynamik ist stetig gewachsen. Dies lag unter anderem auch daran, dass man auch abseits der alltäglichen Arbeit zusammen seine Freizeitaktivitäten gestaltete. So konnte der sogenannte Event-Manager für einen Ausgleich zu der alltäglichen Arbeit schaffen. Dies stärkte das soziale Gruppengefüge. Die Gruppenteilnehmer konnten jedoch auch ihre fachlichen Stärken weiter ausbauen und ihre Schwächen untereinander ausgleichen. Hierbei half es, dass die Konzepte zusammen in Teilgruppen entwickelt wurden und die Zusammenstellung der Gruppen nahezu wöchentlich variierte. Zur durchgeführten „Paarprogrammierung“ wurden Konzepte festgehalten und fortlaufend, falls es erforderlich war, verbessert und überarbeitet.

Die Kenntnisse in der Programmierung konnten durch die Durchführung des Projektes erweitert werden. Alle Teilnehmer konnten voneinander profitieren und sind um viele Erfahrungen reicher geworden.

10.6 Ausblick

Durch die gesammelten Erfahrungen, wie durch aufgetretene Probleme, sind in unterschiedlichen Bereichen durchaus sinnvolle Verbesserungen von Nöten.

Da die Anfragen an den NoGoSolvers im JSON-Format lange Antwortzeiten erfordern, wäre es von Vorteil, wenn der NoGoSolver um eine Funktion ergänzt würde, die es ermöglicht, eine komplette Route anzugeben und daraufhin eine Antwort zu erhalten, die die Gebiete um die Route herum enthält. Auf Client-Seite könnte der NoGoSolver auch asynchron angefragt werden, also nicht erst wenn die Antwort benötigt wird, sondern antizipativ. Des Weiteren ist eine Performanceverbesserung zu erwarten, wenn im MultiThreading bestimmte Operationen berechnet werden. Das Auslagern des A-Stern-Algorithmus wäre ein weiterführendes Ziel. Dies sollte im Dynamischen, wie auch im statischen Bereich passieren.

Um eine weitere Performanceerhöhung zu gewährleisten, ist eine Auslagerung von schweren Rechenoperationen auf die Grafikkarte ein weiterer möglicher Schritt. Zusätzlich kann das Verarbeiten von unterschiedlichen Heuristiken von Vorteil sein.

Da der ChartServer, aufgrund von rechtlichen Schwierigkeiten, nicht öffentlich zugänglich gemacht werden und somit nicht einfach Seekarten von dort abgefragt werden können, stellt dies eine weitere Verbesserungsmöglichkeit dar. Es sollte nur für die folgenden Projektgruppen eine Möglichkeit geschaffen werden, aus dem gesamten Offis-Netzwerk Seekarten abzufragen, um deren Aktualität und die Testbarkeit zu verbessern.

Die Verarbeitung von Seekarten ist zum aktuellen Zeitpunkt sehr zeitintensiv und fehleranfällig. Einerseits liefert der ChartServer fehlerhafte GML-Daten, andererseits ist die Komponente (FileFeatureProvider) zum Parsen von GML-Daten in Java-Objekte sehr langsam und macht es so beinahe unmöglich, reale Seekarten in die Simulation einzubinden. Ebenso ist das Testen von neuen Entwicklungen unter Beachtung von realen Seekarten sehr zeitintensiv. Die aktuelle Lösung ist die Verwendung des NoGoSolvers, die aber vorher genannte Probleme mit sich bringt.

Da der FileFeatureProvider Probleme mit den Systemrechten beim Zugriff auf die Datei, sowie ein nicht optimiertes Verhalten zum aktuellen Zeitpunkt darstellt, ist hier ein weiterer Handlungsbedarf erforderlich. Außerdem sollte der Chart-Server in seine Anbindung verbessert werden, da dieser fehlerhafte gml-Dateien erstellt, sowie eine Anbindung nur über das interne Netzwerk erfolgreich durchgeführt werden kann.

Da die Konfiguration in HAGGIS immer komplexer wird, besteht auch hier die Möglichkeit, in der grafischen Oberfläche mit Wizards zu arbeiten, um den Anwender auf Pflichteinstellung hinzuweisen.

11 Literaturverzeichnis

Atlassian Coorporation, Great Britian, London: https://de.atlassian.com/software/confluence 04.04.2018.

Blaich, M., (2016). Carl von Ossietzky Universität Oldenburg, Fakultät II, Informatik, Wirtschafts- und Rechtswissenschaften - Department für Informatik: Path Planning and Collision Avoidance for Safe Autonomous Vessel Navigation in Dynamic Environments.

Carl von Ossietzky Universität Oldenburg, Germany, Oldenburg: https://www.uni-oldenburg.de/itdienste/services/datenhaltung/cloudstorage/ 04.04.2018.

Dibbern, C., Hahn, A., & Schweigert, S. (2014). Interoperability In Co-Simulatons Of Maritime Systems. In ECMS (pp. 71–77).

Elektronischer Wasserstraßen-Informationsservice (ELWIS), Wasserstraßen- und Schifffahrtsverwaltung des Bundes (1998). Seeschifffahrtsstraßen-Ordnung (SeeSchStrO). (§23).

Elektronischer Wasserstraßen-Informationsservice (ELWIS), Internationale Regeln von 1972 zur Verhütung von Zusammenstößen auf See (Kollisionsverhütungsregeln - KVR). 1972.

Greenslade, B., Ward, R. (2011). IHO S-100: The Universal Hydrographic Data Model.

Hahn, A., Noack, T. (2014). eMaritime Integrated Reference Platform. In Proceedings of 2nd International Symposium of Naval Architecture and Maritime, 733–42. Istanbul, Turkey, 2014.

IEEE Standard for Modeling and Simulation (M Amp;S) High Level Architecture (HLA) - Framework and Rules. (2000). IEEE Std. 1516-2000, i–22. doi:10.1109/IEEESTD.2000.92296 IMO. (2012). Report of the 58th session of IMO Sub-Committee on Safety of Navigation, NAV 58/WP.6/Rev.1.

Kuhn, M. (2010). Git - Fast Version Control System, Universität Hamburg, Fachbereich Informatik.

Kramer, O. (2009). Computational Intelligence: Eine Einführung. Springer-Verlag Heidelberg.

MTSS (2016): Maritime Transportation Systems Simulation: Abschlussdokumentation.

OpenSeaMap - Die freie Seekarte: http://www.openseamap.org/index.php?id=openseamap&no_cache=1* 04.04.2018.

Reddy, H. (2013): Path Finding – Diklstra´s and A* Algorithm http://cs.indstate.edu/hgopireddy/algor.pdf 27.04.2017.

Schweigert, S., Gollücke, V., Hahn, A. (2014) HAGGIS: A modelling and simulation platform for e-Maritime technology assessment. In ECMS (pp. 71–77).

SOLAS Kapitel V (2002): http://www.bsh.de/de/Schifffahrt/Sportschifffahrt/Berichtigungsservice_NfS/Schifffahrtsvorschriften/2002//Beilage-NfS33-2002.pdf 25.04.2017.

Universität Rostock, Germany (2001), Rostock, Professur für Geodäsie und Geoinformatik (GG), Quadtree: http://www.geoinformatik.uni-rostock.de/einzel.asp?ID=1410 04.04.2018.

12 Anhang

Allgemeines maritimes Umfeld

In dieser Seminararbeit wird das allgemeine maritime Umfeld betrachtet. Mit Blick auf das Ziel der Projektgruppe „intelligente Schiffs Simulation", soll ein kurzer Überblick über Seekarten, Verkehrsteilnehmer, Schifffahrtszeichen und die Witterungsbedingungen gegeben werden. Dem Leser soll vermittelt werden, worauf bei der Schiffsführung auf See und in Kanälen und Flüssen geachtet werden muss. Die Fragen, „Was kann durch sensible Sensorik automatisiert werden?" und „Was ist davon für die Simulation entscheidend?" sollen zumindest im Ansatz geklärt werden.

Inhaltsverzeichnis

Seekarten

Für die Navigation auf hoher See, Kanälen und Flüssen werden, wie auch für die Orientierung an Land, Karten genutzt. Im Vergleich zu einer klassischen Landkarte stellt die Seekarte zusätzlich hydrographische Daten und Navigationshilfen bereit. Eine Seekarte muss geeignet und aktuell sein um in der Schifffahrt eingesetzt werden zu können. Normen und Anforderungen an Seekarten werden durch die IHO (International Hydrographic Organisation) festgelegt. Der aktuelle Standard für Kartendaten ist der S-57, welcher in einer separaten Ausarbeitung umfassend erläutert wird. Die Notwendigkeit des Einsatzes von Seekarten regelt die IMO (International Maritime Organization). Gemäß der Empfehlung der SOLAS (International Convention for the Safety of Live at Sea) muss die Reise eines Schiffes schon vor Reiseantritt geplant und in der Karte dokumentiert sein. Hierbei müssen alle Besonderheiten der Reise, soweit vorhersehbar, beachtet und berücksichtigt werden. Während der Reise ist es Aufgabe des wachhabenden Offiziers, den planmäßigen Verlauf der Reise zu überwachen und gegebenenfalls anzupassen. (Handbuch Nautik, Seite 24ff.) Werden für die Navigation Papierseekarten verwendet, muss die Reiseplanung sowie die Überwachung der Reise „per Hand" dokumentiert werden. Elektronische Seekarten bieten hingegen den Vorteil, dass sie sich updaten lassen und den Benutzer mit zusätzlichen Informationen versorgen können. Es gibt verschiedene Arten der elektronischen Seekarten.

ECDIS (Electronic Chart Display and Information System): Hierbei handelt es sich um ein System, welches eine Papierseekarte vollständig ersetzten kann. Es erfüllt alle IMO Anforderungen Auf die konkreten Anforderungen der *IMO Performance Standards for ECDIS* wird an dieser Stelle nicht weiter eingegangen. Die Modellierung von Seekarten gemäß dem internationalen Standard S-57 wird in einer weiteren Ausarbeitung detailliert behandelt..

ECS (Electronic Chart System): Das ECS ist ein elektronisches Seekarten System, welches die IMO Anforderungen nicht erfüllt. Es darf nur ergänzend zur Papierseekarte genutzt werden.

ENC (Electronic Navigational Chart): Die ENC ist ein regional begrenzter Auszug digitaler Seekartendaten. (Handbuch Nautik, Seite 140) Diese basieren auf offiziellen hydrografischen Daten, unterliegen dem Standard S-57 und werden ebenfalls nach diesem Standard berichtigt. Der Datensatz enthält alle navigatorisch relevanten Informationen. (Handbuch Nautik, Seite 144)

RNC (Raster Navigational Chart): Eine RNC ist ein offizieller Datensatz für eine Rasterkarte und endspricht einer digitalen Kopie einer Papierseekarte. Sie weist eine geringere Funktionalität als die ENCs auf.

Die IMO hat eine Ausrüstungsvorschrift erlassen, nach der alle größeren Handels- und Passagierschiffe bis spätestens zum 1. Julie 2018 mit einem zugelassenen ECDIS inklusive aktuellem Kartenmaterial ausgerüstet sein müssen. ((SOLAS V, Regel 19) Handbuch Nautik, Seite 139ff.)

Verkehrsteilnehmer

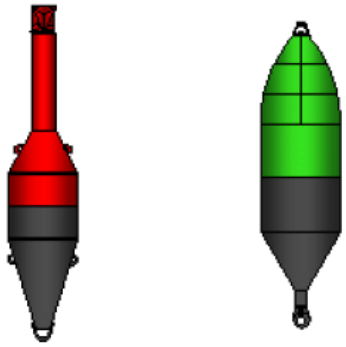
Wesentliche Verkehrsteilnehmer sind andere Schiffe. Diese können große Handelsschiffe (Containerschiffe, Bulkschiffe, Tanker etc.), Passagierschiffe, Behördenschiffe, Marine Schiffe, Schiffe, welche für die Verrichtung bestimmter Arbeiten auf See eingesetzt werden, Arbeitsplattformen sowie private Sportboote sein. Alle Teilnehmer werden durch das Seeverkehrsrecht gesteuert und haben verschiedene Rechte und Pflichten. Von zentraler Bedeutung des Seeverkehrsrechtes sind die Kollisionsverhütungsregeln.

Schifffahrtszeichen

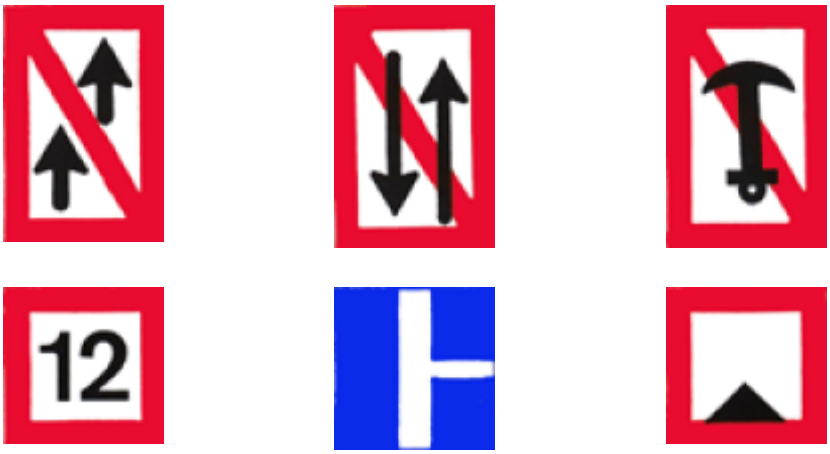
Wie auch im Straßenverkehr gibt es in der Schifffahrt verschiedene Verkehrszeichen und Leitsysteme. Im Rahmen dieser Seminararbeit werden nur einige der Verkehrszeichen und Signale vorgestellt. Für umfassende Informationen zu den Schifffahrtszeichen empfiehlt sich ein Blick auf die Internetseite der Wasserstraßen- und Schifffahrtsverwaltung des Bundes.

Ein allgemein bekanntes Signal in der Schifffahrt ist das Leuchtfeuer. Dieses dient den Schiffen in Küstennähe zur Orientierung und ist i.d.R. ein fest stehendes Signal (Leuchtturm) auf Inseln oder an der Küste des Festlandes. Auch unter den schwimmenden Schifffahrtszeichen gibt es befeuerte Signale, wie z.B. unbemannte Feuerschiffe und Leuchttonnen. Die deutschen Feuerschiffe dienen neben der klassischen Orientierungshilfe durch Lichtsignale, mit Radar und Funkausrüstung. Außerdem liefern sie meteorologische Daten und verfügen über einen Notraum mit Funktelefon für Schiffbrüchige.

Die Gasleuchttonne in gelb/schwarz (je nach Bemusterung mit unterschiedlicher Bedeutung) ist ein kardinales Schifffahrtszeichen und markiert Warnstellen und Hindernisse.
Die grünen Spitztonnen mit ungeraden Zahlen bezeichnen die Steuerbordseite (rechts Stromaufwärts) des Fahrwassers. Die rote Spierentonne mit geraden Zahlen markiert die Backbordseite (links Stromaufwärts) des Fahrwassers. (wsa-whv.wsv.de)



Ergänzend gibt es noch eine Reihe weiterer Schifffahrtszeichen die Gebote, Verbote und Hinweise geben. Im Folgenden werden einige Beispiele gezeigt:



Eine Übersicht aller Schifffahrtszeichen bietet die Internetseite elwis.de.

Auch die Schiffe selbst müssen entsprechend des durchgeführten Manövers oder bei Manövrierunfähigkeit verschiedene Sichtzeichen und Schallsignale geben. So kann z.B. die Absicht an Steuerbord zu überholen, durch zwei lange und ein kurzes akustisches Signal an den Vordermann vermittelt werden. Mit einem langen, einem kurzen Signal zwei Mal hintereinander vermittelt dieser sein Einverständnis. (elwis.de, Handbuch Nautik, Seite 344ff.)
Weitere Informationen hierzu sind in den Kollisionsverhütungsregeln zu finden.

Umwelteinflüsse und ihre Auswirkungen

Schiffe auf hoher See sind besonderen Umwelteinflüssen ausgeliefert. Diese beeinflussen das Verhalten des Schiffes und setzten es besonderen Belastungen aus. Bei zu starken Belastungen oder bei Gefährdung von Ladung und Besatzung, müssen geplante Routen geändert und der aktuellen Situation angepasst werden. Im Folgenden werden die wesentlichen Einflüsse auf die Schifffahrt mit ihren Folgen vorgestellt.

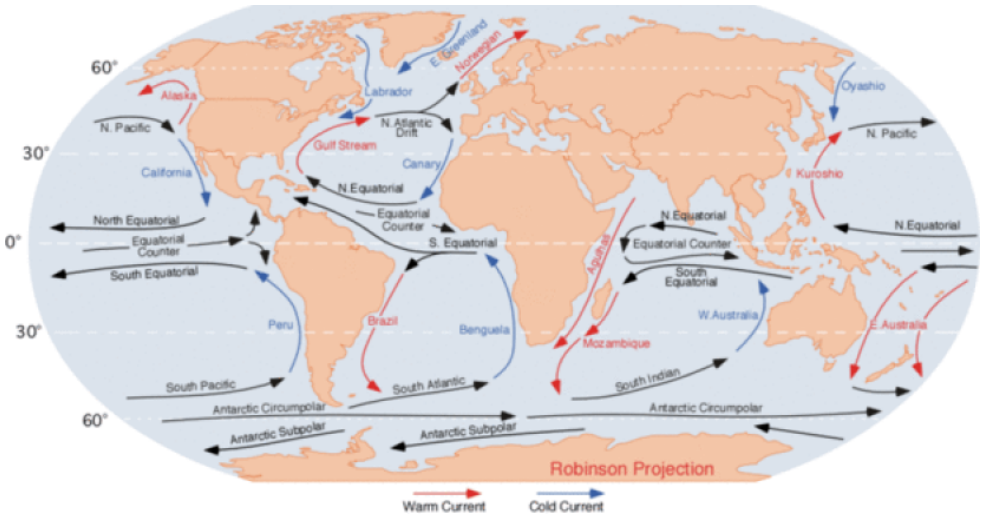
Strömungen

Die Gezeitenströmungen, auch bekannt als Ebbe (Niedrigwasser) und Flut (normaler Wasserstand), werden hervorgerufen durch die Anziehungskraft von Mond und Sonne. Sie beeinflussen den Wasserstand von Ozeanen und Meeren insbesondere in engen Buchten und

Flussmündungen. Beträgt der Tidenhub auf den Ozeanen noch übersichtliche 0.75m, so ist an der deutschen Nordseeküste eine Veränderung des Wasserstandes um 3m zu beobachten. Bei St. Malo am Ärmelkanal kann der Tidenhub sogar 12m betragen. Zusätzlich kann Wind, Windrichtung und der Luftdruck die Gezeiten beeinflussen. (Handbuch Nautik, Seite 297ff.)

Diese Veränderungen des Wasserstandes müssen im Betrieb von Seeschiffen berücksichtigt werden. Schiffe mit großem Tiefgang können gewisse Passagen bei Ebbe nicht mehr befahren. Auch ist beispielsweise der Fährverkehr zu den Inseln auf die Flut angewiesen. Eine vorausschauende Planung ist somit unumgänglich.

Die großen Meeresströmungen wie der Golfstrom folgen einem klaren Kurs und wechseln nicht plötzlich die Richtung. Diese Strömungen (Siehe Abbildung 9) werden im Vergleich zu den Gezeitenströmungen durch Wind verursacht und tragen wesentlich zu dem Wärmetausch auf der Erde bei. Die Transportgeschwindigkeit der Ströme ist mit einer 15 – 30 Seemeilen pro Tag relativ gering. Hinweise zu den Strömungen lassen sich in aktuellen Karten von Hydrographischen Diensten entnehmen. Die Ströme haben insofern Einfluss auf die Schifffahrt, als dass sie die Geschwindigkeit des Schiffes verringern oder beschleunigen können. Zudem haben Sie Einfluss auf die Wellenbildung. Treffen starke Winde auf eine entgegengesetzte Strömung, bilden sich steile Wellen, was zu extremer Wellenhöhe führen kann (Kaventsmann). (Handbuch Nautik, Seite 294ff.)



Luftdruck

Selbst Luft besitzt eine Masse welche Gewicht ausübt. Diese Luftmassen besitzen verschieden Temperaturen, was dazu führt, dass die Dichte schwankt. Der Luftdruck kann an Bord eines Schiffes mit Hilfe eines Barometers gemessen werden, die Maßeinheit ist Hektopascal. Er kann den Wasserstand geringfügig beeinflussen. Wesentlicher für die Schifffahrt ist jedoch der aus der Veränderung des Luftdruckes resultierende Wind. Wie aus dem Wetterbericht bekannt, gibt es Hoch- und Tiefdruckgebiete. Treffen die verschiedenen Druckgebiete aufeinander, gleichen sie sich aus und entsteht Wind. (Handbuch Nautik, Seite 247) Allein die Beobachtung des Barometers an Bord reicht nicht aus, um die Windentwicklung treffsicher vorauszusagen. Fährt das Schiff Beispielsweise vor dem Tief mit Zugrichtung des Tiefs, so fällt der Luftdruck langsam und ein starkes zunehmen des Windes wird demnach nicht vermutet. Tatsächlich kommt es in einer solchen Situation aber zu einer starken Windentwicklung. Vor Antritt der Reise ist also neben der reinen Routenplanung auch eine meteorologische Reiseplanung sinnvoll. (Wetter auf See, Seite 18ff.)

Wind

Überall dort, wo horizontale Druckunterschiede auftreten, sind horizontale Luftbewegung die Folge. Dabei geht der Ausgleich immer vom höheren zum tieferen Druck. In Wetterkarten findet man sogenannte Isobaren. Isobaren sind Linien gleichen Luftdrucks. Je enger die Isobaren auf den Karten beieinanderliegen, desto stärker weht der Wind. Auch die Windstärke kann gemessen werden. Die Windstärke wird anhand der Beaufortskala festgehalten. (Handbuch Nautik, Seite 247)

Winde betreffen die Schifffahrt unmittelbar. Je nach Windstärke, Typ des Schiffes, Position der Aufbauten und Geschwindigkeit des Schiffes, wirkt sich der Wind unterschiedlich auf das Verhalten des Schiffes aus. Um den Kurs zu halten und nicht von der geplanten Route abzuweichen, bedarf es genauer Kenntnisse des Schiffs und der Ruderanlage. Je langsamer ein Schiff unterwegs ist, desto stärker wirkt sich der Wind auf die Steuerfähigkeit aus. Folgende Gegenmaßnahmen sind geeignet um die Steuerfähigkeit zu erhalten.

- Erhöhung der Geschwindigkeit
- Vergrößerung des Tiefgangs, um die Angriffsfläche des Windes zu verkleinern
- Erhöhung der Propellerdrehzahl ohne Steigerung der Geschwindigkeit, um eine kurzfristige Verbesserung der Steuerfähigkeit zu erreichen.

Bei ungünstigen Winden kann es vorkommen, dass der wachhabende Offizier die Mindestgeschwindigkeit errechnen muss, um das Schiff Steuerfähig zu halten. (Handbuch Nautik, Seite 323ff.)

Seegang

Als Folge des Windes entstehen Wellen. Die unmittelbar durch den Wind angefachten Wellen nennt man Windsee. Die Windsee ist für den Beobachter leicht zu erkennen, die Wellen kommen unregelmäßig und haben spitze Wellenkämme. Ihnen voran schreitet die sogenannte Dünung. Die Dünung ist auslaufender Seegang und zeichnet sich durch abgerundete Wellenberge und eine verhältnismäßig lange Wellenlänge aus. Die Dünung ist nicht sofort zu erkennen. Sie besitzt eine höhere Energie als die Windsee und bewegt sich schneller. Treffen zwei, in verschiedene Richtungen laufende Wellenbilder aufeinander, entsteht eine Kreuzsee. Kreuzseen sind gefährlich, da sich in ihr vereinzelte große Wellen bilden können, die nicht vorhersehbar sind. (Handbuch Nautik, Seite 288ff.)

Der Seegang versetzt das Schiff in Schwingungen. Es wird unterschieden in Drehschwingungen (Rollen, Stampfen und Gieren) und in translatorische Schwingungen (Schnellen, Tauchen und Versetzen). Besondere Bedeutung wird den Drehschwingungen Rollen und Stampfen zugesprochen, da diese die Schiffsführung stark beeinträchtigen und zur Gefährdung von Schiff, Ladung und Besatzung führen können. (Handbuch Nautik II, Seite 393) Als Stampfen wird die Bewegung des Schiffes um seine Querachse verstanden. Dabei wird das Schiff vorne abgesenkt und hinten angehoben oder umgekehrt. Dies kann bei starkem Seegang für erheblichen Höhenunterschied zwischen Buck und Heck des Schiffes sorgen. besonders bei Ladung, die an Deck transportiert wird, ist hier schwerer Seegang nicht ohne Risiko. Rollen wird die Bewegung des Schiffes um die eigene Längsachse genannt. Dabei bewegt sich das Schiff seitlich hin und her. Die Neigung des Schiffes wird als Rollwinkel bezeichnet. Bei verhältnismäßig ruhiger See sind Rollwinkel von 10º nicht ungewöhnlich, bei schwerer See kann der Rollwinkel bis zu 30º betragen. Damit wird deutlich, dass diese Bewegungen starken Einfluss auf Schiff und Ladung haben. (Containerhandbuch.de, Kap. 2.3.3) Weitere gefahren, die nur schwer durch automatisierte Vorgänge eingeschätzt werden können, ist beispielsweise ein stabilitätsverlust auf Wellenbergen. wird ein Schiff auf einen Wellenberg gehoben und fährt mit annähernd gleicher Geschwindigkeit wie sich die Welle fortbewegt, so kann dem Schiff ein erheblicher stabilitätsverlost drohen. In solchen Fällen wird empfohlen, die Geschwindigkeit zu drosseln. (Handbuch Nautik II, Seite 400) Als letzter Punkt soll das Slamming genannt werden. Hierunter versteht man hydrodynamische Stöße gegen das Schiff, die durch hartes Einsetzen des Schiffes in die See entstehen, beispielsweise beim einfahren in einen Wellenberg. Dies kann zu Schäden am Schiff führen und darüber hinaus die ganze Schiffsstruktur in Schwingungen versetzen. (Handbuch Nautik II, Seite 403) Der Einfluss des Seeganges auf ein Schiff sollte nicht unterschätzt werden, da er unter Umständen zu erheblicher Gefährdung von Schiff, Ladung und Besatzung führen kann. Eine erfahrene Bestatzung kann Gefahren frühzeitig erkennen und entsprechende Maßnahmen ergreifen. Dies kann dazu führen, dass eine ursprünglich geplante Route nicht eingehalten werden kann und individuell angepasst werden muss.

Meereis

Es treten neben den bereits genannten meteorologischen Einflüssen noch weitere Umwelteinflüsse auf, die die Navigation eines Schiffes individuelle beeinflussen können. So zum Beispiel Meereis.
Meereis lässt sich zwar voraussagen und planen, schränkt die Schifffahrt aber unter Umständen stark ein. Teilgebiete können durch Eisbildung für die Schifffahrt unbefahrbar werden. Einige Strecken werden jedoch permanent durch Eisbrecher freigehalten. Für die Reiseplanung besteht die Möglichkeit Eiskarten anzufordern. (Handbuch Nautik, Seite 299ff.) Bei Fahrten durch extrem kalte Gebiete, kann es zu Schiffsvereisungen aufgrund von Spritzwasser kommen. Bei starker Vereisung ist zu bedenken, dass sich Schwerpunkt und Stabilität des Schiffes ändern. (Handbuch Nautik, Seite 293ff.)

Fazit

Nach einem kurzen Überblick auf die Seekarten, mit Schwerpunkt auf die elektronischen, da diese insbesondere für die Simulation relevant sind, auf die Verkehrsteilnehmer und die Schifffahrtszeichen wurde auch der Einfluss von Wetter und Strömungen vermittelt. In weiteren Seminarthemen werden die elektronischen Seekarten und die Kollisionsverhütungsregeln noch detaillierter betrachtet. Dennoch lässt sich an dieser Stelle bereits absehen, dass die Seekarten eine Menge an Informationen liefern können und diese digital vernetzt sind. Auch die Kommunikation mit anderen Verkehrsteilnehmern sollte mit der heutigen Technik kein Problem sein und somit die Einhaltung des Seeverkehrsrechtes durch entsprechende Algorithmen möglich sein. Strömungen und Witterung stellen hier schon eine größere Herausforderung. Diese lassen sich nicht immer in ausreichendem Umfang vorhersehen und erfordern erfahrene Reaktionen der nautischen Offiziere. Doch auch hierfür gibt es meteorologische Karten, die eine Reiseplanung vereinfachen. Letztlich können die Eingangs aufgestellten Fragestellungen in Teilen beantwortet werden.

Was kann durch sensible Sensorik automatisiert werden? – Zumindest die Kommunikation zwischen den Verkehrsteilnehmern, das gesetzmäßige Verhalten der Verkehrsteilnehmer und die Reaktion der verschiedenen Parteien. Auch die Wettereinflüsse und entsprechende Reaktionen können in Teilen automatisiert werden.

Was ist davon für die Simulation entscheidend? – Wichtig für die Simulation ist das ordnungsgemäße Verhalten der Verkehrsteilnehmer untereinander. Außerdem muss das Seeverkehrsrecht eingehalten werden. Da die Wettereinflüsse in der Simulation direkt gesteuert werden können, ist für die reine Simulation zwar das Verhalten des Schiffes wichtig, jedoch nicht zwingend der „Überlebensinstinkt" anderer Verkehrsteilnehmer.

Literaturverzeichnis

Berking, B., Huth, W., 2010, Handbuch Nautik, DVV Media Group GmbH | Seehafen Verlag, Hamburg
Benedict, K., Wand, C., 2011, Handbuch Nautik II, DVV Media Group GmbH | Seehafen Verlag, Hamburg
Brauner, R., Herrmann, B., Nafzger, HJ. Wetter auf See, DSV Verlag GmbH, Bielefeld

Containerhandbuch.de, Kapitel 2.3.3, Abgerufen am 27. April 2017 https://www.containerhandbuch.de/chb/stra/index.html
Elektronischer Wasserstraßen-Informationsservice (ELWIS), Abgerufen am 27. April 2017 https://elwis.de/Schifffahrtsrecht/Binnenschifffahrtsrecht/BinSchStrO/Anlagen/Anlage-7/index.html
Wasserstraßen- und Schifffahrtsamt Wilhelmshaven, Abgerufen am 27. April 2017 http://www.wsa-whv.wsv.de/wasserstrassen/schifffahrtszeichen/index.html
Wiki Bildungsserver, Abgerufen am 27. April 2017 http://wiki.bildungsserver.de/klimawandel/index.php/Meeresströmungen

Präsentation

Anomaliedetektion

Im Rahmen der Projektgruppe „Intelligente Schiffssimulation“ findet eine Seminarphase über die relevanten Themen statt. Diese Arbeit beschreibt die Anomaliedetektion und zugehörige intelligente Verfahren.

Inhaltsverzeichnis

Einleitung

Anomaliedetektion bezeichnet das Auffinden von Elementen oder Ereignissen, die nicht mit den erwarteten Mustern in einer Datenmenge übereinstimmen. Dabei gibt es keine mathematische Definition was eine Anomalie ist, sondern muss subjektiv für die Anwendung definiert werden. Im Kontext der maritimen Schifffahrt sind Anomalien, die Auftreten und von Interesse sind beispielsweise:

- ein Schiff fährt zu schnell oder zu langsam für den Typ des Schiffs in der Region
- ein Schiff biegt auf einer Route nicht an dem Punkt ab, an dem andere Schiffe für gewöhnlich abbiegen
- ein Schiff fährt in einer Region , in der es für Schiffe dieser Größe ungewöhnlich oder verboten ist
- ein Schiff manövriert ungewöhnlich
- ein Schiff stoppt unerwartet

Um Anomalien zu erkennen, wird zunächst ein Modell auf den Daten gebildet, auf denen dann nach Ausreißern gesucht wird. Dabei gibt es grundsätzlich zwei verschiedene Arten der Anomaliedetektion, überwachte und unüberwachte Anomaliedetektion. Bei der überwachten Anomaliedetektion wird auf einer Trainingsmenge an Daten, die mit den Kategorien normal und anormal beschriftet sind, ein Klassifikator gebildet. Neue Daten werden dann mittels des Klassifikators ihrer Kategorie zugeordnet. Ein Beispiel für diese Art der Anomaliedetektion ist der k-nächste-Nachbarn Algorithmus (KNN). Bei der unüberwachten Anomaliedetektion hingegen wird angenommen, dass der Großteil der Daten normal sind. Daher wird nach den Daten gesucht, die am wenigsten zu den übrigen Daten passen. Dies geschieht in dem man auf den Daten eine Clusteranalyse, welche die Daten nach ihrer Ähnlichkeit gruppiert, durchführt. Ein Beispiel für diese Art Anomaliedetektion ist DBSCAN [1].

Sowohl KNN und DBSCAN werden im Folgenden als allgemeine Verfahren zur Anomaliedetektion und TREAD (Traffic Route Extraction and Anomaly Detection) als spezielles Verfahren zur Anomaliedetektion im Kontext der maritimen Schifffahrt vorgestellt.

k-nächste-Nachbarn

KNN ist ein dichtebasierter Algorithmus zur Klassifikation, der überwachtes Lernen benutzt (allerdings nur in einer sehr einfachen Form). Eine Klassifikation weist anhand einer Trainingsmenge von Daten, von denen bekannt ist zu welcher Kategorie sie gehören, einem neuen Datum seine Kategorie zu. Auf den Daten muss eine Distanz definiert sein, welche man benutzt um die Entfernung der Daten untereinander und damit ihre Ähnlichkeit zu beurteilen. Bei kontinuierlichen Daten wird meist die euklidische Distanz genommen, die gerade bei zwei- und dreidimensionalen Räumen mit dem intuitiven Abstand übereinstimmt. Man kann allerdings auch eine andere Distanz benutzen, wie zum Beispiel die Manhatten-Distanz oder bei diskreten Werten die Hamming-Distanz. Das *k* ist eine vom Benutzer gewählte Konstante. Dem Algorithmus wird eine Trainingsdatenmenge von Vektoren von denen ihre Kategorie bereits bekannt ist, zur Verfügung gestellt.

Die Klassifikation eines neuen Datums geschieht dann wie folgt: Es werden die *k* Datensätze aus der Trainingsmenge ermittelt, die den geringsten Abstand zu dem neuen Datum haben Das neue Datum bekommt dann die Kategorie, welche am Häufigsten in den *k* Datensätzen vorkamen.

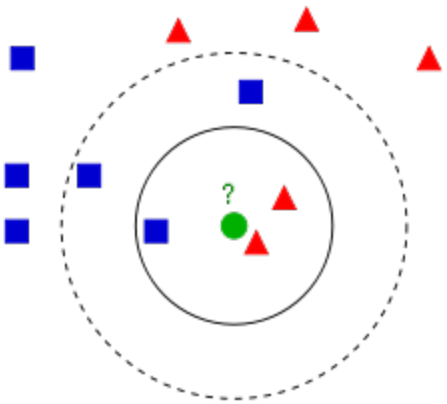


Abbildung 1: k-nächste-Nachbarn (Vgl. Ajanki 2007)

In der Abbildung 1 soll dem grünen Punkt entweder die Kategorie blau oder rot zugeordnet werden. Bei k = 3 wird er rot zugeordnet, da nur zwei rote Dreiecke, aber nur ein blaues Quadrat zu seinen Nachbarn gehören. Bei k = 5 wird er hingegen blau zugeordnet (da 3 blau Quadrate und nur zwei rote Dreiecke zu seinen Nachbarn zählen).

Bei *k = 1* wird dem neuen Datum einfach die Kategorie seines nächsten Nachbarn zugeordnet. Ein Problem des einfachen Verfahrens der Mehrheitsentscheidung ist, dass Kategorien, die in der Trainingsmenge häufiger auftauchen, andere Kategorien dominieren. Um dem zu entgehen, kann man auch eine gewichtete Variante von KNN benutzen, bei der auch noch die Entfernung des neuen Punkts zu seinen k-nächsten-Nachbarn, bei der Klassifikation berücksichtigt wird. Dazu wird die Kategorie eines jeden Punktes der k-nächsten-Nachbarn mit der Inversen der Distanz zu dem neuen Datum multipliziert und von diesen neuen Werten dann die Kategorie, welche am Häufigsten auftauchte, dem neuen Datum als Kategorie zugeordnet.

Das Problem bei KNN ist die sowohl die Wahl eines geeigneten *k*, als auch die Wahl einer geeigneten Trainingsmenge. Ein zu kleines *k* macht das Verfahren anfällig für Rauschen in den Daten, ein zu großes *k* verwischt die Grenzen zwischen den Klassen. Die Trainingsmenge muss die Menge, die ihr zugrunde liegt, gut wiederspiegeln. Zu viele Ausreißer oder die Überrepräsentation von Kategorien können die Güte des Verfahrens sehr verschlechtern.

KNN kann auf zwei Arten zur Anomaliedetektion verwendet werden. Zum einen kann eine Klassifikation nach den Kategorien normalen und anormalen Daten erfolgen. Dazu muss nur eine Trainingsmenge von normalen und anormalen (Anomalien) Daten erstellt werden. Zum anderen kann die Entfernung zu den k-nächsten-Nachbarn auch als lokale Dichteschätzung interpretiert werden. Ist die geringer als ein gewisser Schwellenwert ist das neue Datum wahrscheinlich eine Anomalie [2]. Dies ist auch einer der zugrunde liegenden Ideen von DBSCAN, der als nächstes vorgestellt wird.

DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise - *Dichtebasierte räumliche Clusteranalyse mit Rauschen)* ist ein dichtebasierender Algorithmus zur Clusteranalyse. Der Algorithmus benötigt zwei Eingaben und minPts. Ein Punkt ist von einem anderen Punkt erreichbar, wenn der Abstand der beiden Punkte kleiner als ist. Dabei wird als Abstand wie bei KNN für gewöhnlich die euklidische Distanz genommen. Ein Punkt ist dicht, wenn er minPts erreichbare Nachbarn hat. Es werden drei Arten von Punkten unterschieden:

- Kernpunkte sind Punkte, welche dicht sind.
- Dichte-erreichbare Punkte sind Punkte, welche selbst nicht dicht sind, aber von einem Kernpunkt aus erreichbar sind.

Rauschpunkte sind die übrigen Punkte, die also weder dicht, noch dichte-erreichbar sind.

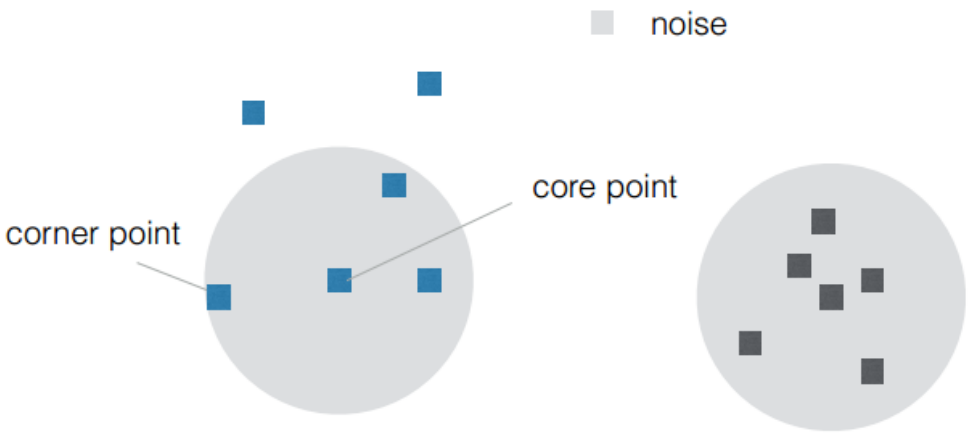


Abbildung 2: DBSCAN (Vgl. Kramer 2009)

Der Algorithmus arbeitet dann wie folgt: Solange es noch einen nicht besuchten Punkt gibt, zähle seine erreichbaren Nachbarn. Handelt es sich um einen Dichtepunkt ist das Zentrum eines neuen Clusters gefunden, ansonsten wird er vorläufig als Rauschpunkt markiert. Wenn ein neuer Cluster gefunden wurde, werden alle Punkte, die vom Dichtepunkt aus erreichbar sind, dem Cluster zugeordnet. Die Punkte, die dem Cluster zugehören, aber bereits besucht wurden, können keine Dichtepunkte sein, sondern sind nur dichte-erreichbare Punkte und werden entsprechend markiert. Die Punkte, die dem Cluster angehören und noch nicht besucht wurden, werden als nächstes besucht. Wenn sie ebenfalls Dichtepunkte sind, werden mit ihren erreichbaren Punkten auf gleiche Art und Weise fortgefahren, wenn sie nur dichte-erreichbar sind, werden sie nur entsprechend markiert und dem Cluster zugeordnet. Wenn es keine weiteren nicht besuchten Punkte, die vom einen Dichtepunkt des Clusters aus erreichbar sind, mehr gibt, wurde das komplette Cluster gefunden und es wird mit dem nächsten nicht besuchten Punkt auf normale Art fortgefahren.
Auf diese Weise wird jeder Punkt nur genau einmal besucht und der Algorithmus ist von seiner Komplexität linear, wenn die Berechnung der erreichbaren Punkte in konstanter Zeit erfolgt, ansonsten quadratisch. Ein dichte-erreichbarer Punkt kann von mehreren Dichtepunkten aus erreichbar sein. Der Algorithmus ordnet ihm nicht-deterministisch einem Cluster zu. Der Rest geschieht deterministisch.

Ein Vorteil gegenüber anderen Clustering-Algorithmen, wie zum Beispiel k-Means besteht darin, dass man nicht vorher die Anzahl der Cluster angeben muss. Das Problem besteht wieder darin, die passenden Werte für die Parameter und minPts zu wählen.

Inkrementeller DBSCAN

Wenn sich Punkte neu hinzugefügt oder gelöscht werden ist es nicht nötig DBSCAN auf den kompletten Datensatz neu durchzuführen. Stattdessen ist es ausreichend nur auf einen Teil der Daten, der durch die Änderungen betroffen ist, DBSCAN erneut durchzuführen. Diese Eigenschaft nutzt IncrementalDBSCAN aus , sodass trotzdem das gleiche Ergebnis herauskommt, wie beim normalen DBSCAN. Für Punkte die eine Entfernung von größer als 2* zu einem Punkt, der neu hinzugefügt oder gelöscht wurde, haben, ändern sich die Art des Punktes nicht. Für Punkte in einer Entfernung von kann sich die Dichteeigenschaft ändern, das heißt aus nicht dichten Punkten können dichte werden und umgekehrt. Für Punkte in einer Entfernung von größer als e und kleiner als 2* ändert sich die Dichteeigenschaft nicht, aber aus dichte-erreichbaren Punkten können Rauschpunkte werden und umgekehrt.

Um zu berechnen, welche Auswirkung das Einfügen eines neuen Punktes p hat, wird zunächst die Menge S = {q|q ist ein Kernpunkt in D {p} und q‘: q‘ ist ein Kernpunkt in D {p}, aber nicht in D und die Distanz von q und q‘ ist kleiner als e} berechnet. In der Menge befinden sich alle jene Kernpunkte, die in der Umgebung eines Punktes liegen, der nur durch einfügen von p zu einem Kernpunkt wurden. Es werden dann folgende Fälle unterschieden:

1. Rauschen (S =): Durch das Hinzufügen von p entstehen keine neuen Kernpunkte, dann ist p entweder ein Rauschpunkt oder ein dichte-erreichbarer Punkt. Der Rest bleibt unverändert.
2. Neues Cluster: In S befinden sich nur Punkte, die vorher keinem Cluster angehörten. Diese bilden zusammen mit p und den dichte-erreichbaren Punkten in ihrer Umgebung ein neues Cluster.
3. Absorption: In S beinhaltet Kernpunkte, die vorher in genau einem Cluster waren. Dann wird p, sowie eventuell vorherige Rauschpunkte diesem Cluster zugeordnet.
4. Vereinigung: In befinden sich Kernpunkte, die vorher verschiedenen Clustern zugehörten. Diese Cluster, sowie p und eventuelle vorherige Rauschpunkte werden zu einem Cluster vereinigt.

