Carl von Ossietzky Universität Oldenburg
Abteilung Systemanalyse und -optimierung
OFFIS – Institut für Informatik

Projektgruppe – Maritime Test- und Experimentierplattform 2

Themensteller: Prof. Dr.-Ing. Axel Hahn
M.Sc. Marius Brinkmann
M.Sc. Nils Hartmann
M.Sc. Mohamed Abdelaal
M.Sc. Peter Tank

Vorgelegt von: Julian Bergmann
Lea Bollerslev
Yvonne Caroline Brück
Linda Christin Büker
Tim Happe
Julius Möller
Dennis Müller
Ole-Christian Rösler
Alexander Stebe
Christian Steger
Hilko Wiards

Abgabetermin: 30.09.2018

Inhaltsverzeichnis

Abstract

Die vorliegende Dokumentation fasst die Ergebnisse der Projektgruppe MATE II zusammen. Die vorangegangene und die jetzige Projektgruppe entwickeln eine Plattform mit dem ein Wasserfahrzeug autonom auf See agieren kann. Hierfür wurden küsten- und schiffsseitige Komponenten entwickelt und in eine vorhandene Systemarchitektur eingebettet. Diese beinhaltet das Forschungsschiff Zuse und das virtuelle Testfeld LABSKAUS. Der thematische Schwerpunkt der Projektgruppe MATE II ist die vollautomatische Erkennung von statischen und dynamischen Objekten sowie die Einleitung geeigneter Gegenmaßnahmen zur Vermeidung von Zusammenstößen unter Berücksichtigung einiger COLREGs Regeln. Die Dokumentation beinhaltet weiterhin die Projektorganisation, die grundlegenden Projektentscheidungen, die Beschreibung genutzter Fremd-Software sowie die gruppeninternen Arbeitstechniken.

1. Einführung in das Projekt

Die Projektgruppe “Maritime Test- und Experimentierplattform II” ist der Nachfolger der PG MATE I. Diese wird im zweiten Jahr in einer Projektgruppe bestehend aus Informatikern und Wirtschaftsinformatikern der Carl von Ossietzky Universität durchgeführt.
Der Schwerpunkt dieser Projektgruppe liegt zum einen auf dem Entwickeln der fehlenden Komponenten zur Autonomisierung des Forschungsbootes Zuse und zum anderen auf der Nutzung der Testplattform LABSKAUS zur wissenschaftlichen Verifikation und Validierung der vorhandenen und entwickelten Komponenten.

1.1. Motivation

Mit zunehmender Zahl global agierender Containerschiffe [Lan], nimmt auch die Gefahr von Unfällen zu. Dabei bestanden, laut der Seeunfallstatistik der *Bundesstelle für Seeuntersuchungen* (BSU) aus dem Jahre 2017, 79.4 % der verzeichneten Seeunfälle im deutschen Hoheitsgebiet aus Kollisionen mit anderen Schiffen oder Objekten und Grundberührungen [See18, S. 33].
Laut [RB12] lässt sich bei einer Reduktion der Geschwindigkeit von 16 kn auf 11 kn der Treibstoffverbrauch um etwa 50 % reduzieren. Auch wenn die ökologischen Vorteile, bei einer Reduktion des CO_2 Ausstoßes von 50 %, nicht abzustreiten sind, ist festzuhalten, dass eine so starke Reduktion der Reisegeschwindigkeit ökonomisch nicht zu tragen ist. Mit einer um 31 % reduzierten Geschwindigkeit, steigt die Dauer der Reise um 45 % an. Es steigen also die Charterkosten der Reise an. Diese erhöhten Kosten reduzieren den ökonomischen Vorteil des geringeren Treibstoffverbrauchs. Ein Großteil der Charterkosten besteht jedoch aus den Kosten, die durch die Besatzung entstehen. Bei dem hier diskutierten Beispiel liegen diese Kosten bei 9.2 % der Gesamtkosten.
Ein teilautonomes System könnte die durch Besatzung und Bunkerung entstehenden Kosten also weiter reduzieren. Im Idealfall, also bei einem vollautonomen Schiff könnten die Kosten der Besatzung sogar komplett gestrichen werden.

Durch technologischen Fortschritt im Bereich der autonomen Schifffahrt wird die *Langsamfahrt* (engl. *Slow Steaming*) also zu einer praktikablen Lösung sowohl für ökologische, als auch ökonomische Probleme der modernen Zeit.

Das hier entwickelte System soll nicht nur als *Proof of Concept* angesehen werden, sondern später auch weiterhin aktiv eingesetzt werden. Es handelt sich um eine Erweiterung des physikalischen Testfelds LABSKAUS. So soll das im Rahmen der PG MATE II entwickelte Produkt für weitere Tests und Experimente in anderen Projekten wie beispielsweise e-Martime Integrated Reference Platform (eMIR) oder anderen Projekten der Abteilung *Systemanalyse und -optimierung* unterstützend zur Verfügung stehen.

1.2. Aufgabenstellung

Das Ziel der PG MATE II ist die Entwicklung und Verifikation eines ganzheitlichen Konzepts zur autonomen Seefahrt (vgl. [Int]). Darunter wird verstanden, dass die Aspekte der Fernsteuerung und -überwachung vereint werden mit denen der automatisierten Schifffahrt. Die Entwicklung geschieht dabei in drei Inkrementen.

Zunächst wird eine *Fernsteuerung* mit direktem Zugang zum Ruder und Motor entwickelt. Dabei gilt es sowohl Schnittstellen zu den Hardwarekomponenten zu schaffen, als auch die Kommunikation zwischen Land- und Seeseite zu ermöglichen.

Auf dieser Technologie aufbauend wird die *Aktionsplanung aus der Ferne* entwickelt. Dabei ist auf Landseite eine Electronic Chart Display and Information System (ECDIS)-ähnliche Anwendung zur Routenplanung als auch zur Überwachung zu verwenden und auf Seeseite ist es möglich, Ziele anzufahren und den Kurs zu halten.

Das letzte Inkrement ist darauf aufbauend die *autonome Fahrt*. Mithilfe von Sensoren und Kameras kann auf die Umgebung reagiert werden und dynamischen Hindernissen kann mit Hilfe einer Pfadplanung ausgewichen werden.

Von der PG MATE I wurde bereits das erste Inkrement umgesetzt. Das zweite Inkrement wurde begonnen und das Produkt ist in der Lage Wegpunkte einer vordefinierten Route an den Controller zu senden. Jedoch konnte die Abfahrt dieser Wegpunkte nicht erfolgreich getestet werden. An dieser Stelle setzt diese Projektgruppe an.

Ein weiterer Schwerpunkt wird auf der Verifikation und Validierung (V+V) liegen. Um das System und die autonome Fahrt testen zu können, wird eine Integration simulierter Daten in das physikalische Testfeld vorgenommen.

1.3. Aufbau des Dokuments

Diese Abschlussdokumentation ist in acht Kapitel gegliedert. Im ersten Kapitel wird die Motivation und die Aufgabenstellung für die PG MATE II gegeben.

Im zweiten Kapitel wird das genutzte Vorgehensmodell des Projekts dargestellt und ein zeitlicher Überblick über den gesamten Ablauf mithilfe eines Meilensteinplans skizziert. Im Anschluss werden die Meilensteine der benötigten Komponenten aufgezeigt. Zum Schluss sind die genutzten Kommunikationstechniken und Werkzeuge aufgelistet und beschrieben.

Eine Darstellung der Theoretischen Grundlagen, auf denen das Projekt aufbaut, wird im dritten Kapitel gegeben. Darunter der Stand aktueller Forschungsprojekte und die Auflistung der genutzten maritimen Standards. Des Weiteren beinhaltet das Kapitel die Grundlagen zur Navigation auf maritimen Gewässern und Kollisionsverhütungskonzepte, die die Daten aus den verwendeten maritimen Sensoren verarbeiten. Die Grundlagen zur Bildverarbeitung mithilfe von Kameradaten, die Sensordatenfusion von AIS- und Radar-Daten, die Regelungstechnik auf dem Forschungsschiff und die «Loop» Testmethoden werden erklärt.

Darauf Aufbauend werden die Anforderungen mithilfe von Methoden der Anforderungserhebung im vierten Kapitel aufgezeigt. Zunächst werden die Anforderungen an das gesamte MATE II-Produkt dargestellt und im Anschluss die der benötigten Komponenten. Die Komponenten bestehen aus dem *Küstenleitstand*, der *Kollisionsverhütung*, der *Hindernisidentifikation*, der *Regelungstechnik* und der Komponente für die *Integration simulierter Daten ins physikalische Testfeld*.

Aus den Anforderungen leiten sich Konzepte der Komponenten, sowie ein Konzept vom Gesamtsystem ab, die im fünften Kapitel enthalten sind. Aus den Konzepten hat sich die Entwicklung eines eigenen *V&V Werkzeugs* ergeben, das die Evaluierung unterstützt.

Das sechste Kapitel wird durch die Darstellung des Gesamtsystems eingeleitet und geht anschließend auf die funktionale und technische Realisierung der Konzepte durch die Komponenten *eNavigation Prototype Displays (EPD)*, *Routen-Planung*, *Pfad-Planung*, *Dynamische Objektidentifizierung*, *Objektdetektion*, *Controller*, *Simulationsadapter* und *Mate Control* ein.

An das V-Modell angelehnt werden die Komponententests, Integrationstests und Systemtests im siebten Kapitel Evaluation dargestellt. Aus den Systemtests abgeleitet wird das Test-Vorgehen mithilfe eines Szenariokatalogs im Rahmen dieses Projekts bewertet.

Im letzten Kapitel wird ein kurzes Fazit für das gesamte Projekt und die Projektgruppe gezogen und ein Ausblick für die Weiterführung des Projekts und der Komponenten gegeben. Der letzte Abschnitt beinhaltet den Anhang mit Testfällen, Anleitungen, einem Interface Control Dokument, der verwendeten Literatur, einem Glossar und einem Abkürzungsverzeichnis.

2. Projektorganisation

Das folgende Kapitel stellt das für die Projektorganisation ausgewählte Vorgehensmodell vor. Des Weiteren werden die im Projekt eingesetzten Werkzeuge und unterstützenden Software-Lösungen aufgelistet und beschrieben.

2.1. Vorgehen

Zur Durchführung eines Software-Projektes bedarf es einer passenden Strategie des gesamten Software-Entwicklungsprozesses. Diese Projektstrategie wird mit Hilfe von Vorgehensmodellen festgelegt. Je nach Anwendungsbereich und Projektgröße existieren verschiedene Vorgehensmodelle, die alle Vor- und Nachteile mit sich bringen. [Sch+10, S. 47-48]
Im Rahmen des MATE II-Projektes wurde in Absprache mit dem Lehrstuhl das V-Modell als Vorgehensmodell ausgewählt. Das V-Modell basiert im Kern auf dem Wasserfall-Modell und teilt die Softwareentwicklung in einzelne Phasen auf. Der große Unterschied zum Wasserfall-Modell ist das Zuordnen einer Testphase zu jeder Phase der Softwareentwicklung. Auf der linken Seite werden absteigend die Spezifizierungsphasen der Softwareentwicklung aufgelistet. Die Phase der Implementierung bildet den letzten Punkt dieser Kette und ist gleichzeitig der Anfang der aufsteigenden Test-Kette auf der rechten Seite. So entsteht bildlich das Namensgebende „V“.
Diese Gegenüberstellung soll zu einer möglichst hohen Testabdeckung führen, weil die Spezifikationen der jeweiligen Entwicklungsstufen die Grundlage für die Tests (Testfälle) in den entsprechenden Teststufen sind. Durch die klar definierte Struktur die das V-Modell vorgibt, ist es besonders gut anwendbar bei Projekten mit vollständigen und stabilen Anforderungen. Hierdurch ist eine gute Planung des Projektes möglich. Durch die starre Struktur ist der Aufwand von Änderungen und der Fehlerbeseitigung in den späteren Phasen des Projektverlaufes vergleichsweise hoch und teuer [Sch+10, S. 51].

Neben dem Vorteil der klaren Projektstruktur wurde das V-Modell aufgrund seines Ansatzes zur Verifikation und Validierung des Software-Produktes ausgewählt. In der Fachliteratur gibt es unterschiedliche Bezeichnungen und Anzahl der einzelnen Phasen, allerdings wird jeder Entwicklungsphase immer eine Teststufe gegenübergestellt. Im Folgenden werden die einzelnen Phasen eines solchen V-Modells aufgelistet und kurz beschrieben.

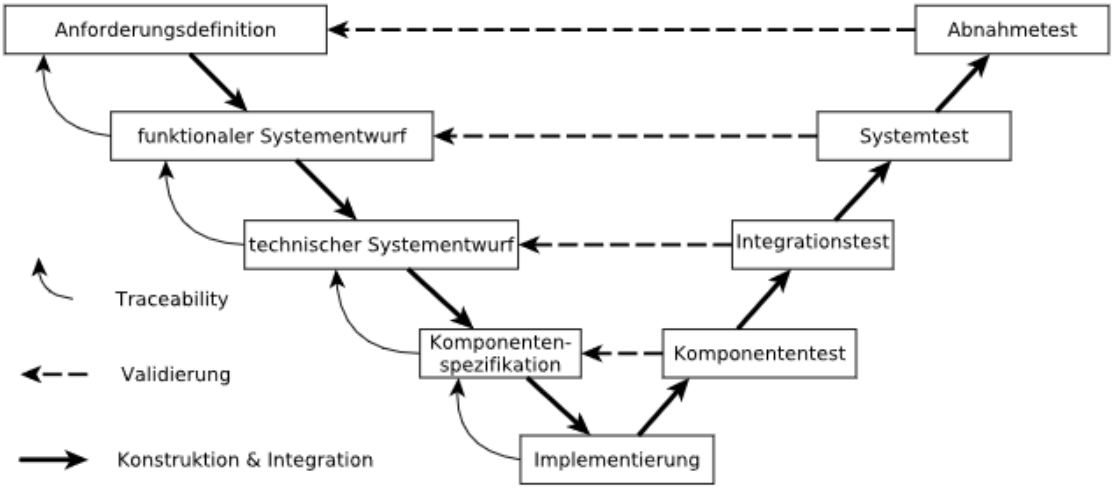


Abbildung 2.1.: V-Modell

Nutzungsanforderungen Im ersten Schritt des V-Modells werden die Nutzungsanforderungen an das System gesammelt. Sie sagen in natürlicher Sprache aus, welche Dienste das System dem Nutzer bieten soll und unter welchen Rahmenbedingungen das System betrieben wird. Sie werden in einer so abstrakten Form definiert, so dass einer möglichen Lösung nicht vorgegriffen wird [Joh]. Je nach V-Modell wird die erste Phase auch als Anforderungsdefinition bezeichnet oder mit dem Lastenheft gleichgesetzt.

Systemanforderungen Im nächsten Schritt des V-Modells werden die Systemanforderungen aufgestellt. Die Systemanforderungen beschreiben aus Black-Box-Sicht, wie das System die zuvor gesammelten Nutzungsanforderungen umsetzt. Dabei wird nicht auf die technische Umsetzung des Systems eingegangen um das gewünschte Black-Box-Verhalten zu erreichen [Joh]. Wie auch die Nutzungsanforderungen werden die Systemanforderungen zumeist in natürlicher Sprache formuliert. Je nach Fachliteratur wird abschließend aus den Systemanforderungen ein erster Architekturentwurf erstellt. Diese Phase wird öfters dem Pflichtenheft gleich gesetzt oder als funktionaler Systementwurf bezeichnet.

Systemarchitektur Die dritte Stufe des V-Modells wird die technische Realisierung des Systems entworfen. In der Systemarchitektur werden u.a. die Schnittstellen zur Systemumwelt definiert sowie das Gesamtsystem in Teilsysteme bzw. Komponenten zerlegt. Hierbei muss die Architektur die zuvor aufgestellten Systemanforderungen umsetzen. Des Weiteren werden die Schnittstellen zwischen den Komponenten definiert. Wenn eine präzise Schnittstellenspezifikation aufgestellt wurde, können Komponenten parallel entworfen und entwickelt werden [Som12, S. 66]. Diese Phase wird häufig auch als technischer Systementwurf bezeichnet.

Komponentenspezifikation Im vierten Schritt des V-Modells werden Aufgabe, Verhalten und innerer Aufbau der einzelnen Komponenten definiert. Dies geschieht mit Hilfe einer einfachen Darstellung der zu erwartenden Funktionalität oder durch ein detailliertes Entwurfsmodell [Som12, S. 67].

Implementierung Der fünfte Schritt, der gleichzeitig die Spitze des V's darstellt, werden die zuvor konzipierten und spezifizierten Komponenten mit Hilfe einer Programmiersprache entwickelt.

Komponententest In der Fachliteratur wird die erste Teststufe des V-Modells unterschiedlich bezeichnet, am geläufigsten sind die Begriffe Modultest, Unittest und Komponententest. Die kleinsten Softwareeinheiten werden je nach ausgewählter Programmiersprache, als Module, Units oder Klassen bezeichnet. Die dazugehörigen Tests werden dementsprechend Modultest, Unittest bzw. Klassentest genannt. Von der genutzten Programmiersprache abstrahiert, wird von der Komponente oder Softwarebaustein gesprochen. Der entsprechende Test wird als Komponententest bezeichnet. Dementsprechend wird beim Komponententest ein einzelner Softwarebaustein getestet. Dies geschieht isoliert von den anderen Einheiten des Systems. So wird sichergestellt das komponentenexterne Einflüsse den Test der einzelnen Komponente nicht beeinflussen. So können auftretende Fehler eindeutig der Komponente zugeordnet werden. Eine Komponente kann aus einem oder mehreren Bausteinen zusammengesetzte Einheit sein. Entscheidend ist, dass komponenteninterne Aspekte geprüft werden, jedoch nicht die Wechselwirkung mit Nachbarkomponenten. Dies geschieht erst im Integrationstest. [SL07, S. 42-43].

Integrationstest Der Integrationstest ist die zweite Teststufe im V-Modell. Das Ziel ist es Schnittstellenfehler in den entwickelten Teilsystemen zu finden. Die getesteten Teilsysteme bestehen hierbei aus zwei oder mehr zuvor im Komponententest überprüften Komponenten.

Das Zusammenspiel zwischen den einzelnen Komponenten zu testen und mögliche Fehlerzustände aufzudecken ist, neben der Überprüfung der Schnittstellen, Aufgabe des Integrationstest [SL07, S. 50]. Im Idealfall wird nach jeder Verknüpfung zweier Komponenten ein Integrationstest des entstandenen Teilsystems durchgeführt. So können Integrationstests von Teilsystemen mit mehreren integrierten Komponenten gewährleistet werden [SL07, S. 52].

Systemtest Die dritte Teststufe des V-Modells ist der Systemtest. Hierbei wird das integrierte System nach Abschluss des Integrationstests gegen die spezifizierten Anforderungen des Produktes getestet. Dies geschieht in einer Testumgebung, die möglichst nahe an die zukünftige reale Umgebung des Systems angelehnt ist. Daher sollte die Testumgebung auf allen Ebenen die später tatsächlich verwendeten Hard- und Softwareprodukte enthalten. Das Ziel des Systemtest ist es festzustellen, ob und wie gut das entwickelte System die zu Beginn erhobenen funktionalen und nicht funktionalen Anforderungen erfüllt[SL07, S. 58].

Abnahmetest In der letzten Stufe des V-Modells wird aus Sicht des Kunden getestet. Der Abnahmetest unterscheidet sich hierbei vom Systemtest im Wesentlichen darin, dass das System nicht mehr in einer Testumgebung, sondern in der Kundenumgebung getestet wird. Auf Basis des Abnahmetests entscheidet der Kunde, ob er das entwickelte System als akzeptabel ansieht und abnimmt [SL07, S. 62].

Anpassung

In der Praxis ist es üblich das ausgewählte Vorgehensmodell an das jeweilige durchzuführende Projekt anzupassen. Dieser Prozess wird „Tailoring“ genannt [Sch+10, S. 65]. Im Folgenden werden die Gründe für Anpassungen am V-Modell erklärt und die dadurch notwendigen Veränderungen am präsentierten V-Modell beschrieben. Dem MATE II Projekt fehlt eine explizite Kundensicht auf das Projekt.
Der Lehrstuhl, der das Projekt organisiert bzw. leitet, kommt dieser Sicht noch am nächsten. Durch die intensive Zusammenarbeit mit den Betreuern vermischen sich interne und externe Projektansichten. Daraus abgeleitet wurde die Phase der Nutzungsanforderungen, die in vergleichbaren V-Modellen in der Praxis mit der Phase des Lastenheftes gleichzusetzen ist [GEF18], mit der Phase der Systemanforderungen (Pflichtenheft) vermischt. So dass die Projektgruppe beide Phasen zu einer Phase, der Anforderungserhebung & Systemspezifikations-Phase, zusammenlegte. Zudem kann das System ohne den fehlenden Kunden nicht abgenommen werden.
Dies war einer der Gründe den Abnahmetest aus dem projektspezifischen V-Modell zu streichen.

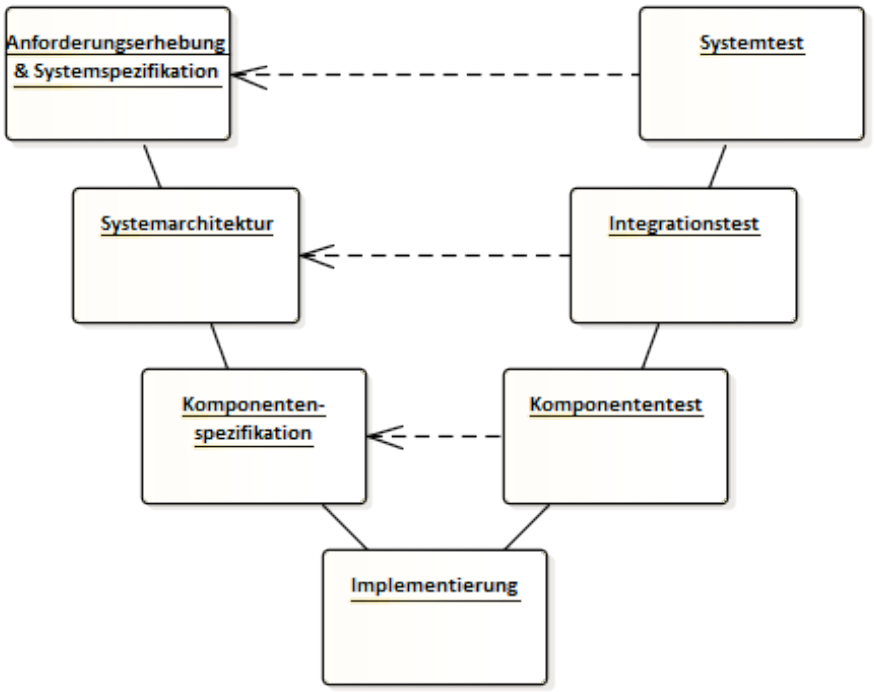


Abbildung 2.2.: An das Projekt angepasste V-Modell

Ein weiterer Grund für diese Entscheidung war die Testumgebung des Systemtests. Für einen vollständigen Systemtest wurde das MATE II Produkt in der realen bzw. finalen Umgebung des Systems getestet. Dies war der Tatsache geschuldet, dass der Projektgruppe als physikalische Testumgebung nur das Forschungsschiff Zuse zur Verfügung stand. Eine Anschaffung eines Schiffes für die Testumgebung war im Rahmen der finanziellen Mittel der Projektgruppe nicht machbar. Durch den Einsatz des Systemtests in der finalen Umgebung und in Kombination mit dem Fehlen eines Kunden ist der Abnahmetest als überflüssig anzusehen und wurde dementsprechend aus dem Vorgehensmodell gestrichen. Aus den genannten Anpassungen ergab sich somit das in Abbildung 2.2 projektspezifische V-Modell.

Durch den zusätzlich gelegten Fokus auf die modellbasierte Systementwicklung wurde bereits frühzeitig in der Entwicklung iterativ getestet. So wurden die einzelnen Komponenten bereits in der Programmier-Phase laufend getestet und aufgedeckte Fehlerquellen eliminiert. Mit Hilfe von Model- und Software In The Loop Tests wurden das System erst im Labor und dann Schritt für Schritt in der Realwelt des Systems getestet.

2.2. Ablaufplan

Im folgenden Kapitel werden die einzelnen Phasen des Projektes kurz skizziert sowie hauptsächlich der Ablaufplan der Implementierung und die dafür aufgestellten Meilensteine vorgestellt.

Zu Beginn der Projektgruppe fand im Oktober die Seminarphase statt. In dieser Zeit schrieb jedes Projektmitglied eine kurze Arbeit über ein projektrelevantes Thema. Diese wurden in den ersten Novemberwochen den anderen Mitgliedern in gemeinsamen Sitzungen mit Vorträgen präsentiert. Von Mitte November bis Weihnachten fand eine zweite Phase zur Einarbeitung statt. Hierfür arbeitete sich jedes Mitglied in Teile des bisherigen Projektes ein und stellte seinen bearbeiteten Part der Projektgruppe vor, sodass sich jedes Projektmitglied auf dem gleichen Stand bezüglich des MATE I-Projektes befand. Des Weiteren wurden bereits kleinere Aufgaben des Lehrstuhls im System umgesetzt. Gleichzeitig wurde damit begonnen Anforderungen an das System zu erstellen sowie die Teilsystem-abhängigen Kleingruppen geschaffen. Im Januar wurden in den Kleingruppen dann die Anforderungen finalisiert und darauf aufbauend erst Konzepte für das System sowie die einzelnen Komponenten erstellt. Auf Grundlage der Konzepte wurden bis Anfang Februar Software Entwürfe entwickelt. Mit den fertigen Entwürfen wurde die Konzept- und Entwurfsphase abgeschlossen und mit der Implementierung begonnen.

Für die Implementierungsphase wurden auf den Entwürfen aufbauend Meilensteine von jeder Kleingruppe erstellt und in einem Zeitstrahl visualisiert. Im Folgenden werden die einzelnen Meilensteine der Kleingruppen aufgelistet und näher beschrieben.

2.2.1. Kollisionsverhütung

Es soll eine Komponente zur Kollisionsverhütung erstellt und in die Gesamtarchitektur von MATE II eingebunden werden.

Entwicklung eines A*-Prototypen (22.02.2018)

Zunächst soll ein prototypischer A*-Algorithmus entwickelt werden. Dieser soll in der Lage sein in einem Occupancy-Grid den kürzesten Weg von einem beliebigen Startpunkt zu einem beliebigen Zielpunkt, unter Berücksichtigung von belegten Zellen, zu finden. Der A*-Algorithmus soll dabei eine Kosten-Funktion und eine Heuristik-Funktion enthalten, wobei die Heuristik-Funktion zunächst lediglich aus der euklidischen Distanz von der aktuellen Position zum Zielpunkt bestehen soll.

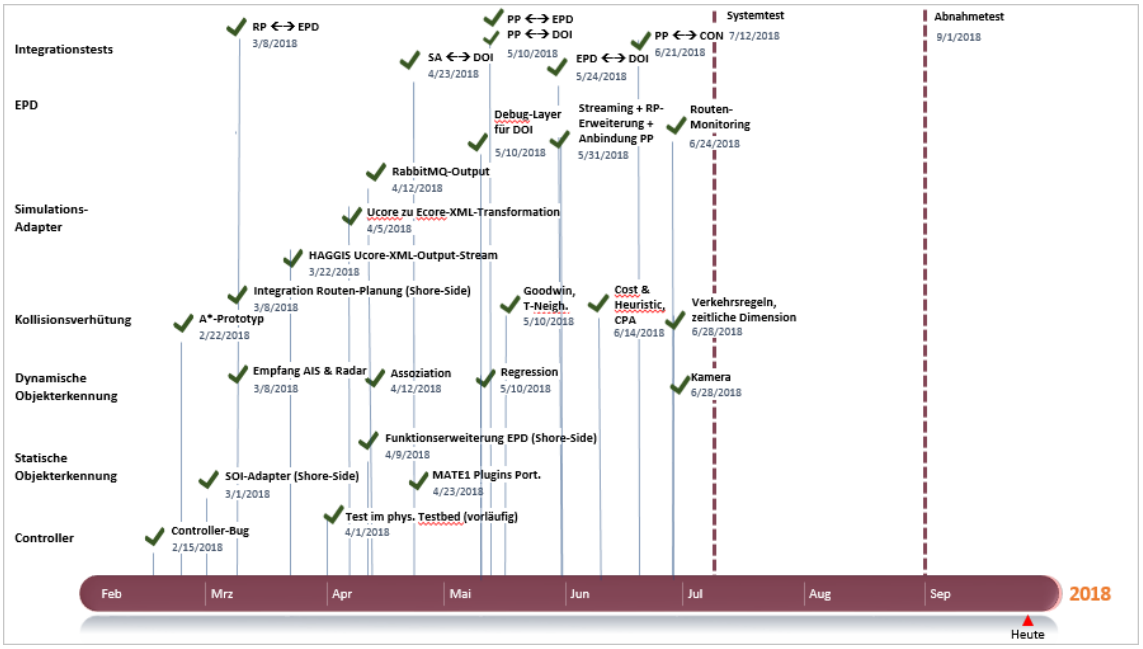


Abbildung 2.3.: Meilenstein-Zeitstrahl

Integration der landseitigen Routen-Planung (08.03.2018)

Es soll eine Routen-Planungs-Komponente erstellt werden, in der der entwickelte A*-Prototyp integriert ist. Danach soll die Komponente in die Gesamtarchitektur von MATE II integriert werden. Dazu sollen Kommunikationsschnittstellen zur landseitigen e-Navigation Prototype Display (EPD) und zum NoGoSolver implementiert werden. Die Kommunikationsschnittstelle zur landseitigen EPD soll Routen von der EPD empfangen und zur EPD versenden können. Die Kommunikationsschnittstelle zum NoGoSolver soll Anfragen bezüglich nicht befahrbaren Gebieten an den NoGoSolver senden und diese Gebiete als Antwort vom NoGoSolver empfangen können.

Erweiterung des A*-Algorithmus um die Konzepte Goodwin, T-Neighbourhood und einer erweiterten Kosten- und Heuristik-Funktion (10.05.2018)

Der prototypische A*-Algorithmus soll zunächst um das 2D Kollisionsverhütungsmodell nach Goodwin erweitert werden. Dadurch automatisch die Convention on the International Regulations for Preventing Collisions at Sea (COLREGs)-Regeln 13 bis 17 eingehalten. Außerdem soll der A*-Algorithmus das T-Neighbourhood-Konzept beachten, um den maximalen Wendekreis der Zuse zu berücksichtigen. Schließlich sollen die Kosten- und Heuristik-Funktion um die Parameter Kosten für eine Wendung und Winkelunterschied des nächsten Zielwegpunktes zur aktuellen Position erweitert werden.

Integration der schiffseitigen Pfad-Planung (10.05.2018)
Die bereits bestehende Pfad-Planungs-Komponente soll um den A*-Algorithmus erweitert werden. Es soll zudem Kommunikationsschnittstellen zur Komponente für die dynamische Objektidentifizierung, zwecks des Empfangens von dynamischen Hindernissen, und zur Komponente für die Objektdetektion, zwecks des Empfangens von statischen Hindernissen, aufgebaut werden. Die Kommunikationsschnittstellen sollen dabei nicht nur Hindernisse empfangen sondern auch entsprechende Anfragen senden können. Die Pfad-Planungs-Komponente soll somit eine Kollisionsverhütung realisieren.

Erweiterung des A*-Algorithmus um ein 3D Kollisionsverhütungsmodell und optional um die Beachtung von Verkehrsregeln (01.07.2018)
Das 2D Kollisionsverhütungsmodell des A*-Algorithmus nach Goodwin soll durch ein 3D Kollisionsverhütungsmodell ersetzt werden. Beim 3D Modell wird hierbei durch Beachtung des Closest Point of Approach (CPA) der Faktor der Zeit mit in der Kollisionsverhütung beachtet, wodurch ein noch sichereres und effektiveres Ausweichen von anderen Schiffen ermöglicht werden soll. Außerdem soll der A*-Algorithmus optional um die Beachtung von ausgewählten maritimen Verkehrsregeln erweitert werden, um das sichere und verkehrskonforme Befahren von Gewässern zu gewährleisten. Die beachteten Verkehrsregeln sind dabei die Beachtung von Fahrwasser, das Rechtsfahrgebot im Fahrwasser, das Überholverbot und die Geschwindigkeitsbegrenzung.

2.2.2. Hindernisidentifikation

Es soll eine neue Komponente für die Objektdetektion sowie für die Objektidentifizierung erstellt und in die Gesamtarchitektur von MATE II implementiert werden.

s100 zu NMEA2000 Transformation und Kommunikationsschnittstellen (08.03.2018)
Es soll eine neue Komponente für die dynamische Objektidentifizierung erstellt werden. Diese soll in der Lage sein, Radar- und Automatisches Identifikationssystem (AIS)-Daten sowie der Steuerkurs und die Position des eigenen Schiffs über das Polymorphic-Interface zu empfangen sowie zu verarbeiten. Damit die Nachrichten vom Polymorphic-Interface an die dynamische Objektidentifizierung gesendet werden kann, müssen Transformationsregeln von S100 Nachrichten in National Marine Electronics Association 2000 (NMEA2000) Nachrichten implementiert werden. Zudem wird eine Kommunikationsschnittstelle von der dynamischen Objektidentifizierung

zur Pfad-Planungs-Komponente benötigt, um dieser alle relevanten Informationen zu umliegenden Schiffen zu schicken. Des Weiteren muss eine Kommunikationsschnittstelle zur EPD als Kontrollfunktion aufgebaut werden.

Daten Fusion - Assoziation (12.04.2018)
Um den ersten Teil der Datenfusion umzusetzen, soll die Datenassoziation implementiert werden. Es sollen die empfangenen Daten untereinander assoziiert werden. Hierzu zählt AIS zu AIS sowie Radar zu AIS. Ziel ist es, dass eine eindeutige Zuordnung von Nachrichten zu Schiffen existiert, dies dient als Basis für den darauffolgenden Meilenstein der Datenregression.

Daten Fusion - Regression (10.05.2018)
Aufbauend auf der Datenassoziation sollen die empfangenen Daten derselben Schiffe untereinander angeglichen werden, sodass möglichst exakte Parameter der erkannten Schiffe generiert werden. Alle erkannte Schiffe sollen daraufhin an die Pfad-Planungs-Komponente versendet werden, auch wenn nur Radar- oder AIS- Tracks vorhanden sind.

Kamera (28.06.2018)
Neben Radar und AIS soll eine Kamera als weiterer Umfeldsensor integriert werden. Hierfür soll eine neue Komponente erstellt werden, welche durch maschinelles Lernen Schiffe auf den Videostream in Echtzeit detektiert. Die detektierten Objekte sollen an die dynamische Objektidentifizierung gesendet werden, um sie in die Datenfusion mit einzubinden und anschließend an die Pfad-Planungs-Komponente weiterzuleiten. Darüber hinaus soll ein Videostream des Kamerabilds mit Hervorhebung der erkannten Objekte als Kontrollfunktion an die EPD übertragen werden.

2.2.3. Regelungssystem

Das bestehende Regelungssystem soll korrigiert und weiter entwickelt werden.

Fehlersuche im Controller (15.02.2018)
Zunächst soll ein Fehler, in dem von MATE I entwickelten Regelungssystem gefunden und behoben werden. Um die Fehlerquelle effektiv eingrenzen zu können und den später, laut *PGMATE Objectives* zu entwickelnden *Model predictive Control* testen zu können, muss dieser in ein virtuelles Testfeld integriert werden.

Test im physikalischen Testfeld (01.04.2018)

Um die Parameter endgültig festlegen zu können, muss ein Test im physikalischen Testfeld vorgenommen werden. Auch kann erst hier festgestellt werden, ob der Fehler mit der im ersten Meilenstein beschriebenen Methodik behoben werden konnte.

Um einen Test im physikalischem Testfeld zu ermöglichen müssen einige Schritte vorgenommen werden.

- Navibox richtig konfigurieren
- Hardware auf Zuse aufbauen

Entwurf des Model predictive Control (01.05.2018)

Die Grundlage der *Model Predictive Control* ist ein mathematisches Modell der Zuse. Darauf aufbauend wird eine Modellprädiktive Regelung entwickelt. Das so erstellte Regelungssystem kann wieder mit der im ersten Meilenstein des Controllers entwickelten Testmethodik evaluiert werden.

Deployment des vorher entworfenen Model predictive Control (21.06.2018)

Abnahmetest des neu entwickelten und implementierten *Model Predictive Control*.

2.2.4. Integration simulierter Daten ins physikalische Testfeld

Es soll eine neue Komponente für die Verknüpfung zwischen dem physikalischen und dem simulativen Testfeld erstellt werden.

HAGGIS Ucore-XML-Output-Stream (22.03.2018)

Der Simulationsadapter soll einen XML-Datenstrom transformieren um Ecore konforme XML-Daten zu erzeugen und weiter zu geben.
Hierfür soll mittels HAGGIS der XML-Datenstrom bereit gestellt werden. Damit dies möglich ist soll ein Protocolhandler in HAGGIS implementiert werden. Dieser Protocolhandler soll über UDP einen XML-Datenstrom liefern, der alle konfigurierbaren Eigenschaften des Simulationsszenarios enthält. Die XML-Dateien mit den Eigenschaften des Simulationsszenarios sollen in festgelegten Zeitabständen gesendet werden.

Ucore zu Ecore-XML-Transformation (05.04.2018)
Das XML-Datenformat in HAGGIS ist Ucore konform und soll für die weitere Verwendung mit dem physikalischen Testfeld in Ecore konforme XML-Dateien transformiert werden. Um diese Transformation vorzunehmen und die benötigten AIS-, Radar- und Positionsnachrichten zu erstellen soll eine XSL Transformation durchgeführt werden. Hierfür sollen Transformationsregeln in Form von XSL Style Sheets erstellt werden.
Es soll zudem möglich sein unterschiedliche Konfigurationen der Kombination von transformierten Nachrichten auszuwählen.

RabbitMQ-Output (12.04.2018)
Um die Weiterverwendung der transformierten XML-Dateien zu ermöglichen soll ein RabbitMQ-Output-Adapter die Dateien an das Polymorphic-Interface schicken.

2.2.5. Küstenleitstand

Es soll eine Softwarelösung für einen küstenseitigen Leitstand gefunden und um die Anforderungen von MATE II erweitert werden.

Portierung MATE I-Plugin (23.04.2018)
Um die manuelle „Notfallsteuerung“ von MATE I weiterhin nutzen zu können, muss sie in die PG MATE II einzusetzenden Lösung zum Küstenleitstand integriert werden. Dazu ist eine Portierung der Schnittstellen, der grafischen Benutzeroberfläche und der Übertragungseinrichtungen durchzuführen.

Anzeige von Rohdaten der dynamischen Hindernisidentifikation (10.05.2018)
Um die Funktion der dynamischen Hindernisidentifikation überprüfen zu können, sollen die zur Identifikation von Hindernissen genutzten Rohdaten in der Kontrollplattform des Küstenleitstandes angezeigt werden. Visualisierung dieser Rohdaten (Koordinaten von AIS- und Radar-Datensätzen sowie fusionierter Schiffe) muss eine empfangsbereite Kommunikationsschnittstelle zur dynamischen Hindernisidentifikation implementiert werden. Zusätzlich muss eine Infrastruktur implementiert werden, um die empfangenen Daten vorhalten und im Küstenleitstand zu visualisieren.

Schnittstelle zur Routen- und Pfadplanung (31.05.2018)
Damit eine Route Hindernisse berücksichtigt und eine gefahrlose Navigation ermöglicht wird, soll eine im Küstenleitstand auswählbare Route an die Routenplanungs-Komponente übertragen werden. Zudem muss die daraufhin optimierte Route im Küstenleitstand empfangen und dargestellt werden. Hierzu ist eine Kommunikationsschnittstelle und eine Erweiterung des Küstenleitstandes zu implementieren. Analog dazu muss eine Funktion implementiert werden, die eine Route an die Pfadplanung sendet. Zudem muss ein Pfad von der Pfadplanung empfangen und im Küstenleitstand kontinuierlich aktualisiert werden.

Anzeige von Überwachungsinformationen der Pfadplanung (10.05.2018)
Zur Überwachung einer autonomen Fahrt und der berechneten Zwischenergebnisse der Teilergebnisse soll eine Schnittstelle implementiert werden, die Daten empfangen und im Küstenleitstand anzeigen kann. Darunter fallen die „GoodWin-Shapes“ und Hindernisse der Pfadplanung. Dies soll farbig voneinander unterscheidbar erfolgen. Eine entsprechende Infrastruktur zur Verwaltung der übertragenen Daten ist zu implementieren.

2.3. Werkzeuge

Zu der Projektinfrastruktur gehören effiziente Kommunikationstechniken und Werkzeuge, um die Zusammenarbeit im Team zu bewerkstelligen. Dazu zählt eine umfassende Versionsverwaltung, eine Anwendung zum Festhalten von Notizen, ein Projektmanagementtool sowie ein Kommunikationstool. Der nachfolgende Teil stellt die eingesetzten Anwendungen und Werkzeuge vor, die während der Projektarbeit von den Teammitgliedern genutzt wurden.

Git Für die Auswahl einer Software zur Versionsverwaltung wurde Git ausgewählt. Git ist eine freie Software zur verteilten Versionsverwaltung von Dateien und findet bei kleinen und großen Projekten Verwendung.

Slack Slack ist eine Software, die für die Kommunikation und Zusammenarbeit zwischen Projektteilnehmern genutzt wird. Für die Benutzung der Software kann zwischen einem webbasierten Client oder nativen Anwendungen für alle gängigen Betriebssysteme gewählt werden. Zu Beginn des Projektes wird ein sogenannter „Workspace“ erstellt dem alle Projektteilnehmer beitreten können. In diesem organisieren sich die Mitglieder untereinander selbst. Die zu nutzenden Funktionen von Slack ermöglichen die Erstellung von öffentlichen oder privaten Kanälen, sowie

die Möglichkeit einzelne Teilnehmer direkt anzuschreiben. Der Vorteil der Plattform ist, dass für die verschiedenen Teilgebiete des Projekts eigene Kanäle erstellt werden können um die Kommunikation innerhalb des Projektes besser zu strukturieren. Des Weiteren werden sogenannte Funktions-integrationen angeboten. Mit diesen kann beispielsweise die Integration von Jira oder Jenkins in Slack realisiert werden. Slack wird von der Projektgruppe in der kostenlosen Version genutzt.

Jira Jira ist eine webbasierte Projektmanagementplattform, die die Zusammenarbeit innerhalb eines Teams ermöglicht. In der Softwareentwicklung unterstützt Jira das Anforderungsmanagement, die Statusverfolgung und später den Fehlerbehebungsprozess. In der Praxis können Teammitglieder sich selbst oder gegenseitig Aufgaben zuweisen und diese bearbeiten. Auch eine umfangreiche Zeiterfassung für die Bearbeitung der Aufgaben ist integriert. In der Gruppe wird zudem ein Agile-Board genutzt, welches die einfache Modellierung der zu erledigenden Aufgaben ermöglicht. Hierbei können die Aufgaben zwischen verschiedenen Phasen verschoben werden (z.B. „in Bearbeitung“ oder „to Review“).

Confluence Confluence ist eine Wiki-Software. Die Software ermöglicht die Arbeit des Teams auf zentrale, organisierte und zugängliche Weise festzuhalten. Neben dem Vorteil einer zentralen Dokumentenverwaltung, bietet Confluence die Möglichkeit Jira einzubinden.

Skype Skype ist ein Kommunikationstool für kostenlose Anrufe und Chats. Mit Hilfe von Skype können Videokonferenzen oder einzelne Anrufe zwischen den Projektteilnehmern organisiert werden. Des Weiteren bietet die Software die Möglichkeit die Funktion der Bildschirmübertragung, mit dessen Hilfe schnell und einfach Probleme gelöst werden können ohne das sich die Mitglieder hierfür an einem Ort treffen müssen.

Enterprise Architect Die erstellten Modelle der Projektgruppe wurden mithilfe von Enterprise Architect realisiert. Enterprise Architect ist ein umfangreiches Softwaremodellierungswerkzeug dessen große Modellpalette den Ausschlag gegenüber anderen Modellierungswerkzeugen gab.

3. Theoretische Grundlagen

Im folgenden Kapitel werden einige theoretische Grundlagen und Begriffe erläutert, die zum Verständnis der sich dem Kapitel anschließenden Themen beitragen. Es wird der aktuelle Stand verwandter Forschungsprojekte zusammengefasst, einen Überblick über maritime Standards, Sensoren und Navigation gegeben, wichtige Konzepte zur Kollisionsverhütung vorgestellt sowie Grundlagen zur Regelungstechnik und Bildverarbeitung behandelt.

