Projektgruppe MAPS

Abschlussdokumentation

23.10.2014

Carl von Ossietzky Universität Oldenburg
Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Abteilung: Systemanalyse und -optimierung
Verantwortlicher: Prof. Dr.-Ing. Axel Hahn

Betreuer

- Prof. Dr.-Ing. Axel Hahn
- Christoph Dibbern
- Sascha Hornauer
- Sören Schweigert

Gruppenmitglieder (alphabetisch)

- Frank Gröger
- Hendrik Müller
- Hewad Osmani
- Wiebke Osmers
- Stephan Robbers
- Patrick Schäfer
- Peter Schmeißer
- Christoph Schmitt
- Kerem Sevimli
- Peter Stedeler

Inhaltsverzeichnis

I Abbildungsverzeichnis

II Tabellenverzeichnis

III Kurzfassung

Das vorliegende Dokument stellt die Abschlussdokumentation der Projektgruppe Manöverplanung und Simulation (PG MAPS) dar, die in der Abteilung Systemanalyse und -optimierung von Herrn Prof. Dr.-Ing. Axel Hahn am Department für Informatik der Carl von Ossietzky Universität Oldenburg von Oktober 2013 bis Oktober 2014 durchgeführt wurde. Die Projektgruppe hat bei der Planung und Entwicklung einer verteilten Simulation von zwei Simulatoren sowie weiteren Software-Komponenten mitgeholfen. Die Haupt-Komponenten sind eine Simulation für den Schiffsverkehr mit dem Arbeitstitel Maritime Traffic Simulation (MTS), ein Umweltsimulator zur Generierung von Umweltszenarien für die MTS sowie eine Analyse-Komponente. Mit Hilfe letzterer Komponente können die durch die MTS generierten Schiffsverkehrsdaten aufgegriffen, analysiert und auf Fragestellungen hin untersucht werden, z.B. hinsichtlich der Effizienz des Schiffverkehrs und dem Kollisionsrisiko von Schiffen auf bestimmten Schifffahrtsstraßen.

Ein erhöhtes Verkehrsaufkommen von Container- und Passagierschiffen auf den Wasserstraßen macht es unausweichlich, den Schiffverkehr effizienter und sicherer zu gestalten. Die Simulation des Schiffverkehrs erlaubt es, Verkehrsszenarien zu analysieren, die unter realen Bedingungen kostenintensiv und mit großem Sicherheitsrisiko behaftet gewesen wären.

Die Arbeit der PG MAPS war darauf ausgerichtet, eine Umgebung von verteilten Co-Simulatoren zu entwickeln. Ziel war es, dass jeder einzelne Simulator eine inhaltlich geschlossene Einheit darstellt, die eine konkrete Aufgabestellung zu bearbeiten hat (wie sie im ersten Absatz dieses Kapitels für die drei Hauptkomponenten genannt wurden). In der verteilten Simulation stellt jede Haupt-Komponente einen Federate dar, der über eine Laufzeitinfrastruktur (RTI, Runtime Infrastructure) im High-Level-Architecture- (HLA-) Standard mit anderen Federates Daten austauscht. Durch diesen architektonischen Aufbau wurde dem Erweiterungsgedanken Rechnung getragen, denn HLA ermöglicht es, weitere Komponenten als Federates anbinden zu können.

Die MTS als Kern-Komponente der verteilten Simulation ist eine diskrete Ereignissimulation, welche Schiffsverkehrsdaten generiert, simuliert und visualisiert. Es wird auf das bestehende Toolkit des Java-Simulations-Frameworks MASON zurückgegriffen. Dieses bietet grundlegende Konzepte und Architekturen für den weiteren Aufbau der MTS an, wie z.B. den Scheduler, der den Simulationsablauf und die Ereignisse verwaltet oder die Standard-MASON-Benutzeroberfläche (GUI, Graphical User Interface), die der PG als Vorlage dient. In der MTS ist das physikalische Modell enthalten, welches die Fahrdynamik der Schiffe definiert. Weiterhin wird hier die Logik der Schiffe festgelegt. Dies geschieht zum einen anhand unterschiedlicher Agententypen (Agents), die sich hinsichtlich der Steuerung der Schiffe unterscheiden. Zum anderen legen Verhaltenslogiken (Behaviors) das Verhalten der Schiffe in bestimmten Situationen fest, z.B. beim Überholen oder in Gefahrensituationen, wie kurz vor Kollisionen. Die MTS umfasst zudem eine, die Standard-MASON-GUI erweiternde Visualisierung, die für die Simulationssteuerung sowie für die grafische Abbildung der Simulation notwendig ist.

Die zweite Haupt-Komponente der verteilten Simulation ist der Umweltsimulator. Dieser stellt aufgezeichnete Umweltdaten von der Nord- und Ostsee zur Verfügung, die entweder aus Datenbankdateien oder In-Memory-Datenbanken ausgelesen oder über eine bereitgestellte GUI durch den Nutzer selbst definiert werden können. Innerhalb der PG MAPS wurde der Einfluss von Strömungs-, Wind-, und Tidendaten in der Simulation implementiert. Es gibt

verschiedene Umweltszenarien, die sich im Hinblick auf die Art und Weise, wie die Umwelteinflüsse auf die Simulation und die Schiffsagenten einwirken, unterscheiden. Z.B. kann eingestellt werden, ob sich die Umweltdaten während des Simulationslaufs ändern sollen (dynamisches Umweltszenario) oder nicht (statisches Umweltszenario).

Zur Aufbereitung und Auswertung von Daten, die während der Simulationsläufe der MTS generiert werden, gibt die PG MAPS dem Nutzer eine Analyse-Komponente an die Hand, die die dritte Haupt-Komponente der verteilten Simulation darstellt. Mit deren Hilfe können Risiko- und Effizienzanalysen für Schifffahrtsstraßen durchgeführt werden. Sie erlaubt die Abfrage der Zustände von MTS-Entitäten mit Hilfe von Queries, die auf die von der MTS an die Analyse-Komponente gesendeten Daten angewendet werden. Die Ergebnisse werden an einen Exporter weitergeleitet, der diese in ein gewünschtes Dateiformat wandelt – von der PG MAPS wurde bisher die Möglichkeit des CSV-Exports implementiert.

Zusätzliche Hilfs-Komponenten, die wie die drei Haupt-Komponenten als Federates an die Laufzeitinfrastruktur im HLA-Standard angebunden sind, runden die durch die PG MAPS geleistete Entwicklungsarbeit ab. Darunter fällt z.B. eine Komponente, die das aktive Steuern von Schiffen der Simulation per Tastatur oder GUI zulässt. Eine weitere ermöglicht das Einlesen von Schiffsrouten im CSV-Format, um die Schiffsagenten der MTS diese Routen abfahren zu lassen.

Die Anbindung weiterer externer Komponenten, wie einer Kollisionsvermeidungs-Komponente und einer Sensordatensimulation, die im universitären Umfeld entwickelt werden, wurde bereits zur Laufzeit der PG realisiert.

1. Einleitung

1.1 Motivation

In den letzten Jahrzehnten ist das Volumen des weltweiten Güter- und Passagiertransports stark angestiegen. Somit hat auch der Schiffverkehr auf offener See, auf befahrbaren Flüssen und in den Küstengebieten zugenommen, was zur Folge hat, dass dessen Ablauf in Zukunft effizienter und sicherer gestaltet werden muss, um weiterhin einen reibungslosen Verkehrsablauf zu ermöglichen.

Der Einsatz von Simulationswerkzeugen erlaubt es, realitätsnahe Verkehrssituationen virtuell – unter Einsatz von Informationstechnologien – zu analysieren und zu testen. So können bestimmte, in der Realität der Schifffahrt auftretende Situationen besser verstanden und „Best Practice"- Handlungsanweisungen erarbeitet werden, wie z.B. bei einer Kolonnenfahrt.

Die Projektgruppe Manöverplanung und Simulation (PG MAPS) reiht sich ein in das wissenschaftliche Projektumfeld der Universität Oldenburg im Bereich maritimer Forschungsarbeiten, innerhalb dessen bereits andere Projektgruppen durchgeführt wurden (PG Marine Observation Platform for Surfaces, MOPS). Daneben existieren auch Dissertationen in dem Forschungsumfeld, wie „Virtual Waterway, eine Simulationsumgebung für Verkehrsabläufe auf Binnenwasserstraßen" von Jörn Beschnidt [Be10].

Ferner sollen die Projektarbeiten der PG MAPS auch zukünftig fortgeführt werden. Es sind weitere PGs geplant, die sich mit der Fortentwicklung der MTS, des Umweltsimulators, der Analyse-Komponente sowie des Scenario Importer beschäftigen.

1.2 Zielsetzung

Im Rahmen der Projektgruppe MAPS haben die Gruppenmitglieder innerhalb des Zeitraums von Mitte Oktober 2013 bis Mitte Oktober 2014 an der Planung und Entwicklung einer Verkehrssimulation für maritime Transportsysteme, eines Umweltsimulators, eines Scenario Importer sowie einer Analyse-Komponente mitgewirkt.

Die Verkehrssimulation, Maritime Traffic Simulation (MTS) genannt, soll innerhalb von Forschungsaktivitäten des Lehrstuhls von Prof. Dr.-Ing. Axel Hahn Verwendung finden. Diese Aktivitäten haben zu einem großen Teil die Optimierung der Verkehrs- und Manöverplanung von Schiffen und Risikountersuchungen von Seewegen zum Gegenstand. Einige spezielle Forschungsfragen, die mit Hilfe der Simulation beantwortet werden sollen, wurden zu Beginn der Projektgruppe exemplarisch vorgestellt:

- Wie viele Schiffe können Gewässer wie die Weser gleichzeitig passieren?
- Wo sind Gefahren wie Schiffskollisionen besonders hoch?

Innerhalb der Verkehrssimulation sollen sich Schiffe als aktive und reaktive Agenten autonom auf Gewässern einer beliebig einlesbaren Seekarte fortbewegen können. Auch das Setzen von Umweltparametern wie Strömung und Wind, die das Verhalten der Agenten beeinflussen, soll möglich gemacht werden. Zudem soll eine Umgebung geschaffen werden, die es ermöglicht, Fragestellungen, wie die in der obigen Aufzählung, beantworten zu können.

Nach eingehender Planung ist aufbauend auf einem Dokument zur Systembeschreibung und -spezifikation eine Architekturumgebung entstanden, die auf oberster Ebene eine Einteilung in die drei folgenden Haupt-Architektur-Komponenten vorsieht:

- Maritime Traffic Simulation (MTS): Umfasst die eigentliche Kern-Komponente der Simulation zur Schiffverkehrsdaten-Generierung, Agentensteuerung und Definition der physikalischen Eigenschaften der Schiffe sowie ihre grafische Benutzeroberfläche (GUI, Graphical User Interface).
- Umweltsimulator: Bereitstellung von Umweltszenarien für die MTS mit Umwelt-Faktoren wie Wind und Strömung (inkl. GUI).
- Analyse-Komponente: Zur Durchführung und Auswertung von Abfragen bestimmter Kennzahlen auf die Simulationsdurchläufe der MTS (inkl. GUI).

Das Hauptaugenmerk bei der Entwicklung lag auf der Erweiterbarkeit der der drei Haupt-Architektur-Komponenten sowie der Möglichkeit der Anbindung externer Komponenten zur Steuerung der Simulatoren. So wurden das Multiagenten-Simulations-Framework MASON und das semantische Datenmodell der Simulationsumgebung des OFFIS Oldenburg – HAGGIS-Datenmodell – verwendet und erweitert.

Es bestand weiterhin die Anforderung, der MTS eine Benutzeroberfläche mitzugeben, die es einem PC-Nutzer ohne tiefgehende Anwender- und Programmiererfahrung erlaubt, Simulationsszenarios aufzusetzen, durchzuführen und auszuwerten. Dem Anwender wurde zudem eine Benutzeroberfläche zur manuellen Steuerung eines Eigenschiffes auf Gewässern wie der Elbe zur Verfügung gestellt. Eine weitere benutzeroberflächenbezogene Funktion ist die grafische Darstellung von statistischen Kennzahlen, die bei der Durchführung von Simulationsszenarien generiert werden.

Neben den funktionalen Anforderungen bestanden weitere nicht-funktionale Anforderungen, wie an Robustheit, Performance, saubere Programmierung sowie Dokumentation der Simulation.

1.3 Aufbau der Dokumentation

Das vorliegende Abschlussdokument ist auf oberster Ebene gegliedert in die folgenden Kapitel:

- Kapitel 2 gibt einen Überblick über die Systemumgebung und -architektur sowie genauere Beschreibungen der einzelnen Architektur-Komponenten der MTS (Kapitel 2.1), die Einordnung der MTS in das HAGGIS-Datenmodell (Kapitel 2.2), eine Erklärung der einzelnen Komponenten der Simulations-Benutzeroberfläche (Kapitel 2.3) sowie eine aufgabenbezogene Auflistung der Funktionen der Simulationssoftware (Kapitel 2.4).
- Kapitel 3 richtet sich speziell an zukünftige Entwickler, die die Erweiterung der MTS im Blick haben. Zunächst wird erklärt, wie die Entwicklungsumgebung zur Erweiterung der MTS eingerichtet wird (Kapitel 3.1). In Kapitel 3.2 wird im Detail auf die Laufzeitinfrastruktur der verteilten Simulation eingegangen, die basierend auf dem High-Level-Architecture- (HLA-) Standard kommuniziert. In Kapitel 3.3 wird schließlich erläutert, wie die einzelnen Architektur-Komponenten der MTS erweitert werden können.
- In Kapitel 4 wird dargelegt, wie die MTS getestet wurde. Zunächst wird das allgemeine Test-Vorgehen bzw. -konzept beschrieben (Kapitel 4.1), ehe die eingesetzten Test-Verfahren – Unit-Tests (Kapitel 4.2), Integrationstests (Kapitel 4.3), Funktionstest der Analyse-Komponente (Kapitel 4.4) und ein Test auf Realitätsnähe mit Hilfe von AIS-

Daten (Daten, u.a. zur Identifikation und Position von Schiffen) (Kapitel 4.5) – skizziert werden.

- Kapitel 5 behandelt projektbezogene, organisatorische Themen der PG MAPS. Zunächst wird die Einteilung der PG Mitglieder in die organisationale und Entwickler-Teams erklärt (Kapitel 5.1.1), im Anschluss wird ein Überblick über die zeitliche Einteilung der Tätigkeiten in Milestones gegeben (Kapitel 5.1.2). Zum Schluss werden die im PG-Umfeld verwendeten Projektmanagement- und Entwicklertools vorgestellt (Kapitel 5.2).
- Abschließend werden in Kapitel 6 noch ein Ausblick (Kapitel 6.3), die Rückbetrachtung der PG (Kapitel 6.1) und die Lessons Learned (Kapitel 6.2) abgegeben.

2. Systemdokumentation

In diesem Kapitel werden – zunächst im Überblick, später im Detail – die System- und Architektur-Komponenten vorgestellt, die die Säulen der durch die PG MAPS entwickelten verteilten Simulation ausmachen und die sich auf oberster Ebene in MTS, Umweltsimulator und Analyse-Komponente einteilen lassen. Danach wird erläutert, wie sich die Simulation in das HAGGIS-Datenmodell einordnen lässt. Eine Beschreibung der Benutzeroberfläche sowie aller Funktionalitäten, die die Software-Komponenten bereithalten, runden das Kapitel ab.

2.1 Systemarchitektur

Das Kapitel Systemarchitektur gibt zunächst einen Überblick über die Gesamtarchitektur der in der PG MAPS entwickelten verteilten Simulation. In Abbildung 1 sind die im Rahmen der PG MAPS entwickelten Komponenten und ihre Beziehungen zueinander überblicksartig dargestellt. Alle Komponenten kommunizieren über die CERTI-RTI (Runtime Infrastructure, deutsch: Laufzeitinfrastruktur), welche HLA als Kommunikationsstandard verwendet. CERTI ist eine Open Source RTI, die den HLA-Kommunikationsstandard implementiert, selbst aber nicht Bestandteil des Standards ist (weitere Informationen zu HLA sind in Kapitel 3.2 nachzulesen).

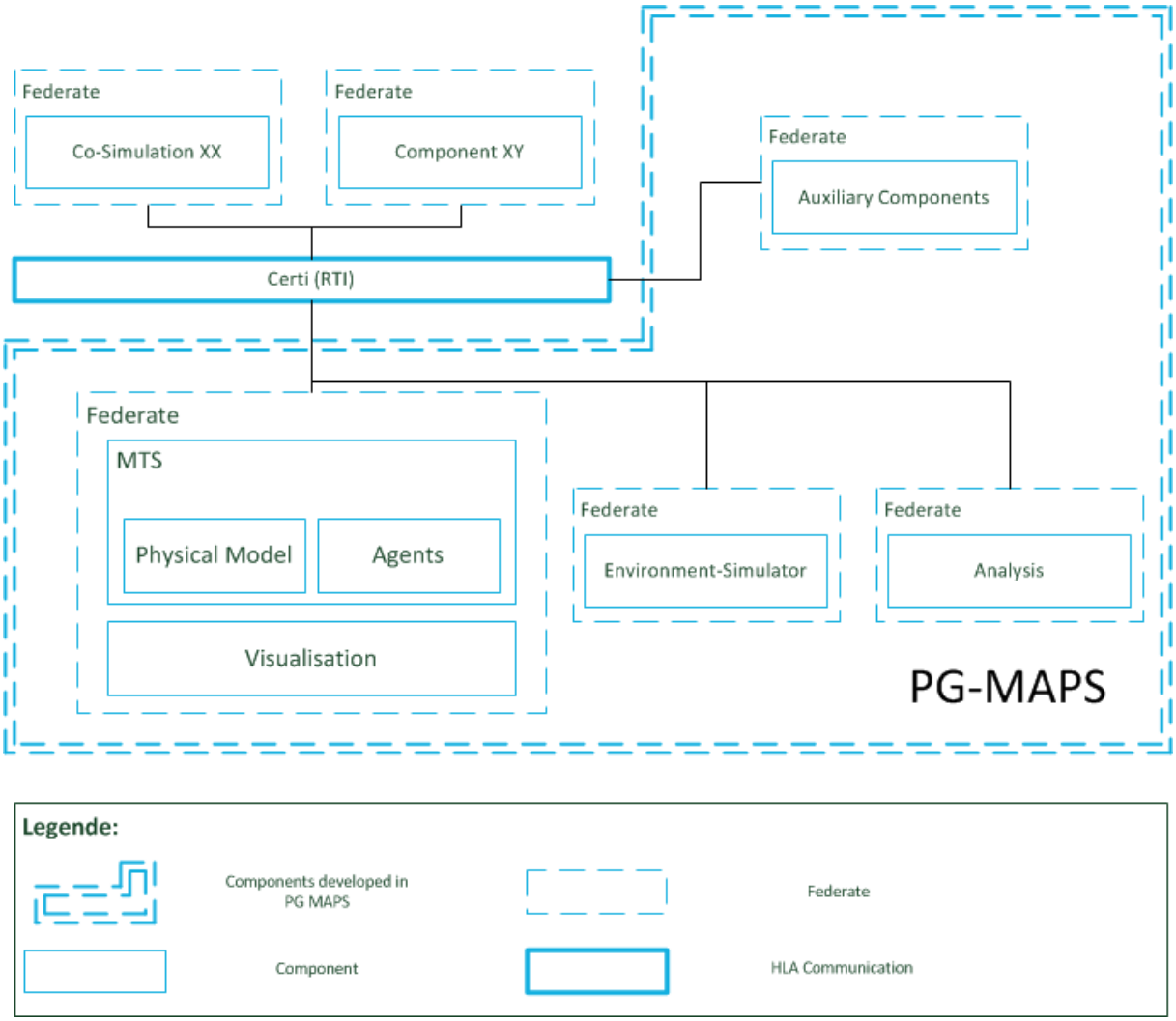

Abbildung 1: Grobarchitektur der von der PG MAPS entwickelten Komponenten

Im folgenden Kapitel (2.1.1) werden die einzelnen Komponenten der obigen Abbildung beschrieben und erläutert, wie sie miteinander in Verbindung stehen. Näher im Detail werden die Komponenten und ihre Funktionen innerhalb der verteilten Simulation in den darauffolgenden Kapiteln (2.1.2 bis 2.1.5) beleuchtet.

2.1.1 Komponentenüberblick

Der Komponentenüberblick zeigt die Beziehungen der im Rahmen der PG MAPS entwickelten Komponenten sowie deren Aufgabe in der verteilten Simulation. Neben der Maritime Traffic Simulation (MTS) wurden ein Umweltsimulator, eine Analyse-Komponente sowie einige weitere Hilfs-Komponenten entwickelt (vgl. Abbildung 2).

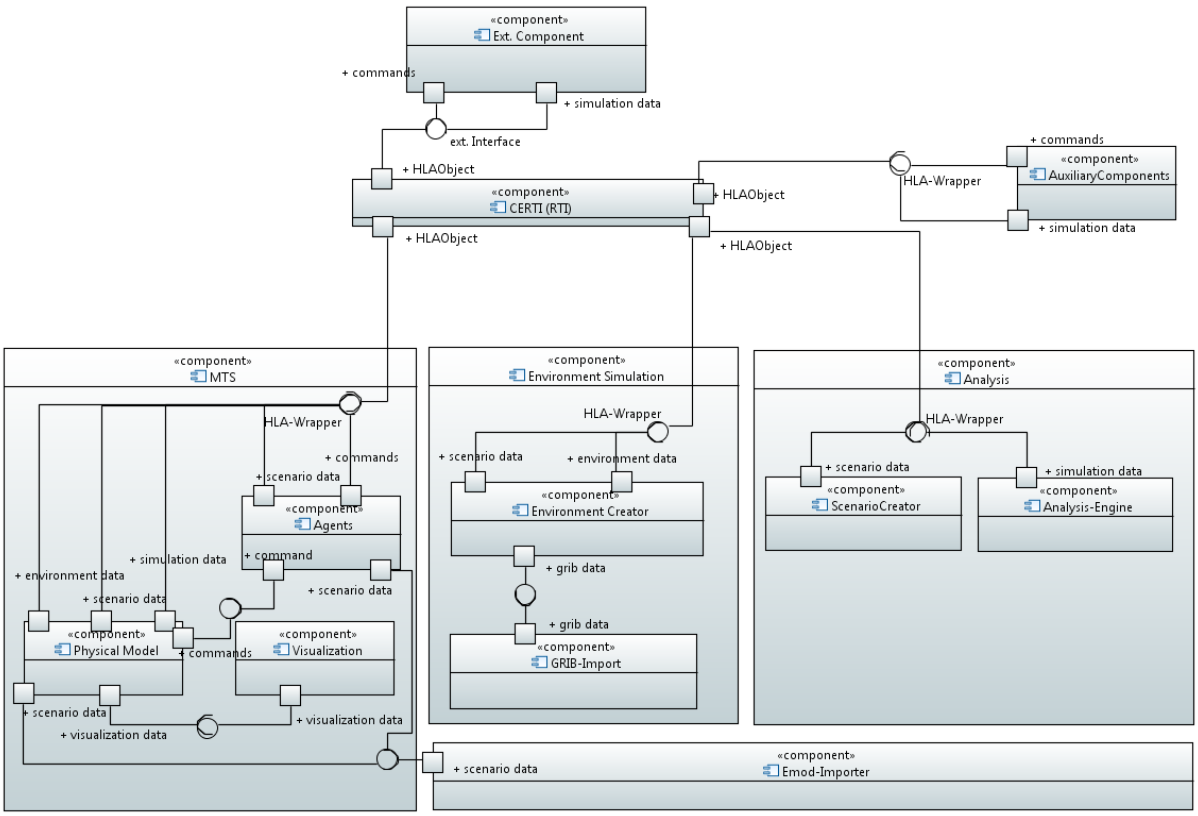

Abbildung 2: Komponentendiagramm

CERTI-Laufzeitinfrastruktur und HLA

Die CERTI- High Level Architecture- (HLA-) Komponente ist die Laufzeitinfrastruktur, die die Kommunikation sowie Zeitsynchronisation zwischen den einzelnen Simulatoren der verteilten Simulation regelt. Über die CERTI-Laufzeitinfrastruktur kommunizieren die einzelnen Simulationen, basierend auf dem HLA-Standard. Die einzelnen Simulatoren werden als eigenständige Federates entwickelt, deren Kommunikation über ein Shared-Memory-Konzept erfolgt (vgl. Kapitel 3.2). Die für den Datenaustausch sowie zur Sicherstellung der Semantik notwendigen Konzepte und Datentypen, welche von den Simulatoren genutzt werden, sind in einem einheitlichen semantischen Datenmodell, dem HAGGIS-Datenmodell zusammengefasst (vgl. Kapitel 2.2).

Maritime Traffic Simulation (MTS)

Die MTS ist der Kern der verteilten Simulation, der die für die Berechnung der Schiffsbewegungen verantwortlichen Komponenten kapselt. Diese Komponenten berücksichtigen sowohl die Steuereinheiten der Schiffe als auch die Schiffsphysik inklusive der die Schiffsbewegung beeinflussenden Umwelteinflüsse: Wind und Strömung. Die Simulation stellt im HLA-Kontext einen eigenen Federate dar, der den Teil-Komponenten die Kommunikation über die CERTI-Laufzeitinfrastruktur ermöglicht. Über HLA kann die Simulation zudem gestartet und gestoppt werden.

Die Simulation lässt sich in die folgenden drei Teil-Komponenten unterteilen:

- **Physical Model:** Das Physical Model ist die zentrale Komponente der MTS. Mit ihr werden die Berechnungen der Schiffsdynamik unter Berücksichtigung aller relevanten Einflüsse durchgeführt.
  Zu Beginn der Simulation wird die im Physical Model befindliche Simulationsumwelt gemäß den über HLA erhaltenen Szenario-Daten aufgesetzt. Diese enthalten Informationen über die gegenwärtigen Umwelteinflüsse sowie die zu simulierenden Schiffe. Das Aufsetzen der Simulation geschieht in der Regel über EMod-Dateien (Dateien, die Informationen über ein Simulationsszenario in XML-Form bereithalten).
  Während des Simulationslaufs werden den anderen Simulationen über die CERTI-Laufzeitinfrastruktur regelmäßig die Simulationsdaten zugänglich gemacht. Unter Simulationsdaten sind alle Daten zu verstehen die im HAGGIS-Datenmodells definiert und im Thumper abgebildet sind. Der Thumper ist für die Erstellung der für die verteilte HAGGIS Simulation benötigten Dateien der einzelnen Federates zuständig. Näheres zum Physical Model ist in Kapitel 2.1.3.2 nachzulesen.
- **Agents:** In der Agenten-Komponente werden die Agententypen mit ihren unterschiedlichen Ausprägungen definiert. Die Agententypen unterscheiden sich vor allem hinsichtlich ihres Verhaltens bei der Steuerung der Schiffe.
  Analog zum Physical Model werden die Agententypen, welche zum Steuern der Schiffe genutzt werden, zu Beginn der Simulation über EMod-Dateien durch Angabe eines Agententyps aufgesetzt.
  Während des Simulationslaufs können einige Agententypen über HLA Steuerbefehle erhalten, diese verarbeiten und das Schiff dementsprechend steuern. Hierzu ist notwendig, mit dem Physical Model zu kommunizieren und die jeweiligen Steuerbefehle über geeignete Aktoren an die Schiffsentitäten weiterzureichen. Die Steuerbefehle werden in Form von Routen oder konkreten Steuerbefehlen, die zu einem bestimmten Zeitpunkt das Zielruder und die Zielgeschwindigkeit bestimmen, übermittelt. Weitere Informationen hierzu finden sich in Kapitel 2.1.3.3.
- **Visualization:** Die Visualisierung bildet den aktuellen Zustand der Simulations-Komponente grafisch ab. Auf der grafischen Benutzeroberfläche werden zudem die Eigenschaften der simulierten Schiffe wie Name, aktuelle Position und Geschwindigkeit abgebildet. Näheres hierzu ist in den Kapiteln 2.1 und 2.3 nachzulesen.

Environment Simulator (Umweltsimulator)

Der Umweltsimulator stellt ebenfalls einen eigenständigen Federate der verteilten Simulation dar. Er stellt der MTS Umwelteinflüsse zur Verfügung, die durch verschiedene Umweltszenarien generiert werden. Eine detaillierte Beschreibung der einzelnen Szenarien sowie eine Beschreibung zur Benutzung der Komponente können den Kapiteln 2.1.4 und 2.4.9 entnommen werden.

Analysis (Analyse-Komponente)

Die Analyse-Komponente ist ein Federate der CERTI-Laufzeitinfrastruktur, der die Struktur einer Abfragelogik vorgibt, um auf diese Weise verschiedene (insbesondere Effizienz- und Risiko-) Analysen auf Basis der Simulationsdaten durchführen zu können. Während des gesamten Simulationslaufs empfängt die Analysis-Engine die von der MTS bereitgestellten Daten und wertet diese anhand der Kennzahlen, welche in dem aktuell angewendeten Analyse-Szenario definiert sind, aus. Die Ergebnisse der Analyse werden aufbereitet und

dem Benutzer anschließend in Form von CSV-Dateien zur Verfügung gestellt. Weitere Informationen über den Aufbau der Analyse-Komponente können dem Kapitel 2.1.5 entnommen werden. Die Durchführung eines Analyseszenarios ist beispielhaft in Kapitel 2.4.10 beschrieben.

Auxiliary Components (Hilfs-Komponenten)

Die Hilfs-Komponenten stellen für sich jeweils einen eigenständigen Federate in der verteilten Simulation dar. Sie unterstützen die MTS bzw. ermöglichen ihr die Nutzung weiterer Funktionen. Nachfolgend werden die einzelnen Hilfs-Komponenten kurz beschrieben:

- **Steering Agent Control:** Der Steering Agent Control Federate dient dazu, einen speziellen Agenten der Simulation, den Steering Agent, über die Tastatur oder eine mobile Brücke zu steuern (vgl. Kapitel 2.4.2).
- **Realtime Synchronizer:** Der Realtime Synchronizer (Realzeitsimulator) kann der verteilen Simulation optional zugeschaltet werden und stellt sicher, dass die simulierte Zeit simultan zu der in der Realität ablaufenden Zeit abläuft. Die konkrete Funktionsweise des Realtime Synchronizer kann dem Kapitel 2.4.3 entnommen werden.
- **Marine Traffic Reader:** Der Marine Traffic Reader (die MarineTraffic-Schnittstelle) ermöglicht das Einlesen von Schiffsrouten im CSV-Format. Diese Schiffsrouten stammen in der Regel aus Messungen von real existierenden Schiffen (vgl. Kapitel 2.4.4). Auf Basis dieser Schiffsrouten können Schiffe in der Simulation generiert werden, um so z.B. die Genauigkeit der MTS zu evaluieren (Der durchgeführte Test auf Realitätsnähe ist in Kapitel 4.5 zu finden).
- **NMEA Reader:** Der NMEA Reader (NMEA = von der National Marine Electronics Association festgelegter Standard zur Kommunikation von Navigationsgeräten auf Schiffen) stellt die Funktion bereit, Daten aus einem Schiffsführungssimulator über das User Datagram Protocol (UDP) zu empfangen oder vom Dateisystem zu lesen und entsprechend in HLA-Befehle für die Simulation umzuwandeln (vgl. Kapitel 2.4.5).

External Components (Externe Komponenten)

Die externen Komponenten stehen stellvertretend für weitere an der verteilten Simulation beteiligte Simulatoren und Komponenten, die mit der MTS oder anderen im Rahmen der PG MAPS entwickelten Simulatoren zusammenarbeiten können. Beispiele hierfür sind das Steuerungswerkzeug Distributed Controlling Toolkit (DistriCT), die Kollisionsvermeidungs-Komponente oder die Sensordatensimulation.

2.1.2 Scenario Importer

Ein Szenario beschreibt zu Beginn des Simulationslaufs alle möglichen physikalischen Objekte – z.B. Schiffe mit ihren unterschiedlichen Ausprägungen – und weitere Eigenschaften, wie z.B. eine Route in Form von Wegpunkten. Das Szenario wird über EMod-Dateien beschrieben. EMod ist ein XML-basiertes Format, in dem die Objekte bzw. Instanzen persistiert sind, die auf der Struktur sowie den Konzepten und Datentypen des HAGGIS-Datenmodells basieren. Da für die Abbildung des Datenmodells das EMF (Eclipse

Modeling Framework [Th14]) bzw. Ecore-Modelle genutzt werden, kann der Import über die Funktionalität des Frameworks geschehen. Auch bei Erweiterungen des HAGGIS-Datenmodells sind hier keine Anpassungen notwendig.

Das ursprüngliche Szenario kann während der Laufzeit nicht mehr geändert werden. Das Scenario wird durch die EMod-Datei beschrieben, die nicht mehr geändert werden kann. Die Simulation kann lediglich pausiert, gestoppt und neugestartet werden. Damit es neugestartet werden kann, wird zwischen Szenario- und Laufzeitdaten unterschieden. Bei Start der Simulation werden die Szenariodaten (EMod) in die Laufzeitdaten kopiert. Die Laufzeitdaten werden durch die Simulation verändert. Weiterhin hat der Nutzer die Möglichkeit die Zerstörung von Schiffen zu induzieren und somit während der Simulation Einfluss zu nehmen.

2.1.3 Maritime Traffic Simulation

2.1.3.1 Architektur der MTS

In diesem Kapitel wird die Architektur der MTS anhand eines Klassendiagramms erläutert. Für die Simulation wird MASON, ein Java-basiertes, domänenunabhängiges Simulationsframework genutzt [Ge14]. Weiterhin wird die Erweiterung GeoMason genutzt, die Funktionen für geometrische Operationen anbietet und die Anzeige von Geometrien vereinfacht [CCW13]. Für die geometrischen Operationen nutzt GeoMason die Java-Bibliothek von JTS [Vi06].

In Abbildung 3 ist die Architektur der MTS in Form eines Klassendiagramms dargestellt. Als Basis dient dabei der Standard MASON-Aufbau. Dieser wird um den Ansatz des Entity-Component-Systems-Pattern (ECS) [Wid14] erweitert. Eine Entität (Entity) beschreibt ein generelles Objekt, das durch seine Komponenten (Components) beschrieben wird. Die ganze Welt (World) wird durch Entities dargestellt. Alle Components beschreiben in ihrer Gesamtheit das Entity als spezifisches Objekt. Die Systeme sind für die Veränderung der World zuständig und führen deshalb Aktionen für bestimmte Components aus und verändern diese [t-07]. Aufgrund der Tatsache, dass in der MTS das ECS-Pattern um ein weiteres Merkmal erweitert wurde, werden die Systeme im Folgenden Aspects und das Pattern ECA (Entity-Component-Aspect) genannt. Die Erweiterung bezieht sich auf die Views, die die Aspects bündeln, um eine bessere Klassifizierung zu erhalten. D.h. es werden zur Laufzeit das Entity und seine Components betrachtet und entsprechend mit einer View assoziiert, die die Aspects für das Entity beinhaltet.

Die Klasse „World“ beinhaltet die Gesamtheit des aktuellen Simulationszustands. Sie kennt alle Entities und deren Zustände. Sie ist von der Klasse „SimState“ von MASON abgeleitet. Die Entities (z.B. Wasserflächen oder Schiffe) stellen ein physikalisches Objekt in der Simulation dar. Sie definieren sich über die Summe ihrer Components und einen eindeutigen Bezeichner. Eine Component (z.B. geometrische Form, Geschwindigkeit) stellt eine Menge von zusammengehörenden Eigenschaften dar. Komponenten definieren eigene Invarianten, die jedoch in einer Beziehung zu anderen Informationen aus der Welt stehen.

Weiterhin besitzt die World einen MTSScheduler, der den Ablauf der Simulation regelt. Für den Simulationsablauf bzw. die Ereignisverarbeitung wird die Klasse „Steppable“ von MASON genutzt. Der MTSScheduler erbt vom Standard-Scheduler von MASON. Der

MTSScheduler benutzt die komplette Funktionalität von MASON, erweitert um die Zeitsynchronisierung mit Co-Simulatoren.

Die Klasse „View“ kapselt das physikalische Verhalten der Objekte der Welt zueinander. Sie dient der Klassifizierung von verschiedenen physikalischen Objekten (z.B. verschiedene Schiffstypen). Die View (z.B. Containerschiff oder Schlepper) hat mehrere Aspects (z.B. Beschleunigung, Ruderauswirkungen), die wiederum mehrere Components besitzen. Welche View verwendet wird ist abhängig von den Components der Entity. Aspects stellen die physikalischen Auswirkungen, die auf eine Entität und seine Components angewendet werden, dar. Sie definieren also das physikalische Verhalten, das in der Welt gilt. Die Aspects sind vom Typ „Steppable“ und werden in bestimmten Intervallen ausgeführt. Die Reihenfolge der Ausführung von Aspects in der View, bestimmt die View selber.

Des Weiteren gibt es Agenten, die pro Entity bzw. View für die Steuerung zuständig sind. Hierzu hat der Agent mehrere Sensoren (z.B. GPS oder Karteninformationen), Aktuatoren (z.B. Steuerrad oder Schubhebel) und Verhalten (Behaviors). Die Agenten können mit Hilfe ihrer Sensoren die Umwelt wahrnehmen, welche durch die verschiedenen Entities und deren Components dargestellt wird. Mit Hilfe der Aktuatoren können die Agenten mit der Umwelt interagieren und diese verändern. Die Behaviors (z.B. Überholen oder Ausweichen) sind für das Verhalten der Schiffe zuständig, d.h. der Agent ist der Kapitän des Schiffes und kann mit Hilfe seiner Behaviors Manöver befehlen. Die Behaviors bekommen dazu die Informationen der Umwelt durch die Sensoren, die sie benötigen und versuchen darauf durch Steuerbefehle an die Aktuatoren mit der Umwelt zu interagieren. Hierbei ist wichtig, dass zwischen dem physikalischen Modell (Kapitel 2.1.3.2), welches die Physik der Welt beschreibt, und der Logik der Agenten (Kapitel 2.1.3.3), die beschreibt wie sich einzelne Agenten in bestimmten Situationen verhalten, unterschieden wird. D.h. die Logik der Agenten kann die Physik nicht überwinden.

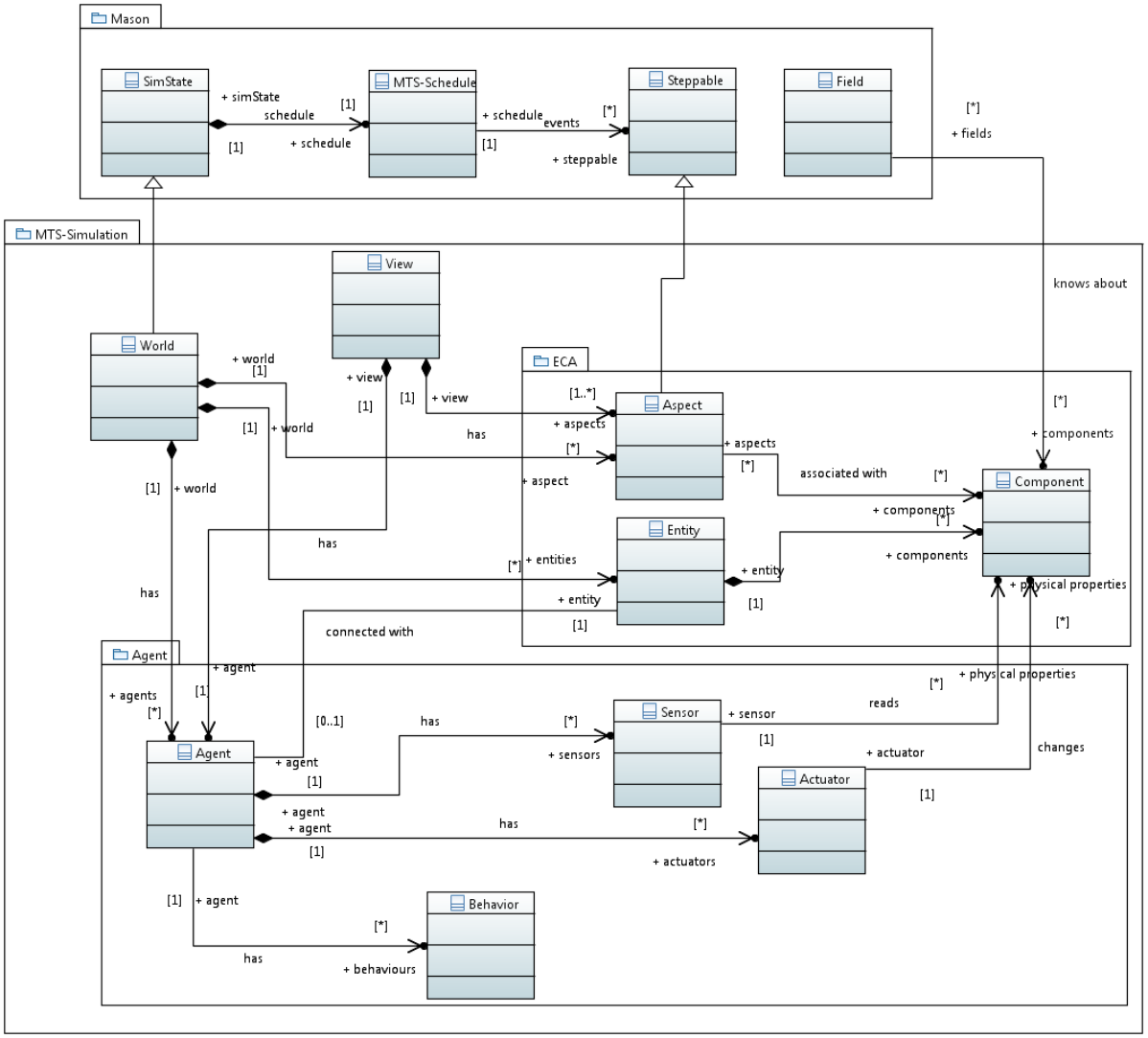

Abbildung 3: Architekturdiagramm der MTS

2.1.3.2 Physikalisches Modell

In diesem Kapitel wird die aktuelle Implementierung des physikalischen Modells beschrieben. Sie entspricht einem leicht erweiterten „3 DOF“-Modell (Degrees of freedom), aufbauend auf dem Modell aus „Handbook of marine craft hydrodynamics and motion control“ von Fossen und Thor [Fo11, S. 5f]. Der Fokus der Implementierung liegt auf der Simulation des (Container-) Schifffahrtsverkehrs. Während die Freiheitsgrade surge, sway und yaw näherungsweise abgebildet sind, wird der Freiheitsgrad heave nur ansatzweise berücksichtig (vgl. Abbildung 4).

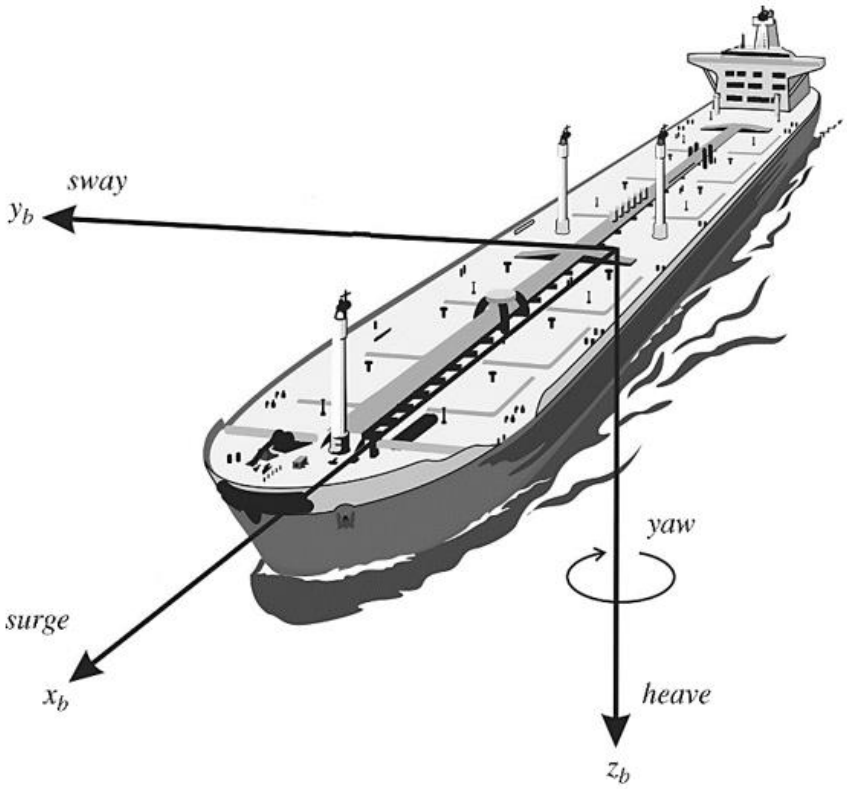

Abbildung 4: "3 DOF"-Modell (inkl. viertem Freiheitsgrad „heave")

Nachfolgend werden die einzelnen Freiheitsgrade gemäß ihrer Implementierung in der MTS beschrieben.

Surge

Surge beschreibt den Versatz eines Schiffes in Abhängigkeit zu der vergangenen Zeit als Geschwindigkeit in Metern pro Sekunde. Die Berechnung der Geschwindigkeit hängt von einem grundlegenden Beschleunigungsfaktor, der aktuellen Beladungssituation, der alten Geschwindigkeit sowie der seit der letzten Berechnung vergangenen Zeit ab. Der Beschleunigungsfaktor wird auf Basis der Schiffslänge berechnet. Zusätzlich zu den vom Schiff ausgehenden Kräften wirken die Umwelt-Faktoren Strömung und Wind auf den Freiheitsgrad surge ein (vgl. Kapitel 2.1.4.2).

vesselLength [m]: Gibt die Länge eines Schiffes an.

maxWeight [kg]: Gibt das Maximalgewicht eines Schiffes an.

currentWeight [kg]: Gibt die aktuelle Beladungssituation eines Schiffes an.

weightFactor []: Gibt einen Faktor an, mit dem der Einfluss des aktuellen Beladungszustands auf die Beschleunigung geändert werden kann. Es wurde im Rahmen der Recherchen hierzu keine eindeutige und allgemein gültige Beziehung von Gewicht zu Beschleunigung ermittelt. Da das Gewicht aber unbestritten einen großen Einfluss auf die Beschleunigung von Schiffen hat, wurde der Faktor eingeführt.

maxSpeed [m/s]: Gibt die Maximalgeschwindigkeit eines Schiffes an.

oldCurrentSpeed [m/s]: Gibt die aktuelle Geschwindigkeit zum Zeitpunkt der letzten Berechnung an.

minAcceleration [m/s²]: Gibt den Beschleunigungsfaktor bei maximaler Beladung an [Wim14a].

acceleration [m/s²]: Gibt den aktuellen Beschleunigungsfaktor für das Schiff in Abhängigkeit vom aktuellen Beladungszustand an.

vesselStoppingDistanceFactor []: Gibt den Wert an, den ein Schiff bei voller Beladung benötigt, um aus voller Fahrt zum Stehen zu kommen. Der Wert wird als Faktor (hier 15) in Abhängigkeit zur Schiffslänge angegebenen [Am06, S.15].

time [s]: Gibt die Zeit in Sekunden an, die seit der letzten Berechnung vergangen ist.

currentSpeed [m/s]: Gibt die aktuelle Geschwindigkeit des Schiffes an.

$$minAcceleration = \frac{(maxSpeed)^2}{2 \times vesselStoppingDistanceFactor \times vesselLength}$$

$$acceleration = minAcceleration \times \frac{maxWeight}{currentWeight} \times weightFactor$$

$$currentSpeed = oldCurrentSpeed \pm acceleration \times time$$

Sway

Da die Schiffe in der Simulation nicht über Seitenantriebe (z.B. Bug- und Heckstrahlruder) verfügen und somit keine vom Schiff ausgehenden Kräfte auf den Freiheitsgrad Sway wirken, müssen lediglich die Umwelt-Faktoren Strömung und Wind berücksichtigt werden.

Die Auswirkungen der Umwelt-Faktoren auf den Freiheitsgrad Sway sind in Kapitel 2.1.4.2 beschrieben.