Beim Löschen eines Punktes p findet eine symmetrische Konstruktion statt. Dazu wird zunächst die Menge S = {q|q ist ein Kernpunkt in D\{p} und q‘: q‘ ist ein Kernpunkt in D, aber nicht in D \{p} und die Distanz von q und q‘ ist kleiner als e} berechnet. Die Menge beinhaltet alle Kernpunkte, die in der Umgebung eines Punktes liegen, welcher durch das Löschen von p nicht mehr dicht ist. Danach werden folgende Fälle unterschieden:

1. Löschen (S =): p wird gelöscht und eventuell werden Punkte in der Umgebung von p aus ihrem Cluster entfernt. Wenn das passiert, wird das komplette Cluster aufgelöst.
2. Reduktion: Alle Punkte in S sind alle jeweils voneinander erreichbar. Dann wird p gelöscht und eventuell werden manche Punkte in der Umgebung von p zu Rauschpunkten.
3. Möglicher Split: Die Punkte in S sind nicht alle voneinander aus erreichbar. Die Punkte gehörten zu einem Cluster vor der Löschung von p. Nun muss überprüft werden, ob diese Punkte von anderen Punkten aus dem ehemaligen Cluster aus erreichbar sind. Je nachdem muss entweder das Cluster aufgeteilt werden oder nicht. [3]

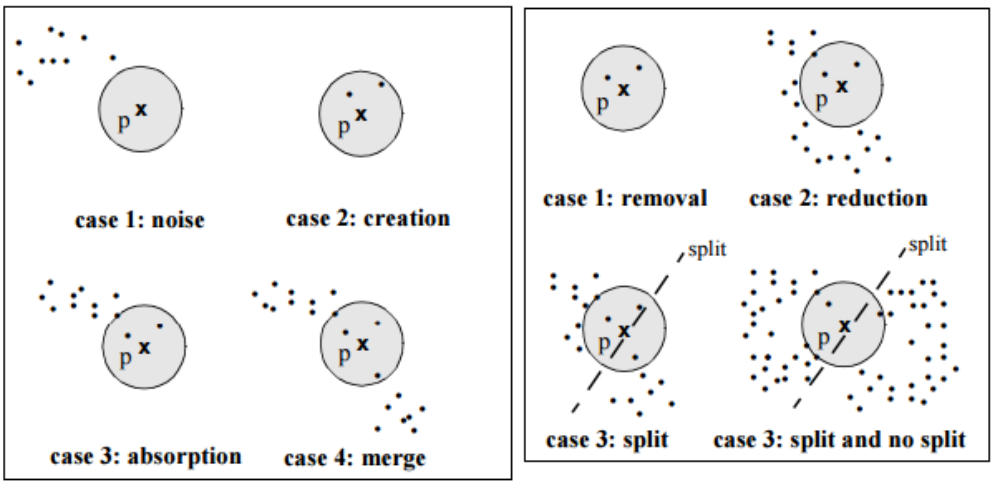


Abbildung 3: Inkrementeller DBSCAN (Vgl. Ester et al. 1998)

TREAD

TREAD (Traffic Route Extraction and Anomaly Detection) ist ein Framework, welches mittels unüberwachtem Lernen von den AIS-Daten von Schiffen Bewegungsmuster in einer Region feststellt und diese zur Vorhersage von Routen von Schiffen, sowie der Erkennung von Anomalien benutzt. Dazu verwendet TREAD unüberwachtes, inkrementelles Lernen um auf veränderte Gegebenheiten zu reagieren (Zum Beispiel veränderte Jahreszeiten oder sich ändernde Routenschema). Dazu benötigt es bis auf die AIS-Daten keine weiteren Informationen. Die AIS-Daten werden auf verschiedenen Ebenen ausgewertet von dem Erkennen von Häfen und Offshore-Plattformen bis hin zur Erkennung von räumlichen und zeitlichen Verteilung der Routen von Schiffen. Dies kann zur regel-basierten Erkennung von Anomalien und zur Erkennung von Anomalien mit geringer Wahrscheinlichkeit, das heißt das Erkennen von Ereignissen, die von der Norm abweichen. Das regel-basierte Erkennen von Anomalien bezeichnet das Erkennen von Anomalien mittels einer Menge von Regeln, wie zum Beispiel die maximale Geschwindigkeit in Häfen oder das Fahren in Gewässern, wo es nicht erlaubt ist. Anomalien mit geringer Wahrscheinlichkeit müssen keine echten Anomalien im operativen Kontext sein, sondern nur Abweichung von der Norm, zum Beispiel ein Schiff biegt auf seiner Route nicht an dem Punkt ab, in dem es andere Schiffe auf dieser Route tun.

Das Wissen über den Schiffsverkehr wird durch die Schiffe und ihren AIS-Daten gebildet. Es wird ein Bereich ausgewählt, der überwacht werden soll. Durch das Clustern von Ereignissen werden Wegpunkte bestimmte, die entweder für stationäre Objekte (zum Beispiel eine Offshore-Plattform), Eingangs- und Ausgangspunkte stehen. Eingangs- und Ausgangspunkte sind die Stellen, in denen Schiffe den überwachten Bereich betreten oder verlassen. Durch das Verknüpfen dieser Punkte entstehen Routenobjekte. Anomalien werden dann anhand dieser historischen Daten und den aktuellen echtzeit Schiffsverkehr bestimmt.

Die Architektur von TREAD funktioniert wie folgt: Der AIS-Datenstrom wird verarbeitet, indem der Schiffs-Objekt-Manager durch inkrementelles Lernen Routenmuster berechnet. Der Schiffs-Objekt-Manager wird durch relevante Ereignisse basierend auf dem zeitlichen und räumlichen Verhalten von Schiffen ausgelöst. Durch das Clustern von solchen Ereignissen werden dann Wegepunkte entdeckt. Diesem Prozess der Wissensentdeckung folgt der Prozess des Wissensnutzung in dem aufgrund des erlernten Wissens über die Routenmuster zum Beispiel eine Anomaliedetektion durchgeführt wird. Abbildung 4 verdeutlicht den Ablauf von TREAD.

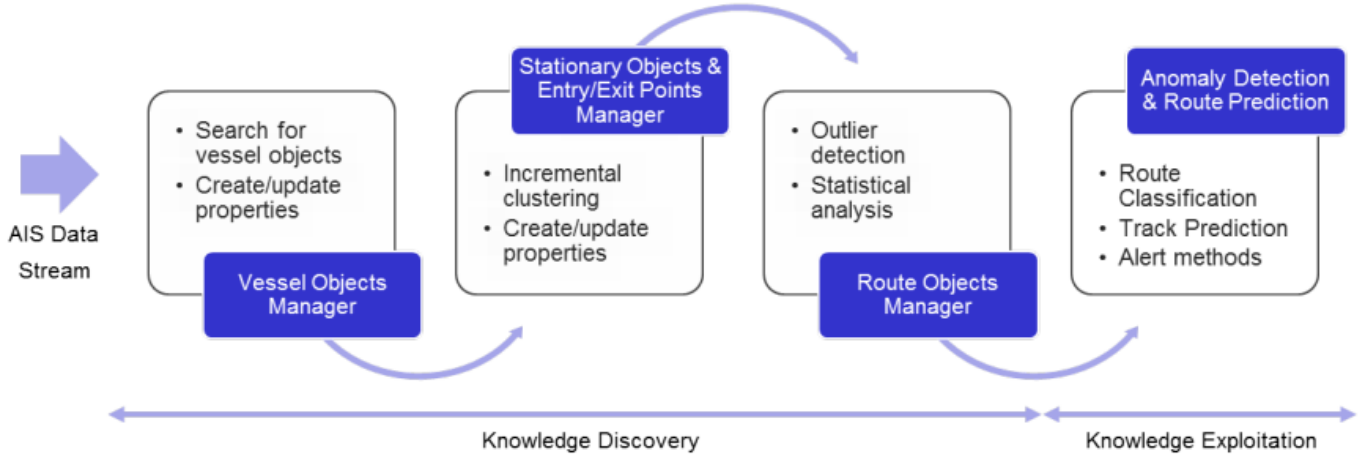


Abbildung 4: TREAD (Vgl. Pallotta et al. 2013)

Immer wenn ein neues Schiff die überwachte Region betritt, wird der Schiffs-Objekt-Manager initialisiert. Die Liste der Schiffe wird mit den Informationen aus den AIS-Daten geupdatet. Durch den Datenstrom der AIS-Daten werden die Status der Schiffe aktualisiert, zum Beispiel wenn ein Schiff stoppt oder wieder zu fahren beginnen. Solche Ereignisse erzeugen Wegpunkte für stationäre Objekte. Diese werden dann vom

Wegepunkte-Objekt-Manager mittels inkrementellen DBSCAN geclustert, um zum Beispiel Häfen zu erkennen.

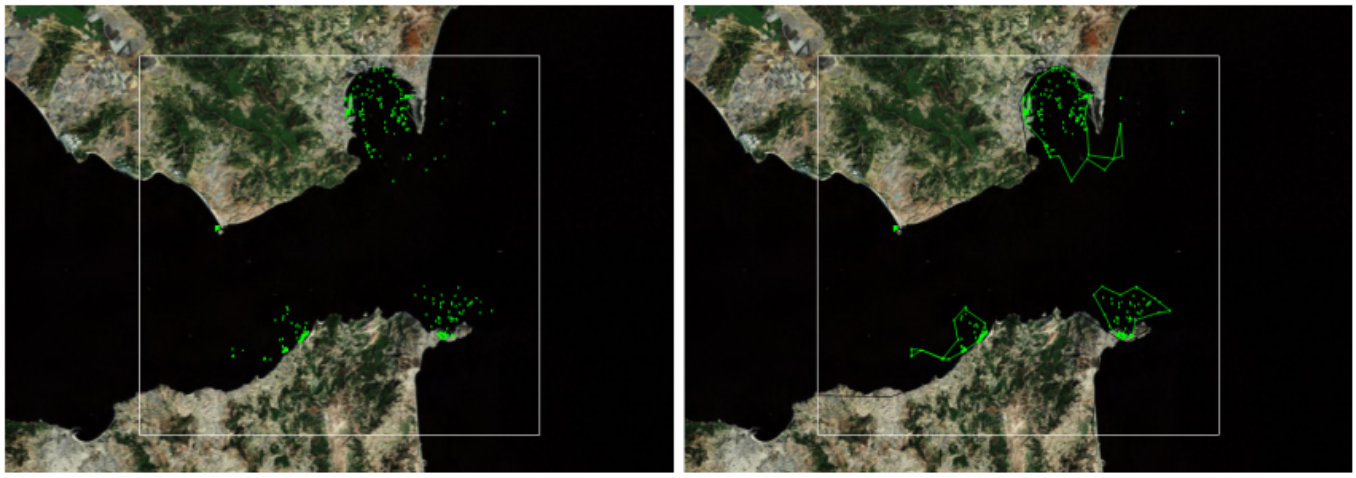


Abbildung 5: Stationäre Punkte in der Straße von Gibraltar (Vgl. Pallotta et al. 2013)

Abbildung 5 zeigt die stationären Punkte (grüne Punkte) die innerhalb von zwei Wochen in der Straße von Gibraltar entdeckt wurden. Rechts sieht man die Clusterung dieser Punkte mittels DBSCAN.

Dasselbe geschieht auch mit den Eingangs- und Ausgangspunkten. DA DBSCAN mit Rauschen umgehen kann, können so auch Ausreißer in den Daten erkannt werden, sodass das Modell nur auf normalen Daten trainiert wird. Zum Schluss updatet der Route-Objekt-Manager noch die Routen, wenn, wenn ein Schiff durch mindestens zwei Wegepunkte gefahren ist. Wenn eine Route bereits existiert, wird diese nur geupdatet, ansonsten wird eine neue Route hinzugefügt. Sobald eine genügend große Anzahl an Schiffen auf dieser Route gefahren ist, wird sie dann aktiviert.

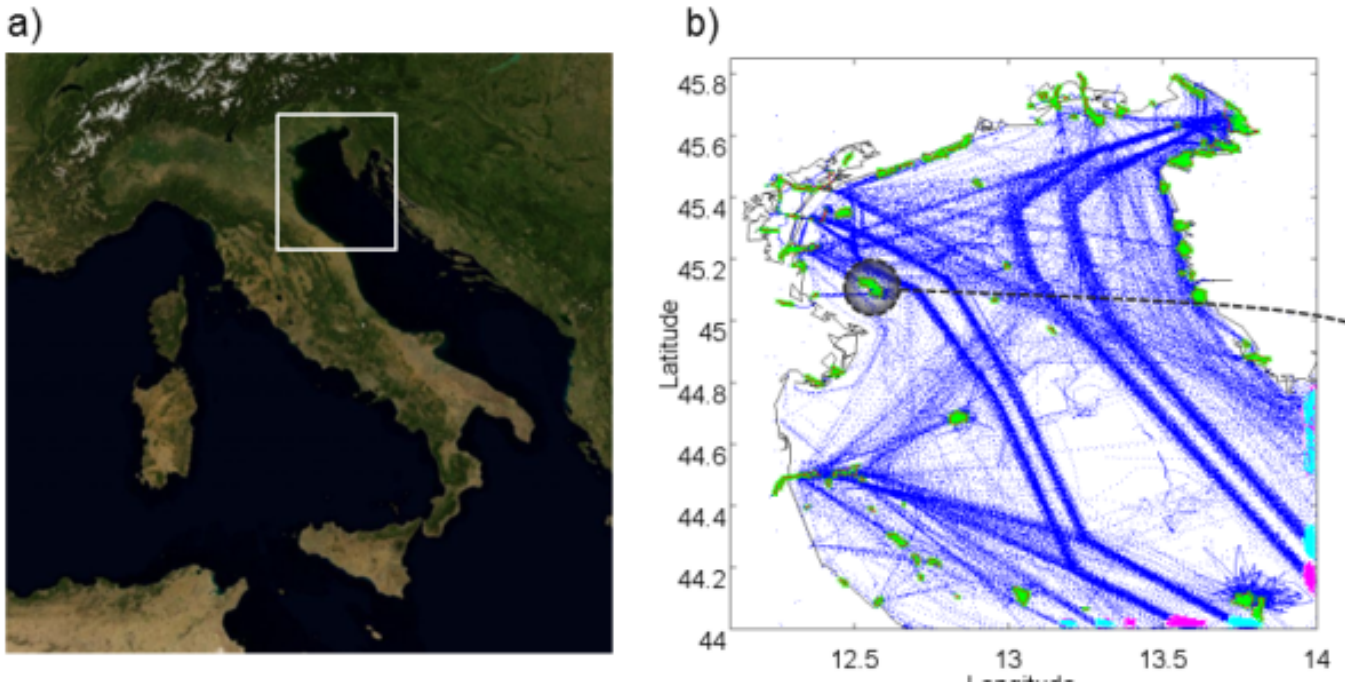


Abbildung 6: Wegpunkterkennung und Routenmustererkennung (Vgl. Pallotta et al. 2013)

Abbildung 6 zeigt die Wegepunkteerkennung und Routenmustererkennung für den gezeigten Bereich in dem Nord-Adratischen-Meer über einen Zeitraum von zwei Wochen. Die grünen Punkte stehen für stationäre Punkte, die türkisen Punkte für die Eingangs- und die pinken Punkte für die Ausgangspunkte.

Mit dem ermittelten Modell der Routen kann dann eine Anomaliedetektion durchgeführt werden. Dazu wird eine Wahrscheinlichkeit berechnet inwieweit die aktuelle Route jedes Schiffes mit den gesammelten historischen Daten übereinstimmen. In Abbildung 7 wird dies exemplarisch an einem Schiff gezeigt dessen Route zuerst im Einklang mit den historischen Daten (dargestellt durch die grauen Pfeile) ist. Während seiner Route dreht dann allerdings das Schiff um und macht eine doppelte U-Drehung. Innerhalb dieser Drehung ist seine Richtung entgegengesetzt zur normalen Fahrtrichtung. Diese Anomalie wurde korrekterweise von TREAD erkannt. [4,5]

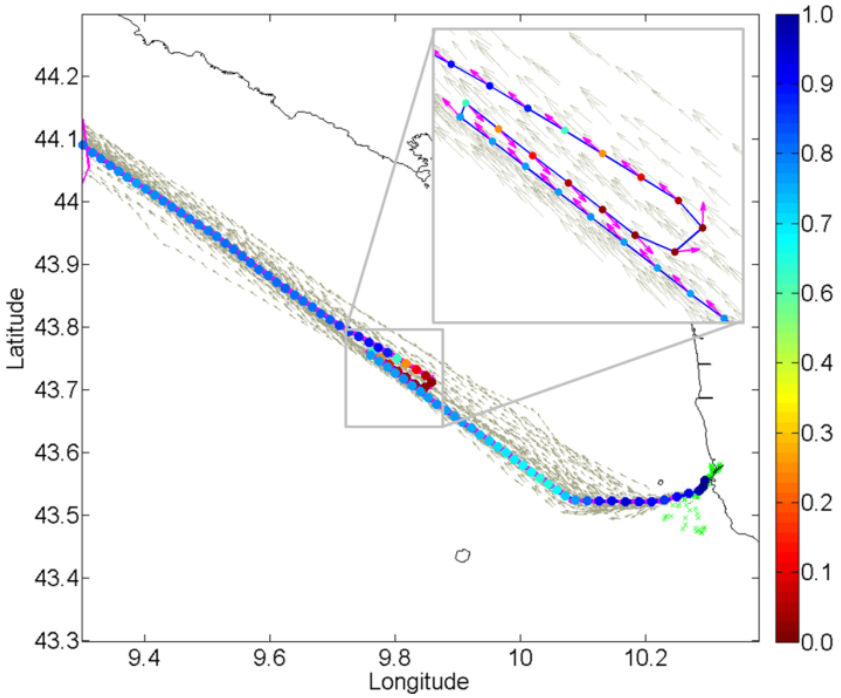


Abbildung 7: Beispiel zur Anomaliedetektion (Vgl. Pallotta et al. 2013)

Weitere Verfahren

Die hier vorgestellten dichte-basierenden Algorithmen gehören zu den bekanntesten Verfahren zur Anomaliedetektion im Allgemeinen und im Kontext der maritimen Schifffahrt. Weitere populäre Verfahren zur Anomaliedetektion im Allgemeinen sind Support Vektor Maschinen, neuronale Netze und Fuzzy-Logik basierende Verfahren. Der Vorteil der hier vorgestellten Verfahren ist allerdings, dass man keine besonderen Vorkenntnisse zum Verstehen und Benutzen der Algorithmen braucht und das sie trotzdem sehr gut bei unüberwachter Anomaliedetektion funktionieren. So benutzt mit TREAD ein bekanntes Verfahren DBSCAN zur Anomaliedetektion im maritimen Kontext. Andere Verfahren im maritimen Kontext benutzen zum Beispiel genetische Algorithmen, in denen die Chromosome durch die Details der Schiffe, ihrer Fracht, usw., sowie die Entscheidungen bezüglich dieses Schiffes.

Bezug zum Projekt

Für eine möglichst intelligente und realistische Schiffssimulation muss ein Schiff in der Lage sein auf ungewöhnliche oder gefährliche Situationen angemessen zu reagieren. Dazu ist es nötig solche Situationen zu erkennen. Dafür benötigt man eine Form von Anomaliedetektion, welche in einem maritimen Kontext schnell und zuverlässig arbeitet, dabei aber nicht zu viele falsch-positive liefert. Mit TREAD wurde ein existierendes Framework vorgestellt, welches gut getestet wurde und diesen Ansprüchen genügend. Das Verfahren kann also als Grundlage für eine Anomaliedetektion benutzt werden. Dafür muss es allerdings für den Kontext unseres Projekts angepasst werden. So müssen unter anderem die Paramater für DBSCAN geeignet gewählt werden oder es muss überprüft werden, welche Informationen, die im AIS gesendet werden, alle mitgespeichert werden müssen. Des Weiteren ist es natürlich auch möglich TREAD zu erweitern, sodass weitere Klassen von Anomalien erkannt werden können. TREAD kann zum Beispiel nicht das Fehlen eines Ereignisses erkennen, welches normalerweise auftritt. Ein Beispiel dafür ist eine Fähre, die immer zu einer bestimmten Uhrzeit eine Route fährt. Des Weiteren kann TREAD auch als Framework zum Vorrausagen der Route eines anderen Schiffes zur Kollisionsverhütung, sowie eine Route für ein Schiff zu ermitteln.

1. https://en.wikipedia.org/wiki/Anomaly_detection
2. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
3. Ester M., Kriegel H.-P., Sander J., Wimmer W., Xu W. (1998). Incremental Clustering for Mining in a Data Warehousing Environment. Proceedings of the 24th VLDB Conference New York, USA.
4. Pallotta G., Vespe M., Bryan K. (2013). Vessel pattern knowledge discovery from AIS data: a framework for anomaly detection and route prediction. In Entropy vol.15 pp. 2218-2245.
5. Pallotta G., Vespe M., Bryan K. (2013). Traffic Knowledge Discovery from AIS Data. Information Fusion (FUSION), 16th International Conference
6. Ajanki A. (2007). https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
7. Kramer O. (2009), Computational Intelligence: Eine Einführung. Springer-Verlag Heidelberg
8. Ester M., Kriegel H.-P., Sander J., Wimmer W., Xu W. (1998). Incremental Clustering for Mining in a Data Warehousing Environment. Proceedings of the 24th VLDB Conference New York, USA.

Präsentation

Architektur autonomer Fahrzeuge

Im Rahmen der Projektgruppe „Intelligente Schiffssimulation“ findet eine Seminarphase über die relevanten Themen statt. Diese Arbeit beschreibt die Architektur autonomer Fahrzeuge und stellt einen Vergleich der Drei-Schichten-Architektur zur Vessel Struktur dar.

Inhaltsverzeichnis

Definitionen

Was ist ein autonomes Fahrzeug?

Unter einem autonomen Fahrzeug versteht man ein Fahrzeug, welches sich entweder vollständig oder zum Teil autonom, also selbstständig, fortbewegt und verhält. Somit spricht man auch öfters von einem „selbstfahrenden Fahrzeug“. Gemeint ist hiermit, dass ein Fahrzeug selbstständig fahren, steuern, einparken oder auch reagieren kann, ohne dass ein Mensch das Fahrzeug bedienen muss. Das Fahrzeug bekommt hierbei seine Informationen aus visuellen Informationsquellen, vergleichbar mit den Informationen die auch einem Fahrer zur Verfügung stehen. Erfolgt nun die Reaktion des Fahrzeugs auf die Informationen selbstständig über Algorithmen und es reagiert dementsprechend, ohne dass ein Fahrer eingreifen muss, spricht man vom autonomen Fahren. Das Fahrzeug kann hierbei zwar sowohl Auto, als auch Flugzeug, Motorrad, Schiff oder Ähnliches sein, im vorliegenden Fall werden jedoch hauptsächlich selbstfahrende Autos behandelt und nachträglich betrachtet was sich auf die Seefahrt übertragen lässt. Des Weiteren unterscheidet man beim autonomen Fahren noch zwischen verschiedenen Stufen bzw. Level des autonomen Fahrens [aut17]

Verschiedene Stufen der Autonomie

Sowohl in Europa, u.a. von der Bundesanstalt für Straßenwesen, als auch in den USA wird die Klassifizierung des autonomen Fahrens in fünf bzw. sechs verschiedene Stufen unterteilt, welche eine Richtlinie für das Thema des autonomen Fahrens vorgeben.

- Level 0: „Driver only“. Der Fahrer führt sowohl die Längsführung des Fahrzeugs (Beschleunigen/Bremsen), als auch die Querführung des Fahrzeugs (Lenken) selbst aus.
- Level 1: Assistiert. Der Fahrer führt entweder die Längs- oder die Querführung des Fahrzeugs aus. Dies bedeutet entweder er bremst und beschleunigt selbstständig, oder er lenkt selbstständig. Die jeweils andere Führung wird von einem Assistenzsystem entweder übernommen oder unterstützt. Der Fahrer muss hierbei durchgehend das System überwachen und jederzeit eingreifen können. Beispiele: Tempomat, Distanzregelung, ABS usw.
- Level 2: Teilautomatisierung. Das System übernimmt entweder zum Teil, oder aber nur in speziellen Situationen sowohl Quer- als auch Längsführung des Fahrzeugs. Auch auf diesem Level muss der Fahrer durchgehend das System überwachen und jederzeit eingreifen können. Beispiele: automatisches Einparken, Spurhaltefunktion, Stauassistent.
- Level 3: Hochautomatisierung. Das System übernimmt für einen bestimmten Zeitraum in spezifischen Situationen Längs- und Querführung. Hierbei muss der Fahrer das System nicht mehr durchgehend überwachen, muss aber bei Bedarf in der Lage sein das Fahrzeug zu übernehmen. Dies könnte beispielsweise ein autonomes Fahren eines Autos auf einer Autobahn sein, was in etwa dem aktuellen technischen Stand entspricht. Gesetzgeber arbeiten daraufhin, Level-3 Fahrzeuge bis 2020 zuzulassen. Ein aktuelles Beispiel hierfür ist der amerikanische Automobilhersteller Tesla, dessen neuestes Modell bereits eine Hochautomatisierung auf Autobahnen zulässt [hei].
- Level 4: Vollautomatisierung. Sowohl Quer- als auch Längsführung des Fahrzeugs werden in einem vorher definierten Anwendungsfall komplett vom System übernommen. Der Fahrer muss das System hierbei nicht überwachen. Sollte sich jedoch etwas Unvorhergesehenes ereignen und das System kann die Aufgaben nicht mehr bewältigen, kann der Fahrer aufgefordert werden das Fahrzeug zu übernehmen. Wenn dies nicht geschieht, versucht das System die Aktion mit dem geringsten Risiko auszuführen.
- Level 5: Kein Fahrer erforderlich. Hierbei wird kein Fahrer mehr benötigt. Es muss nur ein Ziel angegeben werden und das System gestartet werden. Das System bewältigt dann alles von selbst.

[fS12]

Architekturen des Autonomen Fahrens

Anforderungen an Softwarearchitekturen für autonomes Fahren

Neben den üblichen in der Automobilindustrie herrschenden Anforderungen an normale Fahrzeuge, müssen nun bei autonomen Fahrzeugen einige Anforderungen noch einmal verändert werden. Die wichtigste Anforderung an die Softwarearchitektur eines autonomen Fahrzeugs ist die Sicherheit. Da der Mensch bei einem autonomen Fahrzeug nicht mehr eingreifen soll, muss er sich voll und ganz auf das System verlassen können. Dies bedeutet dass das System soweit es geht störungsfrei und ausfallsicher arbeiten sollte. Dies soll durch Kontrollmechanismen gewährleistet werden. Ein weiterer wichtiger Faktor ist die Effizienz des Systems. Da intelligente Systeme wie das autonome Fahren oft um weitere Funktionen erweitert werden, müssen besonders umfangreiche Softwaresysteme unterstützt werden. Diese müssen aufgrund des schnellen technologischen Fortschritts auch zukünftigen Ansprüchen genügen und die Zusammenarbeit von vielen verschiedenen Echtzeitsystemen ermöglichen [Sch15].

Das Drei-Schichten-Modell

Das Drei-Schichten-Modell, ursprünglich auch Drei-Level-Kontrollarchitektur genannt, ist eine Architektur, die speziell für das autonome Fahren von Fahrzeugen in einer dynamischen und unsicheren Umgebung entwickelt wurde. Die Drei-Level-Kontrollarchitektur wurde bereits im Jahr 1997 von den Japanischen Forschern Miura, Ito und Shirai der Universität in Osaka entworfen. Bis zu diesem Zeitpunkt wurde die Architektur in zwei Level unterteilt, dem „tactical level“ und dem „operational level“. Während im taktischen Level die verschiedenen Manöver, welche zum Erreichen des Ziels benötigt werden, bestimmt werden, ist das operative Level dazu da die vorher bestimmten Manöver in echte Operationen des Fahrzeugs umzusetzen. Grund für den Entwurf der Drei-Level-Architektur war die Tatsache, dass die vom taktischen Level getroffenen Entscheidungen, welche Sensoren-Informationen als Grundlage nehmen, relativ unsicher sind, da der Verkehr in einer echten Situation dynamisch und nicht vorhersehbar ist. Deshalb wurde beschlossen, dass bestimmte Verkehrssituationen in der Zukunft vom System vorhergesehen werden sollten, wobei bereits vergangene Situationen als Grundlage zur Bewertung dienen. Hierfür wurde das dritte Level, das „meta-tactical Level“ eingeführt. Dieses soll das taktische Level kontrollieren und nur dann aktivieren, wenn es auch wirklich benötigt wird. Entwickelt wurde diese Drei-Level-Architektur für das autonome Fahren eines Autos auf einer Autobahn. Getestet werden sollte dieses System anschließend in einem Autobahn-Fahrsimulator [eA97].

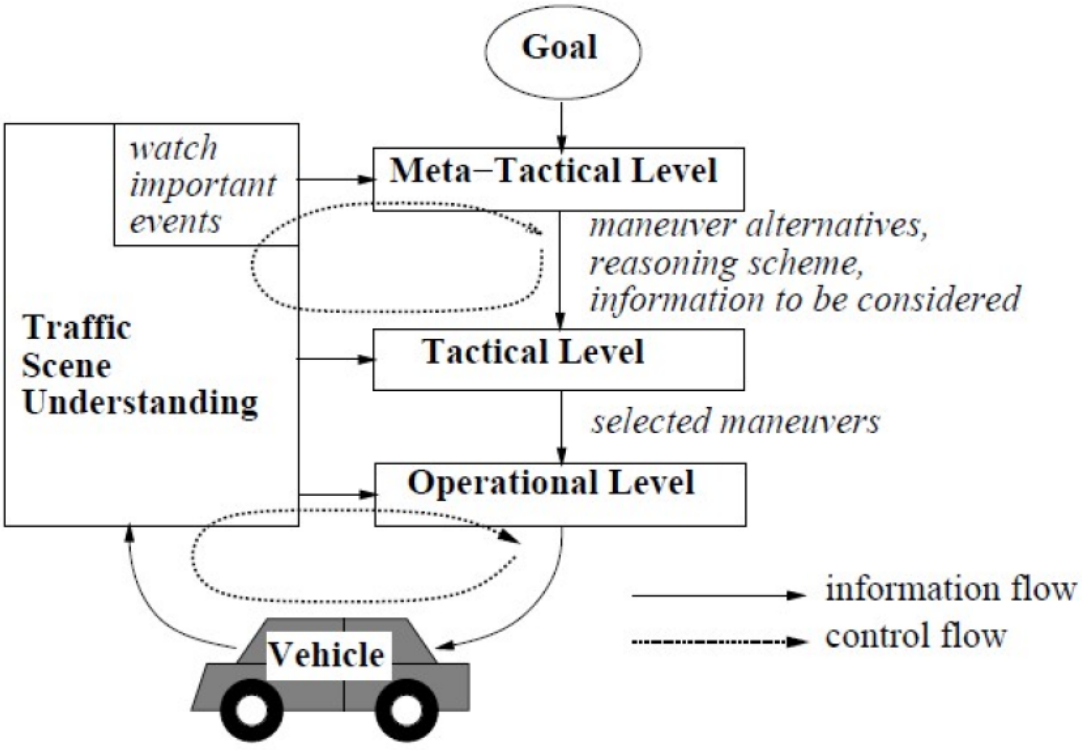


Abbildung 1: Drei-Level-Kontrollarchitektur nach [eA97]

Das Meta-Tactical Level

Das Meta-Tactical Level überwacht durchgehend alle wichtigen Ereignisse im Verkehr. Wichtige Ereignisse können hierbei beispielsweise das Näherkommen des Ziels oder eine Veränderung der aktuellen Geschwindigkeit auf der Spur sein. Des Weiteren berechnet das Meta-Tactical

Level durchgehend die geplante Ankunftszeit am Ziel. Grundlegende Annahmen für dieses Modell sind, dass die Autobahn nur zwei Spuren hat und dass die Geschwindigkeit auf der linken Spur immer höher ist als auf der rechten. Das Ziel der Fahrt ist immer eine vorher ausgewählte Ausfahrt. Da es nun ineffizient wäre, immer alle Ereignisse zu überprüfen, bestimmt das Meta-Tactical Level, welche Ereignisse in der aktuellen Situation wichtig sind und welche nicht. Hierzu wurde von[eA97] ein State-Transition-Graph entworfen, ähnlich einem Zustandsübergangsdiagramm. Dort werden für jeden Zustand die möglichen Ereignisse und die daraus folgenden Aktionen des Tactical Level dargestellt. Befindet sich das Fahrzeug nun Beispielsweise in mittlerer Entfernung zum Ziel auf der rechten Spur, können drei Ereignisse eintreten: Die Geschwindigkeit auf der rechten Spur wird langsamer, die Ankunftszeit verändert sich oder das Ziel kommt näher. Für die ersten beiden Ereignisse würde nun das entsprechende Tactical Level gestartet werden, für das dritte Ereignis würde nur der aktuelle Zustand aktualisiert werden, da dieses Ereignis unwichtig ist.

Bisher ist die einzige Aufgabe des Meta-Tactical Level das auswählen geeigneter Aktionen für das Tactical Level, es kann jedoch auch um weitere Funktionen wie z.B. die Bewertung der verbleibenden Zeit und des aktuellen Platzes auf der Autobahn verwendet werden, um so die auszuführenden Aktionen anzupassen [eA97].

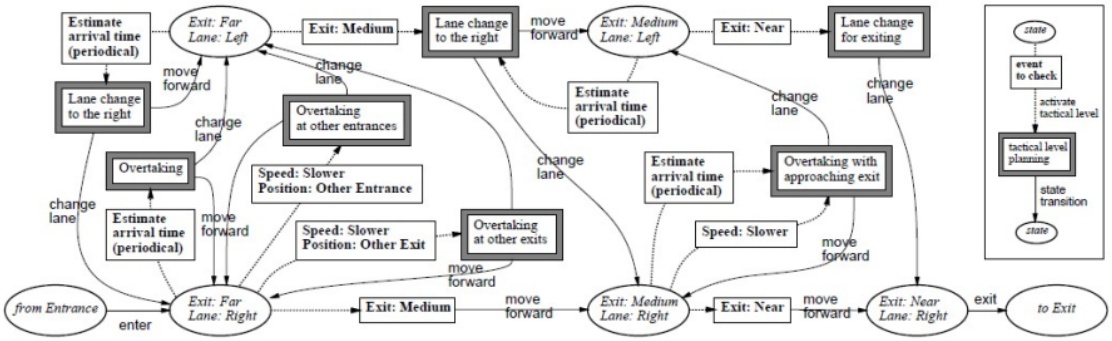


Figure 7: State transition graph for meta-tactical level planning. The meaning of each figure such as an ellipse is explained on the right.

Abbildung 2: State Transition Graph zur Entscheidungsfindung nach [eA97]

Das Tactical Level

Nachdem das Tactical Level vom Meta-Tactical Level aktiviert wurde, bestimmt es aus den vorhandenen Aktionen die beste Alternative. Da der dynamische Verkehr nicht vorhersehbare Faktoren enthält, geschieht die Auswahl der Aktion auf Basis der statistischen Entscheidungstheorie. Wurde die beste Alternative ausgewählt, wird diese zur Ausführung an das Operational Level weitergegeben. Die Hauptziele, welche beim Treffen einer Entscheidung berücksichtigt werden, sind sicher ans Ziel zu kommen und dabei möglichst schnell ans Ziel zu kommen, bzw. in der vorher festgelegten Zeit zu bleiben. Bei der Entscheidungsfindung spielen viele verschiedene Faktoren eine Rolle. Diese Entscheidungsfindung wird von [eA97] anhand des folgenden Beispiels erklärt.

Die geschätzte Zielzeit ist t. Die vorgegebene Zielzeit, welche zu Beginn der Fahrt ermittelt wurde ist t-target. Die Verlustfunktion der Zielzeit L(t) beschreibt die Sollvorgabe an t mit t-target. Je nach Auswahl einer leichten oder engen Sollvorgabe lässt sich die Verlustfunktion dadurch wie in Abbildung 3 darstellen. Nun müssen noch die Zusatzkosten eines Manövers berücksichtigt werden, welche mit der Konstante C beschrieben werden [eA97].

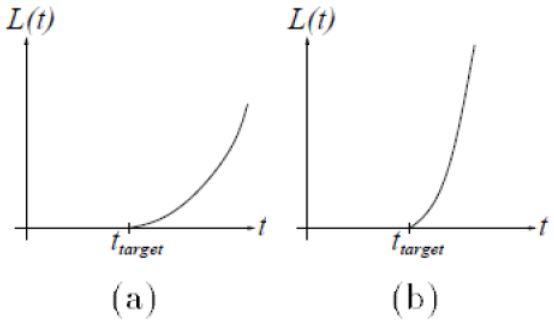


Abbildung 3: Verlustfunktion mit leichter (a) und enger (b) Vorgabe [eA97]

Sollen nun zwei Optionen A und B verglichen werden, wobei A einen Spurwechsel enthält und B auf der Spur bleibt, dann wird Option A gewählt wenn
$L(tB) > L(tA) + C$ (1)
In diesem Fall ist der Verlust der Option A inklusive Zusatzkosten C geringer als der Verlust der Option B. Gilt nun jedoch
$L(tB) < L(tA) + C$ (2)
dann würde Option B gewählt werden, da der Verlust hierbei geringer ist. Dieses Beispiel wird in Abbildung 4 dargestellt [eA97].

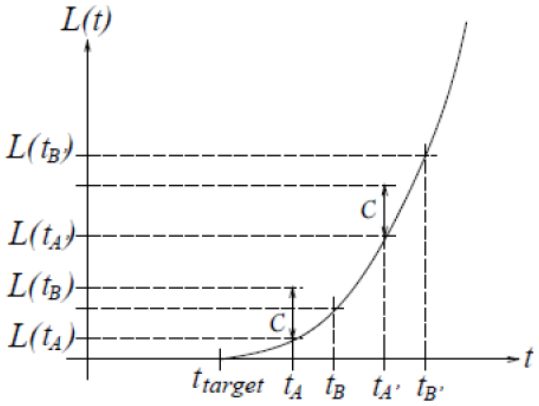


Abbildung 4: Auswahl des Manövers [eA97]

Das Operational Level

Das Operational Level ist für die Ausführung der zuvor ausgewählten Manöver zuständig. Auf diesem Level werden die Manöver in Kontroll-Befehle übersetzt und an das Auto weitergegeben. So werden bei einem Überholmanöver beispielsweise die Befehle zum Blinken nach links und zum Lenken an das Auto gegeben. Des Weiteren ist das Operational Level für Reaktionen des Fahrzeugs verantwortlich. So werden z.B. in Notfallsituationen Befehle zum Abbremsen direkt an dieses Level gegeben, da die Befehle so nicht erst die anderen Level durchlaufen und dadurch Zeit gespart wird [eA97].

Fazit

Dieses Modell stellt eine Softwarearchitektur für das autonome Fahren da, welche aus drei Leveln besteht; Das Meta-Tactical Level, welches für die zeitweise Aktivierung des Tactical Levels verantwortlich ist, das Tactical Level, welches für die Auswahl der Manöver zuständig ist und das Operational Level, welches für die Steuerung des Fahrzeugs verantwortlich ist. Zum Testen dieses Modells wurde es erfolgreich in einem Autobahn-Simulator getestet. Hierfür wurden drei Fahrzeuge getestet, einmal ein aggressives Fahrzeug, welches immer die Spur wechselt, ein defensives Fahrzeug, welches auf der rechten Spur bleibt, und ein intelligentes Fahrzeug, welches die Drei-Level-Architektur nutzt. Die Tests haben hierbei ergeben, dass das intelligente Fahrzeug am schnellsten und effizientesten ans Ziel kam. Somit lässt sich das Modell als Erfolg bezeichnen und als Grundlage für weitere Entwicklungen im Bereich des autonomen Fahren verwenden [eA97].

Autonome Fahrzeuge in der Praxis

In diesem Kapitel wird nun einmal geschaut, wo und wie Architekturen autonomer Fahrzeuge in der Praxis zum Einsatz kommen. Hierfür wird die Darpa Grand Challenge vorgestellt und anschließend einer der Teilnehmer genauer betrachtet.

Darpa Grand Challenge

Die DARPA Grand Challenge war ein Wettbewerb der Defense Advanced Research Projects Agency (DARPA) des US-amerikanischen Verteidigungsministeriums, bei dem autonome Fahrzeuge ohne Fahrer gegeneinander angetreten sind. Durch die Ausschreibung eines Preisgeldes für die Gewinner, also diejenigen deren Autos als erste durch das Ziel fahren, sollte die Entwicklung autonomer Fahrzeuge vorangetrieben werden.

Die Grand Challenge wurde bisher drei mal durchgeführt. Bei der ersten Challenge im Jahr 2004 betrug das Preisgeld 1 Million Dollar und die Strecke war 240 km lang. Der Weg führte durch die Wüste Nevadas. Bei dieser Challenge ist keins der teilnehmenden Teams ins Ziel gekommen. Das beste Fahrzeug schaffte lediglich 12 km.

Im Jahr 2005 wurde die zweite Challenge durchgeführt. Das Preisgeld betrug diesmal 2 Millionen Dollar für das erste Team, welches unter 10 Stunden die Strecke von 212 km zurücklegte. Dieses Mal schafften 5 von 195 Teams die gesamte Strecke, wobei sich das Team der Stanford University als Sieger durchsetzte.

Bei der Grand Challenge im Jahr 2007 führte die Strecke nicht mehr durch die Wüste, sondern auch durch befahrenes Gebiet, was zur Folge hatte dass erstmals auch Straßenverkehrsregeln beachtet werden mussten. Das Preisgeld für den Sieger lag wieder bei 2 Millionen Dollar, die Strecke war jedoch nur knapp 100 km lang. Erstmals nahmen auch zwei Teams der Uni Karlsruhe und der TU Braunschweig aus Deutschland teil, welche es jedoch nicht bis ins Ziel schafften [dar].

Beispiel: Softwarearchitektur des Fahrzeugs von Team Annieway

Im Folgenden soll einmal grob geschaut werden, wie ein Fahrzeug, welches an der Darpa Grand Challenge teilgenommen hat, aufgebaut ist,

bzw. ob Vergleiche zum zuvor behandelten Drei-Schichten-Modell gezogen werden können.

Das Team Annieway war das Team der Universität Karlsruhe, welches an der Darpa Grand Challenge 2007 teilgenommen hat. Bei dem autonomen Fahrzeug des Teams handelt es sich um einen VW Passat. Der Passat ist ausgestattet mit vielen verschiedenen Sensoren zur Umgebungs- und Positionserkennunug. Dazu gehören beispielsweise Stereo Kameras, ein optisches Abstandsmessgerät, ein inertial GPS Messsystem, ein Radarsystem und mehreren Kommunikationssystemen. Als Herzstück des Autos dienen mehrere CPUs, betrieben von mehreren Batterien. Im Rahmen dieser Arbeit ist jedoch nur die Architektur der Software von Bedeutung, bzw. die Struktur des intelligenten Verhalten. Diese soll nun einmal mit dem bereits behandelten Drei-Schichten-Modell verglichen werden.

Die Entscheidungsfällung des Menschen beim Steuern eines Fahrzeugs wurde in drei Kategorien unterteilt: einer reaktiven, taktischen und strategischen Kategorie. Die reaktive Kategorie umfasst spontane bzw nicht geplante Aktionen, wie z.B. dem Ausweichen von Hindernissen. Taktische Aktionen sind solche, die mehr bewusst ausgeführt werden, wie z.B. dem halten vor einer roten Ampel. Diese taktischen Aktionen müssen vom Fahrer durchgehend aktiv geplant werden, haben jedoch keinen Einfluss auf das übergeordnete Ziel. In die strategische Kategorie fallen langfristig geplante Aktionen, wie z.B. dem Folgen der Straße oder dem Abbiegen an einer Kreuzung.

Hieraus hat das Team Annieway sein Konzept zum Verhalten des Fahrzeugs abgeleitet und entsprechend den drei genannten Kategorien drei Schichten zur Reaktion des Fahrzeugs entwickelt, welche die selben Aufgaben haben [ann].

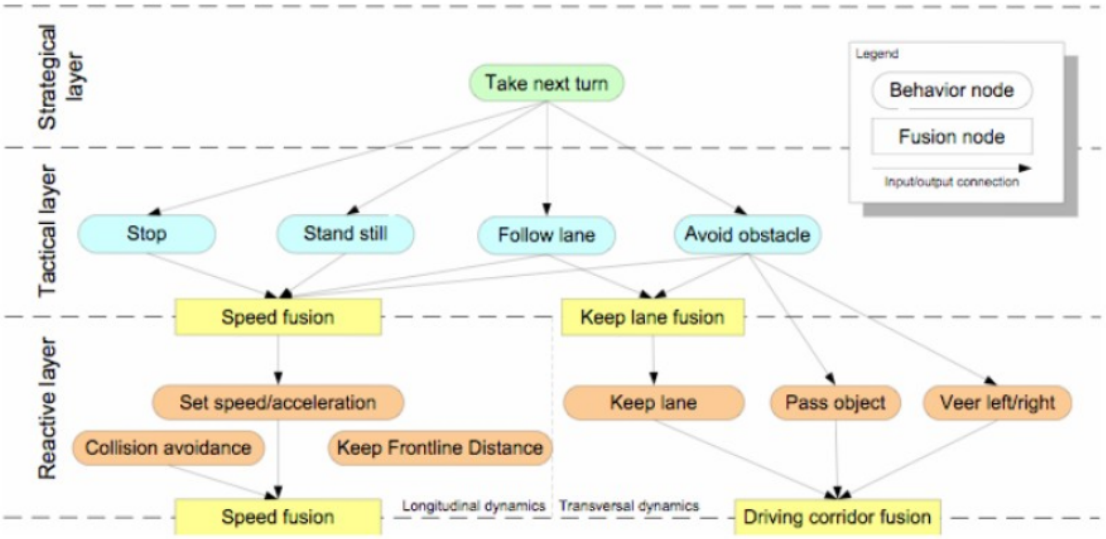


Abbildung 5: Architektur des Verhaltenskonzept des Fahrzeugs von Team Annieway [ann]

Beim betrachten dieser Architektur lässt sich feststellen, dass dieses Prinzip große Ähnlichkeiten zu dem bereits behandelten Drei-Schichten-Modell aufweist. Die Strategische Ebene entspricht in etwa dem Meta-Tactical Level. Hier werden auch grundlegende Entscheidungen getroffen und zielführende Aufgaben geplant. Diese werden dann weitergegeben an die taktische Ebene. Diese taktische Ebene entspricht dem Tactical Level des Drei-Schichten-Modell. Hier werden Aktionen ausgewählt, welche durchgeführt werden sollen. Diese Aktionen werden dann wieder weitergegeben an die reaktive Ebene, welche die selben Aufgaben hat wie das Operational Level. Es werden die Aktionen ausgeführt und gleichzeitig Reaktionen auf Veränderungen oder Notfälle verarbeitet[ann].