3.1. Stand aktueller Forschungsprojekte

In diesem Kapitel werden aktuelle Forschungsprojekte vorgestellt, die relevante Informationen für die Umsetzung eines autonomen Schiffes liefern und den aktuellen Stand der Forschung repräsentieren. Dabei wird zunächst auf die vorangegangene Arbeit und somit die vorhandene Basis dieses Projekts eingegangen. Anschließend werden vorhandene Möglichkeiten zur Routenplanung und Kollisionsverhütung bei autonomen Schiffen aus der Dokumentation der Projektgruppe *Intelligente Schiffssimulation* erläutert. Darauf folgend wird eine Kollisionserkennung, die auf die Sensorik eines autonomen Schiffes basiert aus *Marine Observation Platform for Surfaces IV* vorgestellt. Zum Abschluss wird kurz auf weitere Forschungsprojekte eingegangen.

Maritime Test- und Experimentierplattform, MATE

Das Vorgängerprojekt PG MATE I hatte die Ziele ein autonomes Schiff in drei aufeinander aufbauenden Schritten zu erstellen. Die Unterteilung wurde wie folgt umgesetzt:

Schritt 1 Fernsteuerung eines Schiffes.

Schritt 2 Automatisiertes Abfahren einer Route.

Schritt 3 Vollständig autonomes Schiff, das dynamisch auf Umwelteinflüsse reagieren kann.

Dabei lag besonderes Augenmerk auf der Umsetzung einer Schnittstelle für die dynamische Übersetzung von Nachrichtenformaten wie NMEA2000, um die Kommunikation mit anderen Sensoren, Aktuatoren und Komponenten zu ermöglichen [MAT17]. Diese Schnittstelle, genannt *Polymorphes Interface*, wird wie in Abbildung 3.1 dargestellt eingebunden.

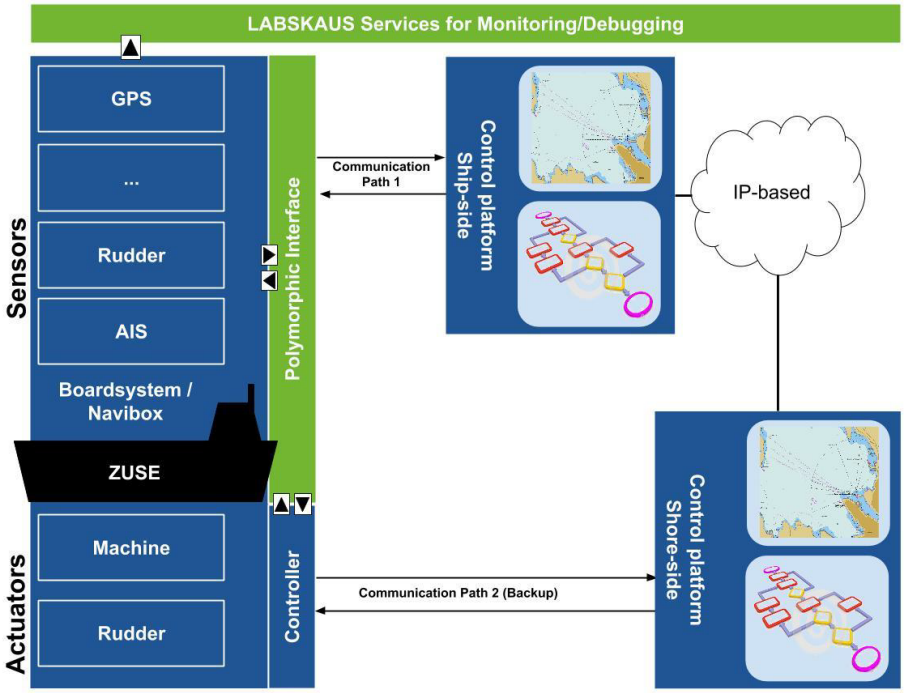


Abbildung 3.1.: Architektur des MATE I Forschungsfeldes

Damit wurden Kommunikationskanäle und Schnittstellen geschaffen, die genutzt wurden, um die Schritte 2 und 3 vollständig umsetzen zu können.

Intelligente Schiffssimulation, ISS

Aus dem Forschungsprojekt ISS, welches sich mit der Integration einer intelligenten Komponente zum Steuern von Schiffen innerhalb einer vorgegebenen Simulationsumgebung beschäftigte, konnten Algorithmen zur Kollisionsvermeidung und -verhütung abgeleitet werden, die in der autonomen Schifffahrt sehr gut genutzt werden können [ISS18]. Insbesondere wurden der A*-Algorithmus (vlg. Abschnitt 3.4.1) für die Routen- und Pfadplanung und der NoGo-Solver für die statische Hinderniserkennung (vgl. Abschnitt 5.3) genutzt. Die genaue Umsetzung und Einbettung in das MATE II Produkt wird im Laufe dieser Ausarbeitung genauer erläutert.

Marine Observation Platform for Surfaces IV, MOPS IV

Das Forschungsprojekt MOPS IV hatte den thematischen Schwerpunkt *„die vollautomatische Erkennung von Kollisionsrisiken sowie die Einleitung geeigneter Gegenmaßnahmen zur Vermeidung von Zusammenstößen“* [IV16] zu entwickeln. Daraus sind Maßnahmen in der Bildverarbeitung und somit der Kollisionserkennung und -verhütung hervorgegangen.

Entwicklung einer sicheren physischen Basisplattform zur echtzeitfähigen Steuerung für die Autonomisierung von hochseetauglichen Schiffen

Das Projekt von M.Sc. Peter Tank, welches im Rahmen einer Masterarbeit an der Carl von Ossietzky Universität Oldenburg entwickelt wurde, liefert die Grundlage für die Nutzung des autonomen Schiffes im Rahmen dieses Projekts. Der Fokus lag dabei auf *„einem sicheren physischen Plattformentwurf zur Autonomisierung von hochseetauglichen Schiffen am Beispiel des Forschungsbootes Zuse des Oldenburger Institut für Informatik (OFFIS) und der Abteilung Systemanalyse und -optimierung der Carl von Ossietzky Universität Oldenburg“*. Dabei wurden alle benötigten Sensoren, Aktoren und Schnittellen umgesetzt, die das Nutzen eines Fahrreglers zur autonomen Steuerung der Zuse ermöglichen [Tan17].

Weitere Forschungsprojekte

Neben der Universität Oldenburg, forscht die Universität in Kiel an der autonomen Schifffahrt im Forschungsprojekt SKAS - Systeme und Komponenten für autonome Schifffahrt. Das Testfeld liegt in der Kieler Förde und die Forschungen begannen 2018. Das Forschungsprojekt will *„transdisziplinär Systeme, Komponenten, Sensoren, Technologien und Kommunikationsleistungen für teil- oder vollautonome Schiffe entwickeln und erforschen“* [Paw]. Das Interesse an der Forschung steigt somit sowohl national, als auch international. So wird beispielsweise in Norwegen seit 2013 in dem Bereich geforscht. Bereits bis 2020 sollen dort autonome Schiffe in der Seefahrt eingesetzt werden. [ZEH].

In Japan gibt es ambitionierte Projekte von Technologiefirmen und Schiffbauern, in denen die Interessenten bis 2025 eine Anzahl von 250 Containerschiffe autonom und ohne Besatzung auslaufen lassen wollen [Nic].

3.2. Maritime Standards

Standards und Normen sind Ordnungsinstrumente um die Zusammenarbeit besonders im technischen Bereich zu erleichtern. Die für das MATE II-Produkt relevanten maritimen Standards werden im folgenden Abschnitt behandelt.

RTZ

Das Route Exchange Format (Route plan exchange format (RTZ)) wurde 2015 als Teil des IEC 61174 Standards der International Electrotechnical Commission eingeführt. Für elektronische Kartendarstellungs- und Informationssysteme (englisch: ECDIS) wurde mit RTZ ein einzelnes standardisiertes Format für den Austausch von Routen Daten festgelegt [Gns]. RTZ legt dabei den minimalen Inhalt und das Format der Daten fest und ermöglicht den Austausch von Routen zwischen verschiedenen Systemen (z.B. zwischen ECDIS und Radar) sowie unterschiedlichen Herstellern. Grundlegend benötigt RTZ nur einen Routennamen sowie mindestens zwei Wegpunkte. Diese Strecke wird auch Leg genannt [Tra].

Ein Beispiel:

```
<?xml version="1.0"?>
<route xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="1.0"
    xmlns="http://www.cirm.org/\gls{RTZ}/1/0">
  <routeInfo routeName="Example Route" />
  <waypoints>
    <waypoint id="0" radius="0.008">
      <position lat="53.518166666666666" lon="8.176416666666666"/>
      <leg speedMax="5.144444444444445"/>
    </waypoint>
    <waypoint id="1" radius="0.008">
      <position lat="53.51928586954941" lon="8.175095103625296"/>
      <leg speedMax="5.144444444444445"/>
    </waypoint>
  </waypoints>
</route>
```

S-100

Der S-100 wurde von der International Hydrographic Organisation (IHO) mit dem Ziel erstellt, einen allgemeinen hydrographischen Datenstandard in der maritimen Welt zu etablieren. Hierfür baut der S-100 auf seinem Vorgänger dem S-57 (Standard für elektronische Seekarten) auf und ist zusätzlich Teil der ISO-19000. Der S-100 geht über die Funktion eines Datenstandards hinaus, da mit seiner Hilfe die Spezifikationen der abzubildenden Daten definiert werden können. Somit wird der S-100 als Framework angesehen.

Das 580 Seiten schwere Dokument des S-100 beschäftigt sich mit der zu verwendenden Modellierungssprache, Datenbanken der IHO, dem grundlegenden Modell des S-100 (General Feature Model) sowie weiteren Punkten wie der Umgang mit Bild- oder Rasterdaten sowie die Nutzung von Symbolen und dem richtigen Einsatz von Encoding Formaten. Der große Vorteil des S-100 in der Zukunft sind die von der IHO bereitgestellten Register. Diese Register enthalten Datenspezifikationen die von allen Nutzern des S-100 verwendet werden können. Fehlende Datenspezifikationen können von den Nutzern erstellt und an die IHO gesendet werden, diese übernimmt die neue Datenspezifikation nach einer Überprüfung in ihre Register oder lehnt die Einsendung aufgrund von Fehlern ab [S100].

S-57

Der IHO S-57 Standard wurde zum Austausch digitaler hydrographischer Daten entwickelt. Der Standard enthält:

- Eine allgemeine Einführung
- Ein theoretisches Datenmodell auf dem der Standard fußt
- Die Datenstruktur und das Format die zur Implementierung des Datenmodells genutzt werden
- Allgemeine Regeln für die Codierung von Daten [RB, S. 1]

Zudem enthält der Standard zwei Anhänge, Anhang A ist der Objekt-Katalog. Dieser enthält das offizielle Datenschema, das zur Beschreibung von Echt-Welt-Entitäten genutzt wird. Anhang B enthält die vom IHO akzeptierten Produkt Spezifikationen. Aktuell wird nur die Produktspezifikation für elektronische Navigationskarten (Electronic Navigational Chart (ENC)) weitläufig genutzt [RB, S. 2].

NMEA0183

Der National Marine Electronics Association 0183 (NMEA0183) ist ein Standard für die Kommunikation zwischen Navigationsgeräten auf Schiffen, der von der gleichnamigen Organisation (NMEA) definiert wurde. Genauer handelt es sich um einen Schnittstellenstandard der Anforderungen an elektrische Signale, an das Datenübertragungsprotokoll und an die Datenübertragungszeit definiert. Der Standard wurde entwickelt, um die serielle unidirektionale Datenübertragung von einem einzigen Sender zu mehreren Empfängern zu unterstützen. Die Daten sind in ASCII-Zeichen auslesbar und enthalten Informationen wie Geschwindigkeit, Position und Tiefe [NM01] .

NMEA2000

National Marine Electronics Association 2000 (NMEA2000) ist ein serielles Netzwerk zur Datenübertragung. Es erlaubt mehreren elektronischen Geräten sich auf einem Kanal miteinander zu verbinden um Daten untereinander auszutauschen. Durch den Netzwerkaufbau ist ein umfassendes Regelwerk von Nöten um einen reibungslosen Austausch zwischen den Netzwerkteilnehmern zu gewährleisten. Diese Regeln werden im NMEA2000 Standard durch den Controller Area Network (CAN) Standard abgedeckt. Wie auch sein Vorgänger National Marine Electronics Association 0183 (NMEA0183) definiert NMEA2000 standardisierte Datenformate und -definitionen.

Zusätzlich verfügt NMEA2000 über ausführliche Regeln für das Netzwerkmanagement, um Knotenpunkte zu identifizieren, Kommandos an Geräte zu senden und Daten abzufragen. NMEA2000 bietet die Möglichkeit bis zu 50 Geräte einzubinden, diese kommunizieren mit 250 kbit/s, was 50-mal schneller als NMEA0183 ist, über eine Entfernung von bis zu 200 Metern [NM20].

International Regulations for Preventing Collisions at Sea

Die *COLREGs* wurden 1972 von der International Maritime Organization veröffentlicht und enthalten unter anderem die Verkehrs- oder die Navigationsregeln für Schiffe auf See um Kollisionen zwischen zwei oder mehreren Schiffen zu verhindern [IMO]. Die COLREGs bestehen aus insgesamt 41 Regeln, aufgeteilt in sechs Kapitel (A bis F). Die für diese Arbeit relevanten Regeln 13 bis 17 stammen alle aus dem Kapitel B, welches näher auf Steuerung und Navigation des Schiffes auf See eingeht, und werden im Folgendem vorgestellt. [Coc11]

Regel 13: „Overtake“

Das überholende Schiff muss dem vor ihm fahrenden Schiff ausweichen.

Regel 14: „Head-On“

Bei einer frontalen Begegnung mit einem anderen Schiff haben beide Schiffe nach Steuerbord auszuweichen.

Regel 15: „Crossing“

Bei einer Begegnung, bei der zwei Schiffe sich kreuzen muss das Schiff ausweichen, welches das Andere zu Steuerbord hat.

Regel 16: „Crossing - Give-Way“

Das nach Regel 15 ausweichpflichtige Schiff muss frühzeitig und durchgreifend handeln.

Regel 17: „Crossing - Stand-On“

Das Schiff, welches nach Regel 15 den Kurs beibehält, hat seinen aktuellen Kurs und seine Geschwindigkeit beizubehalten. Kommt das ausweichpflichtige Schiff seiner Pflicht jedoch nicht nach, muss der Kurshalter selbst manövrieren.

3.3. Maritime Navigation

Das Kapitel beinhaltet Grundlagen zur Navigation, wie elektronische Navigationssysteme, die darin verwendeten Seekarten und die Geography Markup Language.

ENC

Elektronische See- bzw. Navigationskarten die die Anforderungen des IHO-Standards S-57 erfüllen werden als Electronic Navigational Chart (ENC) bezeichnet. Unter die Anforderungen fallen alle Kartenmerkmale für sichere Navigation wie Küstenlinien, Bathymetrie, Bojen und Lichter. ENCs basieren auf Vektorgrafiken und ihre Visualisierung in ECDIS wird durch den IHO-Standard S-52 definiert [ENC, S. 3].

ECDIS

ECDIS ist die Abkürzung für Electronic Chart Display and Information System, was ins Deutsche übersetzt elektronisches Kartendarstellungs- und Informationssystem bedeutet. Bei ECDIS handelt es sich um ein elektronisches Navigationssystem, das den Vorschriften der International

Maritime Organization (IMO) entspricht. Dabei vereint es die Anzeige elektronischer Navigationskarten (ENC), objektbezogene Informationen wie zum Beispiel Schifffahrtzeichen und integriert Sensordaten wie die Positionsangaben des Global Positioning System (GPS) und anderen Navigationssensoren wie Echolot, Radar und AIS [ECI].

Die interne Datenbasis eines ECDIS wird als System Electronic Navigational Chart (SENC) bezeichnet. Sie setzt sich aus der ENC-Datenbasis, den Updates zur ENC und anderen, vom Benutzer hinzugefügten Daten zusammen [SENC]. ENCs sind Vektorgrafiken die die Anforderungen des S-57 Standards der International Hydrographic Organisation (IHO) erfüllen. Weiterhin ist S-57 Bestandteil des IMO Leistung-Standards für ECDIS.

Geography Markup Language

Geography Markup Language (GML) ist eine Auszeichnungssprache, die durch XML erzeugt wird. Sie wird zum Austausch raumbezogener Objekte (Features) eingesetzt. GML dient sowohl als Modellierungssprache für geografische Systeme wie auch als offenes Austauschformat für räumliche Transaktionen über das Internet. GML wird vom Open Geospatial Consortium (OGC) gemeinsam mit dem ISO TC 211, dem technischen Komitee der ISO zur Festlegung digitaler geobezogener Daten, festgelegt und existiert als ISO Standard (ISO 19136:2007) [GML].

3.4. Kollisionsverhütungskonzepte

Im Folgenden werden die für das MATE II Produkt genutzten Kollisionsverhütungskonzepte erläutert.

3.4.1. A*-Algorithmus

Mittels des A*-Algorithmus kann der kostengünstigste Pfad zwischen zwei Knoten, bspw. zwischen einem Start- und einem Zielknoten, innerhalb eines gewichteten Suchbaums ermittelt werden. Anwendbar ist er jedoch auf Suchprobleme, bei denen während der Suche bereits plausible Annahmen über die verbliebene Distanz zum Erreichen des Zielknotens getroffen werden können.[AS10]

Während des Expandierens durch die Knoten des Suchbaums werden die Wegkosten für jeden Knoten mittels der Funktion $f(v) = h(v)+g(v)$ berechnet. Die Funktion $h(v)$ ist gleich die heuristische Distanz (bspw. Manhattan, Euclidean oder Chebyshev) des aktuell untersuchten Knotens bis zum Zielkonten. Die Länge des bereits zurückgelegten Pfades vom Startknoten bis zum aktuellen Knoten wird durch $g(v)$ zurückgegeben. [F+14]

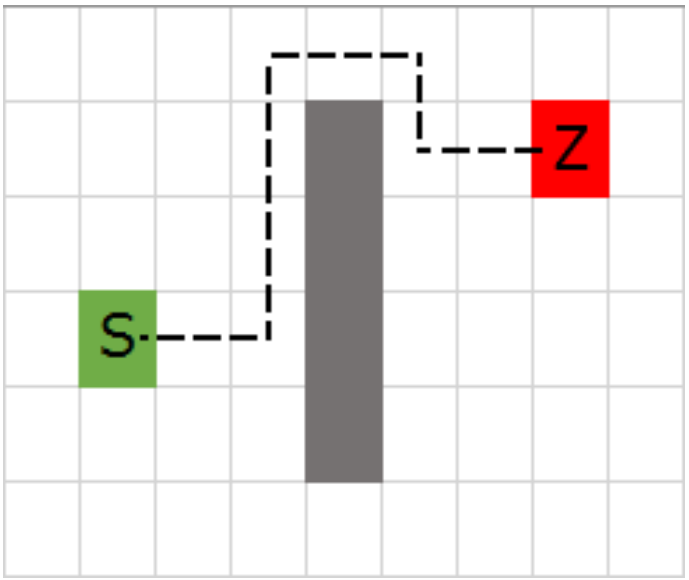


Abbildung 3.2.: Günstigster Pfad

Das Expandieren durch den Suchbaum läuft dabei folgendermaßen ab: Angefangen wird beim Startknoten, darauf folgend werden alle Nachbarknoten betrachtet. Von diesen Nachbarknoten wird derjenige expandiert, dessen Kosten $f(v)$ am geringsten ist. Dies wiederholt sich so lange, bis der Zielknoten erreicht ist. Wenn neben den Kosten für jeden Knoten auch immer der Vorgängerknoten abgespeichert wird, sind letztlich sowohl die Kosten des günstigen Pfades, als auch der günstigste Pfad an sich bekannt. [AS10]

Die programmtechnische Umsetzung entspricht dabei dem Pseudocode Algorithmus 1, wobei die OPEN Liste alle zum aktuellen Zeitpunkt bekannten Knoten und die CLOSED Liste alle abgeschlossen untersuchten Knoten, zu den somit der günstigste Weg bereits bekannt ist, enthält.

Das Ergebnis könnte bspw. wie in Abbildung 3.2 aussehen, wobei S den Startknoten und Z den Zielkonten darstellt. Die gestrichelte Line zeigt den Pfad auf [AS10].

Der kostengünstigste Pfad kann dabei den kürzesten, schnellsten, etc. Pfad widerspiegeln. Dies ist einzig abhängig von der Gewichtungsfunktion $f(v)$. Die Möglichkeit durch die Anpassung

```
1  Generiere eine OPEN Liste;
2  Generiere eine CLOSED Liste;
3  Generiere einen Startknoten und nenne ihn STARTKNOTEN;
4  Generiere einen Zielknoten und nenne ihn ZIELKNOTEN;
5  Füge den Startknoten der OPEN Liste hinzu;
6  while Zielknoten ist nicht erreicht do
7      foreach Knoten do
8          Setze einen Zeiger auf den PARENT-Knoten;
9          Schätze g(v);
10         Berechne die Kosten nach der Formel f(v) = h(v)+g(v);
11         if Knoten bereits in der OPEN Liste and Kosten f(v) geringer oder gleich hoch then
12             verwerfe den neuen Knoten;
13         else if Knoten bereits in der CLOSED Liste and Kosten f(v) geringer oder gleich hoch
            then
14             verwerfe den neuen Knoten;
15         else
16             lösche alle gleichen Knoten aus der OPEN- und der CLOSED Liste;
17         end
18         Füge den aktuellen Knoten der OPEN Liste hinzu;
19     end
20 end
```

Algorithmus 1 : (nach [AS10])

der Gewichtungsfunktion auch Aspekte wie Energieverbrauch, Sicherheit oder Anzahl der Richtungswechsel berücksichtigen zu können, ist ein großer Vorteil des A*-Algorithmus. [F+14]

3.4.2. Occupancy Grid

Ein Occupancy Grid ist ein in der Bodenebene liegendes meist zwei dimensionales Gitter welches zur Approximation der Umgebung des Schiffes dient. Jede Zelle enthält dabei die Wahrscheinlichkeit ob sie durch ein Hindernis belegt ist. Umso höher die Anzahl der Zellen bei immer kleiner werden Zellmaßen wird, umso höher ist die Umgebung aufgelöst. Die Zellen haben dabei alle die selben Maße. In der Regel ist ein Occupancy Grid immer auf das Fahrzeug, das Umfeld und das Einsatzszenario, insbesondere aus Gründen der Effizienz, maßgeschneidert. [C13]

Das Schiff befindet sich immer an einem fixen Punkt innerhalb des Occupancy Grids, bewegt sich das Schiff, so bewegt sich auch das Occupancy Grid mit [A08]. Die Abbildung 3.3 zeigt ein beispielhaftes Occupancy Grid.

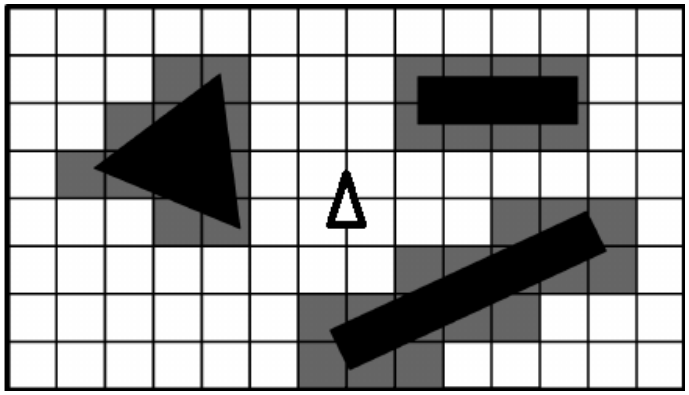
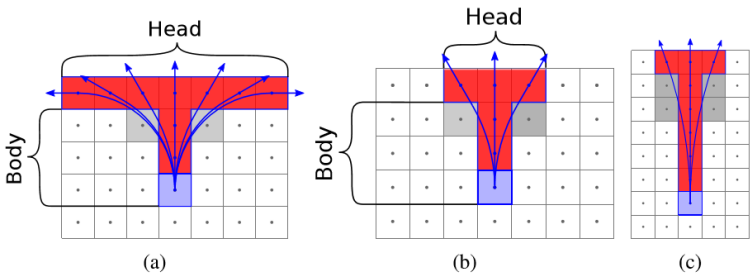


Abbildung 3.3.: Occupancy Grid (nach [F06])

Der Pfeil in der Mitte stellt dabei das Schiff dar, die schwarzen Polygone die Hindernisse. Die graue Färbung einer Zelle repräsentiert die Wahrscheinlichkeit der Blockierung dieser.

3.4.3. T-Neighbourhood

Durch das T-Neighbourhood-Konzept werden die physikalischen Eigenschaften des Schiffes bei der Kollisionsverhütung bzw. bei der Pfad-Planung berücksichtigt. Das T-Neighbourhood-Konzept definiert in einer Schiffsnavigation mit konstanter Geschwindigkeit, welche Zellen des Occupancy-Grids für das eigene Schiff erreichbar sind, wobei hierbei der Wendekreis des Schiffs beachtet wird. Die durch das T-Neighbourhood-Konzept erhaltenen validen Nachbarzellen bilden die Form des Buchstabens „T“, weshalb das Konzept auch T-Neighbourhood genannt wird.



Examples for T-Neighbourhoods corresponding to a turning circle of six time the cell size (a), nine times the cell size (b) and 36 time the cell size (c).

Abbildung 3.4.: T-Neighbourhood

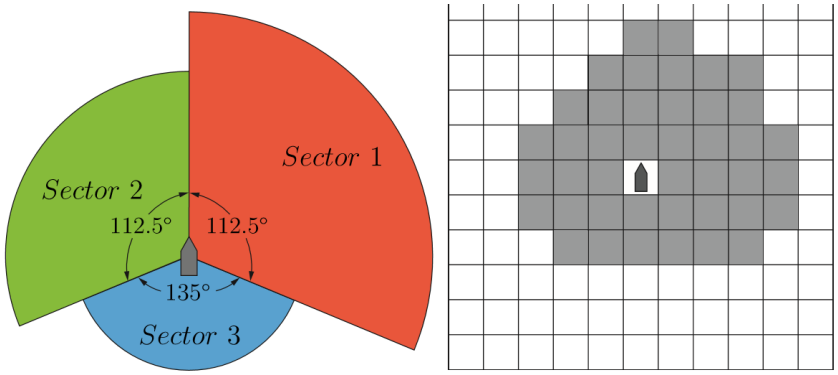
Beispiele des T-Neighbourhood-Konzeptes werden in der nachfolgenden Abbildung Abbildung 3.4 dargestellt. Die Zelle in der sich das eigene Schiff aktuell befindet ist blau, während alle erreich-

baren Zellen in rot dargestellt werden. Des Weiteren sind die erreichbaren Zellen, also das T, in Head- und Body-Zellen unterteilt. Ein Beispiel für ein Schiff mit einem Wendekreis von neun Mal der Zellgröße wird in Abbildung Abbildung 3.4 gezeigt. Das T besteht hier aus drei Body-Zellen und drei Head-Zellen. [Bla16]

Bei der Verwendung des T-Neighbourhood-Konzepts in einem Kollisionsverhütungssystem ist darauf zu achten, dass sowohl alle Body-Zellen, als auch die jeweilige gewünschte Head-Zelle erreichbar bzw. kollisionsfrei ist. Es kann ebenfalls der Fall auftreten, dass durch das T-Neighbourhood-Konzept Zellen gekreuzt, also im maritimen Zusammenhang durchfahren werden, ohne das deren Mittelpunkt erreicht wird. Diese Zellen sind in der Abbildung Abbildung 3.4 grau abgebildet und müssen ebenfalls auf Kollisionsfreiheit geprüft werden. [Bla16]

3.4.4. Goodwin

Um bei der Kollisionsverhütung bzw. bei der Pfad-Planung die COLREGs zu beachten, müssen Fremdschiffe als dynamische Hindernisse gesondert betrachtet werden, da die Umfahrung eines Fremdschiffes möglicherweise gesonderte Regeln mit sich zieht (siehe Kapitel Abschnitt 3.2). Ein Realisierung zur Präsentation der Fremdschiff-Domäne stammt dabei von Goodwin ([Bla12]). Dieser Ansatz nutzt dabei eine nicht symmetrische Fremdschiff-Domäne, welche sich in drei unterschiedliche Sektoren einteilen lässt (siehe Abbildung 3.5). Jeder Sektor definiert einen bestimmten Sicherheitsabstand zum Fremdschiff, um den Gefahrenbereich einer Kollision darzustellen.



The ship domain for safety distances presented by Goodwin (1975) and its grid representation.

Abbildung 3.5.: Goodwin

Sektor eins (rot) definiert den größten Sektor und gibt den Sicherheitsabstand vom Bug zur Steuerbord Heckseite an. Sektor zwei (grün) ist kleiner als Sektor eins und gibt den Sicherheitsabstand vom Bug zur Backbord Heckseite an. Der kleinste Sektor drei (blau) definiert schließlich den Sicherheitsabstand an der Heckseite. Dieser Sicherheitsabstand wird bei der Kollisionsverhütung bei allen erkannten Fremdschiffen berücksichtigt. Genauer heißt dies, dass alle Zellen des Grids, welche von einem Sektor überdeckt werden, blockiert werden. Durch die drei unterschiedlich großen Sektoren und der Beachtung dieser in der Kollisionsverhütung werden nun die in dieser Arbeit ausschlaggebenden COLREGs beachtet. In einer Head-On Situation zum Beispiel würde ein A*-Algorithmus berechnen, dass ein Ausweichmanöver auf Steuerbord-Seite deutlich kürzer wäre als auf der Backbord-Seite. Das Schiff würde dem Fremdschiff folglich auf der Steuerbord-Seite ausweichen, was COLREGs konform ist. [Bla12]

3.4.5. Closest Point of Approach (CPA)

Der Ort der dichtesten Annäherung (eng. Closest Point of Approach) beschreibt den Punkt, an dem die Distanz zweier Schiffe zueinander minimal ist. Zur Ermittlung des Punktes wird dabei angenommen, dass die Kurse und Geschwindigkeiten der zwei Schiffe konstant bleiben [CPA8]. Der CPA wird oft in der Kollisionsvermeidung im Zusammenhang mit der Risikobewertung angewandt. Bereitgestellt wird der CPA vom Automatic Radar Plotting Aids (ARPA) [LZ+16]. Die Abbildung 3.6 veranschaulicht den CPA sowie die Zeit zum Ort der dichtesten Annährung (TCPA).

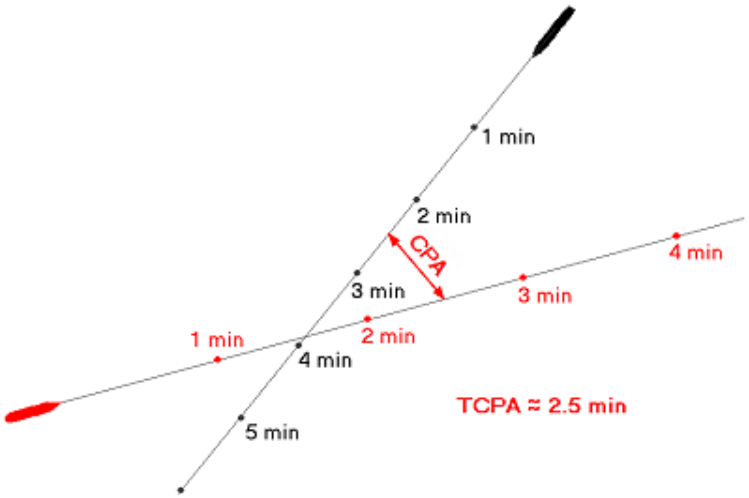


Abbildung 3.6.: CPA [CPA8]

In der Abbildung ist zu erkennen, dass sich die Distanz der beiden Schiffe zueinander für ca. 2,5 Minuten verringert, bis sie sich schließlich wieder vergrößert. Somit beträgt die TCPA 2,5 Minuten und der CPA enthält als Wert die minimale Distanz. [CPA8]

3.5. Maritime Sensoren

Die maritimen Sensoren gehören zur maschinellen Wahrnehmung des physikalischen Testfelds und haben die Aufgabe alle relevanten Verkehrsteilnehmer und die maritime Verkehrsinfrastruktur korrekt wahrzunehmen. Dadurch werden Informationen zur Interpretation der aktuellen Verkehrslage geliefert [Mau+15, S. 422].
Im Folgenden werden die für das MATE II Produkt genutzten Sensoren des physikalischen Testfelds erläutert.

3.5.1. AIS

Der Sensor für das Automatische Schiffsidentifizierungssystem (AIS) empfängt und sendet Daten zwischen Schiffen untereinander, und zwischen Schiffen und Basisstationen. Es dient primär der Kollisionsverhütung und somit der Sicherheit und Regelung des Schiffsverkehrs. Dabei werden statische, dynamische und reisebezogene Daten automatisch und kontinuierlich ausgetauscht.

Zu den dynamischen Schiffsdaten gehören z.B. Kurs und Fahrt über Grund, Position mit Zeitpunkt, gesteuerter Kurs (Heading) und Wendegeschwindigkeit. Statische Schiffsdaten sind z.B. MMSI-Nummer, IMO-Nummer, Rufzeichen, Schiffsnamen und Schiffstyp. Unter reisebezogene Daten fallen z.B. das Reiseziel mit Ankunftszeit, der Tiefgang und die Gefahrengutklasse der Ladung.[Sch16, S. 52], [BH10, S. 209], [Ove16, S. 65]

Die Daten werden anschließend von einem AIS-Anzeigegerät oder einer elektronischen Seekarte empfangen und symbolisch inklusive aktueller Position und dem gegenwärtigen Kurs dargestellt[Sch16].
Es gibt in der IMO definierte AIS Standards. Dabei unterscheiden sich grundlegend die Sende- und Empfängerklassen A und B. Klasse A Transceiver werden bei Ausrüstungspflichtgen Schiffen verwendet. Darunter fallen alle Fahrgastschiffe, Fischereifahrzeuge ab 24 Meter Länge und alle anderen Fahrzeuge ab einer bestimmten Bruttoraumzahl. Mit einem Klasse B Transceiver werden kleinere Schiffe wie z.B. Sportboote und kleinere Fischerboote ausgerüstet. Sie haben technisch eine niedrigere Signalstärke und Senderate [BWN14, S. 258 f.].

Weitere Einsatzgebiete für AIS-Geräte sind Verkehrsschilder, Sicherheitsmeldungen und Warnungen. AIS hat den Vorteil, dass Geräte, die optisch oder via Radar nicht identifizierbar sind, trotzdem identifiziert werden können, indem der Standort und ihre Bewegungen beobachtbar sind [Ove16, S. 65]. Der Nachteil ist, dass nicht alle Schiffe AIS-Ausrüstungspflichtig sind und der Schiffsverkehr somit nicht vollständig dargestellt wird.

3.5.2. Radar

Das Radar soll der Navigation und Kollisionsverhütung assistieren, indem es die Positionen anderer Schiffe mithilfe von Erkennungs- und Ortungsverfahren, in Relation zur Eigenen darstellt. Dies geschieht mithilfe von elektromagnetischen Wellen innerhalb der Radiofrequenz (Funkwellen). Es wird ein Primärsignal gebündelt zu Objekten innerhalb der Reichweite geschickt, die eine Reflektion zurückwerfen und vom Radar-Gerät als Sekundärsignal empfangen werden. Diese werden anschließend ausgewertet und als Echos erkannt.
Die ausgesendeten elektromagnetischen Wellen bilden Felder, innerhalb derer die Objekte erkannt werden. Um eine Richtungserkennung zu ermöglichen, werden die Antennen im Radar Halbmondförmig angeordnet (siehe Abbildung 3.7). Informationen zu Entfernung, relativen Bewegung und Geschwindigkeit des Objekts liefert die Auswertung der elektromagnetischen Wellen unter Zuhilfenahme des Dopplereffekts [Bal].

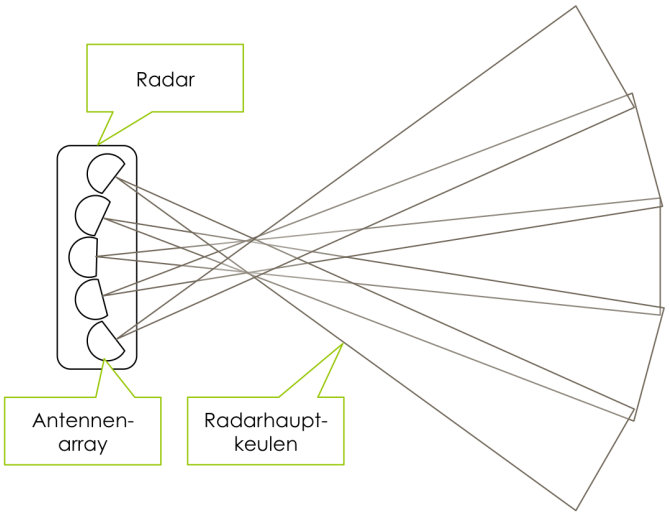


Abbildung 3.7.: Draufsicht eines schematischen Radar-Sensor Aufbaus [Bal]

Das Radar wird meist mit zusätzlichen Assistenzsystemen unterstützt, die Schiffe zum Beispiel mit einem Echo identifizieren, Fehler erkennen, um diese nicht als Objekte darzustellen oder über Kollisionskurse informieren.

Ein Beispiel für ein Radarsystem ist das Automatic Radar Plotting Aid (ARPA). Dieses hilft beim Auswerten der eingehenden Radar Daten. Eine Aufgabe von ARPA ist das Identifizieren und Verfolgen von anderen Schiffen, die automatische Anpassung der Radarparameter, wie beispielsweise die Reichweite [BWN14, S. 407 f.]. Beispiele für weitere Radar-Tools, die die Geschwindigkeit, den Kurs und die Entfernung anderer Schiffe ermitteln, wären die Electronic Bearing Line und der Variable Range Marker [BWN14, S. 297 f.].

3.5.3. Kamera

Für die visuelle Wahrnehmung des physikalischen Testfelds wird eine Kamera verwendet, die einen Mono-Video Daten-Stream erstellt. Dabei wird ein zweidimensionales Abbild der dreidimensionalen Umwelt gemacht und als hochauflösendes Farbbild dargestellt. Im Abbild werden anschließend mithilfe von Algorithmen Kanten gesucht, die zum Beispiel auf andere Schiffe oder sonstige Hindernisse hinweisen [Mau+15, S. 422]. Mithilfe dieser Daten können Objekte erkannt werden, die sonst von anderen Sensoren nicht wahrgenommen werden können. Die Kamera hat durch ihren Aufbau (siehe Abbildung 3.8) Einschränkungen.

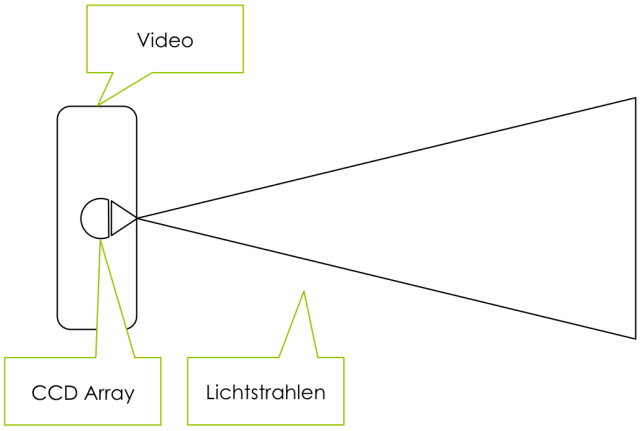


Abbildung 3.8.: Draufsicht eines schematischen Kamera-Sensor Aufbaus [Bal]

Die Lichtstrahlen von einer Kamera werden in nur eine Richtung gesendet. Dadurch ist das Abbild der Umwelt eingeschränkt. Außerdem sind Aussagen über die Entfernung eines Objektes meist fehlerhaft. Dennoch können Aussagen zur Winkelauflösung gemacht werden, die zu erkennen geben, ob ein Objekt links, rechts oder vor dem physikalischen Testfeld liegen [Bal].

3.5.4. LIDAR

Light Detection and Ranging (Lidar) bezeichnet Verfahren, die über die Laufzeit und die Lichtgeschwindigkeit Entfernungen von Objekten bestimmen und steht für Light Detection and Ranging. Dafür werden nicht bewegliche Dioden verwendet(vgl. Abbildung 3.9). Dieser Aufbau ist auch in alle Richtungen (360°) möglich [Bal].

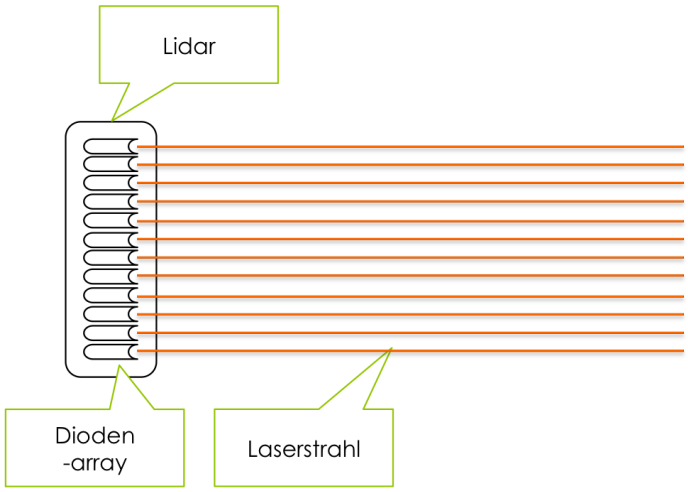


Abbildung 3.9.: Schematischer Aufbau eines Lidar-Sensors [Bal]

3.5.5. DGPS

Die eigene Position des physikalischen Testfeldes wird durch das Differenzial Global Positioning System (DGPS) bestimmt. Dieses System nutzt die Positionsdaten des Global Navigation Satellite System (GNNS). Diese Daten können aus unterschiedlichen Quellen stammen, meist jedoch vom Global Position System (GPS). Ein satellitengestütztes Navigationssystem basiert auf Laufzeitdifferenzen zwischen den Signalen der Satelliten und dem Empfänger. Unterschiede in der Entfernung mehrerer Satelliten sorgen für unterschiedliche Signalempfangszeiten. Es wird angenommen, dass diese Unterschiede proportional verlaufen.

In der Realität kommt es jedoch zu Störungen und Abweichungen in den Zeiten. Diese Fehler werden grundsätzlich korrigiert, dennoch kann es zu Ungenauigkeiten kommen. Deshalb werden in der Schifffahrt oft mehrere GPS-Empfänger und Referenzstationen genutzt. Die Postionen der Referenzstationen sind geodätisch exakt bekannt. Aus der empfangenen Position des Satellitennavigationssystems und der GPS-Position der Referenzstation errechnet diese den Fehler zwischen Referenzstation und Satellitennavigationssystem und schickt diese Abweichung an die DGPS-Empfänger in der Umgebung (siehe Abbildung 3.10) [BH10, S. 107 f.] [BWN14, S. 437 f.]. Darin enthalten sind neben der Abweichung auch Integritätsinformationen, die dem Nutzer zeitnahe Warnungen liefern können, z.B. wenn das DGPS-System Fehler aufweist und nicht mehr genutzt werden darf [Hop]. So können Genauigkeiten von unter einem Meter bis zu zehn Zentimeter erreicht werden [DIF].

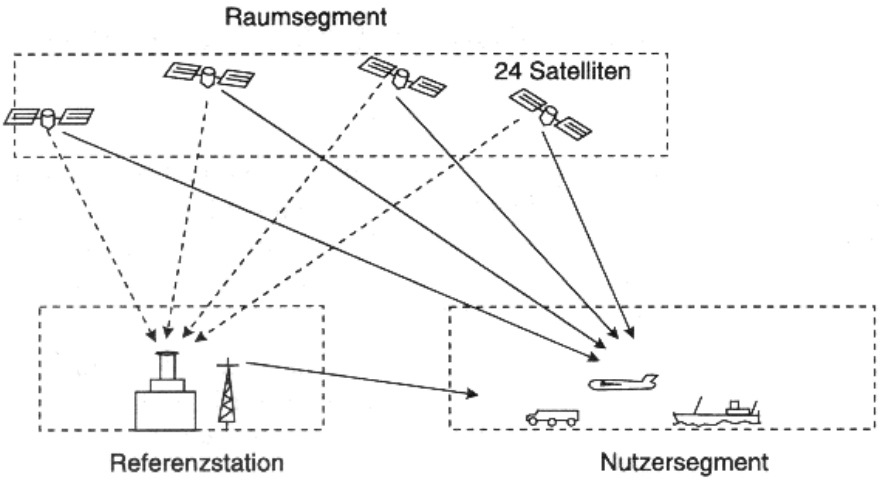


Abbildung 3.10.: Schematischer Aufbau eines DGPS-Systems [Man09, S. 345]

3.5.6. Motorsteuerungssystem

Das Motorsteuerungssystem ist die Schnittstelle, die die elektronische Steuerung mit dem physikalischen Testfeld verbindet. Neben der Steuerung übernimmt das System die Darstellung der Motorsteuerungsfunktionen, liefert Schnittstellen für die Sensorik und Aktuatorik der Motorsteuerung, als auch für die Kommunikation mit anderen internen Steuerungs- und Regelungssystemen. Dabei liegt die zentrale Bedeutung bei der Fehlererkennung und Diagnose des Motors [Ise10].