Yaw

Yaw beschreibt das aktuelle Heading eines Schiffes. Das aktuelle Heading wird mit Hilfe eines Schiffstyp-abhängigen Rotationsfaktors und des Headings zum letzten Berechnungszeitpunkt berechnet. Der Freiheitsgrad Yaw wird in dem vorliegenden Modell nicht von Umwelt-Faktoren beeinflusst.

vesselLength [m]: Gibt die Länge eines Schiffes an.

maxWeight [kg]: Gibt das Maximalgewicht eines Schiffes an.

currentWeight [kg]: Gibt die aktuelle Beladungssituation eines Schiffes an.

currentSpeed [m/s]: Gibt die aktuelle Geschwindigkeit des Schiffes durch das Wasser an.

currentRudderAngle [°]: Gibt die aktuelle Stellung des Ruders an.

maxRudderAngle [°]: Gibt den größtmöglichen Winkel des Ruders an (der Winkel kann entweder nach links oder rechts eingeschlagen sein).

imoConstraintRotationFactor []: Gibt den Wert an, den ein Schiff für ein 180°-Wendemanöver bei voller Beladung benötigt. Der Wert wird als Faktor (hier 5) in Abhängigkeit von der Schiffslänge angegeben [Am06, S. 10f].

rotationFactor [s/m]: Gibt den Rotationsfaktor eines Schiffes in Abhängigkeit zu seiner Größe etc. an.

oldHeading [°]: Gibt das Heading eines Schiffes zum letzten Berechnungszeitpunkt an.

currentHeading [°]: Gibt das aktuelle Heading eines Schiffes an.

$$rotationFactor = \frac{\left(\frac{180°}{vesselLength \times imoContraintRotationFactor}\right)}{maxRudderAngle} \times 1s$$

$$currentHeading = oldHeading + currentRudderAngle \times rotationFactor \times currentSpeed$$

Heave

Der Freiheitsgrad heave wir lediglich zu Beginn der Simulation für jedes Schiff bestimmt und bleibt während der Simulation konstant. Für die Berechnung von heave wird angenommen, dass der Schiffsrumpf einem Quader entspricht dessen Tiefgang sich linear zu dessen Beladungszustand verhält. Als Grenzwerte dienen die Angaben über den minimalen und den maximalen Tiefgang. Bei keiner Beladung wird der minimale Tiefgang des Schiffes angenommen, bei maximaler Beladung dementsprechend der maximale Tiefgang.

currentWeight [kg]: Gibt die aktuelle Beladungssituation eines Schiffes an.

maxWeight [kg]: Gibt das Maximalgewicht eines Schiffes an.

minDraught [m]: Gibt den Tiefgang eines unbeladenen Schiffes an.

maxDraught [m]: Gibt den Tiefgang eines Schiffes bei vollständiger Beladung an.

currentDraught [m]: Gibt den aktuellen Tiefgang eines Schiffes entsprechend seiner Beladung an.

$$currentDraught = minDraught + \left(\frac{currentWeight}{maxWeight} \times (maxDraught - minDraught)\right)$$

Wie das physikalische Modell in die Architektur innerhalb der MTS eingebunden ist, kann in Kapitel 2.1.3.1 nachgelesen werden, wie es sich erweitern lässt, steht in Kapitel 3.3.1.1.4.

2.1.3.3 Agententypen

In diesem Kapitel werden die in der MTS implementierten Agententypen näher beschrieben. Die nachfolgend beschriebenen Agententypen unterscheiden sich vor allem hinsichtlich ihrer Steuerungsmöglichkeiten, d.h. in der Art und Weise, wie die verschiedenen Agenten von externen Federates oder von der MTS selbst gesteuert werden können. Das konkrete Verhalten (die Logik) eines Agenten, d.h. die Definition, wann er welche Steuerbefehle generiert, ist auf die Behaviors ausgelagert (s. Kapitel 2.1.3.4).

Standard Agent

Der Standard Agent erhält über eine EMod-Datei eine Route mit mindestens zwei Punkten (Start- und Zielpunkt) und fährt diese Route ab, in dem er eigenständig dazwischenliegende

Routen bzw. Wegpunkte errechnet. Dies geschieht unter Beachtung der physikalischen Möglichkeiten (vgl. Abbildung 5). Somit ermöglicht der Standard Agent dem Benutzer eine Simulation von realistischen Verkehrssituationen. Der Standard Agent kann in unterschiedlichen Ausprägungen aufgesetzt werden (z.B. durch das Hinzufügen oder Entfernen bestimmter Behaviors). Für den Benutzer besteht dank des Framework-Charakters der MTS die Möglichkeit der Realisierung verschiedener Ausprägungen des Standard Agents.

Der Standard Agent nutzt einen A*- (Wegfindungs-) Algorithmus und die Manhattan-Metrik (Bewertung der Kosten des betrachteten Wegs), um seinen Zielpunkt zu erreichen [JL05, S. 7ff]. Die im Wasser befindliche Strecke zwischen Start- und Zielpunkt wird in Quadrate gerastert (Standard-Rastergröße mit einer Seitenlänge von 50 m). Dabei muss für jedes Quadrat gelten:

- Die Wassertiefe des Quadrates muss tiefer als die vom Schiff benötigte Wassertiefe sein.
- Das Quadrat befindet sich in einem Gebiet, das von Schiffen befahren werden darf.

Alle Quadrate erhalten einen Wert, der proportional zur Entfernung vom Startpunkt ansteigt. Dabei wird der Weg als die räumliche Abfolge von Quadraten beschrieben. Der optimale Weg ist derjenige, bei dem das Zielquadrat den geringsten Wert enthält. Bei Hindernissen werden die umliegenden Quadrate Stück für Stück „abgetastet“. Dabei wird inkrementell jeweils dasjenige Quadrat als optimal angesehen, welches sich näher am Zielquadrat befindet. Es wird nach dem besten Weg gesucht, also dem mit dem am wenigsten angestiegenen Wert. In der Bewertung des optimalen Wegs können bestimmte Wasserflächen, wie z.B. ausgewiesene Wasserstraßen, bevorzugt berücksichtigt werden. Im Anschluss wird eine Schiffsroute über diejenigen Quadrate gelegt, die als optimal eingeschätzt werden. Im Anschluss an die Wegfindung wird die endgültige Schiffsroute berechnet. Bei der Berechnung der Schiffsroute wird der Weg über die optimalen Quadrate geglättet, damit das Schiff die Route möglichst unkompliziert abfahren kann. Vor allem kurz aufeinander folgende Kurswechsel sollen vermieden werden. Weiterhin berücksichtigt die Planung der Route die Schiffsdynamik, um sicherzustellen, dass die berechnete Route auch vom jeweiligen Schiff abgefahren werden kann. Dies kann im schlechtesten Fall jedoch dazu führen, dass es für einen gefundenen Weg (bestehend aus den aufeinander folgenden Quadraten) keine für das Schiff befahrbare Route gibt. Für den Fall, dass es eine befahrbare Route gibt, werden um die einzelnen Eckpunkte der Route jeweils zwei Wegpunkte gelegt. Das Schiff orientiert sich später an diesen Wegpunkten und versucht diese direkt anzufahren (falls kein Behavior den Kurs anderweitig ändert).

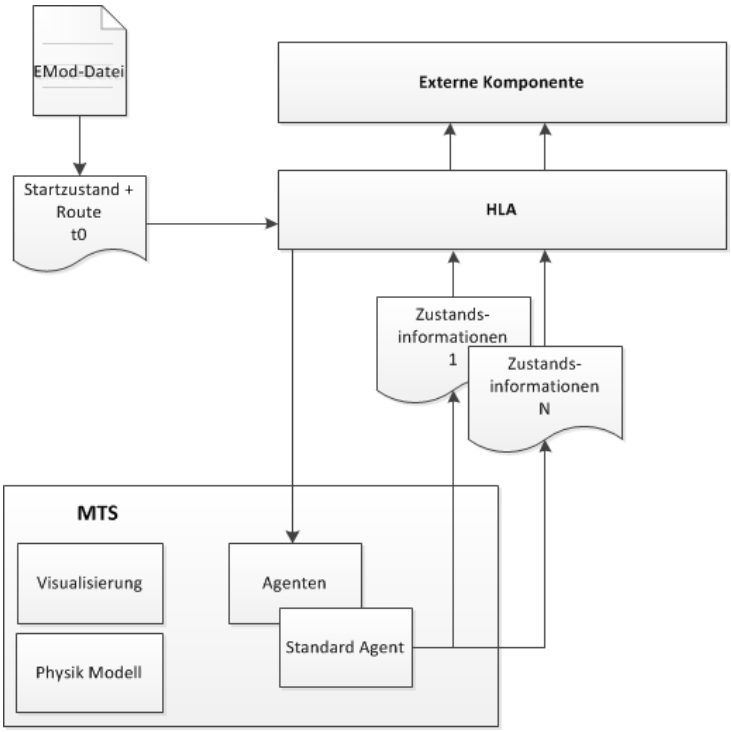

Abbildung 5: Kommunikation des Standard Agent

Routing Agent

Der Routing Agent wurde hauptsächlich für die Erprobung der Kollisionsvermeidungs-Komponente entwickelt. Der Routing Agent erhält über HLA Befehle von einem externen Federate. Die übermittelten Befehle enthalten eine Route, die der Agent im Rahmen seiner ihm durch die Simulation bereitgestellten physikalischen Möglichkeiten automatisch abfährt (vgl. Abbildung 6). Die Route besteht aus einzelnen Routenpunkten, welche durch eine geografische Position angegeben sind (vgl. das HAGGIS-Datenmodell – PathWay in Package „de.emir.model/model/application/TrafficSystem“). Um die Route automatisch abzufahren, nimmt der Agent mit seiner aktuellen Reisegeschwindigkeit direkt Kurs auf den jeweils nächsten Routenpunkt aus der Liste.

Die Route kann während der Laufzeit beliebig verändert werden. Dabei ist zu beachten, dass der Agent die alte Route immer komplett durch die neue Route ersetzt. Dementsprechend ist der externe Federate dafür verantwortlich, immer die gesamte Route zu senden. Dies ist in dieser Form aus dem Grund entwickelt worden, dass beim Austausch einzelner Punkte aus der Route der externe Federate die aktuelle Route ebenfalls kennen muss. Denn nur so kann er entscheiden, welche Punkte der Route ersetzt werden sollen. Da sich der externe Federate die Route der einzelnen Routing Agents notwendigerweise merken muss, wurde die Entscheidung getroffen, die alte Route immer komplett durch die neue Route zu ersetzen.

Neben der Möglichkeit die Route auszutauschen, hat der externe Federate zudem die Möglichkeit, über HLA die Zielreisegeschwindigkeit des Agenten zu bestimmen. Diese Geschwindigkeit wird, genau wie die Zielkoordinaten, im Rahmen der dem Agenten zugewiesenen physikalischen Möglichkeiten umgesetzt.

Im Gegensatz zum Standard Agent verfügt der Routing Agent über keinen Algorithmus für die Wegfindung. Der Routing Agent ist somit nicht in der Lage, zwischen den Routenpunkten liegende Wegpunkte eigenständig zu errechnen. Diese Funktion wäre relativ einfach erweiterbar, ist aber explizit nicht gewünscht. Eine Wegfindungsfunktion würde die Qualität der durch die Kollisionsvermeidungs-Komponente errechneten Routen verfälschen und wäre dementsprechend nicht zielführend.

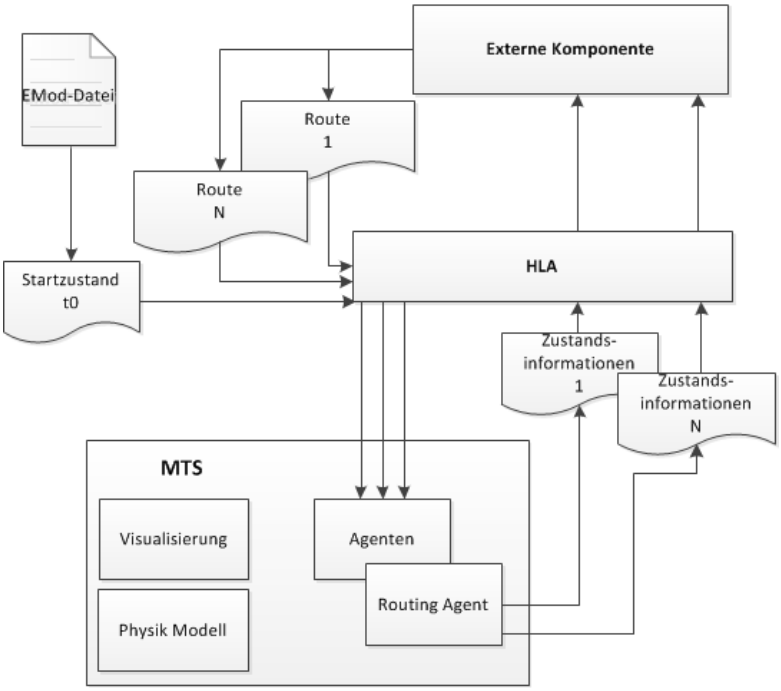

Abbildung 6: Kommunikation des Routing Agent

Steering Agent

Der Steering Agent bietet die Möglichkeit der freien Fahrt. D.h. er stellt keinen Automatismus zur Verfügung, der in irgendeiner Art dazu befähigt, Ziele wie Routenpunkte automatisch anzufahren. Der Steering Agent wird zu Beginn der Simulation über EMod-Dateien aufgesetzt. Zur Laufzeit kann der Steering Agent über HLA manuell über den durch die PG MAPS entwickelten Federate „InteractiveVessel“ (vgl. Kapitel 2.4.2) gesteuert werden. Weiterhin kann jeder weitere externe Federate Steuerungsbefehle über HLA an den Agenten senden (vgl. Abbildung 7).

Ein Steuerbefehl kann aus zwei Bestandteilen bestehen: einer veränderten Zielmotorleistung und einer veränderten Einstellung des Zielruders. Beide Zielwerte werden nach der Übermittlung schnellstmöglich durch den Agenten herbeigeführt.

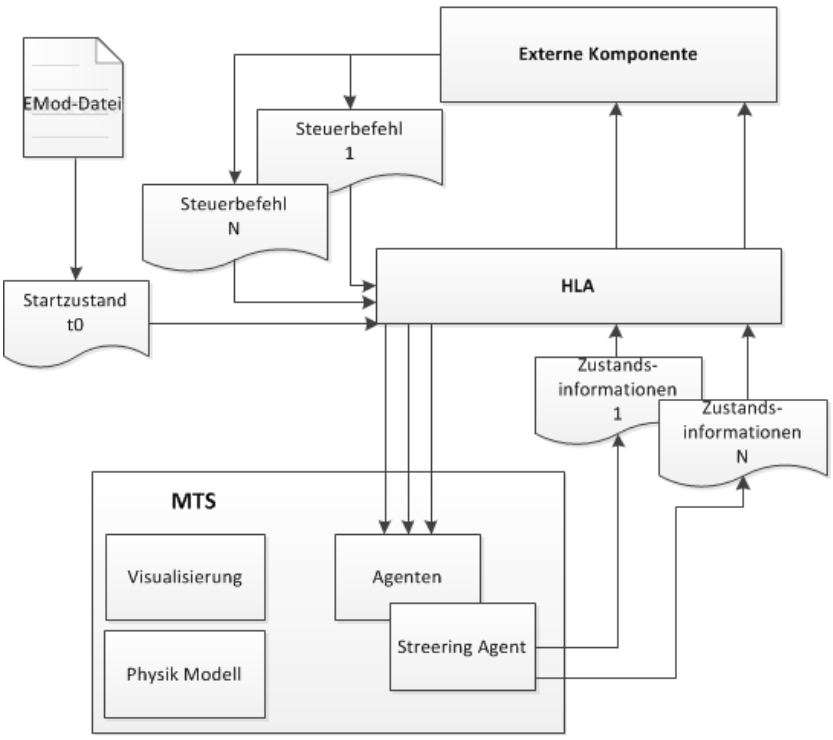

Abbildung 7: Kommunikation des Steering Agent

NMEA Agent

Der NMEA Agent dient vor allem zur Anbindung eines externen Schiffsführungssimulators. Der Schiffsführungssimulator sendet in diesem Fall über UDP Daten an den von der PG MAPS implementierten NMEA Federate. Dieser wiederum wandelt die UDP Signale (NMEA Senteces) in HLA Befehle um und kommuniziert diese an die MTS (s. Abbildung 8). Dieser Agent hat daher auch exklusiv die Berechtigung, seine Physik zu umgehen und der MTS Positionsdaten für das Schiff zu senden. Diese Positionsdaten werden im Anschluss ohne Berücksichtigung der Physik in der Simulation umgesetzt (vgl. Kapitel 2.4.5).

Der NMEA Agent vernachlässigt die Physik jedoch nicht komplett. Wie bei den anderen Agenten kann auch dem NEMEA Agent ein beliebiges physikalisches Verhalten zugeordnet werden. Der NMEA Agent besitzt die Fähigkeit, basierend auf den zuletzt über HLA erhaltenen Daten die weiteren Positionen zu extrapolieren. Konkret greift der Mechanismus auf die zuletzt gesendeten Positions- und Geschwindigkeitsdaten sowie das Heading des Schiffes zurück. Auf Basis dieser Daten wird mit Hilfe der zugeordneten Physik die nächste Position bestimmt. Dies geschieht nur so lange, bis neue Daten vom Schiffsführungssimulator übermittelt werden, da diese für den Agenten führend sind.

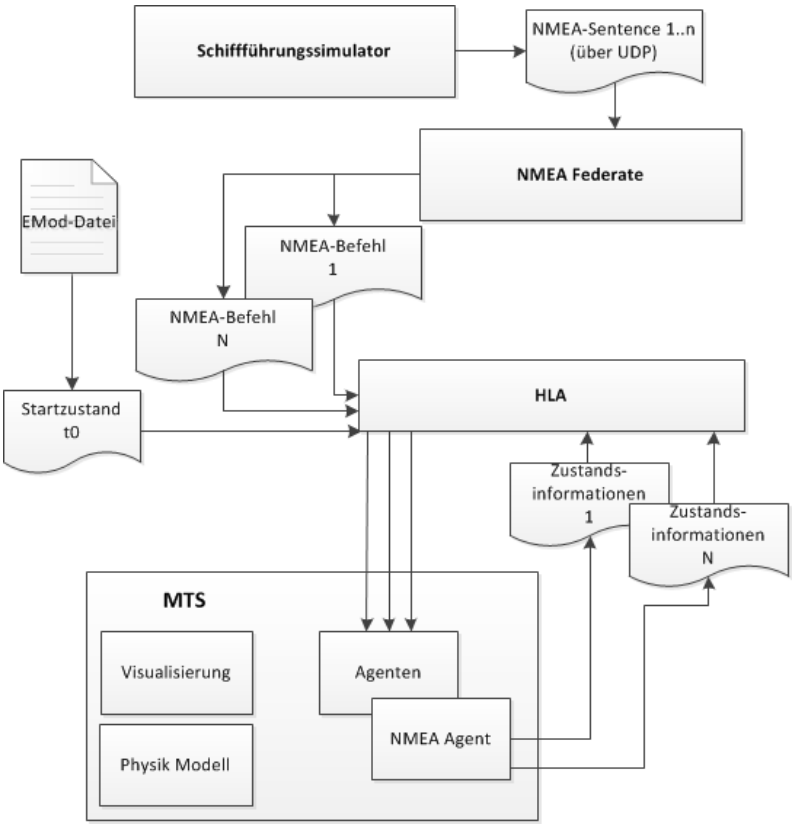

Abbildung 8: Kommunikation des NMEA Agent

Generated Agent

Der Generated Agent ist für den Zweck entwickelt worden, ohne großen Aufwand und möglichst automatisiert ein erhöhtes Verkehrsaufkommen in einem bestimmten Bereich zu erzeugen. Er basiert auf dem Standard Agent und besitzt im Wesentlichen die gleichen Eigenschaften. Im Gegensatz zum Standard Agent werden die Generated Agents mit Hilfe eines Generators vor dem Simulationsstart erzeugt. Der Generator kann durch über die Visualisierung bedient werden. Hierbei können Einstellungen getroffen werden wie lange die Simulation laufen soll und wie viele Generated Agents dabei erzeugt werden sollen. Auch die Routenpunkte müssen beim Generated Agent nicht manuell angegeben werden. Stattdessen fahren die Generated Agents zufällig von einem beliebigen Hafengebiet zu einem anderen beliebigen Hafengebiet. Dieser Vorgang kann sich beliebig oft wiederholen, wobei das Ziel immer neu bestimmt wird. Ein Generated Agent startet, wie ein Standard Agent, immer in einem Hafengebiet (vgl. Kapitel 2.4.6).

2.1.3.4 Verhaltenslogik

Im Folgenden werden die aktuell implementierten Verhaltenslogiken (Behaviors) beschrieben. Bei den Behaviors handelt es sich nicht um die physikalischen Eigenschaften eines Schiffs, sondern um die Verhaltensweisen eines Agenten. Diese Verhaltensweisen haben immer die Aufgabe, eine Art von Steuerbefehlen zu generieren. Diese werden im Anschluss über die Steuermöglichkeiten an das Schiff weitergegeben. Die Verhaltensweisen

der Agenten können im entferntesten Sinne mit dem Verhalten eines Kapitäns bzw. Steuermannes verglichen werden. Die Behaviors können den im vorherigen Kapitel beschriebenen Agenten im Prinzip beliebig zugeordnet werden, auch wenn dies aufgrund der unterschiedlichen Steuerarten der Agenten nicht immer sinnvoll ist. Klassischerweise wird der Standard Agent mit den nachfolgenden Behaviors ausgestatten. Durch die Zuordnung verschiedener Behaviors können so mit dem Standard Agent verschieden ausgeprägte Agenten erstellt werden. Für den Standard Agent sind gewisse Behaviors essentiell. So ist der Standard Agent z.B. ohne die Zuordnung eines Ruder- und Geschwindigkeits-Behavior manövrierunfähig.

Ruder-Behavior

Das Ruder-Behavior ist ein Steuerverhalten, welches Agenten ermöglicht, bestimmte Zielkoordinaten anzufahren. Hierbei errechnet der Agent die Differenz aus dem eigenen Heading und dem Winkel zwischen sich und der Zielkoordinate. Im Anschluss wird die errechnete Differenz als Zielruderwinkel definiert und dieser daraufhin eingeschlagen, um den Winkelabstand zu der Zielkoordinate Stück für Stück zu verringern, bis diese schließlich erreicht wird.

Geschwindigkeits-Behavior

Ähnlich dem Ruder-Behavior errechnet das Geschwindigkeits-Behavior die notwendige Motorleistung, um eine bestimmte Zielgeschwindigkeit zu erreichen. Die Berechnung basiert auf der Voraussetzung, dass sich die Leistung des Motors linear zur Geschwindigkeit verhält. Jedes Schiff verfügt über eine maximale Geschwindigkeit, welche bei hundert Prozent Motorleistung erreicht wird. Bei null Prozent Motorleistung hat das Schiff dementsprechend die Geschwindigkeit null Meter pro Sekunde. Anschließend wird die errechnete Motorleistung über die Steuerungs-Komponenten eingestellt und schnellstmöglich erreicht. Schnellstmöglich bedeutet in diesem Zusammenhang, dass der Motor nicht ohne eine gewisse Zeitspanne von z.B. null auf hundert Prozent beschleunigt werden kann. Wie schnell das Hochfahren des jeweiligen Motors dauert (die Dauer der zuvor genannten Zeitspanne), ist im entsprechenden Aspect definiert.

Rechtsfahr-Behavior

Das Rechtsfahr-Behavior erbt vom Ruder-Behavior und erweitert dieses um das Rechtsfahrgebot auf Wasserstraßen. Dieses Behavior ist nur für Agenten, die Routen als Steuerungsmechanik besitzen, sinnvoll und funktional.

Bei einem Agenten, der dieses Behavior implementiert hat, ermittelt dieser bei Erreichen der einzelnen Routenpunkte, ob die Strecke zwischen dem vorherigen Routenpunkt und dem nächsten Routenpunkt komplett auf einer Wasserstraße verläuft. Diese Prüfung ist dank der verschiedenen Layer im S-57 Seekartenstandard möglich. Falls die besagte Strecke komplett auf einer Wasserfahrstraße verläuft, prüft der Agent, ob er sich zu weit links befindet. Hierzu ermittelt er die breite der Wasserstraße an seiner aktuellen Position. Wie weit links sich ein Schiff bewegen darf, ist im Code anpassbar (In der Methode „ensureDrivingOnRightRule“ in Package „de.wi_ol.be.mts.agent.behavior.InternalDrivingRightBehavior“). Die Angabe enthält einen Schwellenwert in Prozent, der nicht überschritten werden darf. Solange sich das Schiff nicht

zu weit links befindet, wird keine Korrektur des Kurses vorgenommen. Sobald der angegebene Schwellenwert überschritten wird, wird das Ruder-Behavior übersteuert und der nächste Routenpunkt um einen anpassbaren Wert nach rechts verlagert. Sobald das Schiff den Schwellenwert unterschreitet, wird wieder das normale Ruder-Behavior aktiviert.

Überholmanöver-Behavior

Das Überholmanöver-Behavior bietet Agenten die Möglichkeit, langsamere Schiffe, die auf einer gleichen Route fahren, zu überholen. Dabei werden vor allem Sicherheitsmaßnahmen – Mindestabstände zwischen den Schiffen und potenzielle Gefahren durch Gegenverkehr – berücksichtigt. Die im Rahmen des Überholmanövers notwendigen Mindestabstände zwischen den Schiffen sind im Programmcode anpassbar. Dies wird im Folgenden an den jeweiligen Abständen konkret beschrieben.

Bevor ein Schiff ein Überholmanöver startet, müssen verschiedene Bedingungen erfüllt sein:

- **1. Bedingung:** Vor dem Schiff, das das Überholmanöver anwendet (nachfolgend Schiff-S) befindet sich ein weiteres Schiff (nachfolgend Schiff-L) innerhalb dessen Nahbereichs.
- **2. Bedingung:** Schiff-L fährt aktuell einen Kurs, der der Route des Schiffs-S nicht zu ähnlich ist. Nicht zu ähnlich bedeutet in diesem Zusammenhang, dass die Differenz aus Heading von Schiff-L zu der Ausrichtung des Kurses von Schiff-S einen konfigurierbaren Schwellenwert (anpassbar in der Konstante ANGLE_TOLERANT_IN_SAME_DIRECTION_IN_DEGREE in de.wi_ol.be.mts.agent.behavior.Behavior) nicht unterschreitet. Die Ausrichtung des Kurses definiert sich über den Routenpunkt, der dem Messpunkt vorrausgeht, zu dem Routenpunkt, der dem Messpunkt nachgelagert ist. Die Differenz wird an der Position von Schiff-S gemessen.
- **3. Bedingung:** Schiff-L fährt erheblich langsamer als Schiff-S. Hier wird ein absoluter, konfigurierbarer Wert in Metern pro Sekunde als Schwellenwert festgelegt.
- **4. Bedingung:** Die für das Überholmanöver benötigte Strecke ist frei. Das bedeutet, auf ihr befinden sich kein anderes, in entgegengesetzter Richtung fahrendes Schiff oder andere Hindernisse, wie z.B. Land. Die für das Überholmanöver benötigte Strecke ist als Fläche realisiert. Hierfür wird die auf Basis der für das Überholmanöver benötigten Strecke, dem eigentlichen Kurs und einem konfigurierbaren Abstand eine Fläche ermittelt. Auf dieser Fläche dürfen sich keine Hindernisse befinden. Die benötigte Strecke entspricht der zum Zeitpunkt der Prüfung noch zu fahrenden Strecke bis das Überholmanöver abgeschlossen ist. Allerdings wird immer mindestens für die Länge des Nahbereichs geprüft (vgl. Kapitel 2.4.7). Der genannte Abstand berechnet sich aus der Schiffsbreite sowie dem Seitenabstand zwischen Schiffen bei einem Überholmanöver. Dieser Seitenabstand ist konfigurierbar (in der Methode „passingManeuverDistance“ in Package „de.wi_ol.be.mts.agent.behavior.BehaviorUtil“) und berechnet sich aus der Breite von Schiff-S multipliziert mit der Geschwindigkeit von Schiff-S multipliziert mit einem beliebig wählbaren Sicherheitsfaktor für den Seitenabstand bei Überholmanövern.

Falls einer der vorherigen Bedingungen nicht erfüllt ist, wird das Überholmanöver nicht gestartet. Das Schiff-S reduziert dementsprechend sofort seine Geschwindigkeit und fährt unter Einhaltung seines Nahbereichsabstands, bis die Route wieder frei ist.

Für den Fall, dass alle Bedingungen erfüllt sind, beginnt Schiff-S mit dem Überholmanöver. Während die ersten drei Bedingungen nur für den Start des Überholmanövers relevant sind, wird die vierte Bedingung während des gesamten Überholmanövers ständig erneut geprüft. Dies ist der Tatsache geschuldet, dass zu Beginn nicht ermittelt werden kann, ob zu einem späteren Zeitpunkt Hindernisse auf der Überholstrecke bzw. Überholfläche auftauchen. Die vierte Bedingung stellt somit gleichzeitig ein Abbruchkriterium dar. Das genauere Verhalten des Schiff-S in diesem Fall wird zu Ende dieses Abschnitts erklärt.

Das Überholmanöver gliedert sich in drei Schritte:

- **1. Schritt:** Schiff-S beschleunigt auf Maximalgeschwindigkeit, um die zum Überholen benötigte Strecke möglichst kurz zu halten.
- **2. Schritt:** Schiff-S schert zum Überholvorgang nach links aus. Das Ausscheren geschieht, sobald Schiff-S einen konfigurierbaren Sicherheitsabstand zum Ausscheren unterschreitet. Sobald Schiff-S ausschert, wird das Rechtsfahr-Behavior (insofern vorhanden) blockiert. Wie weit Schiff-S nach links ausschert, hängt vom Seitenabstand ab. Dieser Seitenabstand ist konfigurierbar und berechnet sich wie in der 4. Bedingung für den Start eines Überholmanövers (siehe dazu 4. Bedingung in Aufzählung oben) beschrieben.
- **3. Schritt:** Schiff-S schert nach dem Überholen von Schiff-L wieder ein, womit das Überholmanöver abgeschlossen ist. Das Einscheren findet statt, sobald Schiff-L überholt ist und zwischen Schiff-S und Schiff-L ein konfigurierbarer Sicherheitsabstand überschritten wird. Sobald Schiff-S eingeschert ist, wird das Rechtsfahr-Behavior (insofern vorhanden) wieder reaktiviert und die ursprüngliche Reisegeschwindigkeit von Schiff-S wieder als Zielgeschwindigkeit eingestellt.

Wie zuvor beschrieben ist die vierte Bedingung, neben ihrer Funktion als Kriterium für den Start eines Überholmanövers, auch ein Abbruchkriterium. Sobald die Bedingung während des Überholmanövers erfüllt wird, wird ein Abbruch eingeleitet. Dieser Abbruch hat drei Ausprägungen:

- **1. Ausprägung des Abbruchs:** Schiff-S befindet sich im ersten oder zweiten Schritt des Überholvorgangs aber noch hinter Schiff-L. In diesem Fall wird die Geschwindigkeit solange verringert, bis Schiff-S den Nahbereichsabstand zu Schiff-L wieder hergestellt hat. Falls Schiff-S bereits ausgeschert ist, wird parallel zur Geschwindigkeitsreduktion auch wieder mit dem Einscheren begonnen.
- **2. Ausprägung des Abbruchs:** Schiff-S befindet sich im zweiten Schritt des Überholvorgangs und zudem direkt links von Schiff-L. In diesem Fall wird die Geschwindigkeit ebenfalls solange verringert, bis Schiff-S den Nahbereichsabstand zu Schiff-L wieder hergestellt hat. Mit dem Einscheren wird jedoch solange gewartet, bis sich Schiff-S wieder hinter Schiff-L befindet.
- **3. Ausprägung des Abbruchs:** Schiff-S befindet sich im dritten Schritt des Überholvorgangs, hat Schiff-L demnach bereits überholt. In diesem Fall wird die Überholregel – erst nach dem Überschreiten des Sicherheitsabstands für Überholmanöver einzuscheren – missachtet und direkt eingeschert. Schiff-S fährt solange mit Maximalgeschwindigkeit weiter, bis der Sicherheitsabstand nach dem Überholen zu Schiff-L überschritten wird. Anschließend reduziert Schiff-S die Zielgeschwindigkeit wieder auf seine ursprüngliche Reisegeschwindigkeit vor dem Überholvorgang.

Ausweichmanöver-Behavior

Mit dem Ausweichmanöver-Behavior besitzen die Agenten die Fähigkeit, in bestimmten Situationen Kollisionen mit anderen Schiffen vermeiden zu können. Für das Ausweichmanöver werden alle Schiffe, die innerhalb des Nahbereichs als potenzielle Gefahr angesehen werden können, berücksichtigt. Für diese Schiffe wird errechnet, zu welchem Zeitpunkt sie die Route des eigenen Schiffes schneiden werden, insofern sie ihren aktuellen Kurs beibehalten (TCPA, time to closest point of approach). Anschließend wird berechnet, an welchem Punkt sich das eigene Schiff voraussichtlich zu diesem Zeitpunkt befinden wird (CPA, closest point of approach). Unterschreitet der Abstand der beiden Schiffe zu dem errechneten Zeitpunkt einen im Programmcode konfigurierbaren Schwellenwert, wird ein Ausweichmanöver gestartet (in der Methode „getSecurityDistanceEvasion“ in Package „de.wi_ol.be.mts.agent.behavior.InternalEvasionBehavior“).

Das Ausweichmanöver-Behavior ähnelt in seiner Ausprägung dem Rechtsfahr-Behavior. Der Agent setzt den nächsten Routenpunkt weiter nach rechts, um eine mögliche Kollision zu vermeiden. Wie weit der Routenpunkt nach rechts versetzt wird, ist ebenfalls im Programmcode konfigurierbar.

Wie die Agententypen bzw. die Verhaltenslogik in die Architektur innerhalb der MTS eingebunden ist, wird in Kapitel 2.1.3.1 beschrieben, wie sich diese erweitern lassen, kann in Kapitel 3.3.1.1.2 nachgelesen werden. Wie die Funktionalität der Kollisionserkennung erweitert werden kann, steht in Kapitel 3.3.1.1.5.

2.1.3.5 Visualisierung

In diesem Kapitel werden die grundliegenden Technologien zur Visualisierung des simulierten Schiffsverkehrs und dessen Umgebung beschrieben. Hierunter fällt die Visualisierung der Simulation an sich. Die Umweltsimulation und Analysekomponente werden über die CERTI-Laufzeitinfrastruktur angebunden und sind als alleinstehende bzw. austauschbare Anwendungen zu betrachten.

Die grafische Benutzeroberfläche der MTS basiert auf der Standard-Visualisierung MASONs. Die einzelnen grafischen Elemente, wie das Display und die Konsole des MASON-Frameworks wurden zum Teil für die MTS angepasst und um neue Elemente, wie z.B. dem Property-Panel, dem Scenario-Editor oder dem Statistikbereich, erweitert. Diese Anpassungen und Erweiterungen sind mit Hilfe der Grafikbibliotheken AWT [Or14a] und Swing [Or14b] realisiert worden, welche zu den Standardbibliotheken von Java gehören. Wie dies umgesetzt wurde, wird in Kapitel 3.3.1.2 näher erläutert.

Als Grundlage für die grafischen Oberflächen dienen Mockups, die während der Systemspezifikation mit Pencil erstellt wurden [Ev14]. Die Implementierungsphase war ein iterativer Prozess, bei dem die grafischen Oberflächen an die neuen Anforderungen aus den User Stories angepasst wurden. Um eine möglichst hohe Benutzerfreundlichkeit zu gewährleisten, wurde sich während der Erstellung der Mockups und deren Implementierung an die Java Look and Feel Design Guidelines orientiert [Su99].

Zu den neu entwickelten Elementen gehören unter anderem der „Vessel Property“-Reiter innerhalb der MASON-Konsole, welcher während des Simulationslaufs das Anzeigen statischer und dynamischer Eigenschaften einzelner Schiffe ermöglicht (vgl. Kap 2.3.4) und der Statistikbereich, welcher der grafischen Aufbereitung der zum Simulationslauf generierten Daten in Diagrammform dient. Für die Visualisierung des Statistikbereichs wurde

das Framework JFreeChart [Ob14] verwendet. Zur Aktualisierung der Daten der Schiffe im „Vessel Property“-Reiter und der Simulationsdaten im Statistikbereich wurde ein Portrayal erstellt (vgl. Kapitel 3.3.1.2). Nach der Konfiguration und dem Starten der Simulation werden in diesem Portrayal alle Eigenschaften der in der Simulation verfügbaren Schiffe gesammelt und nach jedem Simulationsschritt aktualisiert. Die Informationen, welche im „Vessel Property“-Reiter und Statistikbereich abgebildet werden, entsprechen den Eigenschaften eines „Vessel“ im HAGGIS-Datenmodell. Zur Erweiterung des „Vessel Property“-Reiters bzw. des Statistikbereichs muss zunächst der Datentyp „Vessel“ im HAGGIS-Datenmodell erweitert werden. Anschließend kann der „Vessel Property“-Reiter bzw. der Statistikbereich um die erweiterten Eigenschaften ergänzt werden.

Zur Abbildung der Seegebiete in der GUI der MTS werden standardisierte elektronische Seekarten benötigt, auch Electronic Nautical Charts (ENCs) genannt. Der derzeitige Standard hierfür ist der S-57-Standard, welcher in Zukunft von Standard S-101 abgelöst werden soll [In08]. Allerdings befindet sich dieser noch in der Entwicklung [In12].

Die Karten, nach S-57-Standard, werden von legitimierten Instituten, wie der NOAA (National Oceanic and Atmospheric Administration) [Na14] oder dem BSH (Bundesamt für Seeschifffahrt und Hydrographie) [Bu14c] erstellt. Für den Bereich der amerikanischen Küste werden diese ENCs von der NOAA zur freien Verfügung gestellt. Der Bereich der deutschen Küste wird von autorisierten Vertriebspartnern, wie beispielsweise SevenCs [Se14], bereitgestellt.

Zum Einbinden der Geoinformationen in die MTS benötigt das MASON-Framework die Erweiterung GeoMason. GeoMason bringt sowohl Vor- wie Nachteile mit sich. Zu den Vorteilen gehören:

- GeoMason ist eine Erweiterung von MASON
- Einfache Implementierung mit GeoMason
- Geringere Einarbeitungszeit, da MASON schon bekannt
- Die Karteninformationen liegen direkt in der Simulation vor
- Es lässt sich fast jedes Kartenformat in Shapefiles konvertieren
- Shapefile-Format ist ein offener Standard, welche viele Karten frei verfügbar anbietet

Die Nachteile GeoMasons sind:

- Hoher Konvertierungsaufwand der S-57 Karten
- Keine genaue ECDIS konforme Darstellung

Alternativ wurde betrachtet, ob ein Kartenplotter, wie der Kernel von SevenCs, verwendet werden kann. Jedoch wurden die Vor- und Nachteile gegenübergestellt und sich für die Variante des Umwandelns in Shapefiles und einlesen mittels GeoMason entschieden.

GeoMason unterstützt allerdings nicht die gängigen Standards für ENCs. Demzufolge wurden die vorliegenden Seekarten, im S-57-Standard, mit Hilfe des Geoinformationssystems ArcGIS in sogenannte Shapefiles konvertiert.

Die in den S-57 Standards enthaltenen projektionslosen Daten basieren auf dem WGS84 Koordinatensystem, welches eine winkelgetreue Darstellung der Seekarten gewährleistet [Bu14d] [Go09, S. 50ff]. Beim Exportieren der Seekarten, mittels ArcGIS, kann das bestehende Koordinatensystem beibehalten oder weitere Zielkoordinatensysteme ausgewählt werden. Dieses ermöglicht zum Beispiel das Verwenden von projizierten Koordinatensystemen, wie UTM (Universal Transverse Mercator) [Ar14].

Bei der Konvertierung ist zu beachten, dass das zugrundeliegende Koordinatensystem der Kartendaten auf Längen- und Breitengraden basieren muss. Denn nur dann kann eine WGS84 konforme Berechnung durchgeführt werden. Die Darstellung wird in MASON in Längen- und Breitengrad Informationen aus dem Kartenmaterial realisiert. Dabei entspricht ein Grad einer festen Längeneinheit.

Die im Laufe der Projektarbeit erzeugten Shapefiles sind im Java-Projekt „Simulation“ im Ordner „s57“ zu finden. In diesem Ordner müssen zur Darstellung der Seegebiete in der MTS mindestens die Layer für die Landfläche (Datei „LNDARE_A.shp“), Wasserfläche (Datei „DRGARE_A.shp“) und das Fahrwasser (Datei „DRGARE_A.shp“) vorhanden sein, da diese die grundlegenden Informationen zur Simulation der Seegebiete enthalten. Alle anderen Layer sind optional.

Wie die Shapefiles erstellt werden, wird in Kapitel 2.4.1 beschrieben. Wie diese in die MTS eingebunden werden, wird in Kapitel 2.4.1 und 3.3.1.1.1 dargestellt.

2.1.4 Umweltsimulator

Der Umweltsimulator stellt eine eigenständige Simulations-Komponente dar. Zur Kommunikation mit anderen Komponenten wird auf den HLA-Standard zurückgegriffen (vgl. Kapitel 3.2). Der Umweltsimulator ist damit ein weiterer Federate in der verteilten Simulation. Kommuniziert werden Umweltdatentypen, die im Kapitel 2.1.4.2 näher beschrieben werden. Es werden Strömungs-, Wind-, und Tidendaten unterstützt. Diese Datentypen sind im HAGGIS-Datenmodell enthalten, bzw. implementieren bereits vorhandener Datentypen (vgl. Kapitel 2.2).

Prinzipiell können beliebige Messdaten, die in einer Datenbank gespeichert sind, die Grundlage eines Umweltszenarios sein. Die Messdaten müssen zuvor aus verschiedenen Quellen – z.B. aus GRIB2-Files (GRIdded Binary) [Wim14b] - in die Datenbank eingelesen werden. Dazu wird ein Tool bereitgestellt (vgl. Kapitel 2.4.9.2 und 2.4.9.3). Es werden Datenbanken eingesetzt, da die Handhabung dieser im Gegensatz zum Umgang mit Datenstrukturen wie GRIB2-Files recht trivial ist. Diese Umweltdaten werden dann vom Umweltsimulator an die verteilte Simulation weitergegeben und können dort von den Federates weiterverarbeitet werden, die diese Umweltdaten über HLA anfordern und erhalten.

Es gibt vier verschiedene Umweltszenario-Typen. Der erste Szenario-Typ ist das parametrisierbare Umweltszenario, welches den Nutzer vor dem Simulationsstart auffordert, Umweltdaten zu definieren. Der zweite Szenario-Typ ist das Static Scenario. Hier wird eine Umweltdatenbank ausgewählt, aus welcher die Umweltdaten an die Simulation verschickt werden. Der dritte Typ ist das Dynamic Scenario. Wie beim Static Scenario wird eine Umweltdatenbank als Datengrundlage genommen. Die beiden Szenarien unterscheiden sich darin, dass sich beim statischen Szenario die Umweltdaten für einen bestimmten, in der Datenbank abgebildeten geografischen Punkt mit fortschreitender Simulationszeit nicht ändern. Im dynamischen Szenario hingegen werden zur Laufzeit der Simulation sich verändernde Umweltdaten an die verteilte Simulation geschickt. Das letzte Szenario ist das Umweltzonenszenario. Hierbei kann der Nutzer rechteckige Bereiche auf eine Karte zeichnen und diesen Umweltwerte zuweisen. Die Handhabung der Szenarien wird in Kapitel 2.4.9 genauer erläutert.

2.1.4.1 Eingesetzte Technologien

Die Umweltdaten für die Simulationsszenarien liegen als SQLite-Datenbankdatei vor [SQ14]. Im Gegensatz zu vielen anderen relationalen Datenbanksystemen baut SQLite nicht auf einer Datenbank-Serverarchitektur auf. Die Datenbank besteht nur aus einer Datei und es können die meisten Standardabfragen, die die SQL (Structured Query Language) bereitstellt, genutzt werden. Eingebunden wird SQLite in das Java-Projekt der MTS über ein externes JAR-File. Ein Simulationsszenario in Form einer SQLite-Datenbank kann damit bequem mit einem Datenbank-Bearbeitungstool erstellt und bearbeitet werden.

Bei denjenigen Umweltsimulator-Szenarien, die der Benutzer ohne eine vorliegende Szenarien-Datenbank verwenden kann, wird eine In-Memory-Datenbanklösung namens "H2 Database Engine" verwendet [Mü14]. Diese wird in das Java-Projekt der MTS ebenfalls per externem JAR-File eingebunden. In-Memory-Datenbanken unterscheiden sich von klassischen Datenbanken in erster Linie dadurch, dass sie die Daten nicht auf der Festplatte ablegen, sondern ihre Speicherung, Verwaltung und Bearbeitung im Arbeitsspeicher des Systems stattfindet. Vor allem bei größeren Schreib-/ Zugriffsoperationen oder umfangreichen Abfragen kann hier ein Performance-Vorteil erzielt werden. Des Weiteren kann die gewohnte SQL-Syntax zur Bearbeitung der Daten genutzt werden. Ein weiterer Vorteil für eine In-Memory-Lösung ist, dass die Daten ohnehin nicht persistent nach dem Ende eines Simulationslaufs vorliegen müssen und daher nach dem Simulationslauf nicht gespeichert werden müssen.

Um die Simulation mit realistischen Strömungsdaten zu versorgen, können Umweltszenarien aus GRIB2-Dateien des Bundesamts für Seeschifffahrt und Hydrographie (BSH) [Bu14c] generiert werden. Das BSH stellt auf seinem FTP-Server [Bu14b] täglich aktuelle Strömungsberechnungen des deutschen Seeraumes zum Download zur Verfügung. Um diese im binären GRIB2-Datenformat vorliegenden Daten für die Simulation verwenden zu können, müssen die Daten extrahiert und in eine SQLite-Szenarien-Datenbank importiert werden. Die Extraktion der Daten geschieht mit Hilfe des Programms „wgrib2“ [Na11]. Hierzu werden die Daten als CSV-Dump abgelegt und im nächsten Schritt per Java in eine für den Umweltsimulator lesbare Szenarien-Datenbank überführt. Dieser Vorgang ist in Kapitel 3.3.2.2 und 2.4.9 dokumentiert.

Für die Darstellung des Kartenmaterials im Umweltsimulator wurden die Open Source-Plugins JMapView [No14]und JXMapKit, welche auf OpenStreetMap-Kartenmaterialien zurückgreifen, verwendet [op14].

2.1.4.2 Umwelteinflüsse

Umwelteinflüsse werden vom Umweltsimulator als HLA-Objekte an die Simulation übergeben. Jeder Umwelteinfluss hat dabei ein eigens für ihn definiertes Objekt. Alle Umweltdaten beinhalten mindestens einen Wert, eine Positionsangabe des Messpunktes als GPS-Koordinate mit Längen- und Breitengrad und einen Zeitstempel. Mit diesen Daten ist es möglich, unterschiedliche Messpunkte eines Umwelt-Faktors an die Simulation zu übergeben. Die Simulation kann denjenigen Messpunkt anwenden, der am nächsten an einem Schiff liegt und Einfluss auf diesen ausübt.

In der Simulation werden zum Zeitpunkt des Projektabschlusses drei Umwelt-Faktoren berücksichtigt. Zu diesen gehören die Strömung, der Wind und die Tide. Darüber hinaus

werden der Simulation noch Informationen über Wellen zur Verfügung gestellt. Diese sind ebenfalls bereits im HAGGIS-Datenmodell integriert.

2.1.4.2.1 Strömung

Die Strömungsinformationen des Umweltsimulators wurden nach dem Vorbild des BSH [Bu12] erstellt. Das Datenmodell besteht dabei aus fünf Komponenten. Diese sind die Strömung in vertikaler Richtung, die Strömung in horizontaler Richtung [Vi03], die Uhrzeit zu der Strömungsmessung sowie der Längen- und Breitengrad des Punktes, an dem die Strömung gemessen wurde. Die Strömung setzt sich dabei wie gerade erwähnt aus einer horizontalen und einer vertikalen Strömungsrichtung zu. Damit ist gemeint, dass die gesamte Strömung aus zwei Teilströmungen besteht und errechnet wird. Die horizontale Richtung bezieht sich auf eine Teilströmung, die von Osten nach Westen oder umgekehrt fließt. Die vertikale Richtung ist eine Teilströmung, die von Süden nach Norden oder umgekehrt fließt. Aus den Geschwindigkeiten der beiden Teilströmungen kann dann eine Strömung mit Richtung und Geschwindigkeit errechnet werden.

Zur Anwendung der Strömung auf ein Schiff werden immer die Werte des Messpunktes genommen, welcher den geringsten Abstand zum Schiff hat. Die resultierende Strömung, die auf das Schiff einwirkt, ist dabei immer das Vektorprodukt aus der horizontalen und der vertikalen Strömungs-Komponente.

Ein auf die Strömung reagierendes Schiffsobjekt wird durch die Strömung versetzt. Abbildung 9 zeigt den Versatz durch die Strömung in Kombination mit der Schiffsbewegung, um somit die tatsächliche Schiffsposition zu errechnen. Dies ist möglich, da sowohl die Schiffsgeschwindigkeit und der Kurs als auch die Richtung der Strömung und deren Geschwindigkeit bekannt sind.