Dies untermauert die These des Drei-Schichten-Modells noch einmal und zeigt in der Praxis, dass der Entwurf von 1997 zehn Jahre später in einem echten Fahrzeug Anwendung findet und Erfolg hat.

Vergleich des Drei-Schichten-Modell mit der Vessel Struktur

Im Rahmen der Projektgruppe Intelligente Schiffssimulation soll nun einmal die gegebene Vessel Struktur mit dem behandelten Drei-Schichten-Modell verglichen werden. Ziel ist es, herauszufinden, ob und inwiefern sich das Modell für die Entwicklung einer intelligenten Schiffssimulation benutzen lässt.
Die Vessel Struktur ist eine Architektur zur Darstellung eines simulierten Schiffes. Zu Beginn werden die strategischen Ziele vom Traffic Generator festgelegt, also das Ziel des Schiffes gesetzt. Diese werden dann an das Schiff gegeben. Dieses Schiff hat nun eine Verhaltensebene, auf welcher operative Ziele festgelegt werden. Dies bedeutet, dass auf dieser Verhaltensebene Aktionen wie das Ausweichen oder Überholen entschieden werden. Diese Entscheidungen sind von der Situation abhängig, haben aber keinen Einfluss auf das ursprüngliche Ziel des Schiffes. Diese operativen Entscheidungen werden nun an die Track Control weitergegeben, wo die jeweiligen physischen Aktionen ausgeführt werden. Hier wird nun das Schiff je nach Aktion beschleunigt, gebremst oder aber das Ruder bewegt. Zuletzt erfasst das Dynamic Modell diese Bewegungen und ermittelt somit die aktuelle Position des Schiffes. Dieses Modell wird in Abbildung 6 dargestellt.

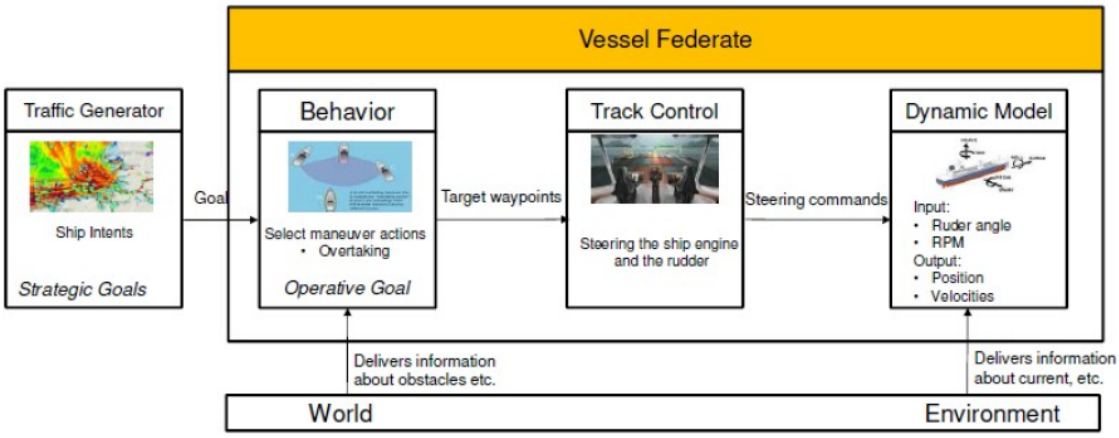


Abbildung 6: Vessel Structure

Das Ziel im Rahmen dieser Projektgruppe ist nun, in einer bereits gegebenen Simulationsumgebung den Verhaltensteil der Vessel Struktur, also die operative Ebene, mit einer entsprechenden Intelligenz zu füllen, sodass die Schiffe von alleine auf äußere Einflüsse und andere Situationen reagieren können. Hierfür soll die Vessel Struktur nun einmal mit dem Drei-Schichten-Modell verglichen werden, um herauszufinden ob sich das Modell in die Seefahrt übertragen lässt.
Zunächst haben beide Modelle einmal gemeinsam, dass ein strategisches Ziel zu Beginn festgelegt wird. Beim Drei-Schichten-Modell wird der Zielort vor Abfahrt festgelegt, bei der Vessel Struktur gibt es den Traffic Generator, der das Ziel festlegt. Nun folgt beim Drei-Schichten-Modell das Meta-Tactical Level, welches festlegt, ob reagiert werden muss und wie wichtig eine Reaktion auf äußere Einflüsse ist. Diese Ebene ist bei der Vessel Struktur nicht vorhanden, würde aber durchaus Sinn machen, da es effizienter ist, erst abzuwägen ob Handlungsbedarf besteht. Als nächstes folgt beim Drei-Schichten-Modell das Tactical Level, welches für die Auswahl der Manöver zuständig ist. Bei der Vessel Struktur steht an dieser Stelle das Behavior, welches die selbe Funktion hat. Auf dieser operativen Ebene werden im Vergleich zum Drei-Schichten-Modell die ersten beiden Schichten in einer Ebene vereint. Es wird entschieden ob und wie gehandelt werden muss und dann entsprechend das jeweilige Manöver ausgewählt. Als nächstes folgt beim Drei-Schichten-Modell das Operational Level, welches für die Ausführung der physischen Aktionen zuständig ist. Die Track Control im Vessel Modell hat auch hier wieder die selbe Funktion. Bei der Vessel Struktur werden nun die ausgeführten Befehle noch an das Dynamic Modell weitergegeben, welches dadurch Daten wie die Position bestimmt. Dieses ist beim Drei-Schichten-Modell nicht gegeben, hier werden die Daten durchgehend ermittelt.

Fazit

Nachdem nun viele Gemeinsamkeiten der beiden Modelle festgestellt wurden, kann man abschließend sagen, dass sich das Drei-Schichten-Modell durchaus dafür eignet, es auf eine maritime Umgebung zu übertragen. Die Autobahn des Drei-Schichten-Modell lässt sich mit einer Seestraße vergleichen. Genau wie auf der Autobahn müssen hier ähnliche Aspekte beachtet werden, wie z.B. der Geschwindigkeit oder aber Ausweich- bzw. Überholmanöver. Da die Aufgabe in dieser Projektgruppe darin besteht, das operative Verhalten des Schiffes zu automatisieren, könnte ein Ansatz darin bestehen, sowohl das Meta-Tactical Level als auch das Tactical Level in einer Ebene zu vereinen und diese zur Entscheidungsfällung des Schiffes und zum Festlegen der operativen Ziele zu benutzen. Dies würde durchaus Sinn machen, da so die einzelnen Konzepte zur Entscheidungsfindung aus dem Tactical Level benutzt und auf das Vessel Modell übertragen werden könnten.

Literaturverzeichnis

[ann] Technische daten des team annieway. http://archive.darpa.mil/grandchallenge/TechPapers/Team A
[aut17]https://www.daimler.com/innovation/autonomes-fahren/special/definition.html, 2017.
[dar] http://archive.darpa.mil/grandchallenge/index.html.
[eA97] J. Miura et. Al. A three-level control architecture for autonomous vehicle driving in a dynamic and uncertain traffic environment. In *Proc. IEEE Conf. on Intelligent Transportation Systems*, Osaka, Japan, 1997.
[fS12] Bundesanstalt fu¨r Straßenwesen. *Rechtsfolgen zunehmender Fahrzeugautoma- tisierung*. http://www.bast.de/DE/Publikationen/Foko/Downloads/2012-11.pdf, Bergisch Gladbach, 2012.
[hei] heise.de. https://www.heise.de/newsticker/meldung/Tesla-wirbt-mit-Video-fuer- das-autonome-Fahren-3492432.html.
[Sch15] Kevin Schmidt. Softwarearchitekturen für automatisier- tesundautonomes fahren.https://www.uni-koblenz- landau.de/de/koblenz/fb4/ist/AGZoebel/Lehre/Sommer2015/SeminarASidA/A1, 2015.

Präsentation

HAGGIS Architektur

Im Rahmen der Projektgruppe „Intelligente Schiffssimulation“ findet eine Seminarphase über die relevanten Themen statt. Diese Arbeit beschreibt den Aufbau des Frameworks HAGGIS und ordnet dieses in den Kontext der integrierten Testumgebung eMIR ein.

Inhaltsverzeichnis

Einleitung

Die Sicherheit in der Seefahrt ist schon immer ein zentraler Aspekt. Essentiell für Sicherheit ist die eingesetzte Technik. Dies gilt auch für e-Navigation. Dahinter verbirgt sich eine Strategie der International Maritime Organization (IMO), die durch die Einführung von fünf priorisierten e-Navigation Lösungen eine höhere Sicherheit der kommerziellen Seefahrt verspricht. Der Strategie-Plan der IMO beinhaltet folgende fünf Aufgaben:

- Verbessertes, harmonisiertes und nutzerfreundliches Brückendesign
- Hilfsmittel für standardisiertes und automatisiertes Reporting
- Verbesserte Zuverlässigkeit, Belastbarkeit und Integrität von Brückenequipment und Navigationsinformationen
- Integration und Präsentation von verfügbaren Informationen, die durch Kommunikationsmittel empfangen wurden, in grafischen Oberflächen
- Verbesserte Kommunikation des Vessel Traffic Services (VTS)

(vgl. IMO, 2012)

Aus diesen formulierten Aufgaben und Zielen lässt sich die erhöhte Erwartungshaltung an zukünftig entwickelte Software ableiten. Zum einen soll sich auf die Qualität der Software fokussiert werden und zum anderen steht der Mensch im Zentrum des Designs. Um die Sicherheit von e-Navigation-Technologien garantieren zu können ist ein ganzheitlicher Ansatz von Nöten, der das gesamte soziotechnische System, das Zusammenspiel von Mensch und Maschine, in seine Umgebung einbezieht (vgl. Hahn et Al., 2014). Für diesen Zweck wurde die Plattform eMIR (eMaritime Reference) geschaffen. Diese setzt sich aus unterschiedlichen Komponenten zusammen, dies ist einerseits eine virtuelle Simulationsumgebung (HAGGIS), als auch eine physische Umgebung und Testplattform (LABSKAUS). HAGGIS unterstützt die Modellierung und formale Analyse von e-Navigationssystemen in einer Co-Simulationsumgebung, unter Einbeziehung von einer Verkehrs- und n-Körpersimulationssystemen. Ebenso Bestandteil von HAGGIS ist ein Modell eines menschlichen Agenten. LABSKAUS hingegen beinhaltet ein experimentelle VTS-System, eine mobile Schiffsbrücke und einen Referenzhafen sowie einen Referenzwasserweg für e-Navigationsexperimente und Demonstrationen. (vgl. Hahn et Al. 2014)

Zentraler Gesichtspunkt dieser Arbeit wird die Beschreibung von HAGGIS sein. Es wird auf dessen Komponenten eingegangen sowie deren Bedeutung für die Projektgruppe „intelligente Schifffahrtssimulation“ (iSS).

Integrierte maritime Testumgebung

Neue Technologien können in Simulationen und in echten Umgebungen getestet werden. Echte maritime Testumgebungen sind teuer und ihrer Verfügbarkeit beschränkt. Dadurch liegt es nahe auf Simulationen zurückzugreifen, um die theoretische Tauglichkeit von neuen Technologien zu beweisen. Überdies trägt die offene Gestaltung, im Sinne der freien Verfügbarkeit, dazu bei, neue Technologien durch eine einfach zugängliche Simulationsumgebung schneller vorwärts zu bringen (vgl. Hahn, Noack, 2014). Unter diesen Voraussetzungen wurde die integrierte Testplattform eMIR geschaffen.

HAGGIS ist ein zentraler Bestandteil dieser Plattform. In dieser Simulationsumgebung können maritime Verkehrssituation, Sensoren, Umgebungen und menschliches Verhalten simuliert werden.

Ein weiterer Bestandteil ist der Hafen von Rostock/Warnemünde. Dort wird versucht realistisches Manöververhalten von Schiffen zu simulieren. Ein System ist das MGBAS (maritime Ground Based Augmentation System), mit dessen Hilfe GNSS-Daten (Globales Navigationssatelitensystem) ausgewertet und evaluiert werden. Das MSCW (Maritime Simulation Centre Warnemünde) komplettiert die Forschungsinfrastruktur, indem es Simulationseinrichtungen und echte Daten aus MGBAS miteinander kombiniert.

LABSKAUS bildet die physische Testumgebung in eMIR und ist in der Deutschen Bucht stationiert. Zu dieser Umgebung gehört ein Referenzwasserweg, welcher durch eine spezielle Überwachung ausgezeichnet wird. Es werden dort maritime Überwachungssysteme genutzt,

um den Seeverkehr mit seinen ganzen Facetten aufzuzeichnen.

Eine mobile Brücke komplettiert eMIR. Sie kann in echte Brücken integriert werden oder auch, mithilfe von anderen Komponenten von eMIR als Präsentationswerkzeug eingesetzt werden. (vgl. Hahn, Noack, 2014)

HAGGIS

HAGGIS ist eine virtuelle Simulations- und Modellierungsumgebung, die es ermöglicht neue e-Maritime Technologien zu testen, ohne den Einsatz von physischem Equipment auf hoher See. Um dieses Ziel zu erreichen wurde HAGGIS aus mehreren Modulen zusammengesetzt, die beliebig in verschiedenen Applikationen orchestriert werden können. (vgl. Schweigert et. al., 2014)

Die Komponenten von HAGGIS basieren auf dem Verteilungsansatz des verteilten gemeinsamen Speichers, wodurch die Interoperabilität der einzelnen Komponenten untereinander sichergestellt wird. Darauf aufbauend basiert die daraus resultierende Architektur der Co-Simulation auf dem High Level Architecture (HLA) Standard für Co-Simulationen (“IEEE 1516,” 2000). HLA sieht vor, dass die gesamte Simulation (Federation) in einzelne Teile (Federates) zerlegt wird. Um die Kommunikation zwischen den einzelnen Federates zu ermöglichen, sind diese über eine Laufzeitinfrastruktur miteinander verbunden.

In Abbildung 1 ist die Gesamtarchitektur von HAGGIS abgebildet. Zu sehen sind dort die einzelnen Komponenten, die in den folgenden Abschnitten näher erläutert werden. Zudem ist der Unterschied zwischen Design Zeit und Laufzeit zu sehen. Zur Design Zeit werden das Scenario und das zu untersuchende Experiment definiert. Während der Laufzeit wird das Experiment mithilfe verschiedener Simulationen, die beliebig zusammen orchestriert werden können, durchgeführt. Eine gleichzeitige Analyse sowie Beobachtung ist während der Laufzeit möglich. Die Architektur unterliegt der HLA. Als Datengrundlage wird das S-100 Datenmodell verwendet. Das Datenmodell S-100 wird an dieser Stelle nicht weiter erläutert, es wird auf die spezifische Seminararbeit verwiesen.

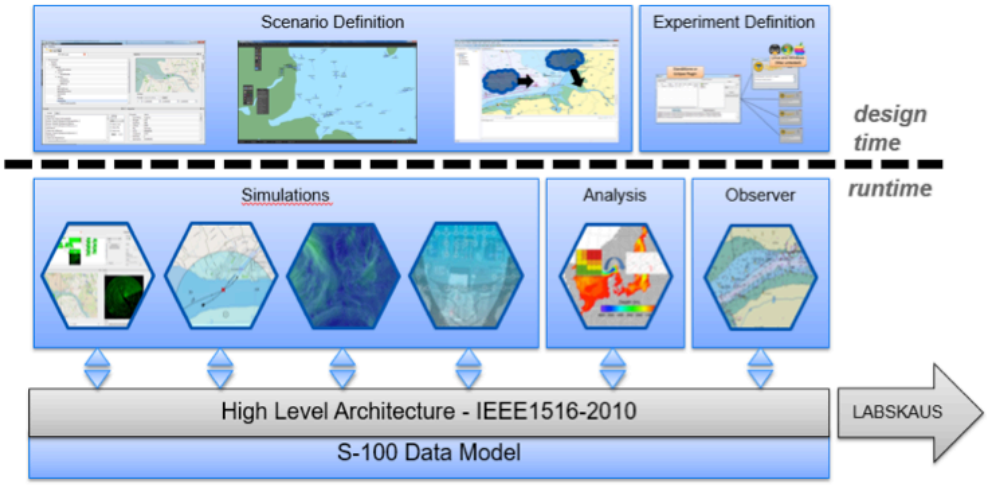


Abbildung 1: Gesamtarchitektur des HAGGIS Frameworks (Gollücke, 2017)

Die folgenden Abschnitte sind nach den Komponenten des HAGGIS Frameworks gegliedert und beschreiben diese näher.

Szenario Definition

Während der Szenario Definition, die während der Design Zeit stattfindet, werden grundsätzliche Parameter des zu untersuchenden Anwendungsfalls definiert. Dazu zählt die Anzahl der Schiffe sowie deren individuelle Ausgestaltung. Spezifische Attribute sind die MMSI (Maritime Mobile Service Identity), IMO-Nummer, Rufzeichen und Schiffsname. Weitere Merkmale definieren den globalen Standort und die Größe des Schiffes. Sensoren können den Schiffen optional zugeordnet werden. (vgl. Schweigert et. al., 2014)

Szenario Konfiguration

Die Konfiguration der Co-Simulation besteht aus der Verteilung der HLA Laufzeit-Infrastruktur und der Simulationskomponenten. Das bedeutet, dass die Komponenten benannt werden müssen die für die Simulation des Szenarios erforderlich sind.

Während der Konfiguration benutzt das Werkzeug DistriCT die Informationen über die Simulationskomponenten, das Testsystem und die Kommunikation der Laufzeitinfrastruktur, um die beteiligten Softwarekomponenten auf verschiedenen Plattformen zu verteilen. (vgl. Schweigert et. al., 2014)

DistriCT verfügt dafür über eine zentrale Kommunikationskomponente, in der alle aktuell kommunizierten Daten der Simulatoren hinterlegt sind. Dieses dient der Risikodistanzberechnung, auf die explizit im Abschnitt 3.5 der automatischen Auswertung eingegangen wird. Darüber hinaus besitzt DistriCT Kontrollkomponenten mit denen Simulator-Programme auf unterschiedlichen Systemen gestartet und gestoppt werden können

oder Zustände gespeichert und geladen werden können. DistriCT ermöglicht auch die Parameterkonfiguration während der Laufzeit der Simulation, um auf eingehende Analyseergebnisse der Kontrollkomponenten reagieren zu können. Somit trägt DistriCT zu einer deutlich erhöhten Flexibilität von HAGGIS bei. (vgl. Gollücke, 2017)

Nachdem die HLA Laufzeit-Infrastruktur und die Simulatoren spezifiziert und die Parameterkonfiguration abgeschlossen wurde, wird der Simulationsplan generiert. Der Simulationsplan ist ein konfigurierbarer Sequenzfluss, der die Simulationsdurchläufe beschreibt.

Nachdem das Szenario modelliert, der Simulationsplan erstellt und die Laufzeitkomponenten auf verschiedene Systeme verteilt wurden, kann die Simulation gestartet werden. (vgl. Schweigert et. al., 2014)

Maritime Traffic Simulation

Die Maritime Traffic Simulation (deutsch: Maritime Verkehrssimulation, kurz: MTS) benötigt für die Initialisierung Daten aus der Szenario Definition, insbesondere die Daten der modellierten Schiffe. Dafür nutzt es die Datei, die zur Beschreibung des Szenarios von DistriCT erstellt wurde. Die Vorbereitung umfasst eine Wegeplanung für die Schiffe, denen kein exakter Weg vorgegeben wurde. Die MTS nutzt für die Wegplanung einen A* Algorithmus über ein 50x50m Gitter, das auf Basis einer Seekarte generiert wird. Die Projektgruppe MTSS hat sich dafür entschieden den A* Algorithmus ohne ein fest definiertes Raster zu implementieren, da dies in bestimmten Szenarien zu unbefriedigenden Ergebnissen führen könnte (vgl. MTSS, 2016).

Der A* Algorithmus gehört zur Gruppe der informierten Suchalgorithmen. Der Algorithmus basiert auf Dijkstras Algorithmus zur Suche nach dem kürzesten Weg innerhalb eines Graphen. Mithilfe von Heuristiken kann eine gute Performanz erreicht werden (vgl. Reddy, 2013) die dem Gesamtsystem HAGGIS zuträglich ist. Die Wegplanung berechnet nicht nur Wegpunkte auf dem vorgenannten Gitternetz, sondern bezieht auch den Tiefgang des konfigurierten Schiffes, die Wassertiefe und Informationen auf der Seekarte mit ein. Der eingesetzte Algorithmus ist auch in der Lage Wasserstraßen und Kanäle abzufahren und ist somit in der Lage die kürzeste Route zwischen zwei Wegpunkten unter realistischen Bedingungen zu errechnen.

Abbildung 2: Beispiel einer Route mit dem A*-Algorithmus (MTSS, 2016)

Abbildung 2 visualisiert den vorher beschriebenen Sachverhalt an dem praktischen Beispiel der Emsmündung und dreier Schiffe mit deren Wegberechnung. Weitere Informationen sind in der Projektdokumentation (MTSS, 2016) des Projekts „Maritime Transportation Systems Simulation“ (MTSS) enthalten.

Diese Details ließen sich von der Projektgruppe iSS wiederverwenden. Darauf aufbauend könnte eine noch dynamischere Wegplanung realisiert werden, die weitere Gefahren, wie z. B. Kollisionsgefahren mit anderen Schiffen einbezieht und darauf reagiert. Dafür müssen die COLREGs in der Implementation intelligenten Verhaltens implementiert werden, damit möglichen Gefahren dynamisch begegnet werden kann. Zudem müssen andere Faktoren dynamischer Natur berücksichtigt werden, z. B. Gezeiten, Wetter, sonstige Hindernisse.

Sensor Simulation

Die Sensor Simulation bietet die Möglichkeit Sensormessungen, wie sie in der echten Welt vorkommen, darzustellen. Dies wird über einen nur lesenden Zugriff auf die Simulationsdaten und die daraus abgeleitete Transformierung der Sensordaten bewerkstelligt. Während der Transformierung werden die Simulationsdaten so angepasst, z. B. werden AIS Daten nur in für einen gewissen Raum ausgegeben, dass sie der Darstellung in der echten Welt bestmöglich nachempfunden werden. Ein wichtiger Teil der Sensor Simulation ist der AIS Emitter. AIS ist gemäß den Vorschriften des SOLAS-Übereinkommens, einer Sonderorganisation der Vereinten Nationen, der Internationalen Seeschifffahrts-Organisation (IMO), auf Schiffen ab einer bestimmten Größe (Bruttoraumanzahl >300) zwingend mitzuführen (vgl. SOLAS Kapitel V, 2002). Der AIS Emitter bezieht unterschiedliche Sensordaten mit ein. So ist beispielsweise der Kurs über Grund ein Teil einer AIS Nachricht, diese Information wird aus der Berechnung der zurückgelegten Strecke und der dafür benötigten Zeit ermittelt. Dies geschieht aus den Daten der Simulation oder, falls entsprechende Sensoren dem Schiff während der Design Zeit zugeordnet wurden, aus den Daten der entsprechenden

Sensoren. Ist ein GPS Sensor einem Schiff zugeordnet, so nimmt dieser die Position des Schiffes, die durch die Simulation errechnet wurde und ordnet dieser Position einen statistischen Fehler zu, um eine realistischere Darstellung zu ermöglichen.

Weitere Elemente der Sensor Simulation simulieren die Funkübertragung, über die AIS-Daten übertragen werden. (vgl. Schweigert, et al., 2014)

Automatische Auswertung

Für die automatische Auswertung eines Simulationsdurchlaufes wird auf das Werkzeug DistriCT zurückgegriffen. Die Updates der einzelnen Simulatoren werden auf die korrespondierenden Datenobjekten der Analyseinstanz abgebildet. Dies erlaubt dem Nutzer eine Berechnung der Distanz zu einem beobachteten Risiko. Für die Berechnung der Distanz wurde eine Risiko Monitor Komponente entwickelt. DistriCT wird in der automatischen Auswertung dafür benutzt, um bei Überschreiten eines gewissen Schwellwertes innerhalb der der Distanzfunktion die aktuellen Status der Simulatoren zu speichern. Diese gespeicherten Status werden genutzt, um die kritischen Situation zu analysieren. (vgl. Schweigert, et al., 2014)

Fazit

In der vorliegenden Arbeit wurde die Architektur HAGGIS zunächst in das Testumgebung eMIR eingeordnet. Neben der Simulationsumgebung HAGGIS beinhaltet eMIR auch eine physische Testumgebung, LABSKAUS. Für die Projektgruppe iSS sind die Komponenten von HAGGIS, die im Kapitel 3 näher beschrieben wurden von besonderer Bedeutung. Wichtig dabei ist die Unterscheidung zwischen den Komponenten der Design- und der Laufzeit der Simulation. Während der Designzeit wird das zu simulierende Szenario modelliert, die teilnehmenden Schiffe mit ihren technischen Spezifikationen definiert und die Simulatoren auf unterschiedlichen Knoten über das Werkzeug DistriCT verteilt. Während der Laufzeit wird durch die einzelnen Komponenten das Simulationsszenario durchgeführt. Die MTS bildet dabei eine wichtige Grundlage für die Projektgruppe iSS, da in dieser Simulationskomponente das Verhalten der Schiffe abgebildet wird. Auf die Arbeit der Projektgruppe MTSS kann dabei aufgesetzt werden und mithilfe der Ergebnisse die diese Gruppe erzielt hat, ist es möglich eine ausgefeilte intelligente Schiffssteuerung zu implementieren. Ein besonderer Ansporn für die Projektgruppe iSS sollte es sein, mit den Sensordaten aus der Sensor Simulation, beschrieben in Abschnitt 3.4, umzugehen anstatt mit den Simulationsdaten, die durch HLA bereitgestellt werden. Dies würde einen realistischen Verhaltenstest abbilden, da der Logik in diesem Fall die gleichen Daten zur Verfügung stünden, wie einem Menschen.

Literaturverzeichnis

Dibbern, C., Hahn, A., & Schweigert, S. (2014). Interoperability In Co-Simulatons Of Maritime Systems. In ECMS (pp. 71–77).

Hahn, A., Noack, T (2014). eMaritime Integrated Reference Platform. In Proceedings of 2nd International Symposium of Naval Architecture and Maritime, 733–42. Istanbul, Turkey, 2014.

IEEE Standard for Modeling and Simulation (M Amp;S) High Level Architecture (HLA) - Framework and Rules. (2000). IEEE Std. 1516-2000, i–22. doi:10.1109/IEEESTD.2000.92296 IMO. (2012). Report of the 58th session of IMO Sub-Committee on Safety of Navigation, NAV 58/WP.6/Rev.1.

MTSS (2016): Maritime Transportation Systems Simulation: Abschlussdokumentation.

Reddy, H. (2013): Path Finding – Diklstra´s and A* Algorithm http://cs.indstate.edu/hgopireddy/algor.pdf 27.04.2017

Schweigert, S., Gollücke, V., Hahn, A. (2014) HAGGIS: A modelling and simulation platform for e-Maritime technology assessment. In ECMS (pp. 71–77).

SOLAS Kapitel V (2002): http://www.bsh.de/de/Schifffahrt/Sportschifffahrt/Berichtigungsservice_NfS/Schifffahrtsvorschriften/2002//Beilage-NfS33-2002.pdf 25.04.2017

Präsentation

Kollisionsgefahrenerkennung

Im Rahmen der Projektgruppe „Intelligente Schiffssimulation“ findet eine Seminarphase über die relevanten Themen statt. Diese Arbeit beschreibt unterschiedliche Arten der Kollisionsgefahrenerkennung und vergleicht diese miteinander.

Inhaltsverzeichnis:

Einleitung

In der nachfolgenden Arbeit sollen die Möglichkeit zur Kollisionsgefahrenerkennung im maritimen Bereich angeführt werden. Hierzu werden auf Basis der Vessel Trafic Services zuerst das ARPA und später das AIS vorgestellt. Nachdem die Arbeitsweisen herausgestellt worden sind, sollen die Vor- und Nachteile beider Systeme dargelegt werden, um später einen Vergleich ziehen zu können. Um einen umfassenden Überblick über die Methoden der Kollisionsgefahrenerkennung aufzeigen zu können, wird mit dem ACAS ein Blick in die Luftfahrt geworfen, um anschließend an dem COLREG Beispiel die Probleme bei der Kollisionsgefahrenerkennung im maritimen Bereich aufdecken zu können. Die Idee der PAW leitet dann über in die Denkweise des CPA, dessen Ansatz den Abschluss der Arbeit darstellen soll.

Motivation

Die Untersuchung der Jahre 1986-2006 in der Schifffahrt zeigt einen massiven Anstieg, was die Anzahl an Handelsschiffen auf den Gewässern dieser Welt betrifft, wobei besonders bei der Anzahl großer Handelsschiffe ein immenses Wachstum stattgefunden hat. Die Folge dieser Entwicklung ist, dass mehr Schiffe sich im Hafen oder auf See befinden. Zudem bedeutet der Anstieg in der Anzahl großer Handelsschiffe, dass Handelsschiffe nun mehr Platz benötigen, um ein Manöver durchführen zu können. Diese beiden Konsequenzen zeigen das zu behandelnde Problem auf, dass das Risiko von Kollisionen zwischen einzelnen Schiffen ansteigt. Deshalb ist es notwendig, dass es eine einwandfreie Gefahrenerkennung gibt, in dessen Folge das Risiko einer Kollision vermindert werden kann. (Lin06)

VTS (Vessel Traffic Services)

Um das Kollisionsrisiko zu reduzieren, hat die International Maritime Organization’s (IMO) die sogenannten vessel traffic services (VTS) definiert, die dem Schiffsoffizier bei der Kollisionsgefahrenerkennung unterstützen soll. VTS stellt somit eine Unterstützungshilfe dar, die zum Zwecke der Sicherheit des eigenen Schiffes sowie zum Schutz vor anderen Schiffen entwickelt wurde. Außerdem kommt ihr die Bedeutung zuteil, einen effektiven Ablauf des Schiffsverkehrs zu ermöglichen. Unter all die Aufgabenbereiche des VTS können sowohl einfache Tätigkeiten fallen, wie die einfache Bereitstellung gebündelter Informationen oder aber auch die Herausforderung einer komplexen Regelung eines regen Schiffsverkehrs in einem Hafen. Zusammenfassend lässt sich also festhalten, dass die Kernkompetenzen der VTS darin bestehen, die relevanten Informationen für den Schiffsoffizier bereitzustellen, möglicherweise angeforderte Navigationshilfen anzubieten und für eine reibungslose Abwicklung des Schiffsverkehrs zu sorgen. (IALA93)

Damit die theoretische Möglichkeit der Gefahrenerkennung auch in der Praxis umsetzbar ist, kommen VHF-Radiotelefone sowie Küstenradare zum Einsatz, sodass die Identifikation eines Schiffes gelingen kann und die Bewegung des Schiffes dementsprechend registriert werden kann. Sobald nun also ein Schiff eine solche VTS-Zone erreicht, teilt es der VTS-Station seinen Schiffsnamen, seine Position sowie seine angestrebte Route mit. Anschließend erkennt die VTS-Station das Schiff auf ihrem Radar und lotst es solange bis es die VTS-Zone wieder sicher verlassen hat. Sobald innerhalb der VTS-Zone die Gefahr einer möglichen Kollision besteht, kann die VTS-Station in Echtzeit eine Warnung oder einen Hinweis an die Schiffsbrücke senden. Deshalb funktioniert das Zusammenspiel zwischen der VTS-Station und dem Schiff, indem vom Schiff ein Anruf erfolgt, sobald die VTS-Zone erreicht wurde und im Nachgang mithilfe des Echos des Radars die aktuelle Position des Schiffs bestimmt werden kann. Jedoch kann es weiterhin zu Schwierigkeiten kommen, so wird das Schiff auf dem Radar zwar angezeigt, aber es treten Fehler bei der Identifizierung auf oder aber ein Schiff meldet sich nach Eintritt in die VTS-Zone nicht oder es ist nicht mit der notwendigen Ausstattung bedacht. All diese Faktoren können weiterhin Gefahrensituationen bewirken und mindern die Effektivität bei der Benutzung von VTS. (Lin06)

Automatic Radar Plotting Aids (ARPA)

Eine erweiterte Form des VTS stellt das sogenannte automatic radar plotting aids (ARPA) dar. Dieses System wurden in den 80-er Jahren entwickelt und dann auch auf den Schiffen installiert. ARPA bot forthin die Möglichkeit, den Kurs und die Geschwindigkeit eines anderen Schiffes zu erkennen und diese Informationen auf einem Radarbildschirm darzustellen. ARPA stellte also einen großen Schritt in der Weiterentwicklung des VTS dar. ARPA kann dahingehend als Stütze der VTS gesehen werden, als dass es als notwendiges Instrument der Überwachung von Schiffen angesehen wird und die erzeugten Daten des Systems auf einem Bildschirm in einer modern ausgestatteten VTS-Station anzeigen kann. ARPA funktioniert vereinfacht erklärt durch das Zusammenspiel von den gesendeten Radarquellen mit dem Computersystem: Der Radarsender entsendet also Wellen und sobald ein Objekt wahrgenommen wurde, wird ein Teil der ausgesendeten Energie reflektiert und vom Originalradar wieder empfangen. Der reflektierte Impuls sendet ein Echo. Während dieses Vorganges wird die Zeit zwischen dem Entsenden des Impulses und dem Empfangen des Echos exakt bemessen, sodass die Entfernung zu einem anderen Schiff oder eines in der Nähe befindlichen Ufers ermittelt werden kann. (Son70)

Für den Betrieb und die Installation der ARPA-Radaranlagen sah sich die IMO jedoch dazu gezwungen, ein paar Mindestanforderungen an ein ARPA-Radar aufzustellen. Diese sind, für mit ARPA ausgestattete Schiffe, bindend und werden im Folgenden einmal vorgestellt:

So sollte die Anlage dafür ausgelegt sein, innerhalb des maximalen Bereichs der Skala, Objekte mit einem Fehlerquotienten von maximal 1,5% zu erkennen oder aber ein Bereich von 70 Metern vollständig abdecken. Zwischen diesen beiden Kriterien, ist jeweils der größere Wert maßgebend für die mindestens zu erfüllende Größe.

Zudem sollte der Kurs und die Lage des Schiffes so genau angezeigt werden, dass die Fehlerspanne nicht mehr als einen Grad im positiven oder negativen Bereich von dem Originalobjekt abweicht. (IMO81)

ARPA besitzt einen großen Vorteil gegenüber der herkömmlichen Art der Gefahrenerkennung. Dort wurde nämlich, sobald ein Echo auf dem Radar erschienen ist, die relative Bewegung des Echos des fremden Objektes zum eigenen Schiff zur Hilfe genommen, um den aktuellen Kurs des anderen Schiffes, die Geschwindigkeit und den sogenannten nächstmöglichen Punkt einer Kollision zu ermitteln. Dieses hatte, neben der, für den Vorgang der Berechnung und Einschätzung, benötigten Zeit, den zusätzlichen Nachteil, dass bei gravierenden Fehlmeldungen des Radars eine Fehleinschätzung des Bewegungsablaufs des anderen Schiffes häufig die Folge war und auf Grundlage dessen falsche Entscheidungen in der Navigation des eigenen Schiffes getroffen wurden. Nachdem man diese Situationen mit einem ARPA-Radar an Deck des Schiffes untersuchte, stellte man fest, dass sich nicht nur die Auslese und Interpretation der Daten auf dem Computer einfacher gestaltete, sondern auch eine Verbesserung bei der Genauigkeit der Standortbestimmung des anderen Schiffes zu verzeichnen war. Des Weiteren konnten durch ARPA Warnungen versendet werden, sobald das Schiff eine Gegend erreichte, die als gefährlich eingestuft wird oder die Grenze zum nächstmöglichen Standort einer unausweichlichen Kollision erreicht wurde. (Lin06)

Nachdem nun einige positive Aspekte von ARPA angeführt wurden, die sich vor allem auf die Unterstützung bei der Navigation eines Schiffes beziehen, gilt es nun, sich auch mit den Gegenargumenten eines Einsatzes von ARPA auseinanderzusetzen. Auf der Contra-Seite spielen vor allem die begrenzten Möglichkeiten bei der Verwendung von Radartechnik eine wesentliche Rolle. So ist die frühzeitige Erkennung eines Objektes davon abhängig in was für einer Höhe sich unsere Antenne befindet und wie sehr sich das andere Objekt aus dem Wasser abhebt. Folglich würde ein kleines Motorboot deutlich schwerer zu erkennen sein, als ein großes Containerschiff. Dies hätte auch Auswirkung auf die Kollisionsgefahrenerkennung, sodass das Erkennen eines kleinen Motorboots erst deutlich später erfolgen würde und dementsprechend auch ein Warnhinweis erst verspätetet an den Schiffsoffizier gesendet werden kann. Neben den Einfluss von Witterungsbedingungen auf die Genauigkeit des Radars, spielt zusätzlich die horizontale Stahlkraft der entsendeten Impulse eine gewichtige Rolle bei der genauen Bestimmung eines fremden Objektes. Die größte Einschränkung bei der Benutzung eines Radars stellt wohl die Problematik bei geographischen Gegebenheiten, die es einem Radar verwehren, Situationen genau einschätzen zu können. Beispielsweise kann die Lage hinter einer Insel oder hinter einer Kurve innerhalb eines Gebirgspasses nicht von dem Radar eingefangen und abgebildet werden. Dieses Problem lässt sich auch weiter ausweiten, wenn man die Situation betrachtet, dass ein großes Schiff in naher Distanz erkannt wird, die ausgesendeten Echowellen jedoch derartig groß sind, dass unser Radar nicht mehr in der Lage ist die Echowellen von kleineren Schiffen wahrzunehmen. (Liu83)

Nichtsdestotrotz hat der Einsatz von ARPA bewiesen, dass es nun möglich ist, Verkehrswege und Fahrrinnen für ein Schiff aufzuzeigen. Dies ist auch der Grund dafür, warum ARPA heutzutage zur Standardausrüstung in der Schiffsfahrtüberwachung herangewachsen ist. Damit die Effektivität und Zuverlässigkeit gewährleistet ist, wurde ein Kriterium ausgewählt, welches es ermöglichen soll, eine objektive Bewertung eines ARPA-Radars vorzunehmen. So sollte das Radar dazu ausgelegt sein, alle beweglichen und stationären Ziele in der Umgebung zu erfassen. Hierfür wird sich an den Kriterien der VTS orientiert: Dementsprechend sollte der VTS-Bereich derart abgedeckt sein, dass er den vorgegebenen Ansprüchen entspricht. Diese besagen, dass jedes Ziel im Durchschnitt mindestens bei fünf von zehn Scans über einen Zeitraum von zwei Minuten erkannt werden sollte und somit auch dazu in der Lage ist, die Fahrtrichtung zu übermitteln. (Kob86) Zudem sollte bei einem kurzzeitigen Verlust des Echosignals, welches dazu führt, dass der aktuelle Kurs des fremden Objekts nicht mehr bestimmt werden kann, sofort ein Warnhinweis an den Offizier ausgesendet werden.

AIS (Automatic Identification System)

Eine Weiterentwicklung zum ARPA-Radar stellte das in den 90-er Jahren von Ländern mit großer maritimer Tradition entworfene AIS dar, welches 1996 von der IMO auf eben diesen Namen getauft wurde. Den Namen bekam es verliehen, da es die Aufgabe der automatischen Erfassung und Übermittlung von Schiffsdaten übernimmt. Verpflichtend war AIS an Bord aller Schiffe mit mehr als 300t Bruttoraumgehalt auf internationalen Fahrten sowie bei allen Schiffen mit mehr als 500t Bruttoraumgehalt auf nationalen Fahrten. (IMO17) Die Einführung von AIS stellte dahingehend eine Revolution dar, als dass die Schiffe mit mehr Informationen in Gefahrensituationen versorgt werden konnten. Dies führte auch zu der hohen Akzeptanz in der Schifffahrt, weil sowohl die Möglichkeiten der Kommunikation als auch die Identifikation anderer Objekte auf ein neues Level gehoben wurden. Die wechselseitige Kommunikationsmöglichkeit zwischen zwei Schiffen milderte fortan auch das Risiko einer Fehlnavigation, weshalb die IMO die Safety of Life at Sea Conventions (SOLAR) im Jahr 2000 dahingehend erweiterte, als dass auf einer gewissen Art von Schiff in einer Zeitspanne von sieben Jahren spätestens bis zum 1.7.2008 AIS verpflichtend auf dem Schiff installiert werden musste. Die verpflichtende Einführung von AIS zeigte die Leistungsfähigkeit des neuen Systems, die nach den Anschlägen vom 11. September noch eine höhere Bedeutung zuteilwurde: Fortan sollte sie nämlich auch dazu dienen, dass terroristische Anschläge auf hoher See vermieden werden sollen. Dies sollte beobachtet werden, indem man Missachtungen der vorgegebenen Navigationsanweisungen erkennt. Die Einsicht in

gewählte Routen ist dank des AIS-Systems nämlich auch dritten Parteien möglich, weshalb die Deadline für die Einführung des AIS-Systems auf den 31.12.2004 vorgezogen wurde. (IALA01)

Die Benutzung von AIS zeichnet sich durch zwei Modi aus: An erster Stelle steht die Vermeidung einer Kollision mit einem anderen Schiff, wozu der Schiffs-zu-Schiffs-Modus dient. Weiterhin gibt es die Hilfe für sogenannte Küstenstaaten, denen das System Informationen über das Schiff und deren Ladung vermittelt. Dies beschreibt das Verkehrsmanagement, welches zwischen den Helfern am Ufer und der Schiffscrew fungiert. (IMO98) Physisch ist das AIS als ein Bordcomputer an Deck des Schiffes installiert und kann über eine bestimmte Frequenz empfangen werden. Das System funktioniert mittels Übertragungen: Die Übertragungszeit innerhalb eines Funkkanals ist in Zeitschlitze mit konstanter Länge aufgeteilt. Mittels des Abstandes der Zeitschlitze kann nun die Geschwindigkeit und der Manövrierstatus eines Schiffes bestimmt werden. (Har00) Der Grund, weshalb das AIS hiermit so genau arbeiten kann, ist, dass AIS 2.000 Zeitschlitze pro Minute pro Kanal verarbeiten kann und so gut wie alle zwei Sekunden einen aktualisierten Status übermitteln kann. (IALA01) Das AIS ist in der Lage, anderen Schiffen, die mit AIS ausgestattet sind sowie Stationen am Ufer Informationen über die statischen Eigenschaften, wie die Funkrufnummer, den Namen sowie die sogenannte Maritime Mobile Service Identity zu übermitteln. Dazu kommt noch das Bereitstellen von dynamischen Informationen, welches sich aus der Schiffsposition, der Geschwindigkeit und der Kursrichtung zusammensetzt. Diese Informationen werden über das GPS, einen Kreiselkompass und der Aufzeichnung der Geschwindigkeit vermittelt. Als letzte Art der Information sind die Reiseinformationen anzuführen: Diese werden vor Reiseantritt durch den Schiffsoffizier eingegeben und setzen sich unter anderem aus dem Tiefgang und der Information über die Beförderung von Gefahrengut zusammen. Ebenso wird einem diese Information zugetragen, wenn ein anderes Schiff ebenso Gefahrengut transportiert. Auch bei der Übertragung der Informationen gilt es, je nach Informationsart zu unterscheiden: Logischerweise ist die Aktualisierung des Informationsstatus bei den dynamischen Informationen von größter Relevanz, sodass ein Hochgeschwindigkeitsschiff alle zwei Sekunden eine Aktualisierung des Kurses anzeigen muss, während ein Schiff, das vor Anker liegt in einem Intervall von drei Minuten eine neue Statusmeldung abgibt. Dementsprechend größere Intervalle gilt es bei den reisebezogenen und statischen Informationen einzuhalten, sodass ein aktualisierter Status alle sechs Minuten weitergegeben wird oder aber sobald die Information aktiv nachgefragt und aufgerufen wird. Durch die schnelle Übermittlung der dynamischen Informationen ist es anderen Schiffen in einem kurzen Zeitfenster möglich, etwaige Kursänderungen zu erkennen. Die zusätzlichen Informationen durch das AIS gewährleisten das Erkennen eines möglichen Aufeinandertreffens zweier Schiffe und ermöglichen somit eine einfachere Kollisionsgefahrenerkennung. Durch die Einführung des AIS-Systems wurde auch im Hinblick auf das VTS ein großer Schritt in die Richtung der Sicherheit und der frühzeitigen Kollisionsgefahrenerkennung genommen. Nachdem das VTS nämlich mit AIS ausgestattet war, wurden die Daten des Schiffes automatisch weiterversendet. Dies verringerte die Zeitspanne bei der Übermittlung von Informationen in möglichen Gefahrensituationen und ermöglichte die Überwindung von fehlenden Sprachkenntnissen oder Schwierigkeiten bei der Verständigung. (Sti04) Weiterhin zeigte sich in Untersuchungen, dass das Erkennen von Schiffen sich nun deutlich einfacher gestaltete. Die VTS-Station konnte so bereits die Nachrichten eines Schiffes erfassen, die sich in der maximalen Spannweite einer VHF-Kommunikation befanden. Diese streckt sich für gewöhnlich deutlich über die Spannweite eines herkömmlichen Radars hinaus. Dies vereinfachte das Erkennen von Schiffen und Schiffe, die sich in der Nähe befanden, konnten bereits frühzeitig über AIS-Nachrichten über die Position und den Kurs anderer Schiffe informiert werden. Neben den Vorteilen, die sich aus der genauen Bestimmung der Schiffsposition, dem automatischen und stetigen Aktualisierung des Kurses, der Unabhängigkeit von den Witterungsbedingungen sowie der Nicht-Beeinflussung bei der Erkennung kleiner Schiffe durch nahgelegene große Schiffe zusammensetzen, sollten auch die negativen Aspekte nicht unter den Teppich gekehrt werden. Wenn beispielsweise ein Schiff das AIS ausgeschaltet hat oder aber ein Fischerboot auf der See ist, das nicht mit AIS ausgestattet ist, dann kann die VTS-Station das Schiff nur durch das Schauen auf Sicht wahrnehmen. Deshalb sollte man sich nicht ausschließlich auf das AIS verlassen und das Schiff mit zusätzlichen Erkennungssystemen ausstatten. Folglich kann es zu gefährlichen Situationen kommen, wenn der Offizier nur auf die Anzeige des AIS vertraut. Zusätzlich kann es natürlich auch weiterhin zu Störungen oder Ausfällen des GPS oder des Kreiselkompasses kommen, sodass die an die VTS-Station oder andere Schiffe überlieferten Informationen gar nicht oder fehlerbehaftet weitergeleitet werden. (Lin06)

Vergleich zwischen ARPA und AIS

Traditionell sind die meisten Schiffe mit ARPA-Radaren ausgestattet, aber auch der Einsatz von AIS ist in den meisten maritimen Ländern mittlerweile Usus. Obwohl beide Methoden das gleiche Ziel verfolgen, konnte in den beiden vorangestellten Kapiteln ein großer Unterschied in der Methodik festgestellt werden. Die Unterschiede im Bestreben der Kollisionsgefahrenerkennung führen deshalb auch zu verschiedenen Vor- und Nachteilen der beiden Geräte. Auf der einen Seite kann mithilfe des AIS ein weiterer Radius eingefangen werden, um fremde Schiffe zu erkennen. Zudem bietet es eine Reihe von zusätzlichen Informationen, die unter anderem die Reiseroute des anderen Schiffes mit beinhalten. Andererseits bietet das ARPA-Radar die Möglichkeit, auch kleine Schiffe zu erkennen, auch wenn diese nicht mit der gleichen technischen Ausrüstung bedacht sein sollten. Aus diesem Grund wird ARPA auch weiterhin auf den meisten Schiffen eingesetzt, wobei AIS als unterstützende Einheit auch mehr und mehr Anklang findet und zusätzlich zum ARPA-Radar als Hilfe bei der Kollisionsgefahrenerkennung eingesetzt wird. Jedoch gilt es zu berücksichtigen, dass zwar die Informationen beider Systeme als nützlich und auch wichtig im Hinblick auf die Verkehrsregelung angesehen werden können, aber es Unterschiede in der Datenaufbereitung gibt. Einzig der Kurs des Schiffes, die Position und die Geschwindigkeit werden in beiden vorgestellten Systemen aufgegriffen. Durch die Tatsache begründet, dass für eine zusätzliche Installation von AIS neben einem bestehenden ARPA-Radar, ein weiterer Konsolentisch benötigt wird, der klar abgegrenzt vom Radar steht, kommt es häufig zu einer Überflutung an Informationen.