3.5.7. Ruderlagenanzeige

Die Ruderlagenanzeige zeigt den aktuellen Ausschlag des Ruders in Grad an. Die physikalische Anzeige befindet sich meist auf der Kommandobrücke oder im Steuerhaus [SAb]. Auf dem physikalischen Testfeld werden zusätzlich Daten an das Motorsteuerungssystem gesendet.

3.5.8. Ausstattung des physikalischen Testfelds

In der folgenden Tabelle wird die genaue genutzte Ausstattung des physikalischen Testfelds aufgeführt.

Sensor	**Modell**
AIS	Navico NAIS-400
Radar	Navico Broadband 4G Radar
DGPS	JRC JLR-21
Kamera	Blackmagic Micro Studio Camera 4K
Rudderindicator Autopilot	Raymarine Autopilot EV200
Rudderindicator & Engine RPM	über Steuergerät Volvo Penta EVC D4-225

3.6. Bildverarbeitung

Die kamerabasierte Hinderniserkennung basiert im MATE II Produkt auf verschiedenen Ansätzen zur Bildverarbeitung. Zum einen wird im Folgenden der Algorithmus zur Horizonterkennung vorgestellt. Darüber hinaus ist das maschinelle Lernen für die Bildverarbeitung relevant.

3.6.1. Horizonterkennung

Die Horizonterkennung ist sinnvoll, um alles oberhalb des Horizonts für die weitere Bildverarbeitung nicht weiter zu betrachten. Nur Objekte die sich auf der Wasseroberfläche befinden, sind für die Kollisionsverhütung relevant, nicht aber Objekte auf dem Land.

Da der Horizont in einem Bild einer Linie gleicht, kann ein Algorithmus zur Liniendetektion eingesetzt werden. Hier eignet sich zum Beispiel die Hough-Transformation. Diese Transformation wird zum Detektieren gerader Linien verwendet. Ziele sind das Finden von vorgegebenen geometrischen Strukturen. Hierbei ist der die Transformation robust gegenüber Rauschen und systematischen Fehlern. Es werden auch unvollständige (z.B. teilweise verdeckte) Linien erkannt. [Hub] Eine genauere Beschreibung dieses Algorithmus' kann der Quelle [Opea] entnommen werden.

3.6.2. Maschinelles Lernen

Eine weitere Herangehensweise der Bildverarbeitung ist das maschinelle Lernen. Dies bietet den Vorteil, dass nicht nur angegeben wird, ob sich ein Objekt im Bild befindet, sondern das Objekt zusätzlich identifiziert wird. Zudem ermöglichen es Methoden des maschinellen Lernens Parameter automatisch aus gegebenen Daten zu lernen.

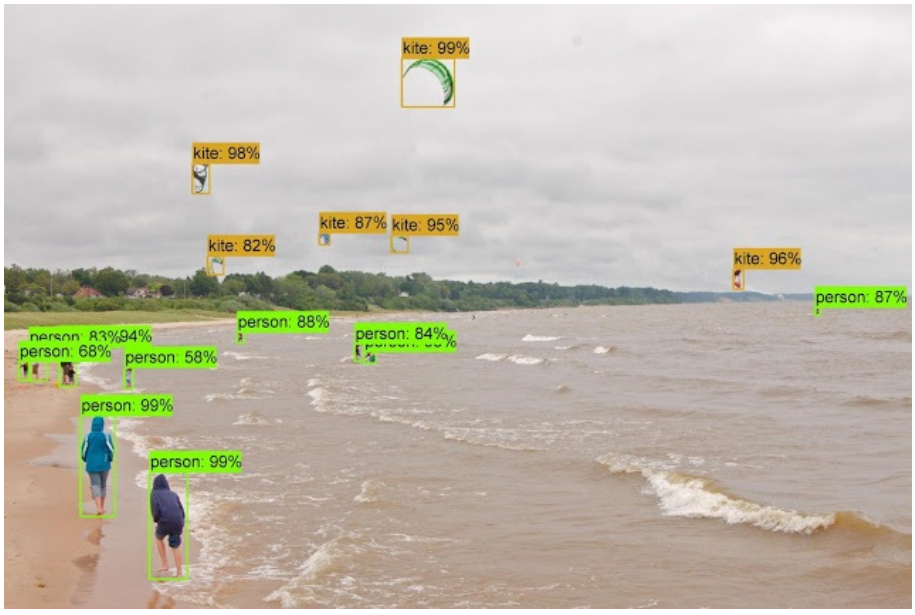


Abbildung 3.11.: Beispielbild einer Objekterkennung mittels Tensorflow [Zop+17]

Google hat beispielsweise die Open-Source Bibliothek „Tensorflow“ entwickelt, welche es ermöglicht, ein künstliches neuronales Netz zu erzeugen, das mittels maschinellem Lernen zuerst trainiert werden kann und dadurch später die gewünschten Objekte im Bild erkennt. Ein Beispiel hierfür ist in Abbildung 3.11 zu sehen. Die Objekte werden durch ein Rechteck markiert. Neben der Art des Objekts wird auch die Wahrscheinlichkeit, dass es sich um das angegebene Objekt handelt, angegeben.

3.7. Sensordatenfusion

Um die im MATE II Produkt verwendeten Sensoren zur Hindernisidentifikation sinnvoll zu nutzen, wird eine Sensordatenfusion von Umfeldsensoren umgesetzt, diese wird im Folgenden kurz erläutert.

„Data fusion is the process of combining data information to estimate or predict entity states.“ [SB08]
Im Gegensatz zur Sensorfusion verwendet die Sensordatenfusion bereits verarbeitete und ausgewertete Informationen der Sensoren über die Umgebung und der gefundenen Objekten [Rei14, S.340]. Der grobe Ablauf der Sensordatenfusion ist auf Abbildung 3.12 zu sehen.

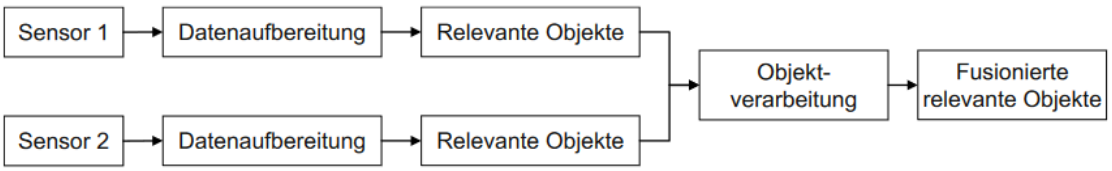


Abbildung 3.12.: Sensordatenfusion [Rei14]

Ziele der Datenfusion ist es die Stärken der jeweiligen Sensoren zu kombinieren und die Schwächen zu minimieren. Die Datenfusion erhöht die Genauigkeit, erzeugt mehr Details zu den gefundenen Objekten, plausibilisiert die Daten und erhöht die Anzahl der gefundenen Objekte [Rei14], [WHW09].

Der Ablauf zur Positionsfusion verschiedener Umgebungssensoren ist wie folgt: Zuerst erfolgt eine Datenassoziation bei dem verschiedene Tracks unterschiedlicher Sensoren zu einem Objekt zusammengefasst werden. Als nächstes sollen möglichst korrekte Schätzwerte für alle relevanten Daten erlangt werden. Hierfür können verschiedene Estimationstechniken wie der Kalman-Filter angewendet werden. [Kla06] Dieser Schritt kann als Datenregression bezeichnet werden, da die Datenwerte untereinander angeglichen werden.

3.8. Regelungstechnik

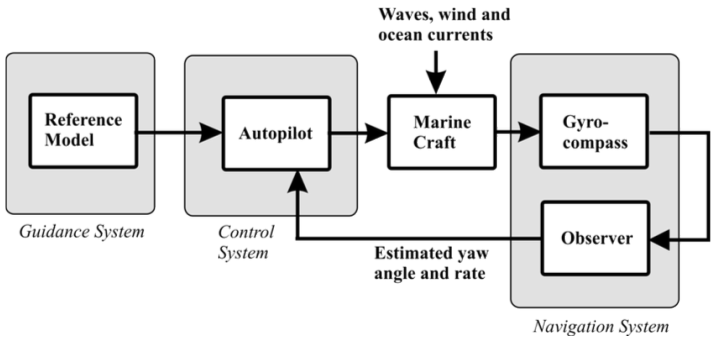


Abbildung 3.13.: Regelkreis eines Autopilots
([Fos11, Figure 9.3])

Bei einer Regelungstechnik handelt es sich um ein System zur Beeinflussung physikalischer Größen. Im Kontext eines maritimen Autopiloten sind diese Größen die Geschwindigkeit und die Ausrichtung des Schiffes.

Bei der Betrachtung von Reglern werden zur klaren Unterscheidung nach [TR15] folgende Begriffe genutzt:

Regelgröße Dies ist der Sollwert, der über die Zeit konstant gehalten werden soll.

Rückführgröße Dies ist die gemessene Größe, der Istwert.

Regeldifferenz Der Fehlerwert zwischen Regelgröße und Rückführgröße.

Stellgröße Gibt den Wert an, auf den die Aktuatoren gesetzt werden.

Störgrößen Einflüsse die von außerhalb auf das zu regelnde System einwirken.

PID-Regler

PID-Regler gehören in der Regelungstechnik zu den am weitesten verbreiteten Reglern. Diese zeichnet dabei vor allem aus, dass sie bei einer vergleichsweise einfachen Implementierung sehr gute Ergebnisse liefern.

PID steht dabei für *Proportional–Integral–Derivative*. Das bedeutet, die Funktion ist aus drei Teilen aufgebaut (nach [Orl13] und [TR15]).

P-Anteil Der proportionale Anteil. Die Regeldifferenz $e(t)$ wird mit einem konstanten Faktor K_p multipliziert. Bei einer Abweichung von $e(t)$ nimmt die Stellgröße $u(t)$ in der nächsten Stellperiode den Wert $u(t) = K_p * e(t)$ an.

I-Anteil Der integrale Anteil. Dieser integriert die Regeldifferenz $e(t)$ über die Zeit auf. Dies führt zu einer über die Zeit wachsenden Stellgröße. Damit handelt es sich hierbei um einen langsamen Regler.

D-Anteil Der differenzierende Anteil. Dieser reagiert nicht auf die Größe der Regeldifferenz sondern nur auf die Änderungsgeschwindigkeit dieser. Dieses Glied wird nicht zur Regelung selbst verwendet

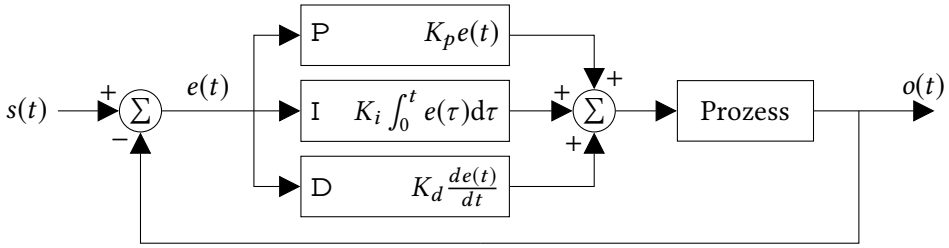


Abbildung 3.14.: Schematische Darstellung eines PID-Reglers

Diese drei Teile werden insgesamt nun zu folgender Funktion für die Stellgröße $o(t)$ addiert:

$$o(t) = K_p e(t) + K_i \int_0^t e(\tau)\,\mathrm{d}\tau + K_d \frac{de(t)}{dt}$$

Dies ist in Abbildung 3.14 schematisch dargestellt. Zu bestimmen sind nun noch die Konstanten $K_{p,i,d}$. Diese lassen sich mittels verschiedener Verfahren [TR15, Kap. 4.2] oder durch Simulation der Regelkomponente experimentell (bspw. in MATLAB/Simulink [Mat]) ermitteln.

Vor- und Nachteile von PID-Reglern:

+ Leicht zu implementieren
+ In der Industrie weit verbreitet.
− Schwer asymmetrisches Verhalten gut zu modellieren.
− Kleine Messfehler können große Auswirkungen auf ableitenden Anteil haben.

3.9. XiL Testmethoden

Mit steigender Komplexität und Anzahl verbauter Assistenzsystem in einem Fahrzeug wird es immer schwerer die Korrektheit des Verhaltens eines neu entwickelten Systems zu verifizieren. Für ein erfolgreiches Zusammenspiel von Sensoren und Aktoren ist die Korrektheit der Kontrollsoftware jedoch essentiell. Da das Zurückgreifen auf Regalware, aufgrund deren eingeschränkten Funktionalität, aber oft keine Option darstellt, ist die Frage zu beantworten, wie man neu entwickelte Systeme testet.

Das Testen und Herbeiführen eines gewünschten Systemzustands, ohne Veränderungen an der zu testenden Komponenten vorzunehmen, könnte man theoretisch auch händisch durchführen. Jedoch ist dies je nach Anzahl und Komplexität der Zustände schnell nicht mehr realistisch. So kann es sein, dass der Zustand des Regelungssystems nicht nur vom aktuellen Eingabevektor abhängt, sondern zusätzlich auf vorherigen Eingaben beruht. Es wird also mit schnell steigender Komplexität schwer, realistische Zustände des Regelungssystems zu erzeugen. Stattdessen gibt es verschiedene Möglichkeiten die Gesamtheit oder Teile eines neu entwickelten Systems zu testen. Die verschiedenen Methoden sind unter dem Sammelbegriff X in the Loop (XiL) zusammengefasst. Im Folgenden wird sich dabei auf *Model in the Loop (MiL)* und *Hardware in the Loop (HiL)* bzw. *Vehicle in the Loop (ViL)* mit besonderem Augenmerk auf *Vessel in the Loop* konzentriert.

Model in the Loop

Model in the Loop (MiL) dient zur Simulation eines neu entwickelten Systems in einem sehr frühen Entwicklungsstadium. Das erstellte Modell wird innerhalb des Modellierungswerkzeug (bspw. Simulink oder MATLAB) getestet. Es wird das entwickelte Modell mithilfe eines Modells der Umwelt, in der das System später eingesetzt wird, getestet. [AE15]

Die oben genannten Modellierungswerkzeuge werden mit einer Vielzahl von Debuggignwerkzeugen ausgeliefert. So können bereits erste Modellierungsfehler identifiziert und behoben werden.

Vehicle in the Loop

Vehicle in the Loop (ViL) soll die Vorteile der Simulation und die der Realfahrt bei der Entwicklung neuer Assistenzsysteme im Automobilbereich miteinander kombinieren. So wird eine sichere reproduzierbare Fahrsituation ähnlich zu der einer Simulation erzeugt. Es besteht allerdings

der Vorteil sich nicht auf Systeme verlassen zu müssen, die nur begrenzt in der Lage sind die komplexen Einflüsse der Umwelt auf das Fahrzeug korrekt darzustellen.

In [Boc12] wird ein solches System für den Automobilbereich vorgestellt. Hier wird ein Bremsassistent entwickelt, der anschließen mit einer ViL Simulation getestet wird. Für die Visualisierung der anderen Verkehrsteilnehmer wird ein *Optical head-mounted display* genutzt, dass der Fahrer des zu testenden Fahrzeugs trägt. So ist es möglich ohne ein erhöhtes Risiko verschiedene Situationen im Straßenverkehr abzubilden und Gefahrensituationen zu simulieren.

Vessel in the Loop

Vessel in the Loop ist ein von Vehicle in the Loop (ViL) abgeleitetes Testkonzept. In diesem konkreten Fall soll die e-Martime Integrated Reference Platform (eMIR) für die Umsetzung des Konzeptes dienen. eMIR bietet eine ganzheitliche Lösung für die Fragestellung wie sich ViL in die Maritime Welt übertragen lässt. eMIR setzt sich aus zwei Testfeldern zusammen. Zum einen beinhaltet eMIR das virtuelle Testfeld HAGGIS, zum anderen das physikalische Testfeld LABSKAUS.

Um den Anforderungen an ein simulatives Testfeld gerecht zu werden bietet HAGGIS folgende Möglichkeiten (nach [HEUa]):

- **Maritime Traffic Simulator** zur realistischen Simulation von Verkehrssituationen mit mehreren Verkehrsteilnehmern.
- **Environment Simulation** zur Simulation der Schiffsumwelt.
- **Sensor Simulation** zur Erzeugung realistischer Sensordaten.

Das Physikalische Testfeld LABSKAUS kombiniert land- und schiffsseitige Komponenten um eine Basis für das Testen der Anforderungen in den verschieden Funktionsschichten der Automatisierung eines Schiffes gerecht zu werden (nach [BH17]):

- **Vessel Traffic Service (VTS)/ Maritime Control Station** ist eine mobile Einheit zur Überwachung und gegebenenfalls auch zur Kontrolle des Schiffsverkehrs.
- **Mobile Bridge** ist eine mobile Schiffsbrücke, die je nach Bedarf auch an Bord des Forschungsschiffs verbaut werden kann.
- **Navibox** integriert die benötigten Sensoren und stellt Daten wie Radar und AIS zur Verfügung.
- **Reference Waterway** ist ein Gebiet, das mit einer grundlegenden Überwachungsinfrastruktur ausgebaut ist.

- **Forschungsboot Zuse** ist ein mit einer mobilen Navibox ausgestatteten Forschungsschiff, zum Testen auf See.
- **e-Navigation Prototype Display (EPD)** ist ein ECDIS-ähnliches Navigationsinformationssystem. Es teilt sich in die EPD-Ship und die EPD-Shore auf.

Mittels HAGGIS ist es möglich HiL-Tests mit den entstehenden Softwaresystemen für ein autonomes Schiff durchzuführen. Bei diesen Tests bleibt jedoch zu beachten, dass ein simuliertes Schiff nur annähernd das Verhalten eines realen physikalischen Schiffs abbilden kann. Um die Testbedingungen so realitätsnah wie möglich zu gestalten kann der HiL-Testaufbau mit dem physikalischen Testfeld LABSKAUS um ein reales Schiff und eine überwachte Testumwelt erweitert werden. Dieser Testaufbau kann als Vehicle in the Loop bezeichnet werden und ermöglicht das Erproben neuer Systeme und Komponenten in der Praxis in Kombination mit der Reproduzierbarkeit und Sicherheit von Verkehrssimulatoren. Der Vehicle in the Loop Testaufbau erweitert die reale Umgebung mit simulierten Schiffen und erzeugt so einen integrierten Testaufbau aus Simulation und Realität.

4. Anforderungen

Die Anforderungen an das MATE II Produkt und dessen Komponenten umfassen die Leistungen und Einschränkungen der Funktionen des Systems. Dabei wird in diesem Kapitel auf die Anforderungen, die Analyse und Findung dieser eingegangen. Im Kapitel der Evaluation (vgl. Abschnitt 7.3) wird die Überprüfung der Anforderungen innerhalb der Systemtests fortgeführt.
Nach Ian Sommerville gibt es zwei Definitionen für Anforderungen. Zum einen die Benutzeranforderungen und zum anderen die Systemanforderungen. Welche Anforderungsdefinition genutzt werden soll hängt von der Zielgruppe der Leser ab (vgl. [Som12, S. 115-116]). Benutzeranforderungen sind in natürlicher Sprache geschrieben und nutzen Diagramme zur Beschreibung der Dienste und Randbedingungen für das Betriebssystem.

Die folgenden Anforderungen sind nach der Definition der Systemanforderungen aufgestellt worden. Sie sind detaillierte Beschreibungen der Funktionen, Dienste und Beschränkungen eines Systems [Som12, S. 115]. Daraus gehen die Funktionen und Implementierungsansätze für das MATE II Produkt hervor.
Anforderungen werden grundsätzlich in funktional und nicht-funktional aufgeteilt.

Funktionale Anforderungen Sie sind explizite Beschreibungen der Dienste des Systems und der Komponenten. Des Weiteren bestimmen sie Reaktionen des Systems auf (Nutzer-)Eingaben und definierte Szenarien. Sie können auch explizit beschreiben, was nicht passieren soll [Som12, S. 116].

Nichtfunktionale Anforderungen Sie sind Beschränkungen der zur Verfügung gestellten Dienste und Funktionen des Systems, die zum Beispiel das Leistungsniveau, die Informationssicherheit oder die Verfügbarkeit spezifizieren. Darunter werden Zeitbeschränkungen oder Beschränkungen durch Standards mit eingeschlossen, wodurch weitere funktionale Anforderungen entstehen können. Im Rahmen des MATE II Produkts stehen Produktanforderungen, die das Verhalten des Systems festlegen oder beschränken (z.B. Effizienz- und Speicherauslastungsanforderungen) im Fokus [Som12, S. 116, 119].

4.1. Methoden der Anforderungserhebung

Bevor die Systemanforderungen des MATE II Produkts spezifiziert wurden, mussten Informationen darüber gesammelt, analysiert und herausgearbeitet werden. Als Informationsquellen standen dafür Dokumentationen ähnlicher und vorangegangener Systeme zur Verfügung. Des Weiteren waren Experten für jede benötigte Komponente des Systems ansprechbar. Diese haben neben dem Lehrstuhl und der Projektbetreuung Interesse am Erfolg des Projekts und sind somit Projektbeteiligte, die Informationen zu Anforderungen liefern können. Daraus ergeben sich die folgenden Methoden der Anforderungserhebung:

Systemanalyse

Dadurch, dass das MATE II Produkt ein Nachfolgeprojekt darstellt, muss das vorhandene System analysiert werden. Die Systemanalyse besteht aus fünf Schritten (vgl. Abbildung 4.1)

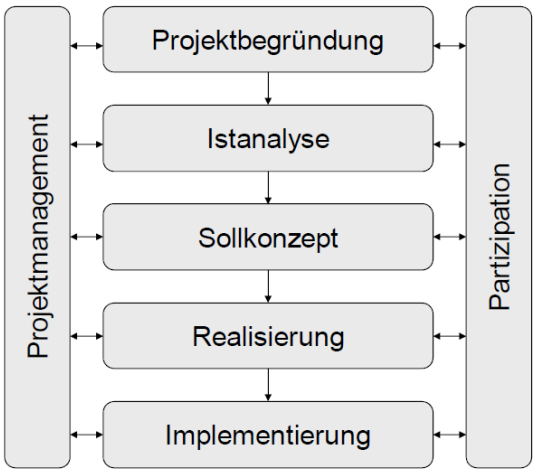


Abbildung 4.1.: Vorgehensmodell der Systemanalyse in einem Unternehmen (vlg. [KBL13, S. 118])

Durch die gegebene Projektstruktur (vgl. Abschnitt 2.1), werden die Schritte der Projektbegründung, Realisierung und Implementierung nicht weiter beachtet. Die Istanalyse des vorhandenen Systems und das Sollkonzept liefern Informationen zu benötigten Anforderungen. Dabei beginnt die Istanalyse mit der Istaufnahme, die die qualitative und quantitative Erfassung des Istzustandes eines abgegrenzten Systems unter Berücksichtigung des Untersuchungszwecks darstellt [KBL13, S. 129].

Dabei gibt es unterschiedliche Methoden der Istaufnahme, die sich in Primärerhebung und Sekundärerhebung unterscheiden lassen. Bei der Primärerhebung werden Informationen erstmalig und nur für die Information gewonnen. Während die Sekundärerhebung auf vorhandene Quellen, Texte, Ausarbeitungen und sonstigen Dokumenten basiert [KBL13, S. 135]. Die genutzten Methoden zur Erhebung des Iszustandes des vorhandenen Systems und der daraus resultierenden Anforderungserhebung sind die Dokumentenanalyse (Sekundärerhebung), Interviews mit Experten und Projektbeteiligten (Primärerhebung) und Szenarioanalysen (Primärerhebung).

Dokumentenanalyse Die Dokumentenanalyse (auch Inventurmethode) besteht aus der kritischen Auseinandersetzung mit vorhandenen schriftlichen Unterlagen und liefert Informationen zu Prozessen und Strukturen im vorhandenen System. Zu Problemen kann es kommen, wenn Informationen nicht festgehalten wurden oder Fakten veraltet sind [KBL13, S. 136].

Die im Abschnitt 3.1 aufgeführten Arbeiten dienten als Grundlage der Dokumentenanalyse und den daraus abgeleiteten Anforderungen. Aber auch Anleitungen, Readmes und Präsentationen, die zur Verfügung gestellt wurden dienten dem Verständnis für benötigte Anforderungen.

Interviews Projektbeteiligte und Experten haben meist ein Interesse daran ein Projekt nach ihrem Wissen zu Fördern und sind in den meisten Prozessen der Anforderungserhebung und Istanalyse beteiligt. In Interviews werden den oder dem Beteiligten Fragen zum aktuellen und zum zu entwickelnden System gestellt. Dabei gibt es zwei Arten von Interviews. Das geschlossene und das offene Interview. Beim geschlossenen Interview werden Fragen vorbereitet, die nacheinander beantwortet werden. Beim offenen Interview werden die offenen Punkte erläutert und der Projektbeteiligte erlangt auf diese Weise ein Verständnis für die Bedürfnisse des Fragenden.

In der Praxis wird meistens jedoch eine Mischung der beiden Möglichkeiten verwendet. Es werden einige kritische Fragen vorbereitet, über die anschließend offener gesprochen und diskutiert wird. Daraus ergibt sich ein leichterer Einstieg in den Dialog und der Interviewte hat die Möglichkeit sein Wissen in den Kontext einzubringen [Som12, S. 137].

Szenarien Ein Szenario ist ein reales Beispiel, indem eine Komponente innerhalb des Softwaresystems interagiert. Sie sind leichter verständlich als abstrakte Beschreibungen und liefern somit mehr potentielle Kritik, die ausgearbeitet werden kann. Dabei wird eine Interaktion (oder einige wenige) abgedeckt. Aus einem groben Szenario werden im Laufe der Anforderungserhebung Details ausgearbeitet, aus denen eine vollständige Beschreibung der Interaktion hervorgeht.

Anhand des Beispiels *Ausweichen* aus dem Forschungsprojekt wird der Aufbau und Inhalt eines Szenarios beschrieben [Som12, S. 139-140]:

1. Zu Beginn gibt es eine Beschreibung, was den Nutzer und das System erwartet, sobald das Szenario anfängt.
   Ein Schiff fährt gegenüber vom physikalischen Testfeld auf direktem Kollisionskurs und weicht nicht aus. Es wird erwartet, dass die Hindernisidentifizierung die Situation erkennt und die Kollisionsverhütung ein autonomes Ausweichmanöver einleitet.
2. Der normale Ereignisablauf wird im Szenario beschrieben.
   Das physikalische Testfeld kollidiert nicht, leitet ein Ausweichmanöver auf einem validen Pfad ein und kehrt anschließend auf die Route zurück.
3. Es wird beschrieben was falsch laufen kann und wie damit umgegangen wird.
   Es kann kein valider Pfad gefunden werden. Dann wird die Geschwindigkeit des physikalischen Testfelds auf ein Minimum gedrosselt.
4. Der Endzustand des Szenarios wird beschrieben.
   Das physikalische Testfeld befindet sich ohne Kollision auf der ursprünglich geplanten Route.

Für die Bestimmung der Szenarios und der einzubeziehenden Details sind Projektbeteiligte und Experten erforderlich, mit deren Hilfe die Szenarios textuell und mithilfe von Bildern oder Diagrammen aufgenommen werden [Som12, S. 139-140].

4.2. MATE II-Produkt

Das MATE II-Produkt stellt das gesamte zu entwickelnde System dar, inklusive aller zur Darstellung und Evaluierung genutzten und erstellten Komponenten. Um die korrekte Umsetzung der Funktionalität zu erzielen, müssen Rahmenbedingungen festgelegt werden, die die Entwicklung eines zielführenden Konzepts ermöglichen. Dafür werden Anforderungen mithilfe der in Abschnitt 4.1 vorgestellten Methoden erhoben. Diese Anforderungen sollen die Grundfunktionalitäten darstellen, die im Testfeld LABSKAUS eine maritime, autonome Fahrt auf dem Schiff ermöglichen. Das stellt die letzte Stufe des Autonomisierungsgrades dar (vgl. Abschnitt 3.1).

Funktionale Anforderungen

Im Folgenden werden die Anforderungen an das Gesamtsystem detaillierter beschrieben. Die Anforderungen wurden nach folgendem Schema aufgeteilt:

ID: Die ID ist die einzigartige Kennung der Anforderung.

Beschreibung: In der Beschreibung findet sich die ausformulierte Anforderung.

Problembeschreibung: In der Problembeschreibung befindet sich der Bezug Kontext der Anforderung.

Abnahmekriterium: Das Abnahmekriterium wird in Abschnitt 7.3 genauer definiert und stellt das Kriterium zur Erfüllung der Anforderung dar.

ID	F-G-1
Beschreibung	Das Gesamtsystem muss eine Strecke auf einer Route autonom abfahren.
Problembeschreibung	Das autonome Abfahren einer aus mindestens zwei Wegpunkten bestehenden Route ist eine grundlegende Funktion der autonomen Seefahrt und somit Bestandteil der Funktionalität des Gesamtsystems.
Abnahmekriterium	Das autonome Schiff muss eine Route in einem vordefinierten Szenario mit einer maximalen vordefinierten Abweichung abfahren.

Tabelle 4.1.: Anforderung an die Routenfahrt

In der PG MATE I wurde der zweite definierte Meilenstein nicht vollständig erreicht (vgl. Abschnitt 3.1), ist aber ein essentieller Bestandteil eines autonomen Schiffes. Damit ist die Erfüllung der Anforderung für die Umsetzung unabdingbar.

ID	F-G-2
Beschreibung	Das Gesamtsystem muss die aktuelle Situation mit Hilfe von Sensoren wahrnehmen.
Problembeschreibung	Dadurch, dass das autonome Schiff auf reale Umwelteinflüsse und dynamische Gegebenheiten reagieren muss, werden Sensoren benutzt (vgl. Abschnitt 3.5) die, die benötigten Informationen über die Umwelt liefern.
Abnahmekriterium	Das autonome Schiff muss Umwelteinflüsse in einem vordefinierten Szenario korrekt erkennen und verarbeiten.

Tabelle 4.2.: Anforderung an die Schiffswahrnehmung

Um eine Reaktion auf die Umwelt umsetzen zu können, erzeugen die Sensoren des physikalischen Testfelds (vgl. Abschnitt 3.5.8) einen Datenstrom, auf den mithilfe der eingebundenen Schnittstellen (vgl. Abschnitt 3.1) zugegriffen werden kann.

ID	F-G-3
Beschreibung	Das Gesamtsystem muss die aktuelle Situation dem Nutzer präsentieren.
Problembeschreibung	Für die Kontrolle und Überwachung des Schiffes und der maritimen Umgebung, muss das System ein ECDIS ähnliches System mit allen relevanten und sicherheitskritischen Anzeigen beinhalten.
Abnahmekriterium	Das ECDIS ähnliche System zeigt ausgewählte Inhalte an korrekter Stelle an und wird mithilfe eines Realitätsabgleichs in der eingesetzten Umgebung innerhalb eines ausgewählten Szenarios überprüft.

Tabelle 4.3.: Anforderung an die Präsentation der maritimen Situation

Die Daten des Datenstroms sollen bearbeitet oder unbearbeitet in einer zentralen Komponente gesammelt und dargestellt werden. Dafür wird ein ECDIS-ähnliches System verwendet (vgl. Abschnitt 3.3).

ID	F-G-4
Beschreibung	Das Gesamtsystem muss auf maritime Situationen reagieren können.
Problembeschreibung	Es gibt viele Situationen in maritimen Umgebungen, die eine dynamische Reaktion eines autonomen Seefahrtsystems erfordern. Im Fokus der Entwicklung standen die Situationen, die in den COLREGs 13-17 beschrieben werden (vgl. Abschnitt 3.2).
Abnahmekriterium	Für jede COLREG-Situation wird ein Szenario im Testfeld erstellt und nach ausgewählten Erfüllungskriterien validiert.

Tabelle 4.4.: Anforderung an die Reaktion auf maritime Situationen

Die in Tabelle 4.2 wahrgenommenen Situationen müssen interpretiert und verarbeitet werden, um eine COLREG-konforme Reaktion daraus herleiten zu können (vgl. Maritime Sensoren in Abschnitt 3.5 und Kollisionsverhütungskonzepte in Abschnitt 3.4).

Es muss die Möglichkeit geben die Schnittstellen des physikalischen Testfelds zu nutzen, um als Nutzer eine Route in Abhängigkeit von Schiffsparametern erstellen zu können, die an das autonome Schiff weitergeleitet wird. Dafür wird das Distributionssystem von PG MATE I verwendet (vgl. Abschnitt 3.1).

ID	F-G-5
Beschreibung	Das Gesamtsystem muss eine gültige Route in Abhängigkeit der Schiffsparameter erstellen.
Problembeschreibung	Das autonome Schiff benötigt eine interpretierbare, valide Route, die für das gewählte Schiff mit seinen Eigenschaften (Länge, Breite, Höhe, Tiefgang), eine sichere Fahrt ermöglicht.
Abnahmekriterium	Zwischen zwei gewählten Wegpunkten wird eine Route erstellt, die statische Hindernisse berücksichtigt und in einem Szenario mit realen Hindernisse validiert wird.

Tabelle 4.5.: Anforderung an die Routenerstellung

ID	F-G-6
Beschreibung	Das Gesamtsystem muss Controller-Eingaben manuell überschreiben können.
Problembeschreibung	Das überschreiben der Controller-Eingaben ermöglicht eine manuelle Steuerung des autonomen Schiffes und bietet die Möglichkeit eine externe Sicherheitskomponente für sicherheitskritische Situationen zu entwickeln. Für die manuelle Steuerung wird ein externes Interface benutzt.
Abnahmekriterium	Es werden die Steuerbefehle für den Ruderwinkel und die Schiffsgeschwindigkeit gesendet, während der Controller aktiv ist und dieser muss sie übernehmen. Nach Beendigung der manuellen Steuerung übernimmt der autonome Controller wieder die Fahrt.

Tabelle 4.6.: Anforderung an die manuelle Schiffsteuerung

Die Anforderung wurde bereits von PG MATE I erfüllt, ist für das Gesamtsystem aber dennoch Relevant (vgl. Abschnitt 3.1).

4.3. Küstenleitstand

Im folgenden Kapitel werden die funktionalen und nicht funktionalen Anforderungen an den Küstenleitstand aufgeführt.

Funktionale Anforderungen

F-EPD-1	Das System muss verschiedene Roh-Daten der verwendeten Systeme im MATE-Produkt auf der Karte anzeigen.
F-EPD-1.1	Das System muss AIS-Daten auf der Karte anzeigen können.
F-EPD-1.2	Das System muss Radar-Daten auf der Karte anzeigen können.
F-EPD-1.3	Das System muss Grid-Daten auf der Karte anzeigen können.
F-EPD-1.4	Das System muss Polygone auf der Karte anzeigen können.
F-EPD-1.5	Das System muss die Schiffsposition auf der Karte anzeigen können.
F-EPD-2	Das System muss es ermöglichen, eine Route mit zwei oder mehr Wegpunkten erstellen zu können.
F-EPD-3	Das System muss eine Maske zur Speicherung von eigenen Schiffsdaten (Höhe, Breite, Tiefgang, Länge) zur Verfügung stellen und diese speichern können.
F-EPD-4	Das System muss eine ausgewählte Route gemeinsam mit den eigenen Schiffsdaten an die Routen-Planung senden können.
F-EPD-5	Das System muss eine optimierte Route von der Routen-Planung empfangen und auf der Karte darstellen können.
F-EPD-6	Das System muss es ermöglichen, einen Kamerastream empfangen und anzeigen zu können.
F-EPD-7	Das System muss die Eingabe und das Senden von manuellen Steuerbefehlen ermöglichen.
F-EPD-8	Das System muss die Steuerbefehle an die Steuereinheit des autonomen Schiffs weiterleiten können.
F-EPD-9	Das System muss das Starten der autonomen Fahrt des autonomen Schiffs ermöglichen.

Nicht-funktionale Anforderungen

NF-EPD-1	Das System muss sicherstellen, dass korrekte Benutzereingaben, valide Daten erzeugen die weiterverarbeitet werden können.
NF-EPD-2	Das System soll möglichst zuverlässig arbeiten.
NF-EPD-3	Das System muss Daten in einer möglichst geringen Laufzeit verarbeiten.

4.4. Routenplanung

Im Folgenden werden die funktionalen und nicht-funktionalen Anforderungen an die Routen-Planung aufgeführt.

Funktionale Anforderungen

F-CA-RP-1	Die Komponente soll statische Hindernisse (Land & Untiefen) von dem NoGoSolver anfragen, entgegennehmen und decodieren können.
F-CA-RP-2	Die globale Route und die Schiffsparameter sollen von der EPD entgegengenommen und decodiert werden.
F-CA-RP-3	Die Komponente soll über eine interne Repräsentation, welche das Gebiet für jeweils zwei aufeinanderfolgende Wegpunkte darstellt, verfügen.
F-CA-RP-4	Die Komponente soll einen Algorithmus bereitstellen, welcher eine valide Route berechnet, die nicht im Konflikt zu statischen Hindernissen steht.
F-CA-RP-5	Die valide, globale Route soll an die EPD zurückgeschickt werden.

Nicht-funktionale Anforderungen

NF-CA-RP-1	Die berechnete valide, globale Route soll relativ effizient sein.
NF-CA-RP-2	Die globale Route soll einen ausreichenden Mindestabstand zu den statischen Hindernissen aufweisen können.

4.5. Kollisionsverhütung

Im folgenden Kapitel werden die funktionalen und nicht funktionalen Anforderungen an die Kollisionsverhütung aufgeführt.

Funktionale Anforderungen

F-CA-PP-1	Die Komponente soll statische Hindernisse (Land & Untiefen) von dem NoGoSolver anfragen, entgegennehmen und decodieren können.
F-CA-PP-2	Die Komponente soll dynamische Hindernisse (andere Schiffe) von der Dynamic Object Identification entgegennehmen und decodieren können.
F-CA-PP-3	Die valide globale Route und die Schiffsparameter sollen von der EPD entgegengenommen und decodiert werden.
F-CA-PP-4	Die Position und das Heading sollen vom Hub entgegengenommen und decodiert werden.
F-CA-PP-5	Die Komponente soll über eine interne Repräsentation, welche das Gebiet für jeweils zwei aufeinanderfolgende Wegpunkte darstellt, verfügen.
F-CA-PP-6	Die Komponente soll einen Algorithmus bereitstellen, welcher einen validen Pfad berechnet, der nicht im Konflikt zu statischen oder dynamischen Hindernissen steht.
F-CA-PP-7	Die Komponente soll bei der Berechnung eines validen Pfades die physikalischen Eigenschaften der Zuse berücksichtigen.
F-CA-PP-8	Die Komponente soll bei der Berechnung eines validen Pfades die COLREGs Regeln 13 bis 17 (Head-On, Crossing-Give-Way, Crossing-Stand-On und Overtake) einhalten.
F-CA-PP-9	Der valide Pfad soll über das Distribution System an den Controller geschickt werden.
F-CA-PP-10	Der valide Pfad soll über das Distribution System an die EPD geschickt werden.
F-CA-PP-11	Bei einem Verbindungsabbruch zum NoGoSolver soll unendlich lange versucht werden die Verbindung wiederherzustellen.
F-CA-PP-12	Bei einem Verbindungsabbruch zur Dynamischen Objektidentifizierung soll unendlich lange versucht werden die Verbindung wiederherzustellen.
F-CA-PP-13	Bei einem Verbindungsabbruch zum Distributionssystem soll unendlich lange versucht werden die Verbindung wiederherzustellen.
F-CA-PP-14	Bei einem Verbindungsabbruch zur EPD soll unendlich lange versucht werden die Verbindung wiederherzustellen.

F-CA-PP-15	Bei einem Verbindungsabbruch zum Controller soll unendlich lange versucht werden die Verbindung wiederherzustellen.
F-CA-PP-16	Die Komponente soll bei einem Verbindungsabbruch zum NoGoSolver, Dynamischen Objektidentifizierung oder Distributionssystem die Berechnungen von neuen validen Pfaden pausieren.
F-CA-PP-17	Die Komponente soll permanent neue valide Pfade berechnen, wenn eine Verbindung zum NoGoSolver, Dynamischen Objektidentifizierung oder Distributionssystem besteht.

Nicht-funktionale Anforderungen

NF-CA-PP-1	Der berechnete valide Pfad soll relativ effizient sein.
NF-CA-PP-2	Der berechnete Pfad soll einen ausreichenden Mindestabstand zu den statischen und dynamischen Hindernissen aufweisen können.

4.6. Hindernisidentifikation

Im folgenden Kapitel werden die funktionalen und nicht funktionalen Anforderungen an die Hindernisidentifikation aufgeteilt in Objektdetektion und dynamische Objektidentifizierung aufgeführt.

4.6.1. Objektdetektion

Nachfolgend befinden sich die funktionalen und nicht funktionalen Anforderungen der Objektdetektion.

Funktionale Anforderungen

F-OD-1	Das System muss die Umgebung mittels einer Kamera erfassen.
F-OD-2	Das System muss andere Schiffe in Fahrtrichtung erkennen.
F-OD-3	Das System muss den Horizont erkennen können.
F-OD-4	Das System muss die Position (rechts/links vom eigenen Schiff) der erkannten Schiffe, welche sich unter dem Horizont befinden, identifizieren können.
F-OD-5	Das System muss die Position (rechts/links) der erkannten und relevanten (unter dem Horizont) Schiffe an die Dynamische Objektidentifizierung übertragen können.

Nicht-funktionale Anforderungen

NF-OD-1	Das System muss mit wechselnden Umweltparametern umgehen können.
NF-OD-1.1	Das System soll auch bei schlechter Witterung funktionieren.
NF-OD-1.2	Das System muss mit leichtem Wellengang umgehen können.
NF-OD-2	Der Algorithmus muss rechtzeitig Ergebnisse liefern.

4.6.2. Dynamische Objektidentifizierung

Nachfolgend befinden sich die funktionalen und nicht funktionalen Anforderungen der dynamischen Objektidentifizierung.

Funktionale Anforderungen

F-DOI-1	Das System muss Sensordaten des AIS empfangen.
F-DOI-2	Das System muss Sensordaten des Radars empfangen.
F-DOI-3	Das System muss kontinuierlich Informationen über dynamische Objekte in der Umgebung erhalten.
F-DOI-3.1	Das System muss Informationen von dynamischen Objekten vor dem Schiff erhalten.
F-DOI-3.2	Das System muss Informationen von dynamischen Objekten neben dem Schiff erhalten.
F-DOI-3.3	Das System muss Informationen von dynamischen Objekten hinter dem Schiff erhalten.
F-DOI-4	Das System muss Schiffe, die sich in einem vorgegebenen Umkreis des eigenen Schiffes befinden, identifizieren können.
F-DOI-5	Das System muss objektspezifische Parameter erkennen.
F-DOI-5.1	Das System muss die Größe von Objekten erkennen.
F-DOI-5.2	Das System muss die Position von Objekten erkennen.
F-DOI-5.3	Das System muss die Geschwindigkeit eines Objekts bestimmen.
F-DOI-5.4	Das System muss den Kurs eines Objekts bestimmen.
F-DOI-6	Das System muss erkennen, dass verschiedene AIS Nachrichten von einem Schiff sind.
F-DOI-7	Das System muss verschiedene AIS Nachrichten und Radar Tracks eines Schiffes diesem zuordnen.

F-DOI-8	Das System muss die objektspezifischen Parameter der fusionierten Tracks der Schiffe so exakt wie möglich schätzen können.
F-DOI-8.1	Das System muss die Position der Schiffe so exakt wie möglich schätzen können.
F-DOI-8.2	Das System muss die Größe der Schiffe so exakt wie möglich schätzen können.
F-DOI-8.3	Das System muss die Geschwindigkeit der Schiffe so exakt wie möglich schätzen können.
F-DOI-8.4	Das System muss den Kurs der Schiffe so exakt wie möglich schätzen können.
F-DOI-9	Das System muss alle in einem vorgegebenen Umkreis erkannte dynamische Objekte mit den Parametern Position, Geschwindigkeit, Kurs und Größe an die Pfad-Planung senden.

Nicht-funktionale Anforderungen

NF-DOI-1	Das System muss mit wechselnden Umweltparametern umgehen können.
NF-DOI-1.1	Das System soll auch bei schlechter Witterung funktionieren.
NF-DOI-1.2	Das System muss mit leichtem Wellengang umgehen können.
NF-DOI-2	Der Algorithmus muss rechtzeitig Ergebnisse liefern.

4.7. Integration simulierter Daten ins physikalische Testfeld

Im folgenden Kapitel werden die funktionalen und nicht funktionalen Anforderungen an die Integration simulierter Daten in das physikalische Testfeld aufgeführt.

Funktionale Anforderungen

F-SA-1	Der Simulationsadapter muss uCore konforme XML Daten aus HAGGIS via UDP in festgelegten Zeitabständen empfangen können.
F-SA-2	Die uCore konformen XML Daten müssen mithilfe von XSL-Transformationen in Ecore konforme XML Daten umgewandelt werden.
F-SA-3	Es müssen verschiedene XSL-Transformationsregeln aufgestellt werden.
F-SA-3.1	Es müssen statische AIS-Daten in Ecore konforme XML Daten transformiert werden.
F-SA-3.2	Es müssen dynamische AIS-Daten in Ecore konforme XML Daten transformiert werden.