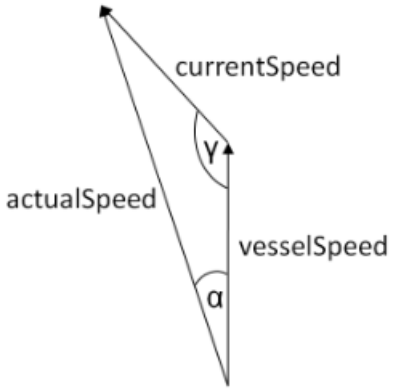

Abbildung 9: Strömungseinwirkung auf ein Schiff

- vesselSpeed [m/s] = Geschwindigkeitsvektor des Schiffes
- currentSpeed [m/s] = Geschwindigkeitsvektor der Strömung
- actualSpeed [m/s] = Resultierender, tatsächlicher Bewegungsvektor des Schiffes
- γ [°] = Innenwinkel zwischen Schiff und Strömung
- α [°] = Abdriftwinkel des Schiffes und somit resultierender Kurs

2.1.4.2.2 Wind

Windinformationen werden in der Simulation ähnlich behandelt wie Strömungsinformationen. Die Windinformationen der bereitgestellten Szenarien beruhen auf Messdaten der Forschungsplattformen in Nord- und Ostsee (FINO) [BuV14] in der Deutschen Bucht und der Ostsee. Neben einer Windrichtung und Windstärke sind auch hier Positions- und Zeitangaben vorhanden.

Anders als beim Einfluss der Strömung, reichen Richtung und Geschwindigkeit beim Wind jedoch nicht aus, um die Auswirkung auf die Bewegung eines Schiffes zu bestimmen. Wind "trägt" im Gegensatz zur Strömung nicht beliebige Objekte gleich schnell. Der Wind wirkt sich in Abhängigkeit zur Masse und zur Windangriffsfläche auf ein Objekt aus [PI79, S. 9ff]. Während der Simulation wird zu jedem Zeitpunkt von einem Schiff die Windangriffsfläche ermittelt und diese in die Berechnung mit einbezogen.

Mit den ermittelten Werten kann nun die Windkraft (F), die auf das Schiff wirkt, bestimmt werden und mit Hilfe der Formel "Kraft = Masse * Beschleunigung" erhält man durch Umstellen (Beschleunigung = Kraft / Masse) die Beschleunigung, die ein Schiff durch den Wind erfährt. Somit lässt sich ein Abtreiben des Schiffes durch Wind simulieren.

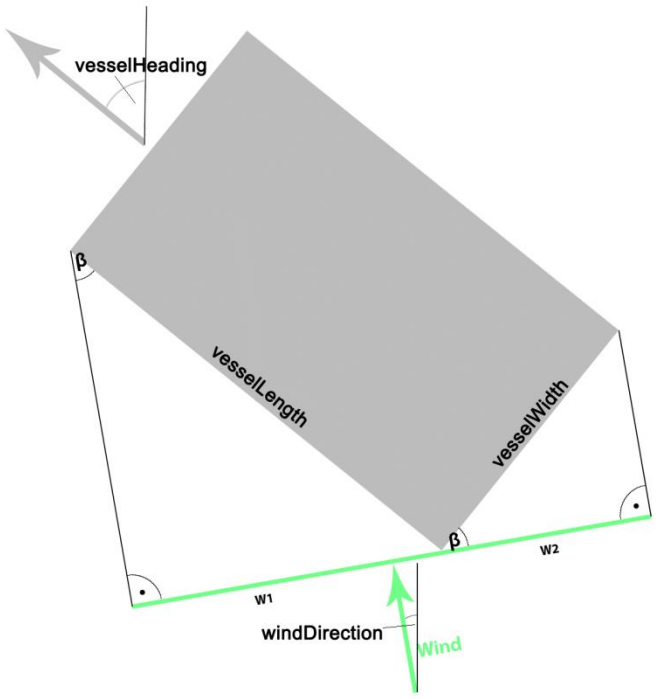

Abbildung 10: Angriffsfläche des Windes auf ein Schiff

2.1.4.2.3 Tide

Tideninformationen beschreiben in der Simulation die Höhe des Wasserstandes über der LAT (Lowest Astronomical Tide). Das Seekartenmaterial der Simulation beschreibt ohne den Umweltsimulator an allen Punkten nur den geringsten Stand des Wassers. Die Tideninformationen werden in Meter angegeben und auf den Wasserstand aufaddiert. Sie

werden dann berücksichtigt, wenn ermittelt werden soll, ob ein Schiff auf Grund läuft oder nicht. Somit lässt sich durch Manipulieren der Tide theoretisch auch das Ausbaggern einer Wasserstraße o.Ä. simulieren. Dazu kann an den fiktiv ausgebaggerten geografischen Koordinaten eine stark erhöhte Tide (z.B. zehn Meter über dem eigentlichen Wasserstand) angegeben werden (vgl. Kapitel 2.4.9.8). Beim parametrisierbaren Umweltszenario gibt es nur einen Tidenwert, der global auf den Seekarten-Wasserstand aufgerechnet wird. Bei den anderen Szenario-Typen können an verschiedenen Orten unterschiedliche Wassertiefen hinterlegt werden.

2.1.4.2.4 Wellen

Welleninformationen sind im Umweltsimulator bereits integriert, jedoch werden sie noch nicht im physikalischen Modell der MTS (s. Kapitel 2.1.3.1) berücksichtigt. Zu den Informationen der Wellen gehören die Wellenrichtung, die Wellenhöhe sowie die Peak-Periode (die Wellenperiode mit der maximalen Energie). Diese Informationen gibt es sowohl für die Dünung als auch für die Windsee. Bei der Dünung handelt es sich um Wellen, die nicht durch Wind, sondern nur durch das Ablaufen des Wassers entstehen. Bei Windsee handelt es sich um den durch Wind erzeugten Wellengang.

Aufgrund der hohen Komplexität der Daten sind diese Welleninformationen bisher nur in Szenarien enthalten, die Informationen aus den FINO-Wetterstationen enthalten. Hierzu gehören das statische und das dynamische Umweltszenario.

2.1.5 Analyse-Komponente

Die Analyse-Komponente dient der Auswertung von per HLA kommunizierten Daten, um so Aussagen über den Vorgang der verteilten Simulation treffen zu können. Insbesondere wird auf diese Weise auch ermöglicht, Statistiken über den Verlauf einer Simulation hinweg zu sammeln. Die Verarbeitung von HLA-Interactions ist entsprechend der im Projekt verwendeten Federate-Bibliothek umgesetzt.

2.1.5.1 Konzept

Dem Framework-Gedanken folgend soll die Analyse-Komponente leicht erweiterbar sein und einen generellen Lösungsansatz darstellen, Co-Simulatoren zu beobachten und auszuwerten. Es ergeben sich ähnliche Eigenschaften zu traditionellen Datenbanken und Monitoring-Systemen. In diesen werden Fragestellungen formuliert, meist in Form einer textuellen Abfragesprache, welche das System dann verarbeitet. Die Abfragen, die von der Analyse-Komponente beantwortet werden können, betrachten die Zustandsveränderungen der in der Simulation betrachteten Entitäten über die Zeit.

Federate in einer verteilten Simulation

Die Analyse-Komponente ist ein passiver Federate innerhalb der CERTI-Laufzeitinfrastruktur. Darunter ist zu verstehen, dass sie per HLA primär nur Daten entgegen nimmt und den anderen Simulationen keine Daten aktiv zur Verfügung stellt.

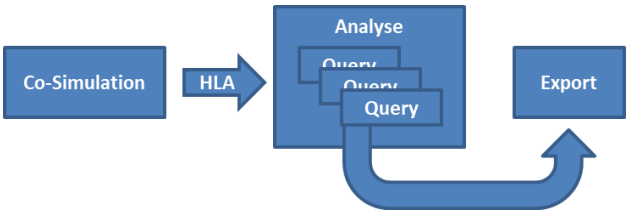

Abbildung 11: Analyse-Komponente innerhalb der verteilten Simulation

Abbildung 11 verdeutlicht diesen Zusammenhang. Die Informationen, die an die Analyse-Komponente gesendet werden, werden entsprechend der eingestellten Abfragen (Queries) ausgewertet und die daraus resultierenden Ergebnisse wiederum an die spezifischen Exporter weitergeleitet.

Queries

Der Kern der Analyse-Komponente ist die Verarbeitung von Queries auf den gesendeten Objekten. Ein Query beschreibt eine lineare Reihenfolge von Zuständen, die ein Objekt – im Kontext des HAGGIS-Datenmodells auch als Element bezeichnet – durchlaufen kann. Jedem Zustand werden hierbei Werte (Values) und Bedingungen (Conditions) zugeordnet. Hinzu kommen noch Beschreibungen, die die vorher definierten Werte dem Ergebnis-Tupel zuordnen (Mapping). Dies wird in Abbildung 12 verdeutlicht.

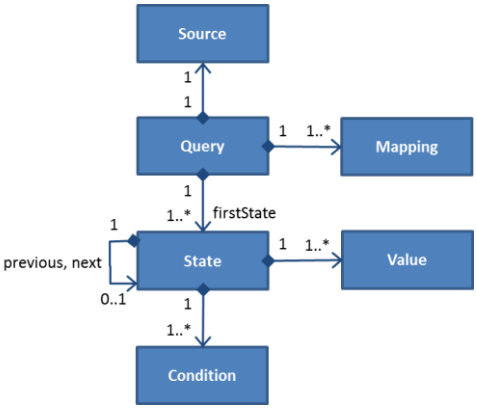

Abbildung 12: Aufbau eines Querys

Zur Überprüfung werden zur Laufzeit entsprechende Automaten für die Elemente aufgebaut. Wenn für ein Element der Endzustand des Automaten erreicht wurde, wird ein dem Query entsprechendes Rückgabe-Tupel erzeugt und an den Export übergeben. In Abbildung 13 ist dieser Verlauf schematisch dargestellt.

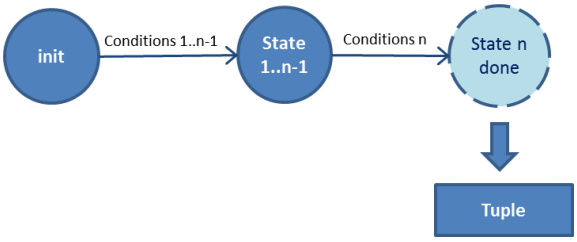

Abbildung 13: Vom Automaten eines Querys zum Tupel

Values

Pro Status im Automaten können sich mehrere Values gemerkt werden, die dann in den Conditions überprüft oder in Mappings als Tupelwert verwendet werden. Ein Value kann beispielsweise die aktuelle Geschwindigkeit oder ein Hafen sein.

StateMachines

Die Automaten oder StateMachines repräsentieren also den Laufzeitzustand eines Querys pro Element und ermitteln so, wann und ob ein Query für ein Element gilt.

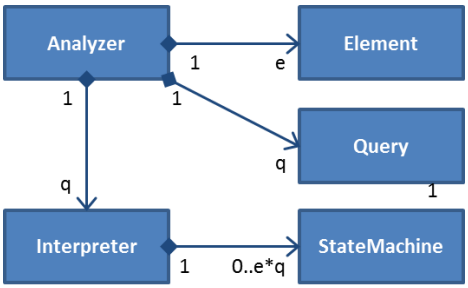

Abbildung 14: Queries, Elements und StateMachines

In Abbildung 14 ist der Zusammenhang zwischen Queries, Elements und StateMachines noch einmal zur Verdeutlichung dargestellt, *e* steht dabei für die Anzahl aller per HLA bekannten Elemente und *q* für die Anzahl der installierten Queries. Entsprechend ist die Anzahl der zur Laufzeit vorhandenen Automaten also maximal die Anzahl aller Queries multipliziert mit der Anzahl aller Elemente. Jeder installierte Query bekommt dabei einen Interpreter zugeordnet, der die zugehörigen Automaten verwaltet.

2.1.5.2 Architektur

Im vorherigen Kapitel wurden die grundlegenden Konzepte der Analyse-Komponente beschrieben. Darauf aufbauend wird in diesem Kapitel ein Überblick über die Architektur der Analyse-Komponente gegeben.

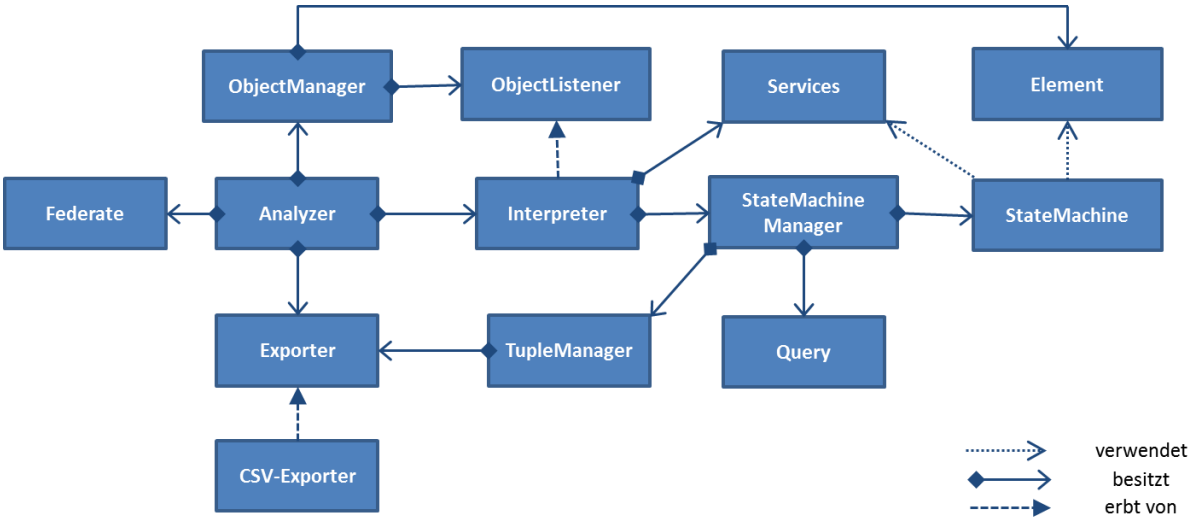

Abbildung 15: Architekturüberblick Analyse-Komponente

Abbildung 15 stellt alle wichtigen Komponenten für die Kommunikation und die Verarbeitung der Queries sowie den Export der Daten dar.

- **Analyzer:** Der Analyzer ist der Kern der Analyse-Komponente. Er führt alle Funktionalitäten zusammen. Über den Federate gelieferte HLA-Objekte werden an den ObjectManager weitergegeben. Für Queries werden die Laufzeitstrukturen aufgebaut und mit den entsprechenden Exportern verknüpft.
- **Query:** Ein Query ist eine formalisierte Beschreibung, auf die der Simulationszustand hin überprüft werden soll.
- **Federate:** Der Federate ist für die Kommunikation über den HLA-Standard verantwortlich.
- **ObjectManager:** Der ObjectManager verwaltet die aktuellen Elemente und informiert alle ObjektListener über Änderungen.
- **Interpreter:** Der Interpreter ist für die Verwaltung der StateMachineManager pro Query verantwortlich und informiert diese über entsprechende Änderungen an Elementen. Zudem stellt er Services bereit, die übergreifend für die Abarbeitung der Queries benötigt werden.
- **StateMachineManager:** Pro Query gibt es einen StateMachineManager. Dieser kümmert sich darum, die Automaten aufzubauen und stellt dem TupleManager bei Erreichen eines Endzustands das entsprechende Tupel zur Verfügung.
- **StateMachine:** Stellt die Laufzeitsituation pro Element und Query dar. Zum Prüfen eines Zustandsübergangs werden sowohl die Services als auch die Ausprägung des zugehörigen Elements benötigt.
- **TupleManager:** Der TupleManager nimmt Tupel entgegen und reicht sie an die registrierten Exporter weiter.
- **Exporter:** Exporter sind Schnittstellen, die die ermittelten Informationen persistieren oder weiterverarbeiten. Ein Beispiel hierfür ist der CSV-Exporter, der die Tupel entgegennimmt und in eine CSV-Datei schreibt.

2.1.5.3 TEU-Berechnung

Für das Anwendungsszenario zur Bewertung einer Wasserstraße anhand des Transportvolumens, war es notwendig zu ermitteln, wie viel TEU[1] ein Schiff transportieren kann. Die Berechnung muss natürlich auf den Daten basieren, die über das Schiff bekannt sind und soll einen möglichst reellen Wert liefern.

Um diesen Anforderungen gerecht zu werden, wurde auf Informationen zurückgegriffen, die der Hamburger Hafen kostenlos zur Verfügung stellt [Ha14]. Es handelt sich hierbei um Daten über Containerschiffe, die im Hafen geankert haben. Diese sind in sechs Gruppen unterteilt. Jeweils diejenigen zehn Schiffe jeder Gruppe, die zuletzt gesichtet wurden, stellen die Zahlenbasis zum Ermitteln der Ladekapazität-Funktion dar.

Generell gilt:

$$Fakt = BewertungsFunktion(\overrightarrow{DimensionsWerte})$$

Übertragen auf die TEU-Berechnung:

$$Ladekapazität = TEUBerechnung(\overrightarrow{Schiffseigenschaften})$$

Das genutzte Datenmaterial sowie die vorhandenen Konzepte und Datentypen innerhalb der Schiffsrepräsentation im HAGGIS-Datenmodell überschneiden sich im Hinblick auf die folgenden Eigenschaften:

- Länge des Schiffes
- Maximale Breite des Schiffes
- Maximaler Tiefgang

Aus diesen Werten wurden folgende virtuelle Größen abgeleitet:

- Fläche: $Fläche = Länge * MaximaleBreite$
- Volumen: $Volumen = Fläche * MaximalerTiefgang$

Mit Hilfe von MS Excel wurden Regressionsfunktionen mit dem besten Bestimmtheitsmaß für $Länge \rightarrow Ladekapazität$, $Fläche \rightarrow Ladekapazität$ und $Volumen \rightarrow Ladekapazität$ ermittelt. Diese sind in den nachfolgenden drei Abbildungen visualisiert:

[1] Twenty-foot Equivalent Unit, zu Deutsch: Standard-Container

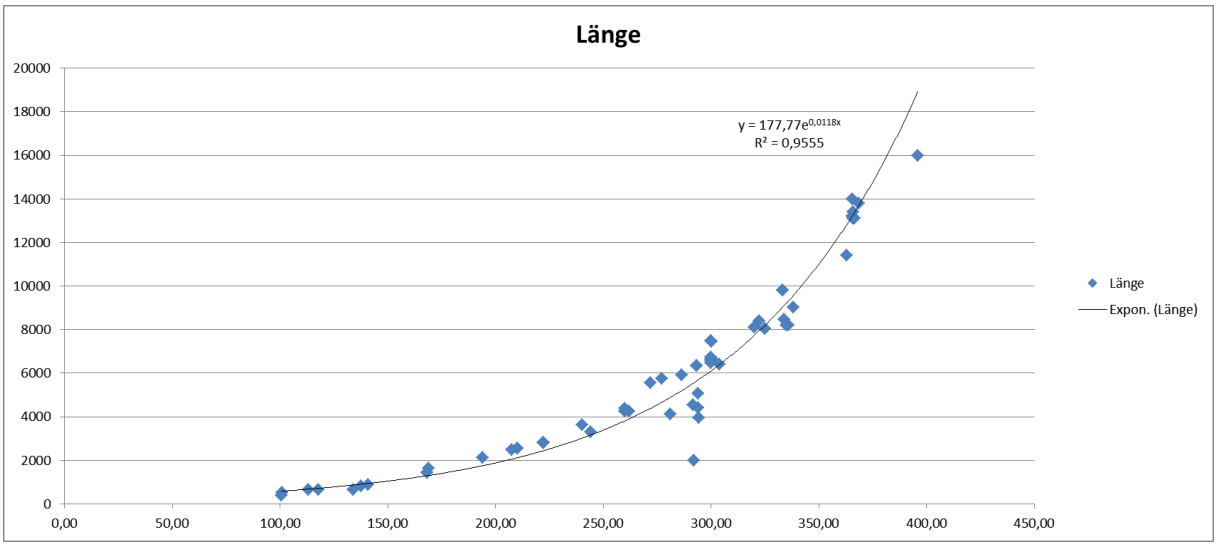

Abbildung 16: Regressfunktion zu Ladekapazität = TEU Berechnung(Länge)

In Abbildung 16 gibt es einige größere Abweichungen. Die ermittelte Funktion approximiert jedoch schon relativ gut.

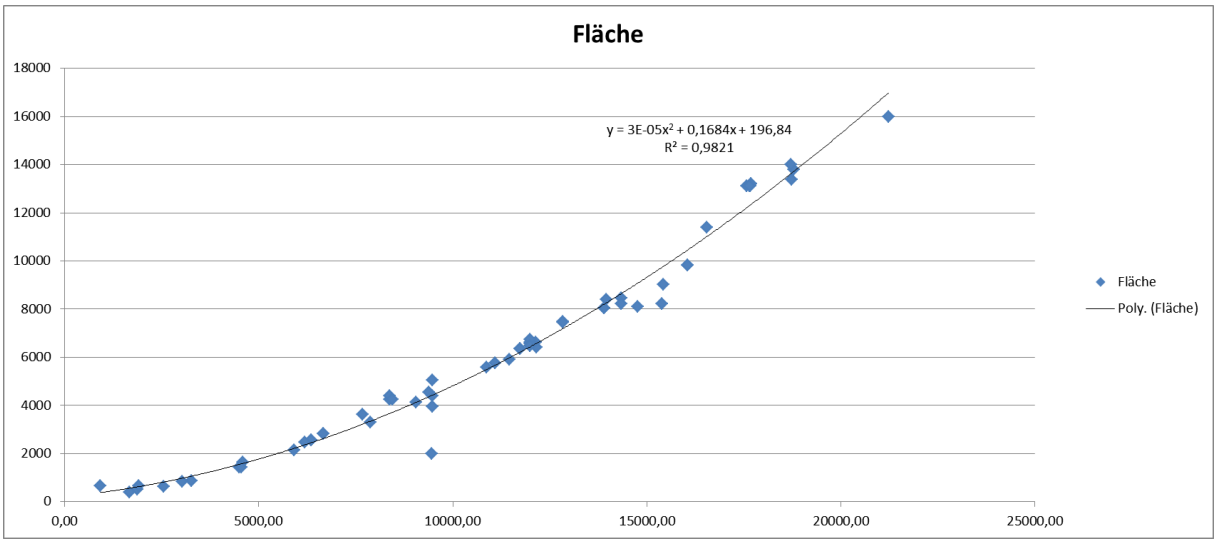

Abbildung 17: Regressfunktion zu Ladekapazität = TEU Berechnung(Fläche)

Wie in Abbildung 17 ersichtlich wird, eignet sich die Funktion basierend auf der Fläche schon deutlich besser. Auffallend ist auch, dass der Trend durch die polynomielle Annäherung besser abgebildet wird als durch die Exponentialfunktion.

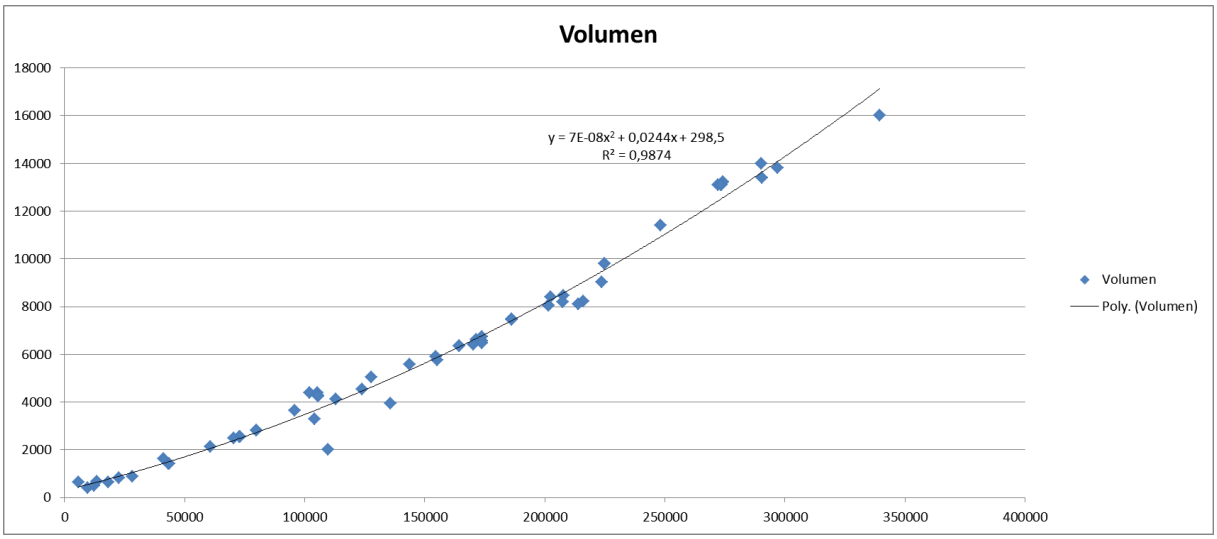

Abbildung 18: Regressfunktion zu Ladekapazität = TEU Berechnung(Volumen)

Abbildung 18 zeigt die beste Funktion, die ermittelt werden konnte.

$$f(Volumen[m^3]) = 0{,}00000007\left[\frac{TEU}{m^6}\right] * Volumen^2 + 0{,}0244\left[\frac{TEU}{m^3}\right] * Volumen + 298{,}5[TEU]$$

2.2 Semantisches Datenmodell HAGGIS

Das HAGGIS-Datenmodell hat die Aufgabe, eine einheitliche formale Semantik zu beschreiben. Es stellt Datentypen und Konzepte bereit, um eine gemeinsame Grundlage für die Kommunikation und den Datenaustausch im Bereich maritimer Systeme und Simulationen zu schaffen.

Der Aufbau des Datenmodells ist in Ebenen strukturiert. Je nach Zuordnung können die beschriebenen Konzepte sehr allgemein oder simulationsspezifisch sein. Zur weiteren Abgrenzung ist innerhalb der einzelnen Ebenen eine Einteilung in unterschiedliche Teilmodelle umgesetzt. Diese Einteilung gewährleistet eine strikte Trennung der semantischen Verantwortlichkeiten und ermöglicht eine einfache Erweiterung des Datenmodells und eine schnelle Wiederverwendung beim Umsetzen neuer Systeme. Das HAGGIS-Datenmodell stellt also einen gemeinsamen Standard dar und sichert so die Interoperabilität von Simulatoren, was besonders bei der Umsetzung von verteilten Simulationen notwendig ist.

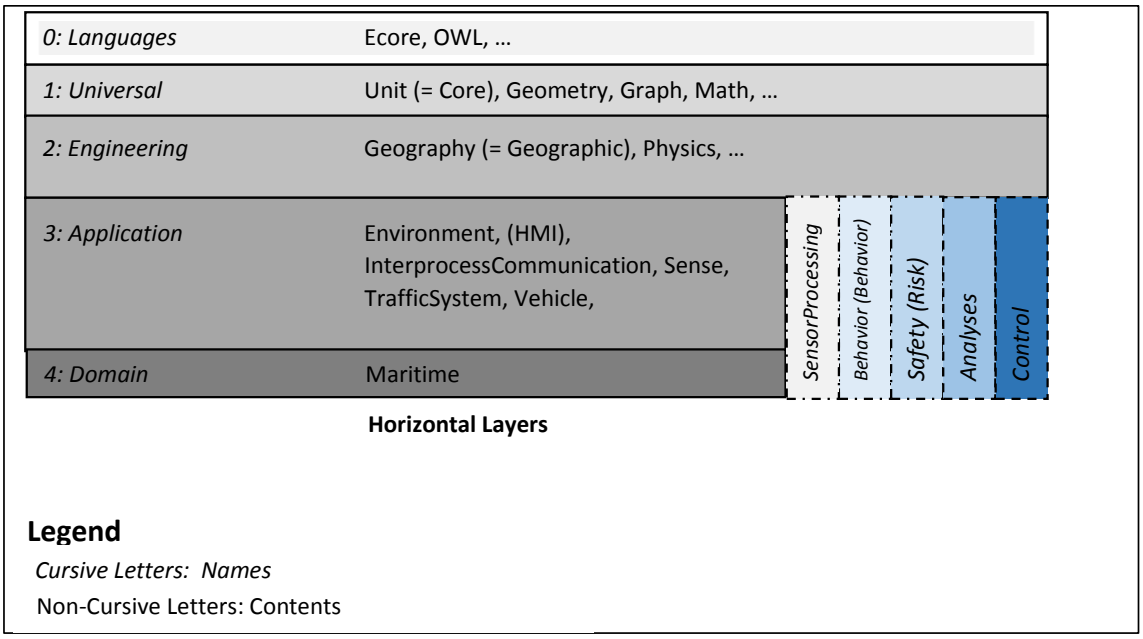

Abbildung 19: Aufbau des HAGGIS-Datenmodells

In Abbildung 19 ist der Schichtenaufbau des HAGGIS-Datenmodells dargestellt. Mit aufsteigender Schichtnummer werden die Teilmodelle immer konkreter bzw. weniger universell. Zusätzlich gibt es noch Ebenen-übergreifende Modelle, deren Konzepte sowohl Ebene drei als auch Ebene vier zugeordnet werden können.

Zur Modellierung der Datenstrukturen wird Ecore verwendet. Hierbei kommt zur textuellen Beschreibung die domänenspezifische Sprache YML zum Einsatz, welche in die Ecore-Modelle transformiert wird.

Auch die MTS verwendet die Strukturen des HAGGIS-Datenmodells, sowohl für die Kommunikation über HLA als auch intern zur Datenhaltung. Innerhalb des Entwicklungsprozesses der MTS hat sich ergeben, dass einige Erweiterungen am HAGGIS-Datenmodell vorgenommen werden müssen, um den eigenen Anforderungen gerecht zu werden.

Für die Simulation des Schiffverkehrs wurden Konzepte eingeführt, die ein Schiff genauer beschreiben. Hierzu gehören die Motorleistung, die Ladungskapazitäten sowie physikalische Einflüsse und die Zuordnung von Agenten zur Steuerung. Zudem kann die Simulation auch verschiedene Ereignisse (z.B. Kollisionen) kommunizieren, die ebenfalls als neue Konzepte eingeführt wurden.

Bei der Umsetzung der Agenten und der Nachrichten, die diese senden und empfangen können, wurden auch Möglichkeiten zur Definition des Agentenverhaltens hinzugefügt.

Die Entwicklung des Umweltsimulators hat das Hinzufügen verschiedener Umwelteinflusse zum Datenmodell bedingt. Die Analyse-Komponente hingegen wertet bereits vorhandene Strukturen im Datenmodell dynamisch aus und benötigt daher keine Erweiterungen.

2.3 Benutzeroberfläche

Im Folgenden wird die Visualisierung der MTS beschrieben. Nach Starten der Simulation erscheint das Hauptfenster (s. Abbildung 20). Dieses lässt sich in die fünf Bereiche Menüleiste (Abbildung 20, Punkt 1), Leiste „MASON-spezifische Eigenschaften“ (Punkt 2), Karten- und Simulationsfenster (Punkt 3), MASON-Konsole (Punkt 4) und Status-Logging-Fenster (Punkt 5) einteilen, welche nachfolgend näher beschrieben werden.

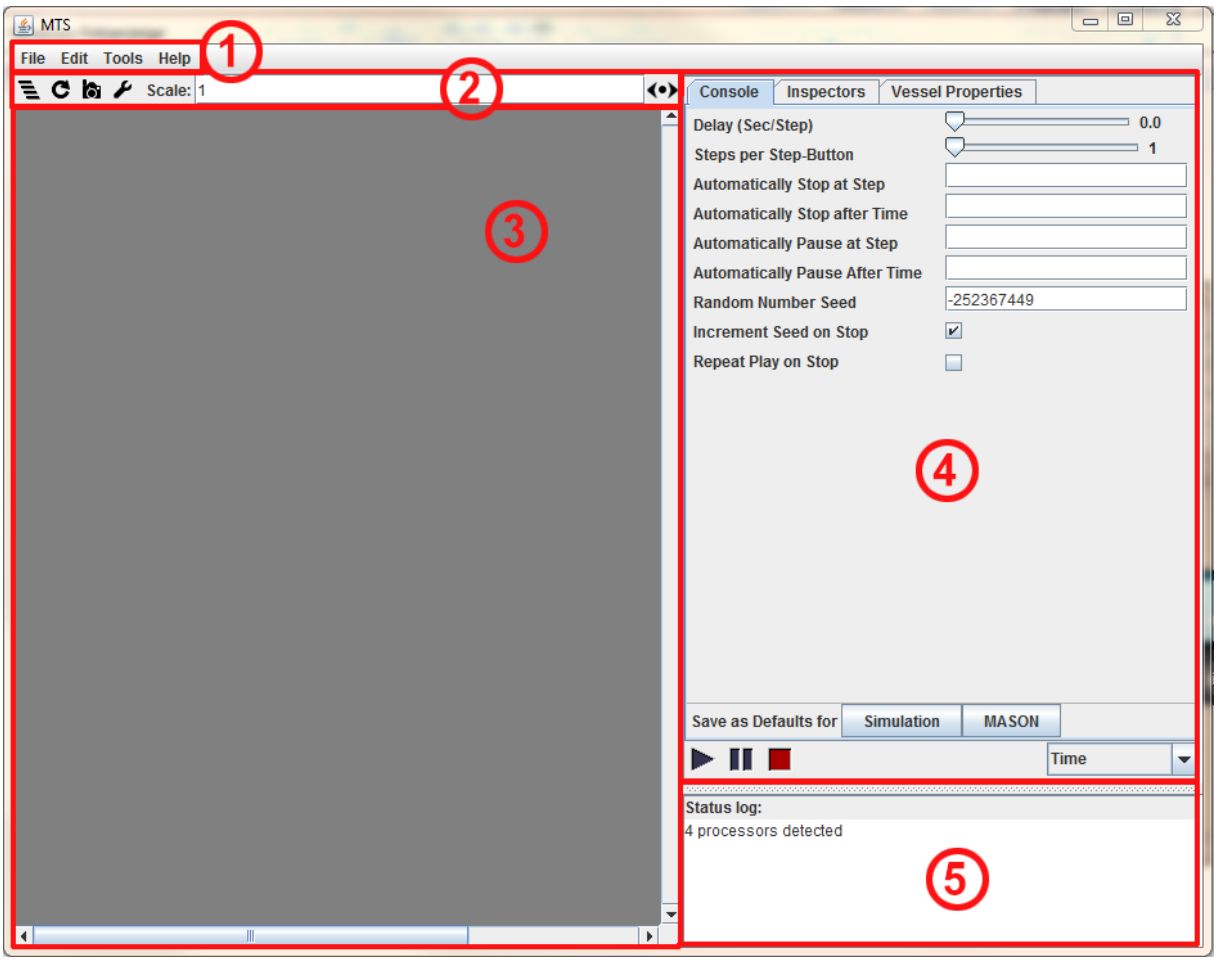

Abbildung 20: Hauptfenster der MTS

2.3.1 Menüleiste

Im oberen Bereich des Hauptfensters (Punkt 1 in Abbildung 20) wird die Menüleiste dargestellt. Diese beinhaltet die Menüpunkte „File“, „Edit“, „Tools“ und „Help“.

Mit Hilfe des Menüpunkts „File“ können neue Simulationsszenarien erstellt, gespeichert oder beendet werden, wie in Abbildung 21 dargestellt. Beim Speichern der Simulation wird eine Java-Serialisierung verwendet. Bei den Ecore-Objekten wird eine EMod-Datei generiert und diese danach serialisiert.

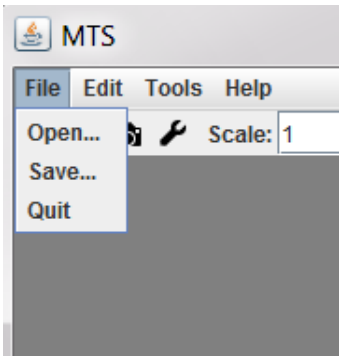

Abbildung 21: Menüpunkt "File"

Der Menüpunkt „Edit“ beinhaltet den „Scenario editor“. Dieser dient zum Konfigurieren der Schiffe in der Simulation. Somit ist es möglich, an einem zentralen Punkt alle zu simulierenden Schiffe zu konfigurieren. Zu den konfigurierbaren Eigenschaften gehören das Laden von Szenarien sowie das zufällige Erstellen von Szenarien. Abbildung 22 zeigt den Menüpunkt „Edit“.

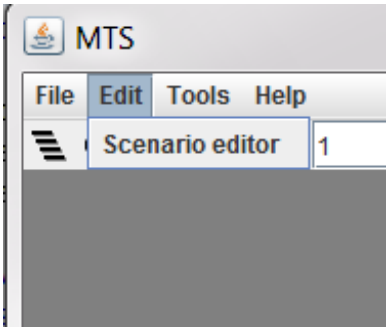

Abbildung 22: Menüpunkt "Edit"

Mit Öffnen des „Scenario editor“ erscheint das Fenster aus Abbildung 23. Zum einen kann/ können eine oder mehrere EMod-Datei(en) eingelesen (Abbildung 23, Punkt 1) oder erstellt werden (Punkt 2). Außerdem kann die automatische Generierung eines Szenarios mit Hilfe des LoadGenerator durchgeführt werden (Punkt 3). Diese drei Möglichkeiten können auch miteinander kombiniert werden.

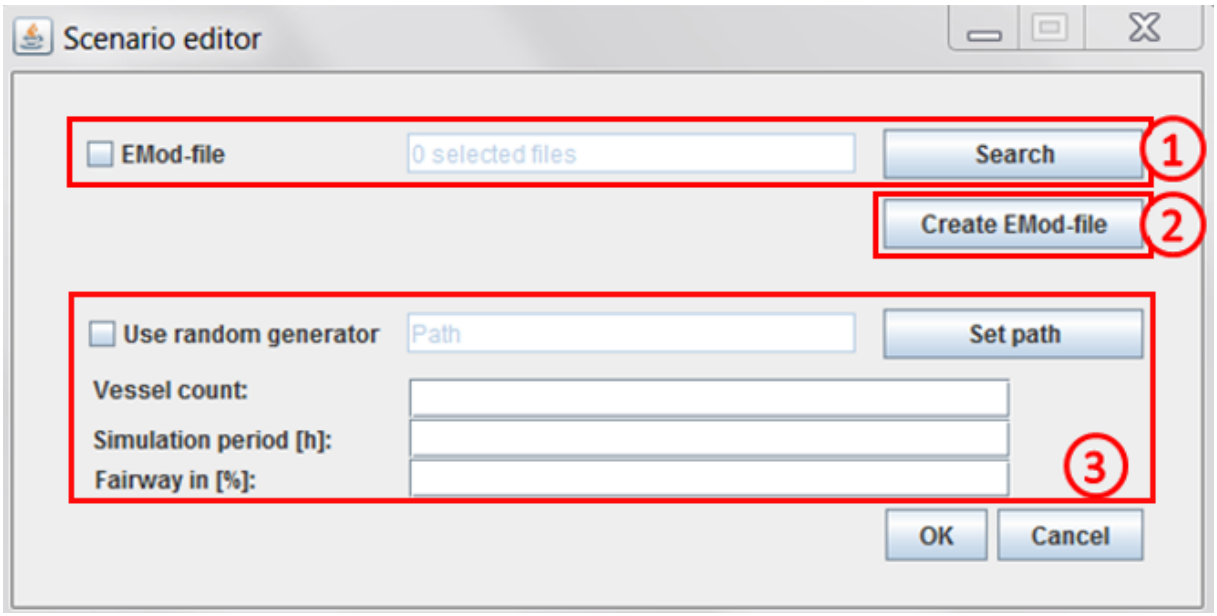

Abbildung 23: Fenster „Scenario editor“

Ist die Checkbox „EMod-file“ aktiviert, kann/ können über den „Search“-Button eine oder mehrere lokale EMod-Datei(en) ausgewählt werden.

Es können über den LoadGenerator auf schnelle Art und Weise eine größere Menge von Schiffen erzeugt werden (Abbildung 23, Punkt 3). Hierfür muss die Checkbox „Use random generator“ aktiviert sein. Anschließend müssen die Anzahl der Schiffe insgesamt („Vessel count“), die zu simulierende Zeit in Stunden („Simulation period [h]“) und eine Prozentangabe zum Anteil von Schiffen an der Menge aller Schiffe, die im Fahrwasser positioniert sein sollen, eingegeben werden. Anschließend können diese Einstellungen in eine EMod-Datei gespeichert werden („Set path“). Als Datenquelle für den LoadGenerator dient eine CSV-Datei, welche ca. 800 Schiffe mit den dazugehörigen Eigenschaften wie MMSI, Name, sowie Länge und Breite, beinhaltet.

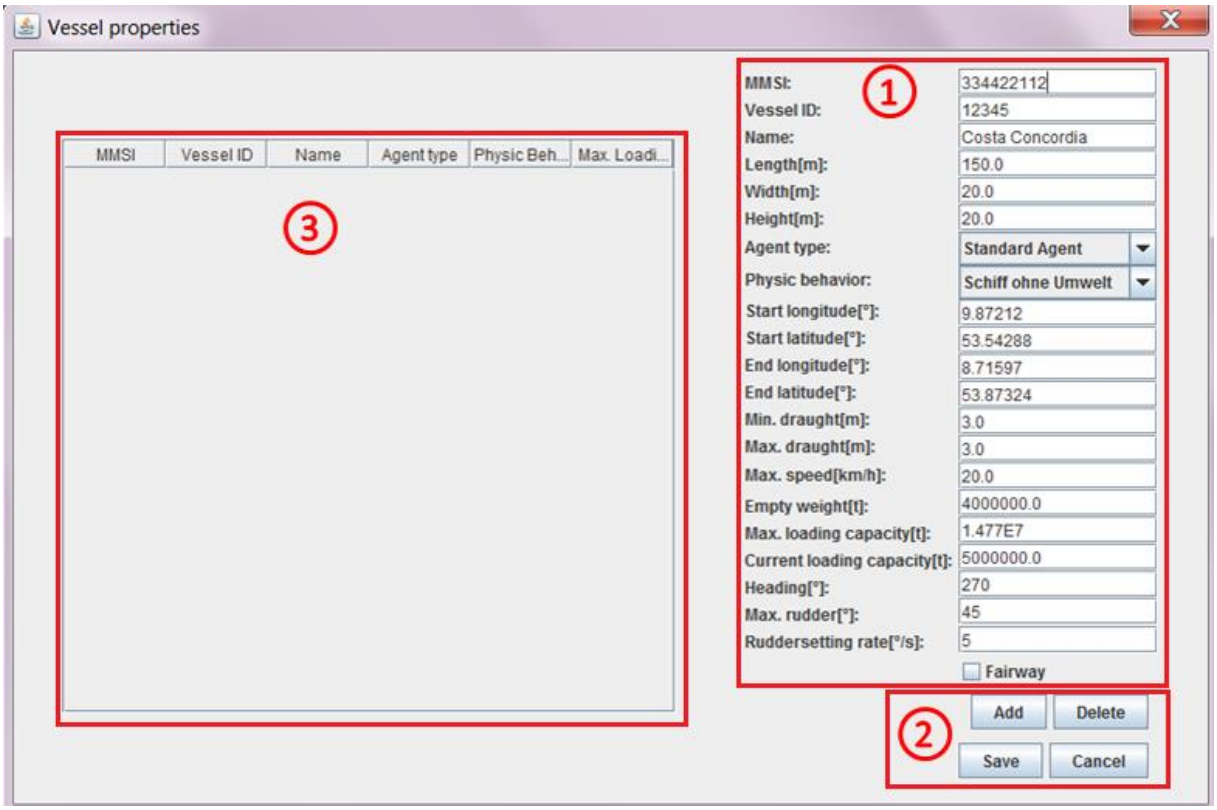

Abbildung 24: Scenario editor – Fenster „Vessel properties“

Beim Erstellen einer neuen EMod-Datei (Punkt 2 aus Abbildung 23) öffnet sich das Fenster aus Abbildung 24. Im rechten Bereich (Abbildung 24, Punkt 1) werden die einzelnen Schiffseigenschaften („Vessel properties“) eingestellt.

Der zweite Bereich (Punkt 2) beinhaltet Buttons zum Hinzufügen eines Schiffes zum Szenario gemäß der eingegebenen Schiffseigenschaften („Add“). Hinzugefügte Schiffe tauchen in einer Liste auf, die im linken Bereich des Fensters (Punkt 3) angezeigt werden. Der „Delete“-Button löscht bereits bestehende Schiffe aus der Liste. Des Weiteren können die Daten gespeichert („Save“) oder der ganze Vorgang abgebrochen werden („Cancel“).

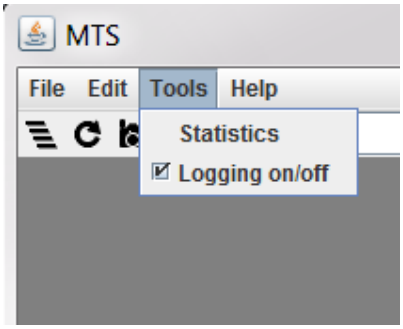

Abbildung 25: Menüpunkt "Tools"

Unter dem Menüpunkt „Tools“ ist der Button zum Aufrufen der Statistik zu finden sowie eine Checkbox zum Ein- bzw. Ausschalten der Logging-Funktion (s. Abbildung 25).

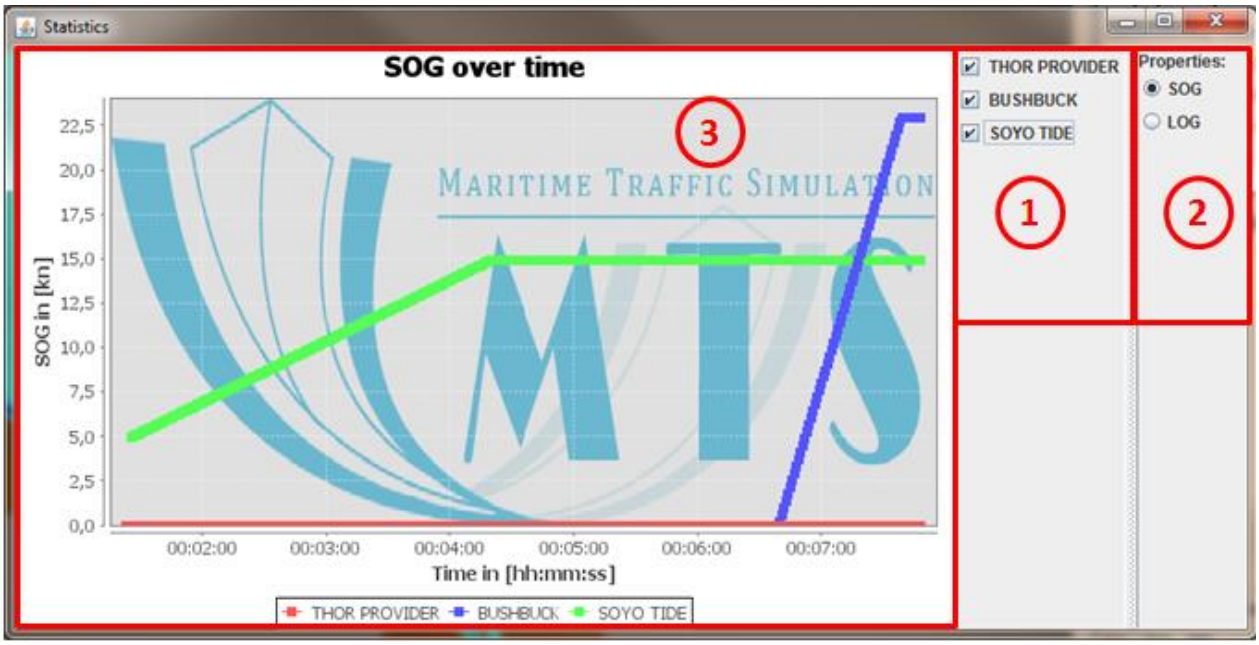

Abbildung 26: Fenster „Statistics“

Das Statistik-Fenster ist in Abbildung 26 dargestellt. In Bereich 1 (Abbildung 26, Punkt 1) werden die Schiffe, welche zum Simulationslauf existieren, aufgelistet. Über die Checkbox können die Statistiken einzelner Schiffe zur grafischen Darstellung in Bereich 3 (Punkt 3) hinzugefügt bzw. entfernt werden.

In Bereich 2 (Punkt 2) lässt sich auswählen, welche Eigenschaften („Properties“) im dritten Bereich (Punkt 3) angezeigt werden sollen. Hierbei kann zwischen LOG[2] (Fahrt durchs Wasser) oder SOG (Speed Over the Ground, Geschwindigkeit über Grund) gewechselt werden. Die Fahrt durchs Wasser (LOG) ist die Geschwindigkeit eines Schiffs relativ zur befahrenen Wassermasse. Bei der Geschwindigkeit über Grund (SOG) werden zusätzlich Strömung und Wind mit berücksichtigt [Sa04].

In Bereich 3 (Punkt 3) werden die Ausprägungen der ausgewählten Eigenschaft jedes der ausgewählten Schiffe über den Simulationszeitraum dargestellt.