ACAS (Airborne Collision Avoidance System)

ACAS ist ein System, welches im Ursprung für den Einsatz im Luftraum verwendet wurde. Es gibt jedoch genügend Eigenschaften dieses Systems, die als Grundlage ausreichen, um auch den Einsatz von ACAS im maritimen Verkehr zu diskutieren. Wie erfolgreich ACAS in der Luftfahrt ist zeigt die Tatsache, dass Untersuchungen von EUROCONTROL ergaben, dass die Einführung des neuesten ACAS-Systems Zusammenstöße um den Faktor vier senken konnte. ACAS funktioniert über das wechselseitige Zusammenspiel eines Sekundärradars und Transpondersignalen. Hierbei werden zwei Zonen unterschieden: Zum einen die Caution Area, in der Ratschläge an das Vehikel ausgesendet werden. Tritt also ein Flugzeug in die Caution Area ein, bietet das System automatisch eine Warnfunktion an. Die erweiterte Form stellt dann die Warning Area dar, in der nicht mehr nur Verkehrsvorschläge gegeben werden, sondern ein konkreter Lösungsvorschlag entsendet wird. Hierbei wird erneut in zwei Formen von Lösungsvorschlägen unterschieden: Einmal einen Korrerkturhinweis, der den Piloten dazu auffordert, den

aktuellen Kurs zu verlassen und ein Manöver durchzuführen, welches die angestrebte Flugbahn ändert. Ebenfalls gibt es die Maßnahme des präventiven Hinweises, wobei der Pilot die Empfehlung erhält, den aktuellen Kurs beizubehalten und kein Manöver durchzuführen. Ein Pilot ist in der Folge eines aufkommenden Hinweises des Betretens einer Warning Area dazu aufgefordert, seine Flughöhe je nach Art des Hinweises zu senken oder zu erhöhen. ACAS ist ein System, welches als letzte Chance angesehen werden kann, um den Kurs zu ändern und somit eine Kollision zu vermeiden. Bei der Art des Alarms wird hierbei zwischen einem Hinweis zur Verkehrslage und dem konkreten Lösungsansatz zur Bewältigung der Situation unterschieden. Beim Hinweis zur Verkehrslage wird der Pilot bei der visuellen Erkennung von Konflikten unterstützt, während beim konkreten Lösungsansatz ein unmissverständlicher und klarer Rat gegeben wird, welches Manöver auszuführen ist, um eine Kollision zu vermeiden. (Bal15)

COLREG (International Regulations for Preventing Collisions on Sea)

Der maritime Gegenpart zu ACAS wird durch das COLREG beschrieben. Gegensätzlich zur Luftfahrt haben die oben angeführten Aspekte zu ARPA und AIS jedoch aufgezeigt, dass es im maritimen Verkehr zwar eine Kollisionsgefahrenerkennung gibt, es jedoch keine konkreten Lösungsvorschläge aufgezeigt wurden, wie sich in einem konfliktbehafteten Bereich zu verhalten ist. Das Fehlen von konkreten Regeln und Regularien in der Schifffahrt wird häufig bemängelt, gerade weil genau dies in der Luftfahrt so exakt durchstrukturiert und geregelt ist. So gibt es keinen konkreten Gefahrenkatalog, der nach dem Erkennen einer Kollisionsgefahr, hätte abgearbeitet werden können. Folglich gibt es neben den fehlenden Regularien auch kein technisches System, das nach klar vorgegebenen Sicherheitsbegrenzungen, Zeitintervallen oder innerhalb eines Bereiches Entscheidungshilfen anbietet. Deshalb hat das COLREG bei all dem Lob, das es einheimsen konnte und den dazugehörigen Weiterentwicklungen, der Seefahrt nie zu einem Durchbruch in puncto Sicherheit verholfen. (Bal15)

PAW (Potential Area of Water)

Göhler definierte für Schiffe eine Zone, die alle möglichen Varianten darstellen sollte, in die ein Manöver nach einer Kollisionsgefahrenerkennung möglich sei. (Göh839 Dieses Verfahren machte sich Inoue zu Nutze und entwickelte eine Risikoprüfung für den potentiellen Bereich eines Schiffes. Die Schwierigkeit hierbei bestand darin, dass die Dimensionen bestimmt werden mussten und eine Erweiterung der Zone nötig ist, sobald das hydrodynamische Verhalten sich aufgrund von Begebenheiten verändert. Deshalb war eine ausreichende Zuverlässigkeit bei der genauen Bestimmung sowie die echtzeitbezogene Verfügbarkeit zuverlässiger Messdaten zwingend nötig. Um die Gefahr einer Kollision zu erkennen und aufzuzeigen, wurde eine einfache Formel entwickelt, die die Wahrscheinlichkeit einer Kollision aufzeigen soll: Hierzu werden alle Steuervorgänge, die durch eine Passage führen, ohne dass es zu einer Kollision kommt, durch alle vorhandenen Steuervorgänge dividiert. (Bal15) Die PAW kann also durch die Vorhersage des Schiffsvektors und der Schiffstrecke in der Zukunft geschätzt werden. Die vorhergesagten vorgegebenen Spuren können hierbei durch folgende beiden Methoden ermittelt werden: So werden die quantitativen Bedingungen aller Komponenten, die den Kurs des Schiffes bestimmen untersucht. Hierunter sind dann der Ruderwinkel, die Hauptmotorenumdrehungen, Schlepper, Triebwerke, die Haltekraft der Anker und ähnliches zu zählen. Zu den weiteren Kräften, die den Kurs des Schiffes beeinflussen zählen dann noch die Schiffsbewegung, die Geschwindigkeit, die Beschleunigung und die sogenannte Giergeschwindigkeit. All diese Komponenten zusammengefasst haben Einfluss auf die Berechnung von Kursen, die vorhergesagt werden können. Für den Zweck der Gefahrenerkennung wird bei der PAW geschaut, in welchen Situationen Zeitabschnitte enthalten sind, wo ein Hindernis vorhanden ist. Hindernisse können hierbei neben anderen Schiffen ebenso Bojen, Wellenbrecher oder ein Kai sein. (Ino98)

Fig. 2. Illustration of unsafe situation

Abbildung 1: Illustration of unsafe situation [Ino07]

CPA (Closest Point of Approach)

Bei der Kollisionsgefahrenerkennung gilt es besonders darauf zu achten, wie nah ein anderer Verkehrsteilnehmer uns kommen wird. Der Punkt, an dem die beiden Schiffe den geringsten Abstand zueinander haben wird dann als sogenannter Closest Point of Approach bezeichnet. Zudem möchte man wissen, wann genau dieser Punkt eintritt, sodass ebenfalls die Time of Closest Point of Approach ein wichtiger Baustein bei der Kollisionsgefahrenerkennung ist. (Ven17).

Hierfür werden eine Vielzahl von Zielfahrzeugen überwacht und die Daten, die bei der Aufdeckung einer Kollisionsgefahr helfen, aufgezeigt und in einem stetigen Zeitintervall aktualisiert. Das Kollisionsvermeidungssystem soll dazu imstande sein, eine Vielzahl von Zielobjekten zu verfolgen und gleichzeitig die Datensignale der anderen Schiffe empfangen. Die empfangenen Daten splitten sich hierbei auf in die Peilung, die Geschwindigkeit und die Reichweite in Relation zu unserem Schiff. Zusätzlich zur aktuellen Position der Schiffe muss eine Vorhersage getroffen werden, wie sich der CPA in nächster Zeit entwickelt. Wichtig hierbei ist, dass der Schiffsoffizier ein Hinweis erhält, sobald der vorhergesagte CPA-Wert unter dem vorgegebenen CPA-Wert sinken sollte. Die Herausforderung ist, all die gesammelten Daten übersichtlich auf einem Display an Bord darstellen zu können. Ein Beispiel wie so etwas aussehen könnte, zeigen die beiden folgenden Abbildungen:

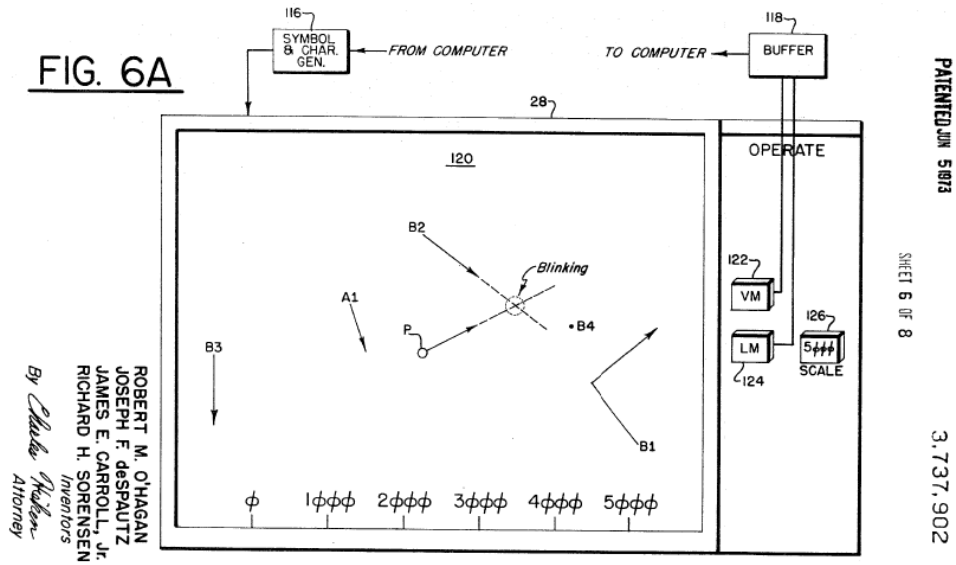


Abbildung 2: Kollisionsgefahrenerkennung (Grafische Aufbereitung) [O'H73]

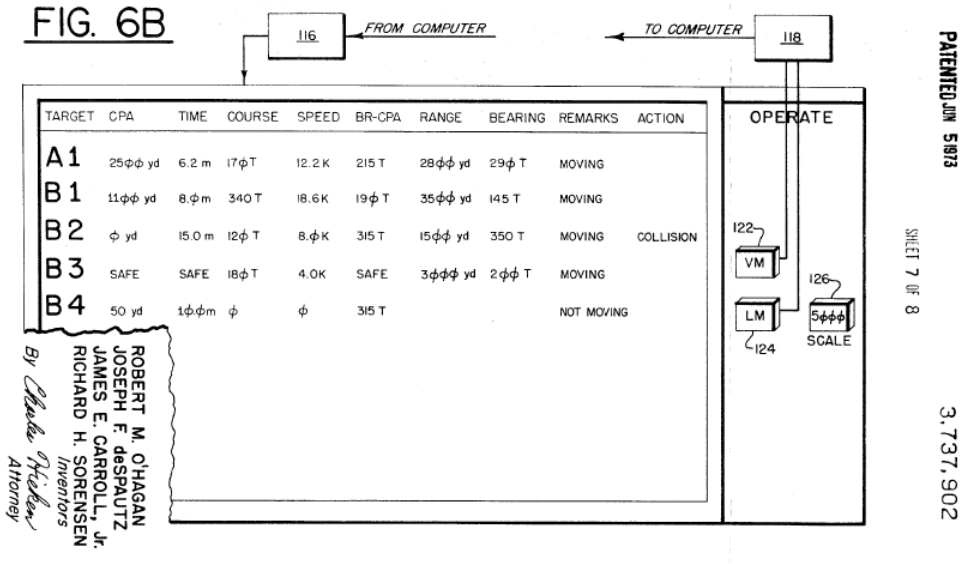


Abbildung 3: Kollisionsgefahrenerkennung (Tabellarische Aufbereitung) [O'H73]

Wie in Abbildung 2 zu erkennen ist, wird die aktuelle Situation aller Zielobjekte mithilfe von Pfeilen dargestellt. Unser Schiff wird durch das P gekennzeichnet und die Länge des Pfeiles beschreibt die aktuelle Geschwindigkeit unseres Schiffes. Die fünf anderen Fahrzeuge A1, B1, B2, B3 und B4 werden ebenfalls mit ihrem aktuellen Kurs und der dazugehörigen Geschwindigkeit, ebenfalls gekennzeichnet durch die Länge der Pfeile, angezeigt. Auf der x-Achse ist zudem eine Skala angegeben, die in 1000-Yard Intervallen den Abstand zu den Zielobjekten abbildet. Auf dem Display wird eine Kollision vorausgesagt, die durch die gepunktet fortgeführten Kurslinien unseres und des Zielschiffes gekennzeichnet wird. Diese Erweiterung der Kurslinie gewährleistet durch ein Aufblinken die Aufmerksamkeit des Offiziers, dass eine Gefahrensituation aufkommt, die es durch Abweichen vom Kurs zu unterbinden gilt. Ebenso interessant ist die Betrachtung des Kurses vom Fahrzeug B1, der ein Rechtsabbiegen des Schiffes weg von dem Gefahrenpunkt zeigt. Die Tatsache, dass B1 kein Pfeil zugeordnet ist, wird höchstwahrscheinlich an der Tatsache liegen, dass es sich hierbei um eine Boje oder einen anderen fest verankerten Gegenstand handelt. Eine Alternative zur grafischen Aufbereitung zeigt die Abbildung 3. Hierbei sind alle relevanten Daten tabellarisch angeführt. Das System zur Datenverarbeitung kann Datensignale liefern, die sich durch folgende Aspekte beschreiben lassen: Den CPA aller umliegenden Zielobjekte, die in Minuten angegebene Zeit bis zum CPA für jedes Zielobjekt, die Geschwindigkeit und den Kurs des Zielobjektes, die aktuelle Peilung des Zielobjektes sowie dessen Peilung beim CPA. Die Datenaufbereitung in Abbildung 2 erfolgt über eine Kathodenstrahlröhre. Hierbei erfolgt eine alphanummerische Aufbereitung der Identifikation und Geschwindigkeit eines jeden Schiffes. Die Laufrichtung der Objekte sind durch Pfeile dargestellt, während das Primärfahrzeug jeweils im Zentrum des Displays beheimatet ist. Alle künftigen Laufrichtungen, die zu einer Kollision führen könnten, sind durch gepunktete Linien aufbereitet, sodass der erwartete Punkt eines Aufpralls projiziert werden kann. Um die Aufmerksamkeit des Schiffsoffiziers zu erhöhen, können diese gepunkteten Linien noch aufblinken. Eine Aktualisierung der Schiffsposition, sowie deren Geschwindigkeit und deren Kurs erfolgt alle sechs Sekunden.

In Abbildung 3 zeigt das System alphanummerisch für jedes Zielobjekt die Zeit und den Ort für den CPA sowie die Geschwindigkeit und den Kurs an. Auch hier werden die aktuelle Peilung sowie die Reichweite alle sechs Sekunden aktualisiert. Auch hier können gefährliche CPAs

aufleuchten, um die Aufmerksamkeit zu erregen. Sobald der CPA eines Objektes passiert wurde, erhält der Offizier die Nachricht „SAFE“ auf seinem Bildschirm. (O’H73)

Schiffsdomänen

Aber neben den technischen Daten, die es bei der Kollisionsgefahrenerkennung zu beachten gilt, spielt auch das Empfinden der Offiziere eine Rolle, wenn sich zwei Schiffe begegnen. Studien sollen helfen, die Zonen hervorzuheben, in denen Schiffe noch willens sind, ein anderes Schiff zu passieren. Hansen untersuchte hierzu die AIS-Daten von Schiffen in einem Abschnitt der Fehmarnbelt. Der betrachtete Abschnitt lag unter einer Brücke, da hier das Zusammenspiel und das Passieren zwischen den Schiffen bestmöglich untersucht werden konnte. Besonderes Augenmerk liegt hierbei auf der Analyse der Schiffsdomäne, sodass später die Abstände zwischen den Schiffen visualisiert werden können. Dieser Prozess kann mithilfe des AIS gestaltet werden. Die präferierten Abstände eines Schiffes zu einem anderen werden aus den relativen Längenunterschiede ermittelt. Es wurde festgelegt, dass ein Schiff nicht in die Schiffsdomäne eines anderen Schiffes eintreten dürfe, während eine Überlappung der beiden Schiffsdomänen möglich sei. Dies ist für eine Wasserstraße beispielhaft in Abbildung 4 dargestellt:

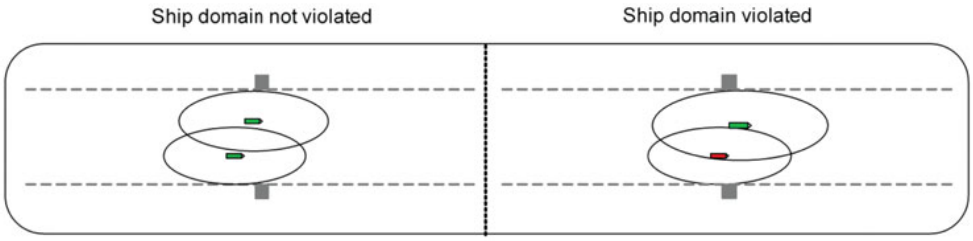


Figure 3. Examples of ships not violating/violating the ship domain for two ships sailing concurrently between two bridge pylons.

Abbildung 4: Beispiele für das Verletzen oder Nicht-Verletzen der Schiffsdomäne [Han13]

Weitere Untersuchungen zeigten auf, wie die Schiffe einem anderen Schiff ausweichen:

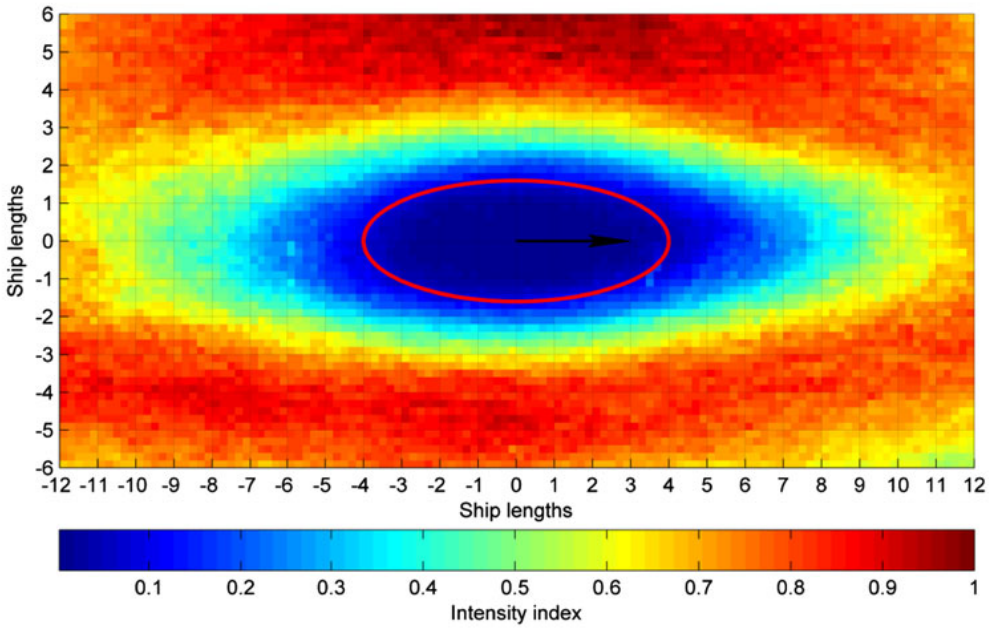


Figure 4. Intensity plot for all AIS registrations in the Fehmarnbelt area together with a plot of the ellipse predicted by the ship domain as established in this paper.

Abbildung 5: AIS-Registrierung von Schiffen in der Fehmarnbelt-Wasserstraße [Han13]

Hierbei fällt die Form einer Ellipse auf, die die Schiffe wählten, um anderen auszuweichen. Die Farben in Abbildung 5 zeigen nämlich auf wie lange sich ein Schiff in Relation zum Zielschiff wo aufgehalten hat. Die Position, die am häufigsten von fremden Schiffen eingenommen wurde, ist durch den Index 1 markiert, sodass fremde Schiffe bei Erkennung einer Kollisionsgefahr dem Zielfahrzeug in Form einer Ellipse auswichen. Auf Grundlage dieser Erkenntnisse konnte nun der minimale, präferierte, komfortable Abstand zwischen den Schiffen ausgelesen werden. Bei den Schiffslängen, die Schiffe zueinander Abstand halten, zeigte sich, dass Schiffsoffiziere bei einer Länge von 4,5 Schiffslängen zum Vordermann und 3,5 Schiffslängen zum hinterherfahrenden Schiff in eine Situation geraten, in der sie sich unwohl fühlen und deshalb mehr Abstand zum Objekt gewinnen möchten. Der Unterschied zwischen den beiden Abständen lässt sich aus der Tatsache ableiten, dass die Offiziere sich eher auf vor sich befindende Objekte konzentriert, als auf Objekte, die er bereits hinter sich hat lassen können. Für die Ermittlung der Abstände in der

Breite verhalf das, mit GPS ausgestattete, AIS bei der Ermittlung der Abstände zwischen den Schiffen. Die minimale Domäne, in der sich die Offiziere noch sicher fühlten lag bei einem Schiffsabstand von 1,6 Schiffslängen. Zudem hielten die Schiffe häufig mehr Abstand zur Steuerbord- als zur Backbordseite, da sie nach COLREG auf dieser Seite der Ausweichpflichtige sind. [Han13]

Literaturverzeichnis

[Bal15]: Baldauf, M; Mehdi, R; Deeb, H; Schröder-Hinrichs, J.-U.; Benedict, K; Krüger, C; Fischer, S; Gluch, M, *Manoeuvring areas to adapt ACAS for the maritime domain*, Scientific Journals oft he Maritime University of Szczecin, 2015, S. 39-47.

[Göh83]: *Estimation of Expectation Areas of Ships considering resistance changes, due to yaw angle and according to Model experiments*. Schiffbauforschung. 4., 1983, S. 235–246.

[Han13]: Hansen, Martin Gamborg*, Empirical Ship Domain based on AIS Data*, The Journal of Navigation 66, 2013, S. 931-940.

[Har00]: Harre, I., *"AIS Adding New Quality to VTS Systems,"* The Journal of Navigation, Vol. 53, 2000, S. 527-539.

[IALA93]: International Association of Lighthouse Authorities (IALA), *IALA Ship Traffic Services Manual*, France, 1993, S. 27-34.

[IALA01]: International Association of Lighthouse Authorities (IALA), *Guideline on AIS as a VTS Tool*, France, 2001, S. 1-16.

[IMO81]: International Maritime Organization (IMO), *"Performance Standards for Radar Equipment,"* IMO Resolution A. 477(XII), 1981, S. 1-5.

[IMO98]: International Maritime Organization (IMO), *"Recommendation on Performance Standards for a Universal Shipborne AIS"* IMO MSC, Vol. 74, No. 69, 1998.

[IMO17]: International Maritime Organization (IMO), *AIS transponders,* Online verfügbar unter: http://www.imo.org/en/OurWork/safety/navigation/pages/ais.aspx, zuletzt geprüft am 03.05.2017.

[Ino98]: Inoue, Kinzo; Sera, Wataru; Masuda, Kenji, *Evaluation of Ship Handling Safety based on the Concept of PAW*, The Journal of Japan Institute of Navigation, No.99, S. 163-171, 1998.

[Ino07]: Inoue, K; Kawase, M, *Innovative Probabilistic Prediction of Accident Occurrence*, TransNav International Journal on Marine Navigation and Safety of Sea Transportation, Vol. 1, No. 1, 2007, S. 19-22.

[Kob86]: Koburger, C.W., *Ship Traffic System*, Cornell Maritime Press, MD, 1986, S. 1-10.

[Lin06]: Lin, Bin; Huang, Chih-Hao, *Comparison between ARPA-Radar and AIS characteristics for Vessel Traffic Services*, Journal of Marine Science and Technology, Vol. 14, No. 3, 2006, S. 182-189.

[Liu83]: Liu, Z.T., *Radar Observation*, Yu-Shih Culture Publication, Taiwan, 1983, S. 149-151.

[O'H73]: O'Hagan, R; De Spautz, J; Carroll, J.; Sorensen, R., United States Patent 3,737,902, 1973.

[Son70]: Sonnenberg, G.J., *Radar and Electronic Navigation*, Newnes-Butterworth, London, 1970, S. 199-270.

[Sti04]: Stitt, I.P.A., *"AIS and Collision Avoidance - a Sense of Déjà vu,"* The Journal of Navigation, Vol. 57, 2004, S. 168-180.

[Ven17]: Venghaus, J., *Sportseeschierschein Sporthochseeschierschein Radarplotten*, Hochschule Stralsund 2017, S. 4.

Präsentation

Projektmanagement

Im Rahmen der Projektgruppe „Intelligente Schiffssimulation“ findet eine Seminarphase über die relevanten Themen statt. Diese Arbeit beschreibt das Projektmanagement inklusive verschiedener Vorgehensmodelle.

Inhaltsverzeichnis

Motivation

Seit einigen Jahren forscht die Carl von Ossietzky Universität in Zusammenarbeit mit dem OFFIS in Oldenburg an der Optimierung des Schiffsverkehrs und ihren angrenzenden Bereichen. Derzeit ist ein Schwerpunkt auf die Intelligente Schiffssimulation gelegt. Dabei handelt es sich um die Simulation von fahrerlosen Wasserfahrzeugen. Bekannt ist dieses Forschungsgebiet aus dem Straßenverkehr worin es darum geht, Kraftfahrzeug völlig autonom am Verkehr teilnehmen zu lassen. Vor der Einführung solcher Systeme sind ausgiebige Tests unabdingbar. Um einen großen Nutzen ohne Risiko und bei geringen Kosten zu erlangen, soll zunächst eine computergestützte Simulation dafür verwendet werden. Schiffssimulationen gib es bereits auf dem Markt, um zum Beispiel zukünftige Schiffskapitäne auszubilden. Diese Simulatoren stellen das, vom Schiffskapitän geführte Schiff inklusive der eintreffenden Umwelteinflüsse sehr gut dar. Andere Verkehrsteilnehmer hingegen werden entweder von dem Betreiber des Simulators gesteuert oder fahren reaktionslos eine zuvor bestimmte Strecke ab. Mit der Intelligenten Schiffssimulation sollen diese Schiffe in Zukunft selbstständig fahren und wie menschliche Verkehrsteilnehmer handeln. Dazu soll die bestehende Software um eine intelligente Steuerung ergänzt werden. Bei diesem Projekt sind mehrere Personen beteiligt, die sowohl aktiv an einer Lösung entwickeln sowie passiv die Rolle eines Stakeholders übernehmen. Um dies zu koordinieren wird ein Projektmanagement benötigt. Nicht alle Beteiligten haben bereits an einem Projekt dieser Größe mitgewirkt. Aus diesem Grund werden im Folgenden die wichtigsten Aspekte des Projektmanagements vorgestellt.

Grundlagen

In diesem Kapitel werden die Grundlagen zum Thema Projektmanagement näher erläutert. So ist der Begriff Projektmanagement von der DIN folgendermaßen beschrieben worden: „*Projektmanagement ist die Gesamtheit von Führungsaufgaben, -organisation, -techniken und -mitteln für die Initiierung, Definition, Planung, Steuerung und den Abschluss von Projekten*" (DIN 69901-5)

Das Projektmanagement stellt also eine übergeordnete Ebene einem Projektes dar und fungiert als Unterstützung für die Ausführende Ebene. Es stellt außerdem auch Techniken und Mittel für eine erfolgreiche Steuerung des Projektes zur verfügung.

Projekt

Als Projekt wird dabei ein Vorhaben bezeichnet, das im Wesentlichen durch Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist. (Vgl. DIN 69901-5)

Es gibt viele verschiedene Projekte aus unterschiedlichen Bereichen und Branchen. Diese werden sowohl von regulären Unternehmen, Vereinen oder öffentlicher Verwaltungen durchgeführt. Dabei reichen die Anwendungsmöglichkeiten des Projektmanagements von kleinen Projekten wie dem Organisieren eines Fußballspiels zwischen zwei Dorfmannschaften bis hin zu großen Projekten wie Fußballweltmeisterschaften. Dazu werden häufig etablierte Methoden des Projektmanagements eingesetzt, um mit Erfolg abzuschließen. Projektmanagement ist unverzichtbar in unserer Gesellschaft und wird zunehmend weiter erforscht und normiert. (Vgl. Meyer und Reher 2016, S. 1–2)

Projekte haben bestimmte Merkmale die sie von der täglichen Tätigkeit im regulären Geschäftsablauf unterscheiden. So sind Projekte der Definition nach zeitlich begrenzt mit einem Start und Endtermin. Ebenso sind sie einmalige Folgen von Tätigkeiten die ein bestimmtes Ziel verfolgen. Projekte mit begrenzten Ressourcen sind aber häufig bereichsübergreifend über verschiedene Abteilungen oder Organisationen durchzuführen. Ein Projekt wird allerdings nur als solches bezeichnet solange das Vorhaben eine gewisse Komplexität enthält. (Vgl. Alam und Gühl 2016, S. 2)

Ziele

Projektmanagement verfolgt verschiedene Ziele und Erwartungen beim Durchführen eines Projektes. Diese resultieren aus der Praxis und sind grob zu unterteilen in 3 Teile. Das erste Ziel befasst sich damit, mögliche Risiken zu identifizieren und diese rechtzeitig aufzuzeigen. Im Projektmanagement sind Risiken die das Projektziel gefährden besonders im Hinblick auf den Projekterfolg wichtig. Hierbei sollte die Eintrittswahrscheinlichkeit berücksichtigt werden, um den Fokus stets auf das Projektziel zu behalten. Risiken mit geringer Eintrittswahrscheinlichkeit oder jene die nicht zielgefährdend sind liegen eher im Aufgabenbereich des unternehmensweiten Risikomanagements. Ein weiteres Ziel ist das Einleiten von Maßnahmen, um den identifizierten Risiken entgegenzuwirken. Die Maßnahmen sind abhängig vom zu erwartenden Schaden und der Eintrittswahrscheinlichkeit. Eines der wichtigsten Ziele für das Projektmanagement verfolgt das Zusammenspiel zwischen allen Projektbeteiligten. In vielen Projekten gibt es interne Mitarbeiter ebenso wie externe Projektbeteiligte. Das Projektmanagement hat die Aufgabe alle Projektbeteiligten zu integrieren und Verantwortlichkeiten zu verteilen. Häufig kommt es sonst in Projekten zu Unstimmigkeiten der Zuständigkeiten. Dies führt automatisch dazu, dass es in jedem Projekt einen Projektleiter geben sollte, der sich hauptsächlich mit der Verfolgung dieser Ziele beschäftigt. Zusätzlich dient das Projektmanagement auch der Einhaltung der allgemeinen Projektziele, abgeleitet aus dem magischen Zieldreieck. (Vgl. Bär et al. 2017, S. 4–7)

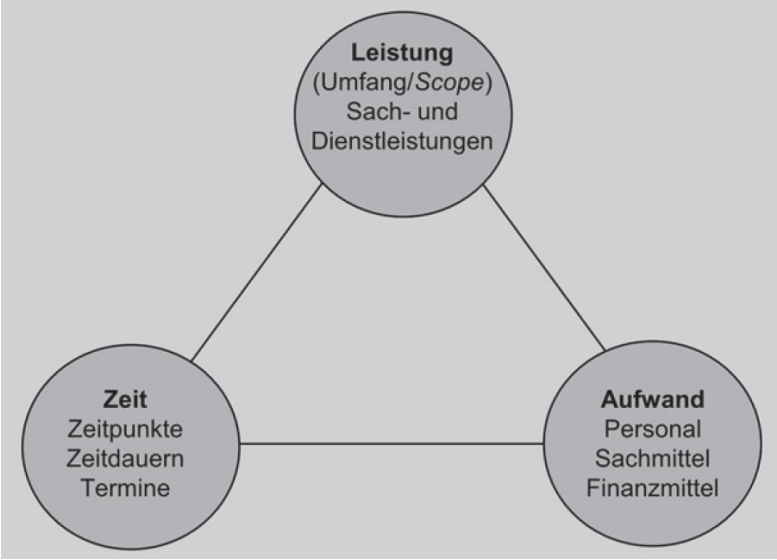


Abbildung 1: Zieldreieck (Meyer und Reher 2016, S. 12)

Das magische Zieldreieck wie in der Abbildung 1 beschreibt den Zielkonflikt in Projekten. Denn jedes Projekt soll in einem bestimmten Zeitrahmen mit bestimmten Mitteln ein genau definiertes Ziel erreichen. Somit ergeben sich drei Eckpunkte (Vgl. Meyer und Reher 2016, S. 10–13):

- Leistung / Qualität: Dieser Punkt definiert das Ergebnis das zu einem bestimmten Zeitpunkt vorhanden sein muss. Diese Ergebnisse können alle Aufgabenbereiche betreffen innerhalb des Projektes, wie zum Beispiel Produkte, Berichte, Software und so weiter. Dazu muss zuvor spezifiziert werden in welcher Form und Qualität die Leistung vom operativen Projektteam zu erbringen ist. Diese Spezifikationen werden im Anforderungsmanagement erhoben und in schriftlicher Form wie einem Lasten-/ Pflichtenheft festgehalten.
- Zeit: Jedes Projekt ist zeitlich begrenzt und hat somit einen Start sowie einen Endtermin. Somit ist eines der übergeordneten Ziele Einhaltung der Zeitvorgaben.
- Aufwand / Kosten: Projekte benötigen Mittel um durchgeführt zu werden. Dabei sind besonders Personalaufwendungen zu berücksichtigen. Die finanziellen Mittel werden im Voraus mit den Auftraggebern als Projektbudget festgelegt und sind somit als verbindliches Maximum einzuhalten.

Innerhalb eines Projektes sollte die Priorität der Projektziele aus dem Zieldreieck geklärt sein. Auch wenn sich dies im Laufe des Projektes sehr wahrscheinlich ändert, so ist es ratsam dies am Anfang grundsätzlich festzulegen. Häufig stehen diese Ziele in Konkurrenz zueinander. Um möglichen Problemen auszuweichen ist es ratsam im Vorfeld zu wissen, dass zum Beispiel die Einhaltung der Qualitätskriterien immer im Vordergrund steht und dafür der Endtermin im Zweifelsfall verschoben werden kann. Das Priorisieren der Projektziele wird vereinfacht durch analysieren von extrem Situationen wie zum Beispiel dem größten Schaden durch Terminüberschreitung. (Vgl. Meyer und Reher 2016, S. 13–14)

Rollen

In Projekten arbeiten meist mehrere verschiedene Personen zusammen. Jedes Projekt birgt viele Tätigkeiten, welche möglichst auf die beteiligten Personen aufgeteilt werden sollten. Um diese Aufgaben bestimmten Personen zuweisen zu können, werden diesen Personen Rollen zugeordnet. Die Rollen wiederum sind mit den Aufgabenbereichen verknüpft. Die in der Abbildung 2 aufgeführten Rollen stellen nur ein Beispiel für übliche Bezeichnungen dar und sind keineswegs vorgeschrieben. So könnte ProjektleiterIn auch ProjektmanagerIn sein oder EntwicklerIn auch ProgrammiererIn sein. Wichtig ist nur eine eindeutige Definition der Rollen. Solche Rollen können, wie für Rollen üblich mehreren Personen zugeordnet sein. So gibt es häufig nicht nur eine Person die entwickelt sondern oft handelt es sich um ein EntwicklerInnen-Team. In dem Fall wäre jeder Person in diesem Team die Rolle EntwicklerIn zugewiesen. Es gibt jedoch auch Projekte, die so klein sind, dass Personen mehrere, verschiedene Rollen zugewiesen werden müssen. Beispielsweise könnten die TesterInnen ihr Wissen zum Testen des Produktes auch gleich in

der Rolle des IT-Trainers zum Schulen der zukünftigen Anwender verwenden. In der Abbildung 2 ist ebenfalls zu erkennen, in welchen übergeordneten Projektphasen die verschiedenen Rollen mitwirken. Die Rolle IT-BeraterIn oder ArchitektIn ist jedoch nicht ausschließlich in den ersten Phasen tätig, sondern kann in späteren Phasen weiterhin beratend tätig sein. (Vgl. Brandt-Pook und Kollmeier 2015, S. 21–22)

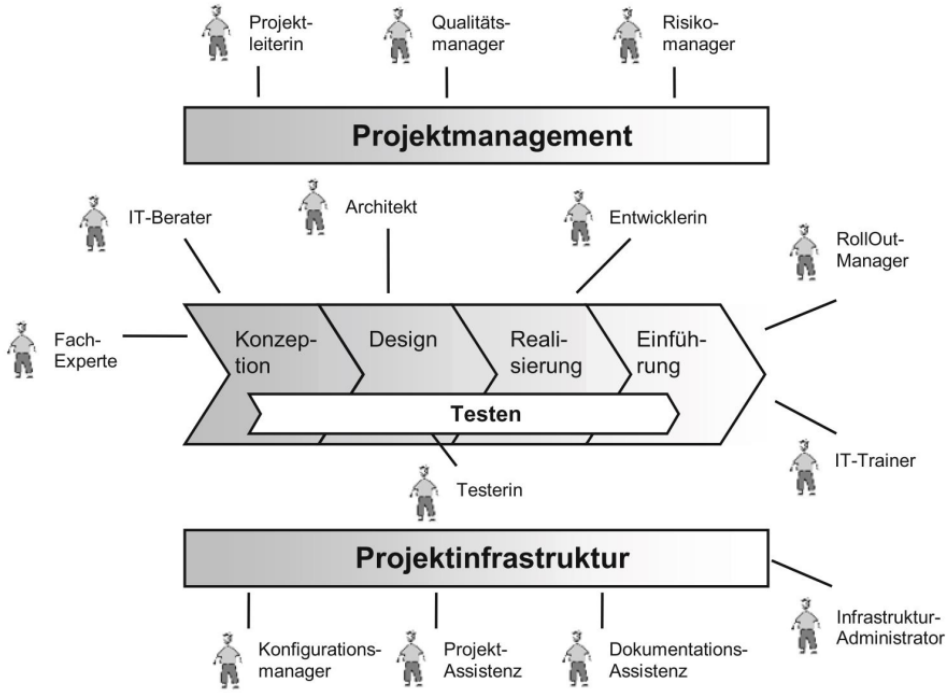


Abbildung 2: Rollen im Projektmanagement (Brandt-Pook und Kollmeier 2015, S. 21)

In der folgenden Tabelle 1 werden die in der Abbildung 2 erwähnten Rollen aufgeführt und beschrieben. Diese Beschreibung der Rollen soll ein einheitliches Verständnis für eine grobe Einordnung der Aufgabenbereiche der jeweiligen Rollen bringen. Aber auch dies kann je nach Projekt variieren und sollte nicht als Norm angesehen werden. (Vgl. Brandt-Pook und Kollmeier 2015, S. 22)

Rolle	**Tätigkeit**
ProjektleiterIn	ProjektleiterIn leiten und verantworten das Projekt als Ganzes. Und dienen als Ansprechpartner für externe Stakeholder.
QualitätsmanagerIn	QualitätsmanagerIn sorgen sich um die Qualität des Projektablaufs und des Projektergebnisses.
RisikomanagerIn	RisikomanagerIn befassen sich – insbesondere in großen Projekten – mit den (finanziellen) Risiken eines Projektes und entwickeln Maßnahmen zur Minimierung.
FachexpertIn	Diese Rolle gehört häufig zum Auftraggeber und gibt den fachlichen Input für das IT-System. Wenn es in größeren Projekten mehrere FachexpertInnen gibt, existiert vielleicht auch eine Kundenprojektleiterin – also eine Person, welche aus Sicht des Kunden für das Projekt verantwortlich ist.
IT-BeraterIn	Sie Stellen das Bindeglied zwischen Geschäft und IT dar. IT-BeraterIn stellen die Anforderungen an das IT-System systematisch und formal korrekt dar, so dass sie im Design und in der Realisierung umgesetzt werden können.
ArchitektIn	ArchitektIn entwickeln die Soft- und Hardwarearchitektur.
EntwicklerIn	EntwicklerIn setzt die zuvor geplant und designten Softwareanforderungen in Quelltext um.
RollOut-ManagerIn	RollOut-ManagerIn bringen das System beim Kunden ans Laufen.
IT-TrainerIn	Diese Rolle ist für die Schulungen der AnwenderInnen verantwortlich und führt diese durch.
TesterIn	TesterIn testen je nach Vorgehensmodell die entwickelte Software bevor sie beim Kunden ausgerollt wird.
KonfigurationsmanagerIn	KonfigurationsmanagerIn sorgen dafür, dass die einzelnen Komponenten des Systems immer gut zusammenpassen.
ProjektassistentIn	ProjektassistentIn organisieren den täglichen Ablauf im Projekt und unterstützen die Projektleitung.

Infrastruktur-AdministratorIn	Infrastruktur-AdministratorIn achten darauf, dass die IT-Infrastruktur (Server, Entwicklungsumgebungen, projektinterne Kommunikationsmedien usw.) für das Projekt zur Verfügung steht und funktioniert.
Dokumentations-Assistenz	Dokumentations-Assistenz sorgen für die Aktualität der notwendigen Dokumente im Projekt.