F-SA-3.3 Es müssen Radar-Daten in Ecore konforme XML Daten transformiert werden.

F-SA-3.4 Es müssen Positions-Daten in Ecore konforme XML Daten transformiert werden.

F-SA-4 Die unterschiedlichen XSL-Transformationsregeln müssen in separaten XSLT-Sheets angelegt sein.

F-SA-5 Um Erweiterbarkeit zu gewährleisten muss es möglich sein weitere XSLT-Sheets einzubinden.

F-SA-6 Die transformierten XML Daten müssen dem S-100 und Ecore konform sein.

F-SA-7 Ein Ecore konformer XML Datenstrom muss via RabbitMQ an das Polymorphe Interface übertragen werden.

Nicht-funktionale Anforderungen

NF-SA-1 Die ausgehenden XML Daten müssen S-100 und Ecore konform sein.

4.8. Regelungstechnik

Im Folgenden Kapitel werden die funktionalen und nicht funktionalen Anforderungen an die Regelungstechnik aufgeführt.

Funktionale Anforderungen

F-CT-1 Das System muss den aktuellen Schiffzustand: Position, Bewegungsrichtung, Geschwindigkeit im NMEA2000-Standard empfangen.

F-CT-2 Das System muss 2 Pfadwegpunkte inkl. Routenstatus (aktiv, inaktiv) und eine gewünschte Geschwindigkeit im NMEA2000 Standard empfangen.

F-CT-3 Das System muss manuelle Steuerinformationen (Ruderwinkel und RPM) im NMEA2000 Standard empfangen können.

F-CT-4 Das System muss fähig sein mit Hilfe der vorliegenden Informationen einen RPM-Befehl zu berechnen und im NMEA2000 Standard herauszusenden, sodass die Zuse in angemessener Zeit die gewünschte Geschwindigkeit erreicht.

F-CT-5 Das System muss mit Hilfe der vorliegenden Informationen kontinuierlich Ruder-Befehle berechnen, die im NMEA2000 Standard herausgesendet werden, sodass in angemessener Zeit der zweite Wegpunkt erreicht wird, indem der Controller den Cross-Track-Error minimiert.

F-CT-6 Das System muss fähig sein durch manuelle Steuerbefehle den autonomen Regelungskreis zu überschreiben und das Schiff durch dieselben zu steuern.

F-CT-7 Das System muss alle ausgehenden Nachrichten in einem konstanten Abstand senden, wobei bei jede Iteration ein gültiger Wert am Ausgang anliegt.

Nicht-funktionale Anforderungen

NF-CT-1 Das System sollte möglichst zuverlässig arbeiten.

5. Konzept des MATE II-Produkts

In diesem Kapitel wird das Konzept des MATE II-Produktes vorgestellt und ausführlich beschrieben. Zu Beginn wird ein Überblick über das Gesamtsystem gegeben sowie die Abgrenzung zur Systemumwelt dargestellt. Darauf aufbauend werden in den folgenden Unterkapiteln die Konzepte der einzelnen Bereiche des Produktes detailliert beschrieben und vorgestellt.

5.1. Systemarchitektur

Die Abbildung 5.1 gibt eine Übersicht über den geplanten Aufbau des MATE II-Produktes. In ihr werden die Teilsysteme des Produktes mit ihren enthaltenen Komponenten dargestellt. Des Weiteren wird die Kommunikation zwischen den Komponenten sowie die Produktumgebung und die damit einhergehenden Schnittstellen zwischen Produkt und Produktumgebung dargestellt.

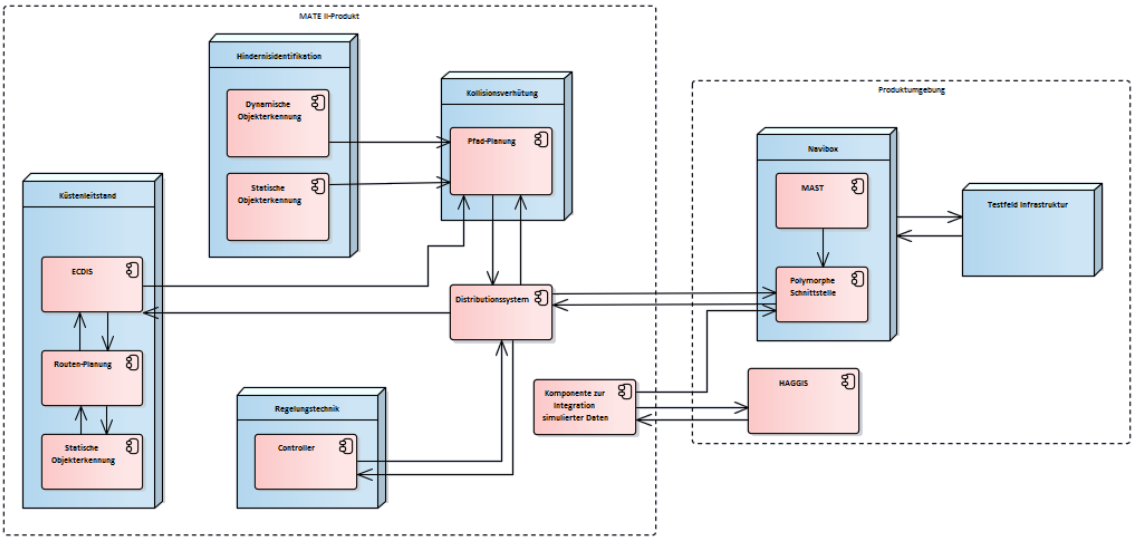


Abbildung 5.1.: Konzept des MATE II-Produkts

Das MATE II-Produkt teilt sich auf in einen Küstenleitstand, eine Hindernisidentifikation, eine Kollisionsverhütung, eine Regelungstechnik sowie in eine Komponente zur Integration simulierter Daten in das physikalische Testfeld und ein Distributionssystem. Das Konzept einer Distributionskomponente wurde hierbei von dem Vorgängerprodukt der MATE I Projektgruppe übernommen. Sie tauscht Daten zwischen den Teilsystemen sowie zwischen dem MATE II-Produkt und seiner Umgebung aus. Die Informationen der Produktumgebung werden hierbei über die polymorphe Schnittstelle in der Navibox des Schiffes an das Distributionssystem weitergeleitet. Eine weitere Schnittstelle zwischen Produkt und Umgebung gibt es zwischen der Komponente zur Integration simulierter Daten und HAGGIS. Hierbei werden Daten von HAGGIS über die Komponente an die Navibox gesendet. So ist die Komponente vom Rest des Produktes isoliert und kommuniziert ausschließlich mit Komponenten bzw. Systemen der Produktumgebung.

Der Küstenleitstand setzt sich aus einer ECDIS-Lösung, einer statischen Hinderniserkennung und einer Komponente zur Routenplanung zusammen. Dabei ist die ECDIS-Lösung die Schaltzentrale des Küstenleitstandes und die einzige Komponente des Teilsystems, die mit anderen Teilsystemen des MATE II-Produktes kommuniziert. Für die Optimierung von Routen fragt die ECDIS-Lösung die Komponente zur Routenplanung an. Diese optimiert die Routen mit Hilfe von Informationen der statischen Hinderniserkennung und leitet sie zurück an das ECDIS.

Die statische Hinderniserkennung wird nicht nur im Küstenleitstand eingesetzt, sondern ist ebenfalls Teil der Hindernisidentifikation. Neben der statischen Hinderniserkennung beinhaltet die Hindernisidentifikation auch die dynamische Hinderniserkennung. Beide Komponenten senden ihre Informationen an die Kollisionsverhütung. Diese beinhaltet die Komponente zur Pfadplanung. Mit Hilfe der Informationen der Hindernisidentifikation berechnet die Pfadplanung einen validen Pfad. Die Regelungstechnik (Controller) erhält daraufhin die nächsten Wegpunkte der Pfad Planung über das Distributionssystem. Mit weiteren Informationen wie der aktuellen Schiffsposition, die der Controller von der Navibox über das Distributionssystem erhält, werden Steuerbefehle berechnet und wieder über das Distributionssystem an die Navibox zurückgesendet.

5.2. Küstenleitstand

Im folgenden Abschnitt wird der Entwurf des in der PG MATE II umzusetzenden Küstenleitstandes beschrieben. Die im Anforderungskapitel aufgestellten Anforderungen an den Küstenleitstand lassen sich allgemein in zwei Hauptteile aufteilen. Der erste Teil beschäftigt sich fachlich mit der Routenerstellung über eine Benutzeroberfläche, während die anderen Anforderungen die zu entwerfenden Algorithmen zur Berechnung der jeweiligen Route beschreiben. Im ersten Abschnitt dieses Kapitels wird der Entwurf der grafischen Benutzeroberfläche des Küstenleitstandes vorgestellt. Dieser teilt sich in vier Bereiche ein, in die Dateneingabe in die grafische Oberfläche des Küstenleitstandes, den Empfang von Daten verschiedener externer Komponenten des Gesamtsystems zur Visualisierung von Sachverhalten, den Versand von Daten an externe Komponenten des Gesamtsystems und der Visualisierung von Daten. Die in MATE II einzusetzende Kontrollplattform im Küstenleitstand soll verschiedene Funktionen erfüllen. Neben der Darstellung aktueller Seekarten, der Darstellung von Positionsdaten und Kursdaten anderer Schiffe, und dem Erstellen von Routen für das Fahren des autonomen Schiffs, soll die Kontrollplattform auch Visualisierungen zur Route- und Pfadplanungsüberwachung und Einrichtungen zum Starten einer autonomen Fahrt gewährleisten.

Nutzung des e-Navigation Prototype Displays (EPD)

Um kein vollständiges ECDIS-System implementieren zu müssen, wurde nach einer bereits fertigen, bzw. auf die Anforderungen der PG MATE II anpassbare Softwarelösung gesucht. Eine mögliche Lösung ist dabei die im OFFIS weiterentwickelte ECDIS-ähnliche Anwendung „e-Navigation Prototype Displays“. Ein Zugriff auf den aktuellen Entwicklungsstand bzw. Quellcode wird über das OFFIS gewährleistet. Vorteilig bei der Verwendung der ECDIS ist zudem, dass die Entwicklung der PG MATE II auch nach dem Projektende weiterverwendet und in die allgemeine Entwicklung der Anwendung einfließen kann.

Pluginbasierte Funktionalitäten

Die EPD ist in der Programmiersprache „Java“ implementiert, modular aufgebaut und kann durch Plugins in ihrer Funktion erweitert werden. Bereits während der Entwurfsphase des Küstenleitstandes der PG MATE II bietet die EPD dabei folgende Plugins, die zur Realisierung genutzt werden können:

MapView	Plugin, welches eine Seekarte darstellt und zudem mittels Layer Informationen darstellen kann
RouteManager	Plugin, welches auf das MapView-Plugin zugreift und Werkzeuge bereitstellt, Routen zu erstellen
NMEASensors	Plugin, welches via TCP oder UDP einen NMEA0183-Datenstrom empfangen, parsen und weiterverarbeiten kann
ownShip-Viewer	Plugin, welches anhand einer einstellbaren glsMMSI das eigene Schiff auf der Seekarte hervorhebt
Radar Raw Viewer	Plugin, welches Radar-Objekte über Webservices empfängt und auf der Seekarte anzeigt
Rosi-Viewer-Plugin	Plugin, welches einen Media-Stream empfangen und in der EPD anzeigen kann

Entwurf der Erweiterungen

Um alle Anforderungen an den Küstenleitstand erfüllen zu können, ist eine Erweiterung der EPD notwendig. Aufgrund der Tatsache, dass die genannten Plugins neben den Basisfunktionalitäten auch Schnittstellen bereitstellen, können sämtliche Anpassungen in einem eigenen Plugin gekapselt werden. So können mit dem neu zu entwerfenden Plugin eigene Layer dem MapView-Plugin ergänzt oder Routen aus dem RouteManager-Plugin ausgelesen und gespeichert werden.

Eingabe von Daten

Um verschiedene Anforderungen an den Küstenleitstand der PG MATE II zu erfüllen, sind vom Anwender Parameter bzw. Daten einzugeben, welche anschließend gespeichert und zur weiteren Verarbeitung genutzt werden sollen.

Eingabe von Parametern der Schiffscharakteristik

Um Eigenschaften des autonomen Schiffes zur hindernisfreien Wegfindung mittels Routenplanung nutzen zu können, müssen diese vom Anwender in den Küstenleitstand eingegeben werden. Dazu ist eine Eingabemaske notwendig (Anforderung „F-EPD-3“). Die Eingabemaske sollte dabei die eingegebenen Parameter speichern und auch nach einem Neustart des umzusetzenden Küstenleitstandes ohne Neueingabe vorhalten. Benötigt werden Eingabefelder für die Höhe, Länge und Breite sowie den Tiefgang des autonomen Schiffes.

Eingabe von Wegpunkten zur Routenerstellung

Um eine Route erstellen zu können (Anforderung „F-EPD-2“) ist es notwendig, eine grafische Oberfläche bereitzustellen, die die Markierung von Wegpunkten auf einer Karte und deren Speicherung ermöglicht. Der im maritimen Umfeld genutzte und anerkannte Standard RTZ ermöglicht das standardisierte Speichern und den Austausch von Routen zwischen verschiedener Systeme. Dabei besteht eine Route aus mindestens zwei Wegpunkten. Das bereits bestehende RouteManager-Plugin der EPD bietet die Möglichkeit, eine Route über Routenwerkzeuge zu setzen, Geschwindigkeiten der einzelnen Wegpunkte zu definieren und diese als RTZ-Route in der EPD zu speichern. Auch ist es möglich, aus dem zu implementierenden Plugin auf diese Route zuzugreifen, um sie an externe Komponenten (wie z.B. die Routen-Planung) weiterzuleiten.

Eingabe von Steuerbefehlen zur Steuerung des autonomen Schiffs

Zur manuellen Steuerung des autonomen Schiffes (Anforderung „F-EPD-7“) kann das Plugin „VirtualHandles“ (von PG MATE I; „Notfallsteuerung“ genannt) genutzt werden. Dazu ist es notwendig, den von MATE I übernommenen Quellcode auf die neue EPD anzupassen, bzw. zu portieren. Darunter fällt die webbasierte grafische Oberfläche sowie die Websocket-Infrastruktur und Listener, die Eingaben über Dreh- und Steuerregler empfangen und weiterverarbeiten.

Versand von Daten

Zur Kommunikation zwischen der EPD und externen Komponenten des Gesamtsystems, wie die Pfad- oder Routen-Planung, ist ein Client notwendig, der die jeweiligen Daten überträgt. Da die EPD bereits mit anderen Schnittstellen über HTTP kommuniziert, ist es sinnvoll, auch hier einen HTTP-Client zu nutzen.

Versand von Routen

Um eine in der EPD erstellte Route an die Routen- und Pfad-Planung übertragen zu können, muss die ausgewählte Route aus dem RouteManager-Plugin ausgelesen werden. Dazu stellt das RouteManager-Plugin entsprechende Methoden zur Verfügung. Die ausgelesene Route soll anschließend über den HTTP-Client an die Routen-Planung gesendet werden. Die eigentliche Übertragung erfolgt als String, sodass die RTZ-Route vor der eigentlichen Übertragung umgewandelt werden muss. Das Übertragen der Routen soll in der Oberfläche initiiert werden, sodass zusätzliche Schaltflächen in das RouteManager-Plugin eingefügt werden müssen. Eine Schaltfläche soll dabei für das Starten der Optimierung einer Route durch die Routen-Planung, also das Berechnen einer hindernisfreien, globalen Route zur autonomen Fahrt, verwendet werden, während eine weitere Schaltfläche eine bereits optimierte Route an die Pfad-Planung senden soll und damit die autonome Fahrt startet (Anforderung „F-EPD-9“). Um Anforderung „F-EPD-4“ realisieren zu können, ist es in beiden Fällen notwendig, dass zusätzlich die in der Konfigurationsmaske des autonomen Schiffes hinterlegten Schiffsinformationen (Höhe, Breite, Tiefgang und Länge) an das jeweilige Ziel übertragen werden. Diese müssen dafür aus dem jeweiligen Kontext der EPD ausgelesen und per HTTP übertragen werden.

Versand von Steuerkommandos

Die bisherige Kommunikation, die Werte der VirtualHandles als NMEA2000-Nachricht über RabbitMQ zu versenden, sollte dabei auf eine standardisierte HTTP-Kommunikation umstrukturiert werden. Hierzu wird ein HTTP-Client notwendig, der die Ruder- und RPM-Steuerkommandos an das Regelungssystem weiterleitet. Damit die automatische Routenabfahrt durch die manuelle Steuerung überschrieben werden kann, muss zudem eine NMEA2000-Nachricht mit dem „RouteStatus“ versendet werden, die die manuelle Steuerung aktiviert. Damit würde Anforderung „F-EPD-8“ realisiert werden.

Empfang von Daten

Für einige Anforderungen ist es unter anderem notwendig, Daten von anderen Komponenten des Gesamtsystems zu empfangen, damit sie weiterverarbeitet und visualisiert werden können.

Empfang, Speicherung und Darstellung von Routen

Um eine von der Routen-Planung optimierte, globale Route empfangen zu können und damit Anforderung „F-EPD-5“ realisieren zu können, wird ein HTTP-Server benötigt. Dieser sollte die Möglichkeit bieten, einen eigenen, von anderen Dateneingängen getrennten Kontext bereitzustellen, über den Daten von der Routen-Planung empfangen werden können. Die empfangene Route soll dabei aus Wegpunkten bestehen, die nach dem Empfang als String wieder in eine RTZ-Route konvertiert und dem RouteManager-Plugin der EPD übergeben werden soll. Zur Darstellung der neuen, optimierten globalen Route ist somit keine weitere Implementierung notwendig, da nach dem erfolgreichen Speichern der Route durch den RouteManager diese in der EPD auswählbar und standardmäßig bereits auf der Karte sichtbar ist. Um eine optimierte von einer manuell erstellten Route unterscheiden zu können, muss der Name der Route im RouteManager mit einem Prä- oder Suffix versehen werden.

Empfang von AIS-Daten

Beim Empfang von AIS-Daten (Bedingung für Anforderung “F-EPD-1.1“) kann zwischen zwei verschiedenen Datenquellen unterschieden werden. Zum einen ist es erforderlich, dass andere Schiffe im direkten Umkreis des autonomen Schiffes in der EPD angezeigt werden, zum anderen ist es hilfreich, alle von der dynamischen Hindernisidentifikation verwendeten Rohdaten in der EPD anzuzeigen. Zum Empfang von AIS-Daten kann das bereits in der EPD vorliegende Plugin „NMEASensors“ verwendet werden.
Mittels dieses Plugins kann ein NMEA-Sensor konfiguriert werden, der per UDP einen Datenstrom empfängt. Das Plugin verarbeitet diesen Datenstrom dabei automatisch und leitet diesen an das MapView-Plugin weiter, wodurch Schiffe in ECDIS-konformer Darstellung auf der Seekarte angezeigt werden. Auch werden unter anderem der Name und der Kurs des jeweiligen Schiffes dargestellt.
Um die eigenen Schiffsposition hervorzuheben und damit von anderen Schiffen unterschieden werden kann (Anforderung “F-EPD-1.5“) muss das ownShip-Plugin aktiviert werden, wodurch die konfigurierte MMSI des autonomen Schiffes im am NMEA-Sensor eingehenden Datenstrom identifiziert und als Schiff besonders auf der Seekarte hervorgehoben wird. Um die Rohdaten (Koordinaten von Radar- und AIS-Datensätzen sowie fusionierte Schiffe) der dynamischen Hindernisidentifikation in der EPD anzuzeigen, müssen diese über einen eigenen HTTP-Kontext empfangen werden. Die Weiterverarbeitung muss dabei aus dem Parsing des JSON-Strings und der Speicherung der Datensätze als Koordinatenpunkte bestehen.

Die Speicherung sollte dabei zentral erfolgen, sodass von anderen Klassen des zu implementierenden Plugins zu jeder Zeit zugegriffen werden kann. Gespeicherte AIS-Rohdaten bzw. Koordinaten sollen erkennbar auf der EPD-Karte angezeigt werden. Beim Eintreffen neuer Rohdaten soll auch stets die Seekarte aktualisiert werden.

Empfang von Radar-Daten

Beim Empfang von Radar-Daten (Bedingung für Anforderung „F-EPD-1.2“) kann zwischen zwei verschiedenen Datenquellen unterschieden werden. Zum einen ist es erforderlich, zur Überprüfung der vom Radar erkannten Objekte, eine weitere Radar-Datenquelle in der EPD anzuzeigen. Dazu muss das bereits implementierte „Radar Raw Viewer“ aktiviert werden und eine Webservice-URL, die Radar-Tracks bereitstellt, in den Einstellungen der EPD konfiguriert werden. Eine weitere Anpassung ist hierfür nicht notwendig. Im Betrieb werden so Radar-Tracks des Webservice in einem eigenen Layer der EPD angezeigt.
Zum anderen müssen die von der dynamischen Hinderniserkennung genutzten Radar-Rohdaten empfangen werden. Dies sollte über einen fachlich von anderen Eingängen getrennten HTTP-Kontext erfolgen. So kann eine individuelle Verarbeitung der eingehenden Daten stattfinden. Die Weiterverarbeitung muss dabei aus dem Parsing des JSON-Strings und der Speicherung der Datensätze als Koordinatenpunkte bestehen. Das Eingehen der Radar-Daten sollte, analog zu den AIS-Rohdaten, zentral erfolgen, sodass von anderen Klassen des zu implementierenden Plugins zu jeder Zeit zugegriffen werden kann. Gespeicherte Radar-Rohdaten bzw. Koordinaten sollen erkennbar auf der EPD-Karte angezeigt werden. Beim Eintreffen neuer Rohdaten soll auch stets die Seekarte aktualisiert werden.

Empfang von Monitoring-Daten zur Routen-Planung und Fahrtüberwachung

Nachdem eine Route an die Routen- oder Pfad-Planung gesendet wurde, erfolgt der Empfang der Ergebnisdaten. Neben der Routendaten werden dabei auch im JSON-Format als String Hindernisdaten (Polygone) und Grid-Daten (blockierte und nicht-blockierte Zellen der verwendeten Algorithmen zur Wegfindung) übertragen. Dies sollte jeweils über einen fachlich von anderen Eingängen getrennten HTTP-Kontext erfolgen. So kann eine individuelle Verarbeitung der eingehenden Daten stattfinden. Die empfangenen Griddaten und Hindernis-Polygone müssen nach dem Empfang in Polygone geparst und zentral anderen Klassen zur Verfügung gestellt werden.

Visualisierung von Daten

Zur Visualisierung der empfangenen Daten bietet die EPD die Möglichkeit, eigene Layer zu erstellen und diese mit Informationen anzureichern. So können Informationen unterschiedlicher Datenquellen in der EPD voneinander getrennt angezeigt und separat zu- oder abgeschaltet werden.

Visualisierung von Monitoring-Daten zur Routen-Planung und Fahrtüberwachung

Zur Anzeige der Polygone (Anforderung „F-EPD-1.3“ und „F-EPD-1.4“) der Routen- und Pfad-Planung sollen eigene Layer angelegt und durch Methoden des MapView-Plugins der EPD registriert werden. So können fachlich unterschiedliche Daten separat in der Seekarte (auch einzeln) angezeigt werden. Dabei muss ein Mechanismus implementiert werden, der beim Speichern von neu eintreffenden Daten diese aus der zentralen Speichereinheit ausliest und in den Layer der EPD schreibt.

Visualisierung von Monitoring-Daten der dynamischen Hinderniserkennung

Zur Anzeige der Roh-Daten (Anforderung „F-EPD-1.1“ und „F-EPD-1.2“) der dynamischen Hindernisidentifikation (AIS- und Radar-Daten sowie Koordinaten von fusionierten Schiffen) soll ein eigener Layer angelegt und durch Methoden des MapView-Plugins der EPD registriert werden. So können fachlich unterschiedliche Daten separat in der Seekarte (auch einzeln) angezeigt werden. Dabei muss ein Mechanismus implementiert werden, der beim Speichern von neu eintreffenden Daten diese aus der zentralen Speichereinheit ausliest und in den Layer der EPD schreibt. Die Rohdaten sollen dabei voneinander unterscheidbar angezeigt werden, zum Beispiel durch unterschiedliche Farben.

Routen-Planung

In der PG MATE I wurde keine Validierung der globalen Route vorgenommen. Das heißt, dass im Küstenleitstand Routen-Wegpunkte beliebig gesetzt werden konnten, auch z.B. auf Land. Um eine Validierung dieser Route im Rahmen der PG MATE II umzusetzen, muss eine neue Komponente angelegt werden. Eine Übersicht über die Bestandteile dieser Komponenten ist in Abbildung Abbildung 5.2 ersichtlich.

Wie bereits in den Anforderungen aufgeführt, ist es zuerst notwendig die globale Route zusammen mit bestimmten Schiffsparametern von dem Küstenleitstand zu empfangen und zu speichern. Dann muss, mit Hilfe des NoGoSolvers, die Routen Validierung durchgeführt werden und die Route muss gegebenenfalls zu einer validen Route abgeändert werden. Schließlich muss diese abgeänderte Route an den Küstenleitstand zurückgeschickt werden.

Globale Route und Schiffsparamter vom Küstenleitstand empfangen und speichern

Bevor die Routen Validierung beginnen kann müssen die zu prüfende Route und die Schiffsparameter gemäß Anforderung F-CA-RP-2 von dem Küstenleitstand empfangen und gespeichert werden können. Hierzu bedarf es zunächst einer Trennung des Empfangs der Route und der Schiffsparameter. Während die Route über einen RouteInputHandler empfangen, in das benötigte Format übersetzt, und dann in einem RouteState abgespeichert werden soll, sollen die Schiffsparamter über einen ShipInputHandler empfangen, in das benötigte Format übersetzt, und dann in einem ShipState abgespeichert werden. Beide State-Klassen benötigen einen ChangeListener, welcher die Komponente benachrichtigen kann, wenn eine neue Route oder neue Schiffsparameter von dem Küstenleitstand empfangen wurden.

Validierung der Route mit Hilfe des NoGoSolvers

Liegen nun eine zu validierende Route und Schiffsparameter vor, soll mit der Validierung, und gegebenenfalls mit der Anpassung, der gegebenen Route begonnen werden. Hierzu soll zur Ermittlung der statischen Hindernisse der NoGoSolver, welcher statische Hindernisse aus dem angefragten Bereich aufgrund von Seekartendaten sendet, verwendet werden.

Anfrage an den NoGoSolver Wurden nun eine zu validierende Route und Schiffsparameter von dem Küstenleitstand empfangen, soll eine Anfrage an den NoGoSolver gesendet werden. Gemäß Anforderung F-CA-RP-1 soll die Routen-Planung Komponente dazu statische Hindernisse als Polygone von dem NoGoSolver anfragen, entgegennehmen und decodieren können. Dazu soll ein NoGoSolverClient implementiert werden, welcher die statischen Hindernisse in Form von Polygonen vom NoGoSolver anfragt. Die Anfrage besteht dabei aus den gegebenen Schiffsinformationen und dem Gebiet, aus welchem die statischen Hindernisse benötigt werden. Die Ergebnisse des NoGoSolvers sollen anschließend in einem eigens dafür erstelltem ObstacleState gespeichert werden. Diese ObstacleState-Klasse benötigen einen ChangeListener, welcher die Komponente benachrichtigen kann, wenn neue statische Hindernisse vom NoGoSolver empfangen wurden.

Aufbau der Routen Kalkulation Nach den Anforderungen F-CA-RP-3 und F-CA-RP-4 soll die Komponente zum einen über eine interne Repräsentation, welche das Gebiet für jeweils zwei aufeinanderfolgende Wegpunkte darstellt, verfügen und zum anderen einen Algorithmus bereitstellen, welcher eine valide Route berechnet, die nicht im Konflikt zu statischen Hindernissen steht. Die Routen Kalkulation, welche den Kern der Komponente darstellt, wurde nach der Doktor-Arbeit von Michael Blaich „Path Planning and Collision Avoidance for Safe Autonomous Vessel Navigation in Dynamic Environments“ aufgebaut [Bla16]. Sie besteht demnach aus einem A*-Algorithmus und einem Occupancy-Grid. Der A*-Algorithmus soll dabei die Berechnung einer validen Route zwischen einem gegebenen Startpunkt und einem gegebenen Zielpunkt, die nicht im Konflikt zu statischen Hindernissen steht, übernehmen. Das Occupancy-Grid soll innerhalb der Routen Kalkulation für die interne Repräsentation eines Gebiets für jeweils zwei aufeinanderfolgende Wegpunkte, also einem Start- und einem Zielpunkt, verantwortlich sein. Hierbei soll nicht nur auf das Grid selbst, sondern auch auf jede einzelne Zelle, aus der das Grid besteht, zugegriffen werden können.

Generierung einer validen Route Wurde der NoGoSolver angefragt und die statischen Hindernisse liegen in Form von Polygonen im ObstacleState vor, soll mit der Generierung einer validen Route begonnen werden. Hierzu soll zunächst das Occupancy-Grid initialisiert werden. Hiernach sollen die statischen Hindernisse in Form von Polygonen in das Grid geladen werden, d.h. es sollen alle Zellen markiert bzw. blockiert werden, welche von den Polygonen des ObstacleStates geschnitten oder umschlossen werden. Danach soll der A*-Algorithmus aufgerufen und eine valide Route berechnet werden.

Valide Route an den Küstenleitstand zurückschicken

Ist die Validierung der globalen Route abgeschlossen, soll die angepasste Route gemäß der Anforderung F-CA-RP-5 an den Küstenleitstand zurückgeschickt werden. Hierzu soll ein RouteOutputHandler implementiert werden, welcher lediglich die zu verschickende Route erhält und diese an den Küstenleitstand zurücksendet.

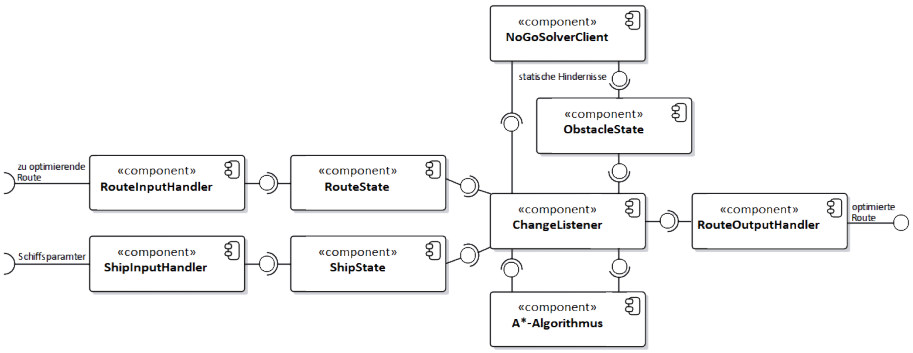


Abbildung 5.2.: Objektdiagramm Routenplanung Konzept

5.3. Kollisionsverhütung

Im Folgenden Kapitel wird das Konzept der Komponente zur Kollisionsverhütung näher erläutert.

Pfad-Planung

In der PG MATE I wurde keine Validierung des berechneten Pfades zum nächsten Wegpunkt vorgenommen. Das heißt, dass von der Pfad-Planung berechnete Pfade durch statische und dynamische Hindernisse führen können, z.B. durch Sandbänke oder Fremdschiffe. Um eine Validierung dieses Pfads umzusetzen, muss die, bereits durch MATE I implementierte, Pfad-Planungs-Komponente erweitert werden. Eine Gesamtübersicht der umgesetzten Komponentenbestandteile zeigt Abbildung 5.3. Wie bereits in den Anforderungen aufgeführt, ist es zuerst notwendig eine valide globale Route zusammen mit bestimmten Schiffsparametern von dem Küstenleitstand zu empfangen und zu speichern. Des Weiteren ist es notwendig statische Hindernisse vom NoGo-Solver und dynamische Hindernisse von der dynamischen Objektidentifizierung zu empfangen und zu speichern. Danach muss ein valider kollisionsfreier Pfad zum nächsten Wegpunkt berechnet werden. Schließlich muss der berechnete valide Pfad an die EPD und an das Regelungssystem geschickt werden.

Identifikation von statischen Hindernissen

Im Rahmen der maritimen Pfadplanung ist es natürlich notwendig statische Hindernisse, wie beispielsweise Land, Untiefen oder Bojen zu identifizieren. Es ist hier aus offensichtlichen Gründen durchaus sinnvoll andere Möglichkeiten in Betracht zu ziehen als Radar oder Kamera um solche Hindernisse zu identifizieren.

In der modernen Schifffahrt werden bereits digitale Seekarten im S-57 Standard zur Navigation verwendet [IHO]. Solche Karten bieten die Möglichkeit jegliche statische Hindernisse abzufragen. Weiterhin sollten bei der Bewertung von statischen Hindernissen bestimmte Schiffsparameter (insbesondere: Länge, Breite, Höhe und Tiefgang) in Betracht gezogen werden. Je nach Tiefgang eines Schiffes, kann eine Untiefe von 1,5 Meter Tiefe passiert werden oder sollte gemieden werden. Unbefahrbare statische Gebiete werden auch als sogenannte *NoGo-Areas* bezeichnet. Es besteht also der Bedarf nach einer Lösung um abhängig von den eigenen Schiffsparametern statische Hindernisse aus existierenden Kartendaten zu berechnen. Genau diese Funktionalität bietet der im OFFIS e.V. bereits entwickelte *NoGo-Solver* an. Basierend auf S-57 Kartendaten im GML-Format löst er statische Hindernisse basierend auf Schiffsparametern auf und stellt diese im WKT-Format zur Verfügung. Die Kommunikation findet hierbei durch eine JSON-Schnittstelle über Websockets statt.

Valide globale Route und Schiffsparameter von dem Küstenleitstand empfangen und speichern

Bevor die Berechnung eines validen kollisionsfreien Pfads beginnen kann müssen die zu prüfende Route und die Schiffsparameter gemäß Anforderung F-CA-PP-3 von dem Küstenleitstand empfangen und gespeichert werden können. Hierzu bedarf es zunächst einer Trennung des Empfangs der validen globalen Route und der Schiffsparameter. Während die valide globale Route über einen RouteInputHandler empfangen, in das benötigte Format übersetzt, und dann in einen RouteState abgespeichert werden soll, sollen die Schiffsparamter über einen ShipInputHandler empfangen, in das benötigte Format übersetzt, und dann in einem ShipState abgespeichert werden. Beide State-Klassen benötigen einen ChangeListener, welcher die Komponente benachrichtigen kann, wenn eine neue Route oder neue Schiffsparameter von dem Küstenleitstand empfangen wurden. Wurde eine neue valide globale Route von dem Küstenleitstand empfangen, soll dies zudem durch eine GlobaleRouteReceived-Message bestätigt werden. Hierfür soll ein GlobalRouteReceivedOutputHandler implementiert werden.

Aktuelle Position und aktuelles Heading vom Distributionssystem empfangen und speichern

Da auch die aktuelle Position sowie das aktuelle Heading zur Berechnung eines validen kollisionsfreien Pfads nach F-CA-PP-4 benötigt werden, müssen diese vom Distributionssystem entgegengenommen und decodiert werden können. Der Empfang der aktuellen Position wurde bereits

von MATE I in Form eines PositionInputHandlers umgesetzt. Diese Entgegennahme der Position wurde jedoch mittels HTTP realisiert. Da jedoch nicht jede vom Distributionssystem übermittelte Position von der Pfad-Planungs-Komponente benötigt wird, wurde ein UDP-InputHandler implementiert welcher sowohl zum Empfang der aktuellen Position als auch der aktuellen Bewegungsrichtung vom Distributionssystem zuständig ist. Zur Erfüllung der Anforderung F-CA-PP-13 soll zudem ein TimeOutListener realisiert werden, welcher einen Verbindungsabbruch zum Distributionssystem erkennt und ggf. versucht eine erneute Verbindung herzustellen.

Statische Hindernisse vom NoGoSolver anfragen, empfangen und speichern

Liegen eine valide globale Route, Schiffsparameter, eine aktuelle Position und ein aktuelles Heading vor, soll mit der Ermittlung der statischen Hindernisse mit Hilfe des NoGoSolvers, welcher statische Hindernisse aus dem angefragten Bereich aufgrund von Seekartendaten sendet, begonnen werden. Gemäß Anforderung F-CA-PP-1 soll die Pfad-Planungs-Komponente dazu statische Hindernisse als Polygone von dem NoGoSolver anfragen, entgegennehmen und decodieren können. Dazu soll ein NoGoSolverClient implementiert werden, welcher die statischen Hindernisse in Form von Polygonen vom NoGoSolver anfragt. Die Anfrage besteht dabei aus den gegebenen Schiffsinformationen und dem Gebiet, aus welchem die statischen Hindernisse benötigt werden. Die Ergebnisse des NoGoSolvers sollen anschließend in einem eigens dafür erstellten StaticObstaclState gespeichert werden. Diese StaticObstaclState-Klasse benötigt einen ChangeListener, welcher die Komponente benachrichtigen kann, wenn neue statische Hindernisse vom NoGoSolver empfangen wurden. Zur Erfüllung der Anforderung F-CA-PP-11 soll zudem ein TimeOutListener realisiert werden, welcher einen Verbindungsabbruch zum NoGoSolver erkennt und ggf. versucht eine erneute Verbindung herzustellen.

Dynamische Hindernisse von der dynamischen Objektidentifizierung empfangen und speichern

Um mit der Berechnung eines validen kollisionsfreien Pfads beginnen zu können müssen nun noch gemäß Anforderung F-CA-PP-2 dynamische Hindernisse von der dynamischen Objektidentifizierung entgegengenommen und decodiert werden können. Dazu soll ein DynamicObjectInputHandler implementiert werden, welcher dynamische Hindernisse in Form von eigens erstellten Ship-Objects von der dynamischen Objektidentifizierung empfängt. Die Ship-Objects sollen anschließend in einem eigens dafür erstelltem DynamicObstaclState gespeichert werden.

Diese DynamicObstaclState-Klasse benötigt einen ChangeListener, welcher die Komponente benachrichtigen kann, wenn neue dynamische Hindernisse empfangen wurden. Zur Erfüllung der Anforderung F-CA-PP-12 soll zudem ein TimeOutListener realisiert werden, welcher einen Verbindungsabbruch zur dynamischen Objektidentifizierung erkennt und ggf. versucht eine erneute Verbindung herzustellen.

Berechnung eines validen kollisionsfreien Pfads zum nächsten Wegpunkt

Liegen eine valide globale Route, statische und dynamische Hindernisse vor, soll mit der Berechnung eines validen kollisionsfreien Pfads zum nächsten Wegpunkt begonnen werden.

Aufbau der Pfad Kalkulation Nach den Anforderungen F-CA-PP-5 und F-CA-PP-6 soll die Komponente zum einen über eine interne Repräsentation, welche das Gebiet für jeweils zwei aufeinanderfolgende Wegpunkte darstellt, verfügen und zum anderen einen Algorithmus bereitstellen, welcher einen validen kollisionsfreien Pfad berechnet, der nicht im Konflikt zu statischen oder dynamischen Hindernissen steht und zum nächsten Wegpunkt. Die Pfad Kalkulation, welche den Kern der Komponente darstellt, wurde nach der Doktor-Arbeit von Michael Blaich "Path Planning and Collision Avoidance for Safe Autonomous Vessel Navigation in Dynamic Environments" [Bla16]. aufgebaut. Sie besteht demnach aus einem A*-Algorithmus und einem Occupancy-Grid. Der A*-Algorithmus soll dabei die Berechnung einer validen Route zwischen einem gegebenen Startpunkt und einem gegebenen Zielpunkt, die nicht im Konflikt zu statischen oder dynamischen Hindernissen steht, übernehmen. Das Occupancy-Grid soll innerhalb der Pfad Kalkulation für die interne Repräsentation eins Gebiets für jeweils zwei aufeinanderfolgende Wegpunkte, also einem Start- und einem Zielpunkt, verantwortlich sein. Hierbei soll nicht nur auf das Grid selbst, sondern auch auf jede einzelne Zelle, aus der das Grid besteht, zugegriffen werden können.

Generierung eines validen kollisionsfreien Pfads Zur Generierung eines validen kollisionsfreien Pfads soll zunächst das Occupancy-Grid initialisiert werden. Hiernach sollen die statischen und dynamischen Hindernisse in Form von Polygonen in das Grid geladen werden, d.h. es sollen alle Zellen markiert bzw. blockiert werden, welche von den Polygonen des StaticObstacleStates und des DynamicObstacleStates geschnitten oder umschlossen werden. Danach soll der A*-Algorithmus aufgerufen und ein valider und kollisionsfreier Pfad berechnet werden.
Nach Anforderung F-CA-PP-7 bedeutet valide in diesem Fall ebenfalls, dass bei der Berechnung des Pfads die physikalischen Eigenschaften der Zuse berücksichtigt werden. Dies soll ebenfalls

nach Michael Blaich [Bla16] mit Hilfe des T-Neighbourhood-Konzepts, welches bereits in Abschnitt 3.4 vorgestellt wurde, realisiert werden. Die Berechnung des Pfades muss ebenfalls eine besondere Beachtung der dynamischen Hindernisse beinhalten. Nach Anforderung F-CA-PP-8 muss die Pfad-Planungs-Komponente bei der Berechnung eines validen kollisionsfreien Pfads die COLREGs Regeln 13 bis 17 einhalten. Dies soll mit zweierlei Konzepten umgesetzt werden. Zum einen soll für die Beachtung der Fremdschiff-Domäne, also des Kollisionsgefahrenbereich des Fremdschiffes, nach Micheal Blaichs Artikel "Extended Grid Based Collision Avoidance Considering COLREGs for Vessels", das Goodwin-Konzept, welches bereits in Abschnitt 3.4 vorgestellt wurde, umgesetzt werden. Des Weiteren soll für die Vorausprojektion der Position der Fremdschiffe nach Michael Blaichs Doktor-Arbeit [Bla16] der CPA-Ansatz, welcher bereits in Abschnitt 3.4 vorgestellt wurde, implementiert werden.

Validen und kollisionsfreien Pfad an Controller und Küstenleitstand senden

Ist die Berechnung eines validen und kollisionsfreien Pfads abgeschlossen, soll der Pfad gemäß der Anforderung F-CA-PP-9 an den Controller, und gemäß Anforderung F-CA-PP-10 and den Küstenleitstand gesendet werden. Zur Versendung des Pfads an den Controller sollen zwei berechnete Wegpunkte über einen WaypointsOutputHandler an den Controller gesendet werden, was bereits von MATE I realisiert wurde, während zur Versendung des Pfads an den Küstenleitstand alle berechneten Wegpunkte an diese gesendet werden (ebenfalls über den WaypointsOutputHandler).

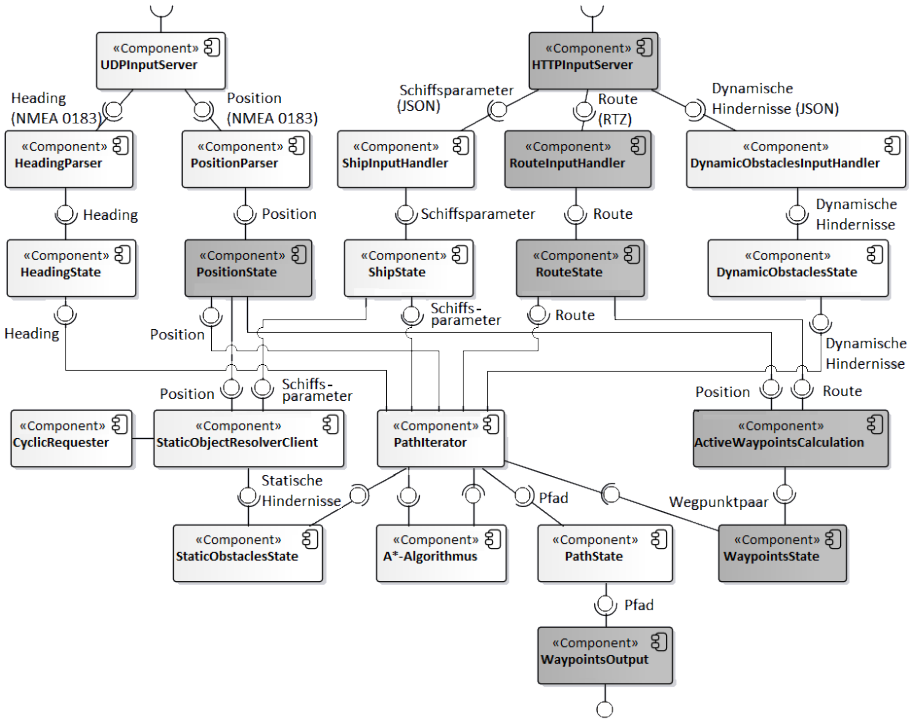


Abbildung 5.3.: Objektdiagramm Pfad-Planung Konzept

5.4. Hindernisidentifikation

In der vorangehenden Projektgruppe MATE I wurden keine dynamischen Hindernisse betrachtet. Um eine Identifikation von dynamischen Hindernissen zu ermöglichen, müssen dynamische Objekte mittels Sensoren erkannt und diese Informationen in einer neuen Komponente verarbeitet werden. Zweck der Objektidentifikation ist es, einer Pfad-Planung alle wichtigen Informationen zu den dynamischen Objekten in der Umgebung zu senden, damit diese hierauf angemessen reagieren kann und Kollisionen vermeidet.
Die Objektidentifikation hat hierbei das Ziel, ein möglichst genaues Bild von der Umwelt zu bekommen, d.h. es sollen Informationen wie die Position von allen Schiffen in der Umgebung bereitgestellt werden.
Jeder Sensor besitzt allerdings verschiedene Vor- und Nachteile, um die Nachteile auszugleichen, müssen unterschiedliche Sensoren eingesetzt werden. Um die Informationen aller eingesetzten Sensoren zu verwenden, muss eine Datenfusion erfolgen. Es ergibt sich dadurch ein besseres Ergebnis zur Umweltwahrnehmung. [Bal14]

Hierbei stellen sich folgende Fragen, die im weiteren Verlauf geklärt werden sollen: Welche Sensoren sollen eingesetzt werden? Wie sollen die Sensordaten verarbeitet werden? Wie können hiermit die Anforderungen (siehe Abschnitt 4.6) erfüllt werden?