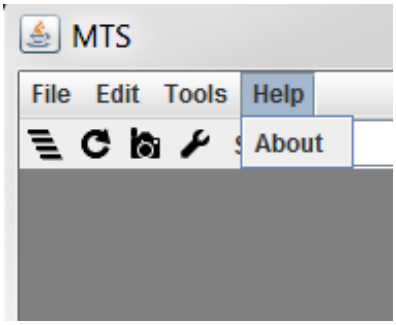

Abbildung 27: Menüpunkt "Help"

[2] Das LOG ist ein Messgerät zur Ermittlung der Fahrt durchs Wasser eines Schiffes.

Der Menüpunkt „Help“ beinhaltet den Unterpunkt „About“ (s. Abbildung 27). Mit Klicken des Unterpunktes öffnet sich die Webseite der PG MAPS.

2.3.2 Leiste „MASON-spezifische Eigenschaften“

Abbildung 28: Leiste „MASON-spezifische Eigenschaften“

Der zweite Bereich des Hauptfensters (Punkt 2 in Abbildung 20) ist eine Leiste, die die MASON-spezifischen Eigenschaften beinhaltet, mit der sich Änderungen an der grafischen Darstellung der Simulation durchführen lassen.

Im Bereich ganz links (Abbildung 28, Punkt 1) befindet sich eine Ansammlung von Buttons. Über den ersten Button von links lassen sich unterschiedliche grafische Ebenen (Layer) zur Karte hinzufügen oder von ihr entfernen (z.B. die Anzeige von Tiefeninformationen auf der Karte, Strömungsverläufe oder die Betonnung). Über den zweiten Button lässt sich einstellen, wann und wie das Display neu gezeichnet werden soll. Der dritte Button erlaubt das Aufnehmen eines Snapshots des aktuellen Kartenausschnitts, welcher im PNG- oder PDF-Format gespeichert werden kann. Der letzte Button der Reihe erlaubt die Einstellungen weiterer Parameter für die Darstellung der Simulation, wie z.B. Tooltips oder ein Raster.

Der Bereich rechts daneben (Abbildung 28, Punkt 2) bietet dem Nutzer die Zoom-Funktionen. Über die pfeilförmigen Buttons kann die Karte näher herangezoomt (Pfeil nach rechts) oder weiter weggezoomt werden. Der Punkt in der Mitte der Pfeile setzt den Zoom wieder auf hundert Prozent (Standard) zurück.

2.3.3 Karten- und Simulationsfenster

Im dritten Bereich des Hauptfensters (Punkt 3 in Abbildung 20) wird die Simulation grafisch auf dem entsprechenden Kartenausschnitt abgebildet und animiert. Es ist möglich mit Hilfe des Mausrades in die Karte hinein- bzw. aus der Karte herauszuzoomen. Entsprechend des Zoomfaktors ändert sich der Detaillierungsgrad der Karte. So werden ab Zoomfaktor sechs die Betonnung und ab Zoomfaktor elf die Tiefeninformationen des Kartenausschnittes sichtbar.

Des Weiteren kann die Karte per „Drag&Move“ verschoben bzw. navigiert werden. Zusätzlich kann auch die Scrollbar am rechten und unteren Rand der Karte zum Navigieren genutzt werden.

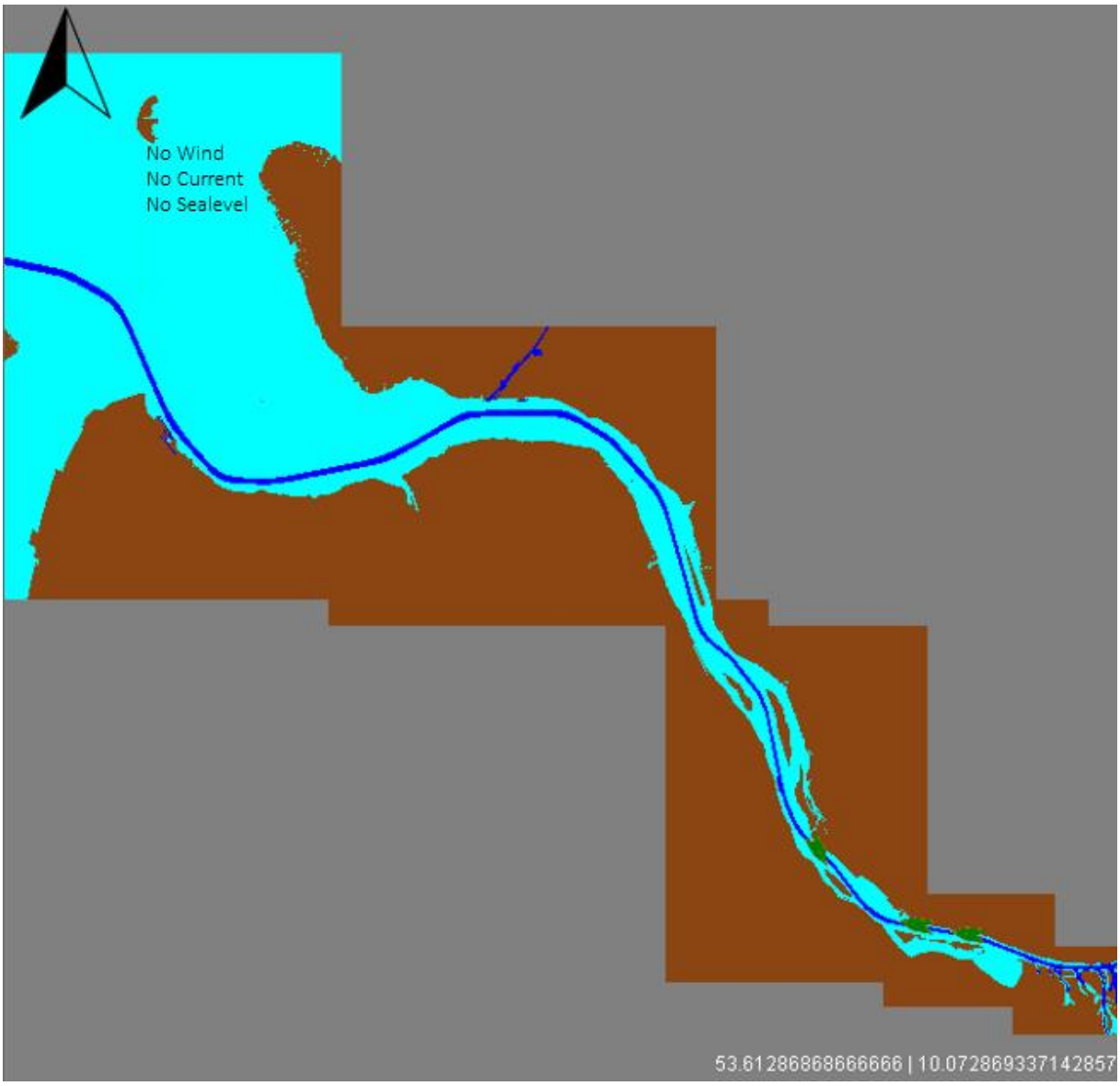

Abbildung 29: Karten- und Simulationsfenster mit Kartenausschnitt

Auf der Karte in Abbildung 29 werden die vorhandenen Schiffe grün dargestellt. Im oberen Bereich werden zudem ein Pfeil (dieser zeigt nach Norden zur Kenntlichmachung der Himmelsrichtungen), Wind, Strömung und die Tide angezeigt. In diesem Beispiel wurden keine Umweltdaten zur Simulation verwendet, daher werden die Werte „No Wind“, „No Current“ und „No Sealevel“ ausgegeben. Im unteren Bereich wird der Mittelpunkt des Kartenausschnittes in Längen- und Breitengrad angezeigt.

2.3.4 MASON-Konsole

Der vierte Bereich des Hauptfensters (Punkt 4 in Abbildung 20) beinhaltet die MASON-Konsole. Hierzu gehören die simulationsspezifischen Eigenschaften (Reiter „Console“), der

Inspektor (Reiter „Inspectors“), die Schiffseigenschaften (Reiter „Vessel Properties“) sowie die Simulationssteuerungs-Konsole am unteren Rand, wie in Abbildung 30 zu sehen ist.

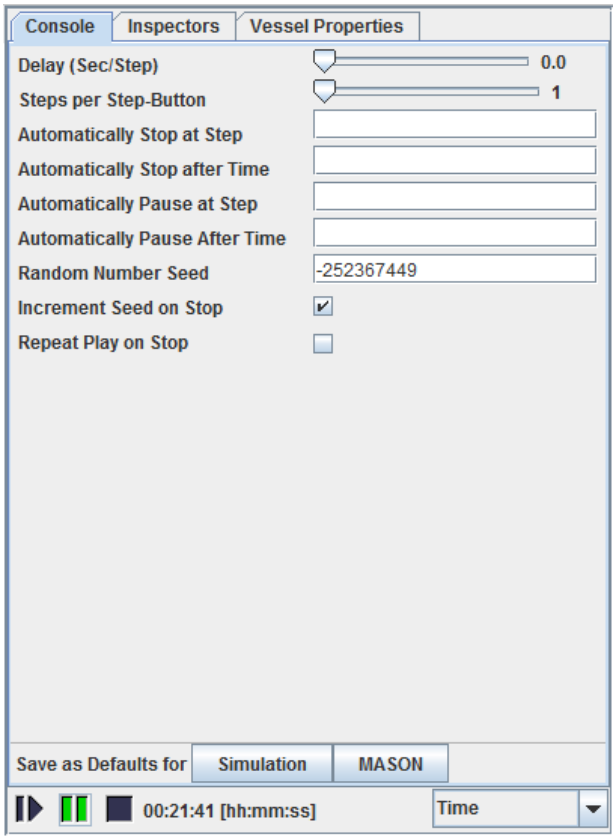

Abbildung 30: MASON-Konsole – Reiter „Console“

Zu den simulationsspezifischen Eigenschaften gehören z.B. ein Regler zur Einstellung der Verzögerung der Simulation (Sekunden pro Simulationsschritt) sowie zum Einstellen, wie viele Schritte pro Betätigen des Schritt-Buttons (das ist der Play-Button im Pause-Modus) übersprungen werden sollen. Außerdem gibt es Eingabefelder zur Definition, ob und nach welcher Zeit die Simulation automatisch gestoppt oder pausiert werden soll. Die simulationsspezifischen Eigenschaften können entweder für das konkrete Simulationsszenario oder MASON-spezifisch (global) gespeichert werden.

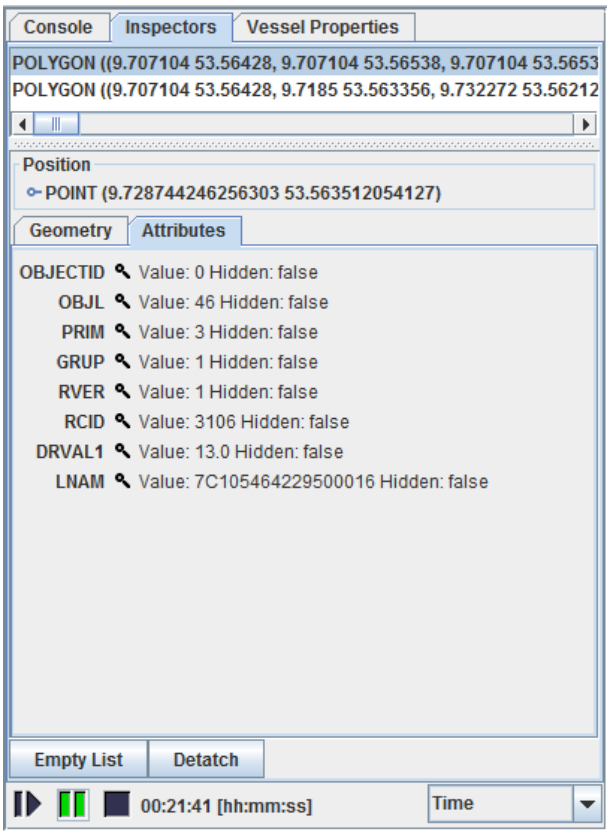

Abbildung 31: MASON-Konsole – Reiter „Inspectors“

Abbildung 31 zeigt den Inspektor. Dieser beinhaltet Informationen über den Punkt auf der Wasserfläche, der per Doppelklick mit der linken Maustaste auf der Karte angeklickt wurde. Zu diesen Informationen gehören beispielsweise die Position des Punktes (Längen- und Breitengrad), die Objekt ID sowie der Name des dazugehörigen Layer. Mit dem Button „Empty List“ können die aktuell angezeigten Informationen des Inspektors zurückgesetzt werden und mit dem Button „Detatch“ wird der Inspektor in einem neuen Fenster geöffnet.

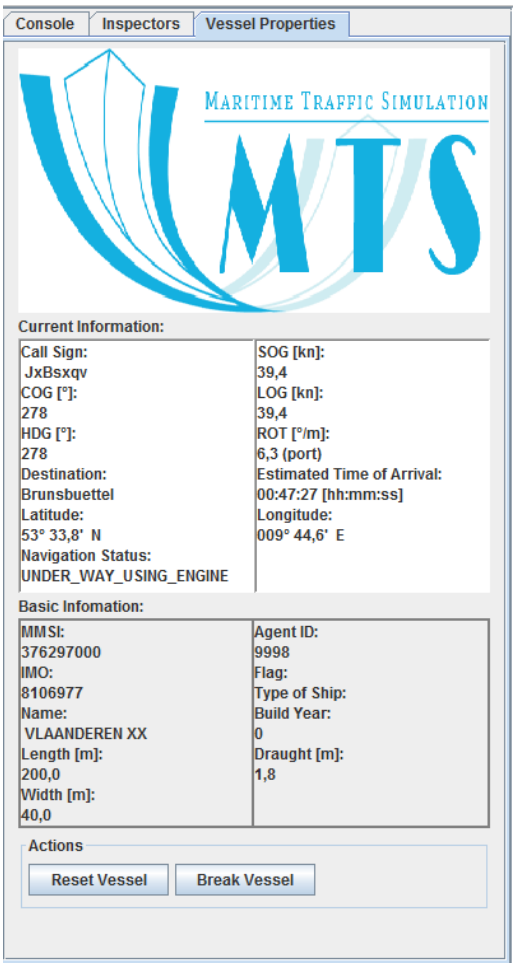

Abbildung 32: MASON-Konsole – Reiter „Vessel Properties“

Der Reiter „Vessel Properties“ dient der Anzeige von Eigenschaften von Schiffen, die in der Karte angeklickt wurden (vgl. Abbildung 32). Diese werden pro Schiff angezeigt und zur Laufzeit aktualisiert. Sofern vorhanden, wird das Foto eines Schiffes im oberen Bereich angezeigt, ansonsten erscheint das MTS-Logo.

Es wird zwischen „Current information“ und „Basic information“ unterschieden. Erstere beschreiben den aktuellen Ort, an dem sich ein Schiff befindet und mit welchen Eigenschaften es sich gerade bewegt, z.B. SOG und ROT (Rate Of Turn – die Geschwindigkeit, mit der ein Schiff sein Heading ändert in Grad pro Sekunde). Diese Informationen ändern sich zur Laufzeit. „Basic information“ beinhalten im Gegensatz dazu diejenigen Informationen eines Schiffs, welche sich nicht zur Laufzeit ändern (z.B. den Namen des Schiffes).

Zusätzlich gibt es noch zwei Buttons. Der Erste ist zum Zurücksetzen eines Schiffes gedacht, in Zuge dessen das Schiff an seine ursprüngliche Ausgangsposition zurückgesetzt wird („Reset Vessel“). Mit dem zweiten Button kann am Schiff ein Motorschaden simuliert werden, so dass es nach und nach an Geschwindigkeit verliert bis es zum Halten kommt und nicht mehr weiter fahren kann („Break Vessel“). Der injizierte Motorschaden kann mit einem weiteren Klick auf denselben Button wieder rückgängig gemacht werden.

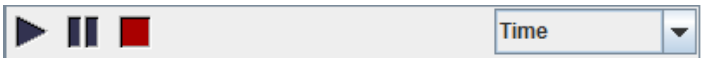

Abbildung 33: Simulationssteuerungs-Konsole – Starten, Pausieren, Stoppen

In Abbildung 33 wird die Simulationssteuerung-Konsole, der Bereich zum Starten, Pausieren und Stoppen der Simulation angezeigt.

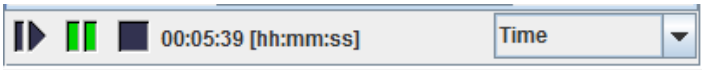

Abbildung 34: Simulationssteuerungs-Konsole – verstrichene Zeit

Abbildung 35: Simulationssteuerungs-Konsole – Simulationsschritte

Über die Auswahlbox auf der rechten Seite der Konsole kann eingestellt werden, ob die verstrichene Zeit (wie in Abbildung 34) oder die Anzahl der verstrichenen Simulationsschritte (wie in Abbildung 35) angezeigt werden sollen.

2.3.5 Status-Logging-Fenster

In Abbildung 36 ist das Status-Logging-Fenster zu sehen (Punkt 5 in Abbildung 20). Dieses zeigt z.B. den Status der Wegfindung der Schiffe an, oder deren Startzeitpunkt. Insgesamt soll dieses Fenster den Nutzer über den aktuellen Zustand der MTS informieren.

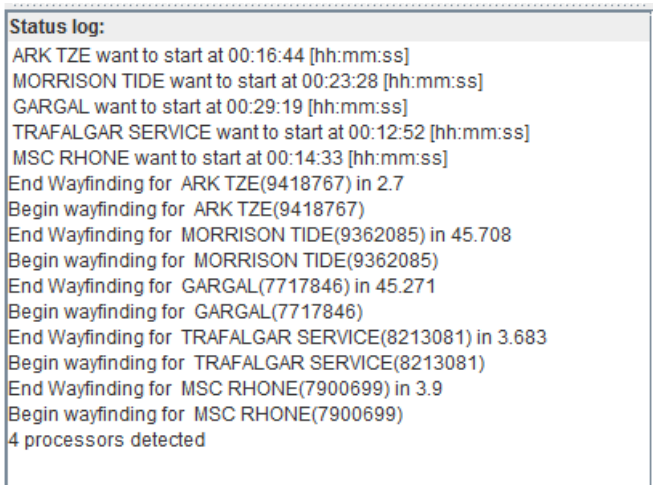

Abbildung 36: Status-Logging-Fenster

2.4 Funktionsübersicht

2.4.1 Kartenausschnitt laden

Wie bereits in Kapitel 2.1 erläutert wurde, werden für die Visualisierung der maritimen Seegebiete, ENCs im S-57-Standard in das Shapefile-Format überführt. Zu diesem Zweck wurde das Geoinformationssystem ArcGIS verwendet. Um mit ArcGIS S-57-ENCs laden zu können, muss die Erweiterung für S-57-Karten installiert sein. Eine kostenlose Testversion des Geoinformationssystems ArcGIS kann auf der Homepage des Unternehmens ESRI heruntergeladen werden, wobei zunächst eine Registrierung notwendig ist [ES14]. Auch die Erweiterung für Karten nach S-57-Standard steht dort zum kostenlosen Download zur Verfügung. Im Folgenden wird die Konvertierung der Karten beschrieben.

Im ersten Schritt wird die Anwendung ArcMap, die Teil der ArcGIS Softwareplattform ist, geöffnet. Nachdem die Anwendung die Gültigkeit der Lizenz überprüft hat, ist das Fenster aus Abbildung 37 sichtbar.

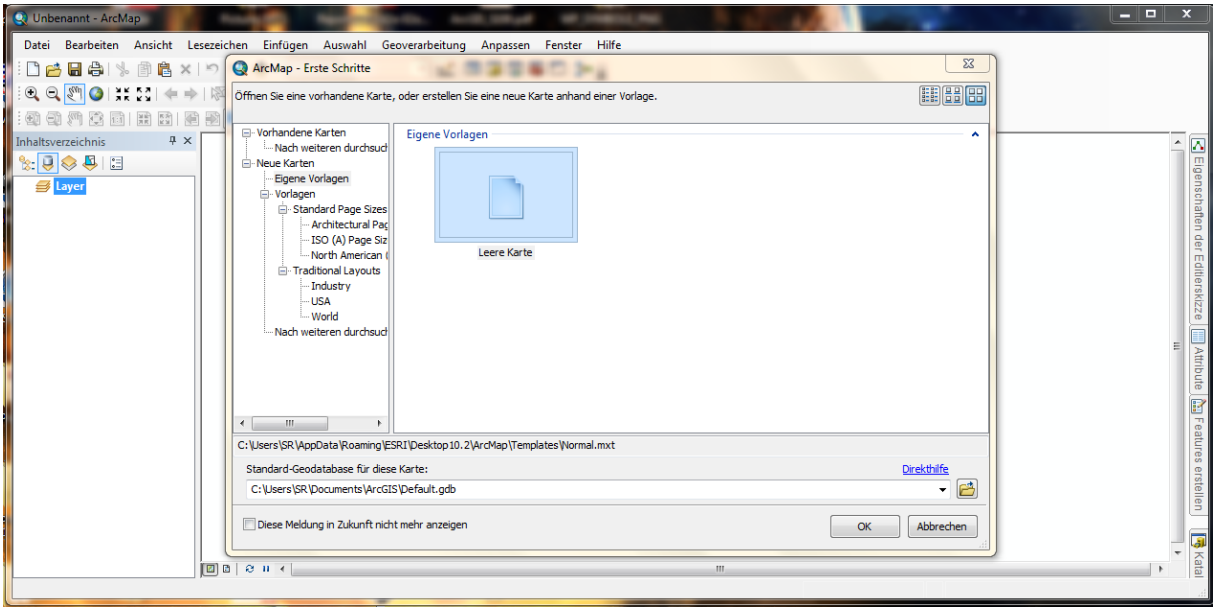

Abbildung 37: ArcMap – Start der Anwendung

Zu Beginn wird eine leere Karte geöffnet, somit kann einfach mit Betätigen von „OK“ bestätigt werden. Daraufhin ist das ArcMap-Editorfenster zu sehen (s. Abbildung 38). In das leere Dokument sollen nun die Seekartendaten aus einem S-57-konformen Kartenausschnitt geladen werden. Hierfür muss auf das Symbol geklickt werden, auf welches der rote Pfeil in Abbildung 38 zeigt.

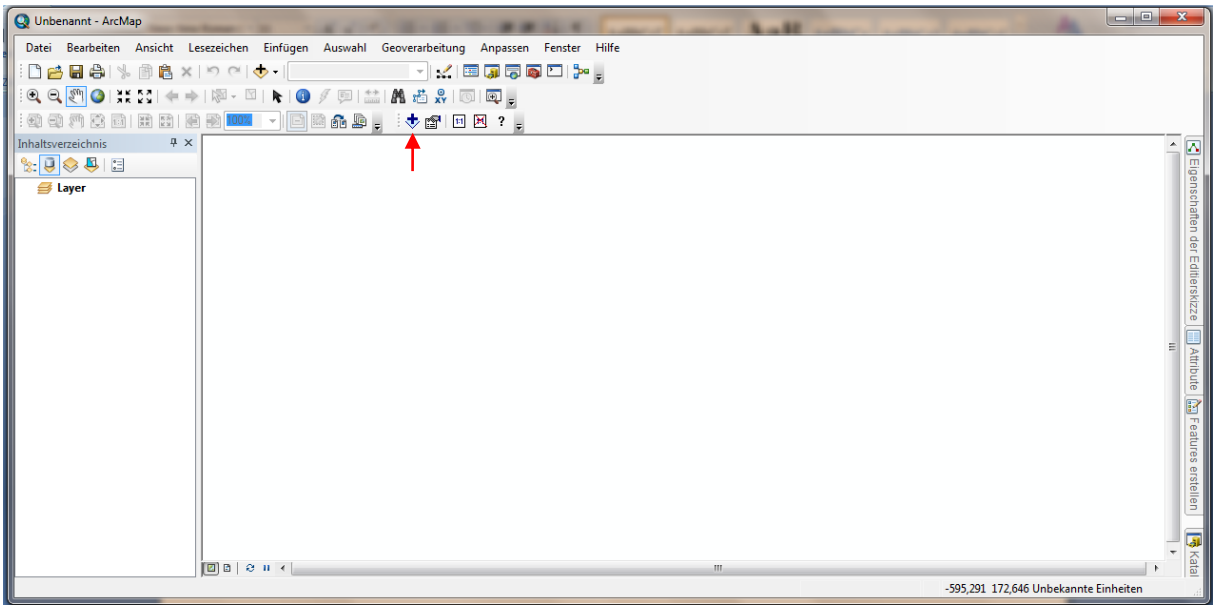

Abbildung 38: ArcMap – Laden einer S-57-Datei

Im sich öffnenden Fenster kann nun eine S-57-Datei geladen werden. Nach Öffnen dieser Datei sollte der ArcMap-Editor ähnlich wie der in Abbildung 39 aussehen.

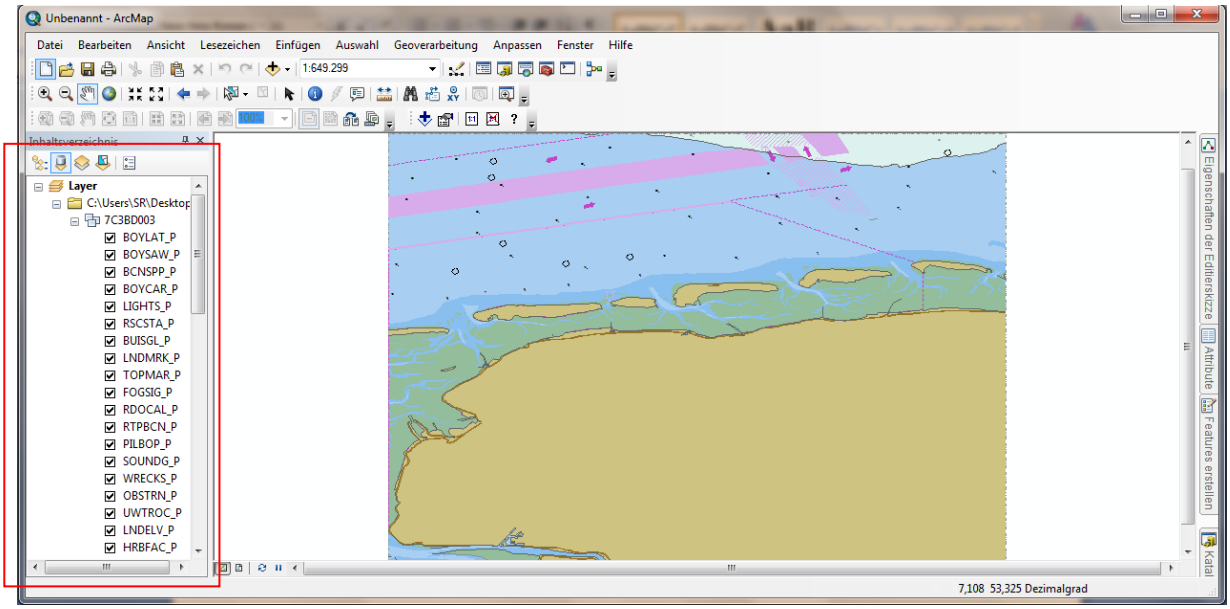

Abbildung 39: ArcMap – S-57-Layer

In der Abbildung ist zu sehen, dass der Kartenausschnitt in den Editor geladen wurde. Auf der linken Seite, im rot umrandeten Bereich, ist zu sehen, dass die einzelnen Layer (Ebenen) der Seekarte geladen wurden. Das Häkchen vor dem jeweiligen Layer bedeutet, dass dieser im Editorfenster angezeigt wird. Um nun einen Layer in ein Shapefile zu konvertieren, wird mit der rechten Maustaste der gewünschte Layer angeklickt und im sich öffnenden Menü der Punkt „Daten“ und im Anschluss „Daten exportieren“ gewählt. Dieser Vorgang wird in Abbildung 40 dargestellt:

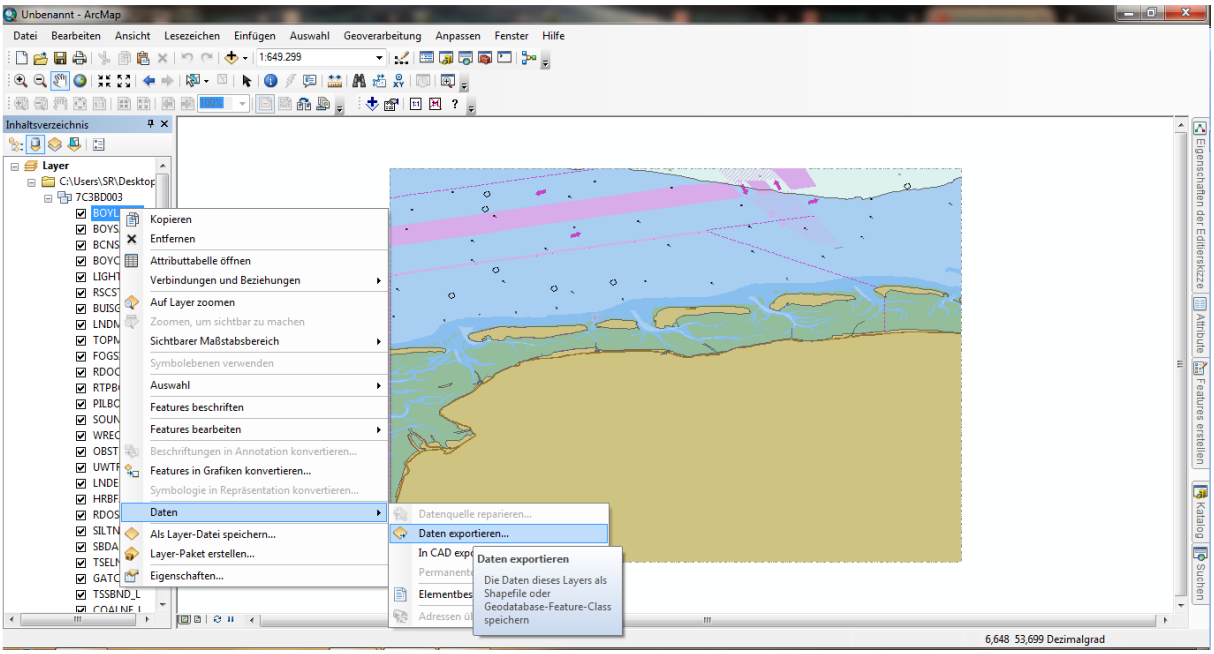

Abbildung 40: ArcMap – Daten exportieren

Nachdem der Menüpunkt „Daten exportieren“ ausgewählt wurde, öffnet sich das Fenster aus Abbildung 41.

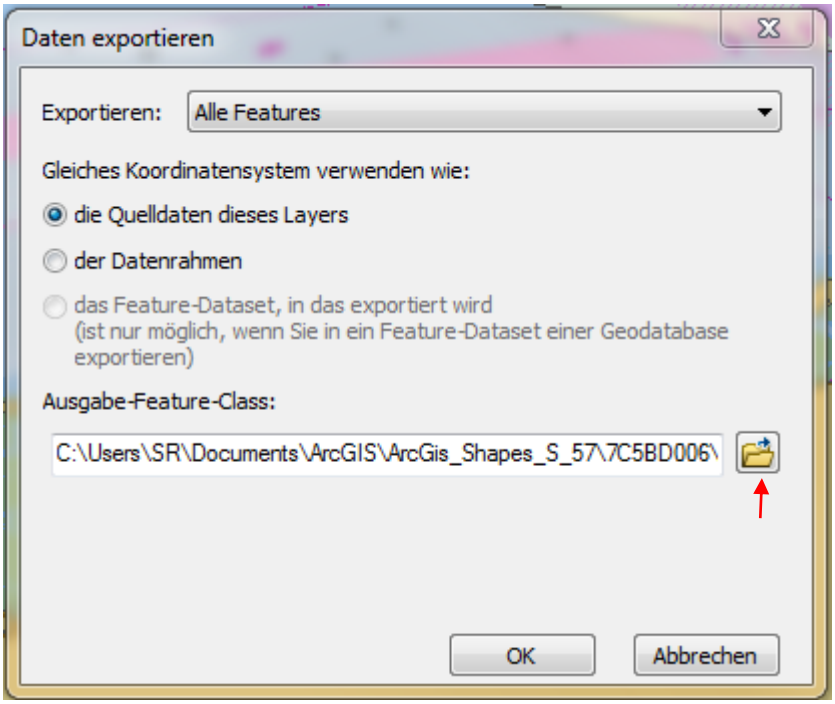

Abbildung 41: ArcMap – Koordinatensystem und Speicherort wählen

In diesem Fenster wird der Pfad für die zu speichernden Shapefiles und das zu verwendende Koordinatensystem festgelegt. Um den Speicherort der zu erstellenden Datei festlegen zu können, wird das Ordnersymbol, welches durch den roten Pfeil gekennzeichnet

ist, angeklickt. Im darauf folgenden Fenster kann nun der Name der zu erstellenden Datei, das Dateiformat und der Speicherort festgelegt werden. Wie in Abbildung 42 zu sehen ist, wird in unserem Fall die Datei im Shapefile-Format gespeichert.

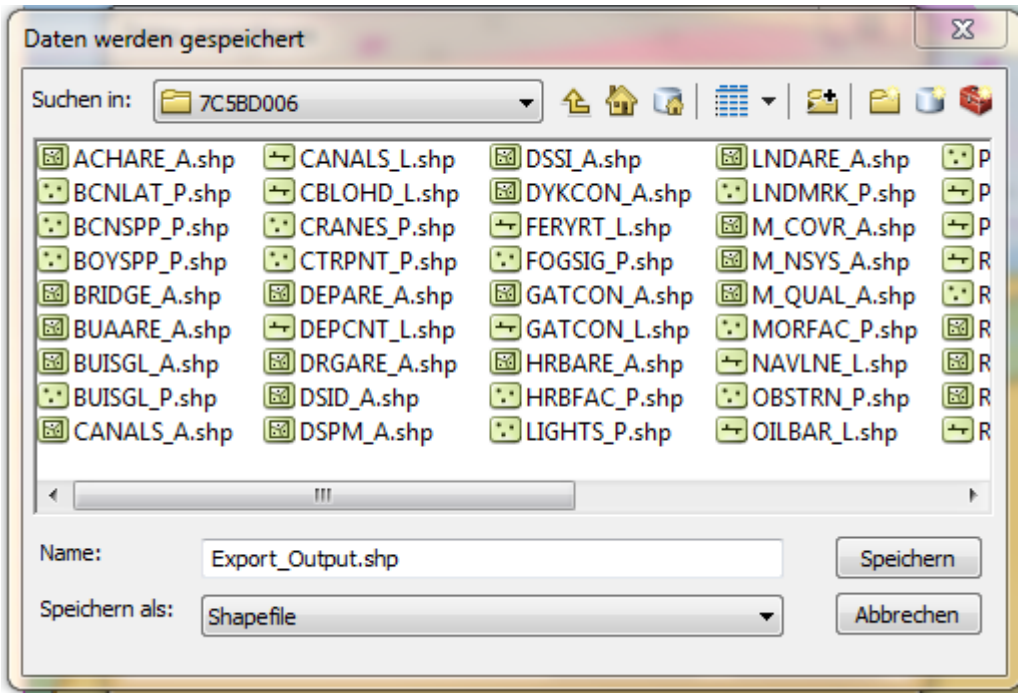

Abbildung 42: ArcMap – Speicherort festlegen

Nach Betätigen des „Speichern“- und des „OK“-Button im Anschluss daran im darunter liegenden Fenster (das aus Abbildung 41) erscheint ein Hinweisfenster mit der Frage, ob die soeben erzeugten Daten im Editor hinzugefügt werden sollen. Diese Nachricht wird mit "Nein" beantwortet. Das bis hierhin beschriebene Vorgehen muss nun für jeden weiteren Layer vorgenommen werden.

Um die Karten in die Simulation einzubinden, ist es notwendig, die Ordner, in dem die Shapefiles liegen, in der Konfigurationsdatei zu definieren. Der XML-Block in Abbildung 43 zeigt den entsprechenden Ausschnitt aus der Konfigurationsdatei (Datei „cfg.xml“ im Java-Projekt „Simulation“). Der Tag „<shapefolder>“ definiert den Ordner, in dem die S-57-Layer als Shapefiles abgelegt worden sind.

```
<map>
	<mapraster file="raster\raster_50m.rmts" enable="true" rasterDistance="50"/>
	<shapes>
		<shapefolder value="s57\7C4BD004"/>
		<shapefolder value="s57\7C4BD006"/>
		<shapefolder value="s57\7C4BD011"/>
		<shapefolder value="s57\7C4BD012"/>
		<shapefolder value="s57\7C4BD013"/>
		<shapefolder value="s57\7C4BD014"/>
		<shapefolder value="s57\7C4BD015"/>
		<shapefolder value="s57\7C4BD016"/>
		<shapefolder value="s57\7C5BD002"/>
		<shapefolder value="s57\7C5BD005"/>
		<shapefolder value="s57\7C5BD006"/>
	</shapes>
</map>
```

Abbildung 43: Konfigurationsdatei zur Definition der Ordner mit den Shapefiles

Insbesondere für die Wegfindung der Agenten ist eine Rasterung des Kartenmaterials unumgänglich. Der XML-Tag „<mapraster>“ definiert die Rasterdatei (Attribut „file“), die Rastergröße (Attribut „rasterDistance“) sowie, ob eine Rasterung notwendig ist (Attribut „enable“). Sofern die Rasterdatei existiert, wird diese geladen. Wichtig hierbei ist, dass die Rasterdatei zum Kartenmaterial passt, denn es findet kein Validierungsverfahren statt. Wenn die Datei geladen werden kann, wird eine Rasterung angestoßen. Die Rastergröße in Metern beschreibt, wie groß ein Raster sein soll. Dabei muss beachtet werden, dass bei kleinen Rastergrößen das Rasterungsverfahren sehr lange dauern kann. Weiterhin muss bei einer Änderung der Rastergröße entweder die bereits existierende Datei gelöscht bzw. umbenannt oder eine neue Datei erstellt werden. Die Erweiterung von Layern im Kartenmaterial wird in Kapitel 3.3.1.1.1 beschrieben.

2.4.2 Steuerung eines Schiffes

Die MTS ermöglicht die direkte Steuerung von Schiffen durch den Benutzer. Dazu gibt es den Federate „InteractiveVessel“, der sich im Java-Projekt „Simulation“ befindet. Nach dem Start erscheint die Benutzeroberfläche aus Abbildung 44. Innerhalb dieser kann ein Schiff, welches zuvor in der Auswahlbox definiert wurde, mit den Pfeiltasten der Tastatur oder mit den Buttons der Benutzeroberfläche gesteuert werden. Das Schiff muss einen Agenten des Typs „SteeringAgent“ besitzen. Mit der Pfeiltaste nach oben kann das Schiff beschleunigt, mit der Pfeiltaste nach unten abgebremst bzw. rückwärts gefahren werden. Die linke und rechte Pfeiltaste lenken das Schiff in die entsprechende Richtung.

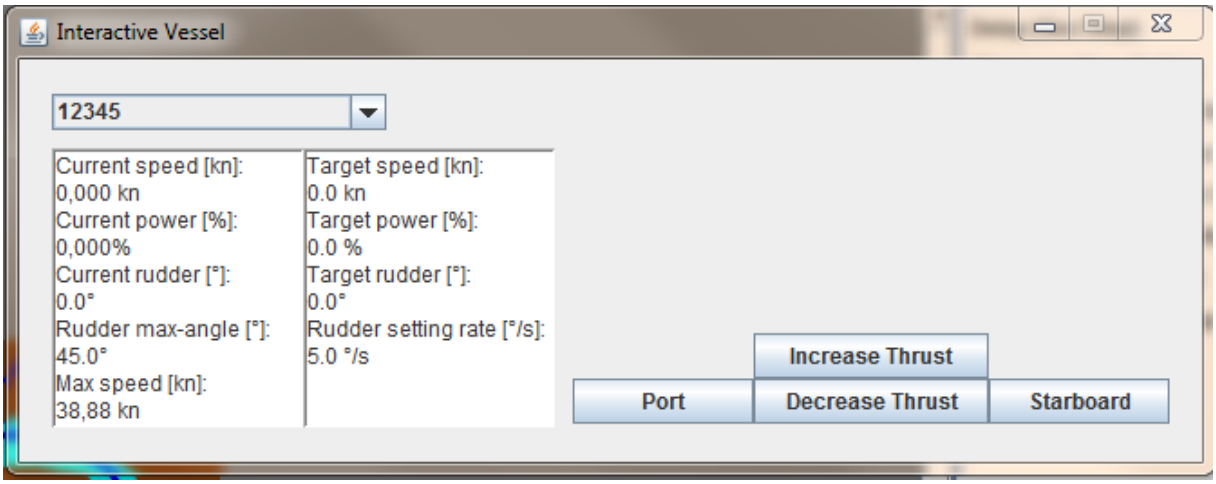

Abbildung 44: Steuerung des Interactive Vessel

Weiterhin kann eine USB-Schiffskonsole angebunden werden. Diese muss vor dem Start des Federates „InteractiveVessel“ bereits angeschlossen sein, damit das Programm die Konsole initialisieren kann.

2.4.3 Realzeitsimulator

Der Realzeitsimulator (Realtime Synchronizer) befindet sich ebenfalls im Java-Projekt „Simulation“ (Package „de.wi_ol.be.mts.realtimeWaiter“). Durch ihn kann die Simulation in Realzeit ablaufen. Dies wird durch einen weiteren Federate ermöglicht. Dieser liest zu Beginn die Datei „realtimewaiter.ini“ (befindet sich im Java-Projekt „Simulation“) ein, um dort das Mapping von einem HLA-Step zu einer Zeiteinheit in Millisekunden auszulesen. Der Federate steppt erst nachdem die ausgelesene Zeit abgelaufen ist. Dabei zieht er die Zeit ab, die die anderen Federates in Anspruch genommen haben. Das bedeutet, dass der Realzeitsimulator die Simulation lediglich verlangsamen, jedoch nicht beschleunigen kann, wenn z.B. ein anderer Federate länger beim Berechnen braucht. Weiterhin wird eine Benutzeroberfläche mit einer Checkbox gestartet, mit der der Realzeitsimulator deaktiviert bzw. wieder aktiviert werden kann.

2.4.4 MarineTraffic-Schnittstelle

Die MarineTraffic-Schnittstelle (Marine Traffic Reader) – im Java-Projekt „Simulation“ – ist ein weiterer Federate zum Steuern von Schiffen. Hierbei wird eine CSV-Datei eingelesen, die über auf der Webseite von MarineTraffic [Ma14] heruntergeladen werden kann (kostenfreie Registrierung erforderlich). Die exportierte CSV-Datei enthält die folgenden Spalten kommasepariert in der genau der dargestellten Reihenfolge:

"MMSI","Zeit (UTC)","Art der Position","Gebiet","Hafen","Geschwindigkeit","Kurs","Latitude","Longitude"

Die Datei („csv\marinetraffic.csv“) wird eingelesen. Anschließend werden die Routen über HLA an die Routing Agents kommuniziert. Während der Simulation wird anhand der Uhrzeit die zu fahrende Geschwindigkeit kommuniziert und entsprechend verändert. Die erste

Uhrzeit, die in der CSV-Datei angegeben wird, entspricht der Simulationszeit null. Bei der nächsten Angabe wird die Uhrzeit und Geschwindigkeit überprüft. Wenn sich die Geschwindigkeit ändert, wird diese zur entsprechenden Simulationszeit (Uhrzeit der nächsten Zeile – erste angegebene Uhrzeit) gesendet.

Zum Einsatz kommt die MarineTraffic-Schnittstelle beim Testen der MTS auf Realitätsnähe. Nähere Informationen dazu in Kapitel 4.5.

2.4.5 Schifffführungssimulator-Schnittstelle

Die Schifffführungssimulator-Schnittstelle sendet an die MTS NMEA-Daten, um die Schiffe entsprechend zu steuern. Die NMEA-Daten werden zur Kommunikation von Schiffs- und Positionsinformationen genutzt. Diese Schnittstelle ist als zusätzlicher Federate implementiert. Der Federate kann NMEA-Daten über UDP empfangen oder Dateien auslesen und weiter interpretieren. Zur Interpretation der NMEA Sentences wird die Java Marine API [Tu14] genutzt. Dabei können mehrere Agenten gesteuert werden. Diese werden in der Datei „nmeaFederate.ini“ konfiguriert. Tabelle 1 zeigt die Attribute, die in der „ini“-Datei genutzt werden können.

Attribut	*Beschreibung*	*Beispiel*
agentsIds	Definiert alle Agenten, die gesteuert werden sollen.	agentsIds=1;2;3 Definiert drei Agenten mit den o.g. IDs.
ip	Definiert die IP-Adresse, von der die UDP Nachrichten mit den NMEA Sentences kommen.	ip=127.0.0.1
file	Definiert eine Datei, in der sich NMEA Sentences befinden und ordnet sie einem Agenten zu.	file=1;D:\nmea.txt Die Sentences aus der Datei „nmea.txt“ werden mit der Agenten-ID 1 assoziiert.
GLL	Definiert den Port für die GLL Nachrichten für einen Agenten. GLL steht für Geographic position, latitude / longitude.	GLL=1;6222 Die GLL Sentences kommen über den Port 6222 für den Agenten mit der ID 1.
VTG	Definiert den Port für die VTG Nachrichten für einen Agenten. VTG steht für Track made good and ground speed	VTG=1;6222 Die VTG Sentences kommen über Port 6222 für den Agenten mit der ID 1.
ZDA	Definiert den Port für die ZDA Nachrichten für einen Agenten. Die ZDA Nachrichten beinhalten Zeit- und Datumangaben.	ZDA=1;6222 Die ZDA Sentences kommen über Port 6222 für den Agenten mit der ID 1.
HDG	Definiert den Port für die HDG Nachrichten für	HDG=2;5400

	einen Agenten. HDG steht für Magnetic heading, deviation, variation.	Die HDG Sentences kommen über Port 5400 für den Agenten mit der ID 2.
HDT	Definiert den Port für die HDT Nachrichten für einen Agenten. Die HDT Nachrichten geben das Heading an.	HDT=3;5400 Die HDT Sentences kommen über Port 5400 für den Agenten mit der ID 3.

Tabelle 1: Beschreibung der Attribute für den NMEA Federate

Wichtig zu erwähnen ist, dass ein Agent entweder über UDP oder über die Datei gesteuert werden kann. Wenn beides angegeben wird, wird die Datei bevorzugt verwendet.

2.4.6 Schiffserzeugung an Häfen

Durch den Szenario-Importer (s. Kapitel 2.1.2) können zu Beginn der Simulation die Schiffe definiert werden. Mit Simulationsbeginn können keine weiteren Schiffe über den Szenario-Importer erstellt werden. Es gibt aber die Option, Schiffe an bestimmten Häfen erzeugen zu lassen. Dies wird in der Konfigurationsdatei „cfg.xml“ beschrieben (s. Abbildung 45).

```
<harbour name="Brunsbuettel" shipPointLong="9.129595" shipPointLat="53.884751"
shipStartHeading="180">
	<harbourcoordinate lat="53.8867784" long="9.12284448"/>
	<harbourcoordinate lat="53.8825020" long="9.12153912"/>
	<harbourcoordinate lat="53.8828142" long="9.13670612"/>
	<harbourcoordinate lat="53.8869657" long="9.13744273"/>
	<autoGeneration value="true" delayInSeconds="10">
		<autoGenerationTemplates>
			<autoGenerationTemplate
src="autoGenerationTemplates\150meter.emod" frequency="1"/>
			<autoGenerationTemplate
src="autoGenerationTemplates\300meter.emod" frequency="3"/>
		</autoGenerationTemplates>
	</autoGeneration>
</harbour>
```

Abbildung 45: Konfigurationsdatei zur Schiffserzeugung

Der Tag „<harbour>“ definiert einen Hafenbereich. Mit dem Attribut „name“ wird dem Hafen ein Name zugeteilt, der u.a. den Schiffen als Ziel mitgegeben wird. Die Attribute „shipPointLong“ und „shipPointLat“ geben die Position an (Longitude und Latitude), von der die Schiffe starten. Das Attribut „shipStartHeading“ gibt das Heading an, welches das Schiff zum Start mitbekommt. Mit den Tags „<harbourcoordinate>“ können GPS-Koordinaten angegeben werden. Die Koordinaten bilden zusammen ein Polygon, das das Hafengebiet kennzeichnet. Die eigentliche automatische Generierung der Schiffe erfolgt mit dem Tag „<autoGeneration>“. Zum Ein- bzw. Ausschalten der Autogenerierung muss das Attribut „value“ auf true oder false gesetzt werden. Das Attribut „delayInSeconds“ bestimmt, wann ein neues Schiff generiert werden soll. Die Zeitangabe ist in Sekunden anzugeben und gilt ab dem Zeitpunkt, zu dem das zuvor erzeugte Schiff das Hafengebiet verlassen hat. Dementsprechend kann mit diesem Attribut das Verkehrsaufkommen von wenig bis sehr dicht gesteuert werden. Durch das Delay zwischen dem Abfahren der Schiffe soll eine

realistische Verkehrssituation simuliert werden, in der die Schiffe in regelmäßigen zeitlichen Abständen hintereinander herfahren.