Tabelle 1: Rollenbeschreibung (Vgl. Brandt-Pook und Kollmeier 2015, S. 22)

Verschiedene Vorgehensmodelle bringen eigene Rollenkonzepte mit sich und erweitern oder verändern das grundlegende Rollenkonzept. Besonders das im Kapitel Agile Vorgehensmodelle beschriebene Vorgehensmodell namens Scrum benötigt eine Anpassung des Rollenkonzeptes.

Vorgehensmodelle

Im Projektmanagement haben sich mit der Zeit Standards und Erfolgsrezepte entwickelt. Diese Vorgehensmodelle sollen sicherstellen, dass Qualitätsstandards eingehalten werden und Tätigkeiten stets nachvollziehbar dokumentiert werden. Diese Vorgehensmodelle unterstützen dabei wie Projekte durchgeführt werden sollen, jedoch unterscheiden sich alle Projekte untereinander. Dies führt häufig dazu, dass die optimierten Vorgehensweisen nicht vollständig durchgeführt werden können oder angepasst werden müssen. In den unterschiedlichen Domänen haben sich jeweils bestimmte Vorgehensmodelle etabliert. In der Informatik haben sich das Wasserfallmodell, das V-Modell/XT, sowie Rational Unified Process und Scrum etabliert. Wie in der Tabelle 2 zu erkennen, unterscheiden sich diese durch verschiedene Merkmale. Sie stellen alle eine Liste mit Abschnitten die abgearbeitet werden sollen zur Verfügung. Der große Unterschied liegt dabei in der Dynamik der Vorgehensmodelle. So ist das Wasserfallmodell zusammen mit dem V-Modell durch die sequenzielle Abarbeitung eher als starr anzusehen während Rational Unified Process und Scrum durch iterative Vorgehensweisen agil sind. (Vgl. Alam und Gühl 2016, S. 5–6)

Wasserfallmodell	Hierbei handelt es sich um ein sequentielles Vorgehensmodell, das häufig in der Softwareentwicklung eingesetzt wird. Typische Phasen sind Analyse, Entwurf, Implementierung, Test und Wartung. Ausgangswerte einer Phase sind Eingangswerte für die Folgephase.
V-Modell	Dies ist eine ursprünglich aus der IT kommende Projektmanagement-Methode für Entwicklungsprojekte. Hierbei ist das Wasserfallmodell um Teststufen erweitert, wobei jede Entwicklungsphase entsprechenden Testphasen gegenübersteht.
Scrum / Kanban	Scrum versteht sich als Vorgehensrahmen im Bereich des Projektmanagements. Es ist ein einfaches Prozessmodell mit wenigen Regeln und begründet sich auf die agile Softwareentwicklung.

Tabelle 2: Vorgehensmodelle (Vgl. Alam und Gühl 2016)

Vorgehensmodelle teilen Projekte in sogenannte Projektphasen. Diese stellen Projektabschnitte dar und werden in der Regel mit abschließenden Meilensteinen markiert. Der reibungslose Ablauf von einer Projektphase in die nächste wird ebenso vom Projektmanagement geleitet wie die Auswahl des richtigen Vorgehensmodells. In der Literatur inklusive der DIN wird das Wasserfallmodell als Grundlage der Vorgehensmodelle verstanden woraus der Begriff Projektphasen geprägt wurde. In anderen Modellen gibt es diese sequenzielle Abarbeitung von Phasen nicht. Aus diesem Grund steht das Wasserfallmodell auch in der Kritik. Es ist eher als theoretisches Ideal anzusehen, da sich in der Realität Phasen überlagern können oder zu einem späteren Zeitpunkt wiederholt werden müssen. Gesamt hat sich jedoch eine vorangestellte aufwendige Planungsphase, genannt Vorstudie, als essentiell herausgestellt. Durch die Vorstudie können Machbarkeit, Erfolgschancen und Wirtschaftlichkeit sowie Ziele und Rahmenbedingungen analysiert und festgelegt werden. Dies bietet eine Hilfe bei vielen Entscheidungen im Projekt. (Vgl. Bär et al. 2017, S. 12–13)

Im Folgenden werden die in der Tabelle 2 aufgeführten Vorgehensmodelle in dem IT-Projektmanagement näher betrachtet, sodass am Ende eine Empfehlung für das geplante Projekt Intelligente Schiffssimulation entnommen werden kann.

Wasserfallmodell

Klassische Vorgehensmodelle wie das Wasserfallmodel arbeiten die Projektphasen sequenziell ab. Diese Vorgehensweise verfolgen also den Top-Down Ansatz, welcher darauf abzielt nach jeder definierten Phase ein bestimmtes Ergebnis zu erreichen. Erst nach Erreichen des Ergebnisses kann in die nächste Phase übergegangen werden. Es ist nicht möglich Phasen zu überspringen. Am Ende jeder Phase wird eine Validierung durchgeführt, welche bei auftauchenden Fehlern zur Korrektur in der jeweiligen Phase führt. Somit ist laut Definition nach der Korrektur jede Phase komplett abgeschlossen. Dadurch besteht keine Notwendigkeit bereits absolvierte Phasen zu wiederholen. Jedoch gibt es in der Realität Rückkopplungen zwischen den benachbarten Phasen. Dieses Vorgehensmodell setzt eine besonders gute Planung und Organisation des gesamten Projektes inklusive aller Phasen voraus. Das ist auch der Grund warum dieses Modell für kleinere und überschaubare Projekte verwendet wird. In größeren Projekten jedoch verändern sich Anforderungen während des Projektes, wodurch es schwierig ist diese starre Vorgehensweise einzuhalten. (Vgl. Aichele und Schönberger 2014, S. 31–32)

Wie der Abbildung 3 zu entnehmen ist, kann das Wasserfallmodell in 5 Phasen aufgeteilt werden: Analyse und Anforderungsermittlung, Architekturentwurf, Implementierung, Verifikation und Integration sowie Betrieb und Wartung. Je nach Literatur wird diesen Phasen noch eine Planungsphase vorangestellt. Diese Aufteilung entsteht durch eine einfache Auftrennung eines gesamten Softwareentwicklungsprozesses nach Schwerpunkten. Jede Phase schließt mit einem Meilenstein und einem entsprechenden Ergebnis ab. Diese Ergebnisse sind häufig in Dokumentenform vorzufinden. Jeder Phase sind bestimmte Arbeitsschritte zugeordnet die genauso abgearbeitet werden müssen. Jede Phase basiert auf dem Ergebnis der vorgelagerten Phase. Dadurch ist es nicht möglich Phasen zu parallelisieren. Doch genau in dieser einfachen Handhabung liegt der größte Vorteil dieses Vorgehensmodells. Die einfache Struktur ist leicht verständlich und lässt kaum Unstimmigkeiten zu. Diese Eigenschaften erleichtern die Organisation solcher Projekte. Das Problem beim Wasserfallmodell liegt bei ungeplanten Änderungen

während des Projektverlaufes. Durch die einmalige frühe Planungsphase und die sehr spät gelegene Testphase werden mögliche Fehler aus der Planung erst sehr spät erkannt. Auch auf Risiken kann mit diesem Modell nicht reagiert werden. (Vgl. Broy und Kuhrmann 2013, S. 88–90)

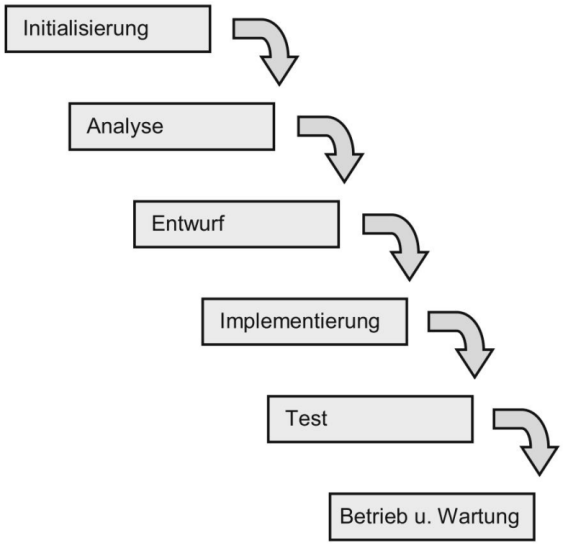


Abbildung 3: Wasserfallmodell (Brandt-Pook und Kollmeier 2015, S. 23)

V-Modell

Das V-Modell ist eine Weiterentwicklung oder auch Erweiterung des Wasserfallmodells. Es hat seinen Namen durch den v-förmigen Aufbau der Phasen wie in Abbildung 4 zu erkennen ist. Die linke Hälfte des V-Modells ist ein angepasstes Wasserfallmodell. Dem gegenüberstehen, auf der rechten Seite des V-Modells, steht eine Reihe von entsprechenden Tests. So folgt nach der Implementierung der einzelnen Softwaremodule ein Modultest. Sind dann alle Module zusammen, wird ein Integrationstest durchgeführt. Die daraus entstandenen Komponenten werden im Zusammenspiel im Systemtest getestet. Abschließend wird die komplette Software anhand der zugrunde liegenden Anforderungen mit einem Abnahmetest überprüft. Dazu müssen in jeder Phase auf der linken Seite Testfälle oder testbare Anforderungen erstellt werden. (Vgl. Brandt-Pook und Kollmeier 2015, S. 24–25)

Das V-Modell ist zwar ähnlich wie das Wasserfallmodell eher als starr zu betrachten. Jedoch führen die Tests zu deutlich besserer Risiko- und Fehlererkennung. Ebenfalls ähnlich wie beim Wasserfallmodell wird am Ende jeder Phase ein Ergebnis erwartet das als Produkt bezeichnet wird. Was diese Produkte beinhalten sollen, ist genau definiert und wird einer dafür verantwortlichen Rolle zugeordnet. Aufgrund dieser detaillierten Aufgabenbeschreibung eignet sich das V-Modell für größere Projekte. Häufig wird dieses Modell auch mit dem Bund in Verbindung gebracht, weil das V-Modell dort für alle Softwareprojekte vorgeschrieben wird. Es gibt seit 2005 auch eine neuere Variante des V-Modells genannt V-Modell XT, welches Anpassungen, auch genannt Tailoring, zulässt, um auch kleinere Projekte zu unterstützen. Tatsächlich wird die aktuellste Version des V-Modell XT von der Bundesregierung bereitgestellt. Viele andere Unternehmen nutzen das V-Modell als Grundlage für ihre eigen entwickelten Vorgehensmodelle. (Vgl. Brandt-Pook und Kollmeier 2015, S. 25)

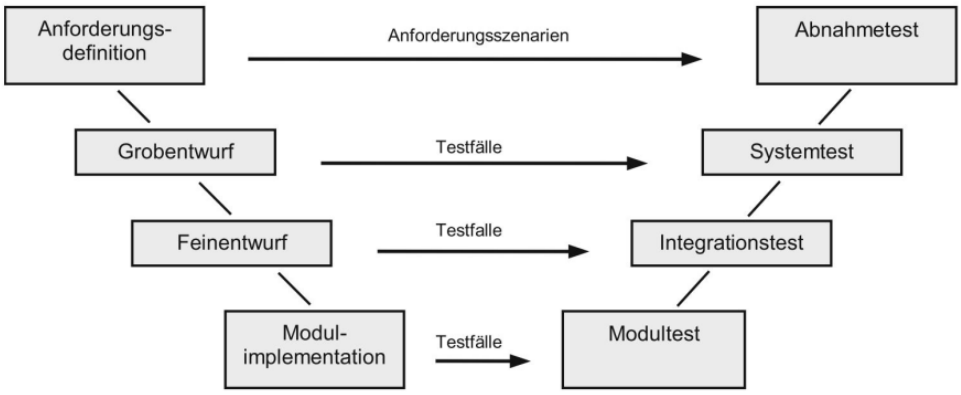


Abbildung 4: V-Modell (Brandt-Pook und Kollmeier 2015, S. 24)

Scrum

Eines der jüngsten Vorgehensmodelle ist das Scrum Modell. Scrum ist ein sehr einfaches und agiles Vorgehensmodell, welches auf einem inkrementellen und iterativen Grundprinzip aufbaut. Wie in der Abbildung 5 zu sehen gibt es ein Product Backlog, worin die Kernaufgabe des Projektes hinterlegt ist. Diese Kernaufgabe aus dem Product Backlog wird aufgeteilt in Backlog Einträge in Form von einzelnen Anforderungen oder Userstories. Diese Backlogeinträge werden dann in einer Iteration, genannt Sprint, abgearbeitet. Die Dauer eines Sprints wird zuvor auf einen bestimmten Zeitraum von 1-4 Wochen festgelegt. Jeder Tag innerhalb eines Sprints beginnt mit dem sogenannten Daily Scrum. Ein Daily Scrum ist ein tägliches kurzes Zusammentreffen aller Projektbeteiligten indem jeder kurz von seinem Fortschritt, von seinem Tagesziel und von seinen Problemen berichtet. Am Ende jedes Sprints werden die Ergebnisse im Scrum Review vorgestellt. Ein Scrum Review fordert Ergebnisse in Form einer lauffähigen Software. Der Auftraggeber kann das Ergebnis dann bewerten und aus dem Feedback zusammen mit dem Projektteam neue Anforderungen erarbeiten. Diese Anforderungen werden dann wieder in das Product Backlog gelegt und in der nächsten Iteration abgearbeitet. Anders als bei klassischen Vorgehensmodellen wie dem Wasserfallmodell oder dem V-Modell gibt es keine lange vorgelagerte Planungsphase sondern sie wird in einzelnen Iterationen aufgeteilt. Abschließend folgt noch die Sprint Retrospektive, welche sich weniger mit der Kernaufgabe, sondern mit der Arbeitsweise und deren stetiger Verbesserung auseinander setzt. (Vgl. Brandt-Pook und Kollmeier 2015, S. 30–31)

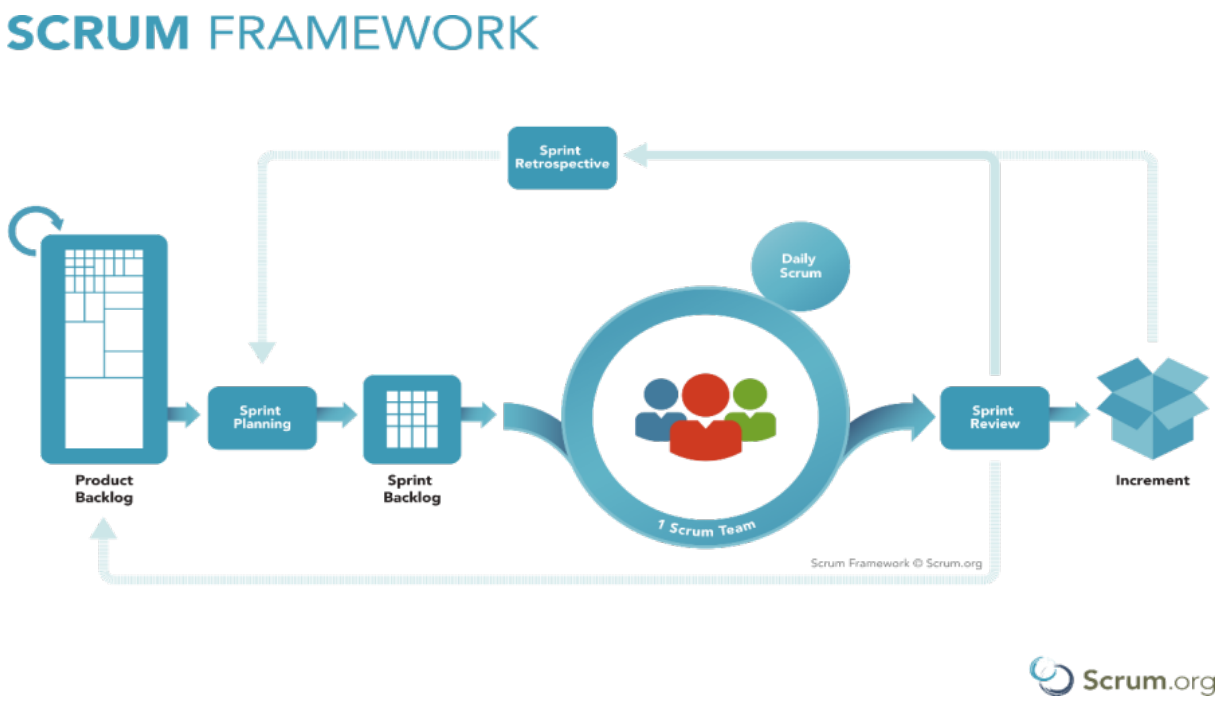


Abbildung 5: Scrum (What is Scrum? 2017)

Scrum bringt wie bereits im Kapitel Rollen beschrieben ein eigenes Rollenkonzept mit sich. Dieses besteht aus 3 verschieden Rollen mit jeweils eigenen Aufgabenbereichen (Vgl. Brandt-Pook und Kollmeier 2015, S. 31):

1. Der Product Owner verfolgt das Gesamtziel und betreut das Product Backlog. Somit koordiniert er die Abarbeitung der Backlog Einträge und priorisiert die Abarbeitung.
2. Das Entwicklungsteam ist die ausführende Instanz und arbeitet die vom Product Owner verwalteten Backlog Einträge weitestgehend selbstständig ab.
3. Der Scrum Master betrachtet das Projekt von außen und überwacht den Ablauf sowie die Regeln und betreut die Sprint Retrospektive.

Scrum gibt einen teamorientierten Führungsstil vor. Das bedeutet, es gibt keinen konkreten Projektleiter. Dadurch entsteht viel Eigenverantwortung für das Projektteam die durch Kompetenzen sowohl fachlich als auch sozial getragen werden muss. In solchen Umgebungen fungiert eine Führungsposition eher als Moderator mit dem Grundsatz Führen durch Dienen. (Vgl. Broy und Kuhrmann 2013, S. 57)

Kanban

Ein weiteres agiles Vorgehensmodell ist Kanban. Es wird als Vorgehensmodell der schlanken Softwareentwicklung bezeichnet und wurde im Jahr 2007 von David J. Anderson von der Industrie für die Softwareentwicklung vorgestellt. Das Vorgehensmodell basiert auf vier Charakteristika (Vgl. Epping 2011, S. 1–2):

1. Arbeit wird genommen, nicht gegeben.
2. Mengen werden limitiert.
3. Informationen werden veröffentlicht.
4. Arbeitsabläufe werden kontinuierlich verbessert.

„Kanban ist ein Beispiel für ein Vorgehensmodell, das aus einer *a priori* nicht strukturierten Zusammenstellung von Elementen besteht. Scrum […] ist ein Beispiel für ein Vorgehensmodell, das aus einer strukturierten Zusammenstellung von Elementen besteht." (Epping 2011, S. 18–19)

Kanban arbeitet ähnlich wie Scrum nach dem Pull-prinzip. Das Pull-Prinzip hat die Eigenschaft das Projektbeteiligte nicht ihre Aufgaben vom Projektleiter zugewiesen bekommen, sondern sich diese aus der Masse an Aufgaben herausnehmen um diese zu bearbeiten. Dafür wird eine hohe Eigenverantwortlichkeit und Selbstorganisation von allen Projektbeteiligten verlangt. Kanban unterscheidet sich in dem Punkt von Scrum, weil Kanban die Menge der herausnehmbaren Arbeitsaufträge beschränkt. Dies führt dazu, dass es in keiner Phase Überlastungen geben kann. (Vgl. Epping 2011, S. 52–53)

Das bekannteste Element von Kanban ist das Kanban-Board. Diese Technik dient zur Visualisierung der festgelegten Wertschöpfungskette eines Projektes. Auf diesem Kanban-Board lassen sich die drei Charakteristika Pull, Limitierte Mengen und Transparente Informationen von Kanban darstellen. (Vgl. Epping 2011, S. 113)

Wie in Abbildung 6 zu erkennen stellen die Spalten die Phasen des Wertschöpfungsprozesses chronologisch von links nach rechts sortiert dar. Darauf werden die so genannten Signalkarten platziert. Diese werden nach dem Pull-Prinzip von einer vorgelagerten Phase in die nächste verschoben um dort abgearbeitet zu werden.

Jede Signalkarte enthält einen Arbeitsauftrag aus dem Backlog. Jedoch beinhaltet jede Signalkarte noch weitere Informationen. So ist diese stets mit einer eindeutigen Nummer versehen um sie identifizieren zu können. Sie enthält Informationen über beteiligte Anwendungen sowie weitere Hinweise. Auch beinhaltet jede Signalkarte eine Aufwandschätzung sowie Eintritts- und Austrittszeitpunkt der Signalkarte auf dem Kanban-Board. Auch hinterlassen alle Personen die an der Signalkarte beteiligt waren einen Stempel zur Rückverfolgbarkeit. (Vgl. Epping 2011, S. 113–114)

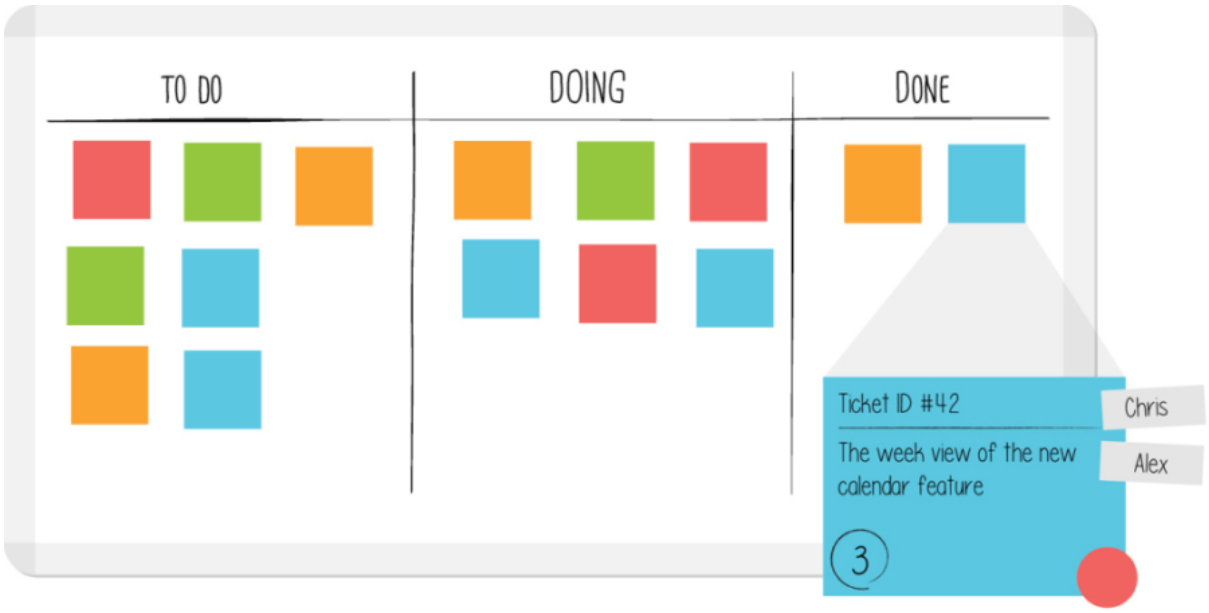


Abbildung 6: Kanban Board (What is a Kanban Board?)

Projektsteuerung

Der Projektstrukturplan ist als eines der zentralen Faktoren der Projektsteuerung zu verstehen. Denn Ziel der Projektsteuerung ist es die richtigen Ressourcen zum richtigen Zeitpunkt für die jeweiligen Arbeitspakete bereitzustellen. Genau diese Zusammenhänge werden im Projektstrukturplan abgebildet, sodass jederzeit darauf zurückgegriffen werden kann. Die drei Faktoren lassen sich zurückführen auf das Zieldreieck aus dem Kapitel Ziele. Durch Abschätzen der benötigten Ressourcen und der Zeit pro Arbeitspaket lässt sich ein Zeit- und Budgetplan ableiten. (Vgl. Ebhart et al. 2014)

Der Projektstrukturplan stellt den vollständigen Funktionsumfang in Form von hierarchisch angeordneten Projektstrukturplan-Elementen dar. Jedes dieser Elemente ist genau definiert und mindestens einer Anforderung zugeordnet. Diese Anforderungen werden aus der Anforderungsspezifikation des Projektes abgeleitet. Dies kann nach verschiedenen Kriterien geschehen. Diese Kriterien ermöglichen beliebige Kombinationen. Grundsätzlich wird dabei in vier Orientierungen unterschieden (Vgl. Valentini et al. 2013, S. 49–50):

1. Phasenorientiert: Bei der phasenorientieren Herangehensweise wird das gesamte Projekt in Unterprojekte aufgeteilt. Die Unterprojekte werden angestoßen und beendet durch Meilensteine. Ein Beispiel wäre das Wasserfallmodell. Aber auch Iterationen aus agilen Verfahren können als Phasen verstanden werden.
2. Fachgebietsorientiert: Eine weitere Möglichkeit ist die Untergliederung des Projektplans nach Fachgebieten wie zum Beispiel der Softwarearchitektur. Dadurch wird die Aufgabenverteilung deutlich erleichtert
3. Funktionsorientiert: Diese Orientierung ist gleichzusetzen mit der Verrichtungsorientierten Organisation aus dem Unternehmenskontext und gliedert das Projekt nach den Funktionsbereichen des Unternehmens.
4. Objektorientiert: Projektpläne können ebenso nach Objekten aufgegliedert werden. Typische Objekte stellen Komponenten in Softwareprojekten dar. So könnte ein Element die Erstellung der grafische Oberfläche sein.

Ein wichtiger Zusammenhang besteht zwischen den Anforderungen und den Projektstrukturplan-Elementen. Es ist unverzichtbar zu wissen

welche Anforderung in welchem Element bearbeitet wird, um Aufwandsabschätzungen und Projektfortschritt zu überwachen. Gerade bei Nicht-Funktionalen Anforderungen ist dies nicht eindeutig zu einem Element zuzuordnen. Sicherheitsanforderungen werden sicherlich nicht nur in der Entwicklung der Software umgesetzt sondern auch beim Einrichten der Datenbank oder dem Rollout. Dies lässt sich durch die Granularität der Elemente verringern, führt jedoch zu Mehraufwand in der Organisation. Solche übergreifenden Anforderungen lassen sich dennoch kaum vermeiden. Aus diesem Grund ist die Kommunikation zwischen allen Projektbeteiligten und die Transparenz der einzelnen Pakte essentiell. Die dabei entstehenden Schnittstellen stellen eine Herausforderung für den Projektleiter dar und sollten nicht außer Acht gelassen werden. (Vgl. Valentini et al. 2013, S. 50–51)

„Mit der Struktur Ihres PSP geben Sie die Struktur des Handelns und Denkens vor!" (Valentini et al. 2013, S. 51)

Ein Projektstrukturplan dient dem Projektteam als grundsätzliche Stütze für Dokumente, Besprechungen und andere Kommunikation. Sie stellt dabei jedoch keine starre Struktur dar, sondern wird abhängig vom Vorgehensmodell während des Projektes angepasst. Der Basisprojektstrukturplan stellt bestenfalls das komplette Projekt mit allen Anforderungen die aus der ersten Planungsphase beziehungsweise Anforderungsanalyse stammen dar. Diese Anforderungen können verfeinert und verändert oder gar weggelassen werden. Der Projektstrukturplan sollte stets mit den Anforderungen synchronisiert sein, da die daraus einstehenden Inkonsistenzen dazu führen, dass Anforderungen vergessen werden oder auch falsch realisiert werden. (Vgl. Valentini et al. 2013, S. 51)

Bei agilen Vorgehensmodellen konkurriert ein zuvor definierter Projektablauf, in Form eines Projektstrukturplans, mit dem Gedanken Anforderungen in jeder Iteration zu verbessern. In diesem Verfahren werden Aufwandsschätzungen und Priorisierung erst kurze Zeit vor Bearbeitung der jeweiligen Arbeitspakete vorgenommen. Jedoch kann der Projektstrukturplan als bereits im Voraus priorisiertes Backlog übernommen werden. Der Product Owner kann dieses dann im weiteren Verlauf zusammen mit seinem Team anpassen und verfeinern. Die einzige Voraussetzung für dieses Vorgehen ist das objektorientierte Gliederungsprinzip, um die Granularität und Synergie mit den Userstorys zu nutzen. (Vgl. Ebhart et al. 2014)

Empfehlungen und Fazit

Grundsätzlich ist Projektmanagement ein wichtiges Instrument um Projekte erfolgreich durchzuführen. In der Literatur sowie dem Internet sind viele Informationen zum Thema Projektmanagement zu finden. Die Herausforderung ist es den Überblick zu behalten. Ähnlich wie im Projektmanagement an sich. Einer der wichtigsten Punkte der immer wieder zu finden ist und auch durch statistische Erhebungen immer wieder deutlich wird ist, dass eine gute Planung eine der größten Erfolgsfaktoren darstellt. Fehler bei der Planung tauchen besonders bei Klassischen Vorgehensmodellen auf wo die Testphasen erst zu einem späteren Zeitpunkt stattfinden. Viel zu spät werden so Fehler erkannt und sind dann nur durch großen Ressourcenaufwand zu beheben. Um den Überblick zu behalten und um Projekte gut planen zu können wird Software benötigt.

Im OFFIS werden für übliche Softwareprojekte die Produkte des Unternehmens Atlassian namens Confluence und Jira verwendet. Confluence ist eine Kollaborationssoftware für Teams, welches den Informationsaustausch zwischen den Projektbeteiligten unterstützen soll. Darin enthalten übliche Funktionen wie sie aus anderer Wiki Software bekannt sind. Ergänzend dazu bietet das Unternehmen eine Projektmanagementsoftware namens Jira an. Jira basiert auf einem Ticketsystem. Dieses Ticketsystem bietet die Möglichkeitkeit Jira in agilen Softwareprojekten zum Beispiel mit Scrum oder Kanban einzusetzen.

Aufgrund der Vorgaben für dieses Projekt eignet sich ein agiles Vorgehensmodell wie die im Kapitel vorgestellten Methoden Scrum oder Kanban. Scrum würde mit den vorgegebenen wöchentlichen Projektgruppentreffen mit den Betreuern zusammenpassen, da Scrum Fortschritte über kurze Zeiträume sichtbar machen kann. Ebenso kann in diesem Fall die vom OFFIS genutzte Softwareumgebung verwendet werden. Dies würde den Administrations- und Einrichtungsaufwand erheblich mindern.

Literaturverzeichnis

Aichele, Christian; Schönberger, Marius (2014): IT-Projektmanagement. Effiziente Einführung in das Management von Projekten. Berlin: Springer Vieweg (essentials). Online verfügbar unter http://ebooks.ciando.com/book/index.cfm/bok_id/1869078.

Alam, Daud M.; Gühl, Uwe F. (2016): Projektmanagement für die Praxis. Ein Leitfaden und Werkzeugkasten für erfolgreiche Projekte. 1. Aufl. 2016. Berlin: Springer Vieweg (Xpert.press). Online verfügbar unter http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=1177680.

Bär, Christian; Fiege, Jens; Weiß, Markus (2017): Anwendungsbezogenes Projektmanagement. Praxis und Theorie für Projektleiter. Berlin: Springer Vieweg (Xpert.press). Online verfügbar unter http://dx.doi.org/10.1007/978-3-662-52974-4

Brandt-Pook, Hans; Kollmeier, Rainer (2015): Softwareentwicklung kompakt und verständlich. Wie Softwaresysteme entstehen. 2. Auflage. Wiesbaden: Springer Vieweg. Online verfügbar unter http://ebooks.ciando.com/book/index.cfm/bok_id/2004270 .

Broy, Manfred; Kuhrmann, Marco (2013): Projektorganisation und Management im Software Engineering. Berlin Heidelberg: Springer Berlin Heidelberg (Xpert.press). Online verfügbar unter http://dx.doi.org/10.1007/978-3-642-29290-3

Ebhart, Dieter; Lehmann, Florian; Thaden, Thorsten von (2014): Projektsteuerung beim Einsatz agiler Vorgehensmodelle (Band 34 Heft 1). Online verfügbar unter http://pi.informatik.uni-siegen.de/stt/34_1/03_Technische_Beitraege/Steuerung_final_04.pdf , zuletzt geprüft am 27.04.2017.

Epping, Thomas (2011): Kanban für die Softwareentwicklung. Berlin u.a.: Springer (Informatik im Fokus).

Meyer, Helga; Reher, Heinz-Josef (2016): Projektmanagement. Von der Definition über die Projektplanung zum erfolgreichen Abschluss. 1. Aufl. 2016. Online verfügbar unter http://dx.doi.org/10.1007/978-3-658-07569-9

DIN 69901-5, 2009: Projektmanagement - Projektmanagementsysteme - Teil 5: Begriffe.

Valentini, Uwe; Weißbach, Rüdiger; Fahney, Ralf; Gartung, Thomas; Glunde, Jörg; Herrmann, Andrea et al. (2013): Requirements Engineering und Projektmanagement. Berlin, Heidelberg: Springer (Xpert.press). Online verfügbar unter http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10640928 .

Präsentation

S-100 Hydrographic Data Model

Im Rahmen der Projektgruppe „Intelligente Schiffssimulation“ findet eine Seminarphase über die relevanten Themen statt. Diese Arbeit beschreibt den Aufbau des S-100 Hydrographic Data Model.

Inhaltsverzeichnis:

Grundlagen

Hydrografie

Als Hydrografie bezeichnet man die Erhebung von Informationen über Gewässer und deren Vermessung. Das Ziel hierbei ist es, genügend Daten über sämtliche Oberflächengewässer der Erde zu sammeln, um diese verantwortungsvoll und gefahrenfrei nutzen zu können und als Lebensraum zu schützen. Die Deutsche Hydrographische Gesellschaft e.V. definiert hierzu folgende drei Tätigkeitsbereiche:

1. Das Vermessen der Gewässer und das Erfassen der gewässerbezogenen Daten;
2. das Aufbereiten der Daten, das Verwalten der Daten in Informationssystemen

und das Analysieren des Datenfundus;

1. das Darstellen der Gewässer in Karten und Informationssystemen

und das Informieren über die Gewässer.
Die Hydrografie macht Aussagen über Gewässertiefen in Relation zu einem Bezugshorizont, die Position von Untiefen, die Position von magnetischen Anomalien, die Form und Struktur des Gewässerbodens usw. (vgl. DHyG, 2017)

International Hydrographic Organisation (IHO)

Die International Hydrographic Organisation ist eine staatenübergreifende Organisation deren Ziel eine sichere Navigation auf dem Meer und der Schutz der Meeresumwelt ist. Sie wurde 1929 gegründet und agiert seither als beratende und technische Organisation. Die IHO hat sich folgende Aufgaben zum Ziel gemacht:

- Koordination der Aktivitäten der nationalen Hydrographischen Ämter
- Vereinheitlichung der Seekarten und Dokumente
- Einführung von zuverlässigen und effizienten Methoden zur Durchführung und Nutzung hydrographischer Erhebungen
- Weiterentwicklung der Wissenschaft und Technik auf dem Gebiet der Hydrographie

(vgl. IHO, 2017)

Definition des S-100 Universal Hydrographic Data Model

Die erste Version den S-100 Universal Hydrographic Data Model, im Folgenden als S-100 bezeichnet, wurde 2010 von der IHO Transfer Standard Maintenance and Applications Development (TSMAD) Arbeitsgruppe entwickelt und von der IHO veröffentlicht. Im Folgenden wird als Grundlage die Version 2.0 verwendet welche im Juni 2015 veröffentlicht wurde. Es handelt sich dabei um ein Daten-Framework, welches unter anderem zur Modellierung elektronischer Navigationssysteme im maritimen Umfeld dient. S-100 basiert auf dem ISO 19100, einer Reihe geografischer Standards, dass bedeutet der S-100 ist mit anderen Standards, die auf dem ISO 19100 basieren, kompatibel. (vgl. IHO, 2015)

Der ISO 19100 enthält verschiedene Standards zur Definition, Beschreibung und Verwaltung von geografischen Informationen. Mittels der

Standards werden Methoden, Werkzeuge und Dienstleistungen zum Management von Informationen spezifiziert, dazu gehört die Definition, Erfassung, Analyse, Zugriff, Präsentation und Übertragung solcher Daten in elektronischer Form zwischen Benutzern, Systemen und Standorten. Die Entwicklung von geografischen Informations- und Anwendungssystemen wird, durch die Möglichkeit Profile aus den ISO 19100 Standards zu definieren, erleichtert. So können verschiedene Standards zusammengestellt werden, um ein System, für ein spezifisches Anwendungsgebiet, zu entwickeln. (vgl. ISO, 2004)

Der S-100 ermöglicht die Entwicklung von modernen Anwendungen, solche Anwendungen können 3-D Technologien, zeitlich variierende Daten und webbasierte Dienste für die Erfassung, Verarbeitung, Analyse, den Zugriff und die Präsentation von hydrographischen Daten, beinhalten. Sämtliche Entwicklungswerkzeuge des S-100 sind in 12 Bestandteile aufgeteilt, diese werden nachfolgend genauer erläutert. (vgl. IHO, 2015)

S-100 Universal Hydrographic Data Model

Nachfolgend werden die Teilbereiche des S-100 beschrieben, diese basieren auf einem oder mehreren Standards der ISO 19100 Serie. Das S-100 Model besteht aus 12 Teilbereichen. Die Teilbereiche 10 und 12 werden nachfolgend nicht genauer betrachtet, da diese zum einen lediglich verschiedene Codierungsstandards und zum anderen die Pflege des S-100 Standards beschreiben.

Part 1: Konzeptionelle Schema Sprache

Als Schema Sprache definiert die IHO die Unified Modelling Language 2 (UML 2) kombiniert mit einer Reihe von Standarddatentypen. S-100 verwendet vor allem Klassen- und Paketdiagramme, ein Klassendiagramm beschreiben die Struktur eines Datenmodells. In Tabelle 1 werden die Elemente die ein Klassendiagramm enthält beschrieben.

Element	Description
Attributes	Attributes must have a unique name within the context of a class. Attributes have a visibility (public, protected, private, derived)
Basic data types	Primitive types: Integer, Real, Boolean, CharacterString, Date, Time, DateTime Complex types: UnlimitedInteger, Matrix, S100_Multiplicity, S100_NumericRange, S100_UnitOfMeasure, S100Measure, S100_Length, S100_Angle
Enumerated types	A list of valid identifiers of mnemonic words.
Relationships and Associations	Types of relationships: Association, Generalisation, Dependency, Refinement, Aggregation, Composition Multiplicity: Exactly one, many – optional zero or more, optional zero or one, at least one, given number
Stereotypes	Interface, Type, Enumeration, MetaClass, DataType
Optional, conditional, mandatory	All attributes are per default mandatory. The default multiplicity for associations is 0..*, and for attributes is 1
Naming and name space	Use precise and understandable technical names for classes and attributes. All classes shall have unique names. Class names start with an upper case letter. Attributes start with a lower case letter.

Tabelle 1: UML Klassendiagramm (vgl. hansc/DGA, 2015)

Als Paketdiagramm bezeichnet man einen Container, welcher Subpakete, Klassen und deren Assoziationen in Gruppen zusammenstellt. Durch die Paketstruktur von UML wird eine hierarchische Anordnung aller Objekte ermöglicht.

Die Dokumentation der Semantik eines Models wird mittels einer Kontexttabelle dokumentiert, diese enthält eine Beschreibung der einzelnen Klassen, Attribute, Assoziationen usw. Tabelle 2 zeigt wie eine mögliche Kontexttabelle aussehen könnte. (vgl. hansc/DGA, 2015)

Column	Description
Role Name	Possible properties: Class, Attribute, Association, Enumeration, Literal
Name	Name of the property
Description	Semantic of the property
Multiplicity	Number of occurrences of the property in the class.

Data Type	Name of the data type of the property
Remarks	Additional information about the property e.g. constraints, conditions

Tabelle 2: Beispiel einer UML Kontexttabelle (vgl. hansc/DGA, 2015)

Part 2: Registermanagement

Das Registermanagement legt Rollen und Verantwortlichkeiten zum Management einer Registry und des dazugehörigen Registers fest. Als Registry bezeichnet man das komplette Informationssystem auf dem das Register betrieben wird. Ein Register ist eine gemanagte Liste, welche die Möglichkeit bietet, Einträge beliebig hinzufügen, ändern oder löschen zu können.
Die IHO S-100 Registry besteht zum größten Teil aus Feature concept dictionarys (FCD), ein FCD enthält Definitionen von Funktionen, Attributen, Aufzählungswerten und Informationstypen, die zur Beschreibung von Hydrografie, Geografie und Metadaten verwendet werden. Des Weiteren werden FCDs zur Erstellung von feature catalogues verwendet, ein feature catalogue beschreibt den Inhalt eines Datenprodukts. Er verwendet verschiedene Objekttypen, wie Attribute oder Funktionen, aus verschiedenen FCDs und verbindet diese miteinander. (vgl. IHO, 2015)
Die IHO stellt für das S-100 eine online Registry zur Verfügung, diese bietet freien Zugang zu verschiedensten Registern.

Part 3: General Feature Model und Regeln für ein Anwendungsschema

Ein General Feature Model (GFM) ist ein konzeptuelles Model zur Definition von Funktionen und Informationstypen, deren Charakteristika und Beziehungen. Das GFM bildet die Grundlage für feature catalogues, welche in Kapitel 2.5 genauer beschrieben werden.
Ein Anwendungsschema beschreibt den Content und die logische Struktur von geografischen Informationen.
Die Grundregeln zur Erstellung von Anwendungsschemata gestalten sich wie folgt:

- Die Datenstruktur des Anwendungsschemas muss innerhalb des Anwendungsschemas modelliert werden.
- Alle Klassen die in einem Anwendungsschema zum Datentransfer verwendet werden müssen instanziierbar sein.

(vgl. IHO, 2015)

Part 4: Metadaten

Das Metadatenprofil des S-100 ist eine Spezifikation zur Beschreibung, Validierung und zum Austausch von Metadaten über geografische Datensätze. Ziel ist die Erstellung von Metadatensätzen, die Informationen über die Identifizierung, räumliche und zeitliche Ausdehnung, Qualität, Anwendungsschema, räumliches Bezugssystem und die Verteilung von digitalen geografischen Daten liefern. Der zugrundeliegende ISO 19115:2005 Standard bietet eine große Auswahl an Elementen die in einem Metadatensatz vorhanden sein können, meist beinhaltet ein Datensatz lediglich einen kleinen Teil dieser Elemente. S-100 definiert daher spezielle Kernmetadaten, diese müssen in jedem Metadatensatz vorhanden sein. Die Kernmetadaten müssen folgende Fragen beantworten können:

1. Existiert ein Datensatz zu einem spezifischen Thema (Was)?
2. Für einen spezifischen Ort (Wo)?
3. Für eine spezifische Zeit oder Periode (Wann)?
4. Ein Kontakt, der es ermöglicht über den Datensatz zu erfahren oder den Datensatz zu kaufen (Wer)?

(vgl. IHO, 2015)

Tabelle 3 zeigt die Kernmetadaten des S-100 in verschiedene Kategorien unterteilt.

Categories of core elements	Description
Identifying a Resource	A resource is uniquely identified through a title, an abstract, a date, a type, a locator and a language.
Classifying Spatial Data	Spatial data will be classified with a topic category, bounding box, extent, vertical extent information (EPSG), spatial reference system and temporal reference.
Describing Data Quality	The quality of data will be described by lineage and spatial resolution.
Relating to Data Usage	Data usage will be described by limitation on public access, conditions applying for access and use, distribution format and online resource.
Relating to Metadata	Metadata on metadata covers file identifier, date, standard name, standard version, language, Point of contact and parent id (link between dataset and series).

Tabelle 3: Kernelemente eine S-100 Metadatensatzes (vgl. hansc/DGA, 2015)

Tabelle 4 zeigt welche Attribute die Beschreibung eines einzelnen Metadatenelements enthält.

Element number	Reference number of the element
Element name	Name of the element (e.g. Resource title)
Requirement	The element is either: Mandatory (M) - the element must be filled in under all circumstances Conditional (C) - the element must be completed if certain conditions are met e.g. Resource language must be completed if the resource contains textual information Optional (O) - the element may be filled in if desired
Occurrence	The number for how often an element occurs in the schema, which will be either one or many?
Field Type	The data allowed in a field: • Free text – enter a text in the field • Controlled vocabulary – select an option from a list • Date or date time - specify a date or a date and time in the format yyyy-mm-dd for dates and hh:mm:ss for times • Numeric - enter only numbers into this field. • URL - specify a full web address
Description	A description of the data, with links to the code list used or websites where the controlled vocabularies can be found.
Example	A textual example of the element.
XML Fragment	A fragment of an xml output from an ISO compliant schema.