Sensorauswahl

Die für die Objekterkennung nützlichen Sensoren, die sich auf der Zuse befinden, sind Radar, Automatisches Identifikationssystem (AIS), Kamera und Echolot. Für eine Erklärung der Funktionsweise dieser Sensoren siehe Kapitel Maritime Sensoren. Bei der Auswahl der Sensoren wurde sich auf die zur Verfügung stehenden Sensoren beschränkt. Daher wurden weitere Sensoren nicht betrachtet.
Neben den oben genannten Sensoren wäre es möglich, ein vorhandenes Lidar-System auf der Zuse zu installieren. Es wurde sich aber dazu entschieden, das vorhandene Lidar nicht zu nutzen, da es nur eine Reichweite von maximal 50 Meter hat und nur eine Ebene betrachten kann. Dies ist für die gegebene Situation nicht ausreichend. [LMS1]
Das Echolot wird ebenfalls nicht für die dynamische Hindernisidentifizierung genutzt, da es lediglich Objekte unter Wasser erkennt. Die dynamische Objektidentifizierung konzentriert sich im Rahmen dieser Projektgruppe allerdings auf die Erkennung von Schiffen oberhalb der Wasseroberfläche.
Da das AIS meist sehr genaue Angaben bezüglich Position, Kurs, Geschwindigkeit etc. sendet, wurde entschieden, dieses für die dynamische Objektidentifizierung einzusetzen. Auf kleineren Schiffen ist allerdings meist kein AIS verbaut, deshalb würde ein AIS allein nicht ausreichen. Kleinere Schiffe werden, genauso wie größere Schiffe vom Radar erkannt. Probleme bei der Objektidentifizierung mit Radar sind, dass nicht jedes Objekt die gleichen Rückstrahleigenschaften besitzt. So können Material, Oberflächenstruktur, Form und Größe des Objekts Einfluss darauf haben, ob und wie stark sie als Radarziel angezeigt werden. Zudem kann ein Hindernis (z. B. reflektierendes Objekt) einen Radarschatten auf dem Radarbild produzieren. Wodurch Objekte, die sich in diesem Bereich befinden, nur schlecht (in Schattensektoren) oder gar nicht (in Blindsektoren) erkannt werden. [BH10, S.173].
Um alles zu überblicken und ein gutes Ergebnis zu erzielen, sollten daher am Besten beide Sensoren gemeinsam genutzt werden. Um zusätzlich die Performance bezogen auf beispielsweise den Ausfall eines Sensors und das Vertrauen durch Bestätigung eines Ziels zu verbessern, sollte eine Kamera eingesetzt werden. Diese kann verwendet werden, um den Bereich vor dem Schiff zu überwachen.

Optischer Sensor

Um mit einer Kamera das Umfeld vor dem eigenen Schiff zu beobachten, muss eine Analyse des aufgenommenen Bildmaterials der Kamera im Bezug auf eine mögliche Kollision stattfinden. Laut Anforderung F-OD-2 wird sich hierbei auf andere Schiffe in Fahrtrichtung konzentriert, alle anderen Hindernisse werden im Rahmen dieser Projektgruppe nicht betrachtet.
Es gibt zwei grundlegend verschiedene Ansätze wichtige Informationen aus diesen Bildern herauszufiltern. Zum einen kann auf die „klassische“ Bildverarbeitung zurückgegriffen werden. Auf der anderen Seite kann maschinelles Lernen zum Analysieren des Bildmaterials verwendet werden. Wichtige Faktoren für die Entscheidung sind die Genauigkeit der Analyseverfahren, aber auch die Verarbeitungsgeschwindigkeit dieser.
Die „klassische“ Bildverarbeitung wurde von der Projektgruppe „MOPS 4“ (2015-2016) eingesetzt. Um relevante Hindernisse zu detektieren, haben sie den Ansatz gewählt, zuerst Verzerrungen aus dem Bild zu entzerren, dann den Horizont zu erkennen, Lichtreflexionen zu entfernen und schließlich Kanten zu detektieren, um so Objekte im Bild zu finden [IV16, S.134]. Vorteile dieses Verfahren sind, dass die Ergebnisse deterministisch und datenunabhängig sind.
Eine weitere Herangehensweise Objekte mithilfe einer Kamera zu erkennen, ist das maschinelle Lernen. Google hat beispielsweise die open-source Bibliothek „Tensorflow“ entwickelt, welche es ermöglicht, ein künstliches neuronales Netz zu erzeugen, das mittels maschinellem Lernen zuerst trainiert werden kann und dadurch später die gewünschten Objekte im Bild erkennt. [Fra17] Diese Art der Objektdetektion kann neben der Information, das ein Objekt im Bild vorhanden ist, zusätzlich die Information ausgegeben, welches Objekt zu sehen ist. Es kann somit zum Beispiel zwischen einem Segelboot und einem Frachtschiff unterscheiden. Die hieraus errungenen Informationen können bei der Kollisionsvermeidung Beachtung finden, um das Schiff nicht regelwidrig ausweichen zulassen (Kollisionsverhütungsregel 18).

Verarbeitung der Sensordaten - Datenfusion

Für ein autonomes Schiff sind bestimmte sicherheitskritische Anforderungen zu beachten. Um anderen Objekten auszuweichen ist es wichtig, die Genauigkeit und Störanfälligkeit bei der Erkennung dieser Hindernisse zu verbessern. Daher werden, wie oben beschrieben, mehrere Sensoren eingesetzt. Diese Informationen von verschiedenen Sensoren müssen fusioniert werden, um nicht jedes Schiff mehrmals zu betrachten. Hierbei beschreibt eine Multisensor-Datenfusion die Zusammenführung und Kombination verschiedener Sensoren.

Wichtige Vorteile dieser Kombination bezogen auf das Projekt sind nach [Kla06] unter anderem:

- Erweiterung der räumlichen Abdeckung
- Robuste operationelle Performanz (z.B. bei Ausfall eines Sensors)
- Erhöhtes Vertrauen (z.B. Bestätigung eines Ziels)
- Verbesserte Verlässlichkeit des Systems

Es gibt zwei verschiedene Arten der Datenfusion. Zum einen die Sensorfusion und zum anderen die Sensordatenfusion. Die Sensorfusion wertet Informationen aus mehreren Sensoren zu umliegenden Objekten gemeinsam aus, während die Sensordatenfusion (siehe Abschnitt 3.7) bereits verarbeitete und ausgewertete Informationen über die Umgebung und die gefundenen Objekte verwendet. [Rei14, S.340]
Eine Datenfusion kann in zwei Teile geteilt werden. Die Datenassoziation und die Datenregression. Bei der Assoziation werden verschiedene Daten unterschiedlicher Sensoren zusammengefasst. Die Datenregression gleicht die Datenwerte der zusammengeführten Daten an.
Durch diese Vorteile lässt sich der Nutzen der Datenfusion für das Projekt ableiten. Die Genauigkeit von Schlussfolgerungen wie bei der Schätzung von Positionszuständen kann verbessert werden, da durch eine Verknüpfung mehrerer Sensoren die Unsicherheit der geschätzten Position verringert wird.
Das Ziel der Datenfusion ist, alle umliegenden erkannten Schiffe, die dasselbe Schiff sind, als solches zu identifizieren und möglichst genaue Informationen zu dem Schiff zu sammeln bzw. zu generieren.
Zu beachten ist hierbei allerdings, dass die Kamera keine Information zur Distanz der Schiffe hat. Daher können die Kameradaten nicht mit fusioniert werden, sondern müssen separat behandelt werden (siehe Abschnitt Optischer Sensor).

Stand der Forschung: Datenfusion AIS und Radar

In der Literatur gibt es verschiedene Möglichkeiten, eine Datenfusion durchzuführen. Im Folgenden werden drei repräsentative Arbeiten mit ihren Ansätzen vorgestellt.

In dem Paper „Study of Data Fusion of AIS and Radar“ stellen Liu Chang und Shi Xiaofei Fuzzy Functions als eine Art der Datenfusion vor [CX09]. Hierfür werden Funktionen für Breitengrad, Längengrad, Kurs und Geschwindigkeit aufgestellt (siehe Abbildung 5.4). Es wird für jeden dieser Werte die Differenz von den zwei betrachtenden Schiffen gebildet und durch die entsprechende Funktion ein Assoziationsgrad g bestimmt.

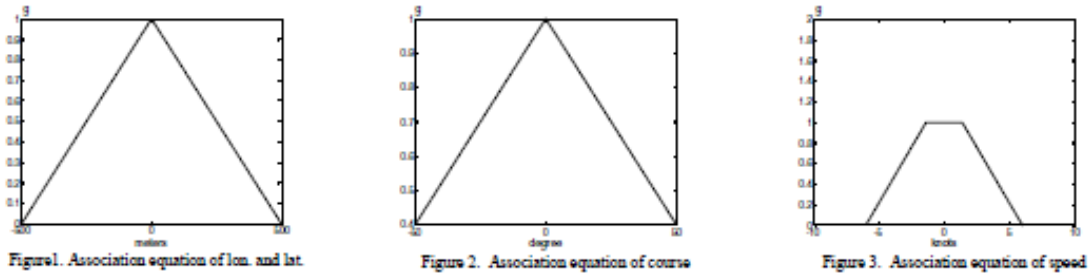


Abbildung 5.4.: Fuzzy-Funktionen für Breitengrad, Längengrad, Kurs und Geschwindigkeit (aus [CX09])

Wenn dieser Grad g für alle Werte größer als ein gegebener Schwellwert $g0$ ist, dann werden die zwei Schiffe als dasselbe Objekt angenommen (siehe Abbildung 5.5). Laut Changs und Xiaofeis Testdaten ist dies eine akkurate Methode, die Schiffe zu fusionieren. [CX09]

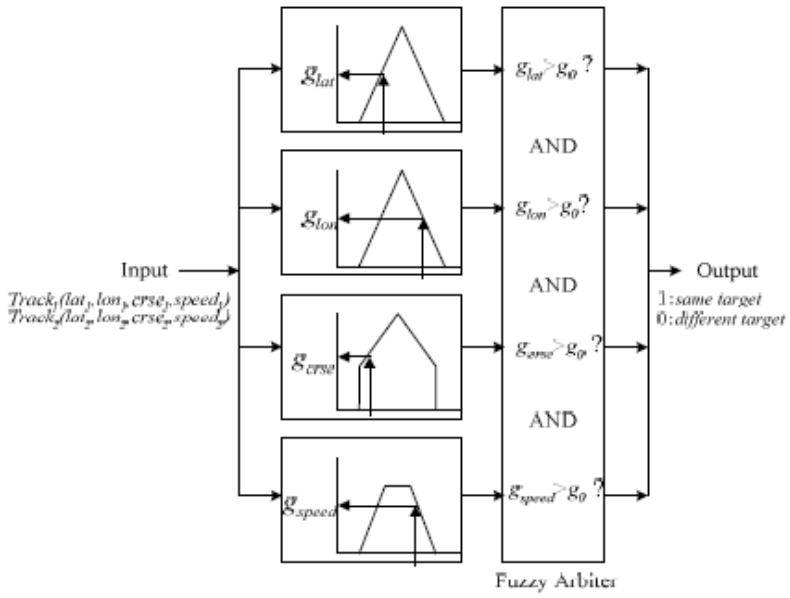


Abbildung 5.5.: Fuzzy Assoziation (aus [CX09])

Einen weiteren möglichen Ansatz beschreibt Witold Kazimierski in „Problems of Data Fusion of Tracking Radar and AIS for the Needs of Integrated Navigation Systems at Sea“ [Kaz13].

Neben einer Vielzahl von Problemen, die bei der Fusionierung von AIS und Radar auftreten können, wird außerdem deutlich, dass zur Datenfusionierung dieser Sensoren meist Filter eingesetzt werden, die auf dem Kalman Filter basieren. [Kaz13]
Dieses Prinzip wird auch in „Radar and AIS Data Fusion for the Needs of the Maritime Navigation“ von Andrzej Stateczny und Andrzey Lisaj beschrieben [SL06].

Ein dritter Ansatz wird in der Bachelorarbeit „Target Association von Schiffstracks unterschiedlicher maritimer Sensoren zur Vorbereitung auf die Datenfusion heterogener Beobachtungsdatensätze“ von Rudolph Greub dargestellt [Gre17]. Hier wird die Distanz zwischen zwei Beobachtungen mittels Harversine-Formel berechnet und dieser Wert als Ähnlichkeitsmaß angenommen. Falls mehrere Kombinationen ein Ähnlichkeitsmaß kleiner eines gesetzten Schwellwerts haben, wird in dem Fall die Zeit hinzugezogen und die Zeitdifferenz mit einem Schwellwert verglichen. Aus den verbliebenen Kombinationen wird diejenige ausgewählt, die den nächsten Nachbarn hat. [Gre17, S.43f.]

Laut Fazit führt dieser Ansatz allerdings nur zu einem gemischten Erfolg, da es bei mehr als 90 Prozent zu Mehrfachassoziationen kommt [Gre17, S.60].
Dies sind alles Konzepte zur Assoziation. Für die Regression können im Allgemeinen verschiedene Filter verwendet werden, um die Daten untereinander anzugleichen. Mögliche Filter können der Kalman-Filter, der Alpha-Beta-Gamma-Filter oder auch der Alpha-Beta-Filter sein. Über den konkreten Einsatz eines solchen Filters, bezogen auf der Fusionierung von AIS und Radar, ist nichts genaueres in der Literatur zu finden.

Entwurf der Komponenten

Aufbauend auf der Sensorauswahl und den oben beschriebenen möglichen Ansätzen zur Hindernisidentifikation, wird im Folgenden der Entwurf der Hindernisidentifikation beschrieben. Hierbei sollen die Bildverarbeitung und die Datenfusion fachlich getrennt voneinander in zwei verschiedenen Komponenten behandelt werden. Wie diese Komponenten zusammenspielen, zeigt das Komponentendiagramm Abbildung 5.6. Die Bildverarbeitung wird in der Komponente Objektdetektion durchgeführt und die Datenfusion in der Komponente Dynamische Objektidentifizierung.

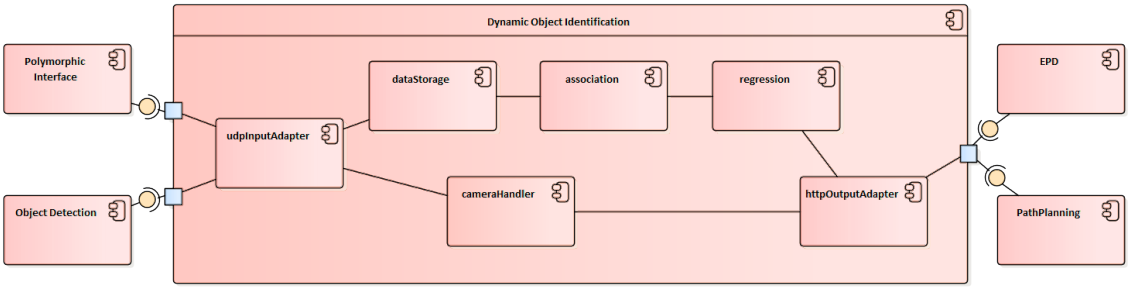


Abbildung 5.6.: Komponentendiagramm zum Entwurf der Hindernisidentifikation

Objektdetektion

Um die Anforderung F-OD-1 zu erfüllen, muss eine Kamera die Umgebung vor dem Schiff erfassen und als Videostream an die Objektdetektion übertragen. Zur Detektion von Schiffen in dem Bildmaterial (Anforderung F-OD-1) in Echtzeit wird ein vortrainiertes neuronales Netz verwendet, welches alle Objekte im Bild erkennt, mit einem Label versieht und die Position im Bild wiedergibt. Hierdurch kann erkannt werden, wo genau im Bild, mit einer angegebenen prozentualen Wahrscheinlichkeit, ein Schiff zu sehen ist. Da ein Mono-Video keine Tiefeninformationen beinhaltet, besteht das Problem, die Position der detektierten Schiffe zu bestimmten. Für die Position soll daher ein fester Wert an die Pfad-Planungs-Komponente weitergeleitet werden. Um die mögliche Position trotzdem weiter einzuschränken, werden alle Schiffe über dem Horizont herausgefiltert (Anforderung F-OD-3), da diese kein relevantes Hindernis darstellen (Anforderung F-OD-4). Sie sind entweder doch kein Schiff oder zu weit weg, um ihnen schon ausweichen zu müssen. Diese Analyse des Bildmaterials ist unabhängig von der Fusionierung der Sensordaten und wird daher in einer eigenen Komponente ausgelagert.
Um zusätzlich einen Abgleich der erkannten Schiffe mit Radar- und AIS-Informationen zu umliegenden Schiffen durchzuführen, müssen die Informationen zu der Position an die Dynamische Objektidentifizierung gesendet werden. Hierfür wird ein OutputAdapter benötigt, welcher die Positionsdaten via UDP an die Dynamische Objektidentifizierung sendet.

Dynamische Objektidentifizierung

Die Dynamische Objektidentifizierung ist zuständig für die Datenfusion verschiedener Sensoren. Um dies Umzusetzen, muss die Komponente eingehende Sensordaten verarbeiten können und auf Basis dieser Daten eine Datenfusion durchführen. Zusätzlich muss eine Schnittstelle zur Kommunikation implementiert werden. Alle benötigten Komponenten werden nachfolgend gegliedert in Datenmanagement, Datenfusion und Kommunikation genauer betrachtet.

Datenmanagement Als Vorbereitung für die Datenfusion müssen zunächst Daten der Sensoren empfangen, vorverarbeitet und gespeichert werden, sodass eine Datenfusion auf den Daten durchgeführt werden kann. Hierbei existieren nicht nur die Daten der realen Sensoren, sondern zu Testzwecken ebenfalls Livedaten der Küste sowie simulierte Daten. Diese Daten sollen im Datenformat NMEA2000 via UDP von der Polymorphen Schnittstelle an die Dynamische Objektidentifzierung übertragen werden. Um die Anforderung F-DOI-1, F-DOI-2 sowie F-DOI-3 zu erfüllen, müssen Daten des AIS und des Radar empfangen werden, zusätzlich sind Daten der eigenen Position sowie des eigenen Kurs relevant. Hierfür müssen eine NMEA2000 konforme Radarnachricht und eine Transformationsregel der S100 Radarnachricht zur NMEA2000 konformen Radarnachricht erstellt werden.
Um nicht zu viel Rechenzeit und Speicher zu benötigen, ist es ausreichend, die Daten in einem bestimmten Umkreis zu betrachten (Anforderung F-DOI-4), sowie nur die relevanten Informationen aus den Daten zu speichern. Hierbei sollen Daten über die Größe, Position, Geschwindigkeit und Kurs ausgelesen werden (Anforderung F-DOI-5).

Datenfusion Im Anschluss müssen die gespeicherten Daten fusioniert werden. Hierfür wird zunächst eine Assoziation und im Anschluss eine Regression durchgeführt.
Bei der Assoziation soll festgestellt werden, welches mittels AIS erkannte Schiff dasselbe Schiff wie welches mittels Radars erkannte Schiff ist (Erfüllung der Anforderung F-DOI-6 und F-DOI-7). Hierbei sollen entweder die Fuzzy Funktionen eingesetzt werden, oder die Positionen über die Zeit verglichen werden. Da die Radardaten sehr ungenau sind, müssen diese im Voraus bereits gefiltert werden.
Aufbauend auf dieser Assoziation werden die verschiedenen Werte der zusammengehörigen Radar- und AIS-Daten angeglichen, das heißt, das sogenannte „Rauschen“ der gemessenen Daten herausrechnen. Um die Anforderung F-DOI-8 zu erfüllen, müssen die korrekten Werte für Schiffslänge, -breite, Position, Kurs und Geschwindigkeit, der sich im Umfeld des eigenen Schiffs befindlichen Objekte, geschätzt werden. Da keine Testdaten zum Validieren des Systems vorhanden sind, ist der Einsatz von Filtern für die Regression nicht zu empfehlen. Hierfür wären physikalische Daten von Schiffen erforderlich (mit Radar und AIS gemessen), bei denen genau festzustellen ist, wie groß das „Rauschen“ ist, also welche die exakt richtigen Werte für Schiffsgröße, -breite, Position, Kurs und Geschwindigkeit sind. Da diese nicht vorhanden sind, lässt sich die Richtigkeit der geschätzten Werte auf Basis der statistischen Analyseverfahren nicht gewährleisten. Durch diese fehlende Testabdeckung sind die genannten Filter für ein sicherheitskritisches System nicht zu empfehlen und werden für die Datenregression nicht weiter betrachtet.

Für ein solches sicherheitskritisches System ist es auch nicht vom Vorteil, nur einem Sensor zu vertrauen und allein diese Daten zu verwenden. Somit ist wichtig, die Daten beider Schiffe zu kombinieren, ohne beispielsweise die Positionsdaten eines Sensors zu verwerfen. Deshalb wird bei der Schätzung der Daten der Ansatz verfolgt die Schiffsgröße bei ungenauen Werten zu vergrößern, sodass das Schiff definitiv in dem jeweiligen Bereich, der an die Pfad-Planungs-Komponente geschickt wird, zu finden ist. Des Weiteren werden auch nicht assoziierte Schiffe, also Schiffe bei denen nur Daten der Radarnachrichten oder AIS-Nachrichten vorhanden sind, an die Pfad-Planungs-Komponente weitergeleitet.

Kommunikation Abschließend müssen die Informationen der erkannten und fusionierten Schiffe zur Beobachtung an einen küstenseitigen Leitstand gesendet werden. Zur Erfüllung der Anforderung F-DOI-9 müssen die dynamischen Hindernisse auch an eine Pfad-Planungs-Komponente gesendet werden, damit diese bei der Planung des Pfades auf die dynamischen Hindernisse reagieren kann. Um richtig auf die Hindernisse reagieren zu können, müssen neben der Position auch Daten über die Größe, den Kurs und der Geschwindigkeit versendet werden. Versendet werden sollen die Daten als JSON-Objekt via UDP. Das JSON-Objekt soll eine Arrayliste mit allen relevanten, fusionierten Objekten enthalten.

5.5. Regelungstechnik

Die Grundlage für die autonome Fahrt ist ein Regelungssystem, das in der Lage ist das Schiff ohne Eingriff eines Menschen sowohl Geschwindigkeit als auch Ruderwinkel anzupassen. Um diese Parameter angemessen einstellen zu können benötigt die Regelungskomponente Eingaben wie die aktuelle Geschwindigkeit, die eigene Position, die gewünschte Position und die gewünschte Geschwindigkeit. Die aus diesen, und weiteren, Parametern berechneten Werte können anschließend an die Schiffssteuerung übergeben werden. Diese setzen die Befehle um und das Schiff wird mit der gewünschten Geschwindigkeit in die gewünschte Richtung bewegt.

Die Komponente an sich bestand bereits aus MATE I. Jedoch wurde sie fehlerhaft an MATE II übergeben. Anstatt in die gewünschte Richtung regelte das System immer nur in die gleiche Richtung mit maximal eingeschlagenem Ruderwinkel.

Line of Sight Controller

Eine neue Komponente des Regelungssystems stellt hingegen der Line of Sight Controller (kurz: LOS Controller) dar. Der zugrunde liegende Gedanke ist, dass der Abstand zu der Gerade, die zwischen zwei Wegpunkten aufgespannt wird, minimiert werden soll.
Abbildung 5.7 verdeutlicht das Prinzip des Line of Sight Controllers. p_{k-1} beschreibt den bereits erreichten Wegpunkt, p_k den aktuell anzufahrenden. Zwischen diesen beiden Wegpunkten befindet sich die aktuelle Postion des Schiffes p. Um diese Postion wird ein Kreis von n-Schiffslängen (nL_{pp}) gelegt. Dieser schneidet den aktuellen Pfad zwischen den beiden Wegpunkten an zwei Stellen. Es wird der Schnittpunkt ausgewählt, der näher an dem anzufahrenden Wegpunkt liegt (p_{los}). Um Konvergenz in Richtung des gewünschten Pfads zu erreichen muss der Gierwinkel Ψ_{los} angepasst werden, sodass Kurs auf p_{los} genommen wird. [FBS03]
Dies geschieht nun nicht mehr, wie bisher, mithilfe eines PID-Reglers, sondern durch die Berechnung einer einfachen Gleichung.

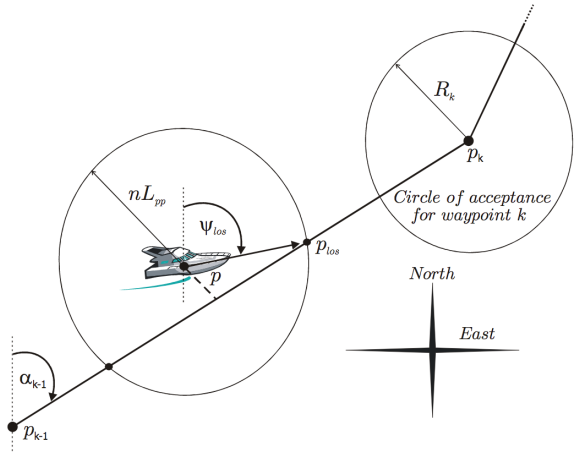


Abbildung 5.7.: Schematische Darstellung eines Line of Sight controllers ([FBS03])

5.6. V&V-Werkzeug

Während der Planung des Testprozesses des MATE II Produktes ergaben sich einige Herausforderungen im Bereich der Validierung und Verifikation. Die große Anzahl an Datenströmen und sonstiger Kommunikation zwischen den verschiedenen Komponenten manuell durch den Prozess der Verifikation und Validierung ohne weitere Hilfsmittel zu führen ist nur schwer möglich und mit einem hohen Zeitaufwand verbunden.

Es ist also sinnvoll ein Werkzeug zu entwickeln, welches Schnittstellen zur Automatisierung dieser Prozesse zur Verfügung stellt. Folgende Anforderungen müssen an ein solches Werkzeug gestellt werden können:

- Das Werkzeug muss eine Übersicht über den aktuellen Zustand aller Systemkomponenten bieten. Besonders Ausfälle einer Komponente müssen erkannt werden können.
- Es muss die Möglichkeit bestehen die Log-Ausgaben aller Systemkomponenten zu überwachen
- Es muss die Möglichkeit bestehen ausgewählte zeitabhängige interne Zustände aller Systemkomponenten zu überwachen.
- Das Werkzeug muss Datenströme im System erkennen, decodieren und lesbar oder grafisch darstellen können.
- Das Werkzeug muss fähig sein Datenströme syntaktisch korrekt zu modifizieren und Werte innerhalb der Datenströme zu ändern (Fehlerinjektion).
- Zur Verifikation und Validierung von Daten durch Experten muss das Werkzeug eine grafische Benutzeroberfläche bereitstellen.
- Das Werkzeug muss fähig sein die Leistung des Gesamtsystems während der Laufzeit zu bewerten.

5.7. Integration simulierter Daten ins physikalische Testfeld

Für die Integration der simulierten Daten in das physikalische Testfeld wird eine Komponente benötigt, die die Interaktion zwischen HAGGIS und LABSKAUS gewährleistet. Im Folgenden wird diese Komponente als Simulationsadapter bezeichnet.
Mittels HAGGIS ist es möglich Hardware in the Loop (HiL) Tests mit den entstehenden Softwaresystemen für ein autonomes Schiff durchzuführen. Bei diesen Tests bleibt jedoch zu beachten, dass ein simuliertes Schiff nur annähernd das Verhalten eines realen physikalischen Schiffs abbilden kann. Um die Testbedingungen so realitätsnah wie möglich zu gestalten kann der HIL Testaufbau mit dem physikalischen Testfeld LABSKAUS um ein reales Schiff und eine überwachte Testumwelt erweitert werden. Dieser Testaufbau mit der Verknüpfung zwischen HAGGIS und LABSKAUS über den Simulationsadapter kann als Vessel-in-the-Loop (VIL) bezeichnet werden und ermöglicht das Erproben neuer Systeme und Komponenten in der Praxis in Kombination mit der Reproduzierbarkeit und Sicherheit von Verkehrssimulatoren.

Die Einbettung des Simulationsadapters in die Gesamtarchitektur ist im Anhang unter Abbildung A.2 zu finden.

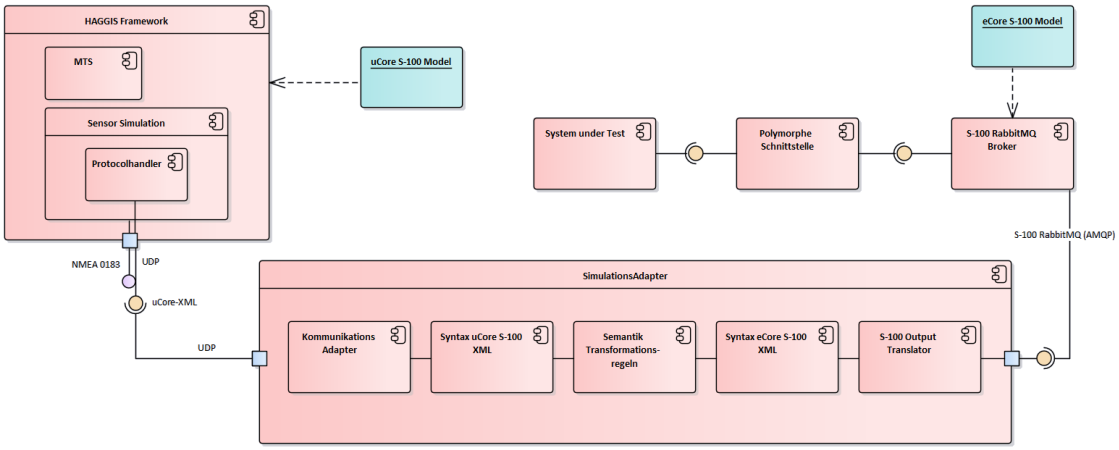


Abbildung 5.8.: Komponentendiagramm zum Entwurf des Simulations-Adapters

Das Konzept für die Umsetzung der Integration simulierter Daten in das physikalsiche Testfeld ist in Abbildung 5.8 ersichtlich. Im Framework HAGGIS muss ein *UCoreXMLProtocolHandler* implementiert werden. Zudem muss für den Simulationsadapter ein *KommunikationsAdapter* umgesetzt werden. Die Syntax der Ucore S-100 XML-Daten muss mit der Syntax der Ecore S-100 XML-Daten verglichen werden, um mögliche Unterschiede in der Transformation zu berücksichtigen. Für die Transformation der XML-Daten wird ein *Semantik Transformations Handler* benötigt. Der Entwurf für die Semantik und Syntax Komponenten entspringt dem SOA-Vorgehen zur Transformation von Datenmodellen [SOA].

Komponenten

In diesem Abschnitt werden die einzelnen Komponenten des Simulationsadapters genauer betrachtet.

ProtocolHandler Für die vorgesehene Simulation wird gemäß Anforderung F-SA-1 aus HAGGIS ein Datenstrom mit Ucore konformen XML-Daten benötigt. In HAGGIS sind der *NMEA0183-ProtocolHandler* und ein *SymbolicRadarProtocolHandler* bereits vorhanden. Diese *ProtocolHandler* erzeugen allerdings nicht die benötigte Ucore konforme XML-Datenausgabe. Dementsprechend muss ein *ProtocolHandler* implementiert werden, der einen OutputStream mit Ucore konformen

XML-Daten erzeugt und über einen der in HAGGIS vorhandenen *TransportHandler* ausgibt. Für die Schnittstelle mit dem Simulationsadapter muss hierfür der *UDPTransportHandler* genutzt werden.

Kommunikations-Adapter Für das Empfangen der von HAGGIS gesendeten Daten muss gemäß Anforderung F-SA-1 eine Schnittstelle zur Kommunikation implementiert werden. Diese Schnittstelle in Form des *Kommunikation-Adapters* muss die Daten über UDP empfangen können und diese dann zur Weiterverarbeitung zwischenspeichern.

Ucore- und Ecore- Syntax Bei der Syntax der Ucore konformen XML Daten und Ecore konformen XML Daten gibt es einige Unterschiede, die dazu führen, dass eine Transformation nicht immer verlustfrei durchgeführt werden kann. Um einen solchen Verlust zu verhindern müssen gemäß Anforderung F-SA-2 die für die Transformation vorgesehenen Datensätze in Ucore betrachtet und die einzelnen Parameter mit den in Ecore vorhandenen Parametern verglichen werden.

XSL - Transformation Für die Transformation der Ucore konformen XML Daten zu Ecore konformen XML Daten werden gemäß der Anforderungen F-SA-3 bis F-SA-6 Transformationsregeln benötigt. Eine Transformation von XML-Dateien kann über die XSLT-Stylesheets erfolgen. Hierfür muss für jede Nachricht ein XSLT-Stylesheet angelegt werden, welches die Transformationsregeln für die übergebenen Daten enthält. Das Einlesen und Speichern der Rückgabe der XSLT-Stylesheets soll so erweiterbar gestaltet werden, dass problemlos weitere XSLT-Stylesheets hinzugefügt werden können. Des Weiteren müssen die transformierten XML Daten gemäß Anforderung NF-SA-1 S-100 und Ecore konform sein.

RabbitMQ-Output Adapter Für den Output der transformierten Daten via RabbitMQ wird gemäß Anforderung F-SA-7 ein Output Adapter benötigt. Der genutzte RabbitMQ-Kommunikations-Adapter wurde bereits von MATE I entwickelt und zu einer ausführbaren Jar-Datei komprimiert. Diese ist in den Repositories unter (*rabbitmq_adapters-1.5.0.jar*) zu finden. Die genaue Umsetzung des Adapters ist unter [MAT17, S.137-140] zu finden.

6. Realisierung des MATE II-Produkts

Im Folgenden wird die Realisierung des MATE II-Produkts auf Basis des vorgestellten Konzepts (vgl. Kapitel 5) beschrieben. Zunächst wird die Architektur des vollständigen Systems vorstellt, anschließend wird die Umsetzung der Komponenten erläutert.

6.1. Systemarchitektur

In diesem Abschnitt werden die Lösungen der beschriebenen Konzepte aus Kapitel 5 vorgestellt. Zu Beginn werden die Realisierungen der Subsysteme präsentiert. Anschließend stehen die Interaktionen untereinander im Fokus und zum Abschluss werden die Schnittstellen konkretisiert. Daraus ergibt sich die Gesamtarchitektur des MATE II-Produkts.

Realisierung der Konzepte

Die in Kapitel 5 vorgestellten Konzepte werden durch abgeschlossene und durch Schnittstellen getrennte Komponenten realisiert. Die Interaktionen zwischen den Komponenten, sind in Abbildung 6.1 zu sehen.

An den Schnittstellen werden die Interaktionen zwischen den Komponenten dargestellt. Die Verarbeitung der Daten wird in den anschließenden Kapiteln zur Realisierung genauer beschrieben. Die Zuordnung der Komponenten zu den Konzepten aus Kapitel 5 ist wie folgt dargestellt:

Der **Küstenleitstand** wird durch die **EPD** realisiert.

Die **Hindernisidentifikation** wird durch die Dynamische Objekterkennung und den **NoGo-Solver** realisiert.

Die **Regelungstechnik** wird durch den **Controller** realisiert.

Die **Komponente zur Integration simulierter Daten** wird durch den **Simulationsadapter** realisiert.

Die **Kollisionsverhütung** wird durch die **Pfad-Planung** realisiert.

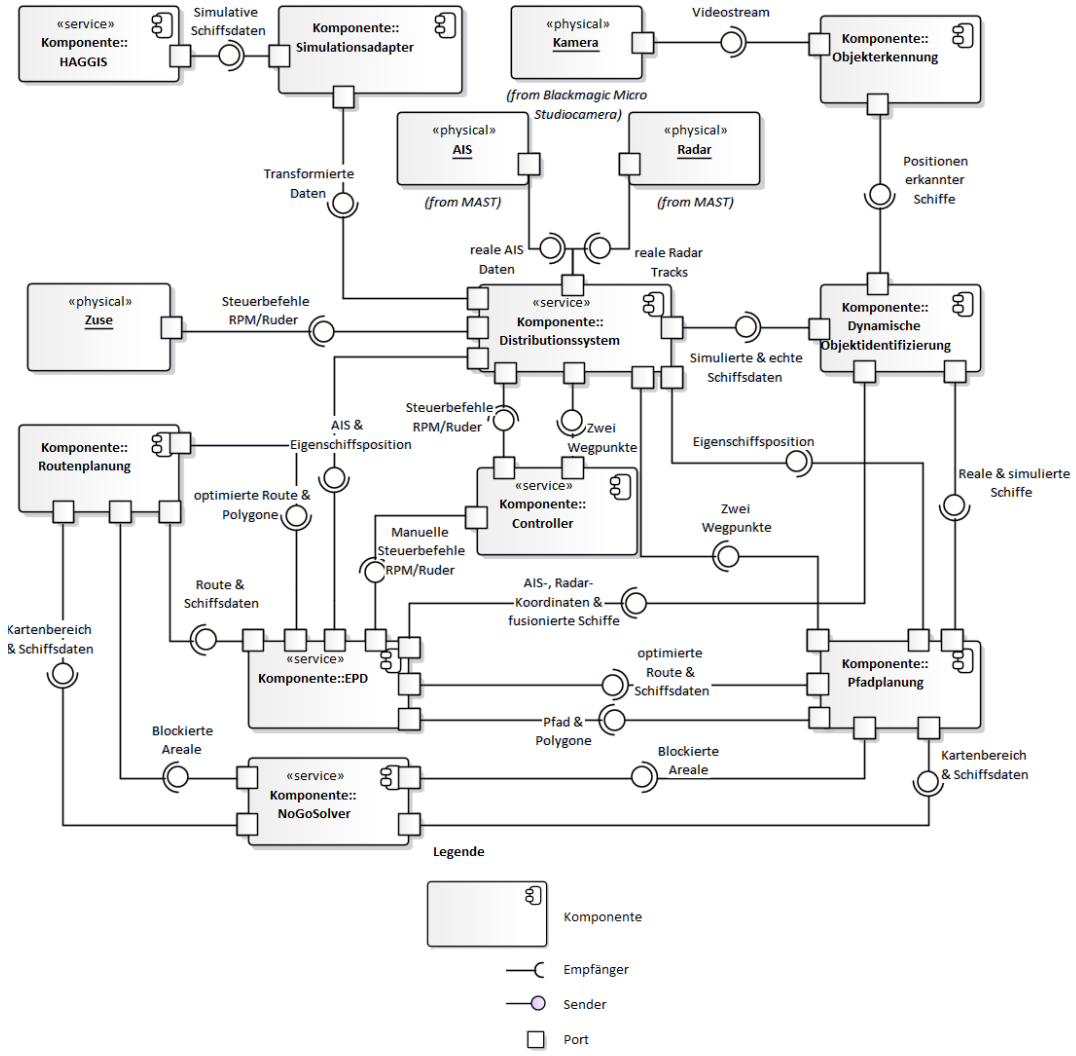


Abbildung 6.1.: Komponentendiagramm des MATE II-Produkts

Die in Abbildung 6.1 dargestellten *«Services»* sind genutzte und zum Teil erweiterte Komponenten, die nicht direkt für die Entwicklung des MATE II-Produkts erstellt wurden. Für die forschungsgetriebene Entwicklung des MATE II-Produkts, sind diese jedoch unerlässlich. Die mit *«Physical»* markierten Komponenten sind genutzte physikalische Objekte, die Daten für das MATE II-Produkt bereitstellen (zum Beispiel Kamera) oder diese verwenden (zum Beispiel Zuse).

Die fehlerfreie Umsetzung der Schnittstellen zur Kommunikation zwischen den Komponenten ist eine zentrale Aufgabe für die Realisierung des MATE II-Produkts. Dafür werden die Kommunikationsprotokolle HTTP, UDP, RabbitMQ, RTP und diverse Websockets verwendet.

In Abbildung 6.2 ist die Kommunikation der einzelnen Komponenten dargestellt. Darunter die verwendeten Standards und die Kommunikationsprotokolle. Die Positionen der Verbindungen sind dabei analog zu Abbildung 6.1 zu betrachten.
Eine detaillierte Betrachtung der Kommunikation inklusive der Ports und Beschreibungen der Komponenten befindet sich im Interface Control Dokument (ICD) im Anhang D.

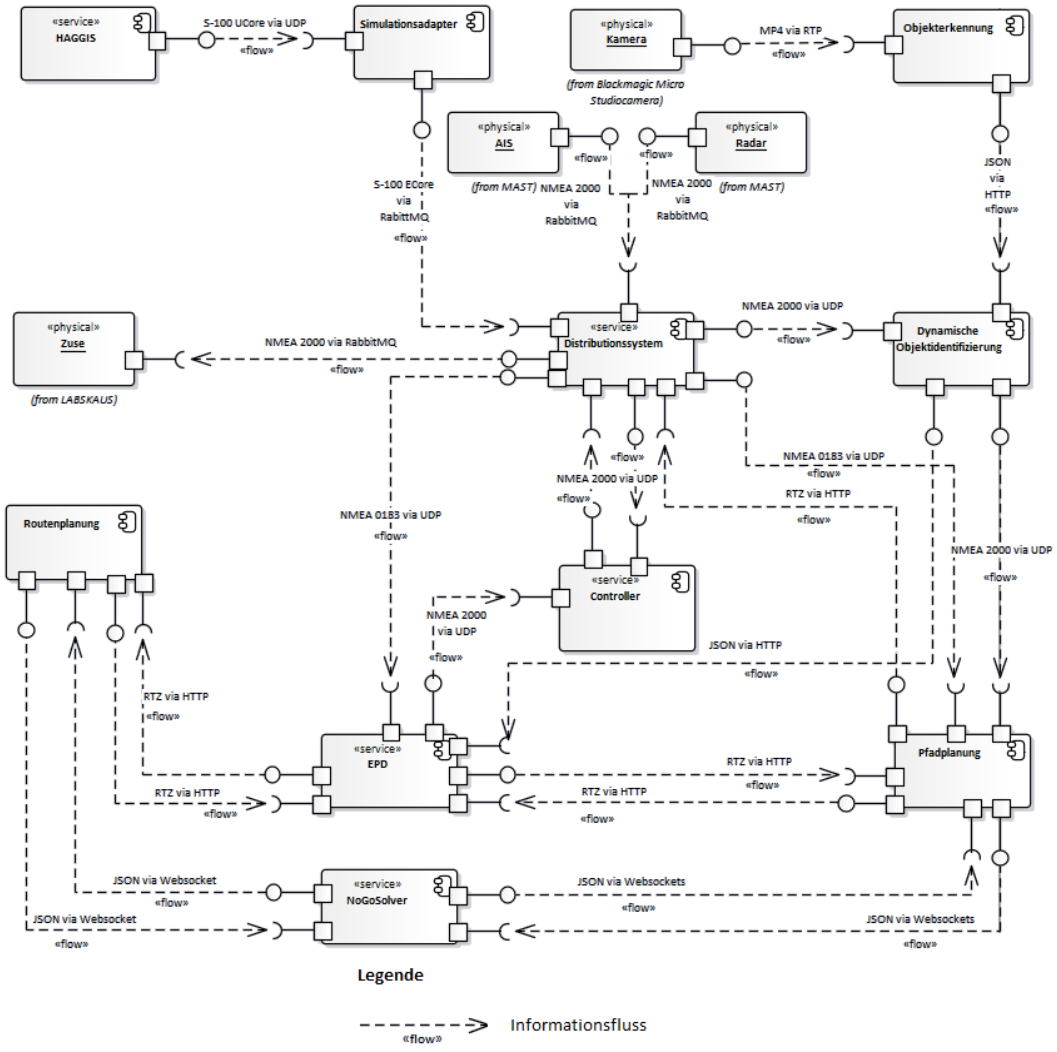


Abbildung 6.2.: Informationsflussdiagramm des MATE II-Produkts

6.2. Küstenleitstand

In diesem Abschnitt wird die Umsetzung des Küstenleitstandes beschrieben. Der Küstenleitstand umfasst dabei die berücksichtigten Funktionen, eine Route für ein spezifisches Schiff erstellen, diese an das autonome Schiff senden und die Fahrt starten zu können. Zudem werden Möglichkeiten der Fahrtüberwachung gegeben, um nachvollziehen zu können, auf welchen Entscheidungen die einzelnen Systeme agieren und das Schiff gesteuert wird.