Welche Schiffstypen generiert werden, kann durch sogenannte Templates definiert werden. Hierzu dient der Tag „<autoGenerationTemplate>“ mit den Attributen „src“, welches den Pfad zur EMod-Datei angibt, und „frequency“, welches die Häufigkeit in Relation zu den anderen Templates angibt. Wichtig hierbei ist, dass in den EMod-Dateien nur ein Schiff definiert und als Agententyp ein „GeneratedAgent“ angegeben wird. Weiterhin ist zu beachten, dass der erste angefahrene Hafen nicht der Hafen ist, der die Autogenerierung innehat. Es können mehrere Häfen die Autogenerierung aktiviert haben.

2.4.7 Nahbereich eines Schiffes

In der Seeschifffahrt sind Nahbereichssituationen zwischen Schiffen zu vermeiden. Eine Nahbereichssituation ist nach Regel 8 der Internationalen Kollisionsverhütungsregeln (KVR) "eine Situation, in der eine erhöhte Gefahr des Zusammenstoßes besteht" [BuJV14]. Ähnlich wie der Begriff "sichere Geschwindigkeit", lässt sich auch eine Nahbereichssituation nicht in festgelegten Zahlen bzw. Abständen ausdrücken. Eine Einschätzung der gefährdenden Situation liegt zuletzt also immer beim verantwortlichen Kapitän des Schiffes. Wie sich das Schiff verhalten muss, sollte es einmal in eine Nahbereichssituation kommen, ist festgelegt in den KVR 8 und 19.

Da in der MTS jedoch die Vermeidung von Nahbereichssituationen zwischen den Agenten der Schiffe implementiert werden soll, muss eine Lösung, die in Formeln oder Zahlen ausgedrückt werden kann, gefunden werden, um diese Situationen zu erkennen und Gegenmaßnahmen einleiten zu können.

Als Regelwert für den Radius eines Nahbereichs wird oftmals die Tragweite der Schallsignale von zwei Seemeilen herangezogen [Bid04]. Diese Tragweite ist in Anlage III – Technische Einzelheiten der Schallsignalanlagen – der KVR für Schiffe von 200 Meter Länge und mehr festgelegt.

Genauer berechnen lässt sich der Nahbereich (N_0 = Nahbereich in voraus 0°) wie folgt:

AC ⇒ Abstand Radarantenne zum Mittelpunkt C (CD, Nahbereich)

RC ⇒ Radius von CD

RN ⇒ Radius des Nahbereichs

ΔL ⇒ Abstand Vorsteven (Bugspitze) - Radarantenne

$S(V_A)$ ⇒ geschwindigkeitsabhängige Stoppstrecke in kbl

V_A ⇒ Schiffsgeschwindigkeit in kn (= Knoten =1 Seemeile/h ≈ 0,514444 m/s)

kbl ⇒ Kabellänge = 1/10 Seemeilen

$RN = AC + V_A / 3 + 1kbl$

$AC = 1{,}15 * S(V_A) + 0{,}5 * \Delta L$

$N_0 = RN + AC$

Wird eine Seitenpeilung [Li11] vorgenommen, ist der Nahbereich für ein seitengepeiltes Objekt wie folgt zu bestimmen:

SP ⇒ Seitenpeilung

$N = N_0 * \cos SP$

Abbildung 46 veranschaulicht den Nahbereich eines Schiffes.

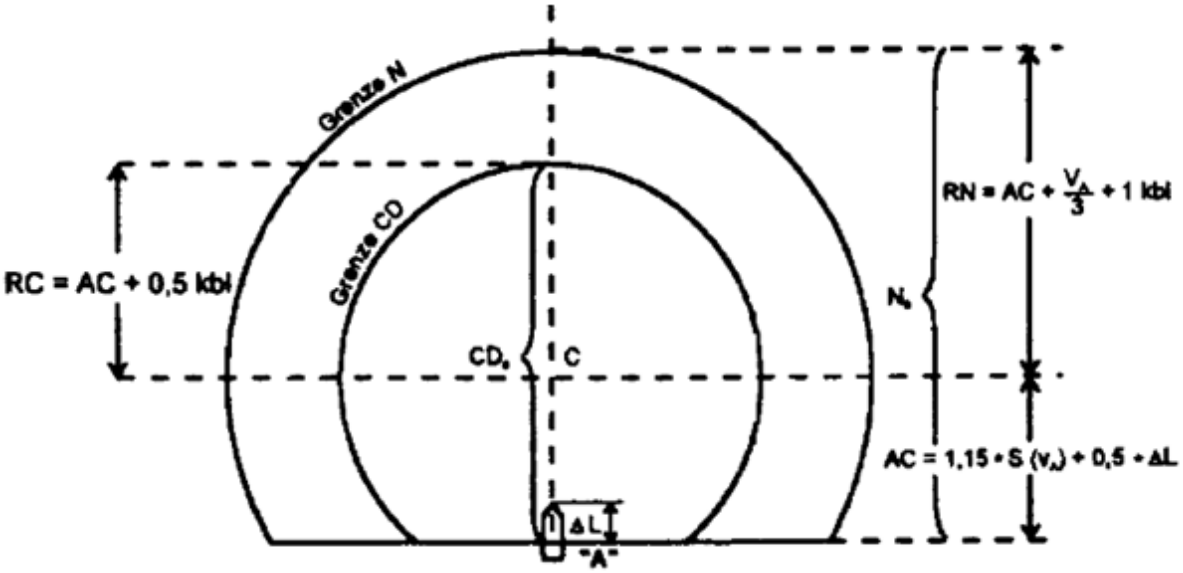

Abbildung 46: Nahbereich eines Schiffes [SmR13, S. 3]

Der Nahbereich für einen Supertanker mit einer Geschwindigkeit von 15 Knoten/h und einem Stopp-Weg von maximal dem 15-fachen der eigenen Schiffslänge beträgt somit ca. 4,5 Seemeilen in Voraus 0°.

2.4.8 Statistiken abrufen

Die Statistiken können über den Menüpunkt „Tools“ in der Simulation aufgerufen werden. Nach Simulationsbeginn werden die erzeugten Schiffe im Statistikfenster angezeigt. Über Auswahl der dazugehörigen Checkbox können die am Simulationslauf beteiligten Schiffe zum Diagramm hinzugefügt bzw. entfernt werden. Im Bereich „Properties“ kann entweder die Geschwindigkeit über Grund (SOG) oder die Fahrt durchs Wasser (LOG) als im Diagramm anzuzeigende Eigenschaft ausgewählt werden. Das Diagramm wird kontinuierlich aktualisiert und zeigt die jeweilige Eigenschaft in Abhängigkeit zu der Zeit, wie in Abbildung 47 zu sehen ist.

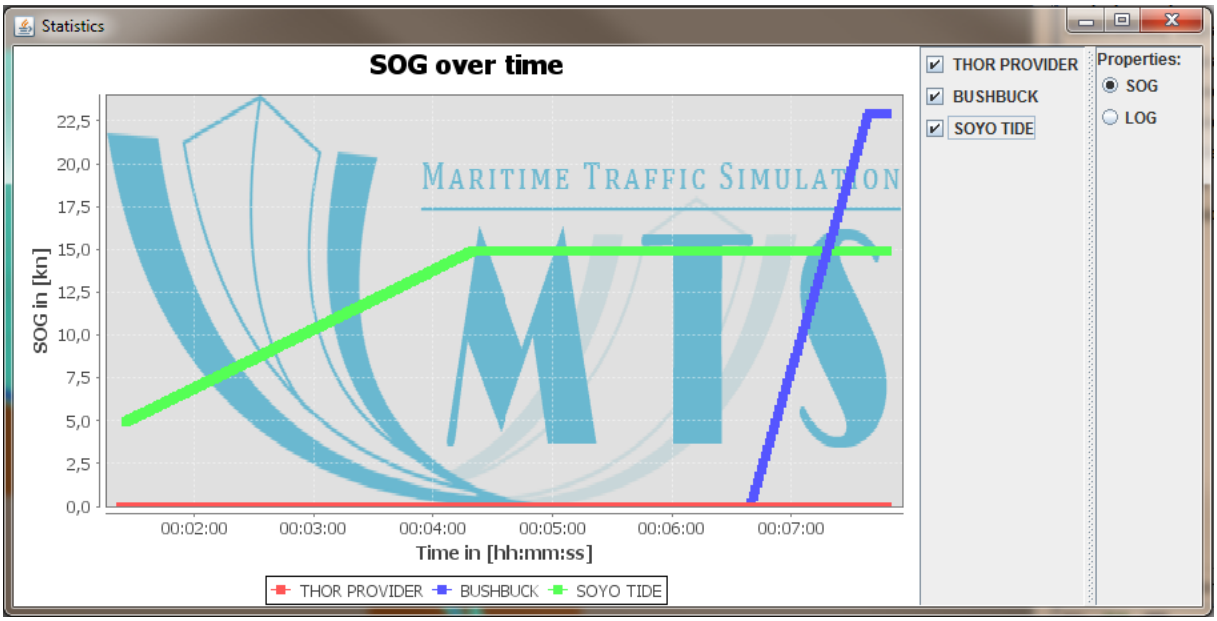

Abbildung 47: Fenster „Statistics“

Zum aktuellen Implementierungsstand des Statistikbereichs können die Geschwindigkeiten der Schiffe miteinander verglichen werden. Haupt-Anwendungsbereich ist dabei die bessere Visualisierung des Geschwindigkeitsverhaltens der Schiffe während eines Überholmanövers. Im Diagramm lässt sich durch den sichtbaren Anstieg des SOG- oder LOG-Balkens beobachten, ab wann ein Schiff mit dem Überholmanöver beginnt. Gleichsam lässt sich auch bei einem Abstieg der Baken beobachten, wann es das Überholmanöver abgeschlossen hat und wieder auf die ursprüngliche Geschwindigkeit zurückkehrt. Wie sich der Statistikbereich erweitern lässt, kann in Abschnitt 3.3.1.2.2 nachgelesen werden.

2.4.9 Umweltsimulator

2.4.9.1 Benutzeroberfläche Hauptfenster

Das Hauptfenster des Umweltsimulators enthält die vom Umweltsimulator angebotenen Szenarien und Tools.

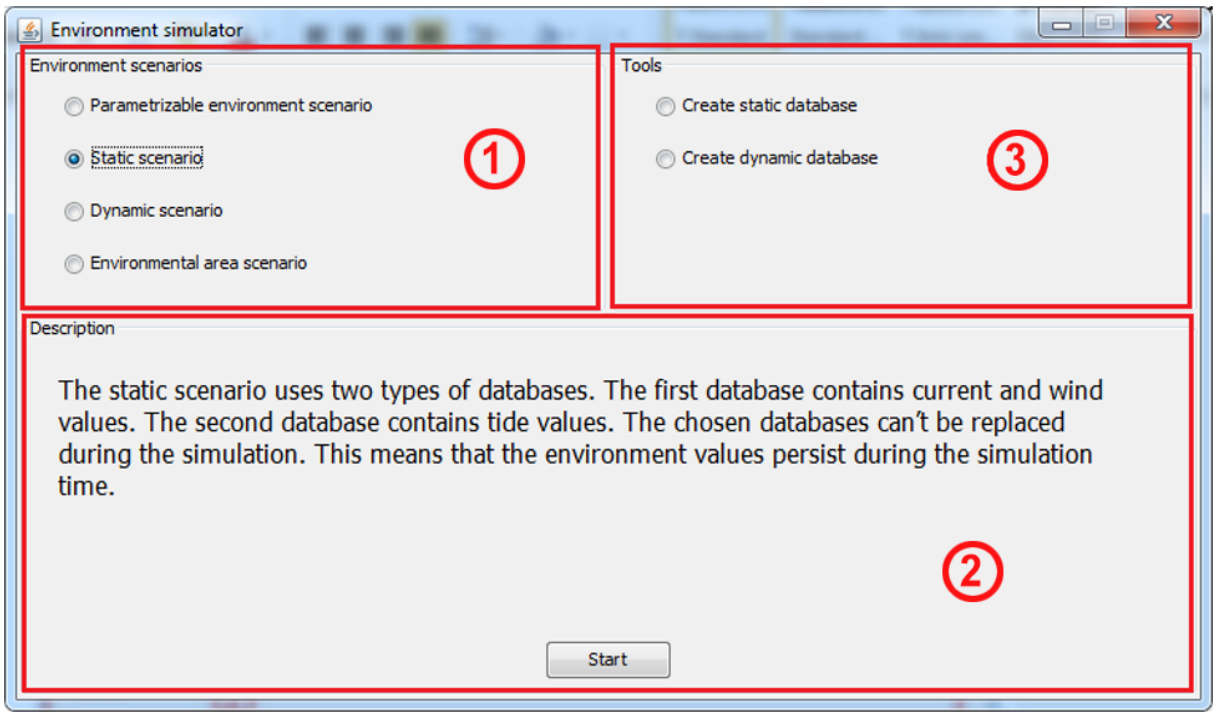

Abbildung 48: Hauptfenster des Umweltsimulators

Das Hauptfenster besteht aus drei Bereichen. Unter der Überschrift „Environment scenarios“ (Abbildung 48, Punkt 1) kann über Radio-Buttons ein Szenarien-Typ ausgewählt werden. Für jedes ausgewählte Szenario steht unter der Überschrift „Description“ (Abbildung 48, Punkt 2) eine kurze Beschreibung. Über den „Start“-Button kann die ausgewählte Szenarienart dann gestartet werden.

Unter der Überschrift „Tools“ (Abbildung 48, Punkt 3) kann ebenfalls per Radio-Button ausgewählt werden, mit welchem Tool weitergearbeitet werden soll. Ist ein Tool ausgewählt, kann der Description erneut eine entsprechende Beschreibung entnommen werden. Bei Betätigen des „Start“-Button wird das ausgewählte Tool gestartet. Wichtig dabei ist, dass nur ein Tool bzw. ein Szenario gleichzeitig ausgewählt und gestartet werden kann.

2.4.9.2 Erstellen einer Strömungsdatenbank für das statische Szenario

Zum Erstellen einer Strömungsdatenbank auf Basis von BSH-GRIB2-Dateien (s. Kapitel 2.1.4.1) wird die Software wgrib2 genutzt (s. hierzu auch Kapitel 3.3.2). Mit Hilfe dieser Software wird ein CSV-Dump der GRIB2-Daten vorgenommen. Das Programm wgrib2 wird per Kommandozeile verwendet. Ein CSV-Dump für das statische Szenario aus einer GRIB2-Datei des BSH wird mit Hilfe des folgenden Kommandos vorgenommen:

Hinweis:

Die wgrib2.exe liegt im Verzeichnis "`c:\wgrib2`"

Name des BSH-GRIB2-Files: `BSH_GRIBFILE.grb2`

Name der Zieldatei: `ZIELDATEI.csv`

```
c:\wgrib2>wgrib2.exe BSH_GRIBFILE.grb2 -for "1:2" -csv ZIELDATEI.csv
```

Ist der CSV-Dump erstellt, kann das Java-Tool für das Erstellen der Strömungsdatenbank für das statische Szenario genutzt werden. Dieses kann über die Startoberfläche des Umweltsimulators gestartet werden (vgl. Abbildung 49).

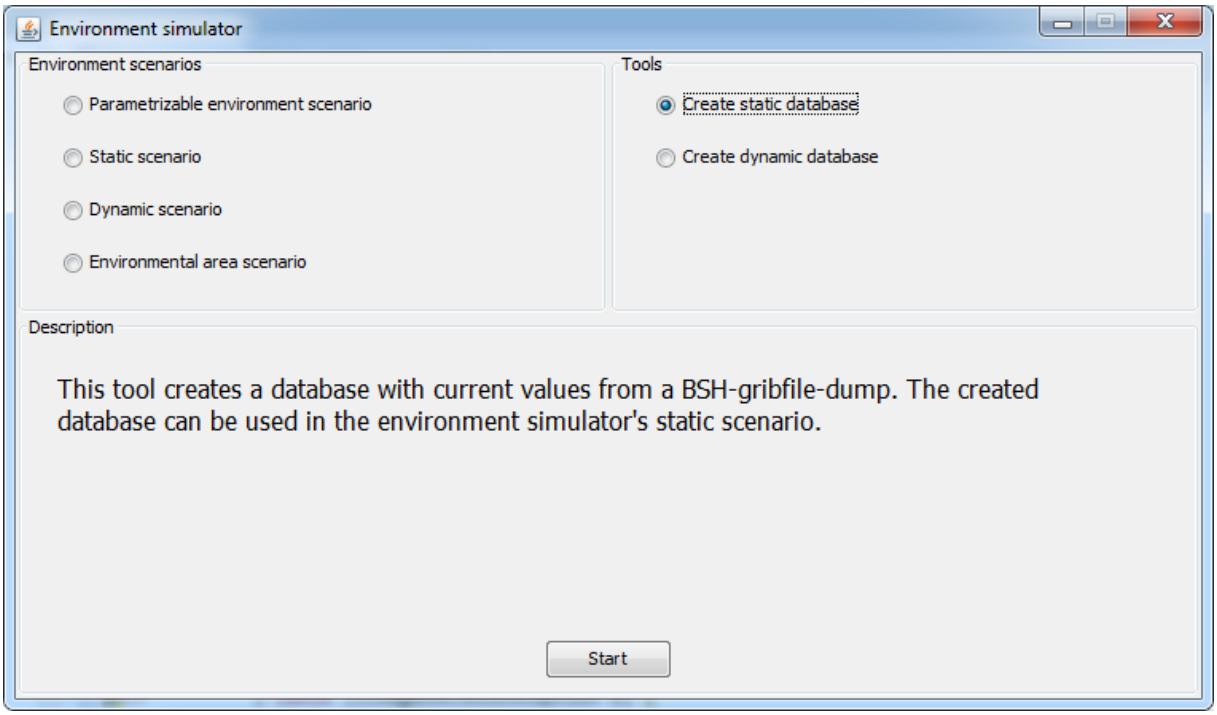

Abbildung 49: Statisches Umweltszenario – Start des "Create static database"-Tools

In der Benutzeroberfläche des "Create static database"-Tools (s. Abbildung 50) kann über den "Select file"-Button der zuvor erstellte CSV-Dump ausgewählt werden. Es erscheint ein Dateiauswahl-Fenster.

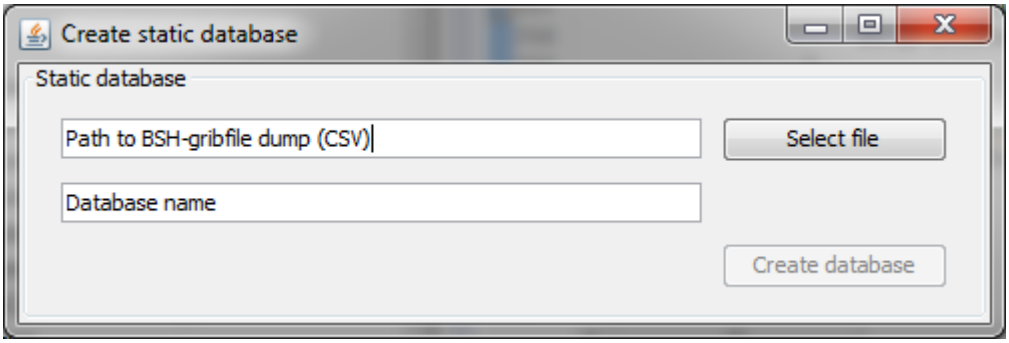

Abbildung 50: Statisches Umweltszenario – Fenster "Create static database"

Nach Auswahl der Datei muss ein Datenbankname eingegeben werden. Eine Dateiendung („.db“) wird automatisch angefügt.

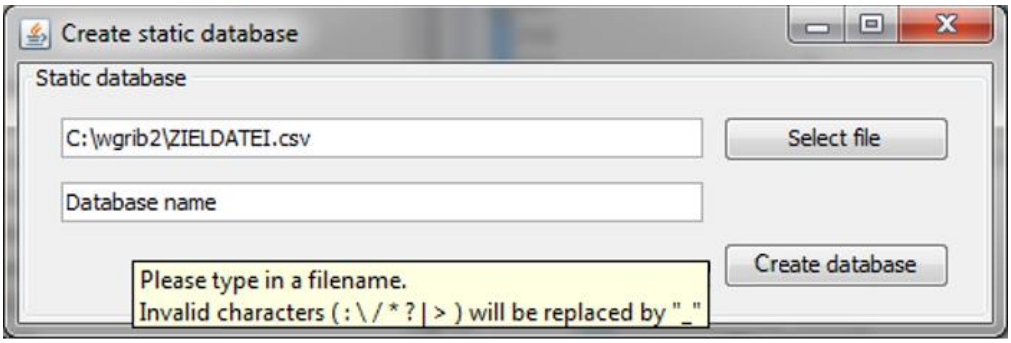

Abbildung 51: Statisches Umweltszenario – Eingabe eines Namens für die Datenbank

Nach Klicken des "Create database"-Buttons wird die Szenarien-Datenbank erstellt und die Strömungsdaten des CSV-Dumps werden in dessen Tabelle "current_final" übernommen. Das Tool legt zudem eine Tabelle "windwave_final" für Wind- und Wellendaten an.

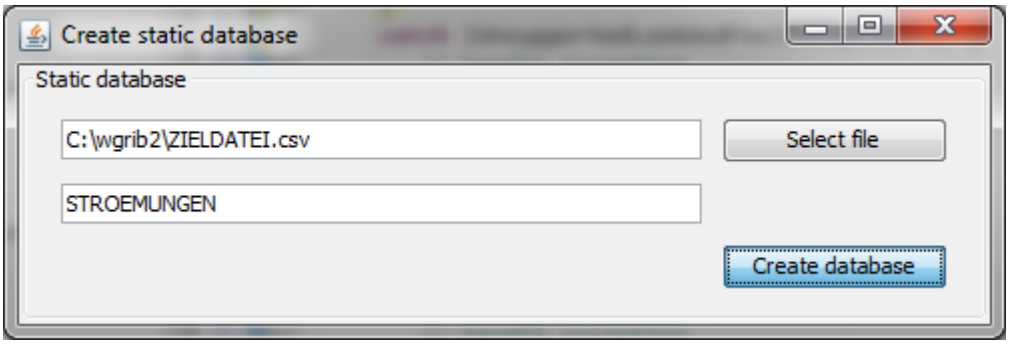

Abbildung 52: Statisches Umweltszenario – Erstellen der Datenbank

Die erzeugte Datenbank wird im selben Ordner gespeichert, in dem sich auch der CSV-Dump befindet (s. Abbildung 53). Sollte sich in diesem Ordner eine temporäre Datenbank namens "temp.db" befinden, kann diese (wegen mangelnder Zugriffsrechte) nicht automatisch gelöscht werden. Sie sollte daher nach erfolgreicher Generierung der Datenbank manuell gelöscht werden.

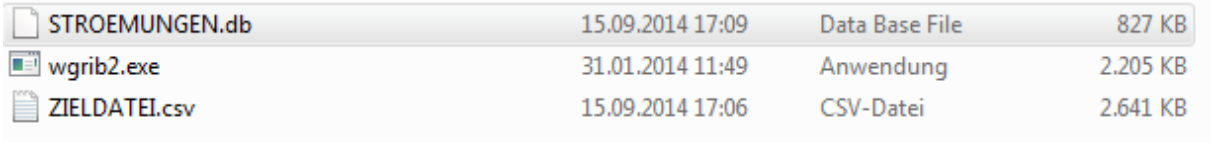

Abbildung 53: Statisches Umweltszenario – Erzeugte Szenarien-Datenbank mit Strömungsdaten

2.4.9.3 Erstellen einer Strömungsdatenbank für das dynamische Szenario

Auch im dynamischen Szenario wird eine Strömungsdatenbank auf Basis von BSH-GRIB2-Dateien genutzt. Ähnlich wie im Kapitel zuvor wird wgrib2 per Kommandozeile verwendet, um einen CSV-Dump aus einer BSH-GRIB2-Datei zu erzeugen. In diesem Szenario müssen allerdings zwei Dump-Dateien erzeugt werden. In der ersten Datei werden nur die ungeraden GRIB-Records exportiert, in der zweiten Datei werden die geraden GRIB-Records exportiert. Der Grund für diese Teilung ist, dass in den BSH-Dateien jeweils abwechselnd Records mit der U- und der V-Komponente (horizontale und vertikale Richtung) einer Strömung für einen

Messpunkt vorliegen. Aus diesen beiden Messwerten kann die Strömungsrichtung und -geschwindigkeit errechnet werden [Vi03].

Der erste Schritt zur Erstellung der dynamischen Strömungsdatenbank ist das Exportieren der GRIB-Records mit den U- und V-Komponenten der Strömung. Da wgrib2 reguläre Ausdrücke für die Auswahl der Records unterstützt, können diese wie folgt exportiert werden:

Hinweis:

Die wgrib2.exe liegt im Verzeichnis "`c:\wgrib2`"

Name des BSH-GRIB2-Files: `BSH_GRIBFILE.grb2`

Export der Records mit den U-Komponenten:

Name der Zieldatei der U-Komponenten: `ZIELDATEI-U.csv`

```
c:\wgrib2>wgrib2.exe BSH_GRIBFILE.grb2 -match "(^[0-9]*[13579]:)" -csv ZIELDATEI-U.csv
```

Export der Records mit den V-Komponenten:

Name der Zieldatei der V-Komponenten: `ZIELDATEI-V.csv`

```
c:\wgrib2>wgrib2.exe BSH_GRIBFILE.grb2 -match "(^[0-9]*[02468]:)" -csv ZIELDATEI-V.csv
```

Sind die CSV-Dumps erstellt, kann das Java-Tool für das Erstellen der Strömungsdatenbank für das dynamische Szenario genutzt werden. Dieses kann über die Startoberfläche des Umweltsimulators gestartet werden (vgl. Abbildung 54).

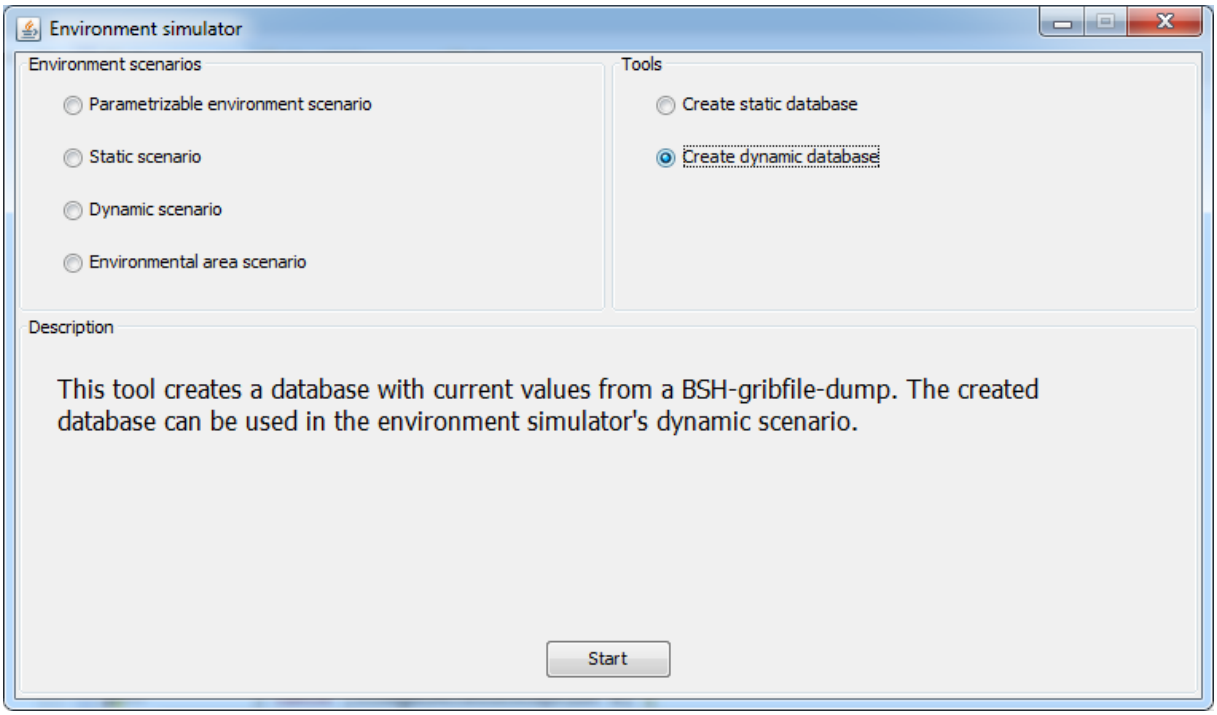

Abbildung 54: Dynamisches Umweltszenario – Start des "Create dynamic database"-Tools

In der Benutzeroberfläche des "Create dynamic database"-Tools (s. Abbildung 55) können mit Hilfe der beiden "Select file"-Buttons die zuvor erstellten CSV-Dumps ausgewählt. Es erscheint jeweils ein Dateiauswahl-Fenster.

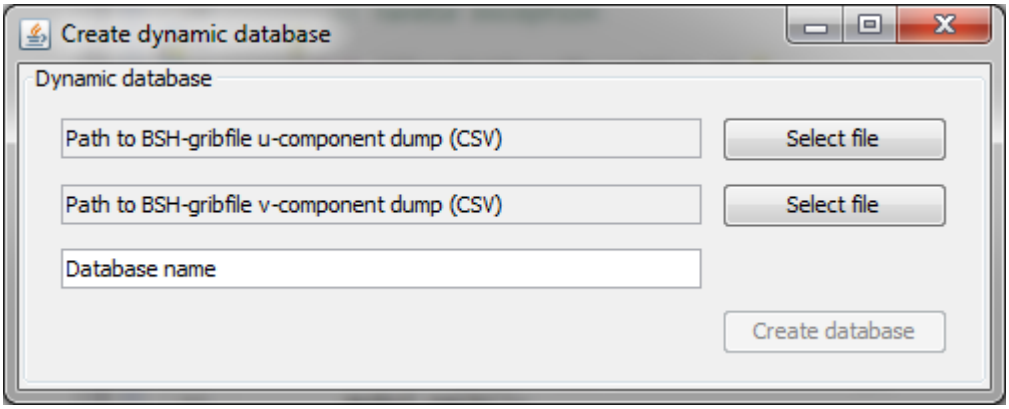

Abbildung 55: Dynamisches Umweltszenario – Fenster "Create dynamic database"

Nach Auswahl der Dateien muss ein Datenbankname eingegeben werden. Eine Dateiendung („.db“) wird automatisch angefügt.

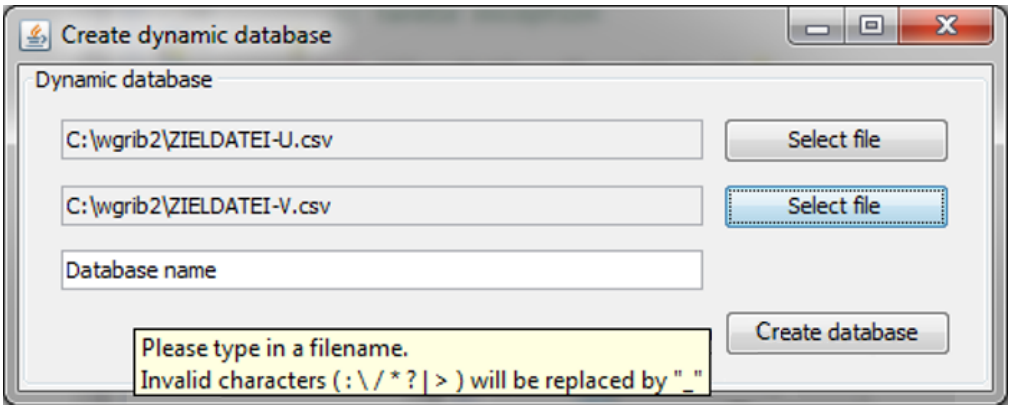

Abbildung 56: Dynamisches Umweltszenario – Eingabe eines Namens für die Datenbank

Nach einem Klick auf "Create database" wird die Szenarien-Datenbank erstellt und die Strömungsdaten der beiden CSV-Dumps in dessen Tabelle "current_final" übernommen. Der Fortschritt der Datenbankerzeugung kann auf der Kommandozeile von Eclipse verfolgt werden. Das Tool legt zudem eine Tabelle "windwave_final" für Wind- und Wellendaten an.

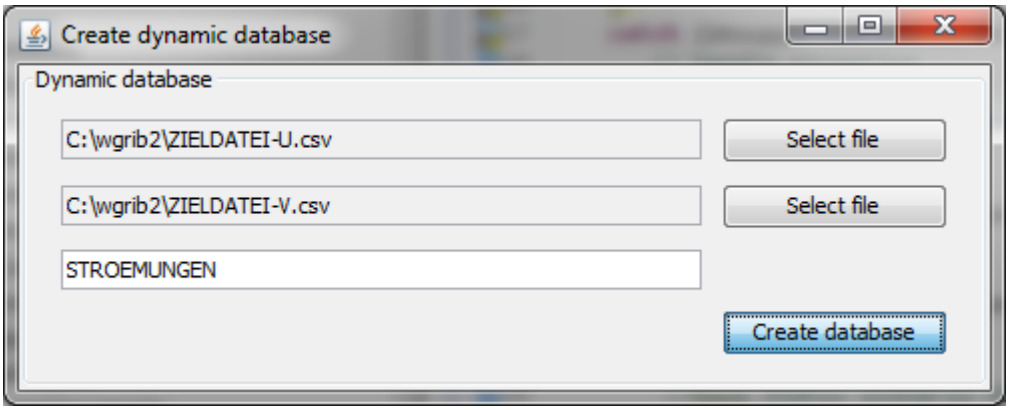

Abbildung 57: Dynamisches Umweltszenario – Erstellen der Datenbank

Die erzeugte Datenbank wird im selben Ordner gespeichert, in dem sich auch die CSV-Dumps befinden (s. Abbildung 58).

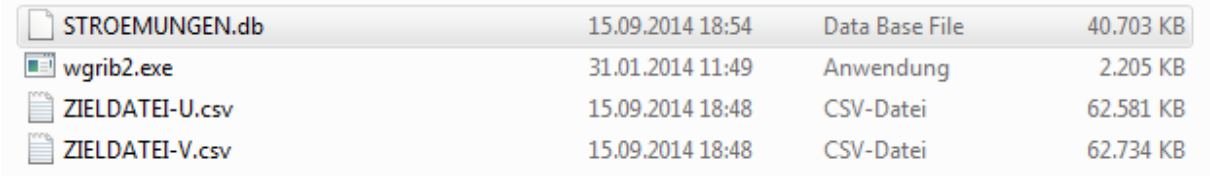

Abbildung 58: Dynamisches Umweltszenario – Erzeugte Szenarien-Datenbank mit Strömungsdaten

2.4.9.4 Starten des parametrisierbaren Umweltszenarios

In diesem Kapitel wird das Erstellen eines parametrisierbaren Umweltszenarios beschrieben. Zunächst wird der Umweltsimulator gestartet, der Radio-Button „Parametrizable environment scenario“ ausgewählt und „Start“ gedrückt (vgl. Abbildung 48). Daraufhin öffnet sich die Benutzeroberfläche des parametrisierbaren Szenarios (vgl. Abbildung 59). Innerhalb der GUI

muss der Benutzer fünf Werte für die erforderlichen Umweltdaten in Textfeldern eingeben. Jeder Wert muss eine Double- oder Integer-Zahl sein. Jedes Textfeld ist mit einem Label versehen, welches die erforderliche Einheit des Eingabewertes angibt. Die genaue Beschreibung der Umweltdaten wird per Tooltip über das jeweilige Label angezeigt. Wird ein ungültiges Zeichen in eines der Textfelder eingetragen, wird dieses rot hinterlegt und kann nicht verlassen werden. Erst wenn das oder die ungültige(n) Zeichen entfernt wurden, wird das Textfeld grün hinterlegt und es kann verlassen werden.

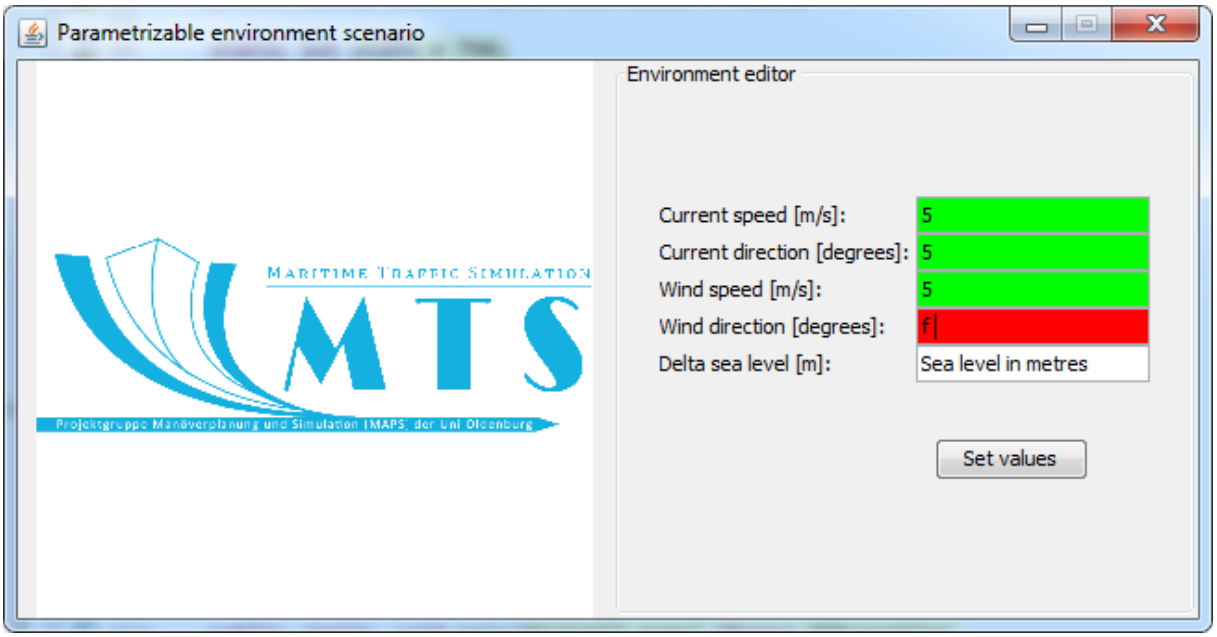

Abbildung 59: Parametrisierbares Umweltszenario – Hauptfenster

Wurden in alle fünf Textfelder valide Werte eingetragen, kann das Szenario über den „Set values“-Button an die MTS abgeschickt werden. Dabei muss die CERTI-Laufzeitinfrastruktur (RTIG.exe, RTIG = Runtime Infrastructure Gateway) laufen, sonst kommt es zu einer Fehlermeldung, da die Daten nicht an die verteilte Simulation gesendet werden können. Nach diesem Vorgang ist die Erstellung eines parametrisierbaren Umweltszenarios abgeschlossen. Die Umweltdaten können jederzeit während der Laufzeit der Simulation geändert und erneut gesendet werden.

2.4.9.5 Starten des statischen Umweltszenarios

Das statische Umweltszenario wird – analog zum parametrisierbaren Umweltszenario – über die Benutzeroberfläche des Umweltsimulators (vgl. Abbildung 48) gestartet. In dieser gibt es drei Textfelder und drei Buttons (s. Abbildung 60).

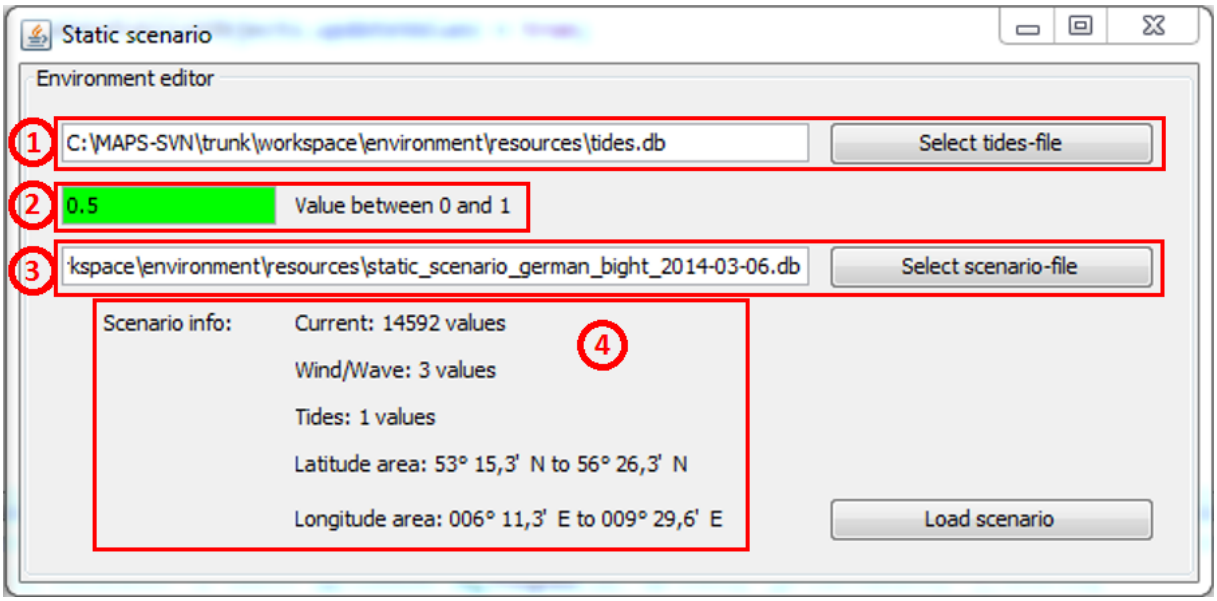

Abbildung 60: Statisches Umweltszenario – Fenster „Static scenario“

Der erste Bereich (Punkt 1 in Abbildung 60) ist zur Auswahl der Tidendatenbank vorgesehen (vgl. dazu Kapitel 3.3.2.2). Über den Button „Select tides-file“ muss eine valide Datenbank ausgewählt werden. Wird eine nicht valide Datenbank ausgewählt, kommt es beim Verschicken der Daten an die Simulation zu Fehlern und zu einem Abbruch des statischen Szenarios. Für die Auswahl der Datenbanken wird ein Dateifenster geöffnet, in dem der Pfad der gewünschten Datenbank aufgerufen werden kann (vgl. Abbildung 61). Ein Doppelklick auf die entsprechende Datei liest die gewünschte Datenbank ein und übernimmt sie in die Benutzeroberfläche des statischen Umweltszenarios.

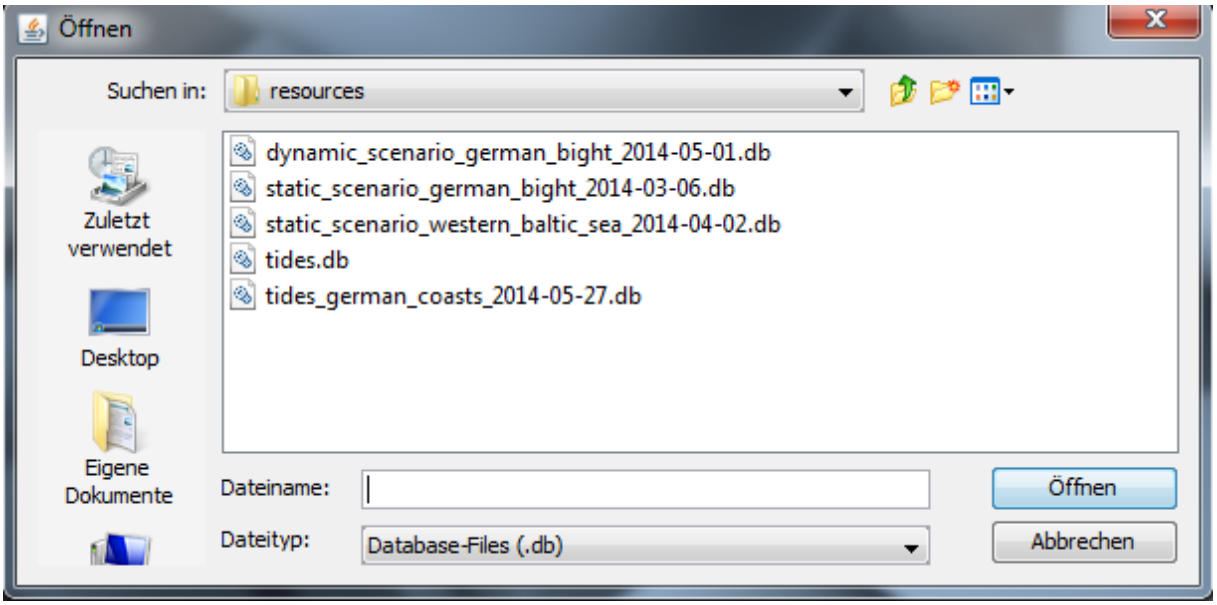

Abbildung 61: Statisches Umweltszenario – Dateifenster zum Öffnen eines statischen Umweltszenarios

In das nächste Textfeld in Bereich 2 (Punkt 2 in Abbildung 60) muss eine Double-Zahl zwischen null und eins (mit Punkt als Trennzeichen) eingegeben werden, die die prozentuale

Höhe der Tide innerhalb der Simulation ausdrückt. In einer Tidendatenbank werden jedem geografischen Punkt dabei eine minimale und eine maximale Tidenhöhe (Ebbe und Flut) zugeordnet. Der prozentuale Wert, dient der Ermittlung eines Wertes zwischen dem Ebbe- und Flutwert.

Beispiel:

- Minimaler Wert (Ebbe): null Meter über LAT (Seekartennull)
- Maximaler Wert (Flut): zwei Meter über LAT
- Nutzerauswahl: 0.5 (entspricht 50 Prozent)
- Tidenhöhe über LAT: ein Meter

Die Eingabe des prozentualen Wertes akzeptiert, wie bereits erwähnt, lediglich Double-Werte zwischen null und eins. Sollte der eingegebene Wert unter null, über eins oder gar keine Double-Zahl sein, wird das Textfeld rot hinterlegt und es kann nicht verlassen werden. Erst wenn ein valider Wert eingegeben wurde, wird das Textfeld grün hinterlegt und es kann verlassen werden.

Das Textfeld in Bereich 3 (Punkt 3 in Abbildung 60) verhält sich entsprechend dem ersten Textfeld. Ihm zugeordnet ist der Button „Select scenario-file“, welcher ebenfalls ein Dateiauswahl-Fenster öffnet. Hier muss nun – bei analogem Vorgehen zur Dateiauswahl einer Tidendatenbank – eine Strömungs- und Winddatenbank ausgewählt und geöffnet werden.

Sind beide Datenbanken korrekt ausgewählt und ein valider Wert für die prozentuale Höhe der Tide eingegeben, kann das Szenario per „Load scenario“-Button bei laufender CERTI-Laufzeitinfrastruktur an die Simulation gesendet werden. Zusätzlich werden Informationen über das ausgewählte Szenario neben dem Label „Scenario info:“ angezeigt (Punkt 4 in Abbildung 60).

Es öffnet sich außerdem ein Map-Fenster mit dem in der Datenbank enthaltenen Gebiet, um einen visuellen Eindruck des von den Umweltdaten beeinflussten Gebietes zu erhalten (vgl. Abbildung 62). Damit ist das Starten eines statischen Szenarios abgeschlossen.

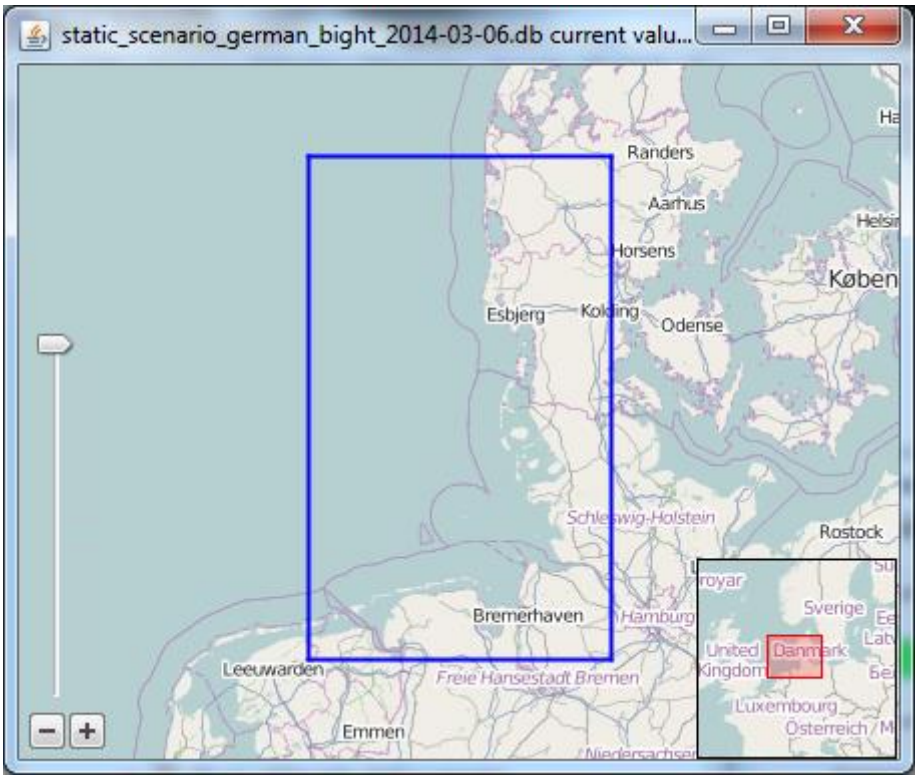

Abbildung 62: Statisches Umweltszenario – Map-Fenster

2.4.9.6 Starten des dynamischen Umweltszenarios

Auch das dynamische Umweltszenario wird über die Benutzeroberfläche des Umweltsimulators gestartet (vgl. Abbildung 48). Daraufhin öffnet sich das Fenster aus Abbildung 63.