Tabelle 4: Beschreibung eines Metadatenelements (vgl. hansc/DGA, 2015)

Part 5: Feature catalogue

Ein feature catalogue ist ein Model zur Klassifikation und Organisation der verschiedenen feature-Typen. Features und Attribute werden in einem feature catalogue miteinander verbunden, deren Definitionen werden aus dem feature concept dictionary übernommen. Ein feature catalogue besteht aus einer Liste von features, Eigenschaften von features und der Information wie diese miteinander verbunden sind. (vgl. IHO, 2015)

Part 6: Koordinatenreferenzsystem

Ein Koordinatenrefenzsystem (engl. Coordinate reference system, CRS) ordnet ein Objekt aus der realen Welt, welches in S-100 durch Koordinaten dargestellt wird, einer Position zu. S-100 ermöglicht es auch Koordinaten von einem CRS zu einem anderen zu transformieren. Sämtliche Elemente die notwendig sind um ein Koordinatenreferenzsystem zu erstellen werden in Tabelle 5 beschrieben, Packages sind hierbei einzelne UML Packages.
(vgl. IHO, 2015)

Package name	Description
Identified Objects	All packages depend on this package which describes the mechanism of linking elements to external definitions.
Coordinate Reference Systems	Describes the base class used for all coordinate reference systems (CRS) and all derived subclasses supported by this component. The following CRS are supported by S-100: • Geodetic CRS • Projected CRS • Vertical CRS • Image CRS
Coordinate Systems	Describes the coordinate systems by their coordinate axes, their number of dimensions, their direction, the value range and the unit of measure.
Datums	A datum is a parameter or set of parameters that defines the position of the origin, the scale, and the orientation of a coordinate system. Three types of datum are described by S-100: • Geodetic datum • Vertical datum • Image datum

Coordinate Operations	Describes the operations for coordinate conversion. The following coordinate operations are defined by S-100: • Coordinate Transformation • Coordinate Conversion • Pass Through Operation • Concatenated Coordinate Operation

Tabelle 5: Elemente eines Koordinatenreferenzsystems (vgl. hansc/DGA, 2015)

Part 7: Räumliches Schema

Das räumliche Schema enthält Informationen zur Beschreibung und Manipulation von räumlichen Charakteristiken geografischer Features. In der aktuellen Version beinhaltet das räumliche Schema lediglich Geometrie, sollte in zukünftigen Versionen auch Topologie benötigt werden, würde diese hinzukommen. Die Geometrie des räumlichen Schemas beruht auf drei Kriterien, die jeweils in verschiedene Level unterteilt sind:

1. Komplexität
    a. Einfache Geometrie
    b. Komplexe Geometrie
2. Dimensionen:
    a. 0-dimensionale Objekte
    b. 0- und 1-dimensionale Objekte
    c. 0-, 1- und 2-dimensionale Objekte
    d. 0-, 1-, 2- und 2,5-dimensionale Objekte
3. Funktionale Komplexität:
    a. Datentypen

(vgl. IHO, 2015)

Part 8: Bild- und Gitterdaten

Bild- und Gitterdaten sind wesentliche Bestandteile des S-100. Geografische Daten werden üblicherweise als Bild- oder Gitterdaten dargestellt, als Bild bezeichnet man in diesem Zusammenhang einen speziellen Typ der Gitterdatenstruktur, welcher visualisiert werden kann. S-100 unterscheidet aktuell 3 verschiedene Typen um Flächen darzustellen. (vgl. IHO, 2015)

Gitter-basierte Flächendeckung:Datenpunkte werden in einer Gitterstruktur, wie in Abbildung 1, dargestellt, dies schließt Erhebungsmodelle mit regelmäßigem Gitterabstand und solche mit unregelmäßigem und variabler Zellgröße ein. (vgl. IHO, 2015)

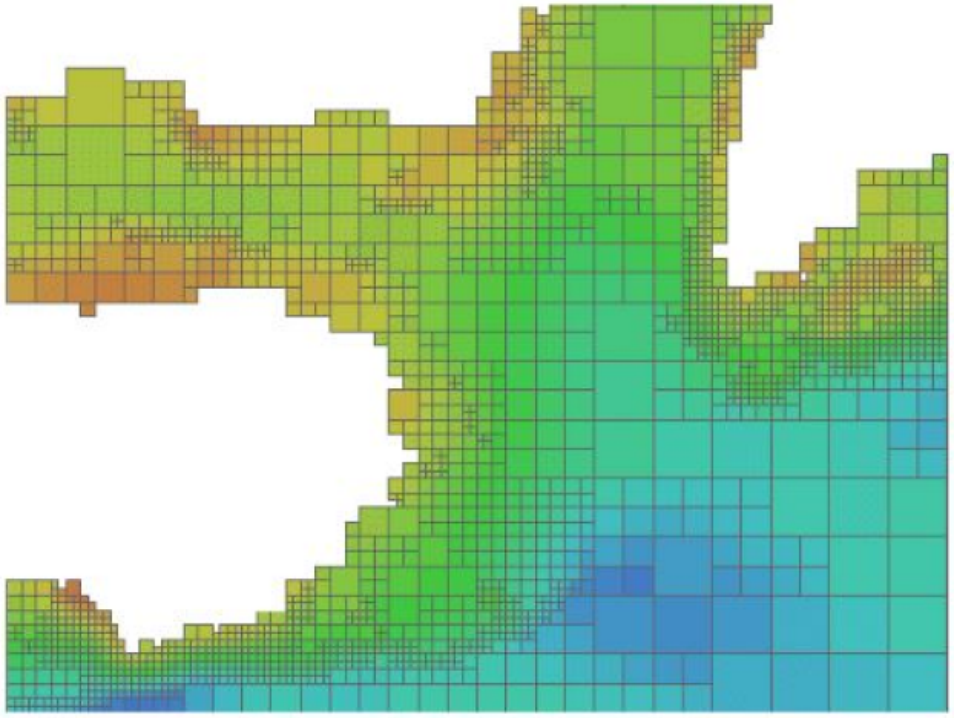


Abbildung 1: Beispiel einer Gitter Flächendeckung (vgl. IHO, 2015)

Punktmengen Flächendeckung:Eine Punktmenge sind 2, 3 oder n-dimensionale Punkte, die in einem Raum, wie in Abbildung 2 dargestellt

dargestellt werden. Diese Punkte werden mit einem Wert verknüpft. Eine Punktmengen Flächendeckung stellt diese 2-dimensionalen Punkt-Wert Paare dar. (vgl. IHO, 2015)

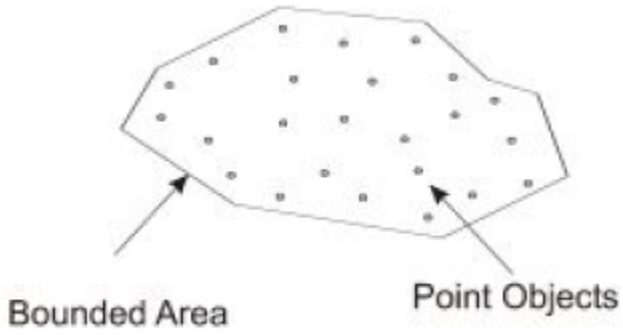


Abbildung 2: Beispiel einer Punktmenge (vgl. IHO, 2015)

TIN (Triangular Irregular Network) Flächendeckung:

Ein TIN sind mehrere zusammengesetzte Dreiecke, die eine Fläche bilden, wie in Abbildung 3 zu sehen. Attribute werden durch die Dreiecke abgebildet und die geometrischen Informationen werden aus den Eckpunkten abgeleitet. (vgl. IHO, 2015)

Abbildung 3: Beispiel einer TIN Flächendeckung (vgl. IHO, 2015)

Part 9: Darstellung

Ziel der Darstellung ist es den selben Content auf verschiedene Arten darstellen zu können. Um dies zu ermöglichen werden Feature-Daten mit Fokus auf den Inhalt modelliert, die richtige Darstellung wird durch Regeln und Funktionen, die den Inhalt mit geeigneten Symbolen und Display-Merkmalen verknüpfen, ermöglicht. (vgl. IHO, 2015)

Part 11: Produkt Spezifikation

Eine Produktspezifikation ist eine technische Beschreibung aller Features, Attribute und Beziehungen sowie des Datensatzes einer Applikation, des Weiteren enthält sie alle wichtigen Informationen über Datenidentifikation-, -inhalt und -struktur, -qualität, -erfassung, Referenzsystem, Lieferung und Metadaten. (vgl. IHO, 2015)

Zusammenfassung

Der S-100 ist ein zukunftsorientierter Standard zur Modellierung mariner Anwendungen. Die Entwicklung des Standards, basierend auch der ISO 19100 Standardserie, ist eine der wichtigsten Vorteile die der S-100 mit sich bringt. Dies bringt eine hohe Kompatibilität mit anderen Systemen mit sich, so können einzelne Komponenten in verschiedenen Systemen verwendet werden.

Literaturverzeichnis

IHO Homepage, https://www.iho.int/srv1/index.php?option=com_content&view=article&id=298&Itemid=297&lang=en, 24.04.2017
DHyG Homepage, https://www.dhyg.de/index.php/hydrographie, 24.04.2017

IHO, S-100-Universal Hydrographic Data Model, Monaco, 2015
International Organization for Standardization (ISO), 19100 Series of Geographic Information Standards, Genua 2004
Hansc/DGA, IHO S-100 Framework- The Essence, 2015

Präsentation

Software Engineering

Im Rahmen der Projektgruppe „Intelligente Schiffssimulation" findet eine Seminarphase über die relevanten Themen statt. Diese Arbeit beschreibt Vorgehensmodelle und Arbeitstechniken im Bereich der Softwareentwicklung. Zusätzlich wird ein Zusammenhang zu unserer Projektarbeit hergestellt.

Inhaltsverzeichnis

Einführung

Um die Konkurrenzfähigkeit gewährleisten und am Markt bestehen zu können, ist es für Unternehmen der meisten Branchen heutzutage unerlässlich auf computergestützte Systeme zurückzugreifen. Hinzu kommt, dass diese Systeme aufgrund der immer kürzer werdenden Produktlebenszyklen anpassbar und erweiterbar gestaltet werden müssen. Folglich müssen Unternehmungen immer mehr Software entwickeln oder entwickeln lassen was dazu geführt hat auch im Bereich Software Engineering Prozesse zu vereinfachen und effizienter zu gestalten. Eng mit der Software Entwicklung verzahnt ist das Projektmanagement. In der Praxis gibt es fließende Übergänge zwischen beiden Disziplinen, weil es in den Abläufen untereinander Abhängigkeiten gibt. Auch bei der Betrachtung von Literatur im Bereich Software Engineering wird deutlich, dass eine klare Abgrenzung schwer fällt. Grund dafür ist, dass Projektmanagement aufgrund größerer Entwicklerteams mittlerweile fester Bestandteil einer erfolgreichen Softwareentwicklung ist. Dennoch soll an dieser Stelle eine kurze Abgrenzung der beiden Gebiete erfolgen:

Während es im Projektmanagement in Bezug auf Software darum geht organisatorische Abläufe zur Entwicklung eines Softwaresystems zu definieren, bildet das Software Engineering eher die technische Seite einer solchen Unternehmung ab. Das Projektmanagement gibt dementsprechend die Struktur der Arbeit und die des Teams vor, das Software Engineering hingegen beschreibt Vorgehensmodelle zur technischen Umsetzung der Arbeit.

Vorgehensmodelle

In der Softwareentwicklung gibt es verschiedene Verfahren, nach denen in der Praxis vorgegangen wird. Zu den vier Hauptbereichen, die bei fast allen Verfahren angewandt werden, gehören die Erhebung einer Anforderungsanalyse, die Anfertigung eines Entwurfs, die konkrete Umsetzung des Entwurfs in Form der Implementation sowie eine Testphase. (vgl. Zuser et Al., 2004)

Neben den klassischen Verfahren wurden in der Vergangenheit einige zusätzliche Vorgehensmodelle entwickelt um auf die gestiegenen und sich ändernden Anforderungen reagieren zu können. Dazu gehören das

- Feature Driven Development,
- das Model Driven Development,
- das Test Driven Development,
- sowie das Behaviour Driven Development.

Diese vier Verfahren unterscheiden sich in ihrer Herangehensweise an die einzelnen Entwicklungsphasen sowie in der Ausführung ihres zeitlichen Ablaufs. In den folgenden Abschnitten sollen diese Vorgehensmodelle näher erläutert werden.

Feature Driven Development (FDD)

Wie der Name bereits impliziert geht es beim Feature Driven Development um eine funktions-getriebene Entwicklung. FDD ist in den neunziger Jahren als Reaktion auf immer länger andauernde Projekte von Jeff de Luca eingeführt worden. Hintergrund seiner Idee war die Vermeidung des Problems, dass Softwareprojekte mit einem hohen Entwicklungszeitraum aufgrund eines schnelllebigen Marktes bei Abschluss bereits nicht mehr den Bedürfnissen entspricht. Dementsprechend beschreibt FDD eine sehr kundenorientierte Entwicklungsmethode, dessen kontinuierlich hervorgebrachte Teilergebnisse stets einen messbaren Mehrwert für den Kunden darstellen. (vgl. Gyger, 2004)
Im Wesentlichen besteht das FDD aus fünf aufeinander folgenden Prozessen:

1. Entwicklung eines Gesamtmodells
2. Erstellung einer Feature-Liste

3. Planung der einzelnen Features
4. Entwurf der einzelnen Features
5. Implementierung der Features

Dabei sei zu beachten, dass die Umsetzung eines Features in seiner Zeitansetzung nicht länger als zwei Wochen dauern sollte. Andernfalls muss das zu lösende Problem weiter aufgeteilt werden. (vgl. Whitepaper it-agile GmbH)
Wie auch im Projektmanagement gibt es in der Ausführung dieser Methode unterschiedliche Rollen. Die wichtigste aller handelnden Personen stellt dabei der Chefprogrammierer (chief programmer) dar, der sich durch überdurchschnittliche hohe Fertigkeiten im Bereich der Programmierung auszeichnen sollte. Dieser ist verantwortlich für die Implementation und das Testen eines Features und damit ebenso für die Leitung eines Feature Teams. Neben dem Chefprogrammierer gibt es noch den Klassenverantwortlichen (class owner) dessen alleiniger Verantwortungsbereich in der Erstellung des Entwurfs und der Implementation seiner Klasse besteht. Der Chefprogrammierer stellt jedes Feature Team aus den Klassenverantwortlichen zusammen, die für eine Anwendung benötigt werden. Folglich entstehen immer neu zusammengesetzte Teams, dessen Mitglieder gleichzeitig mehreren Teams angehören können. (vgl. Gyger, 2004)

Model Driven Development (MDD)

Modellierungsmethoden kommen in der Softwareentwicklung schon seit langem zum Einsatz. Während klassische Ansätze auf modellbasierte Softwareentwicklung zurückgreifen, bei der die Umsetzung vom abstrakten Modell hin zum Programmcode vom Entwickler abhängig ist, unterscheidet sich die modellgetriebene Softwareentwicklung darin, dass die Modelle mit Programmcode gleichzusetzen sind, da der Code auf Grundlage des Modells automatisch generiert werden kann. (vgl. Skatulla, 2009)
Modellgetriebene Softwareentwicklung setzt nach Skatulla folgende drei Komponenten voraus:

- Domänenspezifische Sprachen (DSL) zur Formulierung von Modellen textuell, graphisch oder andere Formate
- Transformationssprachen zur Definition der Abbildung von Modell auf Code
- Generatoren/Transformatoren, die durch Anwendung der Transformationssprache auf ein Modell den Programmcode erzeugt

Vorteile einer modellgetriebenen Entwicklung ergeben sich durch einen hohen Automatisierungsgrad, einer Produktivitätssteigerung sowie einer besseren Wartbarkeit, da Code automatisch generiert wird und Modelländerungen einfacher umgesetzt werden können (Modelle und Code sind stets synchron). Des Weiteren geht man von einer verbesserten Qualität der Software aus, da Fehler im Code nur durch fehlerhafte Generatoren entstehen können und diese nur einmalig gefixt werden müssen. (vgl. Skatulla, 2009)

Test Driven Development (TDD)

Einen wichtigen Teil zur Qualitätssicherung stellt in der Softwareentwicklung das Testen dar. Dies wird in der Praxis oft vernachlässigt und von Entwicklern eher als (notwendiges) Übel angesehen, weshalb es oftmals bis zum Ende aufgeschoben wird oder sogar gänzlich entfällt. Dieses Vorgehen hat zur Folge, dass Fehler erst sehr spät oder gar nicht erkannt werden. Zusätzlich kann es zur Folge haben, dass es für Software aufgrund ihres Designs oder bestimmter Abhängigkeiten unmöglich ist, Tests dafür zu schreiben.
Um diesem Problem vorzubeugen wurde eine Methode entwickelt, die als testgetriebene Entwicklung definiert ist. Abbildung 1 zeigt unter (a) den klassischen Ansatz und unter (b) den testgetriebenen:

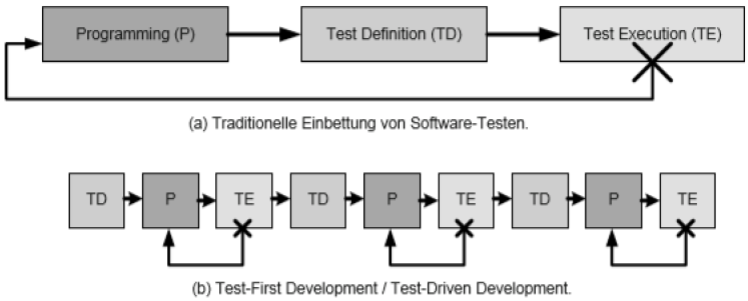


Abbildung 1: Entwicklungsablauf nach Schatten (2010)

Wie unter Punkt (a) zu sehen ist, findet nach klassischem Ansatz zunächst die Implementierung (P) statt, gefolgt von der Testentwicklung (TD) und Testdurchführung (TE). Die testgetriebene Softwareentwicklung hingegen erfordert in ihrem Ablauf eine andere Reihenfolge. Zu Beginn werden Softwaretests geschrieben, die nur exakt so viel Funktionalität abdeckt, wie das Feature bereitstellen soll. Würde der Test anschließend ausgeführt, würde er fehlschlagen. Im nächsten Schritt findet die Implementierung des Features statt, die exakt nur den Code enthält, welcher notwendig ist um den Test erfolgreich durchlaufen zu lassen. Im letzten Schritt wird dann genau das nachgeprüft. Auf diese Weise werden anfangs alle Features bearbeitet, im Hinblick auf ein ganzes System müssen jedoch Schritt für Schritt Tests und Implementierungen auf Komponentenebene und Systemebene erstellt werden.
Das Vorgehen der testgetriebenen Entwicklung bringt einige Vorteile bezüglich Qualitätssicherung mit sich. Zum einen kann der Entwickler sich bei gewissenhafter Arbeit sicher sein, dass sein Code voll funktionsfähig ist, zum anderen wird durch Refactoring zwischen den einzelnen Phasen ein sehr übersichtlicher, gut strukturierter und gepflegter Code aufgebaut. Des Weiteren besitzt der Entwickler durch seine ausführlich dokumentierten Tests bereits eine gute Dokumentation des Codes. Nachteil dieses Verfahrens ist, dass unerfahrene Entwickler sich erst einmal an diese Vorgehensweise gewöhnen müssen und eine gewisse Einarbeitungszeit brauchen um problemlos damit arbeiten zu können.

Behaviour Driven Development (BDD)

Das Behaviour Driven Development definiert eine verhaltensgesteuerte Entwicklung von Tests. Es baut auf der testgetriebenen Softwareentwicklung auf und ergänzt diese durch ein spezielles Vorgehen bei der Entwicklung der Tests. Bei der verhaltensgesteuerten Entwicklung wird der Kunde mit seinen speziellen Anforderungen stärker einbezogen, was dazu führt, dass am Ende auch jede Funktionalität umgesetzt wird, die vom Kunden verlangt wird. Dazu gibt dieser textuell vor, welche Anforderungen erwartet werden und auf diese Weise wird anschließend auch getestet. Oft folgen Entwickler dem Muster „Given – When – Then". Dabei werden in jedem Schritt Anforderungen, Anweisungen und ein zu erwartendes Ergebnis mitgegeben. Wenn beispielsweise ein Button auf einer Website getestet werden soll, der bei einem Klick auf den selbigen ein Popup mit dem Text „Hello world!" zum Vorschein bringt könnte das wie folgt aussehen:

Given there is a button with id „my_button"
When I click on „my_button"
Then I should see „Hello world!"

Tabelle 1: Behavior Driven Development

Das Vorgehen des Behaviour Driven Development bringt vor allem den Vorteil mit sich, dass das direkte Verhalten des Codes auf bestimmte Eingaben getestet werden kann und dass der Kunde mit seinem Produkt hinterher nicht unzufrieden ist. (vgl. Wynne, 2012)

Modellierungsprachen

Bestandteil jeder gewissenhaften Softwareentwicklung ist die Erstellung abstrakter Modelle im Vorfeld der Implementation, die letztendlich den umzusetzenden Entwurf des Systems wiederspiegeln. Dazu gibt es unterschiedliche Werkzeuge, von denen an dieser Stelle das wohl bekannteste und das meist verbreitete kurz vorgestellt werden soll.

Die Unified Modeling Language (UML) beschreibt eine Sprachspezifikation, die es ermöglicht sowohl Geschäftsprozesse, Anforderungen, statische und dynamische Zusammenhänge im Rahmen der Entwurfsphase visuell darzustellen. Dabei steht den Entwicklern eine Reihe von Diagrammtypen zur Verfügung mit denen Objekte, Zustände, Aktivitäten und Zusammenhänge dargestellt werden können. Zu dieser Auswahl gehören unter anderem folgende Arten:

- Anwendungsfalldiagramm
- Klassendiagramm
- Sequenzdiagramm
- Zustandsdiagramm
- Aktivitätsdiagramm

Jedes Diagramm erfüllt dabei einen anderen Anwendungszweck. Durch die Definition einer vorgegebenen Notation in UML ist es zudem möglich aus Modellen Programmcode zu generieren bzw. durch Reverse Engineering aus Programmcode Modell zu erzeugen. (vgl. Zuser, 2004)

Tools

Um den Entwicklungsprozess zu vereinfachen nutzt man in der Software-Entwicklung oft eine Reihe von Tools wie beispielsweise Modellierungswerkzeuge, integrierte Entwicklungsumgebungen und Versionsmanagementsysteme. Auf die beiden letztgenannten wird in den folgenden beiden Abschnitten nochmal genauer eingegangen.

Entwicklungsumgebungen

Mittlerweile gibt es eine Reihe an integrierten Entwicklungsumgebungen (IDEs) auf dem Markt, welche in der Regel über Funktionalitäten vom Texteditor über einen Compiler sowie Interpreter bis hin zu einem Debugger und einer Versionskontrolle verfügen. Die Verwendung solcher IDEs vereinfacht den Entwicklungsprozess und gestaltet diesen durch die Vermeidung von Medienbrüchen effizienter. Zu den gängigsten Java-IDEs gehören Eclipse, NetBeans, IntelliJ und Xcode von denen einige als Open-Source-Projekt zur Verfügung stehen und andere kostenpflichtig sind.

Versionsmanagement

Ein weiterer wichtiger Punkt im Bereich der Softwareentwicklung ist die Versionsverwaltung bzw. das Versionsmanagement. Jedes Entwicklerteam verfügt über ein Versionsmanagement um inkonsistenten Code zu vermeiden, um den Überblick über das bereits geleistete zu behalten, um eine gewisse Kontrollfunktion durch Teammitglieder zu verfolgen und um Datenverlust vorzubeugen.

Da in unserem Projekt mit Git bereits ein Tool zur Versionsverwaltung vorgegeben wird, müssen keine Alternativen diskutiert werden. Stattdessen soll an dieser Stelle eine kurze Vorstellung des Tools erfolgen. Git ist ein zu mächtiges und komplexes Werkzeug um es in kompletter Ausführung hier beschreiben zu können, daher wird es einen groben Überblick über die Funktionsweise und die wichtigsten Befehle geben.

Git kann beschrieben werden als eine Art Dateiverwaltungssystem, welches es mehreren Personen ermöglicht zeitgleich auf gemeinsame Daten zuzugreifen und diese zu bearbeiten, ohne dass an irgendeiner Stelle Datenverluste auftreten können. Dazu wird in der Regel zunächst ein Remote Repository angelegt, von dem sich jeder, der an dem Projekt mitarbeitet, eine Kopie des aktuellen Dateiverzeichnisses auf seinen eigenen lokalen Rechner klont. Diese lokale Kopie wird als „working directory" oder „Arbeitskopie" bezeichnet. Wird nun eine Datei von einem

Entwickler modifiziert, geschieht dies erstmal ausschließlich lokal auf seinem Rechner. Wenn ein Entwickler seine modifizierten Dateien mit den anderen Entwicklern teilen will, muss er diese zunächst mit dem „commit" Befehl in das git-Verzeichnis befördern und anschließend mit dem „push" Befehl in das Remote Repository laden. Von dort aus können alle anderen mit dem „pull" Befehl die aktuelle Version auf ihren lokalen Rechner herunterladen. Im Falle zweier inkonsistenter Dateien, was passieren kann wenn zwei Personen gleichzeitig eine Datei auf unterschiedliche Art und Weise verändert haben und versuchen diese zu pushen, wirft git eine Fehlermeldung und bittet um einen Abgleich der Dateien, die anschließend als einheitliche Version mit dem Befehl „merge" zusammengeführt wird. Git macht es außerdem möglich, auf jeden früheren Zeitpunkt des Verzeichnisses zurückzugreifen.

Ein weiteres wichtiges Konzept in Git ist die Benutzung von Branches. Branches stellen eine Abzweigung des Verzeichnisses dar, in dem isoliert gearbeitet werden kann. In der modernen Softwareentwicklung wird meistens für die Implementierung jedes Features ein eigener Branch angelegt und auf diesem gearbeitet. Dieses Vorgehen hat den Hintergrund, dass der Produktivbranch, der auch Masterbranch genannt wird, zu jedem Zeitpunkt funktionsfähig und korrekt bleibt, auch wenn Programmcode aktuell bearbeiteter Features noch fehlerhaft ist. Ist ein Entwickler mit der Implementation seines Features fertig und hat dessen Funktionsfähigkeit sichergestellt, darf die Datei in den Produktivbranch gemergt werden. Damit jeder Entwickler die Übersicht über das Projekt behalten kann, ist es zwingend erforderlich sich an bestimmte Code- und Namenskonventionen zu halten. Während Variablen, Methoden und Klassen im Programmcode sinnvolle, verständliche und logische Namen erhalten sollten, gilt dies auch für die Namensgebung von Feature Branches.

Ein letztes Konzept, das in diesem Zusammenhang noch erwähnt werden sollte ist die Verwendung von Merge-Requests. Dieses Vorgehen gibt vor, dass ein Entwickler bei einem gewünschten Merge eines Feature Branches in den Produktivbranch zunächst eine Anfrage dazu an bestimmte Teammitglieder stellen muss, bevor der Merge durchgeführt werden kann. Aufgabe der ausgewählten Teammitglieder ist die Kontrolle des Programmcodes. Erst wenn jedes der Mitglieder die Anfrage bestätigt hat, wird der Merge Request ausgeführt.

Zusammenfassung

Zu Beginn dieser Arbeit sind die vier Herangehensweisen FDD, MDD, TDD und BDD in der Softwareentwicklung beschrieben worden. Jedes dieser Verfahren bringt gewisse Vorteile, aber auch Nachteile mit sich. Die Vorteile des Feature Driven Developments bieten für unsere Arbeit keinen großen Mehrwert, da wir uns nicht auf stetig wechselnde Anforderungen einstellen müssen, weil davon auszugehen ist, dass sich die Seeverkehrsregeln beispielsweise nicht regelmäßig ändern werden. Die modellgetriebene Entwicklung hingegen wird voraussichtlich zu anspruchsvoll sein, da wir in unserem jungen Team wohl überwiegend wenn nicht sogar ausschließlich unerfahrene Entwickler haben. Daher scheint auch diese Methode eher ungeeignet. Das testgetriebene Vorgehen hingegen ist zumindest aus der Perspektive sinnvoll, dass unser Produkt, käme es in der Realität zum Einsatz, höchsten qualitativen Ansprüchen genügen muss und eine hundertprozentige Testabdeckung zugrunde liegen würde. Dennoch sollte berücksichtigt werden, dass auch hier eine Einführung nicht ohne Hindernisse möglich ist. Über eine verhaltensorientierte Entwicklung könnte in Folge einer erfolgreich gestarteten TDD nachgedacht werden. Andernfalls muss das Team auf die klassischen Methoden von der Analyse bis hin zum Entwurf, der Implementierung und der Testphase nachdenken.

Was die Modellierung angeht, ist UML als deskriptive Sprache vermutlich alternativlos und sollte durchaus viel genutzt werden um Qualitätsansprüche sicherzustellen und hoch zu halten. Dazu ist noch zu sagen, dass eine Dokumentation von Programmcode unabdingbar ist und in jedem Fall mit der Implementierung einhergehen sollte. Ansonsten ist es anderen Teammitgliedern unter Umständen nicht möglich, neuen Programmcode zu verstehen und Fortschritte zu verfolgen.

Im Bereich der Technologieentscheidungen und Tools ist die Entscheidungsfreiheit ziemlich eingegrenzt. Mit eclipse und netbeans stehen die zwei wohl bekanntesten Open-Source-Projekte im Bereich IDEs zur Verfügung. Dennoch sollte sich auf eine einheitliche Verwendung geeinigt werden um Komplikationen zu vermeiden. Das Funktionsprinzip von Git ist bereits erklärt worden, jedoch ist es unbedingt notwendig, dass sich jedes Gruppenmitglied nochmals intensiv damit auseinandersetzt, weil das Verständnis und die Nutzung dafür zwingend erforderlich sind. Die Einführung der bereits beschriebenen Merge Requests mag dabei sicherlich sinnvoll sein, da einige Teammitglieder im Bereich Programmierung noch nicht so bewandert sind und es generell nicht schadet, wenn mehrere Augen neuen Programmcode in Augenschein nehmen, bevor dieser in den Produktivbranch gemergt wird.

Literaturverzeichnis

Zuser, W. (2004). Software Engineering mit UML und dem Unified Process
Gyger, D. (2004). Feature Driven Development (https://files.ifi.uzh.ch/rerg/amadeus/teaching/seminars/seminar_ws0304/08_Gyger_Fdd_Ausarbeitung.pdf)
Whitepaper it-agile GmbH (https://www.it-agile.de/fileadmin/docs/Whitepaper-FDD.pdf)
Skatulla, S. (2009) Einführung in die MDSD (http://www.informatik.uni-jena.de/dbis/lehre/ss2010/skatulla/MDSD2010_1-1.pdf)
Wynne, M. (2012) The Cucumber Book – Behaviour-Driven Development

Präsentation

Spezielles maritimes Umfeld

Im Rahmen der Projektgruppe „Intelligente Schiffssimulation“ findet eine Seminarphase über die relevanten Themen statt. Diese Arbeit beschreibt das spezielle maritime Umfeld und geht dabei im speziellen auf digitale Seekarten ein. Im speziellen wird der zugrundeliegende S-57 Standard beschrieben.

Inhaltsverzeichnis

Seekarten

Allgemeines

Seekarten bieten für die Schifffahrt die Möglichkeit relevante Gebiete kenntlich zu machen. So zeigen Seekarten beispielsweise Seewege, Küsten, Untiefen, Seezeichen und Fahrrinnen von bestimmten Seegebieten auf.

Seekarten unterscheiden sich in Papierseekarten und digitalen Seekarten. Der Vorteil digitaler Seekarten ist die Aktualisierung der Seekarten durch Updates, sodass Schiffsnavigatoren diese aufwendige Aktualisierung nicht manuell vornehmen müssen. *Digitale bzw. elektrische Seekarten (ENC)* werden gemeinsam mit Sensordaten und objektbezogenen Daten über ein *elektronisches Kartendarstellungs- und Informationssystem (ECDIS)* visualisiert (Vgl. ECD 2017, SKG 2017).

In Deutschland werden Seekarten durch das Bundesamt für Seeschifffahrt und Hydrographie (BSH) herausgegeben. Alternativ gibt es jedoch auch freie Seekarten die unter den Projekten OpenSeaMap und FreieTonne entwickelt und veröffentlicht werden. Für die Navigation können diese freien Karten aufgrund fehlender Aktualität und Richtigkeit jedoch nicht verwendet werden (Vgl BSH 2017, FTO 2015, OSM 2016).

Karteninhalt

Der Inhalt einer Seekarte umfasst in der Regel eine Vielzahl von Hinweisen, die im Folgenden aufgeführt werden. Neben *Küstenlinien* werden *Tiefenlinien* in unterschiedlichen Staffelungen eingezeichnet. So zum Beispiel gibt es in der Regel sowohl Markierungen für Einzeltiefen als auch Markierungen bei 0, 2, 5, 10, 20 und 50 Metern. Hinzu kommen *Seezeichen* als genormte Symbole, die aus der sogenannten Karte 1 (INT 1) entnommen werden können und Angaben zu *Leuchtfeuern*. Bei letzterem ist vermerkt in welchen Sektoren die Leuchtfeuer in welchen Farben und mit welchen Kennungen leuchten. Weiterführend gibt es in den Seekarten Hinweise für *Unterwasserhindernisse*, wie beispielsweise Wracks, große Steine, Pipelines und Seekabel, sowie *Warnungen* für unzuverlässige Tiefenangaben, Anomalien im Erdmagnetfeld und gefährliche Strömungen. Neben *gefährlichen Wasserfahrzeugen*, wie beispielsweise Hochgeschwindigkeitswasserfahrzeuge, werden zudem auch *Verkehrstrennungsgebiete*, *Wasserstraßen* und *Sperrgebiete* in den Seekarten angegeben. In Seekarten sind sowohl das *Gradnetz* (Längen- und Breitengrade) eingezeichnet als auch *Landgebiete*, *Flachwasserbereiche* u nd *trockenfallende Gebiete*, die in unterschiedlichen Farben gekennzeichnet werden. Ein weiterer wichtiger Bestandteil von Seekarten sind Angaben zur *magnetischen Deklination*, der Missweisung zwischen geographischem und magnetischem Norden. (Vgl. NAC 2013).

Spezielles maritimes Umfeld

IHO

Die *Internationale Hydrographische Organisation (IHO)* ist eine Organisation mit 89 Mitgliedstaaten (Vgl. IHO 2017) die ihren Sitz in Monaco hat und die Vereinheitlichung von nautischen Karten und anderen Dokumenten vorantreibt. Die IHO versteht unter dem Begriff der Hydrographie eine angewandte Wissenschaft, die sich mit Küstenregionen und Hochseegebieten aus kartographischen Gesichtspunkten beschäftigt (Vgl. IHO 2015).

S-57 Standard

Das international anerkannte Format für digitale Seekarten ist in dem S-57 Standard von der IHO definiert. Der vollständige Name des S-57 Standards ist *IHO Transfer Standard for hydrographic Data* und befindet sich aktuell in der Version 3.1 vom November 2000. Dem Namen zur Folge beschreibt der Standard Methoden zum digitalen Austausch hydrographischer Daten (Vgl. IHO 2000).

Folglich umfasst der S-57 Standard weit mehr als Features zur Modellierung von Seekarten. Da das Augenmerk dieses Wiki-Eintrages auf der Modellierung von Seekarten liegt, wird im weiteren Verlauf der Fokus auf ENCs gerichtet.

Der Inhalt des Standards teilt sich in drei Teile auf, welche durch zwei Annexes und zwei Appendizes ergänzt werden. Der erste Teil umfasst eine generelle Einführung, die durch Referenzen und Definitionen die im weiteren Verlauf des Standards verwendet werden ergänzt wird. Der zweite Teil beschreibt das theoretische Datenmodell, auf dem der Standard basiert. Dieses umfasst sowohl den Zusammenhang zwischen räumlichen und beschreibenden Eigenschaften als auch Modellierungsmöglichkeiten für räumliche Objekte. In dem dritten Teil wird erklärt, wie das theoretische Datenmodell in eine Datenstruktur bzw. Daten umgemünzt wird. Während Appendix A den Objekt- und Attributkatalog umfasst, befasst sich Appendix B mit der Produktspezifikation. Dort ist definiert, welche Features zu welchem Zeitpunkt und welches Produkt anwendbar sind (Vgl. IHO 2000).

Theoretisches Datenmodell

Das theoretische Datenmodell verfolgt das Ziel, Objekte aus der Hydrographie bzw. Objekte aus der realen Welt abzubilden. In dem Modell wird hierbei zwischen *Feature-* und *Spatial-Objekten* unterschieden. Während Feature-Objekte über deskriptive Eigenschaften und keine geometrischen Informationen verfügen, müssen Spatial-Objekte Informationen über räumliche Eigenschaften verfügen und können ergänzend deskriptive Eigenschaften besitzen (Vgl. IHO 2000).

Bei Feature-Objekten wird zwischen vier Typen unterschieden. Diese vier Typen lauten *Meta-, Cartographic-, Geo-* und *Collection-Objekte*. Näher beschrieben werden diese Objekte in Appendix A (Objektkatalog). Am häufigsten werden Geo-Objekte verwendet, da diese Objekte reale Objekte darstellen, die auf der Karte sichtbar sein müssen. Beispiele für Geo-Objekte können direkt sichtbare Gegenstände wie Tonnen, Brücken oder Leuchttürme sein oder auch nicht greifbare Dinge wie Funkmeldepunkte (Vgl. Geyer 2007). Um Feature-Objekten eine geographische Position zu geben, können diese laut theoretischem Datenmodell in Relation zu beliebig vielen Spatial-Objekten gesetzt werden (Vgl. IHO 2000).

Im Vergleich zu Feature-Objekten müssen räumliche Objekte bzw. Spatial-Objekte über geographische Informationen verfügen. Mehrere Feature-Objekte können auf ein Spatial-Objekt verweisen, jedoch kann kein Spatial-Objekt existieren, ohne dass ein Feature-Objekt auf dieses verweist. Räumliche Objekte können laut S-57 Standard in drei Repräsentationsmethoden dargestellt werden. Neben der *Vektorreprä sentation* kann sowohl die *Raster-* als auch *Matrixrepräsentation* verwendet werden. Im Vergleich zu der Vektorrepräsentation gibt es für die Raster- und Matrixrepräsentation keine spezifischen Definitionen in dem Standard aus dem Jahre 2000. Die Vektorrepräsentation hingegen verfügt über vier verschiedene Modellierungsarten, die sich beispielsweise darin unterscheiden, ob Linien Knoten schneiden oder lediglich berühren müssen (Vgl. IHO 2000). In Abbildung 1 wird das theoretische Datenmodell mit den zugehörigen Objekten dargestellt.

Der Hintergrund für die Trennung zwischen Feature- und Spatial-Objekten liegt darin, dass Redundanz vermieden wird und Objekte wiederverwendet werden können. Als Beispiel hierfür können die Objekte Wasserfläche und Landmasse gesehen werden. Beide Objekte haben eine gemeinsame Begrenzungslinie. Würden die Umrisse dieser Flächen nun in Feature-Objekten abgelegt werden, so müssten die geographischen Daten zweimal kodiert werden. Dies führt zu Inkonsistenzen, da wenn ein Punkt der Begrenzung nicht genau der gleiche zu dem anderen Objekt wäre, so wäre nicht definiert wozu die entstandene Lücke gehören würde. Dieser Bereich könnte sowohl Land als auch Wasser sein (Vgl. Geyer 2007).

Das theoretische Datenmodell gibt keine Regeln zur Präsentnation oder Darstellung der Informationen vor. Es stellt lediglich den Aufbau für die sachliche Beschreibung der realen Welt zur Verfügung. Die Präsentation ist somit unabhängig und ermöglicht eine flexible Darstellung. Somit können die gleichen Daten zu unterschiedlichen Zwecken verwendet werden. Als Beispiel kann hier die graphische Darstellung von Informationen im Vergleich zur textuellen Darstellung aufgeführt werden (Vgl. IHO 2000).

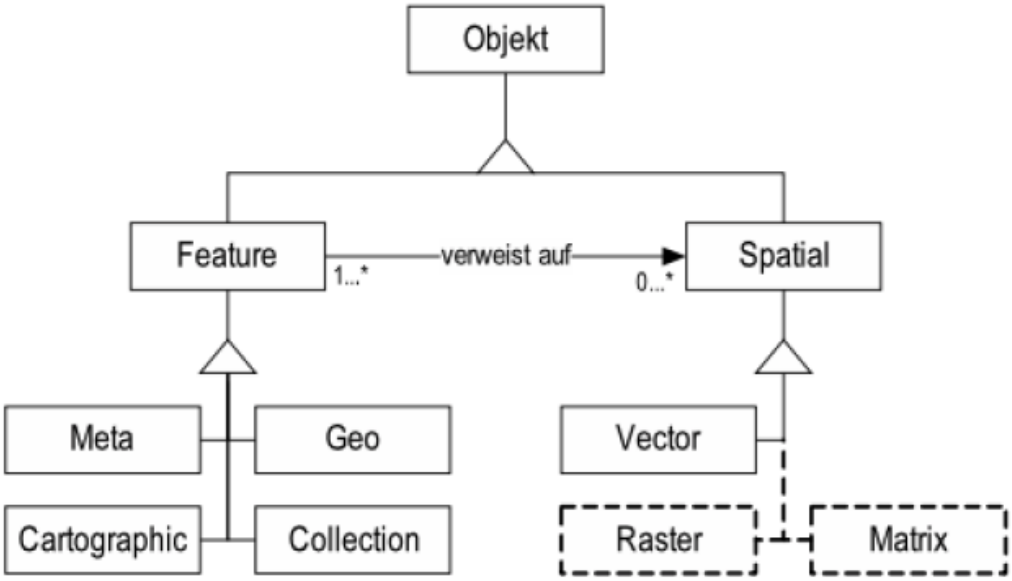


Abbildung 1: Theoretisches Datenmodell (Vgl. Geyer 2007)

Datenstruktur

Der dritte Teil des Standards spezifiziert wie das theoretische Datenmodell in die S-57 Datenstruktur übersetzt wird. Dazu wird zuvor eine genaue Struktur definiert, in der die Daten später abgelegt werden sollen (Vgl. IHO 2000).

Wie auf der linken Seite von Abbildung 2 zu sehen ist, werden Daten als *Exchange Sets* ausgetauscht. Diese Exchange Sets sind eine Menge *zusammengehöriger Dateien*. Die zugehörigen Anforderungen für den Typ, die Anzahl und die Namensgebung werden in den zugehörigen *Product Specifications* festgehalten. In Form einer *Dataset-Datei* innerhalb eines Exchange Sets werden die eigentlichen Modelldaten abgelegt. Jede dieser Dataset-Dateien hat beliebig viele *Records*, die ebenfalls wieder aus 1 bis n *Fields* bestehen. Diese Fields dienen der Gruppierung von Daten. Die hierarchisch unter den Fields liegenden *Subfields* halten jedoch die eigentlichen Werte. Diese Werte müssen laut des Standards in *atomarer Form*, das heißt Daten die nicht weiter teilbar sind, abgelegt werden (Vgl. IHO 2000, Geyer 2007).

Die rechte Seite von Abbildung 2 stellt die gesamte Datenstruktur in Form eines Beispiels dar. Weiterführend gibt es einzelne Besonderheiten, die beachtet werden müssen. So zum Beispiel unterscheiden sich Records in fünf Typen. Während *Data Set Descriptive, Catalogue* und *Data Dictionary* Meta-Informationen über das Dataset und die Daten halten, repräsentieren Feature- und Spatial-Records die entsprechenden Objekte. Die einzelnen Typen verfügen über unterschiedliche Strukturen mit jeweils zugehörigen Fields und Subfields. Ergänzend dazu gibt es optionale (Sub-)Fields, die nicht verwendet werden müssen und bei leeren Daten nicht mit abgespeichert werden (Vgl. Geyer 2007).

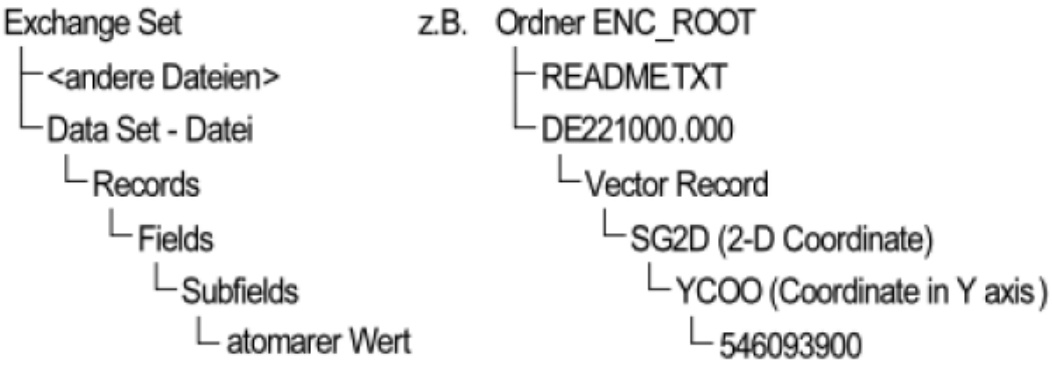


Abbildung 2: Datenstruktur (Vgl. Geyer 2007)

In dem dritten Teil des Standards sind zudem weitere Regeln festgelegt, wie beispielsweise Flächen mit Aussparungen modelliert und beschrieben werden oder in welche Richtung Kanten definiert werden müssen. Weiterführend macht der Standard zusätzliche Vorgaben zu möglichen Koordinatensystemen, Maßeinheiten und geodätischen Daten. Meta-Daten dieser Art können sowohl im Data Descriptive Record, in Meta-Objekten oder in den Attributen des jeweiligen Objekts angegeben werden. Geographische Positionen für Spatial-Objekte werden

zweidimensional modelliert. Falls eine dritte Dimension, wie die Höhe oder Tiefe, zur Verfügung steht, kann diese auf Attributebene ergänzt werden. Es gibt zwei Formen in denen Daten kodiert werden können. Das ist zum einen *ASCII* und zum anderen in *Binärform* (Vgl. Geyer 2007).