Softwareumsetzung des Küstenleitstandes

Nach erfolgreicher Projektübernahme der Projektgruppe MATE I wurde entschieden, ebenfalls auf das eMIR EPD (eNavigation Prototype Displays; im folgenden EPD genannt) zu setzen. Im Gegensatz zur PG MATE I wird hierbei jedoch auf eine vom OFFIS vollständig neu entwickelte und modernisierte Form der EPD gesetzt. Gründe hierbei sind vor allem der nicht mehr weiterentwickelte Versionsstand aus dem Jahre 2015. Dies ermöglicht durch die modulare Programmierung eine einfache Erweiterbarkeit und Anpassung auf die Anforderungen von MATE II.

Damit die Projektergebnisse von MATE II auch in weiteren Entwicklungen der EPD berücksichtigt werden und in ein potentielles Endprodukt der EPD ausgeliefert werden können, wird, anders als bei der PG MATE I, auf die neu entwickelte Version aus dem Stand 2018 gesetzt. Da diese in ihrer Grundstruktur vollständig anders arbeitet, werden Kernfunktionen von MATE I (wie z. B. die „Virtual Handles“) in die neue Software portiert, während sämtliche neuen Funktionen, ausgehend von den MATE II-Anforderungen, neu implementiert werden. Durch die modulare, pluginbasierte Implementierung der EPD wird es möglich, Funktionalitäten ab- oder unabhängig voneinander zu implementieren und diese im laufenden Betrieb zuzuschalten oder zu deaktivieren.

Zunächst werden die Umsetzungen der Anforderungen beschrieben, die bereits durch implementierte und MATE II zur Verfügung stehenden Plugins umgesetzt werden können.

Sensordaten Anforderung F-EPD-1.1, also die Darstellung von AIS-Daten, wird durch den Betrieb des Plugins „NMEASensors“ erfüllt, durch dieses, wahlweise über das Protokoll TCP oder UDP, NMEA0183-Daten empfangen und verarbeitet werden können. Für die Anforderung F-EPD-1.1, also die Darstellung von AIS-Daten in Echtzeit, ist es lediglich notwendig, einen entsprechenden NMEA-Sensor auf eine Quelle zu konfigurieren. Im Falle von MATE II ist hier ein UDP-Socket ausreichend, der, nach erfolgreichem Empfang der NMEA0183-Daten, diese auf dem

AIS-Layer des MapView-Plugins darstellt. Die Darstellung erfolgt dabei ECDIS-konform (die Schiffe werden mit Dreiecken dargestellt; veraltete Daten werden sichtbar durchgestrichen). Zudem wird jeweils der gefahrene Kurs des Schiffes auf der Karte dargestellt.

Anforderung F-EPD-1.3, also die Darstellung der eigenen Schiffsposition wird auf Basis der empfangenen AIS-Daten ebenfalls über ein bereits existierendes Plugin („ownShip-Viewer“) ermöglicht, welches den AIS-Datenstrom auf eine konfigurierbare MMSI filtert und das Schiffsobjekt dieser MMSI gesondert darstellt. Die Darstellung erfolgt dabei ebenfalls ECDIS-konform, hier mittels zwei Kreisen und Positionslinien, welche sowohl die Bewegungsrichtung, den COG und den gefahrenen Track nachvollziehbar auf einem eigenen Layer („ownShip“) des MapView-Plugins darstellt.

Anforderung F-EPD-1.2 wird ebenfalls durch ein bereits bestehendes Plugin erfüllt (Plugin „Radar Raw Viewer“). Dieses ruft ein einstellbares Portal, welches Radar-Tracks bereitstellt, ab und stellt diese ebenfalls in dem MapView-Plugin in einem eigenen Layer („Radar“) dar.

Routendarstellung Anforderung F-EPD-2, also das Setzen von Routen, bestehend aus mindestens zwei Wegpunkten, wird über das existierende Plugin „RouteManager“ realisiert, welches es ermöglicht, Wegpunkte in einem eigenen Layer („RouteLayer“) zu setzen und als zusammenhängende Route zu speichern. Die Wegpunkte als auch Verbindungslinien zwischen den einzelnen Wegpunkten werden dabei ECDIS-konform als Kreise, bzw. mit Richtungspfeilen versehene Linien dargestellt. Zudem stellt der RouteManager Schnittstellen bereit, um gespeicherte Routen aus anderen Plugins auszulesen, zu manipulieren oder neue Routen anzulegen.

Kamerabild Anforderung F-EPD-6, also das Darstellen eines Kamerastreams innerhalb der EPD, wird über das existierende View-Plugin von ROSI realisiert. Dabei kann ein Stream (bzw. die URL) in einem beliebigen, standardisierten Videoformat über eine Konfigurationsmaske hinterlegt und einem lokal installiertem Mediaplayer übergeben werden. Der Player stellt das Videobild bereit, welches anschließend in einem eigenen Plugin-Fenster innerhalb derEPDdargestellt wird.

Routenmonitoring

Im folgenden werden Eigenimplementierungen auf Basis der Anforderungen und des Konzepts zur Realisierung des Küstenleitstandes beschrieben.
Als Dienst zum Empfang von Informationen von anderen Komponenten, wie die Routen-Planung,

wird ein Jetty-Server implementiert. Damit eingehende Informationen fachlich in der Anwendung voneinander getrennt und somit auch separat in der EPD-Karte dargestellt werden können, wird für jeden fachlichen Dateneingang ein eigener Jetty-Kontext implementiert.

Empfang und Darstellung optimierter Routen Der Empfang einer optimierten Route von der Routen-Planung (Anforderung F-EPD-5) wird über den Kontext „/route“ gesteuert. Der darin enthaltene Handler empfängt den übertragenen String, bestehend aus einzelnen RTZ-Wegpunkten, und parst diesen zu einer RTZ-Route. Die einzelnen Wegpunkte werden über „Legs“ miteinander verbunden. Um später automatisch Geschwindigkeiten abhängig von den jeweils berechneten bzw. gesetzten Geschwindigkeiten fahren zu können, wird jedem Leg die von der Routen-Planung jeweils zulässige Maximalgeschwindigkeit übergeben. Um optimierte von nicht-optimierten Routen in der EPD unterscheiden zu können, werden optimierten Routen der String „_optimized_“ hinzugefügt. Als Identifikator wird zudem die ID der optimierten Route (durch die Routen-Planung vergeben und übertragen) hinzugefügt. Anschließend wird die vollständig erstellte Route dem Default-RouteManager des RouteManager-EPD-Plugins übergeben, wodurch im RouteManager eine neue Route sichtbar wird (vgl. Abbildung 6.3).

Schnittstelle und Protokoll Routenplanung Die Routen-Planung erzeugt bei der Berechnung von hindernisfreien Routen einige Rohdaten, die zur nachvollziehbareren Überwachung ebenfalls im Küstenleitstand angezeigt werden sollen. Dazu gehören die bei der Berechnung berücksichtigten statischen Hindernisse (beispielsweise Gefahrenstellen, Schifffahrtszeichen, Brücken oder das Festland).
Die Übertragung der Hindernisse erfolgt in jedem Fall als JSON-String. Dieser enthält unter anderem einen `ContentType`, damit der Inhalt und die Art der JSON-Objekte unterschieden werden kann. Es gibt dabei folgende Typen:

GridResult	enthält Grid-Informationen (blockierte und nicht blockierte Zellen)
NoGoAreaResult	enthält Polygone der statischen Hinderniserkennung (NoGo-Solver)
NoRouteFindableException	enthält Fehlermeldung, die ein Popup in der EPD auslöst.

Im JSON-Objekt `Content` befindet sich der anwendungsspezifische Inhalt, welcher anschließend weiterverarbeitet wird. Im Folgenden wird auf die jeweiligen Inhalte und deren Verarbeitung eingegangen.

Empfang und Darstellung statischer Hindernisse Um Hindernisse empfangen zu können, wird ein eigener Kontext „/routePlanningJsonInput“ implementiert, welcher mittels eines eigenen Handlers JSON-Objekte aus dem übertragenen String erstellt. Dazu wird die Google-Gson-Bibliothek verwendet. Um die empfangenen Objekte an einer zentralen Stelle verwalten (einfügen, löschen und auslesen) zu können, wird eine eigene Manager-Klasse als Singleton implementiert. Dieser besitzt für die von der Routen-Planung übertragenen Daten ArrayLists, in denen die einzelnen Polygone gespeichert werden. Um die Hindernisse auf der Karte der EPD darstellen zu können, wird ein eigener MapLayer implementiert („SOIDataLayer“). Dieser wird dabei vom „AbstractMapLayer“ abgeleitet, wodurch eine eigene Zeichnungsmethode implementiert werden kann.
Zum Verwalten der Polygone wird eine eigene Service-Klasse implementiert, die aus den einzelnen Polygonen neue Polygon-Objekte erstellt (farblich ausgefüllt) und diese anschließend in den „RoutePlanningDataManager“ gespeichert. Von dort aus werden sie aus der Zeichnungsmethode ausgelesen. Die sogenannte „Paint“-Methode wird immer dann aufgerufen, wenn sich der Zoomlevel der EPD-Karte ändert oder andere Ereignisse eintreten, die ein Zeichnen der Karte auslösen. Ein solches Ereignis soll dabei auch das Eintreffen neuer Polygone von der Routen-Planung sein, damit diese unmittelbar in der EPD-Karte angezeigt werden, und nicht erst, wenn manuell der Zoomlevel der Karte geändert wird. Dazu wird ein PublishSubject erzeugt, welches bei neu eintreffenden Hindernissen aktualisiert wird (über die jeweilige onNext()-Methode) und ein Zeichnen der EPD-Karte ausgelöst. Dies wird ermöglicht, indem der zur Darstellung der Daten der Routen-Planung zuständige Layer „SOIDataLayer“ auf das PublishSubject subscribed.

Neben den statischen Hindernissen sollen auch entsprechend Anforderung F-EPD-1.3 Daten, die bei der Routenberechnung entstehen, angezeigt werden. Dazu gehören das zur Berechnung der Route verwendete „Grid“, welches blockierte und nicht blockierte Zellen beinhaltet. Diese werden beim Eintreffen am RoutePlanningJsonInput-Handler in Grid-Objekte geparst, welche wiederum in den „RoutePlanningDataManager“ gespeichert werden. Analog zu den statischen Hindernissen wird dabei ein erneutes Zeichnen (über das Aktualisieren eines PublishSubjects) der EPD-Karte ausgelöst, welches mittels des „PolygonService“ aus den gespeicherten Objekten Polygone erstellt, diese je nach Status (blockiert und nicht-blockiert) einfärbt und anschließend auf der Karte zeichnet (vgl. Abbildung 6.3).

Sämtliche von der Routen-Planung empfangenen Polygon-Informationen werden an die Route, zu der sie berechnet wurden, gebunden und gemeinsam in einer HashMap gespeichert. Dies ermöglicht es, dass ein Ändern der Routensichtbarkeit automatisch auch die Anzeige der zu dem Zeitpunkt gespeicherten Polygone auslöst.

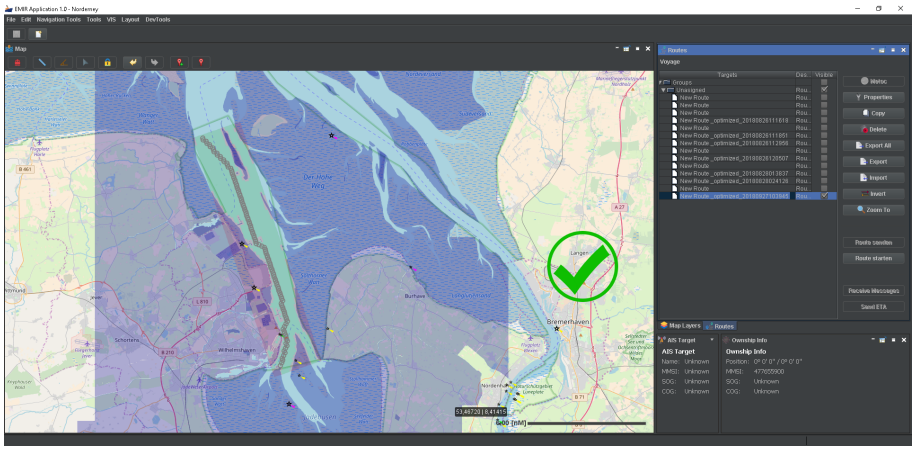


Abbildung 6.3.: Darstellung der optimierten Route, sowie Polygone der statischen Hindernisse

Schnittstelle und Protokoll Pfad-Planung Die Pfad-Planung erzeugt bei der Berechnung von hindernisfreien Teilrouten Rohdaten, die zur nachvollziehbareren Überwachung der Fahrt ebenfalls im Küstenleitstand angezeigt werden sollen. Dazu gehören die bei der Berechnung möglicher Kollisionen mit anderen Schiffen erstellten „Goodwin-Shapes“ sowie die statischen Hindernisse, die vom NoGoSolver zurückgeliefert werden. Die Übertragung der Daten erfolgt als JSON-String. Dieser enthält unter anderem einen `ContentType`, damit der Inhalt und die Art der JSON-Objekte unterschieden werden kann. Es gibt dabei folgende Typen:

GoodwinShapes	JSON-String enthält Polygone, in die das eigene Schiff nicht hineinfahren soll
NoGoAreas	JSON-String enthält Polygone der statischen Hinderniserkennung (NoGoSolver)
GlobalRouteReceived	JSON-String enthält Handshake-Meldung, dass globaler Pfad von der EPD bei der Pfad-Planung angekommen ist.

Im JSON-Objekt `Content` befindet sich der anwendungsspezifische Inhalt, welcher anschließend weiterverarbeitet wird. Im Folgenden wird auf die jeweiligen Inhalte und deren Verarbeitung eingegangen.

Empfang und Darstellung von Rohdaten der Pfadplanung Um Daten von der Pfad-Planung empfangen zu können, wird analog zum Empfang der statischen Hindernisse ein eigener Kontext „/pathPlanningJsonInput“ implementiert, welcher ebenfalls ein JSON-Objekt aus dem übertragenen String erstellt. Dabei werden mittels der Google-Gson-Bibliothek JSON-Objekte erzeugt.

Daraus werden mittels des Polygon-Service Polygone erzeugt, die entweder in die ArrayList der gespeicherten NoGoAreas oder in die der Goodwin-Shapes in den „PathPlanningDataManager“ geschrieben werden.
Um die Daten der Pfad-Planung auf der Karte der EPD darstellen zu können, wird ein eigener MapLayer implementiert („PathPlanningDataLayer“). Dieser wird dabei ebenfalls vom „Abstract-MapLayer“ abgeleitet, wodurch eine eigene Zeichnungsmethode implementiert wird. Bei neu eintreffenden Daten wird über entsprechende PublishSubjects ein automatisches Neuzeichnen des Layers durchgeführt, wodurch mit einer geringen Verzögerung in Echtzeit die Polygone der Pfadplanung in der EPD angezeigt werden (vgl. Abbildung 6.4).

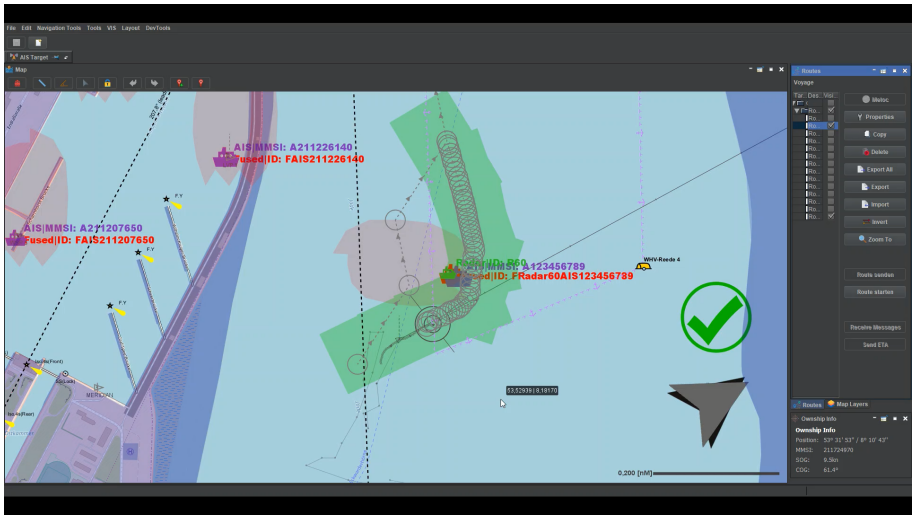


Abbildung 6.4.: Darstellung des abzufahrenden Pfads, GoodWinShapes, sowie Polygone der statischen Hindernisse und Mate Control

Empfang und Darstellung optimierter Pfade Der Empfang der während der Fahrt optimierten Wege bzw. Pfade (Anforderung F-EPD-5) wird über den Kontext „/path“ gesteuert. Der darin enthaltene Handler empfängt den übertragenen String, bestehend aus einzelnen RTZ-Wegpunkten, und parst diesen, analog wie beim Kontext „/route“, zu einer RTZ-Route. Die einzelnen Wegpunkte werden über „Legs“ miteinander verbunden. Um später automatisch Geschwindigkeiten abhängig von den jeweils berechneten bzw. gesetzten Geschwindigkeiten fahren zu können, wird jedem Leg die von der Pfad-Planung jeweils zulässige Maximalgeschwindigkeit übergeben.

Um von der Pfad-Planung berechnete Pfade von nicht-optimierten und optimierten Routen in der EPD unterscheiden zu können, werden optimierten Pfaden der Präfix „Path_“ hinzugefügt. Als Identifikator wird zudem die ID der optimierten Route (durch die Pfad-Planung vergeben und übertragen) hinzugefügt. Anschließend wird die vollständig erstellte Route dem Default-RouteManager des RouteManager-EPD-Plugins übergeben, wodurch im RouteManager eine neue Route sichtbar wird. Aufgrund der Tatsache, dass kontinuierlich ein neuer Pfad empfangen und angezeigt werden muss, wird im RouteManager zuvor geprüft, ob es bereits einen solchen Pfad gibt. Dies erfolgt über die Methode „getVisibleRoutes()“, die eine Liste aller im RouteManager gespeicherten Routen zurückliefert. Im positiven Fall wird die gespeicherte Route überschrieben, sodass ohne Nutzerinteraktion ein Aktualisieren der Route in der EPD stattfindet.

Schnittstelle und Protokoll Dynamische Objektidentifizierung Die Dynamische Objektidentifizierung erzeugt der Erkennung von dynamischen Hindernissen Daten, die zur nachvollziehbareren Überwachung der Fahrt ebenfalls im Küstenleitstand angezeigt werden sollen. Dazu gehören zum einen die Positionen mittels AIS erkannter Schiffe, aber auch die Positionen mittels Radar erkannter Objekte. Das Endergebnis der Dynamischen Objektidentifizierung (fusionierte Objekte und deren Positionen) werden ebenfalls übertragen. Differenziert werden die Objekte der Dynamischen Objektidentifizierung analog zu den Nachrichten der Pfad-Planung und der Routen-Planung über den `ContentType`. Es gibt dabei folgende Typen:

fused JSON-String enthält Koordinaten, die fusionierte Schiffe darstellen

ais JSON-String enthält Koordinaten, die aus den AIS-Nachrichten der Schiffe stammen

radar JSON-String enthält Koordinaten der mittels Radar erkannten Objekte

Im JSON-Objekt `Content` befindet sich der anwendungsspezifische Inhalt, welcher anschließend weiterverarbeitet wird. Im Folgenden wird auf die jeweiligen Inhalte und deren Verarbeitung eingegangen.

Empfang und Darstellung von Rohdaten der Dynamischen Objektidentifizierung Um Daten von der Dynamischen Objekterkennung empfangen zu können, wird analog zum Empfang der statischen Hindernisse ein eigener Kontext „/doiDataInput“ implementiert, welcher ebenfalls ein JSON-Objekt aus dem übertragenen String erstellt. Die darin enthaltenen Daten werden in eine als Singleton implementierte Manager-Klasse („DOIDataManager“) abhängig vom Typ der Daten in eigene ArrayLists gespeichert (jeweils eine für Koordinaten aus AIS, Radar und fusionierten Daten). Um die Daten der dynamischen Objekterkennung auf der Karte der EPD

darstellen zu können, wird ein eigener MapLayer implementiert („DOIDataLayer“). Dieser wird dabei ebenfalls vom „AbstractMapLayer“ abgeleitet, wodurch eine eigene Zeichnungsmethode implementiert werden kann. Bei neu eintreffenden Daten (sowohl bei AIS, Radar als auch Daten der Fusion) wird über entsprechende PublishSubjects ein automatisches Neuzeichnen des Layers durchgeführt (über die Methode „setDirty()“), wodurch mit einer geringen Verzögerung in Echtzeit die Daten der Dynamischen Objektidentifizierung in der EPD angezeigt werden.

Abbildung 6.5.: Darstellung der AIS- und Radardaten sowie Daten fusionierter Schiffe

Overlay-Funktionalitäten Um während der Fahrt überprüfen zu können, ob sich das eigene Schiff auf dem berechneten Pfad befindet und die Richtung zum jeweiligen Zielwegpunkt korrekt ist, existiert ein eigener Layer („StatusOverlayLayer“), welcher in der unteren rechten Ecke der EPD-Karte einen Pfeil für den eigenen Kurs des Schiffes (COG, grau), zudem einen weiteren Pfeil, der die Richtung zum Zielzwischenwegpunkt (schwarz) angibt, darstellt (vgl. Abbildung 6.6). Die Richtung des eigenen Schiffs wird dabei aus dem „ownShip“ des EPD-Models ausgelesen.

Abbildung 6.6.

Wird eine Route an die Pfad- oder Routen-Planung gesendet, wird ein Ladesymbol angezeigt. Trifft eine Antwort (in Form eines gültigen Pfades oder einer optimierten Route) ein, ändert sich das Symbol zu einem grünen Haken (im Erfolgsfall) oder einem roten Kreuz (im Fehlerfall). Gesteuert wird dieser Status über eine Statusvariable, die beim Senden als auch Eintreffen von Routen- oder Pfadinformationen gesetzt wird.

Empfang und Darstellung von Mate Control-Polygonen Um während der Fahrt überprüfen zu können, ob sich das eigene Schiff noch innerhalb des vor der Fahrt festgelegten Soll-Pfades befindet, wird der Kontext „Mate Control“ implementiert. Dieser empfängt, ähnlich wie die bisherigen Handler, ebenfalls JSON-Daten, mit dem Unterschied, dass beim Webservice-Call im JSON-String auch direkt eine Wunschfarbe übergeben werden kann. Dies führt dazu, dass bei der Verarbeitung und Speicherung der Polygonen durch den „PolygonService“ erneut das PublishSubject des „MateControlDataManager“ aktualisiert wird, wodurch die Karte der EPD automatisch neu gezeichnet wird und die übertragenen Polygone auf der Seekarte angezeigt werden (vgl. Abbildung 6.4).

Konfiguration von Kommunikationsparametern Um Konfigurationen zur Kommunikation mit anderen Komponenten über die GUI ändern zu können, existiert eine Settings-Page, welche aus abstrakten Klassen (hier „AbstractSettingsPage“) abgeleitet wird. Die daraus resultierende und zu überschreibende „fillPage()“-Methode stellt zwei Eingabefelder zur Verfügung, dessen Eingabewerte in den Konfigurationskontexts des PGMatePlugins geschrieben wird und innerhalb der EPD ausgelesen werden können. So können IP-Adressen und Ports für die Routen-Planung, die Pfad-Planung und die Regelungstechnik gesetzt werden (vgl. Abbildung 6.7).

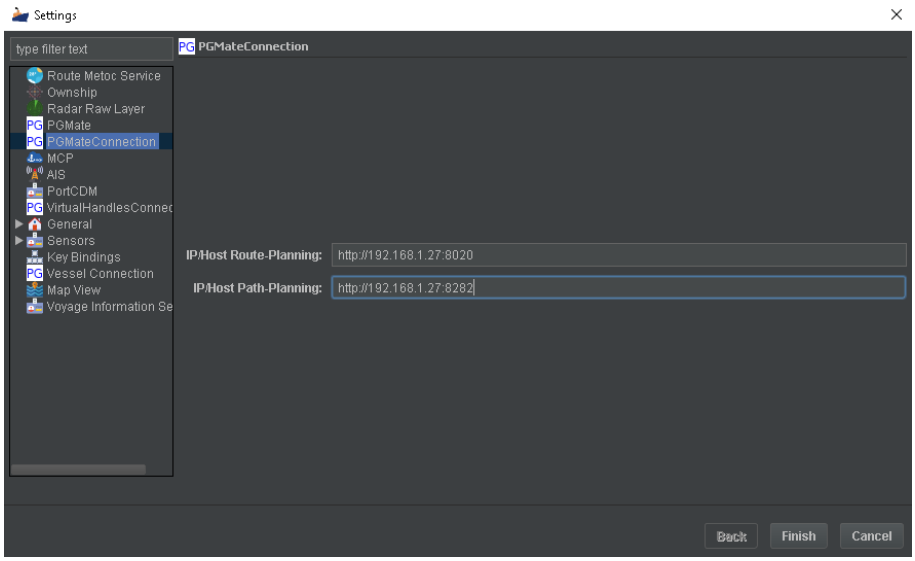


Abbildung 6.7.: Konfigurationsmaske zur Eingabe von URLs für die Pfad- und Routenplanungskomponente

Konfiguration von Schiffsinformationen Zur statischen Hinderniserkennung und damit der Berechnung von Routen und Pfaden sind die jeweiligen Schiffsinformationen des fahrenden Schiffs notwendig. Um diese über die GUI setzen zu können, existiert ebenfalls eine Settings-Page, die ebenfalls von der abstrakten Settings-Page abgeleitet wird. In dieser Maske werden die Höhe, Breite, Länge und der Tiefgang des autonomen Schiffs hinterlegt (vgl. Abbildung 6.8). Gespeichert werden diese Werte fachlich getrennt in einen eigenen Kontext. Hierbei wird Anforderung „F-EPD-3“ realisiert.

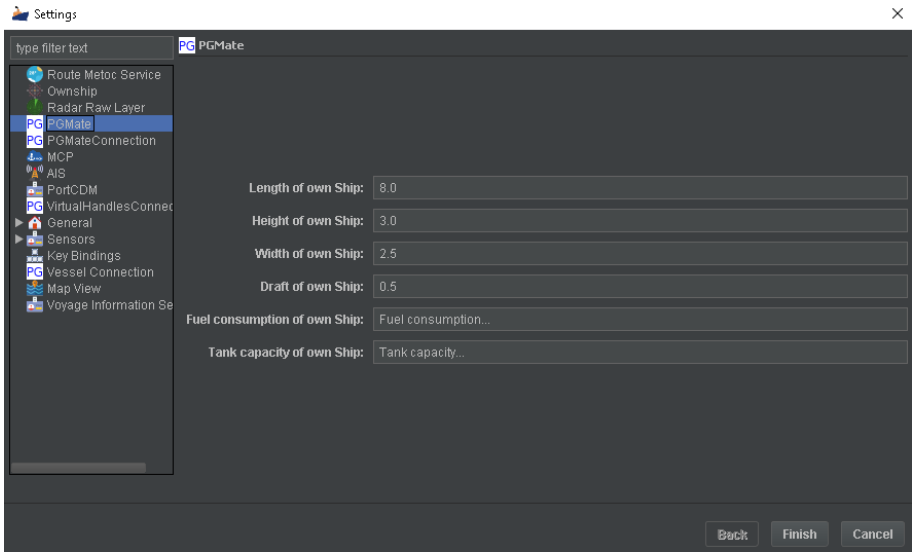


Abbildung 6.8.: Konfigurationsmaske zur Eingabe von Schiffscharakteristik

Senden von Routen an das Route-/Path-Planning Um eine Route durch die Routen-Planung optimieren zu lassen (Anforderung „F-EPD-4“) oder eine autonome Fahrt des Schiffs starten zu können (Anforderung „F-EPD-9“), existieren in der Toolbar des RouteManagers zwei Schaltflächen. Diese werden von der abstrakten Klasse „AbstractCommand“ abgeleitet und führen die notwendigen Befehle aus.

Beim Klick auf „Route optimieren“ wird die aktuell ausgewählte Route und die hinterlegten Schiffsinformationen an die im Kontext konfigurierte IP-Adresse der Routen-Planung gesendet (Route wird an „/route“, bzw. „/path“ gesendet, während Schiffsinformationen an „/ship“ übertragen werden). Die Schaltfläche wird nur aktiv, wenn auch eine nicht-optimierte Route ausgewählt wird. Beim Klick auf „Route starten“ wird die aktuell ausgewählte, optimierte Route und die hinterlegten Schiffsinformationen an die im Kontext konfigurierte IP-Adresse der Pfad-Planung gesendet (Route wird an „/route“, bzw. „/path“ gesendet, während Schiffsinformationen an „/ship“

übertragen werden). Die Schaltfläche wird nur aktiv, wenn auch eine optimierte Route ausgewählt wird.
Zum Versenden der Routen wird ein Jetty-HTTP-Client genutzt. Aufgrund der Tatsache, dass beim Abwarten auf eine Antwort von der Routen- und Pfad-Planung der Hauptthread blockiert wird, werden ausgehende Sende-Befehle in einem eigenen Thread gestartet.

Senden von Steuerbefehle an die Regelungstechnik (Virtual Handles) Mittels des von PG MATE I implementierten und von MATE II portierten Plugins „VirtualHandles“ ist das Senden von Ruder und RPM-Befehlen möglich (Anforderung „F-EPD-8“). Die Zieladresse der Regelungstechnik wird dabei in der beschriebenen Einstellungsmaske hinterlegt. Die manuelle Steuerung über die VirtualHandles wird dabei über eine Schaltfläche im Webinterface gestartet, bzw. gestoppt. Ist die Steuerung aktiv, werden Steuerbefehle als NMEA2000-Nachrichten per UDP an den Controller gesendet. Zudem wird der Routenstatus zurückgesetzt, damit die manuelle Steuerung stets das automatische Abfahren einer optimierten Route unterbrochen werden kann.

Gesamtarchitektur/Schnittstellen Küstenleitstand

In Abbildung 6.9 wird die Gesamtarchitektur der EPD-Erweiterung dargestellt.

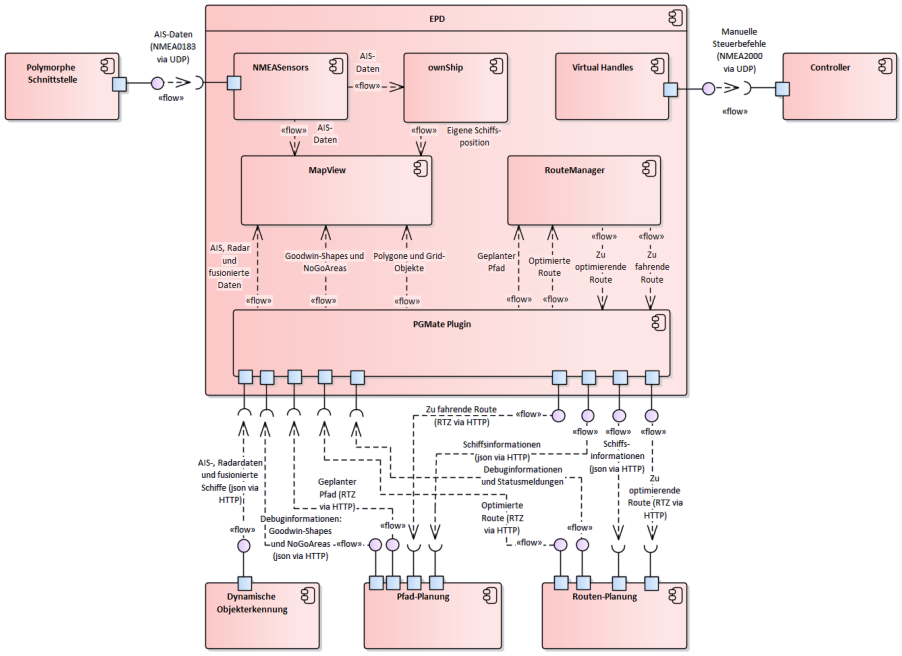


Abbildung 6.9.: Architektur Küstenleitstand/EPD-Plugin

Routen-Planung

Die grobe Architektur der Routen-Planung wurde bereits in Abschnitt 5.2 erläutert. Nachfolgend wird die konkrete Umsetzung gegliedert in Empfang der globalen Route und Schiffsparamter von dem Küstenleitstand, Validierung der Route mit Hilfe des NoGoSolvers und zurücksenden der validen Route an den Küstenleitstand.

Globale Route und Schiffsparamter von dem Küstenleitstand empfangen und speichern

Die globale Route wird im Route Exchange Format (RTZ) von der EPD über HTTP entgegengenommen und decodiert. Dazu wird ein RouteInputHandler implementiert. Der über HTTP empfangende String wird hier mit der DocumentParser-Klasse aus der RTZ-Bibliothek in eine RTZ-Route geparst und im RouteState gespeichert. Der RouteState speichert lediglich die aktuelle Route. Zudem wird ein RouteChangeListener implementiert, welcher die Komponente (genauer die RouteCalculation-Klasse) benachrichtigt, wenn eine neue Route im RouteState gespeichert wurde. Die Schiffsparameter werden als JSON-Objekt von der EPD entgegengenommen und decodiert. Das JSON-Objekt besteht aus einem Schiffsidentifikator, der Länge und Breite des Schiffs, dem Tiefgang und der Höhe des Schiffs. Zur Entgegennahme und Decodierung wird ein ShipInputHandler implementiert. Die empfangende Nachricht wird hier mit der Gson-Klasse aus der Google-Gson-Bibliothek in ein eigens erstelltes Ship-Objekt geparst und im ShipState gespeichert. Der ShipState speichert lediglich das aktuelle Ship-Objekt. Zudem wird analog zum RouteChangeListener ein ShipChangeListener implementiert.

Validierung der Route mit Hilfe des NoGoSolvers

Liegt eine zu validierende Route mit zugehörigen Schiffsparameter vor, wird mit der Validierung, und gegebenenfalls mit der Anpassung, der gegebenen Route begonnen. Hierzu wird zunächst zur Ermittlung der statischen Hindernisse der NoGoSolver angefragt.

Anfrage an den NoGoSolver

Zur Anfrage an den NoGoSolver wird ein NoGoSolverClient implementiert. Dieser verfügt über eine requestPolygons-Methode, welche ein Ship-Objekt und ein Frame-Objekt erwartet. Das eigens erstellte Frame-Objekt besteht aus zwei Koordinatenpunkten welche zur Anfrage an den NoGoSolver benötigt werden. Die requestPolygons-Methode erstellt aus dem übergebenen Parametern ein JSON-Object bestehen aus einem Schiffsidentifikator, der Länge, Breite, Höhe und

Tiefe des Schiffs und den zwei Koordinaten Punkten des Frame-Objects. Das JSON-Objekt wird dann über einen Websocket an den NoGoSolver gesendet. Dieser antwortet mit einem JSON-Objekt in dem sich Points, Linestrings und Polygone befinden. Der NoGoSolverClient empfängt dieses JSON-Objekt, parst es mittels der Gson-Klasse aus der Google-Bibliothek in ein Obstacle-Objekt und speichert dies im ObstacleState.

Aufbau der Routenberechnung

Zum Starten einer Routenberechnung bei empfangenen Routen und Schiffsparametern von der EPD wird die RouteCalculation-Klasse umgesetzt. Sie erbt von allen ChangeListenern, also vom RouteChangeListener, ShipChangeListener und ObstacleChangeListener. Sie besitzt für jeden ChangeListener eine changed-Methode, wodurch sie auf neu empfangende Routen, Schiffsparametern und statischen Hindernissen reagieren kann. Wenn also eine neue Route mit entsprechenden Schiffsparametern empfangen wurde, werden diese in die entsprechenden States geladen wodurch über die ChangeListener die zugehörigen change-Methoden aufgerufen werden. In der change-Methoden des RouteChangeListeners und des ShipChangeListeners wird über eine requestObstacles-Methode die requestPolygon-Methode der NoGoSolverClient-Klasse aufgerufen. Wenn der NoGoSolverClient dann, wie bereits ausführlich beschrieben, den aktuellen ObstacleState überschreibt, wird durch den ObstacleStateListener die zugehörige change-Methode in der RouteCalculation-Klasse aufgerufen. Nun startet diese eine Routenberechnung indem sie zunächst eine Instanz der AStar-Klasse erzeugt und ihr alle benötigten Parameter übergibt. Dazu gehören Start- und Zielpunkt (von der EPD definiert), Grid-Höhe und -Breite, Zellgröße und das Verhältnis von Grid-Höhe zu Grid-Breite. Jeder dieser Paramater mit Ausnahme von Start- und Zielpunkt, welche von der EPD vorgegeben werden, können in der Konfigurationsdatei der Pfad-Planung (`configuration.properties`) festgelegt werden.

Aus dem Konstruktor der AStar-Klasse wird zunächst eine Instanz der Grid-Klasse erzeugt (mit denselben Parameterübergaben) welches das Occupancy-Grid darstellt. Im Konstruktor des Grids wiederum werden Start- und Zielzelle innerhalb des Grids anhand der übergebenen Start- und Zielpunkte berechnet. Dies wurde mittels einer GeoidCalculator-Klasse realisiert.
Die GeoidCalculator-Klasse verwendet die GeoTools-Bibliothek, welche es zunächst ermöglicht Winkel und Entfernungen zwischen zwei Koordinatenpunkten (wie z.B. Start- und Zielpunkt) zu berechnen. Des Weiteren ermöglicht die Bibliothek einen neuen Koordinatenpunkt aus einem vorhandenen Punkt, einem angegebenen Winkel und einer Distanz zum gewünschten Punkt zu berechnen. Die Berechnung der Zielzelle erfolgt jedoch nicht ausschließlich durch die GeoidCalculator-Klasse, sondern auch die calculateCell-Methode. Diese Methode berechnet die Zielzelle und beachtet dabei, dass das rechteckige Grid in Richtung des Zielpunktes gedreht werden soll. Das Grid enthält ebenfalls die Methode getCell welche dafür verantwortlich ist alle Zellen, die in-

nerhalb eines Polygons liegen, oder von einem Polygon oder LineString geschnitten werden, als blockiert zurückzugeben. Wurde das Grid initialisiert wird es von der RouteCalculation-Klasse aus mit statischen Hindernissen, also Obstacle-Objekten, befüllt. Die Obstacle-Objekte werden hier zunächst mittels der WKT-Reader Klasse aus der GeoTools-Bibliothek in die Geomertry-Objekte Polygon, Linestring und Point geparst und in angelegte Array-Listen der Grid-Klasse gespeichert.
Nun wird in der RouteCalculation-Klasse die computeRoute-Methode der AStar-Klasse aufgerufen welche eine neue Instanz der Calculation-Klasse erzeugt. Diese beinhaltet die eigentliche Berechnung des A*-Algorithmus und berechnet einen Pfad, also eine Liste von Zellen, durch das Grid bei dem keine Zelle blockiert ist und der zur Zielzelle führt. Diese Liste wird in der RouteCalculation-Klasse abgespeichert und an die WaypointsOutput-Klasse übergeben.

Valide Route an EPD versenden

Zum Versenden der validen Route wird eine WaypointsOutput-Klasse implementiert. Diese verfügt über eine sendRoute-Methode. Dieser Methode kann eine Liste von Zellen übergeben werden, woraus dann über die RouteBuilder-Klasse aus der RTZ-Bibliothek eine RTZ-Route erstellt werden kann. Die RTZ-Route wird dann über eine HTTPRequestService-Klasse, welche einen einfachen Jetty-Client darstellt, an dieEPDversendet.

6.3. Kollisionsverhütung

Der folgende Abschnitt beschreibt die Realisierung des Konzeptes der Kollisionsverhütung.

Pfad-Planung

Die grobe Architektur der Pfad-Planung wurde bereits in Abschnitt 5.3 erläutert. Nachfolgend wird die konkrete Umsetzung gegliedert in Empfang der globalen Route und Schiffsparamter von der EPD, Empfang der aktuellen Position, des Headings und der statischen und dynamischen Hindernisse, Berechnung eines validen kollisionsfreien Pfads und berechneten Pfad an Controller und EPD senden.

Valide globale Route und Schiffsparameter von EPD empfangen und speichern

Die globale Route wird in RTZ von der EPD über HTTP entgegengenommen und decodiert. Dazu wird ein RouteInputHandler implementiert. Der über HTTP empfangende String wird hier mit der DocumentParser-Klasse aus der RTZ-Bibliothek in eine RTZ-Route geparst und im RouteState gespeichert. Der RouteState speichert lediglich die aktuelle Route. Zudem wird ein RouteChangeListener implementiert, welcher die Komponente (genauer die RouteCalculation-Klasse) benachrichtigt, wenn eine neue Route im RouteState gespeichert wurde. Die Schiffsparamter werden als JSON-Objekt von der EPD entgegengenommen und decodiert.
Das JSON-Objekt besteht aus einem Schiffsidentifikator, der Länge und Breite des Schiffs, dem Tiefgang und der Höhe des Schiffs. Zur Entgegennahme und Decodierung wird ein ShipInputHandler implementiert. Die empfangende Nachricht wird hier mit der Gson-Klasse aus der Google-Bibliothek in ein eigens erstelltes Ship-Objekt geparst und im ShipState gespeichert. Der ShipState speichert lediglich das aktuelle Ship-Objekt. Zudem wird analog zum RouteChangeListener ein ShipChangeListener implementiert. Zur Bestätigung des Empfangs einer neuen validen globalen Route von der EPD wird zudem eine GlobalRouteReceivedOutput-Klasse implementiert, welche über eine HTTPRequestService-Klasse ein JSON-Objekt an die EPD versendet. Das JSON-Objekt beinhaltet dabei nur den String „Route received!“.

Aktuelle Position und aktuelles Heading vom Distributionssystem empfangen und speichern

Die aktuelle Position wird in NMEA0183 von dem Distributionssystem über UDP entgegengenommen und decodiert. Dazu wird eine UDPInputServer-Klasse implementiert. Dieser empfängt den gesendeten String, parst diesen mit Hilfe einer PositionParser-Klasse, welche auf die MarineAPI-Bibliothek zugreift, in einen RTZ-Point und speichert den Point in den aktuellen PositionState. Der PositionState speichert lediglich die aktuelle Route. Zudem wird ein PositionChangeListener implementiert, welcher die Komponente benachrichtigt, wenn eine neue Position im PositionState gespeichert wurde.
Darüber hinaus wird ein PositionTimeOutListener realisiert. Dieser prüft ob mindestens alle sieben Sekunden eine neue Position vom Distributionssystem empfangen wird und setzt dementsprechend eine boolsche-Variable isConnected welche sich über die Methode isConnected des PositionStates abfragen lässt. Das aktuelle Heading wird in NMEA0183 von dem Distributionssystem über UDP entgegengenommen und decodiert. Dazu wird eine UDPInputServerHeading-Klasse implementiert. Dieser empfängt den gesendeten String, parst diesen mit Hilfe einer Hea-

dingParser-Klasse, welche auf die MarineAPI-Bibliothek zugreift, in einen Double-Wert und speichert diesen in den aktuellen HeadingState. Der HeadingState speichert lediglich das aktuelle Heading. Zudem wird ein HeadingChangeListener implementiert, welcher die Komponente benachrichtigt, wenn ein neues Heading im HeadingState gespeichert wurde. Darüber hinaus wird ein HeadingTimeOutListener realisiert. Dieser prüft ob mindestens alle sieben Sekunden ein neues Heading vom Distributionssystem empfangen wird und setzt dementsprechend eine boolsche-Variable isConnected welche sich über die Methode isConnected des HeadingStates abfragen lässt.