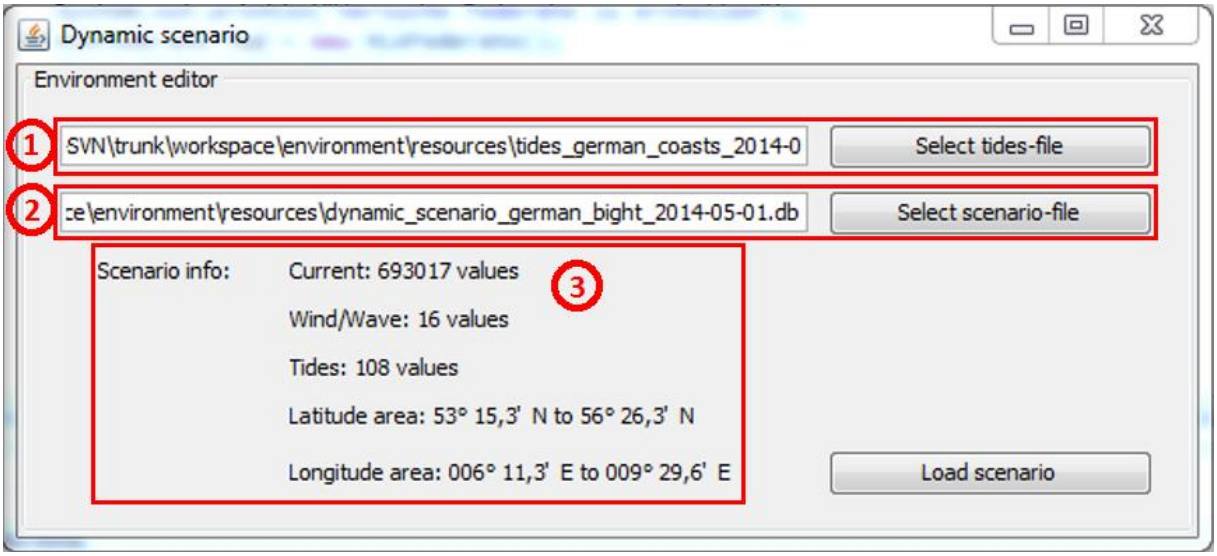

Abbildung 63: Dynamisches Umweltszenario – Fenster „Dynamic scenario“

Wie in der Abbildung zu erkennen ist, besteht die GUI aus zwei Textfeldern und drei Buttons. Bereich 1 (Punkt 1 in Abbildung 63) ist zur Auswahl einer Tidendatenbank über den Button „Select tides-file“ gedacht. Beim Klick auf den Button öffnet sich auch hier wieder ein Dateiauswahl-Fenster in dem der Pfad zur gewünschten Tidendatenbank definiert wird.

In Bereich 2 (Punkt 2 in Abbildung 63) wird eine Strömungs- und Winddatenbank ausgewählt. Über das Dateiauswahl-Fenster, welches durch Betätigen des „Select scenario-file“-Buttons geöffnet wird, muss der Pfad der Datenbank aufgesucht werden, um sie in die Benutzeroberfläche des Dynamic Scenario zu laden.

Sollten valide Datenbanken ausgewählt worden sein, wird neben dem Label „Scenario info:“ (Punkt 3 in Abbildung 63) ein Informationstext über die Ausmaße und Anzahl der Werte in den Datenbanken ausgegeben.

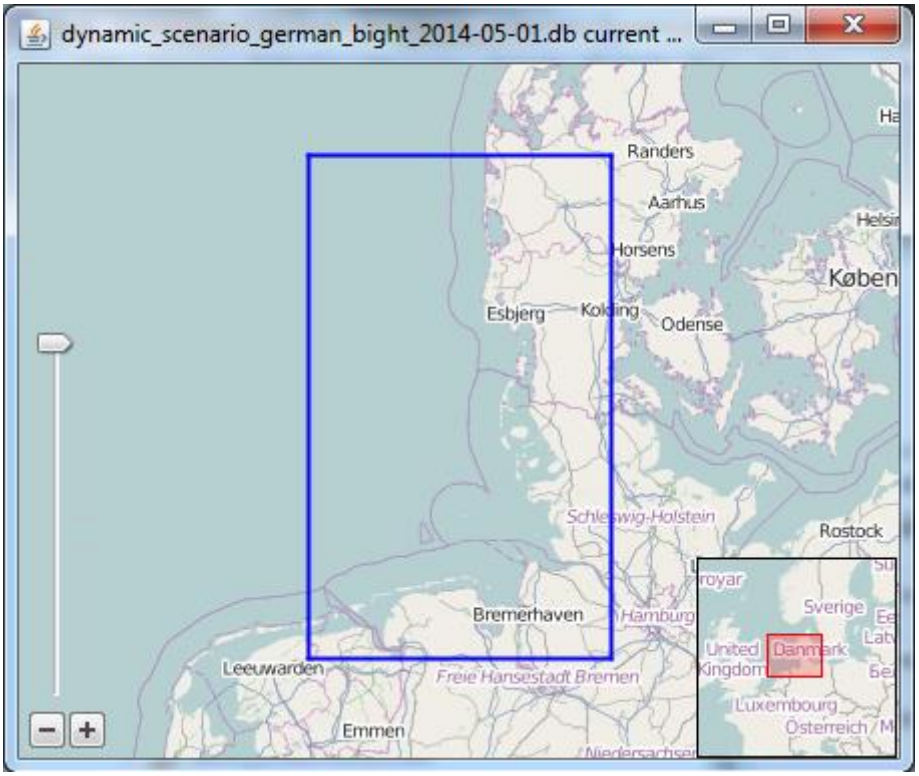

Abbildung 64: Dynamisches Umweltszenario – Map-Fenster

Zudem öffnet sich nach der Auswahl einer Strömungs- und Winddatenbank ein Map-Fenster (s. Abbildung 64). Dieses Fenster zeigt eine Karte mit einem gezeichneten Rechteck. Das Rechteck auf der Karte deutet den Bereich an, der mit Strömungsdaten durch die Datenbanken abgedeckt wird.

Hat sich das Map-Fenster geöffnet und werden die Informationen über das Szenario angezeigt, kann dieses bei laufender CERTI-Laufzeitinfrastruktur über einen Klick auf den „Load scenario“-Button an die MTS gesendet werden. Damit ist das Starten eines dynamischen Szenarios abgeschlossen.

2.4.9.7 Starten des Umweltzonenszenarios

Das Umweltzonenszenario lässt sich ebenfalls über die Benutzeroberfläche des Umweltsimulators starten (vgl. Abbildung 48). Daraufhin öffnet sich das Fenster aus Abbildung 65.

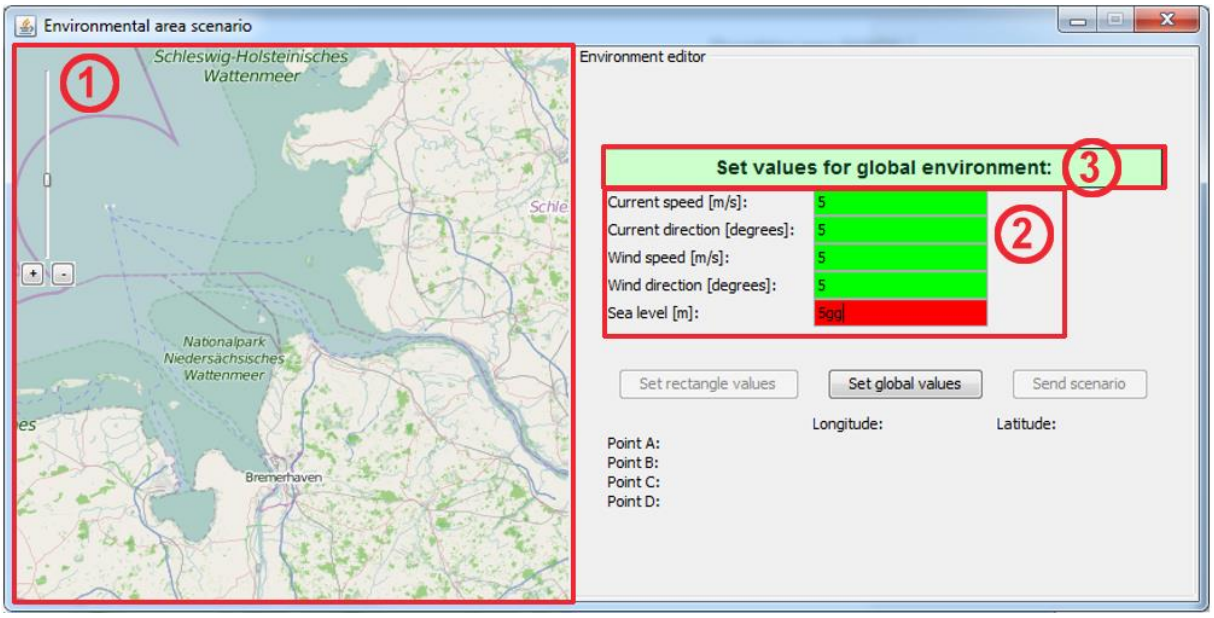

Abbildung 65: Umweltzonenszenario - Hauptfenster

Die Benutzeroberfläche besteht aus zwei Teilen. Auf der linken Seite ist ein Kartenausschnitt zu sehen, der per „Drag&Move“ verschoben werden kann (Punkt 1 in Abbildung 65). Zusätzlich gibt es am linken Rand der Karte eine Zoomleiste, mit der sich der Kartenausschnitt vergrößern und verkleinern lässt.

Auf der rechten Seite der Benutzeroberfläche befinden sich Eingabefelder und Buttons zum Setzen der Parameter für das Umweltzonenszenario. Initial müssen Umweltdatenwerte für Strömungsgeschwindigkeit und -richtung, Windgeschwindigkeit und -richtung sowie Tidenhöhe in die jeweiligen Eingabefelder eingetragen werden (Punkt 2 in Abbildung 65) und im Anschluss per „Set global values“ bestätigt werden. Sind diese Werte einmal bestätigt worden, können sie nicht mehr verändert werden. In den Eingabefeldern werden jeweils Double-Werte erwartet. Sollte ein nicht-valider Wert eingetragen werden, wird das Textfeld rot hinterlegt und es kann nicht verlassen werden bis ein korrekter Wert eingegeben wurde. Wird ein korrekter Wert eingegeben, wird das Textfeld grün hinterlegt und es kann verlassen werden.

Die globalen Werte, welche zunächst gesetzt werden, haben den Zweck, dass diese Umweltdaten außerhalb der Rechtecke gelten und das so an die MTS weitergegeben wird. So wird garantiert, dass auf jedem Punkt der Karte Umweltdaten vorhanden sind. Werden die globalen Werte bestätigt, ändert sich der Informationstext über den Textfeldern von „Set values for global environment“ (Punkt 3 in Abbildung 65) zu „Select an area and set its environment values“ (Punkt 1 in Abbildung 66).

Im Fenster ändert sich bis auf den Inhalt des Informationstextes nichts, außer dass der „Set global values“-Button ausgegraut und der „Set rectangle values“-Button benutzbar wird

(Punkt 2 in Abbildung 66). Um diesen allerdings benutzen zu können, muss zunächst auf der Karte ein Rechteck gezeichnet werden.

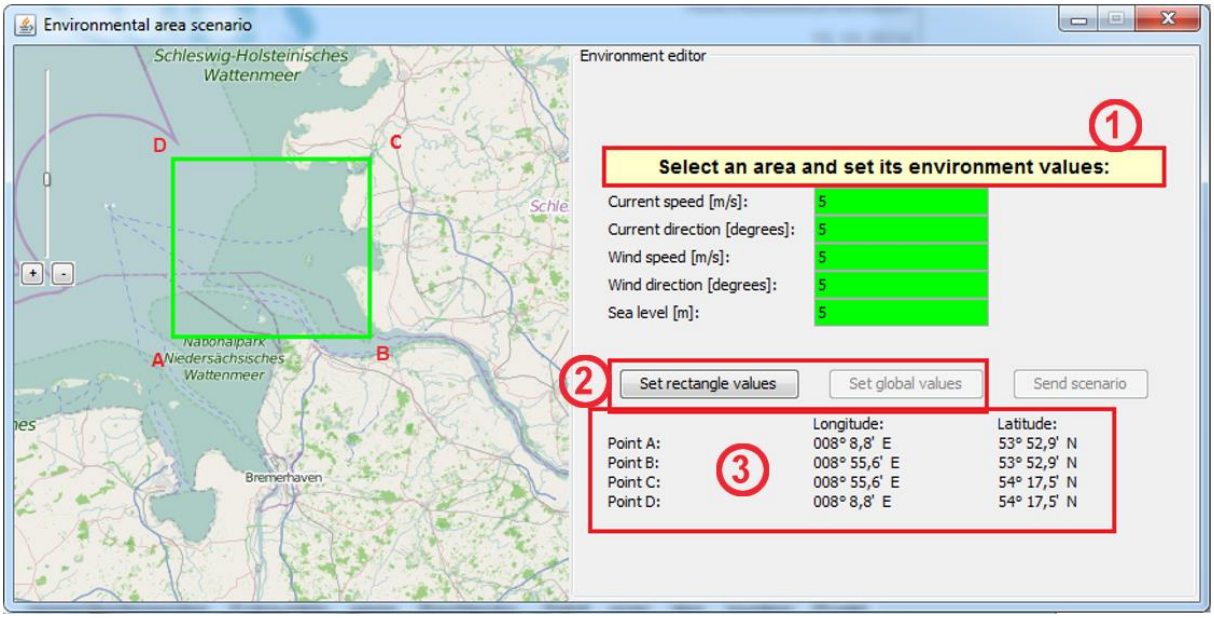

Abbildung 66: Umweltzonenszenario – Zeichnen des Rechtecks

Das Zeichnen des Rechtecks gestaltet sich wie folgt. Es müssen nacheinander zwei beliebige Punkte innerhalb der Karte angeklickt werden. Die beiden Punkte kennzeichnen die beiden gegenüberliegenden Eckpunkte eines Rechtecks. Wird der zweite Punkt beispielsweise höher und weiter rechts auf der Karte gesetzt als der erste Punkt, wird der erste Punkt zu Punkt A (links, unten) und der zweite Punkt wird zu Punkt C (rechts, oben) des Rechtecks aus Abbildung 66. Die anderen beiden Ecken (hier: B und D) des Rechtecks werden automatisch berechnet und hinzugefügt. Zusätzlich werden geografische Informationen (Latitude und Longitude) der jeweiligen Eckpunkte im Informationstext unterhalb der Buttons angezeigt (Punkt 3 in Abbildung 66).

Nach dem Setzen des Rechtecks, müssen die Werte zu den Umweltdaten des spezifischen Rechtecks in den Textfeldern eingegeben werden. Zum Schluss werden das Rechteck und die zugehörigen Werte mit dem Button „Set rectangle values“ bestätigt. Dies führt dazu, dass das Rechteck nicht mehr verändert werden kann, was auch daran erkennbar ist, dass sich die Randfarbe des Rechtecks auf der Karte von grün zu gelb ändert (Punkt 1 in Abbildung 67). Bevor das Rechteck mit „Set rectangle values“ bestätigt wurde, ist es möglich, das Rechteck zu korrigieren. Mit einem Klick auf die mittlere Maustaste wird das soeben gezeichnete Rechteck gelöscht und es kann von neuem gesetzt werden.

Beim Setzen der Rechtecke gibt es ein paar Restriktionen, die das Programm automatisch abfragt und die Operation ggf. verhindert. So können Rechtecke andere Rechtecke nicht überlappen oder diese gänzlich einschließen. Das Programm verhindert dies, indem das Setzen des zweiten Punktes nicht möglich ist.

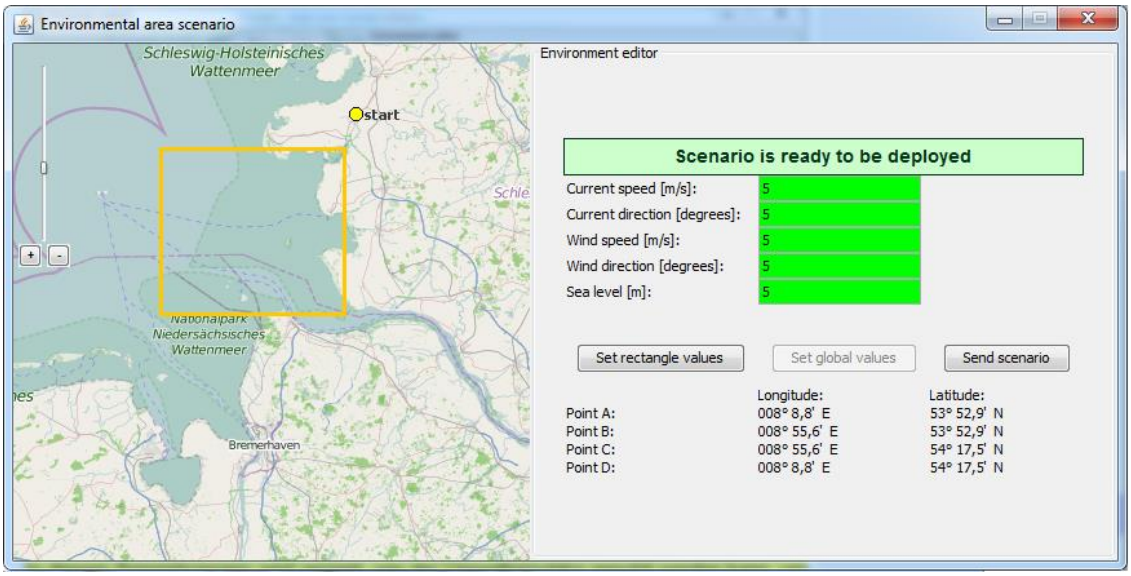

Abbildung 67: Umweltzonenszenario – Senden des Szenarios an die MTS

Sollte mindestens ein korrektes Rechteck angegeben und per „Set rectangle values“ bestätigt worden sein, wird der Button „Send scenario“ aktiv gesetzt. Durch Betätigen dieses Buttons wird das Szenario an die MTS gesendet.

2.4.9.8 Nutzung des Umweltsimulators am Beispiel „Ausbaggerung der Elbe“

In diesem Beispielszenario wird gezeigt, wie der Umweltsimulator genutzt werden kann, um konkrete Anwendungsfälle zu testen. Als Beispiel wird hier ein Szenario gegeben, bei dem ein Teil der Elbe ausgebaggert wird.

Zum Vergleich zwischen ausgebaggertem und nicht ausgebaggertem Zustand ist es notwendig, beide Szenarien einzeln zu simulieren.

1. Vor dem Simulationsstart:

   Als Szenario wird das Umweltzonenszenario gewählt, in welchem Bereiche einzeln definiert werden (vgl. Kapitel 2.4.9.7). Dies erlaubt es, den ausgebaggerten Bereich durch die Rechtecke zu markieren. Zuerst wird das Szenario ohne Ausbaggerung getestet. Hier reicht es, die globalen Werte im Umweltzonenszenario zu setzen (vgl. Abbildung 68). Diese sollten in beiden Simulationsdurchläufen identisch sein, damit die späteren Messungen aussagekräftig sind.

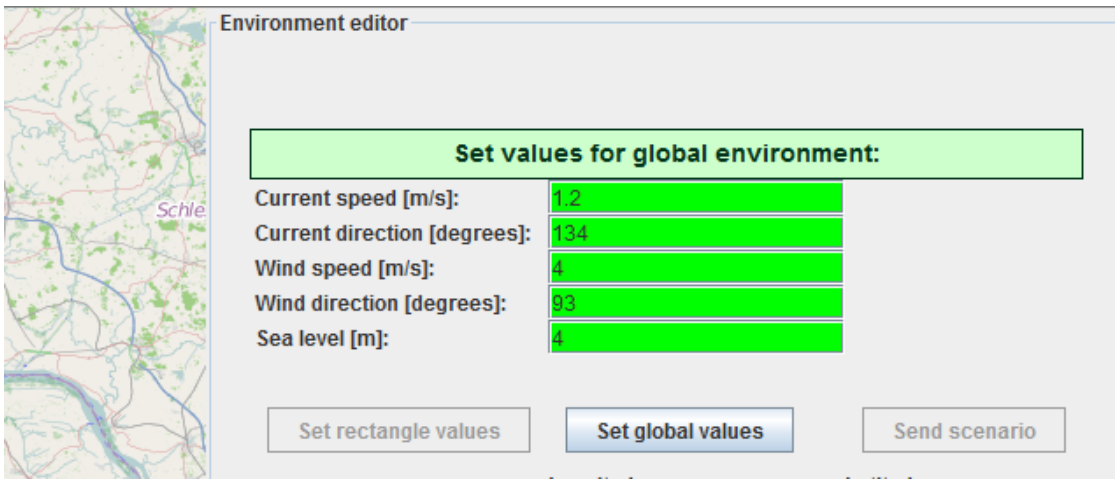

Abbildung 68: Beispiel „Ausbaggerung der Elbe“ - Globale Umweltdaten setzen

2. Einstellen der Analyse-Komponente.
3. Einstellen der EMod-Datei, so dass gewährleistet ist, dass Schiffe durch den ausgebaggerten Bereich fahren.
4. Start der Simulation und Speicherung der Ergebnisse.

Im zweiten Simulationslauf werden dieselben Schiffe bei einem leicht veränderten Umweltszenario durch die Elbe geschickt.

1. Vor dem Simulationsstart:

   Im zweiten Durchlauf wird die Elbe innerhalb der Fahrrinne ausgebaggert. Dies lässt sich durch den Umweltsimulator darstellen, indem auf den Wert des Wasserstandes („Sea level“) der Wert für die auszubaggernde Tiefe addiert wird. Dieser Wert wird nun Rechtecken zugewiesen, die den ausgebaggerten Bereich darstellen. Es müssen also mehrere Rechtecke gesetzt werden, die den ausgebaggerten Bereich der Fahrrinne abdecken (s. Abbildung 69).

Abbildung 69: Beispiel „Ausbaggerung der Elbe“ – Rechtecke um den ausgebaggerten Bereich setzen

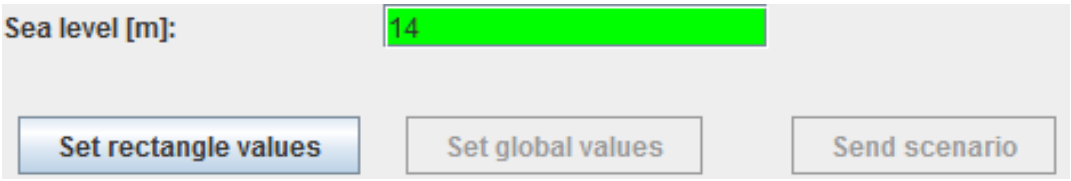

Abbildung 70: Beispiel „Ausbaggerung der Elbe“ – Verändern des "Sea level" für die Rechtecke

2. Gleiche Analyseeinstellungen verwenden wie im vorigen Durchlauf.
3. Ebenfalls dieselbe EMod-Datei verwenden.
4. Start der Simulation.

Nach Beenden des zweiten Simulationslaufs liegen nun zwei Ergebnisse vor. Eines bei normalem Wasserstand der Elbe, und eines bei dem die Elbe – in unserem Beispiel um zehn Meter – ausgebaggert wurde. Aus diesen Werten können nun Schlussfolgerungen über den Effekt, den das Ausbaggern der Elbe mit sich zieht, gezogen werden, indem die Ergebnisse der Analyse-Komponente analysiert werden.

2.4.10 Durchführung eines Analyseszenarios

Die Analyse-Komponente kann entweder per Kommandozeile oder über eine eigene Benutzeroberfläche gestartet werden. Hierbei unterscheidet sich der Funktionsumfang insofern, dass bei der Verwendung der Benutzeroberfläche die Analyse-Komponente zusätzlich noch die Laufzeitinfrastruktur (RTIG) und die anderen Federates startet.

Kommandozeile

Beim Start über die Kommandozeile wird die Analyse-Komponente über die im Hauptverzeichnis liegende Datei „config.json“ konfiguriert. Hier können die Queries, Exporter sowie die FOM-Datei angegeben werden. Es werden standardmäßig keine weiteren Parameter benötigt.

Wenn der CSVExporter verwendet wird, sind in den entsprechenden CSV-Dateien nach dem Simulationslauf die Query-Ausgaben vorhanden. Diese können dann z.B. in MS Excel weiterverarbeitet werden.

Benutzeroberfläche

Mit Verwendung der Benutzeroberfläche werden zusätzlich noch weitere Einstellungen aus der Datei „config.json“ relevant, die unter dem Schlüsselwort „starter“ zu finden sind. Diese betreffen die RTIG und die anderen Federates, die in der verteilten Simulation gestartet werden sollen.

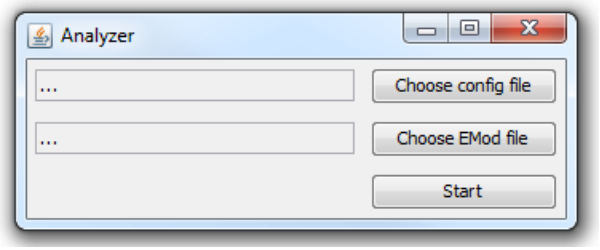

Abbildung 71: Analyse-Komponente – Hauptfenster

Wie in Abbildung 71 ersichtlich wird, können in der Benutzeroberfläche die Konfigurations-Datei und die verwendete EMod-Datei für die Simulation gewählt und die CERTI-Laufzeitinfrastruktur (RTIG) gestartet werden. Nach Klicken auf „Start“ erscheinen anschließend zwei Fenster, die in Abbildung 72 dargestellt werden.

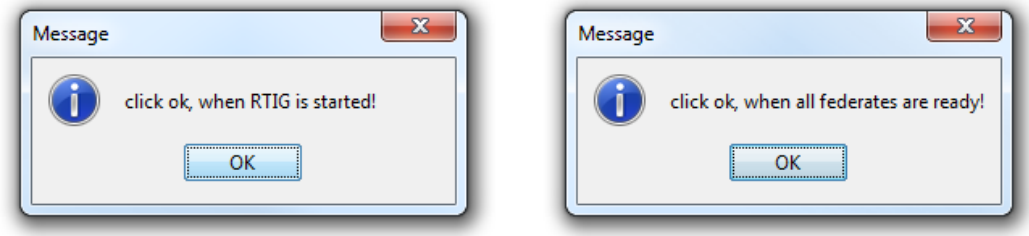

Abbildung 72: Analyse-Komponente – Dialogfenster beim Starten der Analyse

In den Fenstern muss jeweils bestätigt werden, dass das RTIG und die weiteren Federates gestartet wurden. Dies ist notwendig, da automatisiert nicht ermittelt werden kann, ob die Prozesse bereits kommunikationsbereit sind.

2.4.11 Aufruf über Kommandozeilen-Parameter

2.4.11.1 MTS

Die MTS kann über die Kommandozeile mit Parametern gestartet werden. In Tabelle 2 werden diese Parameter erläutert:

Parameter	*Beschreibung*
-ui	Hiermit kann die GUI an- bzw. ausgeschaltet werden.
-emod	Hiermit kann der direkte Pfad zur EMod-Datei angegeben werden (erfolgt nur wenn –ui mit false angegeben wird)
-genNum	Hiermit wird dem LoadGenerator die Anzahl der Schiffe übermittelt (erfolgt nur wenn –ui mit false angegeben wird). Nur in Kombination mit –genTime und –genPercentWaterWay gültig.
-genTime	Hiermit wird dem LoadGenerator die zu simulierende Zeit übermittelt (erfolgt nur wenn –ui mit false angegeben wird). Nur in Kombination mit –genNum und –genPercentWaterWay gültig.
-genPercentWaterWay	Hiermit wird dem LoadGenerator die prozentuale Verteilung der Schiffe auf der Wasserstraße übermittelt. Nur in Kombination mit –genTime und –genNum gültig.
-help	Ruft die Hilfe für die Kommandozeilen-Parameter auf.
-seed	Überschreibt den Seed, der in der Konfigurationsdatei steht.
-cfg	Gibt den Pfad zu der Konfigurationsdatei an (Default: „cfg.xml“)

Tabelle 2: Kommandozeilen-Parameter der MTS

2.4.11.2 Umweltsimulator

Es ist möglich, die meisten Szenarien des Umweltsimulators auch über die Kommandozeile zu starten. Eine Ausnahme ist hier das Umweltzonenszenario, da dieses als eine Art Baukasten für ein variables Szenario verstanden wird. Der Aufbau eines solchen Szenarios lediglich über die Kommandozeile widerspricht dem Sinn und Zweck dieses Tools.

In Tabelle 3 bis Tabelle 5 sind die unterstützten Kommandozeilen-Parameter der einzelnen Szenarien aufgelistet:

Static Scenario

Parameter	*Beschreibung*
-help	Zeigt die Hilfe an.
-scenDb	Pfad zur Szenarien-Datenbank
-tidesDb	Pfad zur Tidendatenbank
-ui	Starte mit („yes“) oder ohne („no“) GUI. Bei „yes“ werden andere Parameter vernachlässigt und die GUI gestartet.

Tabelle 3: Kommandozeilen-Parameter des Umweltsimulators – Static Scenario

Dynamic Scenario:

Parameter	*Beschreibung*
-help	Zeigt die Hilfe an.
-scenDb	Pfad zur Szenarien-Datenbank
-tidesDb	Pfad zur Tidendatenbank
-ui	Starte mit („yes“) oder ohne („no“) UI. Bei "yes" werden andere Parameter vernachlässigt und die GUI gestartet.

Tabelle 4: Kommandozeilen-Parameter des Umweltsimulators – Dynamic Scenario

Parametrizable Environment Scenario:

Parameter	*Beschreibung*
-help	Zeigt die Hilfe an.
-cSpeed	Strömungsgeschwindigkeit
-cDir	Strömungsrichtung
-wSpeed	Windgeschwindigkeit
-wDir	Windrichtung
-sLevel	Wasserstand über/ unter LAT
-ui	Starte mit („yes“) oder ohne („no“) GUI. Bei "yes" werden andere Parameter vernachlässigt und die GUI gestartet.

Tabelle 5: Kommandozeilen-Parameter des Umweltsimulators – Parametrizable Environment Scenario

2.4.12 Konfigurationsdatei des MTS

Nachfolgend wird ein Teil der Konfigurationsdatei abgebildet und anschließend erläutert. Bei dem abgebildeten Teil handelt es sich um die entsprechenden Abschnitte der Konfiguration, die in diesem Dokument nicht weiter betrachtet werden. Dabei unterteilt sich die Konfigurationsdatei in die zwei Bereiche „<simulation>“ (vgl. Abbildung 73) und „<visualization>“ (vgl. Abbildung 74).

```
<mts-cfg>
	<simulation>
		<map>
		…
		</map>
		<seed value=""/>
		<threadable value="false"/>
		<defaultTime value="500" unit="Milliseconds"></defaultTime>
		<processorLoading value="75"/>
		<hlaTimeUnit value="Seconds"/>
		<exchangers>
			<exchanger value="de.wi_ol.be.mts.exchangers.HLAExchange"
enable="true"/>
			<exchanger value="de.wi_ol.be.mts.exchangers.DummyExchange"
enable="false"/>
		</exchangers>
		<viewfactorys>
			<viewfactory vesselId="1" name="Schiff ohne Umwelt"
value="de.wi_ol.be.mts.views.VesselSettingRateViewFactory" enable="true"/>
			<viewfactory vesselId="3" name="Umwelt-Schiff"
value="de.wi_ol.be.mts.views.VesselSettingRateEnvironmentInfluenceViewFactory"
enable="true"/>
		</viewfactorys>
		<vesselgenerator>
			<mintarget value="3"/>
			<maxtarget value="10"/>
			<harbours>
			…
			</harbours>
		</vesselgenerator>
		<collisionSystems>
			…
		</collisionSystems>
		<trafficRegulation>
			…
		</trafficRegulation>
	</simulation>
	…
```

Abbildung 73: Konfigurationsdatei der MTS – Bereich „<simulation>“

Im Bereich „<simulation>“ (vgl. Abbildung 73) befindet sich der Abschnitt „<map>“, der in Kapitel 2.4.1 beschrieben ist. Mit dem Tag „<seed>“ kann ein Seed vergeben werden, der für die Zufallsgenerierung genutzt wird. Der Tag „<threadable>“ beschreibt, ob Threads für die Wegfindung genutzt werden sollen. Insbesondere beim GeneratedAgent kann es bei eingeschalteten Threads zu keiner deterministischen Simulation kommen. Der Tag „<defaultTime>“ beschreibt, in welchen Intervallen die Simulation ablaufen soll. Der Tag „<processorLoading>“ definiert in Prozent, wie viele Prozessoren für Threads definiert werden sollen, d.h. wie viele Threads gleichzeitig laufen können. Wenn threadable deaktiviert ist, ist die Angabe obsolet. Mit dem Tag „<hlaTimeUnit>“ wird die Einheit der Zeit angegeben, die HLA nutzt. Im Abschnitt „<exchangers>“ können verschiedene Schnittstellen definiert werden, die genutzt werden, um die Simulationsdaten zu veröffentlichen. Ein Exchanger ist der HLA-Standard. Hier kann auch HLA ausgeschaltet werden (Attribut

„enable“). Der Abschnitt „<viewfactorys>“ definiert alle verfügbaren ViewFactorys, die die Simulation nutzen kann. Auch hier können die einzelnen ViewFactorys ausgeschaltet werden (Attribut „enable“). Den einzelnen Views können zudem Visualisierungsoptionen mitgegeben werden (Attribut „vesselId“), um die verschiedenen Typen zu unterscheiden. Der Abschnitt „<vesselgenerator>“ ist für den LoadGenerator von Bedeutung. Dabei geben die Tags „<minTarget>“ und „<maxTarget>“ die minimale bzw. maximale Anzahl von Zielen für ein Schiff an. Die Tags „<collisionSystems>“ und „<trafficRegulation>“ werden im Kapitel 3.3.1.1.5 bzw. 3.3.1.1.6 beschrieben.

```
…
<visualization>
            <vessels>
                  <vessel id="1" img="resources\ui\vessel_yellow.png"
                  colorName="yellow"/>
                  <vessel id="2" img="resources\ui\Schiff_fuchsia.png"
                  r="255" g="0" b="255"/>
                  <vessel id="3" img="resources\ui\Schiff_green.png" r="0"
                  g="128" b="0"/>
                  <vessel id="4" img="resources\ui\Schiff_purple.png" r="128"
                  g="0" b="128"/>
                  <vessel id="5" img="resources\ui\Schiff_red.png" r="255"
                  g="0" b="0"/>
                  <vessel id="6" img="resources\ui\Schiff_teal.png" r="0"
                  g="128" b="128"/>
                  <vessel id="7" img="resources\ui\Schiff_black.png" r="0"
                  g="0" b="0"/>
                  <vessel id="8" img="resources\ui\Schiff_blue.png" r="0"
                  g="0" b="255"/>
            </vessels>
            <realVesselImages path="resources\ui\realvessels\"
defaultImage="resources\ui\realvessels\default.png"/>
            <layers>
                  <waterLayer colorName="cyan" title="Water"/>
                  <landLayer r="139" g="69" b="19" title="Land"/>
                  <dredgedWaterLayer colorName="blue" title="Dredged Water"/>
                  <depthLayer colorName="black" title="Depth" />
                  <buoyLateral colorName="black" title="Buoy, lateral" />
            </layers>
            <showAngularMinute value="false"/>
      </visualization>
</mts-cfg>
```

Abbildung 74: Konfigurationsdatei der MTS – Bereich „<visualization>“

Der Bereich „<visualization>“ (vgl. Abbildung 74) ist für die Visualisierung von Bedeutung. Im Abschnitt „<vessels>“ können verschiedene Bilder bzw. Farben von Schiffen definiert werden, die anschließend einer ViewFactory zugewiesen werden (s. Attribut „vesselId“ im Tag „<viewfactory>“). Der Tag „<realVesselImage>“ beschreibt den Pfad, in dem die Bilder der Schiffe gespeichert werden, um diese im „Vessel Properties“-Reiter der MASON-Konsole

anzuzeigen. Hier kann außerdem noch ein Standardbild angegeben werden, das angezeigt wird, wenn kein Bild gefunden wird. Im Abschnitt „<layers>“ können die Layer der Visualisierung konfiguriert werden. Hier können zum einen der angezeigte Name des Layer und zum anderen die Farbe des Layer bearbeitet werden. Mit dem Tag „<showAngularMinute>“ kann definiert werden, ob die angezeigten Koordinaten in Dezimalgrad (Attribut „value“ auf false setzen) oder in Grad, Minuten und Sekunden angezeigt werden sollen.

Die XML-Datei ist dabei nicht ausgelegt, um Leerzeichen zu verarbeiten. D.h. dass z.B. Leerzeichen in Pfad für die Karten unzulässig sind. Dies liegt daran, dass der Shape-File Importer von GeoMason genutzt wird und dieser akzeptiert lediglich URLs, die keine Leerzeichen beinhalten dürfen.

3. Entwicklerhandbuch

Das Entwicklerhandbuch richtet sich an Personen, die zukünftig mit der MTS, dem Umweltsimulator und der Analyse-Komponente arbeiten bzw. die Software fortschreiben möchten. So wird zunächst erklärt, wie die eigene Entwicklungsumgebung eingerichtet wird, um mit der Entwicklungsarbeit beginnen zu können. Im Anschluss wird beschrieben, wie weitere Software-Komponenten über den HLA-Standard angebunden werden können und wie die bereits bestehenden Software-Komponenten selbst modifiziert und erweitert werden können.

3.1 Entwicklungsumgebung

In diesem Kapitel werden die Informationen gegeben, die benötigt werden, um sich in den Repository-Strukturen der PG MAPS zurecht zu finden und die eigene Entwicklungsumgebung einzurichten.

3.1.1 Inhalt und Struktur

Der Einstiegspunkt, um an die Arbeit der PG MAPS anzuknüpfen, ist das Repository. In Tabelle 6 werden die Ordner der obersten Ebene dargestellt, in denen zukünftige Entwickler alle Dateien der Entwicklungsarbeit wiederfinden.

Ordner	***Beschreibung***
documents	Hier befinden sich Dokumentationen, Protokolle, Berichte und Abgaben.
old	Dieser Ordner beinhaltet die alte Repository-Struktur. Er wurde behalten, damit die Verweise aus alten Dokumenten ihre Gültigkeit behalten.
releases	Hier sind alle Releases der MTS zu finden.
source	Beinhaltet den Quelltext sämtlicher Projekte.
src	Ist wie der Ordner „old“ noch aus historischen Gründen

	vorhanden.
tools	Alle verwendeten Software-Werkzeuge sind hier zu finden.
vendor	Hier befinden sich die extern zur Verfügung gestellten Bibliotheken und Tools.

Tabelle 6: Ordnerstruktur der Entwicklungsumgebung

Zudem befinden sich im Wurzelverzeichnis noch die Datei „buildAll.bat“, die notwendig ist, um Releases aus dem aktuellen Quellcode in den Ordner „tmpartifacts“ zu erstellen, und die Datei „eclipse.bat“, welche alle Projekte baut und diese dann in Eclipse startet.

3.1.2 Einrichtung der Entwicklungsumgebung

Die benötigte Software zum Entwickeln wird im Repository bereitgestellt. Zudem muss noch eine 32-Bit Java Version 7 installiert werden. Die mitgelieferte Software umfasst:

- das Build-Tool Maven,
- diverse Batchscripts,
- die HLA-Implementierung CERTI,
- die Werkzeuge aus dem MinGW Projekt, die für CERTI benötigt werden,
- und eine angepasste Eclipse-Version mit vorinstallierten Plugins.

Für die Verwendung von CERTI und den HLA Federates ist notwendig, dass die MinGW-Bibliotheken in der Path-Variable des Betriebssystems gesetzt werden.

Folgend wird beschrieben, wie die Entwicklungsumgebung eingerichtet wird.

1. SVN auschecken
2. JAVA_HOME setzen (Anleitung siehe: http://www.robertsindall.co.uk/blog/setting-java-home-variable-in-windows/)
3. Im Verzeichnis "trunk" die Datei "eclipse.bat" starten. Es werden alle Referenzen der Projekte heruntergeladen (nur einmal). Anschließend startet Eclipse.
4. WICHTIG: Wenn Eclipse startet, muss immer der Workspace aktualisiert werden (Rechtsklick im Package Explorer → Refresh). Das sollte auch immer geschehen, wenn ein SVN-Update gemacht wurde.
5. Zu Beginn kann es vorkommen, dass eine falsche JDK-Compliance eingestellt ist. In diesem Fall hilft ein Rechtsklick auf das Projekt → Properties → Java Compiler. Dort muss der Compiler compliance level auf 1.7 eingestellt werden.
6. Es müssen händisch die HAGGIS Modelle aus /tmp/model eingefügt werden:
    a. Package „de.emir.model“: Hier befinden sich die YML- und Ecore-Modelle.
    b. Alle Projekte, deren Name „de.emir.model.*“ entspricht.
7. Um YML oder Ecore-Modelle zu modellieren, muss das Package aus Punkt 6a eingebunden sein.
8. Um über HLA Daten austauschen zu können, müssen die Projekte aus den Punkten 6a und 6b eingebunden sein (jeweils nur diejenigen, die man nutzt).
9. Damit die CERTI-Laufzeitinfrastruktur funktioniert, muss MinGW (Pfad: „/tools/MinGW_OFFIS/bin“) in den Path eingefügt werden.

3.1.3 HAGGIS-Datenmodell aktualisieren

Im Folgenden wird erklärt, wie das HAGGIS-Datenmodell aktualisiert wird:

- Die neuen Ecore- und YML-Modelle müssen in das Verzeichnis „/tmp/model/de.offis.haggis.model“ kopiert und die alten Modelle überschrieben werden.
- Es müssen Java-Klassen aus dem Ecore-Genmodel generiert werden (Rechtsklick auf Wurzelknoten → Generate Model Code).
- Diese Klassen sollen in die jeweiligen Ordner unter „trunk\tmp\model*“ abgelegt werden.
- Eine jar-Datei mit den aktuell generierten Java-Klassen muss erstellt werden – das betrifft sowohl „.class“-Dateien (bin) als auch „*.java“-Dateien (src).
- Diese müssen dann in das Verzeichnis „/vendor/haggis“ in eine Datei mit dem Namen "haggis-custom.jar" abgespeichert werden.

3.1.4 Automatische Builds erzeugen

Als Werkzeug zum Erzeugen automatischer Builds wird Maven verwendet. Im Ordner „source“ liegt eine projektübergreifende „Parent POM“ (Project Object Model), auf die sich die einzelnen projektbezogenen POMs beziehen. Umgekehrt müssen neue Projekte auch in der „Parent POM“ eingetragen werden, damit sie für das automatische Bauen bekannt sind. Zudem muss die Datei „buildAll.bat“ angepasst werden. Diese baut auf Maven auf und führt zusätzlich noch Dateisystemoperationen für ein autonomes Release durch.

3.2 Kommunikation über HLA

In der durch die PG MAPS entwickelten Simulationsumgebung stellt jede eigenständige Software-Komponente – MTS, Umweltsimulator, Analyse-Komponente sowie Hilfs-Komponenten – einen Federate dar, der über die CERTI-Laufzeitinfrastruktur im HLA-Standard mit den anderen Federates kommuniziert, d.h. Datenaustausch betreibt. Genau wie die durch die Projektgruppe entwickelten (und im Satz vorher genannten) Software-Komponenten müssen alle weiteren (zukünftig entwickelten) eigenständigen Software-Komponenten ebenfalls als Federates an die CERTI-Laufzeitinfrastruktur angebunden werden.

Um einen neuen Federate zu erstellen, müssen alle dll- und exe-Dateien aus dem Verzeichnis „tools\CERTI\bin“ in das Verzeichnis des Projekts kopiert werden. Zudem wird noch die Datei „hlafederate.jar“ aus dem Verzeichnis „vendor\hla_wrapper“ benötigt. Hierauf aufbauend kann der MTSHLAFederate aus dem HLA-Projekt verwendet werden. Dieser kapselt die Kommunikation mit den anderen Federates in eine einfache Schnittstelle. Zur konkreten Verwendung sei auf die Federates der vorhandenen Projekte der PG MAPS verwiesen, die hier Orientierung bieten.

3.3 Erweiterung der einzelnen Komponenten

Neben der Anbindung neuer Software-Komponenten an die verteilte Simulation als Federates, ist es möglich, die existierenden Software-Komponenten abzuändern oder zu erweitern. In diesem Kapitel wird auf die Frage eingegangen, wie die MTS, ihre

Visualisierung, der Umweltsimulator und die Analyse-Komponente anhand von Beispielen erweitert und ergänzt werden können.

3.3.1 Maritime Traffic Simulation

3.3.1.1 Simulations-Komponente

3.3.1.1.1 Erweiterung des Kartenmaterials

Um weitere Layer in der Simulation nutzen zu können, sind die folgenden Anpassungen zu tätigen: Zunächst muss ermittelt werden, wie der Name des Layer im S-57-Standard ist [Cr14]. Dieser muss anschließend in ein Shapefile transformiert werden, wie in Kapitel 2.4.1 beschrieben. Anschließend muss die Klasse „MapLoader“ (Package „de.wi_ol.be.mts.dataloader“) erweitert werden (s. Abbildung 75). Hier muss ein neues StaticVectorField für den Layer angelegt werden, das ein GeomVectorField beinhaltet. In dieses GeomVectorField müssen die Geometrien eingefügt werden, die mit dem Shapefile-Reader (Package „de.wi_ol.be.mts.dataloader.ShapeFile“) eingelesen wurden. Anschließend muss eine neue Komponente erstellt werden, die das Interface „Component“ (Package „de.wi_ol.be.mts.ecs.Component“) implementiert. Diese Komponente wird einer neuen Entity hinzugefügt, die diese als solches markiert. In Abbildung 75 wird dieses Vorgehen exemplarisch für Leuchttürme dargestellt.

```
StaticVectorFieldComponent lightninghouse = new StaticVectorFieldComponent(new GeomVectorField());
…

ShapeFile file = null;
try {
        file = new ShapeFile(path + "\\" + S57.LIGHTNINGHOUSE + S57.FILE_SHAPE);
        GeoMasonUtil.addGeometries(file.getField(), water.getField());
    } catch (NoSuchFileException ex) {
        log.warn("For Folder: " + path + " The Layer: " + S57. LIGHTNINGHOUSE + S57.FILE_SHAPE + " not found");
    }
…
Entity e = new Entity();
e.addComponent(lightninghouse);
e.addComponent(new Lightninghouse());
entities.add(e);
```

Abbildung 75: Erweitern des Kartenmaterials – Klasse „MapLoader“

Damit unter anderem Agenten mit diesem Layer interagieren können, muss dieser noch zusätzlich zur ScenarioMap (Package „de.wi_ol.be.mts.agent.sensors“) hinzugefügt werden. Dies geschieht über die Methode „createScenarioMap“ in der Klasse „ScenarioDescription“ (Package „de.wi_ol.be.mts.ecs“). In Abbildung 76 wird auch dies wieder exemplarisch am Beispiel für Leuchttürme dargestellt. Wie die neuen Layer zur Visualisierung hinzugefügt werden, wird in Kapitel 3.3.1.2 beschrieben.

```
…
else if (e.hasComponent(Lightninghouse.class)) {
        this.scenarioMap.addLightninghouse(field.getField().getGeometries());
}
…
```

Abbildung 76: Erweitern des Kartenmaterials – Klasse „ScenarioDescription“

3.3.1.1.2 Erweiterung der Agenten

Dieses Kapitel befasst sich mit der Erweiterung der Teilkomponente Agenten. Abbildung 77 zeigt einen Teilausschnitt des Klassendiagrams der MTS. Das gesamte Architekturdiagramm ist in Kapitel 2.1.3.1 überblicksartig beschrieben.