Im Annex A wird das Austauschformat beschrieben. Dieses binäre Austauschformat ist nach dem ISO/IEC 8211-Standard aufgebaut, sodass es ermöglicht wird Records, Fields und Subfields baumartig zu gliedern. Die logische Struktur der Daten wird in den Data Descriptive Records festgehalten. Damit einzelne Daten getrennt werden können, werden neben Feldlängenangaben bzw. Offsets Terminatorzeichen verwendet (Vgl. IHO 2000).

Ein Beispiel für eine ASCII-Kodierung eines Objektes in S-57 zeigt Geyer (2007) in Tabelle 1 mit einem roten Turm. Zur Vereinfachung wurden obligatorische Records, Fields und Subfields weggelassen. Als Folge dessen, dass ein roter Turm ein Feature-Objekt ist, wird ein Feature-Record angelegt. Da es jedoch keine Oberklasse *Turm* gibt, wird die Objektklasse *Landmark (LNDMRK)* verwendet. Das zugehörige Attribut *Category of Landmark* beinhaltet den Wert 17, welcher die ID für *Tower* ist. Des Weiteren ist sowohl das Subfield *Record Name (RCNM)* als auch die *Record Identifcation Number (RCID)* vorhanden. Diese dienen zur eindeutigen Identifikation von Records. Die geographische Position wird durch das *NAME*-Subfield des *FSPT*-Fields *(Field Record to Spatial Record Pointer)* zugeordnet.

Wert	**Ebene**	**Bedeutung**
DE221000.000	DataSet	
00001	Record	ISO/IEC 8211 Record Identifier
FRID	Field	Feature Record Identifier Field
RCNM	Subfield	Record Name
100	Subfield-Wert	
OBJL	Subfield	Object Label/Code
LNDMRK	Subfield-Wert	
ATTF	Field	Feature Record Attribute
ATTL	Subfield	Attribute Label/Code
CATLMK	Subfield-Wert	Category of Landmark
ATVL	Subfield	Attribute Value
17	Subfield-Wert	Tower
ATTF	Field	Feature Record Attribute
ATTL	Subfield	Attribute Label/Code
COLOUR	Subfield-Wert	Colour
ATVL	Subfield	Attribute Value
3	Subfield-Wert	Red
FSPT	Field	Feature Record Record to Spatial Record Pointer
NAME	Subfield	Name
6E36090000	Subfield-Wert	Verweis auf einen Spatial-Record

Tabelle 1: Beispiel für die ASCII-Kodierung eines roten Turms in S-57 (Vgl. Geyer 2007)

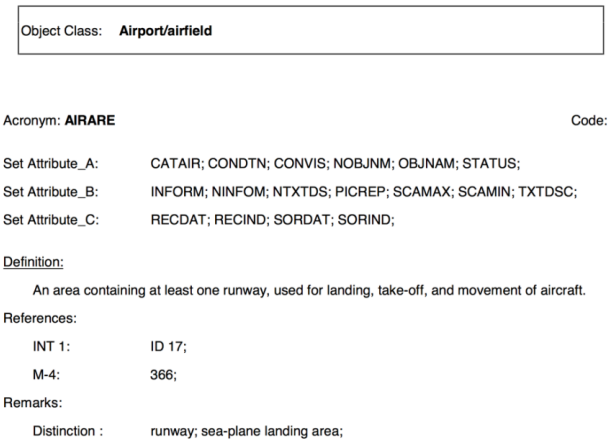
Objekt- und Attributkatalog

Der S-57 Standard verfügt über einen Objekt- und einen Attributkatalog. Diese werden in Appendix A des Standards aufgeführt. Die jeweiligen Kataloge verfügen über weit mehr als 150 Einträge je Katalog. Im Laufe der Zeit wurden jedoch bereits eine Vielzahl an Objekt- und Attributeinträgen aus dem Standard gestrichten. Als Grundlage für die Kataloge wurden Symbole und Schreibweisen aus dem INT 1 Standard, der den Standard für Papierkarten stellt, adaptiert (Vgl. IHO Appendix A Teil 1 2000).

Jedes Objekt verfügt über einen Klarnamen sowie eine sechs Zeichen langes Akronym und einen Objektcode, welcher in der Regel über eine Zahl repräsentiert wird. Hinzukommen sowohl eine kurze Definition eines Objekts als auch Referenzen, wie beispielsweise der INT 1 Standard, die Kennzeichnen wo das Objekt spezifiziert wird. Abschließend gibt es in den Objekten Hinweise wofür diese Objekte angedacht sind (Vgl. IHO Appendix A Teil 1 2000). Ein Beispiel für ein *Airport-Objekt* ist in Abbildung 3 dargestellt.

Auch Attribute sind ähnlich wie Objekte definiert. Attribute verfügen ebenso wie Objekte über einen Klarnamen und ein sechs Zeichen langes

Akronym, sowie einen Zahlencode. Hinzu kommt einer von sechs Attributtypen und ebenfalls eine Definition des Attributs. Vergleichbar zu den Objekten verfügen Attribute ebenso Referenzen zu anderen Standards. Teilweise verfügen Attribute zudem über Hinweise in welchem Format die Daten abgelegt werden müssen (IHO Appendix A 2000). In Abbildung 4 wird das Attribut *Category of canal* exemplarisch visualisiert.



Object Class: **Airport/airfield**

Acronym: **AIRARE** Code: **2**

Set Attribute_A: CATAIR; CONDTN; CONVIS; NOBJNM; OBJNAM; STATUS;

Set Attribute_B: INFORM; NINFOM; NTXTDS; PICREP; SCAMAX; SCAMIN; TXTDSC;

Set Attribute_C: RECDAT; RECIND; SORDAT; SORIND;

Definition:

An area containing at least one runway, used for landing, take-off, and movement of aircraft.

References:

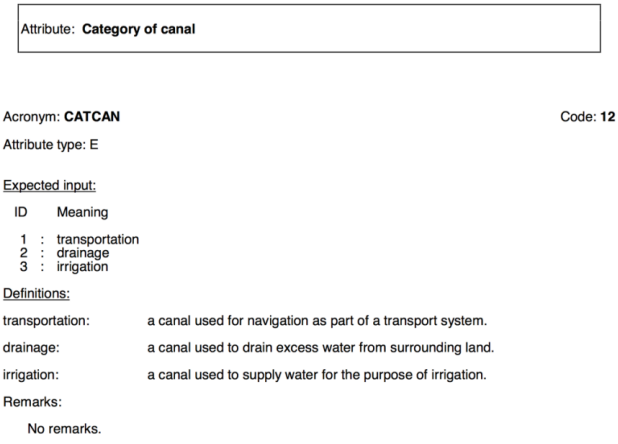
INT 1: ID 17;

M-4: 366;

Remarks:

Distinction : runway; sea-plane landing area;

Abbildung 3: Beispiel für einen Objektkatalogeintrag (Vgl. IHO Appendix A Teil 1 2000)



Attribute: **Category of canal**

Acronym: **CATCAN** Code: **12**

Attribute type: E

Expected input:

ID	Meaning
1	transportation
2	drainage
3	irrigation

Definitions:

transportation: a canal used for navigation as part of a transport system.

drainage: a canal used to drain excess water from surrounding land.

irrigation: a canal used to supply water for the purpose of irrigation.

Remarks:

No remarks.

Abbildung 4: Beispiel für einen Attributkatalogeintrag (Vgl. IHO Appendix A Teil 2 2000)

ENC Product Specification

Als Folge dessen, dass der S-57 Standard für alle Formen von hydrographischen Karten entwickelt wurde, kann der Standard nicht auf spezifische Anforderungen Rücksicht nehmen. Als Lösung dafür wurden *Product Specifications* eingeführt. Diese Product Specifications können den Standard sowohl einschränken, als auch erweitern. Dies kann in Form von Verboten für bestimmte Objektklassen im Objektkatalog erreicht werden oder durch das Einführen neuer Objektklassen. Wenn eine Product Specification als Teil des S-57 Standards veröffentlicht wurde, gelten diese Daten als vollkommen Standardkonform und müssen allerseits akzeptiert werden. Die einzige offizielle veröffentlichte Product Specification in der Version von 2000 ist die *ENC Product Specification* (Vgl. IHO Appendix B 2000, Geyer 2007).

Neben Einschränkungen im Objekt- und Attributkatalog werden in der ENC Product Specification Modellierungsvorgaben eingeführt. Folglich ist definiert wie reale Objekte, mithilfe der in den Katalogen zur Verfügung stehenden Objekt- und Attributklassen, nachgebildet werden müssen. Dies ist besonders wichtig, wenn sich die realen Objekte aus mehreren Objekten des Objektkatalogs zusammensetzen (Vgl. IHO Appendix B 2000, Geyer 2007).

Die ENC Product Specification spezifiziert Daten, bei denen unterschiedliche Möglichkeiten zur Auswahl standen, sodass keine Fragen aufkommen sollten. Zum Beispiel müssen Koordinatensysteme als Bezug Längen- und Breitengrade haben oder Tiefen, Höhen und Positionsungenauigkeiten in Metern angegeben werden. Hinzu kommt, dass die Datenstruktur durch eine klare Definition für den Aufbau von Exchange Sets eingeschränkt wird. So wird in der Product Specification angegeben wie viele Dateien von welchem Typ vorhanden sein müssen und welche Größe diese haben dürfen. Zudem wird spezifiziert, dass lediglich die Binär-Implementierung des S-57 Standards verwendet werden darf (Vgl. IHO Appendix B 2000, Geyer 2007).

Für die räumlichen Angaben für ENCs wird die Vektorrepräsentation *Chain-Node Level* verwendet. Dabei spiegelt ein *Knoten* einen Punkt, für den die geographische Position bekannt ist, dar. *Isolated Nodes* werden für Positionsangaben von Feature-Objekten verwendet und *Connect ed Nodes* sind Teile von Kanten. Jeder der Knoten wird durch einen *Spatial-Record* dargestellt. Kanten wiederum werden durch eine Menge von Punkten, deren geographische Position bekannt ist, verbunden. Jede Kante verweist auf einen Connected Node als Startknoten und einen als Endknoten (Vgl. Geyer 2007).

Kritik

Als grundlegende Kritik kann die Strukturierung des Standards angemerkt werden. Der S-57 Standard ist nicht in einem zusammenhängenden Dokument veröffentlicht. Um auf zugehörige Appendizes zugreifen zu können müssen diese extra auf der Internetseite des IHO abgerufen werden. Auch diese werden nicht gesammelt zur Verfügung gestellt, sondern als einzelne Publikationen veröffentlicht. Hinzu kommt, dass keine fortlaufende Seitennummerierung vorhanden ist, die eine Suche in dem Standard vereinfachen würde.

Durch die gegebene Gliederung sind relevante Informationen in dem gesamten Standard verstreut. So zum Beispiel finden sich Informationen zu einzelnen Themen sowohl im Hauptdokument als auch in der Product Specification, welche als Appendix eine eigene Publikation darstellt. Folglich entstehen Fragen, die Geyer (2007) in einem Beispiel sehr anschaulich beschreibt. Es gibt sechs Objektklassen für Tonnen, die wiederum über eine Vielzahl von Attributwerten verfügt um unterschiedliche Tonnentypen zu modellieren. Jedoch besteht keine der Modellierung für beleuchtete Tonnen. Lichter werden hierbei als separate Objekte modelliert. Somit besteht keine konsistente Modellierung für beleuchtete Tonnen. In einem allgemeinen Abschnitt der ENC Product Specification wird eine verbindliche Methode vorgegeben. Nicht jedoch in dem Abschnitt für Tonnen. Auch ein Verweis zwischen den Abschnitten besteht nicht. Hinzu kommen Regeln für Ausnahmefälle, sodass in anderen Teilen des Standards weitergelesen werden muss, um die gewünschten Informationen zu beziehen. Ergänzend kann kritisiert werden, dass in der Product Specification weitere Modellierung genehmigt werden und nicht eine Modellierung festgeschrieben wird. Als Folge dieser Probleme wurde das *Internantional Centre for ENCs* gegründet, um die Erstellung und Verbreitung von ENCs zu harmonisieren.

Abschließend ist zu sagen, dass auch aus Sicht der Informatik Kritikpunkte an dem Standard aufkommen. Ein einheitlicher Objektkatalog würde der verwirrenden Aufteilung zwischen Feature-Records und Objektklassen und Attribute entgegenwirken. Teilweise sind Informationen über die Feature-Record-Struktur definiert und teilweise durch den Objekt- und Attributkatalog (Vgl. Geyer 2007).

Bezug zum Projekt

In Bezug auf das Projekt wird ein grundlegendes Verständnis des S-57 Standards von Nöten sein, damit eine intelligente Schiffsimulation erstellt werden kann. Dies wird dadurch bedingt, da ENCs die Grundlage für die Navigation auf See bilden und diese wiederum durch den S-57 Standard reglementiert werden. Damit in der Simulation auf Daten der ENCs zugegriffen werden kann, muss die Datenstruktur und die damit verbundene Modellierung verstanden werden.

Kontextbetrachtung

S-52 Standard

Der S-57 Standard steht in engem Zusammenhang mit dem S-52 Standard. Der genaue Name des Standards lautet *Specifications for Chart Content and Display Aspects of ECDIS*. Die aktuellste Version ist die Version 6.0 aus dem März 2010. Der Standard wurde entwickelt, um die Sicherheit im Umgang mit ECDIS-Systemen zu erhöhen. So sollen die angezeigten Informationen selbsterklärend und eindeutig sein, sodass keine Fragen während der Nutzung aufkommen (Vgl. IHO 2010).

Folglich müssen die im S-52 Standard definierten Spezifikationen für ECDIS im S-57 Standard berücksichtigt werden, sodass diese nach dem theoretischen Datenmodell abgelegt und anschließend als Daten transferiert werden können. Hinzu kommt, dass der S-52 Standard die Objekte spezifiziert, die in ENCs und auf ECDIS-Systemen dargestellt werden sollen und somit die Feature-Objekte im S-57 Standard vorgegeben werden.

S-63 Standard

Einzelne Länder produzieren standardisierte ENCs für deren Zuständigkeitsbereiche. Dazu gehören beispielsweise innere Gewässer, Küstenmeere und Festlandsöckel. Im Vergleich zu den USA sind die ENCs in Europa nicht kostenfrei verfügbar und müssen somit über autorisierte Vertriebsstellen bezogen werden. In Europa vertreibt das *International Centre for ENCs* beispielsweise die von der BSH produzierten elektronischen Karten (Vgl. BSH 2017)

Damit die ENCs nicht durch Piraterie vertrieben werden können wurde von der IHO der S-63 Standard entwickelt. In der aktuellen Version 1.1.1 vom April 2012 wird der Standard unter dem Namen *IHO Data Protection Scheme* veröffentlicht. Der Standard soll bewirken, dass die ENCs über die regulären Wege vertrieben werden und kein Datenmissbrauch zustande kommen kann (Vgl. IHO 2012).

Literaturverzeichnis

BSH (2017) Bezugsquellen für ENCs. Online: http://www.bsh.de/de/Produkte/Karten/Elektronische_Seekarten/354.jsp)
ECD (2017) What is ECDIS?. Online: http://www.ecdis-info.com/about_ecdis.html
FTO (2015) FreieTonne. Online: https://www.freietonne.de
Geyer, F. (2007) Entwicklung einer Schiffsumweltsimulation in C++ auf Basis digitaler Seekarten im S-57-Format. Online: http://www2.uni-jena.de/philosophie/IWK-neu/typo3/fileadmin/forschung/projekte/iwk_dgon_memo_3.pdf
IHO (2000) IHO Transfer Standard for hydrographic Data. Online: https://www.iho.int/iho_pubs/standard/S-57Ed3.1/31Main.pdf
IHO Appendix A Teil 1 (2000) IHO Transfer Standard for hydrographic Data. Online: https://www.iho.int/iho_pubs/standard/S-57Ed3.1/31ApAch1.pdf
IHO Appendix A Teil 2 (2000) IHO Transfer Standard for hydrographic Data. Online: https://www.iho.int/iho_pubs/standard/S-57Ed3.1/31ApAch2.pdf
IHO Appendix B (2000) IHO Transfer Standard for hydrographic Data. Online: https://www.iho.int/iho_pubs/standard/S-57Ed3.1/20ApB1.pdf
IHO (2010) Specifications for Chart Content and Display Aspects of ECDIS. Online: https://www.iho.int/iho_pubs/standard/S-52/S-52_e6.0_EN.pdf
IHO (2012) IHO Data Protection Scheme. Online: https://www.iho.int/iho_pubs/standard/S-63/S-63_e1.1.1_EN_Apr12.pdf
IHO (2015) Definition of Hydrography. Online: http://www.iho.int/srv1/index.php?option=com_content&view=article&id=299:definition-of-hydrography&catid=42&Itemid=289&lang=en
IHO (2017) Member States Information Online: http://www.iho.int/srv1/index.php?option=com_wrapper&view=wrapper&Itemid=452&lang=en
NAC (2013) Symbols, Abbreviations and Terms used on Paper and Electronic Navigational Charts. Online: https://www.nauticalcharts.noaa.gov/mcd/chart1/ChartNo1.pdf
OSM (2016) OpenSeaMap. Online: http://wiki.openstreetmap.org/wiki/DE:OpenSeaMap
SKG (2017) Elektronische Seekarten. Online: https://www.skipperguide.de/wiki/Elektronische_Seekarten

Präsentation

Verkehrsregeln auf See

Im Rahmen der Projektgruppe „Intelligente Schiffssimulation“ findet eine Seminarphase über die relevanten Themen statt. Diese Arbeit beschreibt die Verkehrsregeln auf See und betrachtet im speziellen die die COLREGs.

Inhaltsverzeichnis

Überblick

Die COLREGS - International Regulations for Preventing Collisions at Sea wurden im Oktober 1972 auf einer internationalen Konferenz ausgearbeitet. Schon auf der internationalen Konferenz 1960 wurden Regularien getroffen. Allerdings wurde sehr schnell deutlich, dass die Einführung des Radars und die Entwicklung immer größerer und schnellerer Schiffe neuer Regeln Bedarf. So wurden 1972 erweiterte Regularien aufgestellt, welche 1977 in Kraft traten.

Auch davor gab es Regeln in der Schifffahrt, um Kollisionen zu vermeiden, jedoch waren diese keine geregelten, von Gesetz her in Kraft getretenen Regeln. Diese wurden erstmals 1840 im London Trinity House festgelegt, in denen es hauptsächlich darum ging, wohin welches Schiff in einem engen Kanal ausweichen soll. Diese Regularien wurden 1846 vom Parlament verordnet.

Im Jahr 1863 konsultierten erstmals das British Board of Trade und die französische Regierung zusammen, um neue Regularien auszuarbeiten. Ende 1864 unterschrieben neben den USA und Deutschland über 30 Staaten dieses Abkommen, bzw. bis zu diesem Stand noch Artikel.

Die bis dato geltenden Regel, welche 1977 in Kraft traten, bestehen aus 5 Teilen, A - E, und 4 Anhängen. Diese 5 Teile bestehen aus insgesamt 38 Regeln, wobei Teil B aus 3 Untersektionen besteht. Zusätzlich hat jedes ein Oberthema, Teil A Allgemeines, Teil B Ausweich- und Fahrregeln, Teil C Lichterführung und Signalkörper, Teil D Schall- und Lichtsignale und Teil E Befreiungen. Im Folgenden werden alle Regeln in ihren Sektionen erwähnt. Hierbei werden nicht alle Kommentare, welche in der Ausarbeitung aus dem Jahr 1972 stehen, aufgeführt. Weiter werden die Regeln aufgelistet werden, da dieses einen einfachen und schnellen Überblick gewährt und eine einfache Sortierung ermöglicht.

COLREG

Teil A: Allgemeines

Rule 1: Application/Anwendungsbereiche

- Betrifft alle Schiffe auf hoher See und in allen befahrbaren Gewässern
- Diese Regeln beeinträchtigen keine speziellen Regeln von Staaten oder Herrscher der Gewässer, Seen, Häfen o.ä.
- Spezielle Gefährte, welche die Regularien nicht komplett erfüllen, sollen mit Respekt behandelt werden

Rule 2: Responsibility/Verantwortlichkeit

- Diese Regeln schützen niemanden vor der Verantwortung bei Vernachlässigung
- Die Umstände bei einer Kollision oder einem Ausweichen müssen immer bedacht werden

Rule 3: General definitions/allgemeine Begriffsbestimmungen

- Vessel
  - Jegliches Wasserfahrzeug, auch WIGs und Wasserflugzeuge
- Sailing vessel
  - Jegliches Schiff, welches unter Segeln und ohne externen Antrieb fährt, Segelschiff
- Power-driven vessel

    - Schiffe, welche durch Motoren angetrieben werden
- Vessel engaged in fishing
    - Jegliche Schiffe, welche mit Netzen, Leinen oder anderen Fischerutensilien, welche die Manövrierfähigkeit einschränken, fischen, Fischerboot/Schlepp
- Seaplane
    - Flugzeuge, welche auf See fahren können
- Not under command
    - Schiffe, welche unter außergewöhnlichen Umständen nicht manövrieren können, Nicht Kommandiert
- Restricted in her ability to manoeuvre
    - Schiffe, welche auf Grund ihrer Bauweise in Ihrer Manövrierfähigkeit eingeschränkt sind. Dazu zählen:
        - Liegende Schiffe, wartende Schiffe und Schiffe, welche Gegenstände aufnehmen
        - Schiffe zum Baggern, zum Vermessen und für Unterwasseroperationen
        - Schiffe zum Transportieren von Menschen oder für den Nachschub
        - Flugzeugträger
        - Minenräumschiffe
        - Schlepper
- Underway
    - Schiffe, welche sich bewegen
- Length and breadth
    - Länge und Breite der Schiffe
- Restricted visibility
    - Eingeschränkte Sicht, z.B. bei Nebel, Schnee, Regenstürme, etc.
- Wing in Ground
    - Erdbodennahes Flugzeug

Teil B: Ausweich- und Fahrregeln

Section 1

Rule 4: Application/Anwendungsbereich

- Rules in this section apply in any condition of visibility

Rule 5: Look-out/Umgebungsbeobachtung

- Jedes Gefährt sollte in jeder Situation, den Umständen entsprechend, visuell und hörbar sein und sich auch so verhalten

Rule 6: Safe speed/sichere Geschwindigkeit

- Alle Schiffe müssen ihre Geschwindigkeit auf Grundlage der Sichtweite, des Verkehrs, der Manövrierfähigkeit, der Tageszeit, der Wettergegebenheiten und des Entwurfes des Schiffes anpassen
- Bei Schiffen mit Radar sollte die Geschwindigkeit anhand von Effizienz und Charakteristik des Radars, der Weite des Radars, Wettergegebenheiten, die Möglichkeit von Objekten die dem Radar entgehen, die Anzahl der entdeckten Schiffe und die Entfernung der Schiffe angepasst werden

Rule 7: Risk of collision/Risiko einer Kollision

- Jedes Schiff muss sich entsprechend der Umstände das Risiko einer Kollision bewusst sein und dementsprechend handeln
- Risikovermeidung durch Radar und durch Anpassen an die Wettergegebenheiten und allgemeinen Umstände
- keine Annahmen durch spärliche Informationen treffen
- Risiko erhöht sich bei ausfallendem Kompass oder bei einer Abschleppung o.ä.

Rule 8: Action to avoid collision/Aktionen zur Vermeidung von Kollisionen

- Ausweichen anhand der Regeln
- Änderung des Kurses/Geschwindigkeit groß genug, immer auf Grundlage der Umstände
- Wenn der Seeraum groß genug ist à aus sicherer Distanz Kurs ändern und Geschwindigkeit drosseln
- Falls ein Schiff nicht ausweichen kann sollte er genug Raum schaffen für das andere schiff
- Schiffe welche den Verkehr bzw. andere Schiffe nicht behindern, haben Schuldminderung bei eventuellen Kollisionen und, sollte die Aktion früh passieren, komplett von der Schuld entbunden

Rule 9: Narrow channels/Kanäle

- Schiffe in einem Kanal sollten möglichst eng am Rand des Kanals zu ihrer Steuerbord Seite fahren à Rechtsfahrgebot, somit auch Rechts vor Links
- Fischerschiffe und Schiffe die kleiner als 20 Meter sind dürfen die Kanäle nicht behindern
- Kanäle dürfen nicht gekreuzt werden, wenn dies behindern sollte à soundsignale, Beim Überholen à soundsignale
- Niemals vor Anker gehen!

Rule 10: Traffic separation schemes/Verkehrregelungsschemata

- Alle Schiffe sollten in der Verkehrsspur bleiben, die Verkehrsspur freihalten, am Ende der Bahn die Spur verlassen/einsteigen à sollte dies nicht möglich sein, dann in direkter Richtung zu der Spur einkehren

- Möglichst nicht kreuzen, wenn doch im rechten Winkel einkehren
- Nur Segelschiffe, Fischerschiffe und 20 Meter lange Schiffe dürfen die Küstenverkehrszone nehmen, wenn es notwendig ist(Hafen oder ähnliches),
- Spur darf normalerweise nicht verlassen werden, außer zum Fischen in seperaten Zonen, um Gefahr auszuweichen
- Niemals vor Anker gehen
- Schiffe bis 20 Metern sollten nicht die Sicherheitszonen von maschinenbetriebenen Schiffen benutzen
- Schiffe, welche beschränkt in ihrer Manövrierfähigkeit sind, auf Grund ihrer Bauweise, sind von diesen Regeln ausgenommen

Section 2

Rule 11: Application/Anwendungsbereich

- Rules in this section to vessels in sight of one another

Rule 12: Sailing vessels/Segelschiffe

- Wenn der Wind für beide Schiffe von verschiedenen Seiten kommt, weicht das Schiff aus, welches den Wind von Backboard bekommt
- Wenn der Wind von der gleichen Seite kommt, weicht das Schiff aus, welches zum Wind hin liegt à Schiff mit Wind zur Lee Seite(Wind abgewandte Seite) bleibt
- Bei Zweifeln sollte das Schiff ausweichen, welches den Wind auf der Steuerbord Seite hat

Rule 13: Overtaking/Überholen

- Die Überholenden sollten nicht den Weg der Überholten kreuzen
- Bei 22,5 „degrees abaft her beam“ Sichtweite durch Licht, da dies die Sichtweite eines Lichts bei Nacht ist
- bei Zweifel à handeln und überholen
- Wenn Überholen angesetzt wurde, muss dies zuende gebracht werden

Rule 14: Head-on situation/Frontal Situationen

- Maschinenbetriebene schiffe sollen ihren Kurs zu Steuerbord ändern, sodass sie Backboard aneinander vorbei können
- Diese Regeln anwenden, sobald ein Schiff oder ein Licht in Sicht kommt
- Bei Zweifel sollte die Regel trotzdem ausgeführt werden

Rule 15: crossing situation/Situation des Kreuzens

- Bei maschinenbetriebenen Schiffen, welche sich kreuzen würden, muss das Schiff, welches das andere Schiff auf Steuerbord-Seite hat, ausweichen, solange es die Umstände zulassen

Rule 16: action by give-way vessel/Ausweichende Schiffe

- Schiffe, welche anhand der Regeln ausweichen müssen, sollten schnelle und deutliche Aktionen ausführen

Rule 17: action by stand-on vessel/Bleibende Schiffe

- Das nicht ausweichende Schiff sollte den Kurs und die Geschwindigkeit beibehalten
- Das langsamer werdende Schiff sollte als einziges ausweichen
- Sollte es doch zu einer Kollision kommen können, sollte auch das nicht ausweichende Schiff seinen Kurs so bewusst wie möglich ändern

Rule 18: Responsibilities between vessels/Verantwortlichkeiten

- Rules 9,10,13 ausgenommen
- Ein maschinenbetriebenes Schiff weicht folgenden Schiffen aus
    - nicht kontrollierten, eingeschränkten, fischend, segelnd
- Segelschiff weicht aus
    - nicht kontrollierten, eingeschränkten, fischend
- Fischerschiff weicht aus
    - nicht kontrollierten, eingeschränkten
- Speziell gebaute schiffe müssen nicht ausweichen, müssen aber vorsichtig fahren
- Seeflugzeuge müssen auf alle achten, genauso ein WIG

Section 3

Rule 19: Conduct of vessels in restricted visibility/Verhalten von Schiffen bei eingeschränkter Sicht

- Sichere Geschwindigkeit
- Maschinenbetriebene Schiffe sollten ihre Maschinen bereit haben
- Regel befolgen, soweit es die Umstände zulassen
- Bei Erkennen eines Schiffes auf Radar und/oder die Möglichkeit einer Kollision liegt vor, ändert das Schiff den Kurs, sodass das Licht des anderen Schiffes ausgewichen werden kann
- Sollte das Nebelhorn zu hören sein, muss die Geschwindigkeit gedrosselt werden

Teil C: Lichterführung und Signalkörper

Rule 20: Application/Anwendungsbereich

- Diese Regeln gelten von Sonnenaufgang bis Sonnenuntergang, bei jedem Wetter und jedem Wetterumstand
- Die Regeln für die Lichter und Formen müssen mit Annex I übereinstimmen

Rule 21: Defintions/Definitionen

- Masthead light --> weißes Licht, sowohl nach vorne, als auch nach hinten(225 degrees)
- Sidelights --> grünes Licht Steuerboard und rotes Licht Backboard (112.5 degrees)
- Sternlight --> weißes Licht am Heck (135 degrees)
- Towing light --> gelbes Licht am Heck, zum Abschleppen (135 degrees)
- All-round light --> 360 Grad Licht
- Flashing light --> 120 Blitze pro Minute

Rule 22: Visibility of lights/Sichtbarkeit der Lichter

- Bei Schiffen von 50 Metern oder mehr
  - masthead light, 6 miles;
  - sidelight, 3 miles;
  - sternlight, 3 miles;
  - towing light, 3 miles;
  - white, red, green or yellow all-round light, 3 miles.
- Bei Schiffen zwischen 12 und 50 Metern
  - masthead light, 5 miles; except that where the length of the vessel is less than 20 metres, 3 miles;
  - sidelight, 2 miles;
  - sternlight, 2 miles;
  - towing light, 2 miles;
  - white, red, green or yellow all-round light, 2 miles.
- Bei Schiffen bis zu 12 Metern
  - masthead light, 2 miles;
  - sidelight, 1 mile;
  - sternlight, 2 miles;
  - towing light, 2 miles;
  - white, red, green or yellow all-round light, 2 miles.
- Bei unauffälligen, untergetauchten oder abgeschleppten Schiffen
  - white all-round light, 3 miles.

Rule 23: Power-driven vessels underway/Maschinenbetriebene Schiffe unterwegs

- Diese Schiffe sollten haben
  - Masthead light
  - ein höher gelegenes Masthead light, bei einer Größe von über 50 Metern
  - sidelights, sternlight
- Luftkissenboot
  - ein rundum gelbes licht
- WIG
  - ein rundum rotes licht
- Maschinenbetriebene Schiffe bis 12 Meter
  - Ein rundum weißes Licht und Seitenlichter, zusammen in einem Licht
  - Schiffe weniger als 7 Meter sollten ein rundum weißes Licht besitzen

Rule 24: Towing and pushing/Abgeschleppte oder geschobene Schiffe

- <u>Wenn gezogen:</u>
  - 2 Masthead light in einer vertikalen Linie
  - Seitenlichter, Hecklicht, Abschlepplicht in vertikaler Linie zum Hecklicht
  - Seil größer 200 Meter à Licht in Diamantenform am Ende des Schiffes und da wo es am besten gesehen wird
  - <u>Wenn ziehend:</u>
    - 2 Masthead light in einer vertikalen Linie
    - Seitenlichter, Hecklicht
    - Weniger als 25 Meter breit à kein zusätzliches licht
    - Mehr als 25 Meter breit à 2 rundum Lichter an den Seitenwänden
    - 100 Meter lang à mehr Lichter, sodass die nicht mehr als 100 Meter auseinander liegen

Rule 25: Sailing vessels underway and vessels under oars/Segelschiffe und Ruderboote

- <u>Segelschiffe</u>
  - Seitenlichter und ein Hecklicht, kann 2 Masthead Lichter (höhere rot, niedrigere grün) haben
  - Ein Ruderboot oder ein Segelschiff kleiner als 7 Meter muss diese Lichter nicht haben, sollte aber ein Handlicht besitzen

Rule 26: Fishing vessels/Fischerschiffe

- Schlepper
    - 2 rundum Lichter, das höhere grün, das niedrigere weiß
    - Bei Schiffen bis 50 Metern kann es ein Masthead light geben
    - Bei Schleppern, die sich bewegen sollte es zusätzlich Seitenlichter und ein Hecklicht geben
    - Fischerboot
        - 2 rundum Lichter, das höhere rot, das niedrigere weiß
        - Bei frei liegendem Equipment im Wasser muss dieses beleuchtet werden
        - Bei Fahren des Schiffes, Seitenlichter und ein Hecklicht
        - Ansonsten Regeln der maschinenbetriebenen Schiffe beachten

Rule 27: Vessels not under command or restricted in their ability to manoeuvre/Schiffe nicht unter Kontrolle oder eingeschränkt in der Beweglichkeit

- Not under control
    - 2 rote vertikale Lichter oder 2 rote Lichter in Ballform
    - 2 grüne vertikale Lichter oder 2 Diamandformlichter, an der Seite, an der überholt werden kann
    - Bei Bewegung zusätzlich Seitenlichter und Hecklicht
    - Eingeschränktes Schiff
        - 3 rundum Lichter in der Reihenfolge: Rot-Weiß-Rot
        - 3 Lichter in den Formen: Ball-Diamand-Ball
        - Bei Bewegung zusätzlich Seitenlichter und Hecklicht
        - Bei Ankerung Regel 30 beachten
        - Schiffe beim Abschleppen oder beim Ausbaggern halten sich an die Regel für eingeschränkte Schiffe
    - Schiffe, welche Minenräumarbeiten machen oder Unterwasserarbeiten ausführen
        - 3 rundum Lichter in vertikaler Linie mit der Reihenfolge: Rot - Weiß - Rot

Rule 28: Vessels constrained by their draught/Schiffe, die Aufgrund ihrer Bauweise eingeschränkt sind

- 3 rote rundum Lichter in vertikaler Linie

Rule 29: Pilot vessels/Lotsenboot

- 2 rundum Lichter, das obere weiß das untere rot
- Wenn in Bewegung zusätzlich Seitenlichter und ein Hecklicht
- Wenn vor Ankerung, siehe Regel 30
- Ansonsten anpassen an die Größe des Gefährts

Rule 30: anchored vessels and vessels aground/Ankernde und auf Grund liegende Schiffe

- Ankernde Schiffe sollten dort Lichter haben, wo sie am besten gesehen werden
    - Im vorderen Teil ein weißes Licht (Ball)
    - am Heck ein weißes rundum Licht
    - Schiffe bis zu 50 Metern sollten zusätzlich ein rundum weißes Licht haben
    - Schiffe größer als 100 Metern sollten zusätzlich Lichter haben, die das Deck erleuchten
    - Ein gestrandetes Schiff sollte haben
        - 2 rote rundum Lichter
        - 3 Lichtbälle in vertikaler Linie

Rule 31: seaplanes/Seeflugzeuge

- Da wo es möglich ist Lichter anbringen

Teil D: Schall- und Lichtsignale

Rule 32: Definition

- Whistle à Ton, der anhand von Annex III getätigt wird
- Short blast à 1 second
- Prolonged blast à 4-6 seconds

Rule 33: Equipment for sound signals/Ausrüstung für Tonsignale

- Schiffe bis 12 Metern
    - Sollten ein ähnliches Tonsignal dabei haben
    - Schiffe zwischen 12 und 20 Metern
        - Whistle
        - Schiffe zwischen 20 und 100 Metern
            - Glocke
            - Schiffe größer als 100 Meter
                - Gong

Rule 34: Manoeuvring and warning signals/Manövrier- und Warnsignale

- Ein Schiff, welches entweder auf ein anderes trifft, manövriert oder ausweicht muss folgende Signale geben

        - 1 short blast: Kursänderung zu Steuerbord
        - 2 short blast: Kursänderung zu Backboard
        - 3 short blast: Betreibe Achterantrieb
        - Zusätzliche Lichtsignale erforderlich
            - 1 Sekunde licht, 1 Sekunde pause
            - weißes Licht in 5 Miles zu sehen sein
            - 1 flash: Kursänderung zu Steuerbord
            - 2 flash: Kursänderung zu Backboard
            - 3 flash: Betreibe Achterantrieb
    - Ein Schiff, welches ein Überholmanöver einleitet
        - 2 verlängerte + 1 kurzen: überholen an steuerbord
        - 2 verlängerte und 2 kurze: überholen an backbord
    - Ein Schiff, welches überholt wird
        - 1 langen, 1 kurzen, 1 langen und einen kurzen
        - Bei Zweifel: 5 kurze und schnelle töne + 5 lichter
    - Schiffe, welche in eine Fahrbahn einbiegen oder hinter einer Biegung, sind müssen einen verlängerten Ton abgeben, wenn dieser beantwortet werden kann, dann nur mit einem verlängerten Ton

Rule 35: Sound signals in restricted visibility/Tonsignale bei eingeschränkter Sicht

- Maschinenbetrieben Schiffe sollten bei Bewegung alle 2 Minuten einen langen Ton abgeben
- Bei nicht Bewegung 2 lange alle 2 Minuten
- Schiffe, welche nicht unter Kontrolle, welche eingeschränkt in der Manövrierfähigkeit, welche wegen ihrer Bauweise eingeschränkt sind, Segelschiffe, Fischerschiffe, welche ziehen oder schieben werden müssen 3 kurze Töne alle 2 Minuten abgeben
- Ein Schiffe, welches gezogen wird sollte 1 langen und 3 kurze Töne innerhalb von 2 Minuten abgeben
- Schiffe, welche ankern müssen die Glocke für 5 Sekunden jede Minute läuten, Schiffe die größer als 100 Meter sind zusätzlich noch den Gong
- Schiffe bis 20 Meter sollen ähnliche Töne abgeben

Rule 36: signals to attract attention/Signale zum Erreichen der Aufmerksamkeit

- Licht und Tonsignale in Abweichungen zu den Regeln

Rule 37: distress signals/Notsignale

- Notsignale werden in Annex IV beschrieben

Teil E: Befreiungen

Rule 38: exemptions/Befreiungen

- Hauptsächlich Befreiungen bei den Lichtkonfigurationen für Schiffe, welche Ausrüstungen haben, die vor der Einführung der Regularien existierten

Annex I - Positioning and Technical Details of Lights and Shapes

1. Definition
2. Vertical Positioning and spacing of lights
3. Horizontal positioning and spacing of lights
4. Detail of location of direction-indicating lights for fishing vessels, dredgers and vessels engaged in underwater operations
5. Screens for sidelights
6. Shapes
7. Colour specification of lights
8. Intensity of lights
9. Horizontal sectors
10. Vertical sectors
11. Intensity of non-electric lights
12. Manoeuvring light
13. High-speed craft
14. Approval

Annex II - Additional signals for fishing vessels fishing in close proximity

1. General
2. Signals of trawlers
3. Signals for purse seiners

Annex III - Technical details of sound signal appliances

1. Whistles
    a. Frequencies and range of audibility
    b. Limits of fundamental frequencies
    c. Sound signal intensity and range of audibility
    d. Directional properties
    e. Positioning of whistles
    f. Fitting of more than one whistle
    g. Combined whistle systems
    h. Bell or gong
        i. Intensity of signal
        ii. Construction
        iii. Approval

Annex IV - Distress signals

Ergänzungen

Die Ausführungen in Annex I-IV sind hauptsächlich zur Positionierung und für die technische Umsetzung der Licht- und Tonsignale. Allerdings zielt die Projektarbeit nicht auf den Bau eines Schiffes auf, sondern eher auf die Umsetzung der Fahrregeln für die Schiffe. Somit wird Annex I-IV in dieser Ausarbeitung nicht weiter ausgeführt. Nachgelesen kann dies in den „COLREGS - International Regulations for Preventing Collisions at Sea“, welche 1972 aufgestellt worden sind.

Um die Ausweichregeln aus den COLREGS noch einmal zu verdeutlichen, kann eine Abbildungsstrecke von Ausweichmanövern im Anhang angeschaut werden.

Hierbei wird deutlich, dass in jeder Situation, sei es überholen oder kreuzen, eine sichere Distanz gewahrt werden muss, da sonst das Risiko einer Kollision weiterhin besteht. Bei diesen Schiffen handelt sich ausschließlich um maschinenbetrieben Schiffe, da nur diese automatisiert werden können. Dementsprechend müssen auch deren Regeln angewandt werden.

Zusätzlich zu diesen Regeln gelten auf Deutschen Seestraßen Seeschifffahrtsstraßen-Ordnung (SeeSchStrO) bzw. die Schifffahrtsordnung Emsmündung (EmsSchO).

Bei der SeeSchStrO gib es insgesamt 62 Paragraphen, die in 9 Abschnitte unterteilt sind. Diese Abschnitte unterteilen sich in Allgemeine Bestimmungen, Sichtzeichen und Schallsignale der Fahrzeuge, Fahrregeln, Ruhender Verkehr, Sonstige Vorschriften, Ergänzende Vorschriften für den Nord-Ostsee-Kanal, Aufgaben und Zuständigkeiten der Behörden der Wasserstraßen- und Schifffahrtsverwaltung des Bundes und die Bußgeld- und Schlussvorschriften.

Für die im weiteren verfolgten Ziele der Projektgruppe sind gerade Abschnitt 4 und 7 wichtige Unterteilungen. Hierbei geht es um die Fahrregeln und spezielle ergänzende Vorschriften für den Nord-Ostsee-Kanal.

Die EmsSchoO beinhaltet 30 Artikel, die in 6 Abschnitte unterteilt sind. Diese sind die Allgemeinen Bestimmungen, die Sichtzeichen der Fahrzeuge, die Schallsignale der Fahrzeuge, die Fahrregeln, die Regeln für das Stilllegen und die sonstigen Vorschriften.

Gerade der 4. Abschnitt über die Fahrregeln sollte für diese PG von äußerster Wichtigkeit sein, da dort u.a. das Rechtsfahrgebot und die Ausnahmen davon erwähnt werden.

Link für die SeeSchStrO: https://www.elwis.de/Schifffahrtsrecht/Seeschifffahrtsrecht/SeeSchStrO/

Link für die EmsSchO: https://www.elwis.de/Schifffahrtsrecht/Verzeichnis-Rechtsverordnungen-Gesetze/EmsSchO.pdf

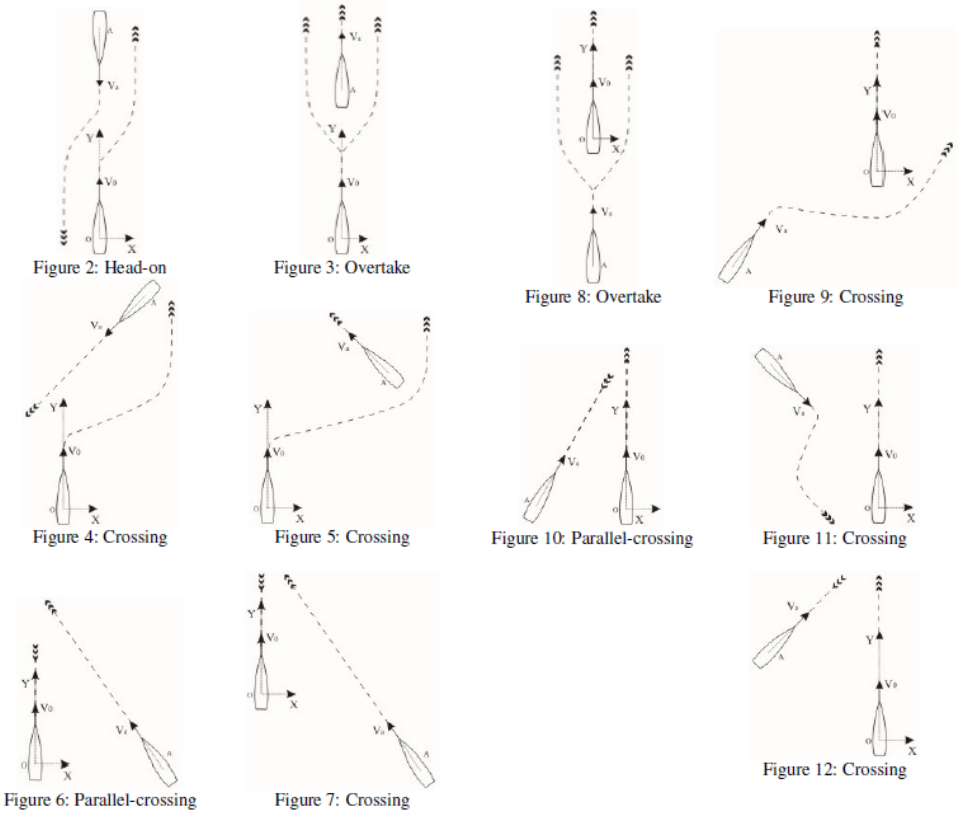


Abbildung 1: Schifffahrtsregeln fürs Crossing, Overtake und Head-on.