Statische Hindernisse vom NoGoSolver anfragen, empfangen und speichern

Zur Anfrage an den NoGoSolver wird ein NoGoSolverClient implementiert. Dieser verfügt über eine requestPolygons-Methode, welche ein Ship-Objekt und ein Frame-Objekt erwartet. Das eigens erstellte Frame-Objekt besteht aus zwei Koordinatenpunkten, welche zur Anfrage an den NoGoSolver benötigt werden. Die requestPolygons-Methode erstellt aus den übergebenen Parametern ein JSON-Object bestehen aus einem Schiffsidentifikator, der Länge, Breite, Höhe und Tiefe des Schiffs und den zwei Koordinaten Punkten des Frame-Objects. Das JSON-Objekt wird dann über einen Websocket an den NoGoSolver gesendet. Dieser antwortet mit einem JSON-Objekt in dem sich Points, Linestrings und Polygone befinden.
Der NoGoSolverClient empfängt dieses JSON-Objekt, parst es mittels der Gson-Klasse aus der Google-Bibliothek in ein StaticObstacle-Objekt und speichert dies im StaticObstacleState. Der StaticObstacleState speichert lediglich die aktuellen StaticObstacle-Objekte, also die aktuellen statischen Hindernisse. Zudem wird ein StaticObstacleChangeListener implementiert, welcher die Komponente benachrichtigt, wenn neuen StaticObstacle-Objekte im StaticObstacleState gespeichert werden. Darüber hinaus wird ein StaticTimeOutListener realisiert. Dieser prüft ob mindestens alle 40 Sekunden neue statische Hindernisse empfangen werden und setzt dementsprechend eine boolsche-Variable isConnected welche sich über die Methode isConnected des StaticObstacleStates abfragen lässt.

Dynamische Hindernisse von der Dynamischen Objektidentifizierung empfangen und speichern

Zum Empfang der dynamischen Hindernisse von der Dynamischen Objektidentifizierung wird eine DynamicObjectInputHandler-Klasse implementiert. Dieser empfängt den gesendeten String und parst diesen mit Hilfe einer DynamicObjectParser-Klasse in ein DynamicObstacle-Objekt.

Ein DynamicObstacle-Objekt besteht aus einem Längen- und Breitengrad, einer Schiffslänge, einer Schiffsbreite, einer Kurs-über-Grund-Angabe und aus einer Geschwindigkeit-über-Grund-Angabe.
Die DynamicObjectInputHandler-Klasse speichert die geparsten Dynamic-Obstacles schließlich in den DynamicObstacleState. Der DynamicObstacleState speichert lediglich die aktuellen DynamicObstacle-Objekte, also die aktuellen dynamischen Hindernisse. Zudem wird ein DynamicObstacleChangeListener implementiert, welcher die Komponente benachrichtigt, wenn neue DynamicObtacle-Objekte im DynamicObstacleState gespeichert werden.
Darüber hinaus wird ein DynamicTimeOutListener realisiert. Dieser prüft ob mindestens alle zehn Sekunden neue dynamische Hindernisse empfangen werden und setzt dementsprechend eine boolsche-Variable isConnected welche sich über die Methode isConnected des DynamicObstacleStates abfragen lässt.

Berechnung eines validen kollisionsfreien Pfads zum nächsten Wegpunkt

Zum Starten einer Pfadberechnung wird die PathIterator-Klasse implementiert. Sie kennt alle State-Klassen, also die PositionState-, HeadingState-, StaticObstacleState-, DynamicObstacleState-, RouteState-, WaypointsState- und PathState-Klasse. Der RouteState speichert lediglich die aktuelle abzufahrende Route von der EPD. Zudem wird ein RouteChangeListener implementiert, welcher die Komponente benachrichtigt, wenn eine neue Route im RouteState gespeichert wurde.

Der WaypointState speichert den aktuellen nächsten anzufahrenden Wegpunkt aus der aktuell abzufahrenden Route des RouteStates als Koordinatenpunkt. Zudem wird ein WaypointChangeListener implementiert, welcher die Komponente benachrichtigt, wenn ein anzufahrender Wegpunkt erreicht wird und ein neuer Wegpunkt im WaypointState gespeichert wurde. Für die Erkennung ob ein anzufahrender Wegpunkt erreicht wurde ist die ActiveWaypointCalculation-Klasse verantwortlich. Diese wurde bereits von MATE I realisiert.

Der PathState speichert den aktuell berechneten validen kollisionsfreien Pfad für den Controller und für die EPD. Diese unterscheiden sich dahingehend, dass der Controller immer nur die nächsten zwei Wegpunkte aus dem aktuellen Pfad erwartet, während die EPD alle Wegpunkte des Pfads entgegennimmt. Zudem wird ein PathChangeListener implementiert, welcher die Komponente benachrichtigt, wenn ein Pfad oder EPD-Pfad gespeichert wurde.

Die PathIterator-Klasse versucht immer wieder einen neuen Pfad zu berechnen. Zunächst überprüft sie, ob eine aktuelle Position und ein aktuelles Heading, ein aktueller abzufahrender Weg-

punkt und statische und dynamische Hindernisse vorhanden sind. Dies wird überprüft in dem die jeweilige isConnected-Methode der zugehörigen State-Klassen aufgerufen werden. Sind alle erforderlichen Verbindungen vorhanden, werden zunächst die StaticObstacle-Objekte aus dem StaticObstacleState in eine vom PathIterator erzeugte ArrayList vom Typ String geladen. Analog dazu werden alle DynamicObstacle-Objekte geladen.

Als nächstes werden die aktuelle Position und der anzufahrende Wegpunkt aus dem PositionState und dem WaypointState geladen. Danach wird eine Instanz der AStar-Klasse erzeugt und ihr alle benötigten Parameter übergibt. Dazu gehören Start- und Zielpunkt, Zellgröße, Grid-Breite, ein Sicherheitsabstandsfaktor um Point-Hindernisse vom NoGoSolver, ein Sicherheitsabstandsfaktor für dynamische Hindernisse, ein maximaler Wendewinkel, aktuelle Position und das aktuelle Heading. Während das Heading im HeadingState gespeichert wird, kann jeder andere Parameter in der Konfigurationsdatei der Pfad-Planung (configuration.properties) festgelegt werden. Aus dem Konstruktor der AStar-Klasse wird zunächst eine Instanz der Grid-Klasse erzeugt (mit denselben Parameterübergaben) welches das Occupancy-Grid darstellt. Im Konstruktor des Grids wiederum wird die Startzelle innerhalb des Grids berechnet. Dies wird mittels einer GeoidCalculator-Klasse realisiert. Die GeoidCalculator-Klasse verwendet die GeoTools-Bibliothek, welche es zunächst ermöglicht Winkel und Entfernungen zwischen zwei Koordinatenpunkten (wie z.B. Start- und Zielpunkt) zu berechnen. Des Weiteren ermöglicht die Bibliothek einen neuen Koordinatenpunkt aus einem vorhandenen Punkt, einem angegebenen Winkel und einer Distanz zum gewünschten Punkt zu berechnen. Wurde das Grid initialisiert wird es von der PathIterator-Klasse aus mit statischen und dynamischen Hindernissen, also Static- und DynamicObstacle-Objekten, befüllt. Die Static- und DynamicObstacle-Objekte werden hier zunächst mittels der WKT-Reader Klasse aus der GeoTools-Bibliothek in die Geomertry-Objekte Polygon, Linestring und Point geparst und in angelegte Array-Listen der Grid-Klasse gespeichert. Die DynamicObstacle-Objekte werden im Grid besonders aufgrund der Fremdschiffdomäne und des CPA-Konzepts besonders behandelt (siehe Abschnitt 3.4). Wenn die DynamicObstacle-Objekte in das Grid geladen werden, und damit die addDynamicObstacles-Methode des Grids aufgerufen wird, wird ebenfalls eine Instanz der CPA-Utils-Klasse erzeugt. Diese verwendet die CPA-Implementierung von MTCAS, berechnet die Position an dem das Fremdschiff den kleinsten Abstand zum eigenen Schiff aufweist und verändert die Positionsangabe des Fremdschiff auf die entsprechende berechnete Position.

Hiernach wird im Grid die drawOuterGoodwinRounded-Methode aufgerufen, welche nach dem Goodwin-Konzept eine Fremdschiffdomäne, also einen Sicherheitsabstand, um die DynamicObstacle-Object zieht. Dies wird realisiert indem ein zusätzliches Polygon für jedes DynamicObstacle-Objekt ins Grid geladen wird. Danach wird in der PathIterator-Klasse die computeRoute-

Methode der AStar-Klasse aufgerufen, welche eine neue Instanz der Calculation-Klasse erzeugt. Dadurch wird zunächst die Zielzelle im Grid berechnet wobei beachtet wird, dass das rechteckige Grid in Richtung des Zielpunktes gedreht werden soll.

Dann wird die computeRoute()-Methode der Calculation-Klasse aufgerufen, welche die eigentliche Berechnung des A*-Algorithmus beinhaltet. Der A*-Algorithmus erhält hierbei jedoch nur diejenigen Zellen für die OpenList, welche von der getSuccessorsTNeighbourhood-Methode zurückgegeben werden (siehe Abschnitt 3.4). So wird ein Pfad berechnet, also eine Liste von Zellen, durch das Grid bei dem keine Zelle blockiert ist und der zur Zielzelle führt. Diese Liste wird in der PathIterator-Klasse abgespeichert und an die WaypointsOutput-Klasse übergeben. Schließlich wird eine neue Iteration des beschriebenen Prozesses gestartet.

Validen und kollisionsfreien Pfad an Controller und EPD senden

Zum Versenden des validen und kollisionsfreien Pfads wird eine WaypointsOutput-Klasse implementiert. Diese erbt von einer TimerTask-Klasse aus der Java.Utils-Bibliothek und verfügt somit über eine run-Methode. Diese versucht in einem einstellbaren Intervall den aktuellen Pfad aus dem PathState über eine HTTPRequestService-Klasse an den Controller und die EPD zu versenden.

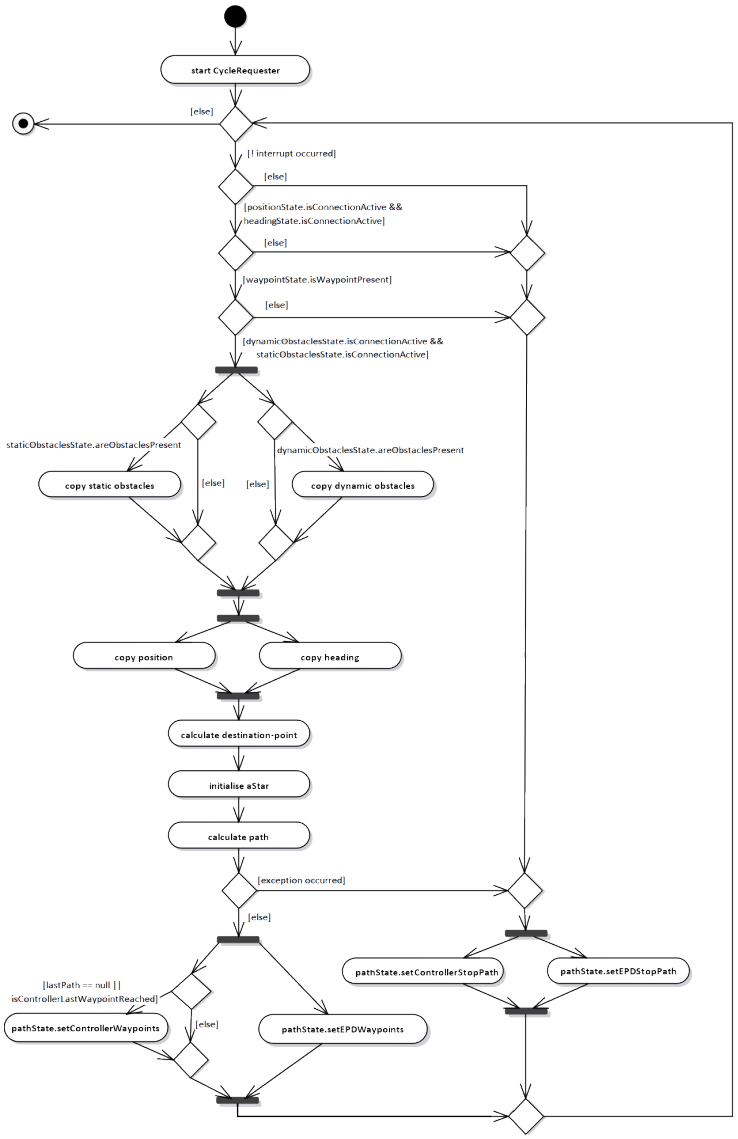


Abbildung 6.10.: Ablaufdiagramm PathIterator Realisierung

6.4. Hindernisidentifikation

Um die Hindernisidentifikation wie in Abschnitt 5.4 beschrieben umzusetzen, werden zwei weitere Komponenten benötigt: DynamicObjectIdentification (im weiteren Verlauf Dynamische Objektidentifizierung genannt) für die Verarbeitung und Fusionierung von Radar und AIS sowie ObjectDetection (im weiteren Verlauf Objektdetektion genannt) für die Bildverarbeitung der aufgenommenen Kamerabilder. Ein Komponentendiagramm der beiden Komponenten zeigt die

Abbildung 6.11. Die konkrete Realisierung dieser Komponenten wird im folgenden Kapitel beschrieben.

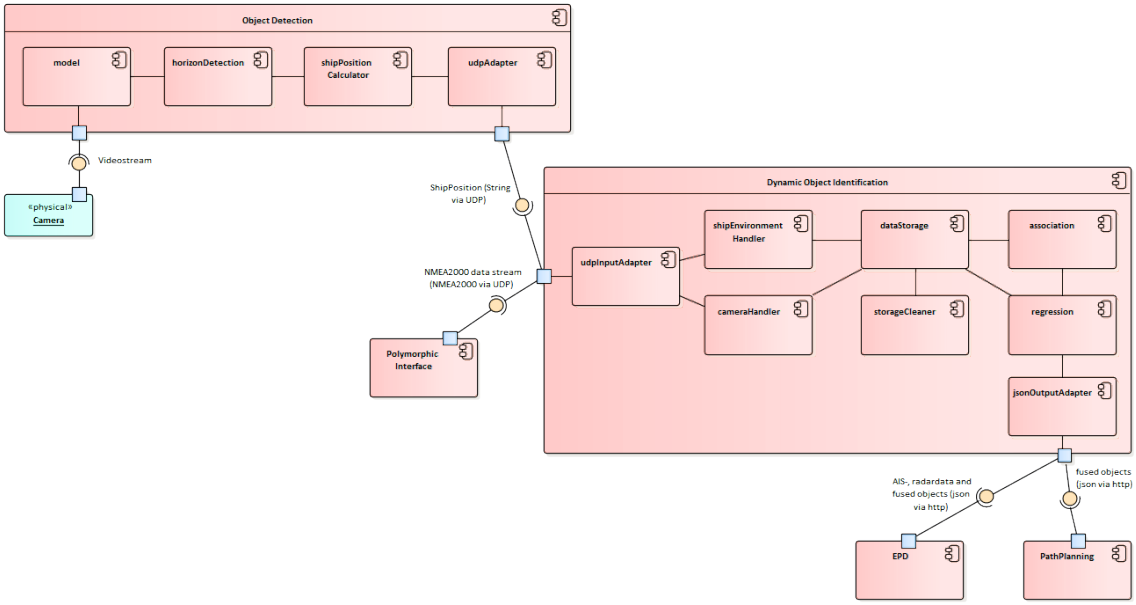


Abbildung 6.11.: Komponentendiagramm der Hindernisidentifikation

6.4.1. Dynamische Objektidentifizierung

Die Dynamische Objektidentifizierungs-Komponente muss relevante Nachrichten des Hub's empfangen, auslesen und speichern können. Im Anschluss muss sie die AIS- und Radar-Nachrichten fusionieren und den Komponenten Pfad-Planung sowie EPD alle von ihnen benötigten Informationen zu den Schiffen zusenden. Im Folgenden wird die Umsetzung der Komponentenbestandteile beschrieben.

Datenmanagement

Um eine Datenfusion auf den Radar- sowie AIS-Daten ausführen zu können, müssen zunächst diese Informationen zur Komponente gelangen, dort ausgelesen und gespeichert werden.

Transformation der S100 Radarnachrichten in eine NMEA2000 konforme Nachricht
Da die Projektgruppe MATE I ohne Radar gearbeitet hat, existierte in dem übernommenen Pro-

jekt noch keine Transformation der S100-konformen Radarnachricht zu einer NMEA2000 Radarnachricht. Des Weiteren existiert im NMEA2000-Standard keine Nachricht, die die für diese Komponente relevanten Informationen des Radars beinhaltet. Aus diesen Gründen muss zunächst eine NMEA2000 konforme Nachricht erzeugt werden, und im Anschluss die Transformation der S100-konformen Radarnachricht zu dieser NMEA2000 konformen Radarnachricht implementiert werden.

Zur Erzeugung einer NMEA2000 Radarnachricht muss diese in der „Schemas"-Komponente in die Datei `nmea2000-description.xml` eingefügt werden. Es wurde sich dazu entschieden, die Parameter Gruppennummer (PGN) 000002 für Radar zu verwenden. Neben Name, Länge und Priorität der Nachricht an sich, muss für jedes Feld der Nachricht dessen Name, BitOffset, BitLength, Format, Unit, Scale und Beschreibung angegeben werden. Die einzelnen Felder sind Schiffs ID, Längengrad, Breitengrad, Zeitstempel, Kurs und Geschwindigkeit.

Des Weiteren muss in der Datei `s100-to-nmea2000.xsl` der Komponente „Schemas" die Transition von der S100 Radarnachricht zur NMEA2000-konformen Radarnachricht eingefügt werden. Hierbei wird für jedes Feld der NMEA2000 Nachricht der Wert aus den in der S100 Nachricht gegebenen Informationen berechnet.

UDP-Input-Adapter Um die AIS-, Radar-, eigene Position- und eigener Kurs-Nachrichten zu empfangen, wird ein Adapter benötigt, über den die Nachrichten vom Hub zur Komponente gelangen können. Es wurde sich dazu entschieden, hierfür einen UDP-Input-Adapter einzusetzen. Hierfür müssen in der Datei `hub-sim.xml` der Komponente „Polymorphic Interface Instance" ein OutputAdapter *Udp-Output-Adapter-Dynamic-Object-Identification* sowie eine Pipe *Polymorphic-Interface-to-Dynamic-Object-Identification* angelegt werden.

In der Komponente dynamische Objektidentifizierung muss um die Nachrichten zu empfangen ein udpInputAdapter implementiert werden. Diesem wird ein InputListener gesetzt. Dieser Listener wird direkt beim Starten der Komponente erzeugt und beinhaltet, was mit den hereinkommenden Bytes geschehen soll. Im Falle der oben genannten Nachrichten werden die Nachrichten mittels NMEA2000InputTranslator in einen xml-String umgewandelt und anschließend zur Verarbeitung weitergereicht.

Verarbeitung der NMEA2000 Nachrichten Nachdem die NMEA2000 Nachrichten in einen XML-String umgewandelt wurden, werden sie der Klasse ReadInformation übergeben, die für die Verarbeitung und Speicherung der Nachrichten zuständig ist. Hier wird als erstes nach der PGN,

also nach Art der Nachricht, unterschieden. Bei Radarnachrichten wird die Methode processRadarValues, bei dynamischen AIS-Nachrichten die Methode processAISValues, bei statischen AIS-Nachrichten die Methode processAISSize, bei der eigenen Position die Methode processOwnPosition und beim eigenen Kurs die Methode processVesselHeading aufgerufen.
Die relevanten PGN sind in Tabelle 6.1 zu sehen. Alle anderen hereinkommenden Nachrichten werden ignoriert.

Nachricht	PGN der realen Schiffe	PGN der simulierten Schiffe
Radar	000002	000006
dynamisches AIS	129038; 129039	000004
statisches AIS	129794; 129810	000005
eigene Position	129025	000007
eigener Kurs	127250	

Tabelle 6.1.: relevante PGN's

Im Folgenden werden einige Methoden der Nachrichtenverarbeitung näher erläutert:

processRadarValues: Falls die Schiffs ID in der Nachricht vorhanden ist, werden Längengrad und Breitengrad aus der Nachricht ausgelesen. Mittels dieser Werte wird geprüft, ob das Schiff innerhalb eines bestimmten Radius um das eigene Schiff liegt. Dieser Radius liegt bei zwei Seemeilen. Falls ja, werden die weiteren Werte (Zeitstempel, Kurs, Geschwindigkeit) ausgelesen. Diese Werte werden nun alle gespeichert (siehe nächster Abschnitt).

processAISValues: Diese Methode geht zum Auslesen der Werte aus der AIS-Nachricht äquivalent zur Methode processRadarValues vor und wird daher nicht näher beschrieben.

processAISSize: Wenn die Schiffs ID in der Nachricht vorhanden ist, werden Länge und Breite des Schiffs ausgelesen. Es wird geprüft, ob bereits ein Schiff mit dieser ID gespeichert ist. Falls ja, werden bei diesem Schiff die Länge und Breite auf die ausgelesenen Werte gesetzt. Falls nein, wird zuerst ein neues Schiff in der Speicherstruktur angelegt und im Anschluss die Länge und Breite gesetzt.

processOwnPosition: Es werden die Werte der Felder Breitengrad und Längengrad ausgelesen und in double-Werte umgewandelt. Anschließend werden sie in der Klasse ShipEnvironment, welche die Eigenschaften des eigenen Schiffs und dessen Umgebung behandelt, abgespeichert. Zusätzlich wird sich in der Klasse ReadInformation gemerkt, dass eine eigene Position vorhanden ist, da nur in diesem Fall die Nachrichten der Schiffe auf eine bestimmte Umgebung begrenzt werden können.

processVesselHeading: Es wird der Wert, der im Feld des Kurses steht ausgelesen, in ein double-Wert umgewandelt, und als Kurs in der Klasse ShipEnvironment abgespeichert.

Vor dem Abspeichern werden die Werte Breitengrad, Längengrad und Zeitstempel in einen Double umgewandelt. Des Weiteren wird der Zeitstempel bearbeitet, da der gegebene Zeitstempel nur die Sekunden beinhaltet. Das heißt, nach der 59 kommt wieder eine 0. Daher wird neben dem ausgelesenen Zeitstempel ein neuer erzeugt, der fortlaufend durch alle ankommenden Nachrichten eines Schiffes ist.

Speicherstruktur der relevanten Daten Nachdem die Werte einer Nachricht ausgelesen und verarbeitet wurden, müssen sie abgespeichert werden. Dafür wurde ein DataStorage angelegt. Dieser DataStorage enthält vier verschiedene Storages: `radarDataStorage`, `aisDataStorage`, `staticAISDataStorage` und `cameraDataStorage`.

Der staticAISDataStorage beinhaltet eine ArrayList vom Typ staticAISData, welcher wiederum eine ID, eine Länge und eine Breite beinhaltet. Hier werden die Größeninformationen zu Schiffen abgespeichert, zu denen noch keine weiteren Daten vorhanden sind.

Der cameraDataStorage beinhaltet eine ArrayList in der HashMaps mit den relevanten Informationen Längengrad, Breitengrad, Schiffsbreite, Schiffslänge, Kurs, Geschwindigkeit und Schiffs ID gespeichert werden. Näheres hierzu im Unterkapitel Objektdetektion.

Im `radarDataStorage` beziehungsweise im `aisDataStorage` befindet sich jeweils eine ArrayList mit radarData bzw. aisData welche beide von der Klasse Data erben. Diese Klasse beinhaltet einen Ringbuffer einer bestimmten, einstellbaren Größe, in welchem HashMaps mit den Attributen Breitengrad, Längengrad, Originalzeitstempel, neuer erstellter Zeitstempel, Kurs, Geschwindigkeit und Zeitpunkt des Abspeicherns gespeichert werden. Durch diesen Ringbuffer ist es möglich, die letzten ankommenden Nachrichten eines Schiffes zu speichern und somit die Datenfusion nicht nur auf einem Informationsblock, sondern auf mehreren auszuführen. In der Klasse aisData ist neben dem Ringbuffer noch die Schiffsgröße gespeichert.

Datenfusion

Die grobe Einteilung und mögliche Arten der Durchführung einer Datenfusion wurden bereits im Abschnitt 5.4 erläutert. Nachfolgend wird die konkrete Umsetzung gegliedert in Datenassoziation und -regression beschrieben.

Dynamische Objektidentifizierung - Datenassoziation In diesem Abschnitt wird die Realisierung des ersten Teils der Datenfusion beschrieben, der Assoziation. Bei der Assoziation soll festgestellt werden, welches mittels AIS erkannte Schiff das selbe Schiff wie welches mittels Radar erkannte Schiff ist.

Während der Planungsphase kam die Frage auf, woran genau festgelegt werden soll, wann zwei Schiffe assoziiert werden sollen. Zur Verfügung standen die Größe des Schiffs, Geschwindigkeit, Kurs, Längengrad und Breitengrad, da diese Werte in der Radar- und der AIS-Nachricht vorhanden sind.

Da bei den Radarnachrichten die Größe des Schiffs allerdings sehr ungenau ist, wurde die Größe für den Vergleich ausgeschlossen. Die Geschwindigkeit, der Kurs und die Position wurden alle als wichtig befunden.

Für die Assoziation wurde zunächst der bereits im Abschnitt Stand der Forschung: Datenfusion AIS und Radar beschriebene Ansatz der Fuzzy Funktionen umgesetzt. Da hierbei allerdings die Zeit keine Rolle spielt, und diese für sehr Wichtig erachtet wurde, ist man zu dem Entschluss gekommen, eine Abwandlung von Greubs Ansatz [Gre17] umzusetzen und die Positionen zu vergleichen. Da bei diesem Ansatz aber auch die Zeit außen vor gelassen wurde, werden nun Funktionen durch Positionen über die Zeit gelegt und diese miteinander verglichen. Hierbei können die Angaben Geschwindigkeit und Kurs ignoriert werden, da sie implizit in den Funktionen mit drin stecken.

Über die Zeit wurde festgestellt, dass die Radardaten sehr ungenau sind und springen. Das heißt, die erkannten Positionen über die Zeit sind sehr um die eigentlichen Positionen verteilt. Daher wurde beschlossen, die Radardaten vor dem Erstellen der Funktion zu filtern. Hierfür soll ein Alpha-Beta-Filter eingesetzt werden, um die erkannten Positionen den eigentlichen Positionen anzunähern.

Der Ablauf der Assoziation ist im Ablaufdiagramm Abbildung 6.12 zu sehen. Wie in Abschnitt Datenmanagement bereits beschrieben, werden die letzten Nachrichten (vier bei AIS und 15 bei Radar) eines Schiffes gespeichert. Sobald dieser Speicher voll ist, wird eine Assoziation gestartet. Hierfür werden zunächst Funktionen für Längengrad beziehungsweise Breitengrad über die Zeit erstellt. Diese Funktionen werden im Anschluss mit allen Funktionen vom anderen Schiffstyp verglichen und es wird eine durchschnittliche Entfernung berechnet. Wenn diese kleiner als ein bestimmter Schwellwert s ist, wird sich das Schiff und die Entfernung gemerkt. Anschließend werden diese gespeicherten Schiffe nach der Entfernung aufsteigend sortiert. Angefangen

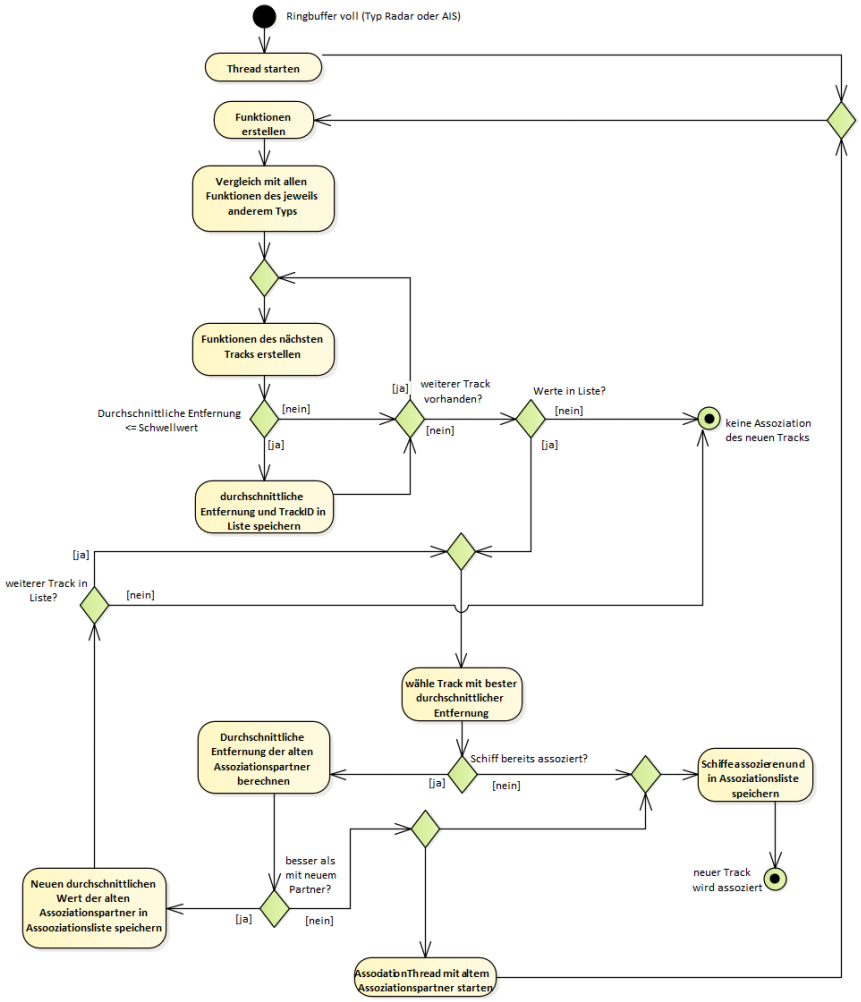


Abbildung 6.12.: Aktivitätsdiagramm der Datenassoziation

mit der kleinsten Entfernung wird überprüft, ob das Schiff bereits mit einem anderen Schiff assoziiert wurde. Falls nicht, werden die beiden Schiffe assoziiert und in eine Assoziations-Liste gespeichert. Ist dies doch der Fall, wird überprüft, ob die Assoziation besser, also die Entfernung kleiner, als die der alten Assoziation ist. In diesem Fall werden die neuen Schiffe assoziiert und für das nun nicht mehr assoziierte alte Schiff wird die Assoziation genauso durchgeführt. Für den Fall, dass das Schiff bereits assoziiert war, und die alte Assoziation besser als die jetzige ist, wird die alte Assoziation beibehalten und es wird sich das Schiff mit der nächst kleinsten Entfernung angeschaut.

Nachdem die Funktionen verglichen wurden, wird mittels eines Schwellwertes wie oben bereits beschrieben festgelegt, ob die Schiffe assoziiert werden könnten oder nicht. Die International Association of Marine Aids to Navigation and Lighthouse Authorities (IALA) gibt in ihrem Navigationsleitfaden eine Ungenauigkeit für Radarsysteme von 1,5% der maximalen Reichweite bzw. 70 Meter als Richtwert an [IA01]. Aufgrund des verwendeten Radars (Navico Broadband 4G Radar) wird der Schwellwert s auf 70 Meter festgelegt. Bei Verwendung eines anderen Radarsystems kann der Schwellwert in der Konfigurationsdatei angepasst werden.

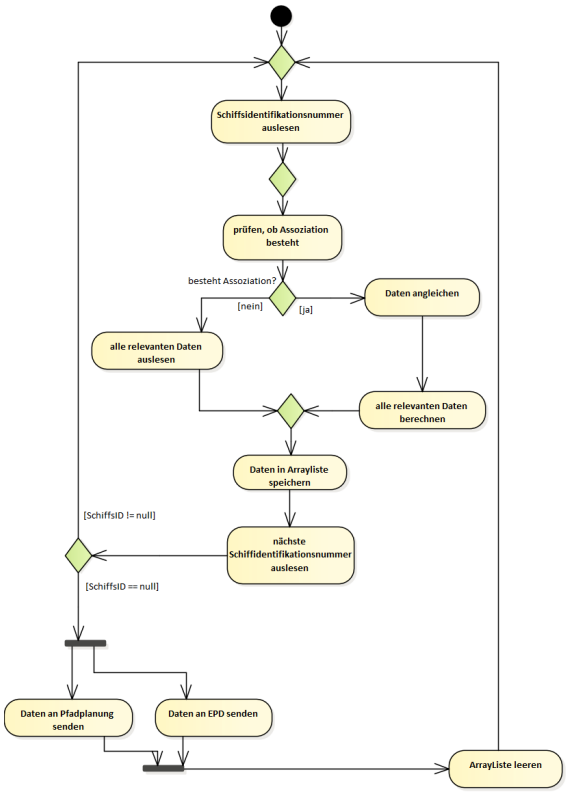


Abbildung 6.13.: Aktivitätsdiagramm der Datenregression

Datenregression Die Datenregression gleicht die benötigten Informationen der assoziierten Schiffen an. Hierbei werden wie bereits im Konzept beschrieben, die Werte für Schiffsgröße, -breite, Position, Kurs und Geschwindigkeit aus den assoziierten statischen und dynamischen AIS-Nachrichten sowie Radarnachrichten berechnet.

Wie die Datenregression genau abläuft, zeigt das Aktivitätsdiagramm Abbildung 6.13. Für jeden Eintrag im DataStorage wird geprüft, ob eine Assoziation besteht. Es können drei Fälle eintreten, im Folgenden wird beschrieben, welche Daten in den Fällen jeweils weitergeleitet werden:

Schiff wird mit Radar und AIS erkannt Für die Position wird jeweils der Mittelwert der Längen- und Breitengrad Angaben für die AIS- und Radardaten berechnet. Der Kurs- und Geschwindigkeits-Wert wird von der dynamischen AIS-Nachricht verwendet, da dieser genauer ist, als bei der Radarnachricht. Damit die Pfad-Planungs-Komponente alle Zellen blockt indem ein Schiff sein könnte, wird die Größe des Schiffes entsprechend angepasst. Als Basis wird die Größe und Breite -falls vorhanden- aus den Static AIS-Nachricht verwendet, da die Angaben zu Größe und Breite, welche in der Radarnachricht angegeben sind, zu ungenau sind. Um die neue Größe zu berechnen, wird überprüft wie das Schiff ausgerichtet ist. Dann wird der Abstand der beiden angegebenen Positionen des Schiffes berechnet und je nachdem wie das Schiff ausgerichtet ist auf die Breite oder die Länge addiert. Ergebnis ist ein Rahmen, indem das Schiff mit Sicherheit positioniert ist und der die komplette eigentliche Größe des Schiffes miteinschließt.

Schiff wird nur mit AIS erkannt In diesem Fall werden die Werte Kurs, Geschwindigkeit, Längengrad und Breitengrad aus der letzten dynamischen AIS-Nachricht herausgelesen und die Schiffsbreite und -länge aus den neusten vorhandenen statischen Schiffsdaten.

Schiff wird nur mit Radar erkannt Die Informationen zu Kurs, Geschwindigkeit, Breitengrad und Längengrad werden von der neusten Radarnachrichten verwendet. Da die angegebene Länge und Breite des Schiffes zu ungenau ist, wird hierfür die Länge von 20 Meter eingesetzt. Dies ist die Obergrenze von nicht ausrüstungspflichtigen Schiffen mit AIS. Daher kann davon ausgegangen werden, dass wenn ein Schiff, welches mit Radar erkannt wurde und nicht assoziiert wurde, unter dieser Grenze liegt.

Senden der relevanten Schiffsinformationen

Nach der Ermittlung der relevanten Schiffsinformationen in der Regression müssen diese an die Komponenten Pfad-Planung und EPD gesendet werden. Dies soll mittels HTTP und JSON geschehen.
Die relevanten Daten werden pro Schiff in eine HashMap gespeichert. Diese HashMaps wiederum werden in einer Liste zusammengefasst, die dann mittels implementierten JSONOutputAdapter versendet wird. Hierfür muss die Liste vorher zu einem JSON-Objekt umgewandelt werden. Dies geschieht in den Klassen OutputToPathPlanning und OutputToEPD.

Diese Liste wird alle 1,5 Sekunden erstellt und im Anschluss versendet. Hierdurch wird gewährleistet, dass die Komponenten Pfad-Planung und EPD immer auf den neusten Stand gesetzt werden.
An die EPD wird neben den Ergebnissen der Regression auch die in die Komponente reinkommenden AIS- und Radar-Nachrichten auf die selbe Art und Weise gesendet. Dies dient der Überwachung und des Testens der Datenfusionierung.
Für die EPD wird an den Port 7000 mit dem Kontext /doiDataInput gesendet, für die Pfad-Planung an 8282 und /dynamicObjects. Dies kann bei Änderungen in der Konfigurationsdatei angepasst werden.

Löschen alter Schiffe

Wenn nach einer bestimmten Zeit keine Nachrichten eines Schiffes mehr abgespeichert werden, liegt dies vermutlich daran, dass das Schiff nicht mehr im Umkreis um das eigene Schiff liegt. Da das Schiff nicht mehr da ist, sollte es auch aus der Liste aller Schiffe gelöscht werden. Hierfür wird ein Timer implementiert der den DataStorage „säubert“. Dieser Timer ist als Thread implementiert, sodass er neben dem Rest der Komponente arbeitet. Alle sechs Sekunden überprüft er, ob Schiffe gelöscht werden müssen. Falls ja, löscht er sie aus dem DataStorage und aus der Assoziationsklasse.
Ob ein Schiff gelöscht werden muss, ist an der aktuellen Zeit und dem Attribut der Zeit beim Abspeichern des aktuellsten Datensatzes eines Schiffes zu erkennen. Ist die aktuellste Zeit beim Abspeichern plus zehn Sekunden beim Radar, bzw. 30 Sekunden beim AIS, kleiner als die aktuelle Zeit, dann muss dieses Schiff gelöscht werden, da es in einer zu langen Zeitspanne keine neue Nachricht mehr gab.

6.4.2. Objektdetektion

Die kamerabasierte Hindernisdetektion wird mit Hilfe der neu angelegten Komponente Objektdetektion umgesetzt. Der grobe Ablauf der Komponente ist in Abbildung 6.14 abgebildet. Im Folgenden werden die einzelnen Bestandteile der Objektdetektion genauer erläutert.

Hardware Für die visuelle Wahrnehmung des physikalischen Testfelds wird eine Kamera verwendet, welches ein Mono-Video Daten-Stream erstellt. Hierbei ist die Schnittstelle der Software so programmiert, dass die Kamera austauschbar ist. Da die angeschlossene Hardware deutliche

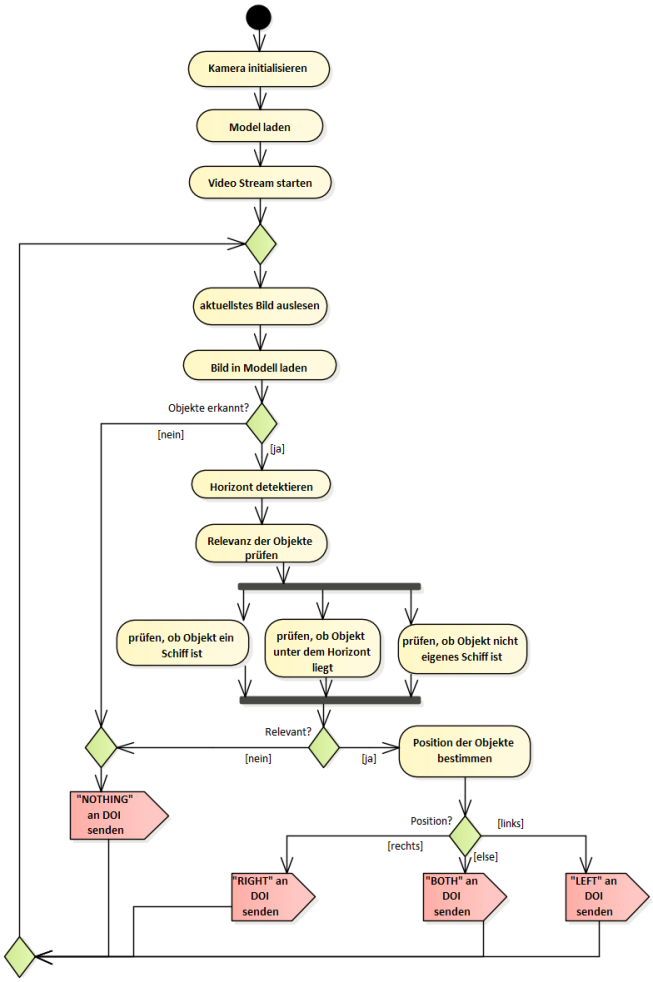


Abbildung 6.14.: Ablaufdiagramm der Objektdetektion

Einflüsse auf die Qualität der Objektdetektion hat, wird im MATE II Produkt zwei verschiedene Kameras getestet. Diese werden in Fahrtrichtung hinter der Windschutzscheibe in der Zuse angebracht. Verwendete Kameras im MATE II Produkt sind, die Logitech C920 HD Pro und die Blackmagic Micro Studio Camera 4K. Bei der Logitech C920 HD Pro handelt es sich um eine Webcam mit einer Auflösung von 1080p/30 FPS – 720p/30 FPS. Weiterführende Informationen zu den technische Daten finden sich in Quelle [SAa].

Eingang Kamerastream Um das aktuelle Kamerabild zur weiteren Verarbeitung zur Verfügung zu stellen, wird auf bereits implementierte und getestete Bibliothek[1] zurückgegriffen. Dies bietet mehrere Vorteile gegenüber einer neuen Implementierung. So besteht die Möglichkeit, verschiedene Aufnahme-Frameworks zu verwenden und so eventuelle Probleme schnell zu lösen. Angeschlossene Kameras und Capture Cards werden automatisch erkannt, und können anschließend ausgewählt werden. Das aktuelle Kamerabild wird als *BufferedImage* zur weiteren Bearbeitung zur Verfügung gestellt. Die Geschwindigkeit, in der die Kamerabilder zur Verfügung gestellt werden, hängt dabei von der Geschwindigkeit ab, in der die Bilder verarbeitet werden können. Ein neues Bild wird erst bereitgestellt, wenn das Letzte durch alle Schritte der Verarbeitung gegangen ist.

Neuronales Netz Für die Objekterkennung wurde sich nach umfangreicher Literaturrecherche für ein auf den COCO-Datensatz[2] vortrainiertes ssd_inception_v2 Model entschieden. Dieses Model kann über [Ten] heruntergeladen werden. Faktoren, welche zur Entscheidungsfindung beigetragen haben, sind, dass dieses Model die Bilder sehr schnell analysiert und dabei eine hohe Genauigkeit aufweist. [Ten]
Um dieses Model nutzen zu können, wird eine Klasse namens Model benötigt, welche mithilfe des ssd_inception_v2 Model die Objekte in einem Bild erkennt. Bei der Implementierung dieser Klasse wurde sich nach der Klasse DetectObjects auf GitHub von asimshankar gerichtet [asi18], sowie die Bibliothek Tensorflow genutzt, welche für das Bilden, Speichern, Laden und Ausführen von Modellen gebaut wurde.
Der Klasse Model muss beim Instanziieren das Model sowie eine Liste aller Label, auf die das Model trainiert ist, übergeben werden. Mittels der von asimshankar gegebenen Klasse StringIntLabelMapOuterClass wird aus der Datei, in der die Label aufgelistet sind, ein Array mit allen Labeln gebildet.
Um in einem Bild die Objekte zu erkennen, muss die Methode detectObjects aufgerufen werden. Hier wird zuerst aus dem übergebenen Bild ein Tensor geformt. Tensor ist ein multi-dimensionales Array vom Typ T und wird von Tensorflow gestellt. Wenn das Bild vom Typ 3Byte_BGR oder Int_RGB ist, wird es in einem byte-Array abgespeichert. Anschließend wird ein Tensor mit diesem byte-Array und einer gegebenen Form (hier: BatchSize = 1, gegebene Höhe und Breite des Bildes, Anzahl Kanäle = 3) erstellt.
Anschließend wird auf diesem Tensor-Bild das gegebene Model ausgeführt. Als Ergebnis lassen sich drei weitere Tensors generieren, scoresT, classesT, sowie boxesT, welche im Anschluss alle

[1] https://github.com/sarxos/webcam-capture
[2] http://cocodataset.org/#home

in Arrays des Typs float umgewandelt werden. In diesen Arrays befinden sich die Angaben zu allen gefundenen Objekten. Classes gibt an, von welchem Typ die gefundenen Objekte sind (zum Beispiel boat oder person), scores gibt an, zu wie viel Prozent das Objekt zu dieser Klasse gehört, und in boxes sind die Koordinaten der Eckpunkte der Box gespeichert, welche um das Objekt gezeichnet werden kann.
Im weiteren Verlauf werden nur jene Objekte betrachtet, bei denen der Prozentsatz höher als 50 Prozent liegt, da bei niedrigerem Prozentsatz vermutlich doch kein Objekt dieser Klasse vorhanden ist. Falls dies zutrifft, und das Objekt nicht das eigene Schiff ist (siehe unten), dann wird das Objekt in eine Liste der gefundenen Objekte gespeichert. Diese Liste wird zum Schluss zurückgegeben.