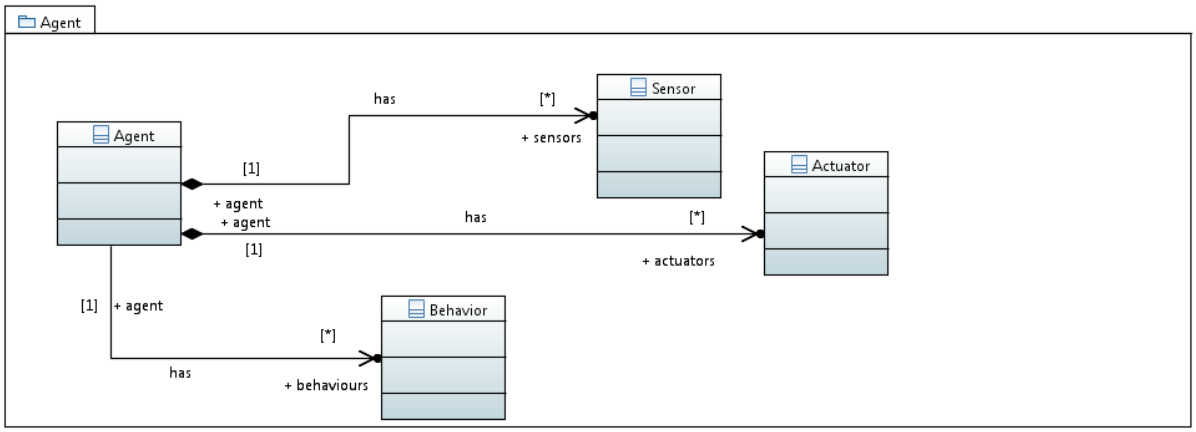

Abbildung 77: Architektur der Agentenlogik im MTS

Nachfolgend werden die Begriffe und Konzepte der Teilkomponente „Agenten“ eingeführt, um anschließend die Erweiterung der Teilkomponente in die verschiedenen Ausprägungen anschaulich darzustellen. Die Teilkomponente „Agenten“ stellt eine Steuereinheit mit verschiedenen Intelligenz-Ausprägungen dar, die ein Schiff im Rahmen seiner physikalischen Gegebenheiten steuern kann. Dabei orientiert sich der Agent an einem Ziel (z.B. Zielort oder Zielrichtung/-geschwindigkeit). Ein Agent ermöglicht es somit, mit Hilfe von künstlicher Intelligenz einen Kapitän bzw. Steuermann für ein Schiff nachzubilden. Die Teilkomponente „Agenten“ beinhaltet im Wesentlichen die Konzepte „Agent“, „Behavior“, „Sensor“ und „Actuator“. Der Agent kapselt alle Bestandteile (Behaviors, Sensors, Actuators). Zusätzlich wird in ihm definiert, ob und wie er von außen gesteuert werden kann. Die Behaviors eines Agenten bilden einzelne Verhaltensweise ab. In Summe ergeben sie die Intelligenz des Agenten. So können den Agenten verschiedene Intelligenz-Ausprägungen zugewiesen werden. Dadurch wird ermöglicht, dass Agenten, die die Ziele auf gleiche Weise übermittelt bekommen, unterschiedliche Lösungen ermitteln. Durch die Actuators (z.B. Steuerrad oder Schubhebel) kann der Agent die Schiffskomponenten (z.B. Ruder oder Motor) beeinflussen. Vorteil dieses Konzept ist, dass der Agent die physikalischen Gegebenheiten eines Schiffs nicht umgehen kann. Die Sensors dienen der Erhebung von Schiffs- und Umgebungsinformationen mit Hilfe von Sensoren (z.B. Echolot und GPS).

Ausgangspunkt für die Erweiterung der Agenten ist die abstrakte Klasse „Agent“ (Package „de.wi_ol.be.mts.agent“). Von dieser Klasse muss jeder neue Agent ableiten. Wenn es sich um einen automatisierten Agenten handelt, sollte von der Klasse „StandardAgent“ (Package „de.wi_ol.be.mts.agent“) abgeleitet werden, oder – bei einer neuen Verhaltenslogik – eine neue Behavior entworfen werden. Der StandardAgent hat bereits alle notwendigen Methoden, um verschiedene Verhalten (Behavior) und verschiedene Wegfindungs-Algorithmen auszuführen. Bei neuen Agenten muss auch das HAGGIS-Datenmodell um den Agenten erweitert werden, damit dieser in einer EMod-Datei definiert werden kann. Beim Starten der Simulation werden die Agenten aus der EMod-Datei zu den internen Agenten gemappt. Das Mapping findet in der Methode „createEntityFromVesselComponent“ in der

Klasse „ScenarioDescription“ (Package „de.wi_ol.be.mts.ecs“) statt. Das Mapping wird in Abbildung 78 exemplarisch für den NMEA Agent dargestellt. Hierbei wird ein Briefing für den Agenten erstellt, der den Agententypen sowie Informationen zum Schiff beinhaltet.

```
else if (v.vessel.getAgentType() instanceof NMEAAgent) {
      VesselComponentInformation info = new VesselComponentInformation(v);
      briefing = new Briefing<InternalNMEAAgent,
VesselComponentInformation>(InternalNMEAAgent.class, info);
}
```

Abbildung 78: Erweiterung des Mapping vom Agenten des HAGGIS-Datenmodells zum internen Agenten

Notwendig bei den Agenten ist, dass in der „start“-Methode die Variable „started“ auf true gesetzt wird. Das gibt der Simulation den Hinweis, dass der Agent nun bereit ist und die Simulation gestartet werden kann. Weiterhin ist die Methode „pullNotices“ von großer Bedeutung. Diese plant sogenannte Notices ein, die ein Ereignis (Event) für eine bestimmte Zeit einplanen. Die Events können unter anderem ein Steuerbefehl sein oder das Zeichen geben, dass der Anker gelegt werden soll. Die Methode „pullNotices“ ist also dazu da, Manöver einzuplanen. Die Events werden dann zur angegebenen Zeit ausgeführt bzw. der Agent wird angetriggert, dass dieser das Event durchführt. Für die Durchführung der Events muss es eine entsprechende Methode mit der Annotation „@OnEvent“ (Package „de.wi_ol.be.mts.ecs.annotations“) geben, die als einzigen Parameter das Event selbst erhält. Auch die Events können erweitert werden. Dafür ist lediglich eine Ableitung der Klasse „Event“ (Package „de.wi_ol.be.mts.ecs“) notwendig. In Abbildung 79 ist ein Auszug aus der Klasse des NMEA Agent zu sehen. Wie bereits oben erwähnt, muss hier die Variable „started“ auf true gesetzt werden. Die Methode „pullNotices“ liefert keine Notices, da der NMEA Agent keine Manöver plant. Die Methode „receiveNMEAMessage“ wird aufgerufen, wenn eine neue NMEAMessage von einem Federate gesendet wird.

```
public class InternalNMEAAgent extends Agent {
	private VesselComponentInformation information;
	public InternalNMEAAgent(final VesselComponentInformation information) {
		this.information = information;
	}

	…

	@Override
	public void start() {
		this.started = true;
	}

	@Override
	public void finish() {

	}

	@Override
	public Set<Notice> pullNotices(final double time, final World world) {
		return new HashSet<>();
	}

	…
	public void receiveNMEAMessage(final NMEAMessage message) {
	…
	}
```

Abbildung 79: Auszug der Klasse des NMEA Agent

Die Agenten können außerdem Aktuatoren haben, mit denen die Agenten auf Komponenten in der Simulationswelt selbst einwirken können. Die Aktuatoren sind als Variablen definiert und mit der Annotation „@Actuator“ (Package „de.wi_ol.be.mts.ecs.annotations“) markiert. Diese werden über Reflection beim Erstellen der Agenten (Methode „connectAgents“ in Package „de.wi_ol.be.mts.ecs.WorldRuntimeData“) instanziiert. In der Methode „connectAgents“ müssen keine Änderungen durchgeführt werden. In den Aktuator-Klassen können Komponenten (abgeleitete Klassen von Components) definiert werden, die auch über Reflection gesetzt werden. Mit der Annotation „@ProvideMap“ wird dem Agenten Kartenmaterial zur Verfügung gestellt.

Das Verhalten der Agenten wird durch Behaviors abgebildet. Hier wird zwischen strategischen und steuernden Behaviors unterschieden. Als erstes werden die strategischen Behaviors ausgeführt und anschließend die Steuernden. Ein Behavior kann andere Behaviors (seiner Art) blockieren, so dass diese nicht zum Einsatz kommen. Wenn ein Behavior das Interface „BlockableBehavior“ (Package „de.wi_ol.be.mts.agent.Behavior“) implementiert, kann es auch von Behaviors anderer Arten blockiert werden. Die strategischen Behaviors sind dazu ausgelegt, die Koordinate, die angefahren wird, zu berechnen. Diese Behaviors werden von der abstrakten Klasse „InternalStrategicalBehavior“ abgeleitet. Die steuernden Behaviors leiten von der Klasse „InternalSteeringBehavior“ ab und berechnen die benötigten Manöver, um die in den strategischen Behaviors festgelegte Koordinate am besten zu erreichen. Diese Behaviors sind somit komplett austauschbar mit anderen Behaviors. Damit die Agenten mit verschiedenen Behaviors ausgestattet werden

können, müssen diese auch im HAGGIS-Datenmodell abgebildet werden. Weiterhin müssen die Behaviors auch vom Datenmodell zu den internen Behaviors gemappt werden. Das Mapping wird in der Methode „createEntityFromVesselComponent“ der Klasse „ScenarioDescription“ festgelegt. In Abbildung 80 wird das Mapping für die Behaviors dargestellt.

```
else if (cap instanceof NewBehavior) {
	behaviors.add(InternalNewBehavior.class);
}
```

Abbildung 80: Erweiterung des Mappings von Behaviors

3.3.1.1.3 Erweiterung der Wegfindung

Die Wegfindung der Agenten kann ebenfalls erweitert werden. Dazu muss das Interface „IWayFinder“ (Package „de.wi_ol.be.mts.agent.wayfinder“) implementiert werden. Die Durchführung der Wegfindung kann dann in der „start“-Methode implementiert werden. Die Methode „getSortedCoordinate“ gibt eine sortierte Liste mit den Routenpunkten zurück. Die Methode „getFindedRoute“ gibt eine Linie als JTS-Geometry zurück. Genau wie die Agententypen und die Behaviors, kann der Wegfindungs-Algorithmus in der EMod-Datei angegeben werden. Dementsprechend müssen auch das HAGGIS-Datenmodell und das Mapping erweitert werden. Das Mapping hierzu befindet sich ebenfalls in der Methode „createEntityFromVesselComponent“ der Klasse „ScenarioDescription“. In Abbildung 81 wird das Mapping für die Wegfindung dargestellt.

```
else if (v.vessel.getAgentType().getWayFinder() instanceof NewWayFinder) {
	wayFinderClass = InternalNewWayFinder.class;
}
```

Abbildung 81: Erweiterung des Mappings von Wegfindungen

3.3.1.1.4 Erweiterung der Physik

Dieses Kapitel befasst sich mit der Erweiterung des physikalischen Modells. Unter Erweiterungen sind auch komplette Neuimplementierungen von physikalischen Modellen zu verstehen. Abbildung 82 zeigt einen Teilausschnitt des Klassendiagrams der MTS. Das gesamte Architekturdiagramm ist in Kapitel 2.1.3.1 überblicksartig beschrieben.

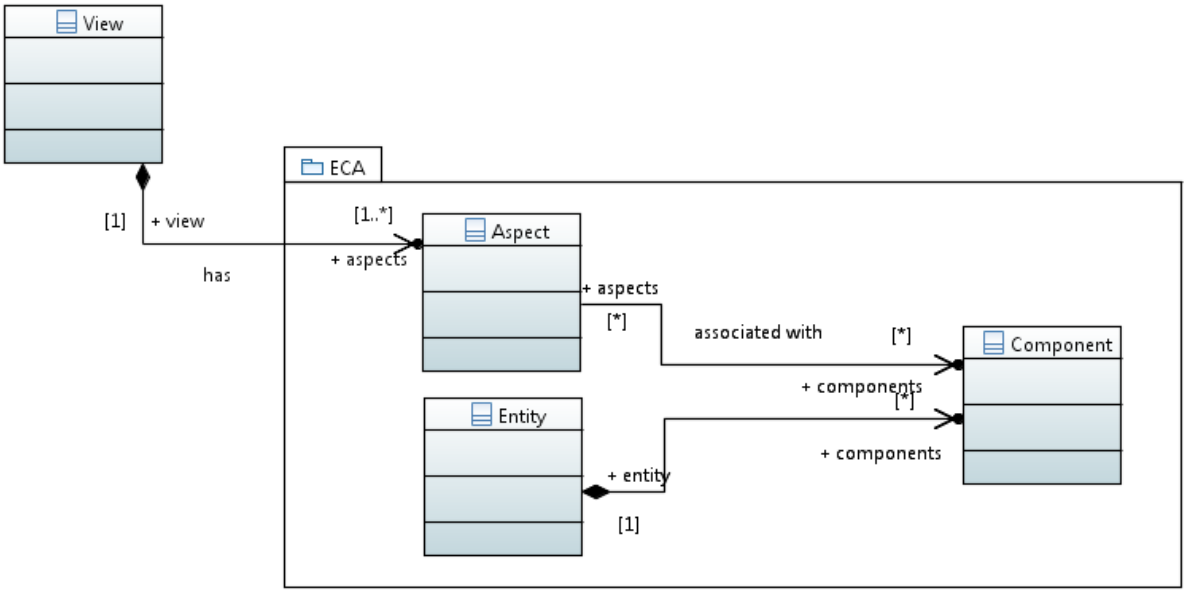

Abbildung 82: Architektur des Physikmodells des MTS

Nachfolgend werden die Begriffe und Konzepte des physikalischen Modells erläutert. Ein physikalisches Modell beschreibt eine logisch abgeschlossene Einheit, die die gesamten Gesetzmäßigkeiten für eine Schiffsentität definiert. Es wird so ermöglicht, dass Schiffe unterschiedliche Detailgrade ihrer Physik abbilden können. Es können so aber auch physikalische Modelle für unterschiedliche Schiffstypen (z.B. Tanker oder Containerschiffe) modelliert werden. Das Schiff kann sich nur innerhalb der durch das physikalische Modell definierten physikalischen Grenzen bewegen. Die Entity stellt dabei das Objekt in der realen Welt dar. Dabei kann es sich um ein statisches Objekt – z.B. Landmasse – oder ein dynamisches Objekt – z.B. ein Schiff – handeln. Die Definition der Entity findet durch seine Components statt. Components beschreiben die Attribute bzw. Komponenten, die Bestandteile einer Entity sind. Eine Schiffsentität hat z.B. eine Ruder-, Motor- und Rumpf-Komponente, die jeweils wiederum Attribute besitzen. Es ist auch möglich, dass eine einzelne Komponente die gesamte Entity beschreibt. Aspects beschreiben die physikalischen Gesetze, die die Attribute einer oder mehrerer Components entsprechend der betrachteten Einflüsse beeinflussen. Die betrachteten Einflüsse werden vom Aspect selbst definiert. Somit wird ermöglicht, einzelne Einflüsse auszulagern und für andere physikalische Modelle wiederzuverwenden. Die View beschreibt ein in sich geschlossenes physikalisches Modell. Durch die View können somit Entity-Component-Aspect-Kombinationen definiert und – dem Framework-Charakter folgend – für den Benutzer wiederverwendbar gestaltet werden.

Bei der Erweiterung der Physik sind die Klassen „View“ (Package „de.wi_ol.be.mts.ecs“), „ViewFactory“ (Package „de.wi_ol.be.mts.ecs“) und „Aspect“ (Package

„de.wi_ol.be.mts.aspects“) von Belang. Die View stellt ein in sich schlüssiges Physikmodell dar. Die ViewFactory überprüft, ob die Entity für die View ausgelegt ist und erstellt letztere. Die Aspects sind die einzelnen physikalischen Berechnungen, die auch zwischen den Views wiederverwendet werden können. Die View gibt an, in welcher Reihenfolge die Aspects ausgeführt werden sollen. Weiterhin definieren die Aspects, in welchen Intervallen die Aspects ausgeführt werden. Es muss darauf geachtet werden, dass die Entities lediglich bei einer ViewFactory als gültige View überprüft werden, da es sonst zu einer RuntimeException kommt. Um dies zu umgehen, kann bei gleichen Komponenten aber unterschiedlichen Views eine Markier-Komponente behilflich sein. In der EMod-Datei können die Views auch angegeben werden. Dazu wird der vollständige Klassenname der View in das Attribut „physicsBehavior“ gespeichert. In Abbildung 83 wird eine neue ViewFactory dargestellt. Die Methode „isApplicableFor“ prüft die Entities, ob dieses alle Components für die View besitzen. Die Methode „createView“ erstellt die View. Die Methode „getViewClass“ gibt die Klasse der View zurück. Die Methode „getMarkerComponents“ gibt alle Komponenten zurück, die als Markier-Komponenten benötigt werden.

```
public final class NewViewFactory implements ViewFactory {

	private static final long serialVersionUID = 1L;

	@Override
	public boolean isApplicableFor(final Entity entity) {
		return entity.hasComponent(VesselComponent.class)
				&& entity.hasComponent(NewMarkerComponent.class);
				&& entity.hasComponent(EnvironmentInfluence.class);
	}

	@Override
	public View createView(final Entity entity) {
		return new NewView(entity.get(VesselComponent.class));
	}

	@Override
	public Class<? extends View> getViewClass() {
		return NewView.class;
	}

	@Override
	public Set<Component> getMarkerComponents() {
		Set<Component> comps = new HashSet<Component>();
		comps.add(new EnvironmentInfluence());
		comps.add(new NewMarkerComponent());
		return comps;
	}
}
```

Abbildung 83: Erstellen einer neuen Klasse „ViewFactory“

Abbildung 84 zeigt die neue View-Klasse. Im Konstruktor werden die Aspects instanziiert, die für die View benötigt werden. Die Methode „getAspects“ liefert alle Aspekte in der Reihenfolge, in der sie ausgeführt werden müssen.

```
public final class NewView extends View {

	private static final long serialVersionUID = 1212944184599784962L;

	private final MovementAspect movementAspect;
	private final DraughtAspect draftAspect;
	private final RudderAspect rudderAspect;
	private final RotationAspect rotationAspect;
	private final AccelerationAspect accelerationAspect;
	private final NewSpecificAspect newSpecificAspect;


	public VesselSettingRateView(final VesselComponent vessel) {
		this.movementAspect = new MovementAspect(vessel);
		this.draftAspect = new DraughtAspect(vessel);
		this.rudderAspect = new RudderAspect(vessel);
		this.rotationAspect = new RotationAspect(vessel);
		this.accelerationAspect = new AccelerationAspect(vessel);
		this.newSpecificAspect = new NewSpecificAspect(vessel);
	}

	@Override
	public List<Aspect> getAspects() {
		List<Aspect> aspects = new ArrayList<>();
		aspects.add(this.draftAspect);
		aspects.add(this.accelerationAspect);
		aspects.add(this.newSpecificAspect);
		aspects.add(this.rudderAspect);
		aspects.add(this.rotationAspect);
		aspects.add(this.movementAspect);
		return aspects;
	}

}
```

Abbildung 84: Erstellen einer neuen Klasse „View“

3.3.1.1.5 Erweiterung der Kollisionserkennung

In der Konfigurationsdatei aus Abbildung 85 werden die Kollisionserkennungssysteme definiert.

```
<collisionSystems>
	<collisionSystem
value="de.wi_ol.be.mts.collisiondetection.LandCollisionDetection" priority="0"
enable="true" stopSimulation="false" />
	<collisionSystem
value="de.wi_ol.be.mts.collisiondetection.VesselCollisionDetection" priority="1"
enable="true" stopSimulation="false" />
	<collisionSystem
value="de.wi_ol.be.mts.collisiondetection.VesselInHabourDetection" priority="2"
enable="true" stopSimulation="false" />
	<collisionSystem
value="de.wi_ol.be.mts.collisiondetection.VesselGroundDetection" priority="3"
enable="true" stopSimulation="false" />
</collisionSystems>
```

Abbildung 85: Konfigurationsdatei Kollisionserkennung

Mit dem Attribut „value“ wird der vollständige Klassenname des Kollisionserkennungssystems angegeben. Das Attribut „priority“ gibt an, in welcher Reihenfolge die Systeme ausgeführt werden, wobei die höchste Priorität als erstes ausgeführt wird. Mit dem Attribut „stopSimulation“ kann angegeben werden, ob die Simulation bei einer Kollision beendet werden soll. Mit dem Attribut „enable“ kann das Kollisionserkennungssystem aktiviert bzw. deaktiviert werden.

Um ein neues Kollisionserkennungssystem zu implementieren, muss von der abstrakten Klasse „CollisionDetectionSystem“ (Package „de.wi_ol.be.mts.collisiondetection“) abgeleitet werden. Die neue Klasse muss genau den gleichen Konstruktor haben wie die abgeleitete Klasse, da hier über Reflection die Klasse instanziiert wird. Die übergebenen Entities werden, je nachdem welche Komponenten notwendig sind, entsprechend übergeben. Die notwendigen Komponenten werden über die Annotation „RequiredComponents“ (Package „de.wi_ol.be.mts.ecs.annotations“) definiert. Die übergebenen Entities beinhalten dann mindestens eine der definierten Komponenten. Weiterhin können über die Annotation „@ProvideMap“ das Kartenmaterial und über „@ProvideAgents“ die Agenten injiziert werden.

3.3.1.1.6 Erweiterung der Verkehrsregeln

In der Konfigurationsdatei aus Abbildung 86 werden die Verkehrsregeln definiert.

```
<trafficRegulation>
	<trafficRegulation
value="de.wi_ol.be.mts.trafficRegulation.RightHandTraffic" enable="true" />
	<trafficRegulation
value="de.wi_ol.be.mts.trafficRegulation.SecurityDistance" enable="true" />
</trafficRegulation>
```

Abbildung 86: Konfigurationsdatei Verkehrsregeln

Der Aufbau der Verkehrsregeln ähnelt dem der Kollisionserkennungssysteme. Über das Attribut „value“ wird der vollständige Klassenname der Verkehrsregel definiert. Das Attribut „enable“ besagt, ob die Verkehrsregel aktiviert oder deaktiviert ist. Für eine neue Verkehrsregel ist eine Ableitung der abstrakten Klasse „TrafficRegulation“ (Package „de.wi_ol.be.mts.trafficRegulation“) notwendig. Hier können dieselben Annotationen genutzt werden wie bei den Kollisionserkennungssystemen.

3.3.1.1.7 Erweiterung des Datenempfangs über HLA

Um den Datenempfang zu erweitern, muss die Klasse „World“ (Package „de.wi_ol.be.mts.ecs“) angepasst werden. Hier wird die Methode „exchangeStepAndUpdate“ erweitert. In Abbildung 87 wird exemplarisch der Empfang und die Weiterleitung der Daten für den NMEA Agent dargestellt.

```
else if (element instanceof NMEAMessage) {
	NMEAMessage m = (NMEAMessage) element;
	Agent agent = this.runtimeData.getAgents().get(m.getAgent().getId());
	if (agent instanceof InternalNMEAAgent) {
		((InternalNMEAAgent) agent).receiveNMEAMessage(m);
	}
}
```

Abbildung 87: Erweiterung des Datenempfangs über HLA um NMEA Nachrichten

3.3.1.2 Visualisierung

3.3.1.2.1 Erweiterung der Standard-GUI-Komponenten von MASON

Die Visualisierung besteht zum einen aus den Standard-Komponenten des MASON-Frameworks und zum anderen aus selbst entwickelten Bestandteilen, welche auf Basis der AWT- und Swing-Bibliotheken entwickelt wurden. Alle Klassen der grafischen Benutzeroberfläche sind im Package „de.wi_ol.be.mts.de“ im Projekt „Simulation“ zu finden. Ausgangspunkt für Anpassungen oder Erweiterung der grafischen Benutzeroberfläche des MTS ist die Klasse "WorldWithUI", welche von der abstrakten Klasse „GUIState“ aus dem MASON-Framework abgeleitet ist. Dieses Vorgehen ermöglicht die Nutzung der grafischen Komponenten des Frameworks. Im Folgenden wird auf wichtigsten Bestandteile der Klasse “WorldWithUI“ eingegangen, um einen Überblick über die Implementierung der Benutzeroberfläche der MTS zu gewinnen. Für nähere Erläuterungen rund um das MASON-Framework wird auf die Dokumentation des Frameworks verwiesen [Ge14].

Wie bereits erläutert, ist die Klasse „WorldWithUI“ der Ausgangspunkt aller Benutzeroberflächen-bezogenen Komponenten der MTS. Wichtigste Methode dieser Klasse ist die „init“-Methode, welche unter anderem für die Initialisierung der Display-Komponente zuständig ist. Eine Hauptaufgabe ist das Hinzufügen der Layer (Karteninformationen) zum Display. Ein Layer wird durch ein Portrayal in MASON repräsentiert. Für die in der MTS implementierten Layer wurde jeweils eine Klasse erstellt, in der die Eigenschaften des jeweils zu zeichnenden Layer definiert werden. Eigenschaften sind z.B. die Größe des Objektes, die Farbe oder eine Bilddatei.

Neben der Initialisierung der Layer und des Displays in der „init“-Methode, wird als weiterer wichtiger Bestandteil die Klasse „MTSConsole“ initialisiert, welche von der MASON-Standardkonsole abgeleitet ist. Dieses Vorgehen ermöglicht die Anpassung der grafischen Darstellung der Konsole.

Zudem wurden neben der Anpassung der Standard-GUI-Elemente von MASON neue GUI-Elemente implementiert, indem von den Standard-Elementen abgeleitet wurde. Auch hier ist der Ausgangspunkt die Klasse "WorldWithUI". An dieser Stelle werden die grafischen MASON-Komponenten in einen JFrame mit JMenuBar eingefügt. Diese Komponente wird in der Klasse "MainFrame" erstellt und an die Klasse "WorldWithUI" übergeben. In der Klasse "WorldWithUI" wird des Weiteren der „Vessel Properties“-Reiter und das Status-Logging-Fenster hinzugefügt. Der „Vessel Properties“-Reiter wird in der Klasse "VesselPropertyTabItem" erstellt. Hier werden die stetig aktualisierten Eigenschaften des auf der Karte ausgewählten Schiffes aus der Klasse "VesselPropertyPortrayal" für die Darstellung in der GUI aufbereitet.

Alle zusätzlich erstellten Fenster werden durch die Komponente "JMenuBar" im MainFrame aufgerufen. Hier ist im Bereich des Menüpunkts „File“ die Möglichkeit gegeben, ein ganzes Szenario zu speichern oder zu öffnen. Diese beiden Fenster werden durch die Standard-Swing-Komponente "JFileChooser" erstellt und sind in der Klasse "SaveOpenFrame" implementiert. Im Bereich des Menüpunkts „Edit“ kann das Fenster „Scenario editor“ geöffnet werden. Dieser ist in der Klasse "SetupVesselFrame" zu finden. In diesem Fenster wird ebenfalls der JFileChooser benutzt, um EMod-Dateien zu laden oder zu speichern. Des Weiteren lässt sich hier das Fenster "Create EMod-file" öffnen. Der daraufhin sichtbare JFrame wird in der Klasse "EditVesseFrame" erstellt. Er besteht hauptsächlich aus einer JTable, in der neu erstellte Schiffe durch das Befüllen der JTextField-Komponenten hinzugefügt werden.

3.3.1.2.2 Erweiterung des Statistikbereichs

Im Bereich des Menüpunkts „Tools“ im Hauptfenster wird der Statistikbereich aufgerufen und das Status-logging aktiviert bzw. deaktiviert. Das Statistikfenster wird in der Klasse "StatisticFrame" erstellt. Hier werden mit Hilfe des Frameworks „JFreeChart“ Statistiken für bestimmte Eigenschaften der simulierten Schiffe visualisiert. Hauptelement ist hier die Komponente „JFreeChart“, welche das sichtbare Koordinatensystem repräsentiert. Der JFreeChart besteht aus den Komponenten „XYPlot“, die für die Eigenschaften des Koordinatensystems zuständig ist, und „XYLineAndShapeRenderer“, die die Eigenschaften des zu zeichnenden Graphen enthält. Eine dritte Komponente ist „SeriesCollection“, welche eine Serie hinzugefügt. Eine solche Serie repräsentiert im Falle der MTS ein Schiff mit der zu zeichnenden Eigenschaft. Die SeriesCollection erkennt nach dem Hinzufügen oder Löschen einer Serie automatisch, dass sich der Zustand verändert.

Alle in Kapitel 3.3.1.2.1 und 3.3.1.2.2 erläuterten GUI-Elemente lassen sich entweder durch Ableiten oder direktes Verändern in den jeweiligen Klassen anpassen. Zudem kann in den beschriebenen Bereichen die Benutzeroberfläche um weitere Komponente erweitert werden.

3.3.2 Umweltsimulator

3.3.2.1 Hinzufügen von Umweltaspekten

Der Umweltsimulator kann um beliebige Umwelt-Faktoren ergänzt werden. Voraussetzung dafür ist, dass diese Faktoren samt Attributen in den Szenarien-Datenbanken des Umweltsimulators vorliegen. Zudem muss die Simulation die neuen Umwelt-Faktoren berücksichtigen können. Der generelle Vorgang zur Implementierung eines neuen Umwelt-Faktors gliedert sich in die drei nachfolgend beschriebenen Schritte:

Schritt 1: Umweltsimulator erweitern

Der erste Schritt ist die Erweiterung des Umweltsimulators. Umweltobjekte werden als HLA-Objekte im Umweltsimulator instanziiert. Die Eigenschaften der Umweltobjekte stammen in der Regel aus einem zuvor erstellten Szenario. Damit das Szenario neue Umwelt-Faktoren erkennt, müssen diese in die Datenbankstruktur integriert werden. Hierzu könnte z.B. eine weitere Tabelle mit den neuen Umweltobjekten angelegt werden. Je nachdem welches Szenario mit diesem neuen Umwelt-Faktor verwendet werden soll, sollte die Benutzeroberfläche des entsprechenden Szenarios angepasst werden. Beispielsweise kann hier die Eingabe von Luftdruck im parametrisierbaren Umweltszenario durch das Einfügen weiterer Textfelder realisiert werden.

Schritt 2: Simulation erweitern

Damit die Simulation die über HLA empfangenen Objekte verarbeiten kann, ist es notwendig, die Simulation für diese Umweltobjekte vorzubereiten. Hierzu müssen neue Aspects definiert (Package „de.wi_ol.be.mts.aspects“) und die Environment-Komponente in der Simulation erweitert werden. Der Umweltsimulator ist dafür zuständig, zu jeder Zeit der verteilten Simulation die Umwelteinflüsse, die vom Umweltsimulator bereitgestellt werden, zu sammeln und im Speicher zu halten. Damit die vorhandenen Umwelteinflüsse auch eine Auswirkung haben, werden die Aspects benötigt. Diese beschreiben, wie und mit welchen Daten sich ein Umwelt-Faktor auf ein Schiff auswirkt. Das physikalische Modell jedes Umwelteinflusses wird als Aspect implementiert. Eine Ausnahme bildet der Wasserstand, weil sich dieser nicht unmittelbar auf die Schiffsphysik auswirkt. Der Wasserstand wird in der Klasse "VesselGroundDetection" (Package „de.wi_ol.be.mts.collisiondetection“) verarbeitet. Die Klasse überprüft, ob das Schiff wegen des Wasserstands auf Grund läuft.

Schritt 3: HLA und HAGGIS-Datenmodell erweitern

Damit die Umweltdaten vom Umweltsimulator in die MTS übertragen werden, müssen sie über die CERTI-Laufzeitinfrastruktur kommuniziert werden. Hier kommt hinzu, dass in der verteilten Simulation über HLA nur Objekte übertragen werden dürfen, die allen Federates bekannt sind. Kurzum werden nur Objekte, die im HAGGIS-Datenmodell festgelegt wurden, anerkannt. Um nun neue Umwelteinflüsse zu definieren, ist es notwendig, diese als Daten im HAGGIS-Datenmodell hinzuzufügen. Dabei ist zu beachten, dass alle Attribute, die die neuen Umwelt-Faktoren haben, hinzugefügt werden müssen. Bereits bestehende Konzepte und Datentypen des HAGGIS-Datenmodells sollten wiederverwendet werden.

Beim Hinzufügen eines Umwelt-Faktors Luftdruck z.B. ist es erforderlich, folgende Attribute hinzuzufügen, damit dieser mit dem Umweltsimulator funktioniert:

(1) Attribut „Pressure“ als Datentyp „Double;
(2) Attribut „GPS“ als Datentyp „geoPos“;

Alle Werte sind auch hier bereits existierende Daten des HAGGIS-Datenmodells, wobei geoPos eine Position aus Längen- und Breitengrad darstellt. Der Datentyp "geoPos" ist an anderer Stelle im HAGGIS-Datenmodell hinterlegt und daher ist ein Verweis möglich.

3.3.2.2 Umweltdatenbanken

Das statische und das dynamische Umweltszenario bestehen jeweils aus zwei Datenbanken. Eine Datenbank beinhaltet die Strömungs-, Wind- und Wellendaten. Die zweite Datenbank beinhaltet die Tideninformationen. Das Datenbankschema ist in Abbildung 88 abgebildet. Der Aufbau der Datenbanktabellen ist abgeleitet von den zur Verfügung stehenden Datenquellen. Die Strömungstabelle "current_final" ist angelehnt an die Strömungsdaten des BSH. Eine Zeile des CSV-Dumps eines GRIB-Records sieht z.B. wie folgt aus:

```
"2014-02-02 00:00:00", "2014-02-02 00:30:00", "UOGRD", "mean sea level", 10.1319, 53.4208, -0.102
```

Das dazugehörige Schema: Startdatum, Enddatum, U-/V-Komponente (U-Komponente = UOGRD, V-Komponente = VOGRD), Informationen zur Wassertiefe der Strömung (mean sea level = mittlerer Meeresspiegel), Breitengrad, Längengrad, Wert

Für den Umweltsimulator wird der erste Zeitstempel, der Längen- und Breitengrad sowie der gemessene Wert am Ende der Zeile benötigt.

Die Wind- und Wellentabelle "windwave_final" ist angelehnt an die uns zur Verfügung gestellten Wetterdaten der drei FINO-Forschungsplattformen in der Nord- und Ostsee [Fo14]. Da es für die Wind- und Wellendaten jedoch nur die drei FINO-Messstationen gibt, sind diese Daten weit weniger umfangreich als die Strömungsdaten des BSH.

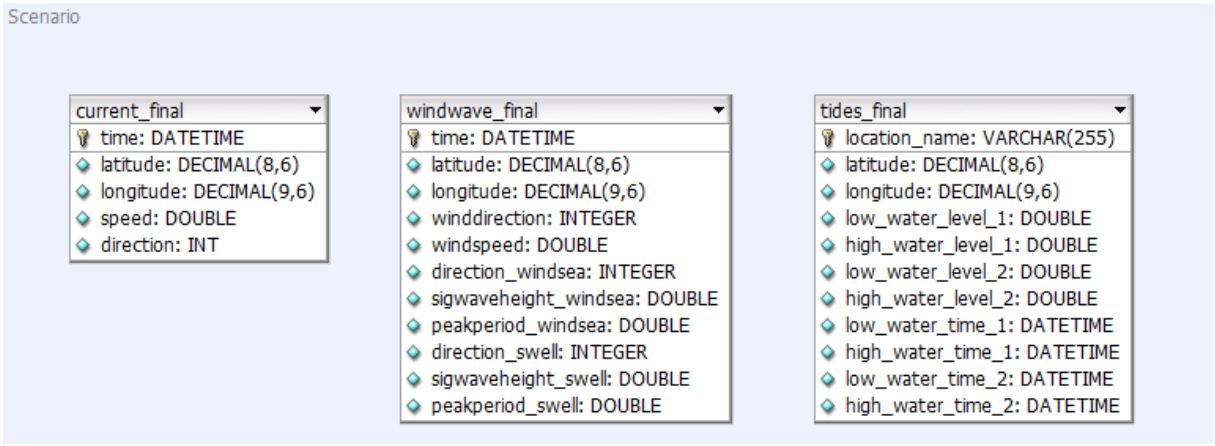

Abbildung 88: Datenbankschema des Umweltsimulators

Die Datenbanken des statischen und dynamischen Szenarios unterscheiden sich in erster Linie dadurch, dass bei der ersterem nur ein Zeitstempel mit den jeweiligen Umweltdaten existiert. So wird aus jeder Tabellenzeile das jeweilige HLA-Umweltobjekt gebildet und zum Simulationsbeginn an die MTS gesendet. Ein HLA-Umweltobjekt ist ein Java-Objekt, welches

über die CERTI-Laufzeitinfrastruktur versendet werden kann. Beim dynamischen Szenario werden in Abhängigkeit zur vergangenen Simulationszeit neue HLA-Umweltobjekte aus der Datenbank erstellt, sofern deren Zeitstempel mit der aktuellen Simulationszeit übereinstimmen. So entsteht eine dynamische bzw. kontinuierliche Versorgung der MTS mit sich verändernden Umweltdaten.

Die BSH-Strömungsdatendateien enthalten immer ganztätige Aufzeichnungen. Dieses Prinzip wurde für das dynamische Szenario übernommen. Es sind damit 24 Stunden dauernde Szenarien möglich, die sich nach Überschreiten der 24 Stunden wiederholen. Dasselbe Prinzip wird auch für die Wind- und Wellendaten verwendet. Die Zeitstempel der Strömungsdaten sowie Wind- und Wellendaten können dabei unterschiedlich sein. So können die Strömungsdaten beispielsweise stündlich getaktet sein, die Wind- und Wellendaten hingegen nur dreistündlich.

Die Tidendatenbanktabelle ist anders aufgebaut als die bisher beschriebenen Tabellen. Sie enthält aber ebenfalls Daten, die 24 Stunden umfassen. Da die Gezeiten sich über den Tagesverlauf wiederholen und es im Regelfall pro Tag an einem Ort jeweils zwei Mal Niedrig- und Hochwasser gibt, besteht die Gezeitentabelle jeweils aus zwei Niedrig- und zwei Hochwasserständen mit den zugehörigen Zeitpunkten. Um eine höchstmögliche Dynamik in den Gezeitendaten zu ermöglichen, kann für jeden beliebigen Tageszeitpunkt ein Wasserstand aus den Messdaten interpoliert werden. Hierzu wird die Zeitspanne zwischen Niedrig- und Hochwasser bzw. Hoch- und Niedrigwasser, in die der aktuelle Simulationszeitpunkt fällt, gemessen und der aktuelle Wasserstand linear interpoliert. Befindet sich die Simulation beispielsweise zeitlich genau zwischen Hoch- und Niedrigwasser, wird der Mittelwert zwischen Hoch- und Niedrigwasser gewählt. Um die Simulation nicht in jedem Simulationsschritt mit neuen Gezeitendaten zu überladen, werden diese nur dann an die Simulation gesendet, wenn neue Strömungs- oder Wind- und Wellendaten an die Simulation gesendet werden.

Beim statischen Szenario kann der Benutzer vor dem Simulationsstart einen Prozentsatz definieren. Dieser Prozentsatz bestimmt die Höhe des Gezeitenstandes zwischen dem erstem Niedrig- und Hochwasser aus der Gezeitendatenbank. Null Prozent entspricht genau dem ersten Niedrigwasser, hundert Prozent entspricht genau dem ersten Hochwasser. Jeder zwischen null und hundert liegende Prozentsatz ist wählbar. Durch dieses Verfahren lässt sich die exakt selbe Gezeitendatenbank sowohl für statische als auch für dynamische Szenarien verwenden.

Bisher wurde eine umfangreiche Gezeitendatenbank für die verteilte Simulation händisch aus Daten der Webseite des BSH des deutschen Seeraumes erstellt [Bu14a]. Es werden seitens des BSH leider keine Gezeitendaten auf dem FTP-Server zur Verfügung gestellt, wie dies bei den Strömungsdaten der Fall ist. Daher ist das Erstellen der Gezeitendatenbank vergleichsweise unkomfortabel.

Durch die zuvor beschriebene Struktur der Szenarien-Datenbanken ist es möglich, eigene Szenarien auch aus beliebigen anderen Quellen zu erstellen. Für die Bearbeitung oder Erstellung der Datenbankdateien ist lediglich eine Software notwendig, die SQLite-Datenbanken verarbeiten kann. Als Empfehlung wird für diesen Zweck die Software "SQLite Expert (Personal)" [Co14] empfohlen, da sie kostenfrei ist und eine übersichtliche Benutzeroberfläche bietet.

Soll ein vollständig neues Szenario entwickelt werden, ist in jedem Fall ein HLA Federate erforderlich, um die Umweltdaten in Form von HLA-Objekten an die verteilte Simulation zu schicken. Ansonsten existieren keine weiteren Einschränkungen für die Entwicklung neuer

Szenarien. Entsprechend ist der HLA Federate die einzig wiederkehrende Technologie in allen vier Umweltszenario-Typen des Umweltsimulators.

3.3.3 Analyse-Komponente

Die Analyse-Komponente ist entsprechend ihres Einsatzzweckes so entworfen worden, dass sie einfach erweitert und konfiguriert werden kann.

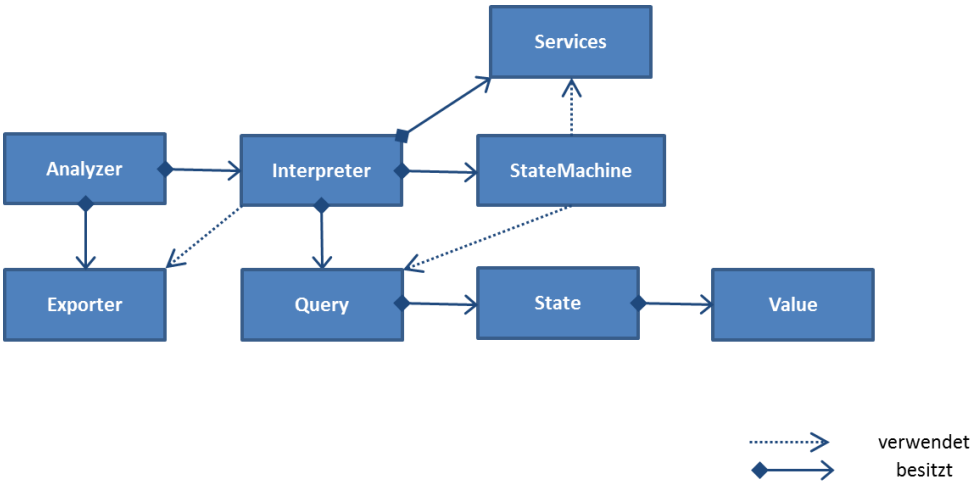

Abbildung 89: Zusammenspiel der Erweiterungsmöglichkeiten

In Abbildung 89 sind zur Wiederholung nochmal die Beziehungen wichtiger Konzepte der Analyse-Komponente dargestellt.

Query

Um ein neues Query zu erstellen und zu verwenden, muss eine Klasse vom Interface „QueryFactory“ abgeleitet werden. Zusätzlich muss diese noch in die Datei „config.json“ im Wurzelverzeichnis des Projekts eingetragen werden. Die Aufgabe der QueryFactory ist es, zur Laufzeit ein Query-Objekt bereitzustellen. Damit sie instanziiert werden kann, muss sie einen parameterlosen Konstruktor besitzen.

Für die Erstellung eines Query-Objektes wurde darauf geachtet, eine sehr einfache Programmierschnittstelle (API) bereitzustellen (DSL, Domain-specific Language). Wie diese verwendet wird, ergibt sich aus den Tests innerhalb des Java-Projekts. Zudem ist in Abbildung 92 ein Beispiel für die Umsetzung eines Querys dargestellt.

Value

Um in Querys neue Eigenschaften und Berechnungen verwenden zu können, wie z.B. die TEU-Berechnung, muss eine Klasse erstellt werden, die von Value erbt. Ein Value hat immer einen definierten Rückgabetyp in Form eines EClassifiers. Zudem muss die abstrakte Methode „Object createFor(Element element, Services services)“ auskommentiert werden. Diese ist für die Berechnung des jeweiligen Wertes verantwortlich. Der Parameter „element“ verweist auf die aktuell vom Query betrachtete Instanz. Der Parameter „services“ stellt

weitere globale Funktionalitäten bereit, wie z.B. den Zugriff auf geografische Operationen, über die der aktuelle Hafen ermittelt werden kann.

Für die Verwendung von Werten, die über das HAGGIS-Datenmodell bereits definiert sind, z.B. über die Methode „Vessel.getId“, wird der Entwickler von der API unterstützt. Hierzu sei wieder auf die Beispiele im Projekt verwiesen.

Service

Wenn für eine Berechnung Element-übergreifende Informationen benötigt werden, muss ein entsprechender Service hierfür definiert und in der Datei „config,json“ registriert werden. Ein Service wird über die Änderung jedes bekannten Objektes informiert. Als Beispiel sei ein geografischer Index genannt, um Distanzberechnungen zu ermöglichen.

Exporter

Um einen neuen Exporter zu definieren, muss eine Klasse das Interface „Exporter“ implementieren und dies in der Konfiguration vermerkt werden. Der Exporter wird über die Tupel-Struktur jedes Querys einmalig informiert und erhält dann jedes Tupel, das vom TupleManager generiert wird.

Beispiel

Um eine Auswertung zu erhalten, die alle Schiffe ermittelt, die sich in einer Gefahrensituation befunden haben, muss wie folgt vorgegangen werden:

- Zunächst muss eine Klasse „HazardService“ erstellt werden, die die Gefahrenmeldungen verwaltet und die Zuordnung zu einem Schiff vornehmen kann (vgl. Abbildung 90).

```
public final class HazardService extends Service {

    @Override
    public void updateObject(final Element element) {
        if (element instanceof Hazard) {
            Hazard hazard = (Hazard) element;
            this.memorize(hazard);
        }
    }

    public Hazard getHazardFor(final Element element) {
        return this.rememberForHazards(element);
    }
    ...
}
```

Abbildung 90: Erweiterung Analyse-Komponente – Klasse „HazardService“

- Anschließend ist die Definition eines HazardValue notwendig, der über den HazardService die aktuellen Gefahrensituationen für das jeweilige Schiff ausliest und jeweils eine Gefahrensituation entnimmt (vgl. Abbildung 91).

```
public final class HazardValue extends Value {

    protected HazardValue() {
        super(RiskPackage.eINSTANCE.getHazard());
    }

    @Override
    public Object createFor(final Element element, final Services services) {
        HazardService hazardService = services.get(HazardService.class);
        Hazard hazard = hazardService.getHazardFor(element);
        return hazard;
    }
    ...
}
```

Abbildung 91: Erweiterung Analyse-Komponente – Klasse „HazardValue“

- Ein Query muss erstellt werden, der den HazardValue überprüft und, wenn vorhanden, Schiffs-ID und Hazard in ein Tupel schreibt. Ein Query wird mehrfach pro Objekt ausgeführt (vgl. Abbildung 92).

```
public final class HazardQueryFactory implements QueryFactory {

    @Override
    public Query create() {
        Builder b = DSL.query("hazards");
        Vessel vessel = b.from(Vessel.class);
        b.select(vessel.getId(), "id");
        Value hazard = new HazardValue();
        b.select(hazard, "hazard");
        Condition hasHazard = new IsNotNullCondition(hazard);
        b.where(hasHazard);
        b.setRepeat();
        return b.generate();
    }
```

Abbildung 92: Erweiterung Analyse-Komponente – Klasse „HazardQueryFactory“

- Ein neuer Exporter muss definiert werden, der die Tupel zur gewünschten Ausgabe aggregiert (vgl. Abbildung 93).

```
public final class HazardExporter implements Exporter {

    @Override
    public void handleSchema(final String name, final Schema schema) {
        // nothing to do here
    }

    @Override
    public void handleTuple(final String name, final Tuple tuple) {
        if (name.equals("hazards")) {
            int id = (Integer) tuple.getValue(0);
            Hazard hazard = (Hazard) tuple.getValue(1);
            this.process(id, hazard);
        }

    }
    ...
}
```

Abbildung 93: Erweiterung Analyse-Komponente – Klasse „HazardExporter“

- Zum Schluss müssen das Query und der Exporter noch in der Konfiguration dargestellt werden.

4. Evaluierung

Dieses Kapitel befasst sich mit allen evaluierungsbezogenen Themen der verteilten Simulation. Zunächst wird das allgemeine Vorgehen zu den Testing-Verfahren beschrieben, anschließend werden die durchgeführten Tests im Einzelnen erläutert. Dazu zählen Unit-Test, Integrationstest, Funktionstest der Analyse-Komponente und ein Test der MTS auf Realitätsnähe durch den Vergleich mit AIS-Daten.

4.1 Allgemeines Test-Vorgehen

Mit dem Testen einer Software soll u.a. überprüft werden, ob diese der vorgegeben Softwarespezifikation entspricht und die zuvor im Pflichtenheft beschriebenen Funktionen auch ordnungsgemäß sowie frei von Fehlern ablaufen [No07, S. 172ff] [Ak06, S. 374]. Fehler werden dabei als „jede Abweichung der tatsächlichen Ausprägung eines Qualitätsmerkmals von der vorgesehenen Soll-Ausprägung, jede Inkonsistenz zwischen der Spezifikation und der Implementierung und jedes strukturelle Merkmal des Programmtextes, das ein fehlerhaftes Verhalten des Programms verursacht“ [Ba99, S. 502] verstanden. Aus diesem Grund wurden durchgehend Tests parallel zur Konzeption, Spezifikation und Entwicklung der MTS durchgeführt [An03, S. 295]. Im Folgenden werden die durchgeführten Testarten erläutert sowie beschrieben.