Literaturverzeichnis

Articles of the Convention on the International Regulations for Preventing Collisions at Sea (1972), “COLREGS - International Regulations for Preventing Collisions at Sea”, Lloyd´s Register Rulefinder 2005 - Version 9.4

Cockcroft, A. N., Lameijer, J. N. F., “A Guide To The Collision Avoidance Rules”, Incorporates the 1993 and 2011 amendments

Perera, L.P., Carvalho, J.P., Guedes Soares, C., “Autonomous guidance and navigation based on the COLREGs rules and regulations of collision avoidance”, Taylor and Francis Group, London, 2010

Präsentation

Kollisionsverhütung

Dieser Arbeit verschafft einen Überblick über die aktuellen Methoden zu Kollisionsverhütung und knüpft an der vorherigen Arbeit der Kollisionsgefahrenerkennung an.
Zudem werden die Systeme TCAS/ACAS aus der Luftfahrt vorgestellt. Des Weiteren werden die Prozesse des TCAS beschrieben. Darüber hinaus werden aus dem maritimen Bereich das MTCAS dargestellt, so wie ein Bespiel zur Verhütung von Schiffskollisionen beschrieben. Hierbei wird auf die Reduzierung und Abwendung eingegangen.
Zusätzlich werden verschieden Methoden zur Kollisionverhütung vorgestellt.

Motivation

Trotz der stetig wachsenden Sicherheit in der Luftfahrt sind Kollisionen bis 2015 noch zu verzeichnen. 2014 gab es selbst noch Unfälle trotz ausgerüstetem TCAS. Hierbei ist zu erkennen, dass die zunehmende Weiterentwicklung von Systemen eine höhere Sicherheit bietet. (Vgl. goldner, S. 36–47)

Im maritimen Bereich sind trotz der Weiterentwicklung der Navigationstechniken, Schiffskollisionen immer noch ein großer Prozentsatz an Seeunfällen, die einen negativen Einfluss auf das Leben, die Wirtschaft und Umwelt haben. (Vgl. Kim et al. 2017, S. 1–2)

Kollisionsverhütung in der Luftfahrt durch TCAS/ACAS

Um Kollisionen zu vermeiden, wird in der Luftfahrt das sogenannte Traffic Alert and Collision Avoidance System (TCAS) eingesetzt. (Vgl. Lufthansa Technik)
Das TCAS sammelt Informationen im engeren Luftraum des eigenen Luftfahrzeuges. Diese Informationen werden dem Piloten zur Verfügung gestellt, welches unabhängig von der eigentlichen Radar-Station am Boden ist. Zur Informationssammlung werden zwei Antennen benötigt, die jeweils unten und oben am Luftfahrzeug montiert sind. Zum einem um die Signale des Transponders zu empfangen und zum anderen um sie zu senden. Dieser Transponder wird auf einer festen Frequenz abgefragt und auf einer weiteren Frequenz werden Informationen wie Höhe und die individuelle Erkennung des Luftfahrzeuges übertragen. (Vgl. Lufthansa Technik)

Das TCAS ist also ein Gerät welches nach dem Prinzip eines Radar-Systems funktioniert. Es besteht aus einem audiovisuellem Gerät und dem oben erwähnten Transponder. Die eigene Position wird, indem spezifischen Bereich über 1030 MHz-Hochfrequenz übertragen. Alle anderen Luftfahrzeuge antworten über die Frequenz von über 1090 MHz. Diese Kommunikation kann mehrmals pro Sekunde durchgeführt werden. Dabei werden die Informationen über den errechneten Abstand, die Lage und Höhe des Luftfahrzeuges in Bezug auf die eigene Position übertragen und ermittelt. Über diese Daten wird dann über das TCAS eine mögliche Bedrohung einer Kollision errechnet. (Vgl. Murugan und Oblah 2010, S. 46)

Wenn eine mögliche Kollision berechnet wurde, erzeugt das TCAS ein akustisches und visuelles Signal für den Piloten. Falls das andere Luftfahrzeug ebenfalls über ein entsprechendes System verfügt, wird über die Tonhöhe eine Konfliktlösung koordiniert, sodass beide Luftfahrzeuge nicht in die gleiche Richtung manövrieren. (Vgl. Murugan und Oblah 2010, S. 46)

Ankündigung eines möglichen Konfliktes

In der Übertragung eines möglichen Konfliktes wird in zwei Arten unterschieden. Zum Einem in Traffic Advisory (TA) und zum anderen Resolution Advisory (RA). Das TCAS I kann nur die TA ausführen, also nur Verkehrshinweise. Das TCAS II hat die Möglichkeit zusätzlich eine RA, also eine Ausweichempfehlung zu berechnen. (Vgl. Murugan und Oblah 2010, S. 46–47)

Bei der TA wird dem Piloten visuell das Wort TRAFFIC in gelber Schrift auf dem Display eingeblendet, zudem wird akustisch dem Piloten die Wörter Traffic, Traffic übertragen. Dieses macht dem Piloten zunächst aufmerksam auf einen entstehen Konflikt, was noch nicht die höchste Warnung entspricht. (Vgl. Murugan und Oblah 2010, S. 46)

Abbildung 1: Visuelle Benachrichtigung (Murugan und Oblah 2010, S. 46)

Die Abbildung 1 zeigt die visuelle Benachrichtigung des TCAS eines möglichen Konfliktes an, welches über die TA ausgelöst wird.
Wenn keine Konfliktminderung eintritt, wird die RA ausgelöst. Diese tritt zwischen 10-15 Sekunden nach der TA auf. Hierbei wird eine Stimme abgespielt, dem den Piloten anleitet zu sinken oder zu steigern und wenn nötig die Geschwindigkeit zu verändern. (Vgl. Murugan und Oblah 2010, S. 46)

Abbildung 2: Bereiche der TA und RA (Murugan und Oblah 2010, S. 47)

In der Abbildung 2 sind zwei Bereiche zu erkennen. Der blaue Bereich beschreibt den Raum, in dem eine TA ausgelöst wird, während der rote Raum das Arial darstellt, indem eine RA ausgelöst wird. Der rote Region wird auch TAU genannt und stellt einen gesonderten Schutzbereich um das eigentliche Luftfahrzeug dar. In diesem Bereich löst das TCAS einen Alarm aus. Dieser Bereich hat einen Schwellwert, der über die Zeit definiert wird. Anders ausgedrückt ist TAU der Bereich, der signalisiert die Zeit-zum-gehen. Dieses wird Closest Point of Approach (CPA) genannt. Der CPA ist somit der Punkt der dichtesten Annäherung. Die Dauer TAU, die Zeit bis der Punkt erreicht ist. Die Zeit-zum-gehen wird berechnet über den Abstand geteilt durch die Annäherungsrate, vertikal sowie horizontal. (Vgl. Murugan und Oblah 2010, S. 46)

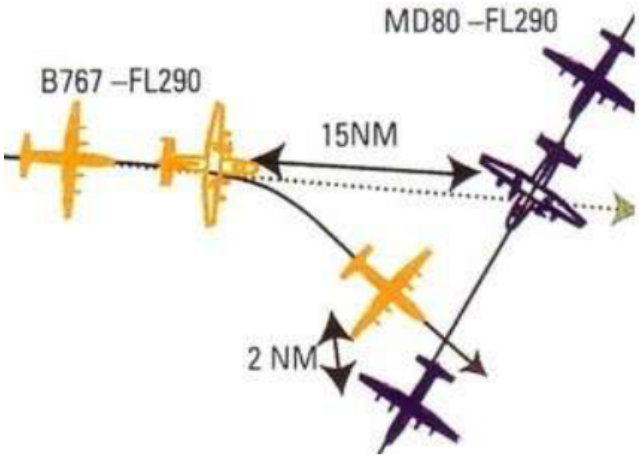


Abbildung 3: Mögliche Ausweichempfehlung (Murugan und Oblah 2010, S. 47)

Die Abbildung 3 zeigt eine Grafik, in der zwei Luftfahrzeuge über das TCAS in dem Schutzbereich RA eine Ausweichempfehlung erhalten.

TCAS Prozesse

Wenn ein anderes Luftfahrzeug in den eigenen Bereich eindringt, werden über Sensoren Zustandsinformationen gesammelt, wie zum Beispiel relative Position und Geschwindigkeit. Diese werden von einem Satz von Algorithmen bearbeitet um eine eventuelle Bedrohung zu bestimmen. Wenn eine mögliche Kollision entstehen könnte, werden die Daten an einen weiteren Algorithmus übergeben, der bestimmen muss, wie weiter zu verfahren ist. Wenn das eingedrungene Luftfahrzeug ebenfalls mit TCAS ausgerüstet ist, wird eine mögliche Lösung mit dem anderen TCAS kommuniziert und koordiniert. Dieses dient zur Sicherstellung der Ausweichmöglichkeiten. Die Piloten haben hierauf die manuelle Kontrolle zu übernehmen und den Empfehlungen des TCAS zu folgen, es sei denn, es würde die Sicherheit gefährden. (Vgl. Kuchar, S. 279)

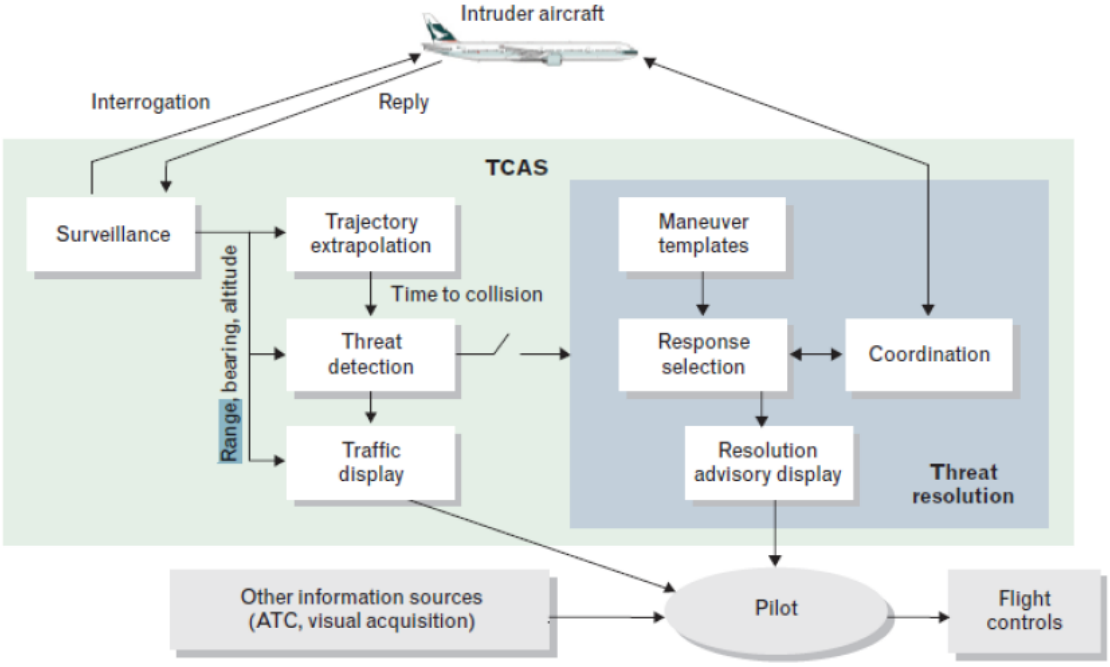


Abbildung 4: Prozesse des TCAS (Kuchar, S. 279)

Die folgende Auflistung beschreiben die Abbildung 4 (Vgl. Kuchar, S. 279–282):

- Surveillance (Überwachung):

  Der Überwachung des Luftraumes erfolgt über die Sensoren von Luft-Luft Abfragen, die einmal pro Sekunde ausgestrahlt werden.

- Threat detection and display (Bedrohungserkennung und Anzeige):

Die Algorithmen von TCAS analysieren die Bedrohung und klassifizieren die Eindringlinge in dem gegebenen Bereich. Das System führt eine Extrapolation durch, um die Position des Flugzeuges in den nächsten Zeitraum einzuschätzen. Dieses wird über die Geschwindigkeit des Luftfahrzeuges durchgeführt. Durch das Durchführen von Schlüsselmetriken wird ausgewertet, ob das eingedrungene Luftfahrzeug eine Bedrohung ist. Zusätzlich werden die vertikalen und schrägen Entfernungen zwischen den Flugzeugen ermittelt. Des Weiteren wird der Zeitpunkt TAU errechnet.

- Maneuver templates (Manövervorlagen):

Wenn die Kriterien für eine Ausgabe eine RA erfüllt ist, werden durch die Algorithmen des TCAS ein geeignetes Manöver bestimmt, um die anstehende Kollision zu vermeiden. Zuerst wird bestimmt, ob das Luftfahrzeug steigen oder sinken muss. Des Weiteren wird die Stärke des RA`s festgelegt. Mit anderen Worten, wie schnell das Flugzeug steigen oder sinken muss. Manöver in der horizontalen Ebene können nicht vorgeschlagen werden, da die Lagegenauigkeit nicht ausreicht.

- Response Selection (Auswahl der Antwort):

Durch das TCAS werden zwei Manöver geprüft. Zum einen das Steigen, zum anderen das Sinken. Jede Berechnung benötigt eine Verzögerung von 5 Sekunden von einer Antwort beginnend. Dazu kommt eine vertikale Beschleunigung von 0,25 g bis einer Geschwindigkeit von 1500 ft/min. Der Algorithmus errechnet dann die größte Trennung zum CPA.

- Resolution advisory display (Anzeige der Lösung):

Das vorgeschlagene Manöver wird dem Piloten über die entsprechende Anzeige visualisiert.

- Coordination:

Das TCAS koordiniert mit dem TCAS aus dem anderen Luftfahrzeug das Ausweichen. Das System kann unter bestimmten Bedingungen die Entscheidung umkehren, von Sinken zu steigen, wenn sich die aktuelle Situation verschlechtert.

Nachteile von TCASII

Das aktuelle TCASII kann nur drei Luftfahrzeuge auf einmal berechnen. Das ist darauf zurückzuführen, dass in der RA die Lösungen vorgeschlagen werden, zu steigen, zu sinken oder nichts zu tun. Das bedeutet, dass nur vertikale Lösungen bereitgestellt werden. Zusätzlich entstehen weitere Probleme bei extremen Höhen oder auch bei Tiefflügen. (Vgl. Murugan und Oblah 2010, S. 49)

Des Weiteren wird eine Kollisionswarnung in einem vorkonfigurierten Bereich abgegeben, der über die Zeit berechnet wird. Der Bereich zwischen den Flugzeugen, der bekannten Position und der Höhe reichen manchmal nicht aus um einen Zusammenstoß zu vermeiden. (Vgl. Murugan und Oblah 2010, S. 49)

Zudem muss jedes Flugzeug seinen Flugplan einreichen. Dieser Flugplan wird in keinem TCAS hinterlegt. Das TCAS ist vollkommen abhängig von dem, was es erkennt und kann nicht auf die Daten der Flugpläne zurückgreifen. (Vgl. Murugan und Oblah 2010, S. 49)

Kollisionsverhütung in der Seefahrt durch MTCAS

Um effizient Kollisionen zu vermeiden werden auf bestimmt navigatorische Mittel zurückgegriffen, die aktuell bzw. noch in der Weiterentwicklung sich befinden. Zum Einem spielt das Automatic Radar Plotting Aid (ARPA) eine Rolle. Diese Plothilfe, die in dem abgedeckten Radarbereich befindlichen Schiffe erkennen und verfolgen werden die Daten auf modernen ausgestatteten Schiffsbrücken zusammengestellt und in einem elektronischen Kartendisplay dargestellt. Das Radar/ARPA stellt grundlegende Bewegungen der Schiffe dar. Unter Voraussetzung eines konstanten Kurses und gleichbleibender Geschwindigkeit wird über eine vektorbasierte Berechnung, Kollisionen vorherbestimmt.

Das automatische Identifikationssystem (AIS), welches sich das Radarsystem zu nutzen macht, übertragen die Daten über das VHF-System. Diese Daten beinhalten Schiffs- und Navigationsdaten, wie Längen- und Breitengrad, Kurs über Boden, Geschwindigkeit über Boden und werden zwischen Schiffen und Uferstationen ausgetauscht. (Vgl. European Navigation Conference und ENC 2016, S. 2)

Das Maritime Traffic Alert und Collision Advoidance System impliziert das Aircorne Collision Avoidance System (ACAS), welches im TCAS umgesetzt wird.

MTCAS ist somit angelehnt an das TCAS aus der Luftfahrt. Dieses System unterstützt die Erkennung und Lösungsfindung bei möglichen Kollisionen. Diese Erkennung wird unter Berücksichtigung des Schiffsverkehrs, Umweltbedingungen und der Regeln und Vorschriften des Schiffsverkehrs getroffen. Das MTCAS trifft im Gegensatz zum TCAS keine Entscheidung über das Ausweichen, unterstützt jedoch die Seemannschaft über eine effiziente Konfliktlösung. (Vgl. European Navigation Conference und ENC 2016, S. 1)

MTCAS dient für ein verbessertes Situationsbewusstsein, zu den kontextsensitiven Vorhersagen, einer N-Trajektorien-Optimierung und tritt in automatischen Verhandlungen von strategischen Manövern. (Vgl. European Navigation Conference und ENC 2016, S. 3–4)

Verbessertes Situationsbewusstsein

MTCAS trägt einen wesentlichen Beitrag zur Verbesserung der Sicherheit und der Effizient. Diese Verbesserung soll über das VTS-System gewährleistet werden. In diesem VTS-System sind die Routen integriert. Diese Daten werden über MMS übertragen. Diese VTS-Technologie ist im MTCAS integriert und dient somit einer Konflikterkennung und einem möglichen Ausweichen. Um eine erforderliche betriebliche Präzision zu erlangen, ist diese Technologie mit einer Integritätsüberwachung erweitert worden und ein Austausch der entsprechenden Schiffsdynamik wird

mit einbezogen. Anstelle von einem Streckenaustausch können auch Trajektorien und Tertiärdaten ausgetauscht werden. Also die Daten, der Bahnkurve, um die sich ein Merkmal bewegt, zum Beispiel der Schwerpunkt eines Körpers. Die Tertiärdaten können Daten über aktuelle lokale Regelungen, Wetterdaten oder auch Informationen über gefährliche Gebiete sein. (Vgl. European Navigation Conference und ENC 2016, S. 3–4)

Kontextsensitiven Vorhersagen

Bei den kontextsensitiven Vorhersagen versteht man das Einfließen der aktuellen Schiffsverkehr-Informationen und der Schiffsdynamik. Das MTCAS berechnet Vorhersagen von Schiffsbewegungen und einer kurzfristigen Entwicklung des Schiffsverkehrs. Dieses impliziert beispielsweise die Topologie der Wasserwege, Schiffsziel und Bathymetrie. Durch das Berücksichtigen der Vorschriften und Regeln, sowie die Informationen des VTS werden auch falsche Alarme reduziert. (Vgl. European Navigation Conference und ENC 2016, S. 4)

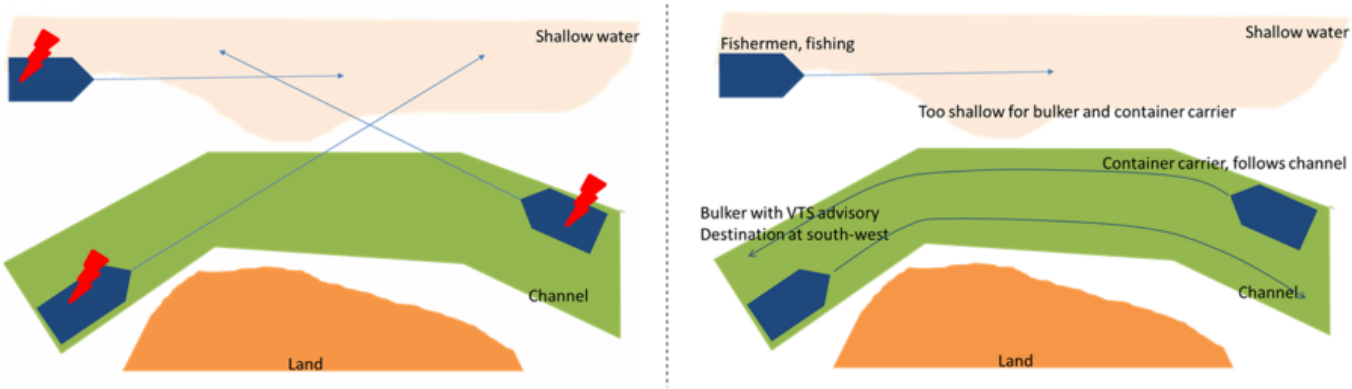


Abbildung 5: Verbesserung der Gefährdungsanalyse (European Navigation Conference und ENC 2016, S. 4)

In der Abbildung 5 sind auf der linken Seite drei Verkehrsteilnehmer zu erkennen. Dieses würde zu drei Warnungen führen. Durch das Verwenden von MTCAS und der Berücksichtigung aller Daten würde hier ein verbessertes Betriebsbild ergeben, welches auf der rechten Seite dargestellt ist und eine eventuelle Fehlermeldung vermeiden. (Vgl. European Navigation Conference und ENC 2016, S. 4)

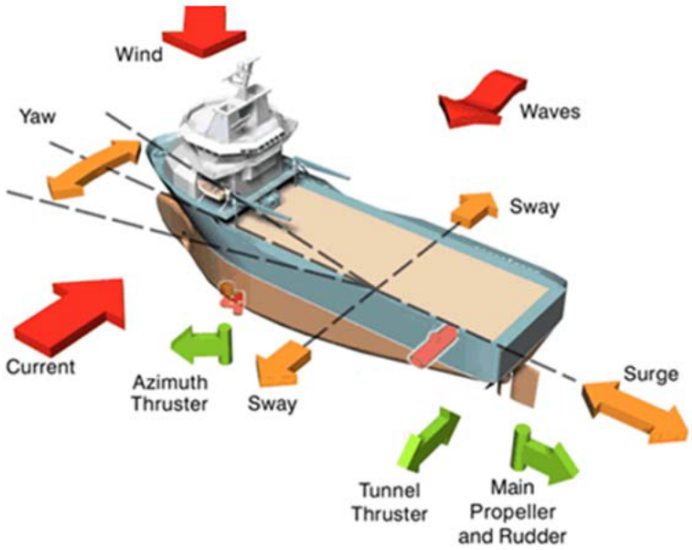


Abbildung 6: einfließende Kräfte im Schiffsmanöver (European Navigation Conference und ENC 2016, S. 4)

Bei dem durchführen eines Wendemanövers müssen verschieden Kräfte, die auf das Schiff Einflussnehmen mit berechnet werden. Diese sind in Abbildung 6 dargestellt. Beispielsweise sind das die Kräfte vom Triebwerk, das eigentliche Gieren, das Schwanken des Schiffes, die Wellen, der Wind und die eigentlichen Kräfte der Schiffsschraube. Diese Vorhersagen werden von Schiff zu Schiff ausgetauscht, gemeinsam koordiniert und angepasst, welches über den gesamten Zeitraum durchgeführt wird. (Vgl. European Navigation Conference und ENC 2016, S. 4)

Diese hydrodynamischen Eigenschaften bestimmen die Manövrierfähigkeiten eines Schiffes und werden also beeinflusst durch die Rumpfform, den Propeller, Motortyp, vorherrschenden Umgebungsbedingungen, wie Wasser, Wellen, Luft und Wind. All diese Kräfte sind Bestandteil des mathematischen Modells und nehmen Einfluss auf die Simulation, wie auch auf die Trajektorienplanung. (Vgl. European Navigation Conference und ENC 2016, S. 3)

N-Trajektorien-Optimierung

In der Schiffsnavigation wird unterschieden zwischen der Reiseplanung mit Wegpunkten und die Planung von taktischen Manövern mit Hilfe von Manövrierpunkten. Bei diesen Punkten muss die Steuerung dementsprechend taktisch angepasst werden. Im Gegensatz zu einer Route beschreibt eine Trajektorie die Bewegung des Schiffes in einem bestimmten Zeitraum, als Beispiel, wo sich das Schiff in den nächsten 5 Minuten

sein wird. (Vgl. European Navigation Conference und ENC 2016, S. 3)

Zu den oben erwähnten Methoden gehören zusätzlich auch kontextabhängige Vorhersagen. Diese werden ebenfalls zur Berechnung von Ausweichmöglichkeiten berechnet. MTCAS hat nicht nur die Möglichkeit eigene, sondern auch die Schiffsmanöver zu berechnen, die das jeweilige andere Schiff durchführen könnte. Über eine Verbindung, speziell einem datalink werden Trajektorien ausgetauscht. Dieses dient zu zur Bestimmung eines sicheren und effizienten Optimums. Schiffe die nicht mit einem MTCAS ausgerüstet sind, werden mit einberechnet und berücksichtigt. Das System befolgt also Regeln, wie Convention on the International Regulations for Preventing Collisions (COLREG). Zudem überwacht das MTCAS die Übereinstimmung mit den jeweiligen ausgehandelten Trajektorien und informiert die Besatzung über wesentliche Abweichungen. Das bedeutet, dass die Crew weiterhing die Verantwortung trägt. Das MTCAS entscheidet nicht über das entsprechende Manöver, sondern schlägt automatische berechnet Ausweichrouten vor. (Vgl. European Navigation Conference und ENC 2016, S. 7)

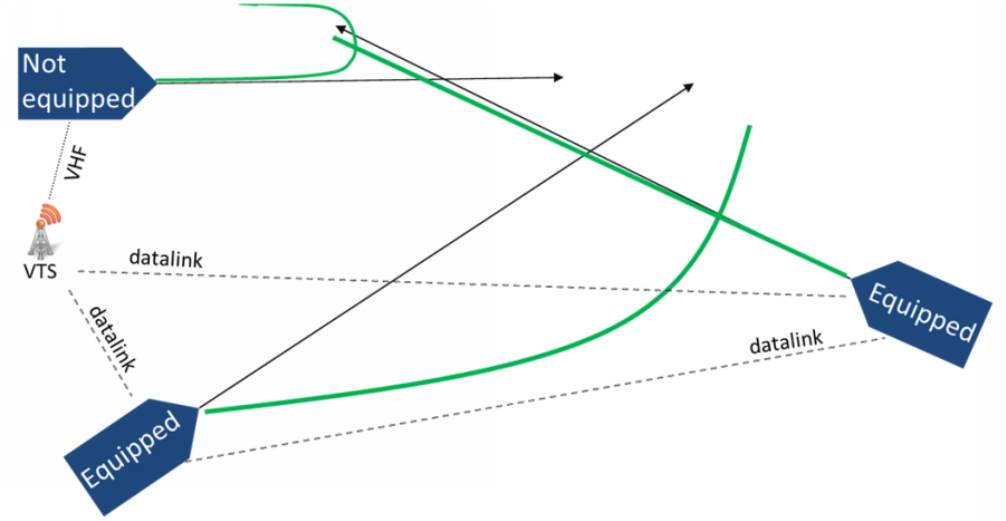


Abbildung 7: Prinzip einer N-Trajektorien-Optimierung (European Navigation Conference und ENC 2016, S. 7)

Die Abbildung 7 stellt eine n-Trajektorie dar, die mit Hilfe des MTCAS-System optimiert wurde. Dabei ist ein Schiff nicht mit dem MTCAS ausgestattet. Weiterhin wird das VTS mit einbezogen. Die schwarzen Pfeile sind die Geschwindigkeitsvektoren und die grünen sind die entsprechenden errechneten Trajektorien. Die hier dargestellte Situation beschreibt einen leichten Kurswechsel, für das Schiff, welches aus Osten kommt. Dieses Manöver würde den Kurs kompensieren, während das aus Westen kommende Schiff, das mit MTCAS ausgerüstet ist, nach Steuerbord korrigiert. Manöver, die jedoch gegen die COLREGs verstoßen werden hierbei nicht empfohlen. (Vgl. European Navigation Conference und ENC 2016, S. 7–8)

Die ausgerüsteten MTCAS Schiffe sind mit dem datalink zur VTS und untereinander verbunden, während das nicht ausgerüstete Schiff über VHF mit der VTS-Station verknüpft ist.

Weiter Methoden zur Vermeidung von Schiffskollisionen

Schon 1972 wurden Regeln zur Vermeidung von Schiffskollisionen festgelegt. Diese internationale Verordnung zur Verhütung von Zusammenstößen auf See (COLREGS) ist jedoch nicht ausreichend um Konflikte zu lösen, da es zu schwierig ist, alle möglichen Bedingungen in Form von Regeln zu beschreiben. Die tatsächliche maritime Umgebung ist zu komplex für diese Verordnung. Diese Verordnung legte fest, dass zur Vermeidung von Kollisionen, die Schiffe auf See, die Navigation der anderen folgt. (Vgl. Kim et al. 2017, S. 2–3)

Um die Besatzung des Schiffes technologisch zu unterstützen, sind weitere Methoden entwickelt worden. Diese sind zu einem die eigene Schiffsdomäne und zum anderen die Ameisenkolonie. Die Schiffsdomäne legt einen eigenen Sicherheitsbereich fest, indem ein Algorithmus den Sicherheitsbereich errechnet, wo das eigene Schiff verhindert, dass andere Schiffe diesen Bereich durchdringen. Die Optimierung einer Ameisenkolonie, als auch einen genetischen Algorithmus dienen, um einen vermeintlichen sicheren Kurs zu identifizieren. Hierbei werden diverse biologische Phänomene nachgeahmt, wie die Futtersuche oder die Kämpfe fürs Überleben. (Vgl. Kim et al. 2017, S. 2–3)

Ein anderer Ansatz zur Kollisionsvermeidung war die Einführung eines Algorithmus, der sicheren Trajektorien bei Begegnung mehrere Schiffe. Diese Funktion berechnet die Art und Weise Schiffe um andere Schiffe, sowie andere Hindernisse zu navigieren. Diese Methode konzentriert sich dabei auf die Einhaltung der COLREGS Regel. Hierbei wurde eine Erweiterung hinzugefügt, die bei der Durchdringung eines Schiffsgebietes, den Unterschied zwischen dem Kurs und der Entfernung von anderen Schiffen festlegt. Jedoch ist bei dieser Methode ein zentralisiertes System von Nöten, wie das VTS. Sind die Schiffe dabei außerhalb des Kontrollbereiches des VTS, wäre das Anwenden schwierig. Das muss die Anzahl der zu begegnenden Schiffe weniger als fünf sein. (Vgl. Kim et al. 2017, S. 2–3)

Durch die Poisson-Verteilung, wird die Wahrscheinlichkeit von mehreren Begegnungen berechnet. Bei diesem Verfahren wird jedoch von einem gleichmäßigen Verkehrsfluss über eine spezifische Spur ausgegangen. Durch die Errechnung der Wahrscheinlichkeit, dass sich Schiffe hierbei kreuzen, werden die Kollisionen versucht zu verhindern. Für die Berechnung der entstehenden Situationen wird der Schiffstyp, Geschwindigkeit, der Domänenradius und der Spurverkehrsfluss als Berechnungsgrundlage genutzt. Weiterhin wurde die Beziehung zwischen den Schiffen, die Manövrierfähigkeit, die veränderbar ist und die Veränderung der Geschwindigkeit mit einbezogen, um das Verfahren zu optimieren. Zusätzlich wird nicht nur das erste Schiff betrachtet, sondern auch das zweite Schiff, welches sich in Gefahr befindet. Diese Methode zur Kollisionsverhütung fokussiert sich darauf, den sicheren Kurs eines Schiffes bei mehreren Begegnungen zu finden, also eine Eins-zu-Viele-Situation. (Vgl. Kim et al. 2017, S. 2–3)

Durch die Nutzung der geplanten Strecken wurde ein Framework namens Collision-Advoidance Negotiation Framework (CANFO) untersucht. Hierbei handelt es sich, um Protokolle die geplanten Strecken kontrollieren. Diese Thematik betrachtet jedoch nur eine Eins-zu-eins-Situation. (Vg . Kim et al. 2017, S. 2–3)

Des Weiteren wurde einen dezentralisierten Algorithmus vorgeschlagen, der die Trajektorien optimiert. Dieser Algorithmus kooperiert teilweise mit anderen Teilnehmern. Die Schiffe, die nicht die Möglichkeit haben, mit anderen Schiffen zu kooperieren, werden mit den Daten des AIS berechnet. Bei diesem Bayes`schen Modell werden die Wahrscheinlichkeiten der geschätzten Position, anderer Schiffe prognostiziert. Durch die anschließenden berechneten Trajektorien sind bei einer Begegnung mit drei Schiffen anwendbar. Allerdings wurde dafür kein explizierter Algorithmus zur Verfügung gestellt. (Vgl. Kim et al. 2017, S. 2–3)

Die vorgestellten Methoden werden dabei als dezentral betrachtet, wenn die Schiffe selbst versuchen, eine bevorstehende Kollision zu verhindern, hierbei aber kein explizites und durchführbares Kommunikationsprotokoll vorgesehen ist. (Vgl. Kim et al. 2017, S. 2–3)

Beispiel der Vermeidung von Schiffskollisionen anhand des Such- und Kontrollverfahrens

Jedes Schiff versucht ihren eigenen Kurs zu entscheiden. Diese Entscheidungen wirken unweigerlich auf die zukünftigen Entscheidungen anderer Schiffe aus und umgekehrt. Durch diese Komplexität müssen Szenarien berücksichtigt werden, die eine Situation von viele-zu-viele abwägt. Dabei muss ein Schiff bei der Modellierung als eigener Agenten behandelt werden, der den nächsten beabsichtigten Kurs mit anderen Schiffen kommuniziert, um einen sicheren Kurs autonom zu finden. (Vgl. Kim et al. 2017, S. 1–2)

Für die Vermeidung von Schiffskollisionen werden zwei Verfahren vorgestellt. Zum einem das Kontrollverfahren und zum anderen das Suchverfahren. (Vgl. Kim et al. 2017, S. 3–4)

Bei dem Kontrollverfahren trifft das Schiff die Entscheidung, ob es zur nächsten Position gehen soll. Erst wenn kein Schiff innerhalb in einer gewissen Distanz ist und auch nicht in der Nähe der Zielposition ist, geht es zur nächsten Position über. (Vgl. Kim et al. 2017, S. 3–4)

Durch das Suchverfahren sollen Kollisionen vermieden werden. Dieses Suchverfahren wird mithilfe eines Algorithmus durchgeführt. Durch ablaufend er Rechenzeit oder aber wenn jedes Schiff kein Risiko findet, aktualisiert sich die nächste Position und der Standort wird gewechselt. Die Rechenzeit hat somit eine Begrenzung, wo die Kommunikation zwischen den Schiffen durchgeführt wird. Sobald die Zeit abgelaufen ist, wird die Position verändert, trotz der Möglichkeit, dass der angegebene Kurs nicht sicher ist, diese aber den geringsten Aufwand haben. Jedes Schiff wechselt zwischen Suche und Kontrolle, bis es an dem Bestimmungsort angekommen ist. (Vgl. Kim et al. 2017, S. 3–4)

Bei der Suche für die optimale Schiffsstrecke, um Kollisionen zu vermeiden, werden zwei Algorithmen verwendet. Zu Einem der Distribution Local Suchalgorithmus (DLSA) und der Tabu-Suchalgorithmus (DTSA). Wenn mehr als zwei Schiffen ein den vorgegebenen Bereich kommen, wird das Verfahren komplexer. Da eine große Menge von Daten verarbeitet werden müssen, nimmt die Berechnung Zeit in Anspruch, welches ein Risiko bei schnellen Entscheidungen ist. (Vgl. Kim et al. 2017, S. 1)

Jedoch um dieses entgegen zu wirken, wird ein stochastischer Suchalgorithmus (DSSA) eingesetzt, der es ermöglicht die Absichten eines Schiffes, sofort nach Erhalt der Nachricht, von den Zielschiffen zu ändern. (Vgl. Kim et al. 2017, S. 1)

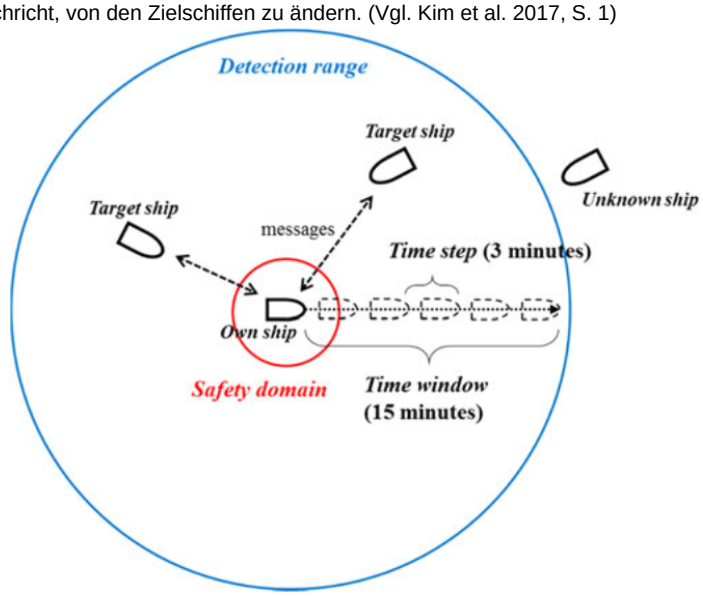


Abbildung 8: Kollisionswarnung (Kim et al. 2017, S. 5)

Die Abbildung 8 beschreibt exemplarisch das Verhalten einer Kollisionswarnung. In der Mitte befindet sich das eigene Schiff. Dieses Schiff hat einen eigenen Erfassungsbereich. In diesem Bereich erkennt es andere Verkehrsteilnehmer und hat die Möglichkeit Daten mit diesen auszutauschen. Mit dem Schiff, das sich außerhalb dieses Erfassungsbereich befindet, können keine Daten über das MTCAS ausgetauscht werden. Jedes Schiff hat einen eigenen Sicherheitsbereich. Die Größe der Sicherheitsbereich variiert je nach Schiffstyp. Wenn der eigene Sicherheitsbereich durchdrungen wird, ist eine Kollision möglich. (Vgl. Kim et al. 2017, S. 4–7)

Annahme: Ein Schiff hat eine Reisegeschwindigkeit mit 12 Knoten. Aufgrund der eingeschränkten Schiffsbewegung kann der Course nicht schnell

geändert werden. Sobald also eine Route gewählt wurde, muss diese für einen gewissen Zeitraum eingehalten werden. In diesem Beispiel beträgt die Einhaltung des Kurs 3 Minuten oder anders ausgedrückt, das Schiff kann/muss die Route alle 3 Minuten ändern. Das entspricht eine Entfernung zwischen 0 und 6 Seemeilen. Das Schiff hat nun ein Zeitfenster von fünf Zeitschritten festgelegt, somit 15 Minuten von der aktuellen Position. Durch die bestehende Kommunikation zwischen den anderen Schiffen werden weiterhin Daten wie Position, Kennung und Geschwindigkeit übertragen. Je größer das Zeitfenster angegeben ist, je länger hat das Schiff eine Sicht auf zukünftige Ereignisse. (Vgl. Kim et al. 2017, S. 4–7)

Auf Basis der aktuellen Position, Schiffsdaten und Geschwindigkeit der Zielschiffe, berechnet ein Schiff den Aufwand für einen sogenannten Kandidatenkurs. Der Kandidatenkurs wird aus Winkeln für wechselnde Kurse ausgewählt. Der Kurswechsel kann unter Berücksichtigung der typischen Schiffsmanöver von -45° auf der Backbordseite bis +45° auf der Steuerbordseite in 5° Schritten verändert werden. (Vgl. Kim et al. 2017, S. 4–7)

Der Time to Closest Point of Approach (TCPA) kann über den Kurs und Geschwindigkeit bestimmt werden. Dieser Punkt beträgt in diesem Beispiel 12 min und ist in der Abbildung 9 zu erkennen. (Vgl. Kim et al. 2017, S. 4–7)

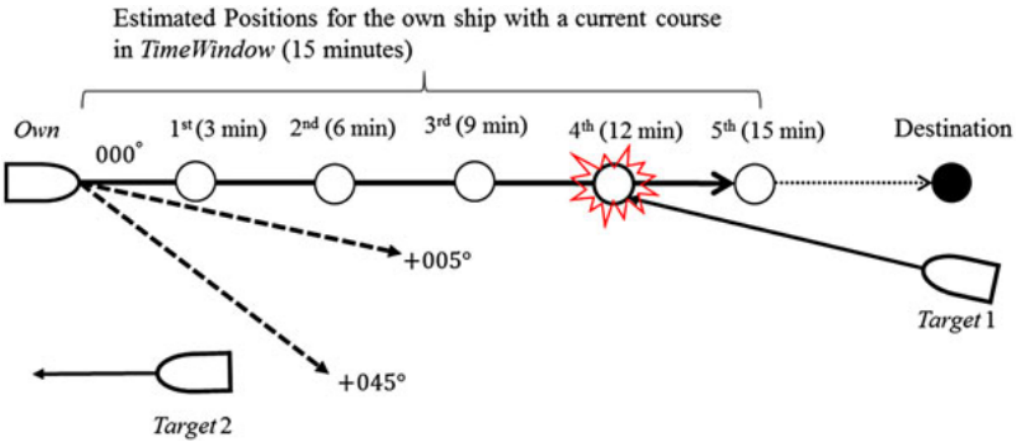


Abbildung 9: Kollisionsentwicklung (Kim et al. 2017, S. 5)

Der Aufwand kann beispielsweise über einen DLSA-Suchalgorithmus errechnet werden. Dabei wird der geringste Aufwand ermittelt, welches Schiff den Kurs um wieviel Grad anpassen sollte. (Vgl. Kim et al. 2017, S. 4–7)

Aktuell Absicht: COST (000) = 15/12 + 0° / 180° = 1.25
Abweichung 45° : COST (045) = 0 +45° / 180° = 0.25
Abweichung 5°:COST (005) = 0 +5° / 180° 0.028
(Vgl. Kim et al. 2017, S. 4–7)

Ausblick

Um Schiffskollisionen zu vermeiden sind verschieden Methoden und Ideen im Laufe der Zeit verfolgt und weiterverwendet worden. Unteranderem das Regelwerk COLREGS, 1972 oder auch die Definition von Wasserstraßen, wo Schiffsgebiete festgelegt werden. (Vgl. Goodwin 1975). Des Weiteren tragen verschiedene Algorithmen im Laufe der Zeit zur Verbesserung der Vermeidung von Kollisionen bei. Nicht nur Algorithmen steigern die Effizienz, auch die Methoden der Statistik fördern die Risikovermeidung. (Vgl. Kim et al. 2017Table 1. Major features of preceding studies on ship collision avoidance) (2017, S. 4)

Um Kollisionen in der Luftfahrt noch weiter zu verbessern, wird das ACAS (Airborne Collision Avoidance System) weiter entwickelt. Das TCAS I/ACAS I unterstützt ausschließlich die Verkehrsinformationen (Traffic Advisory, TA) nachdem Prinzip, Sehen und Ausweichen. Es kann jedoch keine Ausweichempfehlungen (Resolution Advisory, RA) berechnen. Das TCAS II /ACAS II ergänzt die TA mit der RA und empfiehlt eine vertikale Ausweichempfehlung (Seigen oder Sinken). Die anschließende Weiterentwicklung von TCAS III/ACAS III soll eine zusätzliche horizontale Ausweichempfehlung geben können. Jedoch wird dieses noch als technisch schwierig angesehen.
Unter dem Namen TCAS IV/ACAS X wird an einer Weiterentwicklung gearbeitet, die zwischen ACAS II und ACAS III herausgebracht werden soll. Diese wird jedoch nicht vor 2020 erwartet. (Vgl. Bundesministerium für Verkehr und digitale Infrastruktur 2015)

Literaturverzeichnis

Bundesministerium für Verkehr und digitale Infrastruktur (Hg.) (2015): Kollisionswarnsystem (ACAS/TCAS). Online verfügbar unter https://www.forschungsinformationssystem.de/servlet/is/69257/, zuletzt geprüft am 27.04.2017.
European Navigation Conference; ENC (2016): 2016 European Navigation Conference (ENC). Helsinki, Finland, 30 May 2016-2 June 2016. Piscataway, NJ.
goldner: Sudie-DE-803.1-17.docx. Hg. v. Bundesstelle für Flugunfalluntersuchung.
Goodwin, Elisabeth M. (1975): A Statistical Study of Ship Domains. In: *J. Navigation* 28 (03), S. 328–344. DOI: 10.1017/S0373463300041230.
Kim, Donggyun; Hirayama, Katsutoshi; Okimoto, Tenda (2017): Distributed Stochastic Search Algorithm for Multi-ship Encounter Situations.
Kuchar, James K.: The Traffic Alert and Collision Avoidance System.

Lufthansa Technik (Hg.): TCAS: Sicherheit während des Fluges. "Elektronischer Schutzschild" für jede Lufthansa-Maschine. Online verfügbar unter https://www.lufthansa-technik.com/de/collision-warning-system, zuletzt geprüft am 24.04.2017.
Murugan, Sathyan; Oblah, Aniruth A. (2010): TCAS Functioning and Enhancements (8).

Präsentation