4.2 Unit-Tests

Unit-Tests sind allgemein als ein wichtiges Mittel für die Verbesserung von Software anerkannt, wobei die einzelnen Methoden eines Moduls parallel zur Entwicklung getestet werden [TdX10, S. 483] [An03, S. 295f]. Dabei stellt ein Modul die kleinstmögliche zu testende Einheit eines Programms dar, bei dem ein erfolgreicher Test grün und ein nicht gelungener Test rot markiert wird [WA11, S. 159]. Es wurden Unit-Tests mit JUnit für alle Komponenten der Projekte der PG MAPS durchgeführt.

Im Folgenden wird die Vorgehensweise für die Erstellung der durchgeführten Unit-Tests beschrieben:

(1) Für jede zu testende Klasse wird eine Testklasse erstellt.
(2) Für jede zu testende Methode innerhalb der zu überprüfenden Klasse wird eine Testmethode erstellt. Mit der Testmethode wird eine korrekte Funktionsweise der zu testenden Methode überprüft.
(3) Test-Verifikation: JUnit Test Grün → erfolgreich oder JUnit Test Rot → nicht erfolgreich.

4.3 Integrationstests

Integrationstests ermöglichen es, Fehler zu identifizieren, die aufgrund des Zusammenwirkens verschiedener Softwaremodule auftreten können. Bei dem Zusammenwirken verschiedener Komponenten bei verteilten Simulationen muss auf eine fehlerfreie Interaktaktion dieser unterschiedlichen Komponenten geachtet werden. Selbst für den Fall, dass sich die einzelnen Komponenten korrekt verhalten, können bei deren Interaktion fehlerhafte Resultate entstehen [PRW13, S. 11]. Ziel des Integrationstests ist es sicherzustellen, dass bei der Interaktion zwischen verschiedenen Komponenten (z.B. zwischen der MTS und dem Umweltsimulator) keine Fehler entstehen [Si03, S. 298].

Es wurden umfangreiche Testfälle in einem Testdrehbuch erarbeitet, um mit diesen die Korrektheit der MTS mit seinen verschiedenen Komponenten zu überprüfen und eine hohe Softwarequalität garantieren zu können [Sa12, S. 405]. Um die MTS auf eine korrekte Funktionsweise hin zu prüfen, ist ein Testdrehbuch mit vordefinierten Ergebnissen aus folgenden Gründen sinnvoll: (1) es wird dadurch sichergestellt, dass die Funktionen der MTS und ihre Komponenten ordnungsgemäß ablaufen und wie erwartet bzw. beschrieben funktionieren; (2) somit können Fehler innerhalb der MTS festgestellt werden, die sonst evtl. zu spät oder gar nicht identifiziert worden wären; (3) außerdem ermöglichen Testfälle, die MTS und ihre Komponenten bei Veränderungen auf vorher nicht aufgetretene Fehler bzw. eine korrekte Funktionsweise hin zu kontrollieren. Daher wurden auch zu jeder neuen Funktion die dazugehörigen Testfälle erstellt. Es ist zu beachten, dass Standardfunktionen von MASON nicht getestet wurden, da diese außerhalb des Einflussbereichs der MTS liegen.

Im Folgenden werden der Aufbau und die Vorgehensweise der konzipierten Testfälle innerhalb des Testdrehbuchs beschrieben. Zusammengehörige Testfälle (bestimmte Funktionalitäten, Testarten, Testsysteme etc.) werden in Testgruppen gegliedert. Jeder Testfall besteht aus den nachfolgenden Komponenten und wird durchgehend nach diesem Schema erstellt [Cl10, S. 216ff] [No07, S. 170ff] [Si03, S. 308]:

- **Testfall ID:** Nummer (z.B. 1-00) zur eindeutigen Identifikation des jeweiligen Testfalls.
- **Testname:** Name des jeweiligen Testfalls.
- **Beschreibung:** Beschreibt, was getestet wird bzw. die Funktion des jeweiligen Testfalls.
- **Testschritte:** Formulierung der notwendigen, sukzessiv aufeinander aufbauenden Schritte, um den jeweiligen Testfall durchführen zu können.
- **Erwartetes Ergebnis:** Detaillierte Beschreibung des zum jeweiligen Testfall erwarteten Ergebnisses.
- **Testergebnis:** Grün (vollständige Funktionalität), gelb (teilweise Funktionalität) und rot (fehlerhafte Funktionalität).
- **Kommentar:** Falls notwendig eine genauere Erläuterung oder Anmerkung zum jeweiligen Testfall.

Für jede Funktion der MTS wurde ein Testfall erstellt, welcher aus den zuvor beschriebenen Bestandteilen (Testfall ID, Testname, Beschreibung, Testschritte, Erwartetes Ergebnis, Testergebnis und Kommentar) besteht. Die jeweilige Funktion wurde anschließend ausgeführt und die Testergebnisse in einer MS Excel-Datei protokolliert. MS Excel ist dafür besonders geeignet, weil es die Ergänzung und Überprüfung der Testfälle in einer praktischen Art und Weise ermöglicht (in Hinblick auf Übersichtlichkeit und Erweiterbarkeit der Testfälle) [Si03, S. 309]. Getestet wurden MTS und die dazugehörigen Komponenten, d.h. die Simulation, die Visualisierung, der Umweltsimulator und die Analyse-Komponente.

4.4 Funktionstest Analyse-Komponente

Um zu testen, ob die Analyse-Komponente funktioniert, muss ein Query verwendet werden, der für alle Objekte die ID ermittelt. Zudem muss der CSVExporter konfiguriert werden. Nach der Durchführung eines Simulationslaufs sollten sich in der CSV-Datei alle IDs von Objekten befinden, die per HLA an die Analyse-Komponente kommuniziert wurden.

4.5 Test auf Realitätsnähe

4.5.1 Zielsetzung

Ziel dieses Konzepts ist es, durch weitestgehend automatisierte Verfahren Aussagen über die Genauigkeit der Schiffsdynamik der MTS auf makroskopischer Ebene zu liefern. Weiterhin sollen die Verfahren dem Zweck dienen, spätere Anpassungen der Schiffsdynamik dahingehend zu prüfen, in welchem Maß sie die verfolgten Ziele (meist Genauigkeitsverbesserungen) erreichen. Wichtig hierbei ist die Abgrenzung der Schiffsdynamik von der Logik bzw. Intelligenz der unterschiedlichen Agenten.

Die Schiffsdynamik beschreibt ausschließlich das physikalische Verhalten der simulierten Schiffe in Gewässern. Berücksichtigt werden hierbei neben den vom Schiff selbst erzeugten Kräften (durch Motorleistung und Ruder) auch die auf das Schiff einwirkenden Umwelteinflüsse (wie Strömung und Wind).

Die Logik der Agenten befasst sich mit der Steuerung der in der MTS simulierten Schiffe unter Berücksichtigung der Schiffsdynamik. Sie stellt daher je nach Anwendungsfall die Steuerungs-Komponente der einzelnen Schiffe dar. Diese Anwendungsfälle unterscheiden sich hinsichtlich ihrer Anforderungen deutlich, wodurch die einzelnen Agentenlogiken von sehr einfach bis komplex gefächert sind. Besonders hervorzuheben ist hierbei, dass die Logik der Agenten – analog zur realen Welt – keinen Einfluss auf die Schiffsdynamik (die physikalischen Kräfte) hat. Vielmehr ist es ihre Aufgabe, sich der Schiffsdynamik (ggf. auch dem für das jeweilige Schiff geltenden physikalischen Modell[3]) zu beugen und sich den Gegebenheiten anzupassen. Da die Logik der Agenten (je nach Ausprägung) den gefahrenen Weg der einzelnen Schiffe stark beeinflusst, gilt der Wahl der für die Genauigkeitsmessung verwendeten Agentenlogik besondere Aufmerksamkeit, um unerwünschte Seiteneffekte zu vermeiden.

4.5.2 Anforderungen

Die Genauigkeitsmessungen haben den Anspruch, möglichst weit normiert zu werden, um eine gute Vergleichbarkeit zu erhalten. Außerdem muss die Genauigkeitsmessung mit den vorhanden Mitteln und insgesamt angemessenem Aufwand umsetzbar sein. Vorhandene Mittel sind: (1) sämtliche Parameter der Simulation (inkl. Logik der Agenten), (2) AIS-Daten (Daten, u.a. zur Identifikation und Position von Schiffen) der MarineTraffic-Webseite [Ma14].

4.5.3 Konzept

Zu Beginn wird der Routenverlauf der zu betrachtenden Schiffe im CSV-Format benötigt. Als mögliche Quelle hierfür dient die MarineTraffic-Webseite. Bei der Wahl eines geeigneten Routenverlaufs muss besonders auf die Sendefrequenz der AIS-Signale geachtet werden. Ein Abstand von wenigen Minuten ist für die Evaluierung optimal, da für die Zeit zwischen den Signalen keine Informationen (wie Geschwindigkeits- oder Kursänderungen) ermittelt werden können. Mit geringerer Sendefrequenz verringert sich gleichzeitig die Aussagekraft der durchgeführten Vergleiche.

Anschließend werden die betrachteten Schiffe innerhalb der MTS parametrisiert abgebildet und es wird ihren Agenten der Auftrag erteilt, die entsprechenden Routen nach ihrem realen Vorbild abzufahren.

Im Anschluss an den Simulationslauf wird die benötigte Zeit für jedes Schiff gemessen und mit der des realen Schiffes verglichen. Berücksichtigt werden muss hierbei, dass die AIS-Signale keine Informationen über die zur Zeit der Messung aktuellen Umwelteinflüsse enthalten. Die Simulation verläuft daher immer ohne die Berücksichtigung von Umwelteinflüssen. Vereinzelt starke Abweichungen in den Ergebnissen können daher ggf. auf sehr starke Umwelteinflüsse während der Messungen zurückgehen.

[3] Im Rahmen der PG ist für die Simulation der Schiffe bisher nur ein physikalisches Modell vorgesehen. Die Architektur der MTS erlaubt es jedoch, verschiedenen Schiffstypen in einer Simulation verschiedene physikalische Modelle zuzuordnen.

4.5.4 Vorgehen

Nachfolgend werden die notwendigen Schritte beschrieben, um die Genauigkeitsmessung durchzuführen:

1. Schritt

Der Routenverlauf der zu betrachtenden Schiffe wird, wie oben beschrieben, im CSV-Format benötigt. Die CSV-Datei besitzt die in Kapitel 2.4.4 beschriebenen Parameter. Über die MarineTraffic-Webseite kann über die Angabe der MMSI (Maritime Mobile Service Identity, der Rufnummer des mobilen Seefunkdienstes) das zu betrachtende Schiff gefunden werden. Hierzu muss in dem dafür bereitgestellten Suchfeld die entsprechende MMSI eingegeben werden (s. Abbildung 94).

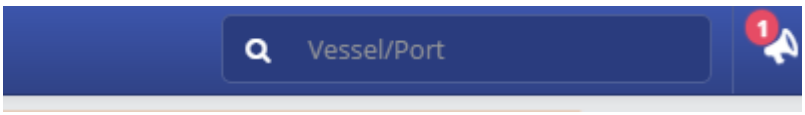

Abbildung 94: MarineTraffic-Webseite – Suchen anhand der MMSI

Anschließend wird die Detailseite zum Schiff mit allen weiteren verfügbaren Informationen angezeigt. Hier kann über den Aufruf des Links „Latest Positions“ der aktuelle Routenverlauf angezeigt werden (s. Abbildung 95).

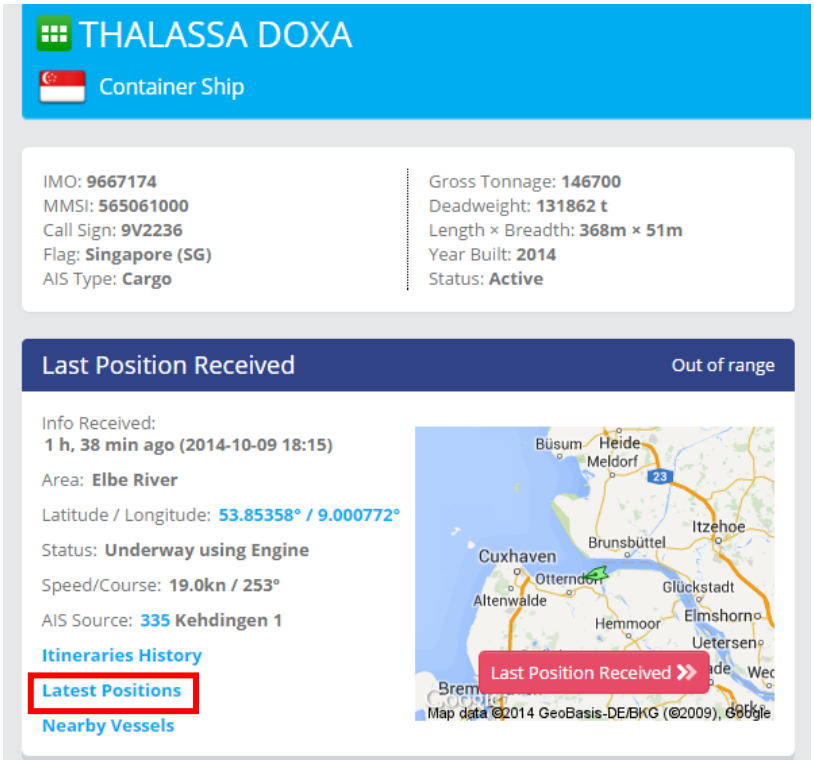

Abbildung 95: MarineTraffic-Webseite – Detailseite eines Schiffes

Der aktuelle Routenverlauf wird in einer Tabelle angezeigt, zu sehen in Abbildung 96. Insofern die Sendefrequenz den Anforderungen entspricht und der Datensatz somit als

brauchbar angesehen werden kann, kann der Routenverlauf über einen Klick auf den Button „Export Data“ (roter Kasten in Abbildung 96) im gewünschten CSV-Format exportiert werden).

Zeitstempel	AIS-Quelle	Geschwindigkeit	Longitude	Latitude	Kurs	Auf Echtzeitkarte anzeigen
2014-10-09 19:34	T-AIS	18.0	8.492588	53.9737	280	
2014-10-09 19:32	T-AIS	17.9	8.508575	53.97195	280	
2014-10-09 19:30	T-AIS	18.7	8.524495	53.97029	278	
2014-10-09 19:28	T-AIS	18.4	8.541168	53.9688	278	
2014-10-09 19:26	T-AIS	17.9	8.565725	53.96653	280	
2014-10-09 19:23	T-AIS	17.9	8.581962	53.96463	283	
2014-10-09 19:21	T-AIS	17.8	8.597848	53.962	289	
2014-10-09 19:19	T-AIS	18.0	8.612879	53.95853	293	
2014-10-09 19:17	T-AIS	18.4	8.633888	53.95129	308	
2014-10-09 19:14	T-AIS	18.6	8.652132	53.94156	318	

Abbildung 96: MarineTraffic-Webseite – Routenverlauf eines Schiffes

2. Schritt

Die EMod-Datei wird mit den jeweiligen Eigenschaften des Schiffes, die aus der MarineTraffic-Webseite ausgelesen werden, gefüllt (Abmessungen, Gewicht etc.). Hierbei muss darauf geachtet werden, dass die MMSI in der CSV-Datei mit der MMSI in der EMod-Datei übereinstimmt. Außerdem muss der Agententyp „Routing Agent“ zugewiesen werden. Zum Einlesen der CSV-Datei in den MarineTraffic Federate müssen die in Kapitel 2.4.4 beschrieben Anweisungen befolgt werden.

3. Schritt

Zuletzt werden die MTS sowie der MarineTraffic Federate gestartet. Nach dem Start der MTS wird die erstellte EMod-Datei ausgewählt. Zu Beginn der Simulation werden alle Routenpunkte vom MarineTraffic Federate über HLA an die MTS übermittelt. Zusätzlich werden zur Laufzeit vom MarineTraffic Federate die in der CSV-Datei enthalten Geschwindigkeiten an die MTS übermittelt. Die Geschwindigkeiten werden Agenten immer zu dem Zeitpunkt als neue Sollgeschwindigkeit übergeben, an dem das dazugehörige AIS-Signal des betrachteten Schiffes gesendet wurde. Anschließend können die Sollwerte mit den Istwerten für die Fahrtdauer verglichen und etwaige Abweichungen analysiert werden. Die Fahrtdauer der Soll- und Istwerte ergibt sich aus der zeitlichen Differenz zwischen dem ersten (Start) und dem letzten (Ziel) betrachteten Routenpunkt.

5. Projektorganisation

Dieses Kapitel behandelt einige organisatorische Aspekte der Projektgruppenarbeit. Es wird auf die organisatorische und entwicklungsbezogene Teameinteilung und den zeitlichen Ablauf bzw. die über das Jahr verteilten Meilensteine der PG eingegangen. Daran anschließend werden die wichtigsten Projektmanagement- und Entwicklertools der PG MAPS vorgestellt.

5.1 Organisatorische Umsetzung

5.1.1 Teams

Die PG bestand aus zehn studentischen Mitgliedern. Es fand sowohl eine organisationale Einteilung der Mitglieder statt als auch eine Einteilung in Entwickler-Teams.

Aus organisationaler Sicht wurden die Mitglieder in folgende Teams aufgeteilt:

- Projektleiter: Kerem Sevimli, Christoph Schmitt
- Administratoren: Hendrik Müller, Peter Schmeißer
- Dokumentationsbeauftragte: Hewad Osmani, Patrick Schäfer
- Webseitenbeauftragte: Frank Gröger, Peter Stedeler
- Social-Event-Manager: Wiebke Osmers, Stephan Robbers

Die Einteilung der Entwickler-Teams wurde entlang der der drei Haupt-Komponenten der verteilten Simulation, getroffen. Diese Einteilung schien plausibel, da die Haupt-Komponenten die größten zusammenhängenden Aufgabenpakete definierten. Zusätzlich wurde ein Entwickler-Team für die Bearbeitung aller GUI-spezifischen Aufgaben eingeteilt. Somit entstanden folgende Entwickler-Teams:

- MTS/ Agenten/ physikalisches Modell: Kerem Sevimli, Christoph Schmitt, Hewad Osmani
- Visualisierung/ GUI: Wiebke Osmers, Stephan Robbers
- Umweltsimulator: Frank Gröger, Patrick Schäfer, Peter Schmeißer
- Analyse-Komponente: Hendrik Müller, Peter Stedeler

5.1.2 Meilensteine

Die zeitliche Organisation der Projektgruppe erfolgte entlang von Meilensteinen, die im Voraus nach dem Wasserfall-Prinzip geplant wurden. Während der einzelnen Meilensteine wurden die Funktionen, den agilen Projektmanagementmethoden folgend, in User Stories beschrieben und abgearbeitet. Zur Verwaltung der offenen Aufgaben sowie des Projektfortschritts wurde das Projektmanagementtool JIRA verwendet (s. Kapitel 5.2.4). In diesem Tool sind auch alle User Stories zur Nachvollziehbarkeit abgebildet.

Im Folgenden werden die einzelnen Meilensteine der PG MAPS aufgezählt und erläutert:

1. Meilenstein 20.12.2013 – Erstellung des Prototypen

Für den 1. Meilenstein stand die Einarbeitung in die einzelnen Technologien sowie die Erstellung eines Prototyps auf Basis von MASON und GeoMason im Vordergrund. Weiterhin wurden die Seminarthemen, die einen inhaltlichen Bezug zur Projektgruppe haben, verfasst.

2. Meilenstein 13.01.2014 – Ergebnisse der Evaluierung

Der 2. Meilenstein umfasste die Evaluierung eines geeigneten Frameworks für die Umsetzung der MTS. Hierbei wurden die Funktionen des Frameworks (MASON und SevenCs) im Hinblick auf die Anforderungen der MTS überprüft.

3. Meilenstein 31.03.2014 – Abgabe der Systemspezifikation

Zum 3. Meilenstein wurden die Systemspezifikationen in UML-Notation dokumentiert. In den Systemspezifikationen wurden die Komponenten detailliert dargestellt. Weiterhin wurde eine einfache Simulation mit Visualisierung bereitgestellt.

4. Meilenstein 12.05.2014 – Physikalische Simulation

Der 4. Meilenstein beinhaltete die Fertigstellung der Simulations-Komponente (der MTS) in ihrer Grundstruktur. Dies bezieht sich vor allem auf die Systemarchitektur. Zudem wurden bis hierhin Tests zur Ergebnisüberprüfung der Simulation durchgeführt. Ab dem Release zum 4. Meilenstein konnten die an der verteilten Simulation beteiligten Komponenten der MTS über den HLA-Standard kommunizieren. Weiterhin wurde der Umweltsimulator fertiggestellt und über die Kommunikationsschnittstelle angebunden.

5. Meilenstein 30.06.2014 – Fertigstellung der Analyse-Komponente

Das Ziel bis zum 5. Meilenstein war es, die Analyse-Komponente fertigzustellen.

6. Meilenstein 01.09.2014 – Feature-Freeze

Bis zum 6. Meilenstein sollten alle wesentlichen Features implementiert werden. Ab diesem Zeitpunkt wurde nur vereinzelt mit der Implementierung neuer Features begonnen.

7. Meilenstein 15.10.2014 – Abgabe des Endberichts und Abschlussreview

Mit dem 7. Meilenstein endete die Projektgruppe MAPS. Zur Abgabe wurde eine Projektdokumentation bzw. ein Endbericht verfasst. Außerdem fand ein Abschlussreview statt.

5.2 Projektmanagement- und Entwicklertools

5.2.1 Eclipse

Eclipse ist eine weit verbreitete, integrierte Entwicklungsumgebung (Integrated Development Environment, IDE), die in der Projektgruppe Anwendung findet. Bei der eingesetzten Version handelt es sich um das Kepler-Release. Die verwendete Konfiguration ist im Repository unter „\tools\special-maps-eclipse“ zu finden.

Folgende Plugins werden zusätzlich verwendet:

- Checkstyle, um weitestgehend einheitliche Codekonventionen einzuhalten.
- Thumper, um die SOM-, FOM- und Mapping-Dateien zu generieren.
- YML, um mit den Konzepten und Datentypen des HAGGIS-Datenmodells arbeiten zu können.

Insbesondere aufgrund des Thumper- und YML-Plugins war die Verwendung von Eclipse auch für die Programmierung in Java naheliegend. Zudem hatten die meisten PG-Mitglieder bereits Erfahrung mit Eclipse.

5.2.2 Maven

Maven ist ein Werkzeug zum Automatisieren des Build-Prozesses, das vor allem im Java-Umfeld eingesetzt wird. In der Projektgruppe wurde es primär verwendet, um die Abhängigkeiten zu externen Bibliotheken aufzulösen sowie die jeweiligen Releases automatisiert zu erstellen.

5.2.3 SVN

Apache Subversion (SVN) ist eines der meistgenutzten Open-Source Versionsverwaltungs-systeme. Alle hochgeladenen Dateien werden mit einer einfachen Revisionszählung in einem zentralen Repository versioniert. Sämtliche Bestandteile der MTS wurden über das SVN versioniert und migriert.

Die meisten Mitglieder hatten bereits Erfahrung mit SVN, insbesondere mit dem GUI-Client TortoiseSVN, weshalb die Wahl auf dieses Tool fiel.

5.2.4 JIRA

JIRA ist eine Webanwendung für Projektmanagement-Tätigkeiten. Im Zusammenhang mit der Projektgruppe wurden über JIRA die anstehenden Aufgabenpakete geplant, auf die einzelnen PG-Mitglieder bzw. -Gruppen aufgeteilt und die Planung und Steuerung von Meilensteinen und User Stories durchgeführt.

5.2.5 Confluence

Confluence dient der allgemeinen Dokumentation, z.B. der Beschreibung der Komponenten der MTS oder der Entwicklungsarchitektur. Die webbasierte Anwendung setzt beim Aufbau und bei der Verwaltung von Artikeln auf dem bekannten Wiki-Stil (ähnlich wie Wikipedia) auf und sorgt dabei für bessere Lesbarkeit und Nachverfolgbarkeit (mit Hilfe von querverweisenden Links)

6. Projektfazit

In diesem Kapitel soll ein kurzes Resümee von der PG-Arbeit gezogen werden. Es wird dargelegt, was zukünftige PGs in der täglichen Arbeit besser machen können bzw. vermeiden sollten. Zum Schluss wird noch ein kurzer Blick in eine mögliche Zukunft der in der PG entstandenen verteilten Simulation gewagt.

6.1 Rückbetrachtung

Rückbetrachtet verlief die Arbeit in der Projektgruppe insgesamt harmonisch und ohne größere Zwischen- bzw. Problemfälle. Der Zeitplan der Projektgruppe konnte eingehalten werden.

Mit der Zeit kristallisierten sich auf Basis der Vorkenntnisse spezifische Aufgabengebiete der PG-Mitglieder heraus. So haben sich einige Mitglieder mehr auf die Entwicklung, andere mehr auf konzeptuelles Arbeiten und die Dokumentation konzentriert.

6.2 Lessons Learned

Für die meisten Teammitglieder war die Projektgruppe die erste umfangreichere Erfahrung mit Projektarbeit. Das kollaborative Arbeiten, die Aufgabenverteilung und das Arbeiten mit zuvor unbekannten Projektmanagement- und Entwicklerprogrammen und -methoden war für viele eine neue Erfahrung, die mit hohem Wert mit Blick auf den weiteren Berufsweg einzuschätzen ist. Auch in der Anwendungsentwicklung konnten die PG-Mitglieder an Erfahrung dazugewinnen.

Obgleich sich die Rollenverteilung auf die einzelnen PG-Mitglieder als gut gewählt herausstellte, ließ sich jedoch eine leichte Ungleichverteilung der Aufgabenpakete und der Zeit, die jeder einzelne in die Projektgruppenarbeit stecken musste, erkennen. Gerade die Rolle der Projektleitung war mit zeitlichem Mehraufwand verknüpft.

Als sehr hilfreich erwiesen sich die zu Beginn der Projektgruppe abgehaltenen Programmierwochenenden. Sie haben die Programmierfähigkeiten der PG-Mitglieder sowie das Gruppengefühl gestärkt.

Der Einsatz von Projektmanagement- und Softwareentwicklungswerkzeugen wie JIRA, Confluence und TortoiseSVN haben den Aufwand für die Koordination und Durchführung

des Entwickler-Prozesses gemindert. Ein Teil der PG konnte bereits Vorerfahrung mit den Tools vorweisen, diese Personen haben deren Anwendung im PG-Umfeld überhaupt erst möglich gemacht. Dennoch wurde die Dokumentation der individuellen Tätigkeiten in JIRA ab und an nicht konsequent durchgezogen bzw. erfolgte in größeren Abarbeitungsblöcken.

6.3 Ausblick

Die verteilte Simulation, bestehend aus MTS, Umweltsimulator und Analyse-Komponente, erlaubt dem Anwender, Verkehrsszenarien der Schifffahrt zu definieren, simulieren und auszuwerten. Im Forschungsumfeld der Abteilung Systemanalyse und -optimierung des Departments für Informatik wird die Simulation weiter Anwendung finden.

Eine Anschluss-Projektgruppe, die sich mit der Fortentwicklung der MTS befasst, ist in Aussicht. Die Umsetzung des Framework-Charakters und die Anbindung an HLA garantiert zukünftigen Projektgruppen eine leichte Erweiterbarkeit sowohl der MTS als auch des Umweltsimulators und der Analyse-Komponente der verteilten Simulation. Die hier vorliegende Abschlussdokumentation, die sich insbesondere an Personen richtet, die mit der Fortentwicklung der MTS zu tun haben, sollte sie dabei unterstützen.

Während der Projektgruppenarbeit sind einige Ideen für sinnvolle Erweiterungen der MTS aufgekommen, die jedoch aufgrund von Ressourcen- und Zeitmangel noch nicht umgesetzt werden konnten und somit genügend Aufgaben für zukünftige Projektgruppen bereithalten. Als Beispiel ist hier die Abbildung von Wellen im physikalischen Modell genannt.

IV Glossar

- **Agent:** Ein Agent ist eine eigenständige und eigendynamisch handelnde Software-Komponente. In unserem Fall ist jedes Schiff durch einen Agenten abgebildet. Es gibt unterschiedliche Agententypen, die sich hinsichtlich ihres Verhaltens bzw. ihrer Intelligenz unterscheiden.
- **AIS:** Das AIS (Automatic Identification System) dient der Identifizierung von Schiffen. Sie geben über das System Auskunft über eigenen Informationen, wie z.B. Name, Schiffstyp und Position des Schiffes.
- **Analyse-Komponente:** Sie ist ein eigenständiger Federate innerhalb der RTI.
- **ArcGIS:** ArcGIS ist eine Softwareplattform an GIS- (Geoinformationssystem-) vom Environmental Systems Research Institute (ESRI), mit dem u. a. elektronische Karten mit Zusatzinformationen angereichert werden können.
- **ArcMap:** ArcMap ist eine Anwendung der ArcGIS Softwareplattform, mit dem ENCs im S-57-Standard in das Shapefile-Format überführt werden.
- **Automaten:** s. StateMachine
- **Behavior:** Die Behaviors umfassen alle Komponenten der MTS, die die Verhaltensweisen eines Agenten beschreiben, wie z.B. Ruder- oder Rechtsfahr-Behavior.
- **BSH:** Bundesamts für Seeschifffahrt und Hydrographie.
- **CERTI:** CERTI ist eine Open-Source HLA-RTI.
- **DOF:** Die DOF (Degrees of Freedom) sind ein Konzept, um die Bewegungsfreiheiten von Schiffen zu beschreiben. Freiheitsgrade, die in der MTS implementiert wurden sind surge, sway und yaw und – ansatzweise – heave.
- **Dump:** Bezeichnet eine Kopie oder einen Auszugs eines Speicher- bzw. Dateiinhaltes.
- **ECS:** ECS (Entity-Component-System) ist ein Software-Architektur-Modell. Für die MTS wird dieses Schema zum Entity-Component-Aspects- (ECA-) Modell abgeändert.
- **Ecore:** Ecore ist ein Metamodell, welches die Objektmodelle im EMF beschreibt. Ecore ist die Modellierungssprache, die für das HAGGIS-Datenmodell verwendet wird.
- **ENC:** ENCs (Electronic Nautical Charts) sind elektronische Seekarten, die mit Hilfe des ECDIS genutzt werden können. Eine ENC muss dem S-57-Standard entsprechen.
- **ECDIS:** ECDIS (Electronic Chart Display and Information System) ist ein computerbasiertes Navigationsinformationssystem, mit dem u.a. ENCs angezeigt werden können.
- **EMF:** Das EMF (Eclipse Modeling Framework) ist ein Java-basiertes Framework, das der automatisierten Erzeugung von Quellcode aus Modellen dient.
- **EMod:** EMod ist ein Dateiformat in XML-Syntax, die zum Aufsetzen der Simulation genutzt wird.
- **Federate:** Ein Federate ist ein einzelner Simulator (z.B. Umweltsimulator) innerhalb der Federation.
- **Federation:** Die Federation ist die komplette ausführbare Simulation inkl. aller Simulatoren und der RTI.

- **FINO-Forschungsplattformen:** Die FINO- (Forschungsplattformen in Nord- und Ostsee) Messstationen sind drei Plattformen, die der PG als Datenquelle für Wind- und Wellendaten dienen.
- **FOM:** Das FOM (Federation Object Model) beschreibt alle SOM und beinhaltet Informationen über alle Federates und deren Objekte und Interaktionen.
- **GeoMason:** Erweiterung von MASON zum Verarbeiten von geometrische Operationen und Daten.
- **GRIB:** GRIB (GRIdded Binary) ist ein standardisiertes Datenformat, in dem historische und vorausberechnete Wetterdaten gespeichert werden.
- **HAGGIS-Datenmodell:** s. Semantisches Datenmodell
- **Heave:** Heave bezeichnet die vertikal-verlaufende lineare Schiffsbewegung, die Auf- und Ab-Bewegung des Schiffes
- **Heading:** Das Heading beschreibt die Richtung, in der das Bug (die vordere Spitze) des Schiffes zeigt.
- **HLA:** HLA (High Level Architecture) ist eine Softwarearchitektur zur integrierten und verteilten Simulation. Sie wurde vom U.S. Department of Defense entwickelt.
- **HLA-Wrapper:** Der HLA-Wrapper übersetzt „HLA C++“-Funktionen in Java.
- **JFreeChart**: JFreeChart ist ein Java-Framework für die grafische Erstellung von verschiedenen Diagrammen.
- **JSON:** JSON (JavaScript Object Notation) ist ein Datenformat, das den Datenaustausch in einfacher lesbarer Form erlaubt. In der PG dient sie der Konfiguration der Analyse-Komponente.
- **JTS:** Die JTS (Java Topology Suite) ist eine Java Programmbibliothek für räumliche Datenverarbeitung. GeoMason benutzt die Bibliothek für geometrischen Operationen.
- **LAT:** LAT (Lowest Astronomical Tide) ist In Deutschland und den Nordsee-Staaten definiert als der örtlich „niedrigste mögliche Gezeitenwasserstand“ (Seekartennull).
- **MarineTraffic:** MarineTraffic ist ein Service, der aktuelle AIS-Daten von weltweit fahrenden Schiffen kostenlos auf ihrer Homepage www.marinetraffic.com zur Verfügung stellt. Auf der Webseite können die Schiffe und ihre Routen auf einer Karte nachverfolgt werden.
- **MASON:** MASON (Multi-Agent Simulator Of Neighborhoods… or Networks… or something…) ist ein Simulations-Framework (-Toolkit) in Java.
- **MMSI:** Die MMSI (Maritime Mobile Service Identity) ist die eindeutige, weltweit gültige Rufnummer des mobilen Seefunkdienstes Kennzeichnung für eine See- oder Küstenfunkstelle.
- **MTS:** Die MTS ist die Kernkomponente der von der PG MAPS entwickelten verteilten Simulation. Sie umfasst die Komponenten zur Definition des Physical Model, der Agententypen, der Verhaltenslogik und ihre Visualisierung. Die MTS ist ein eigenständiger Federate innerhalb der RTI.
- **MTSScheduler:** Der MTSScheduler regelt den Ablauf der MTS und die Zeitsynchronisierung mit ihren Co-Simulatoren. Er baut auf dem Standard-MASON-Scheduler auf.
- **NMEA:** NMEA ist ein Standard, der die Kommunikation zwischen Navigationsgeräten auf Schiffen untereinander bestimmt. Er wurde von der National Marine Electronics Association (NMEA) festgelegt.
- **NOAA:** Die NOAA (National Oceanic and Atmospheric Administration, zu Deutsch Nationale Ozean- und Atmosphärenverwaltung) ist die amerikanische Behörde für Wetter und Ozeanografie.

- **OMT:** Das OMT (Object Model Template) ist Grundlage für SOM-, FOM- und XML-Dateien.
- **PG MAPS:** Abkürzung für Projektgruppe Manöverplanung und Simulation.
- **Physical Model:** Das Physical Model beschreibt die Schiffsdynamik unter Berücksichtigung aller relevanten Einflüsse (z.B. Umwelteinflüsse) und ihre Berechnung.
- **Query:** Ein Query ist eine Abfrage zum Beispiel auf einer Datenbank, die einen Rückgabewert in Form eines formalen Ausdrucks erwartet.
- **RTI:** Die RTI (Runtime Infrastructure, deutsch: Laufzeitinfrastruktur) ist das zentrale Kommunikationselement zwischen den über den HLA-Standard kommunizierenden Federates innerhalb der Federation. Das RTIG (Runtime Infrastructure Gateway) ist dabei der zentrale Prozess, der die Kommunikation der Federates koordiniert.
- **Semantisches Datenmodell:** Ein semantisches Datenmodell ist eine abstrakte, formale Beschreibung und Darstellung eines Ausschnittes der wahrgenommenen Welt. Innerhalb der PG MAPS wird das HAGGIS-Datenmodell verwendet.
- **Simulation:** Die Simulation ist eine Analysemethode für die Untersuchung von komplexen Systemen, wie z.B. zeitlichem Ablaufverhalten.
- **Simulator:** Ein Simulator ist die Nachbildung eines zu simulierenden Systems für verschiedenartige Zwecke (z.B. zu Test- und Forschungszwecken).
- **SOM:** Das SOM (Simulation Object Model) beinhaltet Informationen über Objekte und Interaktionen eines einzelnen Federates.
- **Shapefile:** Shapefile ist ein von der Firma ESRI entwickeltes Datenformat für Geodaten.
- **S-57:** S-57 ist ein von der Internationalen Hydrographischen Organisation (IHO) festgelegter internationaler Standard, mit dem nautische Daten (wie Seekarten) beschrieben werden. Er wird in Zukunft vom S-101-Standard abgelöst.
- **StateMachine:** Die StateMachine (auch Automat) repräsentiert den Laufzeitzustand eines Querys pro Element und ermitteln so, wann und ob ein Query für ein Element gilt.
- **Surge:** Surge bezeichnet die längsverlaufende lineare Schiffsbewegung, demnach das Vor- und Zurückfahren des Schiffes.
- **Sway:** Sway bezeichnet die seitlich verlaufende lineare Schiffsbewegung.
- **TEU:** Die TEU (Twenty-foot Equivalent Unit) ist eine Einheit, die durch die Anzahl von ISO-Standard-Containern verschiedener Größen ausgedrückt wird. Sie dient der PG MAPS als Beschreibung der Ladekapazität der Schiffe.
- **Thumper:** Der Thumper erstellt aus Klassen in HLA SOM- und FOM-Dateien.
- **UDP:** UDP (User Datagram Protocol) ist ein verbindungsloser Protokoll-Typ, das auf IP-Netzwerken läuft, ähnlich dem TCP.
- **Umweltsimulator:** Der Umweltsimulator stellt der MTS Umwelteinflüsse zur Verfügung, die durch verschiedene Umweltszenarien generiert werden. Er ist ein eigenständiger Federate innerhalb der RTI.
- **Visualisierung:** Die Visualisierung umfasst alle Aufgaben innerhalb der MTS, die für die grafische Abbildung des aktuellen Zustands der Simulations-Komponente und ihrer Benutzeroberfläche zuständig ist.
- **Yaw:** Yaw bezeichnet die Rotation des Schiffes um seine vertikale Achse.
- **YML:** YML ist eine textuellen Modellierungssprache, die an XML angelehnt ist. Sie dient der Definition der Datenstrukturen des semantischen Datenmodells.

V Literaturverzeichnis

[Ak06] Akingbehin, K.: Towards Destructive Software Testing. In 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse, 2006; S. 374–377.

[Am06] American Bureau of Shipping: Guide for Vessel Maneuverability. http://www.eagle.org/eagleExternalPortalWEB/ShowProperty/BEA%20Repository/Rules&Guides/Current/145_VesselManeuverability/VesselManeuverabilityGuide_June06, 09.10.2014.

[An03] Andresen, A.: Komponentenbasierte Softwareentwicklung. Mit MDA, UML und XML. Hanser, München, Wien, 2003.

[Ar14] Arbeitsgemeinschaft der Vermessungsverwaltungen - ADV-Online: Universale-Transversale-Mercator-Projektion (UTM-Abbildung). http://www.adv-online.de/icc/c/Geodaetische-Grundlagen/UTM-Abbildung/, 13.10.2014.

[Ba99] Balzert, H.: Lehrbuch Grundlagen der Informatik. Konzepte und Notationen in UML, Java und C++, Algorithmik und Software-Technik, Anwendungen ; mit 2 CD-ROMs. Spektrum, Akad. Verl, Heidelberg [u.a.], 1999.

[Be10] Beschnidt, J.: Virtual Waterway, eine Simulationsumgebung für Verkehrsabläufe auf Binnenwasserstraßen, Holzgartenstr. 16, 70174 Stuttgart, 2010.

[Bid04] Bierwirth, M.: Schuldverteilung bei Schiffskollisionen. Diplomarbeit, Bremen, 2004.

[Bu12] Bundesamt für Seeschifffahrt und Hydrographie (BSH): GRIB2 Readme. ftp://ftp.bsh.de/Stroemungsvorhersagen/grib2/README_de.txt, 28.08.2014.

[Bu14a] Bundesamt für Seeschifffahrt und Hydrographie (BSH): Gezeiten. http://www.bsh.de/de/Meeresdaten/Vorhersagen/Gezeiten/, 28.08.2014.

[Bu14b] Bundesamt für Seeschifffahrt und Hydrographie (BSH): Strömungen. http://www.bsh.de/aktdat/modell/stroemungen/stroemungspakete.htm, 28.08.2014.

[Bu14c] Bundesamt für Seeschifffahrt und Hydrographie (BSH): BSH Startseite. http://www.bsh.de/de/index.jsp, 09.10.2014.

[Bu14d] Bundesamt für Seeschifffahrt und Hydrographie (BSH): Informationen zu ENCs. http://www.bsh.de/de/Produkte/Karten/Elektronische_Seekarten/ENCInformationen.jsp, 14.10.2014.

[BuJV14] Bundesministerium der Justiz und für Verbraucherschutz: Internationale Regeln von 1972 zur Verhütung von Zusammenstößen auf See. (Anlage zu § 1 der Verordnung zu den Internationalen Regeln von 1972 zur Verhütung von Zusammenstößen auf See) (Kollisionsverhütungsregeln - KVR). http://www.gesetze-im-internet.de/seestro_1972/BJNR008160977.html, 14.09.2014.

[BuV14] Bundesministerium für Verkehr und digitale Infrastruktur: Das Projekt FINO-WIND. http://www.dwd.de/fino-wind/, 09.10.2014.

[CCW13] Coletti, M.; Crooks, A.; Wise, S.: GeoMason. GeoSpatial Support for MASON. http://cs.gmu.edu/~eclab/projects/mason/extensions/geomason/geomason.pdf, 02.08.2014.

[Cl10] Cleff, T.: Basiswissen Testen von Software. Vorbereitung zum Certified Tester (Foundation Level) nach ISTQB-Standard. W3L, Herdecke [u.a.], 2010.

[Co14] Coral Creek Software: SQLite Expert. The expert way to SQLite. http://www.sqliteexpert.com/, 27.08.2014.

[Cr14] Cruise L.l.c.: IHO S-57 / Electronic Nautical Charts (ENCs). Object and Attribute Catalogue. http://www.s-57.com/, 09.10.2014.

[ES14] ESRI Inc.: ArcGIS for Desktop. http://www.esri.com/software/arcgis/arcgis-for-desktop, 19.07.2014.

[Ev14] Evolution Solutions Co.: Pencil Project. http://pencil.evolus.vn/, 22.10.2014.

[Fo11] Fossen, T. I.: Handbook of marine craft hydrodynamics and motion control. Vademecum de Navium Motu Contra Aquas et de Motu Gubernando. Wiley, Chichester, West Sussex, 2011.

[Fo14] Forschungs- und Entwicklungszentrum Fachhochschule Kiel GmbH: FINO1,2,3 - Forschungsplattformen in Nord- und Ostsee Nr. 1,2,3. http://www.fino-offshore.de/de/, 09.10.2014.

[Ge14] George Mason University, Department of Computer Science: MASON Multiagent Simulation Toolkit. http://cs.gmu.edu/~eclab/projects/mason/, 19.08.2014.

[Go09] Gomarasca, M. A.: Basics of geomatics. Springer, Dordrecht, London, 2009.

[Ha14] Hafen Hamburg Marketing e.V.: Containerschiffe nach Größenklasse (Tragkraft in TEU). http://www.hafen-hamburg.de/content/containerschiffe, 09.09.2014.

[In08] International Hydrographic Organization (IHO): S-101 ENC Product Specification. (Next Generation ENCs). http://www.iho.int/mtg_docs/industry/s-101_workshop_08/ECDIS_WS-3_S-101_InfoPaper.pdf, 26.09.2014.

[In12] International Hydrographic Organization (IHO): IHO S-101. The Next Generation ENC Product Specification. http://www.iho.int/mtg_docs/com_wg/TSMAD/TSMAD_Misc/S-101/S-101_info_paper_Jun12_EN.pdf, 26.09.2014.

[JL05] Jena, S.; Liebelt, N.: Heuristische Algorithmen. am Beispiel des A*-Algorithmus / 8-Puzzle, 2005.

[Li11] Lippert, F.: Seitenpeilung. http://www.sportbootnavigation.de/seitenpeilung/, 14.07.2014.

[Ma14] Maltenoz Limited: AIS Marine Traffic. AIS-Schiffspositionen/-verkehr in Echtzeit. http://www.marinetraffic.com/de/, 02.10.2014.

[Mü14] Müller, T.: H2 Database Engine. http://www.h2database.com/html/main.html, 24.08.2014.

[Na11] National Weather Service - Climate Prediction Center: wgrib2: wgrib for GRIB-2. read and write grib2 files. http://www.cpc.noaa.gov/products/wesley/wgrib2/, 27.08.2014.

[Na14] National Oceanic and Atmospheric Administration (NOAA), US Department of Commerce: National Oceanic and Atmospheric Administration (NOAA) Home Page. http://www.noaa.gov/, 09.10.2014.

[No07] Noé, M.: Projektbegleitendes Qualitätsmanagement. Der Weg zu besserem Projekterfolg. PUBLICIS, Erlangen, 2007.

[No14] Novak, C.: JMapView. http://sourceforge.net/projects/jmapview/, 13.10.2014.

[Ob14] Object Refinery Limited: JFreeChart. http://www.jfree.org/jfreechart/, 31.09.2014.

[op14] openstreetmap.de - FOSSGIS e.V.: JMapViewer (JOSM). http://josm.openstreetmap.de/doc/org/openstreetmap/gui/jmapviewer/JMapViewer.html, 14.10.2014.

[Or14a] Oracle Corporation: Package java.awt (Java Platform SE 7). http://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html, 10.10.2014.

[Or14b] Oracle Corporation: Package javax.swing (Java Platform SE 7). http://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html, 10.10.2014.

[PI79] Methods for analysing wind, wave and swell data to estimate on an annual basis the number of days, and the maximum duration of periods during which port and ship operations will be impeded by these elements. Pianc, 1979.

[PRW13] Pezze, M.; Rubinov, K.; Wuttke, J.: Generating Effective Integration Test Cases from Unit Ones. In 6th International Conference on Software Testing, Verification and Validation, 2013; S. 11–20.

[Sa04] Sailtrain.co.uk: Navigation and Chart work - Speed Over the Ground(SOG). http://www.sailtrain.co.uk/navigation/sog.htm, 13.10.2014.

[Sa12] Saldaña-Ramos, J. et al.: Design of a competence model for testing teams. In IET Software, 2012, 6; S. 405–415.

[Se14] SevenCs GmbH: SevenCs Startseite. http://www.sevencs.com/, 09.10.2014.

[Si03] Siedersleben, J.: Softwaretechnik. Praxiswissen für Software-Ingenieure. Hanser, München, Wien, 2003.

[SmR13] Beege, R.: Regel 19. Verhalten von Fahrzeugen bei verminderter Sicht. http://www.segelnmitromeo.de/index_htm_files/schleppen_anlegen_ankern.pdf, 12.10.2014.

[SQ14] SQLite-Team: SQLite. http://www.sqlite.org/, 24.08.2014.

[Su99] Sun Microsystems Inc.: Java look and feel design guidelines. http://www.ashcavai.com/JLFDG05.pdf, 13.10.2014.

[t-07] t-machine.org: Entity Systems are the future of MMOG development – Part 2. http://t-machine.org/index.php/2007/11/11/entity-systems-are-the-future-of-mmog-development-part-2/, 09.10.2014.

[TdX10] Tillmann, N.; de Halleux, J.; Xie, T.: Parameterized Unit Testing: Theory and Practice. In 32nd International Conference on Software Engineering, 2010; S. 483–484.

[Th14] The Eclipse Foundation: Eclipse Modeling Project. http://www.eclipse.org/modeling/emf/, 09.10.2014.

[Tu14] Tuukkanen, K.: Java Marine API. NMEA 0183 library for Java. http://ktuukkan.github.io/marine-api/, 09.10.2014.

[Vi03] Virginia Institute of Marine Science: Measuring Currents. http://web.vims.edu/physical/research/TCTutorial/currentmeasure.htm, 09.10.2014.

[Vi06] Vivid Solutions, Inc.: JTS Topology Suite.

[WA11] Wahid, M.; Almalaise, A.: JUnit Framework: An Interactive Approach for Basic Unit Testing Learning in Software Engineering. In 3rd International Conference on Engineering Education, 2011; S. 159–164.

[Wid14] Wikidot Inc.: What's an Entity System? - Entity Systems Wiki. http://entity-systems-wiki.t-machine.org/, 19.09.2014.

[Wim14a] Wikimedia Foundation Inc.: Bremsverzögerung. http://de.wikipedia.org/w/index.php?oldid=130200635, 08.09.2014.

[Wim14b] Wikimedia Foundation Inc.: GRIB. http://de.wikipedia.org/w/index.php?oldid=134478608, 24.08.2014.