Horizonterkennung Da der Horizont im Prinzip eine Linie zwischen Meer und Himmel ist, wurde beschlossen, hierfür eine Liniendetektion einzusetzen. Hierbei wird sich an den Code aus [Opeb] gehalten und dieser an die gegebene Situation angepasst.
Der Ablauf der Horizonterkennung ist folgender: Nachdem das aktuellste Bild eingelesen wurde, wird auf diesem der CannyEdge-Filter angewendet, der alle Kanten im Bild detektiert. Im Anschluss werden mittels dem HoughLines-Algorithmus aus diesen Kanten alle Linien gefiltert und in einer Liste gespeichert. Hierbei wird die Polarschreibweise verwendet, das heißt zu jeder Linie existieren die zwei Parameter Rho und Theta. Durch schrittweises Ausprobieren wurde ermittelt, wie groß das Theta sein darf, damit eine Linie eher als horizontal als als vertikal zählt. Hierbei kam raus, dass Theta in dem Wertebereich (1,2) liegen muss.

Ist dies der Fall, wird die Linie in einer weiteren Liste gespeichert. Diese Liste wird im Anschluss durchlaufen und es wird diejenige Linie gesucht, die durchschnittlich am höchsten im Bild liegt. Diese Linie wird als Horizont angenommen.

Filtern der relevanten Objekte Das neuronale Netz gibt alle Objekte auf welches dieses trainiert wurde als ArrayList vom Typ DetectedObject zurück. Diese Objekte sind neben Schiffen z. B. auch Personen, Flugzeuge oder Autos. Diese Klasse beinhaltet die Attribute label, score, y_min, x_min, y_max, x_max. Zusätzlich werden Attribute, ob sich ein Objekt unter dem Horizont befindet, hinzugefügt.
Um nun nur die relevanten Objekte aus den zurückgegeben Objekten herauszufiltern, wird im ersten Schritt geprüft, ob ein Objekt das eigene Schiff ist. Dies passiert mit der Methode isOwnShip welcher die Position des Objektes übergeben wird. Diese Funktion ist notwendig, da die Kamera auf der Zuse nur so platziert werden kann, dass das eigene Schiff mit auf dem Bildausschnitt zu sehen ist. Die Überprüfung, ob es sich um das eigene Schiff handelt, wird mittels der Koordinaten der Box des Objektes sowie einer prozentualen Angabe der Position des eigenen Schiffs aus der Konfigurationsdatei geprüft. Diese Angabe der eigenen Position gibt an, bei wie viel Prozent vom oberen Bildrand aus gesehen, das eigene Schiff anfängt. Wenn das gefundene Objekt unterhalb dieser Stelle im Bild liegt, wird es als eigenes Schiff angesehen und nicht weiter betrachtet.
Ist das detektierte Objekt nicht das eigene Schiff wird im zweiten Schritt festgestellt, ob das Objekt unter dem Horizont liegt. Dies geschieht mithilfe des zuvor detektierten Horizonts (siehe vorherigen Abschnitt). Die Methode checkIfUnderHorizon prüft, ob das Objekt unter diesem Horizont liegt. Als letzten Schritt wird geprüft, ob das detektierte Objekt ein Schiff ist (label: boat). Für alle relevanten Objekte wird anschließend die Position bestimmt, um diese an die dynamische Objektidentifizierung zu senden.

Positionsbestimmung Für alle relevanten Objekte wird mithilfe der Klasse ShipPositionCalculator berechnet auf welchem Bildabschnitt sie sich befinden. Die Dynamische Objektidentifizierung erwartet von der Objektdetektion einen String welcher die Ausprägung `NOTHING`, `RIGHT`, `LEFT` oder `BOTH` annehmen kann. Um den Schwankungen beim Detektieren eines Schiffs entgegenzuwirken (z. B. Schiff wird abwechselnd auf Bildern erkannt bzw. nicht extistierendes Schiff wird erkannt), wird ein Schiff nur weitergeleitet, wenn es auf einer bestimmten Anzahl an Bildern detektiert wurde. Diese Parameter können in der Konfigurationsdatei angepasst werden.

Kommunikation mit der Dynamischen Objektidentifizierung Die Übertragung der Positionsnachrichten der detektierten Schiffe an die Dynamische Objektidentifizierung geschieht mittels einer UDP Verbindung. Zum Versenden wird eine SendShipOutput-Klasse implementiert. Diese versendet über die UdpOutput-Klasse die Positionen der detektierten, relevanten Schiffe.

6.5. Regelungstechnik

Die Regelungstechnik von MATE II wurde mit Hilfe eines im folgenden *Controller* genannten MATLAB/Simulink-Projekts umgesetzt. Aus den erhobenen Anforderungen an die Regelungstechnik (siehe Abschnitt 4.8) ergab sich, dass der von der Projektgruppe MATE I entwickelte Controller als Basis für das Produkt genutzt werden kann.

Im Ebenenmodell des autonomen Fahrens (vgl. [Ran95]) stellt der Controller die unterste Ebende dar: Die Regelungsebene. Auf dieser Ebene werden atomare Steuerbefehle an das physikalische System umgesetzt. Von der taktischen Ebene (der Pfad-Planung) erhält der Controller zwei Wegpunkte, die eine Strecke bilden, der das Schiff autonom folgen soll. Ebenfalls wird eine gewünschte Geschwindigkeit übergeben. Der Controller versucht nun die gewünschten Werte zu erreichen. Dazu ist es ihm möglich, die aktuelle Drehzahl der Schiffsschraube und den aktuellen Ruderwinkel des Schiffsruders zu setzen. Bei der Implementierung des Controllers wurden hierzu zwei separate Subcontroller entworfen.

Kommunikation

Die Kommunikation des Controllers geschieht mittels UDP Verbindungen und einer Kodierung der Nachrichten in NMEA2000 (siehe Abschnitt 3.2). Empfangen werden Steuersignale (zu setzender Ruderwinkel und Schiffsschraubengeschwindigkeit) der Virtual Handles aus der EPD und Wegpunkte für die autonome Fahrt von der Pfad-Planung. Nach der Berechnung sendet der Controller mit eine Abtastzeit von 400 Millisekunden das aktuelle berechnete Ruder- und Schiffsschraubenkommando an den RaspberryPI, welcher die Hardware anspricht (siehe Abschnitt 6.5).

Für Informationen zum Parsing der Nachrichten sei auf die Abschlussdokumentation von MATE I verwiesen [MAT17, Kap. 6.7].

Speed-Controller

Der erste Subcontroller ist für das Erreichen der korrekten Geschwindigkeit verantwortlich. Als Eingaben erhält dieser eine zu erreichende Soll-Geschwindigkeit als auch eine momentane Ist-Geschwindigkeit in *m/s*. Auf Basis des Fehlers berechnet nun ein PID-Controller (Proportional–Integral–Derivative Controller) die zu setzenden *rpm* der Schiffsschraube.

Rudder-Controller

Der zweite Untercontroller setzt parallel zum Speed-Controller den Ruderwinkel. Dieser berechnet aus den beiden Wegpunkten der Pfad-Planung, der eigenen Postion und dem eigenen Heading einen Ruderwinkel. Die Berechnung ist dabei in drei Komponenten gegliedert.

XTE und Peilung Die erste Komponente berechnet aus den beiden Wegpunkten einen Winkel aus der Graden mit Bezug zu Norden (Peilung/bearing). Und den Cross Track Error (XTE), der den Abstand der eigenen Position zu der aus den beiden Wegpunkten aufgespannten Graden berechnet.

LOS Controller Der Line-of-Sight-Controller berechnet nun aus dem XTE, der Peilung und dem eigenen Heading einen zu erreichenden Kurs.

Course Controller Bei dem Course-Controller handelt es sich um einen PID-Controller, der aus der Abweichung zwischen aktuellem Heading und zu erreichendem Kurs den Ruderwinkel bestimmt.

Manuelle Steuerung

Die manuelle Steuerungsfunktion des Controllers ist ein paralleler Pfad, welcher durch setzten des Routestatus auf 1 in der NMEA2000-Nachricht mit *PGN:130066* aktiviert wird. In diesem Fall werden die Nachrichten *PGN:127245* für den Ruderwinkel und *PGN:000001* für die Schiffsschraubengeschwindigkeit akzeptiert. Diese werden in diesem Fall jedoch unverändert an den Ausgang des Controllers angelegt.

Dieser Aufbau erlaubt es, jederzeit die autonome Fahrt zu überschreiben um die Kontrolle über das System zurück zu erhalten. Weiterhin ist durch diesen Aufbau der Controller die einzige Komponente, die Steuersignale an die Hardware sendet. Durch die Möglichkeiten zum Monitoring von Werten in MATLAB/Simulink ist damit an einer Stelle ersichtlich, welche Werte zu den einzelnen Zeitpunkten gesetzt wurden.

Hardware Umsetzung

Die für das tatsächliche Einstellen der Schiffsschraube und Drehzahl des Motors genutzte Komponente wurde nicht von MATE II entwickelt. Tatsächlich geht diese aus einer an der *Carl von Ossietzky Universität Oldenburg* veröffentlichten Masterarbeit hervor. Aufgrund dessen wird in

diesem Abschnitt nur oberflächlich auf die verwendete Technik eingegangen. Genauere Informationen sind in [Tan17] zu finden.

Herz der Implementierung ist ein Raspberry Pi 3B. Dieser empfängt die Sollgeschwindigkeit mittels UPD über Ethernet. Aufgrund der Eingaben werden neue Sollwerte, sowohl für Ruder als auch Motor berechnet. Diese wird mithilfe zweier PI-Regler durchgeführt. Wobei jeweils ein PI-Regler für die Berechnung der Motordrehzahl, bzw. des Ruderwinkels verantwortlich ist. Die neuen Sollwerte werden anschließend an die Navibox geschickt. Von dort aus werden mithilfe der sogenannten *MAST-Software* die Anpassung des Ruderwinkels und der Motordrehzahl vorgenommen. [Tan17]

Identifikation bestehender Probleme

Wie bereits erwähnt wurde im Rahmen der Projektübernahme mit einem bereits existierenden Controller gestartet. Aufgrund der Tatsache, dass der zuvor von MATE I entworfene und implementierte Controller abschließend jedoch leider nicht erfolgreich getestet wurde, musste hier direkt mit dem Testen begonnen werden um Fehler zu identifizieren und beseitigen zu können. Zur Erfüllung dieser Aufgabe wurde eine Vielzahl von Modultests entworfen, die die Funktionalität des Controllers abdecken. Hierzu wurden folgende Module des Controllers getestet:

- **NMEA2000 Kommunikation** Prüfung der Korrektheit von empfangenen Statusinformationen sowie berechneten Steuerbefehlen.
- **Geschwindigkeitsregler** Sprungantwort auf aktuelle und gewünschte Werte (PID-Controller).
- **Ruderregler** Sprungantwort auf aktuelle und gewünschte Werte (PID-Controller), Berechnungen zur Konvertierung zwischen Wegpunkten und aktueller Position zu *XTE* und *Bearing*, Validierung von Winkelberechnungen.
- **Manuelle Steuerung** Überbrückung der automatischen Regler.

Zur Durchführung der Tests wurden die Analysetools von MATLAB/Simulink genutzt um Werteveränderungen wie z.B. die über die Zeit veränderliche Sprungantwort zu überwachen. So ist es zum Beispiel möglich eine Zeitserie an beobachteten Geschwindigkeitsdaten in den Controller einzuspeisen und die Sprungantwort des PID-Reglers über die Zeit zu überwachen. Zudem

wurden für einzelne Modultests konstante Werte in Berechnungsmodule eingespeist um nur lokale Berechnungen testen zu können und sich von den Gesamtkomponententests zu distanzieren. Weiterhin wurden die einzelnen Bestandteile der Regelungskomponente mit verschiedensten Werten getestet um auch Randfälle abzudecken (bspw. negatives Heading, Wegpunkte in acht Richtungen, etc.). Die einzelnen schematisch definierten Testfälle befinden sich im Anhang. Dort sind weitere Details zur verwendeten Methodik zu entnehmen (siehe Testcases „TOb-Cont-Internal-*“).

Resultate

Nach der Übernahme der von MATE I angefertigten Regelungskomponente galt es den beschriebenen Fehler zu finden. Nachdem die Kommunikation überprüft wurde und eine inkorrekte Übertragung der Daten ausgeschlossen werden konnte, wurden die einzelnen PID-Regler getestet. Auch mit verschiedenen Parametern konnte keine direkte Modulfehlfunktion identifiziert werden. Auch konnten Situationen reproduziert werden, in denen die Regelungskomponente je nach Eingabedaten Ruderbefehle in beide Richtungen (positive und negative Ruderkommandos) erzeugte und nicht wie zunächst vermutet, nur Befehle für eine Richtung. Somit konnte ausgeschlossen werden, dass die Regelungskomponente bei korrektem Dateneingang falsche Ruderkommandos berechnet. Die weitere Evaluation des Controllers wird in Kapitel 7 beschrieben.

Genauigkeit

Da der Controller durch die langsame Reaktionszeit im Wasser nicht die Genauigkeit der Pfad-Planung abbilden kann, ist auf dieser Ebene schon mit einem Fehler zu rechnen. Dieser ergibt sich daraus, dass die Wegpunkte die der Controller von der Pfad-Planung erhält, der 1. und 5. des aktuellen Pfades sind.

Um eine Abschätzung der Leistung des Controllers durchführen zu können, gilt es zunächst zu bestimmen, in welchem Genauigkeitsrahmen sich die Eingaben für den Controller selbst befinden. Vergleiche hierfür die Abbildung 6.15. Der grau dargestellte Pfad der Pfad-Planung besitzt in diesem Grid die maximal mögliche Auslenkung nach rechts auf dem 2. und 3. Wegpunkt und mit dem 4. und 5. Wegpunkt die Maximale nach links. Da der Controller jedoch nur den 1. und 5. Wegpunkt erhält, ergibt sich für diesen ein Sollpfad der in der Grafik durch die rote Linie dargestellt ist.

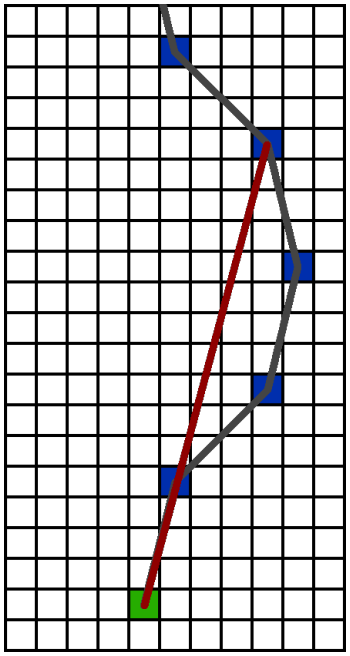


Abbildung 6.15.: Darstellung eines Worst-Case-Pfads des Controllers. Eigentlicher Pfad in Grau, Controllerpfad in Rot

Die Pfad-Planung arbeitet intern mit einer Darstellung durch Zellen. Durch die Verwendung von T-Neighbourhood (siehe Abschnitt 3.4.3) ist der Radius der Bewegung eingeschränkt, weshalb sich das Bild wie in Abbildung 6.15 dargestellt ergibt. Die maximale Abweichung E ergibt sich damit durch den kleinsten Abstand des 3. Wegpunkts zur Strecke aufgespannt von Wegpunkt 1 und 5.

$$cell = 4m$$
$$WP_1 = (0,0) \quad WP_3 = (4,7) \quad WP_5 = (4,15)$$
$$E = \frac{|(y_5 - y_1)x_3 - (x_5 - x_1)y_3 + x_5 * y_1 - y_5 * x_1|}{\sqrt{(y_5 - y_1)^2 + (x_5 - x_1)^2}} * cell = 8.2452m$$

Der in Abbildung 6.15 dargestellte Pfad stellt den Worst-Case-Fall dar, da die Auslenkungen maximal sind. Damit ist auch die höchstmögliche Abweichung vom eigentlichen Pfad ermittelbar. Bei einer aktuell verwendeten Zellengröße von vier Metern ergibt sich eine größtmögliche Abweichung von $E = 8,25m$ zwischen dem Pfad-Planungs-Pfad und dem Controller-Pfad.

Da es sich um das Worst-Case-Szenario handelt, ist in der Praxis jedoch oft mit einer geringeren initialen Abweichung zu rechnen.

Brine

Um den Controller im Labor testen zu können, wurde eine Simulation benötigt, die in der Lage ist die physikalischen Eigenschaften sowohl eines Schiffes, als auch die Einflüsse der Umgebungsparameter auf dieses realitätsgetreu nachzustellen. Die PG MATE II entschied sich für eine vom OFFIS entwickelte und bereits im Einsatz befindliche Teilkomponente (*Brine*) des *eMIR* Projekts. Brine ist eine 3D Simulation mit sechs Freiheitsgraden. Außerdem haben Wind, Wellen und Strömung Einfluss auf das eigene simulierte Schiff. [Brine]

Um eine Komunkikation zwischen Controller und simulativem Testfeld zu ermöglichen, mussten lediglich die Steuerbefehle, die der Controller schickt in das JSON Format umgewandelt werden. Zusätzlich mussten alle für den Controller benötigten Ausgaben der Simulation vom JSON Format in das NMEA2000 Format umgewandelt werden. Gerade bei der Suche nach dem Fehler hat sich diese Form des Testens als effizient herausgestellt. So konnte sowohl der Fehler identifiziert und behoben werden. Anschließend musste man keinen Test im physikalischen Testfeld durchführen, sondern konnte die grundlegende Funktionalität mithilfe des simulierten Testfelds überprüfen.

Lediglich bei der Findung der Parameter für das PID-Regelsystem erwies sich Brine nicht als hilfreich. Für diese Aufgabe war das implementierte Modell nicht genau genug auf das der Zuse angepasst. So musste diese Aufgabe im physikalischen Testfeld vor Ort durchgeführt werden.

6.6. Simulationsadapter

Um die Integration simulierter Daten in das physikalische Testfeld wie in Abschnitt 5.7 umzusetzen wird zum einen eine Komponente zum Empfangen, Transformieren und Versenden der XML Daten benötig und zum anderen ein Protocolhandler in HAGGIS. Die Komponente zur Verarbeitung der XML Daten wird im Folgenden als Simulationsadapter bezeichnet. Die Realisierung des Protocolhandlers in HAGGIS und des Simulationsadapter wird im folgenden Kapitel beschrieben.

Protocolhandler HAGGIS

Um Ucore konforme XML Daten aus HAGGIS zu versenden wird ein Protocolhandler in HAGGIS benötigt. HAGGIS stellt einen NMEA0183ProtocolHandler und einen SymbolicRadarProtocolHandler zur Verfügung. Die von diesen Protocolhandlern zum Versenden zur Verfügung gestellten Daten entsprechen nicht den benötigten Ucore konformen XML Daten.
Um den benötigten Datenstrom zu erhalten ist der UCoreProtocolHandler im Package uCoreXMLCodecs in HAGGIS implementiert worden. Der UCoreProtocolHandler erzeugt den erforderlichen Datenstrom mit Ucore konformen XML Dateien. Diese XML Dateien beinhalten die Daten der im Szenario dargestellten Schiffe und sind aufgrund der Komplexität der Szenarien zu groß um über UDP versendet zu werden. Daher werden die zu versendenden XML Dateien mittels GZIP aus dem Package java.util.zip komprimiert und im Anschluss über den durch HAGGIS bereitgestellten UDPTransportHandler versendet.

Simulationsadapter

Der Simulationsadapter unterteilt sich wie im Komponentendiagramm in Abbildung 6.16 in drei größere Komponentenbestandteile, deren Realisierung und Funktionsweise im Folgenden betrachtet werden.

UDP Schnittstelle Das Versenden der Daten aus HAGGIS erfolgt wie im vorangegangenen Abschnitt beschrieben über einen UDPTransportHandler. Um die über den UDPTransportHandler ausgehenden Daten zu empfangen ist im Simulationsadapter als Kommunikationsschnittstelle die UDP Schnittstelle UDPGZIPInputAdapter implementiert. Diese empfängt die versendeten Daten und übernimmt hierbei direkt das Dekomprimieren der XML Daten.

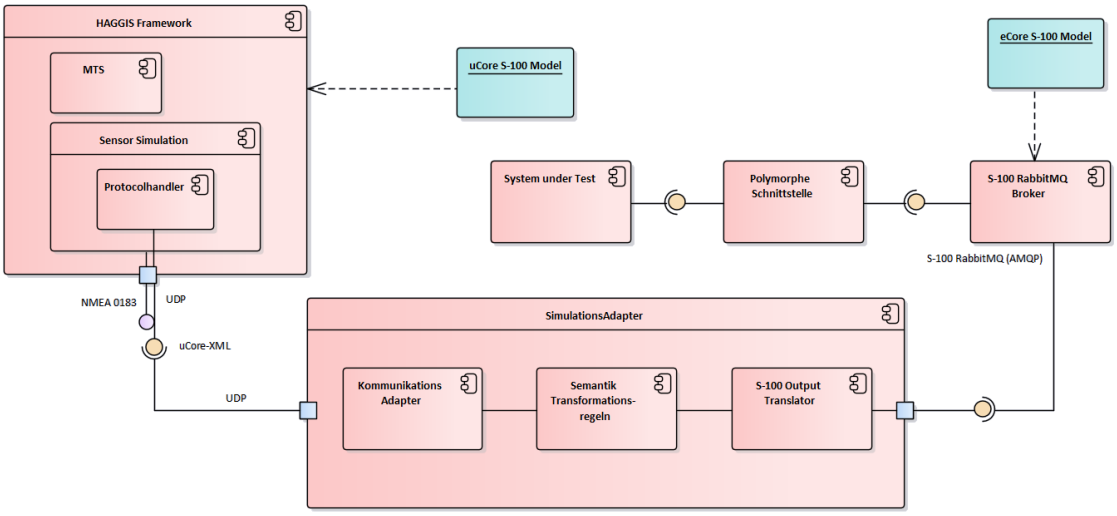


Abbildung 6.16.: Komponentendiagramm zum Entwurf des Simulations-Adapters

Transformation der XML Daten Die zu transformierenden Daten für das Szenario aus HAGGIS werden von HAGGIS als Ucore konforme XML Daten versendet. Um die Daten der simulierten Schiffe in das physikalische Testfeld zu integrieren müssen die Ucore konformen XML Daten in das im physikalischen Testfeld verwendete Ecore XML Datenformat transformiert werden. Für die Transformation von XML Dateien kann eine XSL Transformation (XSLT) vorgenommen werden. Hierbei wird in einem XSLT-Stylesheet eine logische Baumstruktur eines XML Dokumentes angelegt. Die XSLT-Stylesheets werden mittels eines XSLT-Prozessors eingelesen, welcher mit der im XSLT-Stylesheet vorhandenen Struktur ein oder mehrere XML Dokumente in die gewünscht Form umwandelt. [MS]
Bei der Transformation von Ucore zu Ecore konformen XML Dateien muss berücksichtigt werden, dass in Ucore Artefakte vorhanden sind, die in Ecore nicht dargestellt werden können. Bevor die Transformationsregeln angelegt werden können muss sichergestellt werden, dass die Transformation keine Ucore Artefakte vernachlässigt und dadurch zu einer Verfälschung oder Unvollständigkeit der XML Dateien führt.
Für die Integration der simulierten Schiffe in das physikalische Testfeld sind Transformationsregeln zum Erstellen von statischen und dynamischen AIS-, Radar- und für Testzwecke Positionsnachrichten des das eigene Schiff darstellenden simulierten Schiffs nötig. Bei diesen Transformationsregeln sind keine Ucore Artefakte vorhanden die in eCore nicht dargestellt werden können, sodass hierdurch keine Probleme auftreten.
Im Simulationsadapter erfolgt die Transformation der XML Dateien mittels XSLT. Hierbei entste-

hen aus jeweils einer XML Datei, empfangen über den UDPGZIPInputAdapter, die XML Dateien für statische und dynamisch AIS-Nachrichten, sowie die Positionsnachrichten.
Um die Testszenarien möglichst realitätsnah zu gestalten werden, zu den bereits genannten Nachrichten, Radar Nachrichten benötigt. Die Simulation in HAGGIS stellt keine Radar Nachrichten für die simulierten Schiffe zur Verfügung. Dieses Problem löst der Simulationsadapter durch das Nachbilden von Radar Nachrichten anhand der Schiffsposition. Radar Nachrichten von realen Schiffen können eine Abweichung zur tatsächlichen Position haben. Diese Abweichung wird bei der Nachbildung der Radar Nachrichten durch eine normalverteilte Zufallsabweichung angewandt auf die Position des Schiffs berücksichtigt. Die Radar Nachrichten werden hierfür zunächst mittels XSLT erzeugt und im Anschluss durch die Zufallsabweichung angepasst.
Bei der Transformation der XML Dateien wird für jede Nachricht eine eigene ReferenceID beginnend mit den Ziffern 09XXXXX vergeben. Dies ist notwendig um eine Unterscheidung zwischen den Nachrichten der simulierten und den Nachrichten der realen Schiffe im Test machen zu können. Hierfür ist im Polymorphic-Interface die Transformation von S-100 zu NMEA2000 ebenfalls mit den neuen ReferenceID Werten angepasst, sodass für die simulierten Schiffe eine eigene PGN Nummer in den NMEA2000 Nachrichten verwendet wird.
Die Transformationen werden in der Reihenfolge Radar, statisches AIS, dynamisches AIS und Positionsnachricht durchgeführt. Die Positionsnachricht wird jedoch nur für Labortests benötigt.
Vor dem Versenden der Daten wird eine Serialisierung mittels des S100OutputTranslator durchgeführt, um das für das Polymorphic-Interface nötige Datenformat zu erhalten. Der S100OutputTranslator konvertiert die übergebene Nachricht zu einer äquivalenten EMF Ecore Modellinstanz. Diese Modellinstanz wird zu einem Array des Datentyps Byte umgewandelt und durch den S100OutputTranslator zurückgegeben. [MAT17, p. 149]

RabbitMQ Schnittstelle Für das Versenden der transformierten Daten wird eine RabbitMQ Schnittstelle verwendet. Die Konfigurationen der Schnittstelle befinden sich in der RMQConfig Datei.

6.7. Mate Control

Da die Verifikations- und Validierungskomponente MateControl nicht Teil des zu testenden Systems ist, soll hier nur kurz auf die technische Realisierung eingegangen werden. MateControl teilt sich grundsätzlich in den MateControl-Server und den zur Laufzeit des Gesamtsystems mehrfach instantiierten MateControl-Client auf.

Client

Die Kommunikation zwischen Server und Client wurde mithilfe von WebSockets implementiert. Daten werden zwischen dem Client und dem Server mithilfe eines eigenen Datenmodells im JSON-Format ausgetauscht. Dabei werden 3 Grundfunktionalitäten unterstützt:

- Das Senden von Log-Ausgaben zur Überwachung einer Komponente.
- Das Senden von sich über die Zeit ändernden numerischen Werten.
- Das Umleiten eines Datenstroms durch den MateControl-Server und zurück.

Hierbei wurde auf eine einfache Nutzbarkeit des MateControl-Clients wert gelegt. Dieser kann in beliebige Projekte als Maven-Dependency eingebunden werden. Die Verbindung zum Server muss nicht konfiguriert werden und wird automatisch mithilfe von UDP Broadcasts hergestellt, solange sie sich im selben Netzwerk befinden. Dies lässt sich mit dem Konfigurationsbeispiel wie in Listing 6.1 dargestellt beim Start der Komponente erreichen.

```
1    MateControl.getInstance().setComponentName("Komponenten-Name");
2    MateControl.getInstance().setComponentDescription("Komponenten-Beschreibung");
3    MateControl.getInstance().connect();
```

Listing 6.1: Bsp. Einbindung MateControl-Client

Die drei erwähnten Funktionen können jeweils durch eine Zeile Code aufgerufen werden. Zusätzlich werden Log-Ausgaben, die durch MateControl erzeugt wurden automatisch lokal und serverseitig gespeichert. Um das Bestehen der Verbindung zwischen Server und Client zu überprüfen wurde zusätzlich ein Heartbeat eingeführt, durch den alle drei Sekunden verifiziert werden kann, dass eine Verbindung besteht.

Server

Im Folgenden wird die Realisierung des Servers näher beschrieben.

Grundfunktionalität

Der MateControl-Server verwaltet die Verbindungen zu allen Clients und stellt zur Überwachung des Systems eine durch JavaFX realisierte Benutzeroberfläche bereit. Diese wurde nach dem Model View Controller Modell entworfen und implementiert. Über die Benutzeroberfläche können Log-Ausgaben abgelesen werden, die numerischen Werte visualiert werden und ein Datenstrom analysiert und modifiziert werden. Zusätzlich wurden serverseitig einige Hilfsmittel implementiert um einen S100-Datenstrom zu analysieren. So können beispielsweise Eigenschiffsinformationen und Fremdschiffsinformationen ausgelesen und dargestellt werden.

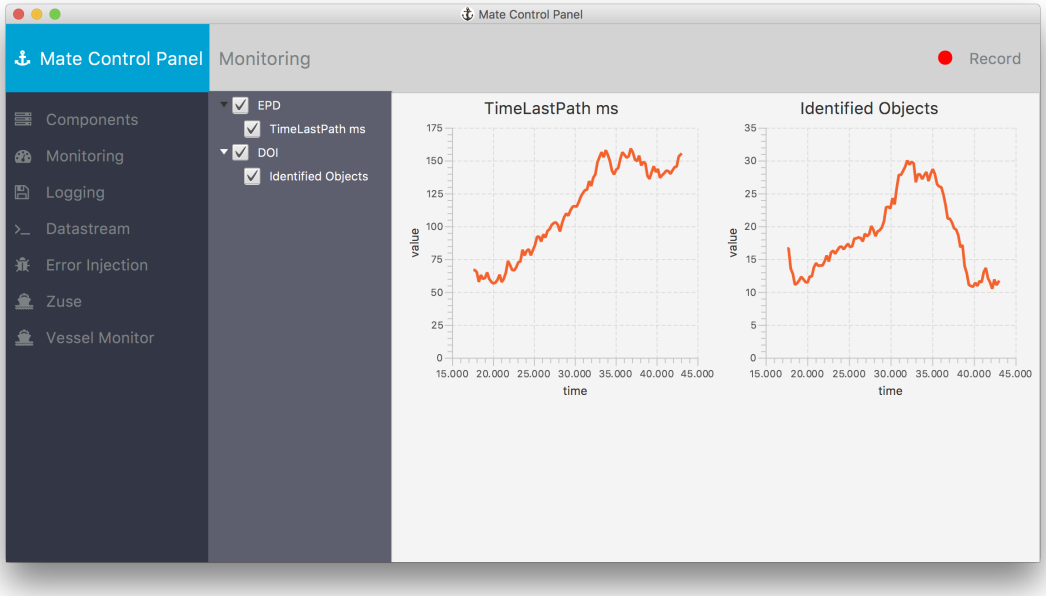


Abbildung 6.17.: Monitoring verschiedener Komponenten

Bewertung der Systemperformance

Zudem ist es möglich die gesamte Systemperformance für zuvor definierte Testszenarien zu bewerten. Hierzu wird zu einem Testszenario eine Soll-Route entworfen, die das System im Opti-

malfall abfahren sollte. Die Route kann im RTZ-Format in der MateControl Benutzeroberfläche geladen werden. MateControl berechnet daraufhin mit einer vorgegebenen Maximalabweichung r einen Bereich um die Route, indem für je zwei aufeinanderfolgende Wegpunkte des Sollpfads nach dem in Abbildung 6.18 dargestellten Schema ein Polygon erstellt wird.

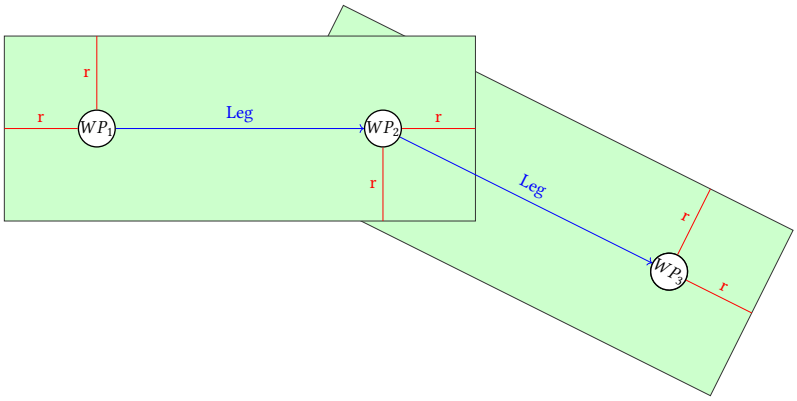


Abbildung 6.18.: Aufbau der Polygone aus einem Sollpfad

Diese Polygone werden dann zu einem einzelnen Polygon verbunden und mit der aktuellen Position wird während des Tests überprüft ob diese sich in dem zulässigen Polygon befindet, damit dieser Test als erfolgreich gilt. Zur Ausführungszeit des Testes kann dieser Bereich auf der EPD (siehe Abbildung 6.4) visualisiert werden und Live festgestellt werden, ob sich das Schiff im definierten Bereich befindet.

Fehlerinjektion

Für eine Fehlerinjektion ist es möglich durch den MateControl Server umgeleitete Datenströme zu verändern und somit Fehler in das System zu injizieren. Hierzu wird ein Adapter, der den MateControl Client einbindet zwischen den Sensordatenhub der Zuse (MAST) und dem zu testenden MATE II Produkt eingebunden. Dieser Adapter verbindet sich mit dem MateControl Server und leitet den S-100 Datenstrom der Navibox um, sodass im Server Modifikationen dieses Datenstromes durchgeführt werden können, bevor er in das Mate II Produkt über die polymorphe Schnittstelle eingespeist wird.

7. Evaluation

Im folgenden Kapitel wird auf die Evaluierung des MATE II-Projektes eingegangen. In der Einleitung wird die Aufteilung der Evaluierung in einzelne Abschnitte vorgenommen, sowie kurz das Ziel der jeweiligen Testphase erläutert. Anschließend werden in den folgenden Kapiteln der jeweilige Testaufbau, sowie die Testdurchführung und die damit gewonnen Ergebnisse präsentiert.

Durch das von der Projektgruppe ausgewählte V-Modell unterteilt sich die Evaluation des Produktes in drei Phasen. In der ersten Phase werden in den Komponententests, die Komponenten und ihre einzelnen Funktionen getestet. Ziel ist es die vor der Implementierung erstellten Komponentenspezifikationen zu verifizieren. Dies geschieht unter anderem mit der Hilfe von JUnit-Tests, der genaue Aufbau und die resultierenden Ergebnisse für die Komponenten sind dem jeweiligen Unterkapitel des Komponententest Kapitels zu entnehmen.

In der zweiten Phase, dem Integrationstest, wird das Zusammenspiel zwischen den einzelnen Komponenten getestet. Die Tests dienen dazu die internen Schnittstellen des Produktes zu verifizieren. Die semantische Korrektheit der in den Tests genutzten Daten werden von den Projektmitgliedern selbst beurteilt, der Aufbau und die Ergebnisse sind im Kapitel Integrationstest zu finden.

In der dritten und letzten Phase der Evaluation, dem Systemtest, werden die Anforderungen an das Produkt sowie das grundlegende, auf den Anforderungen basierende, Konzept verifiziert und validiert. Hierfür wurden Testszenarien konzipiert, die reale Situationen der Schifffahrt nachstellen. Mit Hilfe der Szenarien und den enthaltenen Testfällen wurde das Produkt zuerst im Labor getestet und anschließend auf dem Forschungsschiff Zuse in der realen Umgebung des Produktes. Durch den Einsatz des Monitoring-Tools Mate Control wurden die Datenströme während der Szenarien erfasst und anschließend ausgewertet. Mit Hilfe der Auswertung konnte der Ist-Zustand des Systems abgebildet werden und mit dem vor den Tests geplanten Soll-Zustand abgeglichen werden. Auf Basis dieses Vergleiches können valide Aussagen über das Produkt getroffen werden. Diese Aussagen sowie der Aufbau und weitere Ergebnisse der Labor- und Systemtests sind im Kapitel Systemtest zu finden.

Neben dem Fokus auf den Testphasen des V-Modells wurden Methoden des modellgetriebenen Testens eingesetzt. Das modellgetriebene Testen kann bereits in frühen Phasen der Entwicklung eingesetzt werden, da für das Testen nicht das gesamte System sondern nur das Modell oder Teile des Modells benötigt werden [Zan09]. Hierfür wurde auf die Methoden des Model-, Software- und Vessel In The Loop Test zurückgegriffen. Mit diesen Methoden konnten die Funktionen der Komponenten bereits früh im Labor per Simulation getestet werden bevor sie in der realen Umgebung einem Test unterzogen wurden. Auch die Integration zwischen den Komponenten sowie die Systemtests konnten so bereits vorab getestet werden.

7.1. Komponententests

Im folgenden Abschnitt werden die Komponententests der einzelnen Bestandteile des „PG MATE II “-Gesamtsystems beschrieben.

7.1.1. EPD

Das folgende Kapitel behandelt den Versuchsaufbau und die Evaluation der Komponententests der Komponente „EPD“.

Versuchsaufbau

Zur Überprüfung der komponentenspezifischen Anforderungen müssen folgende Punkte getestet werden:

- Zeigt die Komponente Roh-Daten (AIS-, Radar-, Grid-Daten, Polygone und die eigene Schiffsposition) auf der Karte an?
- Kann innerhalb der Komponente eine Route mit zwei oder mehr Wegpunkten erstellt werden?
- Kann die von der Komponente bereitgestellte Maske für Schiffsinformationen die eingegebenen Daten korrekt abspeichern?
- Kann die Komponente Eingaben von manuellen Steuerbefehlen korrekt erfassen und an andere Systeme weitersenden?
- Kann die Komponente Routen empfangen und darstellen?
- Kann die Komponente Routen und Schiffsinformationen absenden?
- Kann die Komponente einen Kamerastream empfangen und anzeigen?

Für die Anzeige der Roh-Daten wurde für jeden der Datentypen ein Testfall erstellt. Grundlegend gleichen sich die Testfälle, da die Komponente die Daten immer empfangen, verarbeiten und auf der Karte darstellen muss. Allerdings unterscheidet sich je nach Datentyp die eingesetzte Datenquelle:

- Für AIS- und Radar-Daten wurden die Daten per JSON von der DOI-Komponente gesendet.
- Für Grid-Daten und Polygone wurden die Daten per JSON von der Routen- bzw. Pfad-Planung gesendet.
- Für die eigene Schiffsposition wurden die Daten per NMEA0183 von der Polymorphen Schnittstelle gesendet.

Die Tests konnten positiv bewertet werden, wenn die jeweiligen Daten auf der Karte visualisiert wurden.
Für die Erstellung einer Route mit zwei oder mehr Wegpunkten, wurde ein Testfall erstellt, der den Tester, anweist eine Route mit Hilfe des Routen-Werkzeugs der EPD zu erstellen. Der Test galt als positiv, wenn die Route mit ihren Wegpunkten im Route-Manager abgespeichert und angezeigt wird.
Das Testen der Schiffsinformations-Maske galt als positiv, wenn die in der Maske eingegebenen Daten des Testers abgespeichert wurden und nach einem Neustart der Komponente weiterhin in der Maske angezeigt wurden. Um die Maske zu öffnen muss der Tester die Settings-Seite der EPD öffnen und den PGMate Plugin-Tab auswählen.
Für die manuellen Steuerbefehle wurde das von der PG MATE I entwickelte Virtual-Handles Plugin erneut getestet. Dieser Test war durch den Wechsel der EPD und die damit einhergehende Portierung des Plugins notwendig geworden. Auf Basis der Testfälle der PG MATE I führte der Tester einige Eingaben von Steuerbefehlen durch. Diese wurde anschließend mit den im System ausgegebenen Werten verglichen. Die Tests galten als positiv, wenn sich die Werte von System und Eingabe nicht unterschieden.
Für das Testen der zu empfangenden Routen wurden Routen per RTZ von der Routen-Planung an die EPD gesendet. Der Test galt als positiv, wenn die Route im Route-Manager abgespeichert wurde und sie nach der Selektion im Route-Manager auf der Karte angezeigt wurde.
Zum Testen des Absendens von Routen gemeinsam mit Schiffsinformationen wurde die Routen-Planung als Empfänger benötigt. Die Routen wurden per RTZ, die Schiffsinformationen per JSON versendet. Der galt als positiv, wenn die Route und die Informationen von der Routen-Planung empfangen wurden.

Der Test des Kamerastreams wurde mit Hilfe der Objektdetektion und dem Rosi-Videoplugin getestet. Die Objektdetektion sendete einen Stream an die EPD, diese musste den Stream empfangen und anzeigen können. Konnte der Stream nicht oder nur fehlerhaft dargestellt werden, galt der Test als negativ.

Evaluation

Die oben genannten Testfälle sind alle positiv verlaufen. Die Komponente kann somit die Anforderungen zu Roh-Daten (F-EPD-1 bis F-EPD-1.5) erfüllen. Des Weiteren kann die Komponente Routen erstellen (F-EPD-2) sowie Routen empfangen und darstellen (Grundvorrausetzung für F-EPD-5) und sie mit den Schiffsinformationen gemeinsam versenden (Vorrausetzung für F-EPD-4). Die Schiffsinformations-Maske ist funktionstüchtig (F-EPD-3). Die Eingabe und das mögliche Senden von manuellen Steuerbefehlen (F-EPD-7) und das Empfangen und Anzeigen des Kamerastreams (F-EPD-6) sind ebenfalls möglich.

7.1.2. Routen-Planung

Dieses Unterkapitel befasst sich mit dem Versuchsaufbau und Evaluation des Komponententests der Routen-Planungs-Komponente.

Versuchsaufbau

Um zu überprüfen, ob die Komponente die komponentenspezifischen Anforderungen erfüllt, muss Folgendes getestet werden:

- Ist die Komponente in der Lage über eine interne Repräsentation, das Gebiet für jeweils zwei aufeinanderfolgende Wegpunkte, inklusive der darin enthaltenen Hindernisse, darzustellen? (F-CA-RP-3)
- Verfügt die Komponente über einen Algorithmus, welcher in der Lage ist eine Route zwischen jeweils zwei aufeinanderfolgenden Wegpunkten zu berechnen, die durch keines der Hindernisse hindurch führt? (F-CA-RP-4)
- Ist die berechnete valide Route relativ effizient? (NF-CA-RP-1)
- Weist die Route einen ausreichenden Mindestabstand zu den Hindernissen auf? (NF-CA-RP-2)

Hierfür wurden verschiedene Testarten durchgeführt, die im Folgenden zusammen mit den Ergebnissen näher beschrieben werden.
Um zu überprüfen ob die Komponente das Gebiet zwischen zwei gegebenen Wegpunkte korrekt abbildet, wurde eine neue Debug-Schnittstelle zur EPD geschrieben. Mithilfe dieser kann auf der EPD das Grid visualisiert werden. Hier zeigte sich, dass die in der digitalen Seekarte hinterlegten statischen Hindernisse zu einer Blockierung der darüber abgebildeten Zelles des Grids führt. Eine Verkleinerung der Zellen-Größe führte zudem zu einer Verfeinerung der Hindernis-Nachbildung im Grid.

Abbildung 7.1.: Debug-Schnittstelle

Innerhalb der EPD wurde auch die Pfad-Berechnung mittels A*-Algorithmus, unter Einsatz verschiedensten Hindernissen, visualisiert. Dabei führte keiner der Pfade durch eine blockierte Zelle und somit durch kein Hindernis.
Da ein A*-Algorithmus verwendet wurde, dessen Heuristik-Funktion optimistisch ausgelegt und somit immer kleiner oder gleich den realen Kosten ist, wird immer garantiert der kürzeste und somit ein relativ effizienter Pfad gefunden. [HEUb]
Im Rahmen der Simulationen war zudem zu beobachten, dass immer ein ausreichender Abstand zu den Hindernissen eingehalten wurde. Dies lag vor allem daran, dass eine Zelle bereits als blockiert gilt, wenn ein Hindernis sie auch nur an kleinster Ecke schneidet.

7.1.3. Controller

Im folgenden Kapitel werden der Versuchsaufbau und die Evaluation der Komponententests des Controllers beschrieben.

Versuchsaufbau

Um zu prüfen ob die Regelungskomponente alle Komponentenanforderungen erfüllt, ist es notwendig die Komponente insgesamt zu testen. Zunächst scheint es jedoch schwer eine solche Komponente, die direkt mit ihrem physischen Umfeld kommuniziert ohne eben dieses Umfeld zu testen. Auf der - nach den Modultests - weiter andauernden Fehlersuche wurden die Komponententests in drei aufeinander aufbauende Phasen eingeteilt:

- **(Phase I)** Test mit realen, aufgezeichneten Daten. Hierzu wurden Daten von der Zuse während der Fahrt aufgezeichnet. Diese wurden in unseren Tests abgespielt und als Schiffsstatus in die Regelungskomponente eingespeist.
- **(Phase II)** Test mithilfe einer 3D-Simulationsumgebung (Brine). Hierzu wurde eine Schnittstelle entworfen, durch welche die Regelungskomponente mit der Simulation kommunizieren kann.
- **(Phase III)** Test im physikalischen Testfeld (LABSKAUS) mit der Zuse.

Bei allen Tests wurden konstante Wunsch-Wegpunkte, Routenstati und Wunschgeschwindigkeiten verwendet um sich von der Integration mit weiteren Komponenten zu distanzieren. Es ist hier jedoch nicht möglich Komponententests ohne eine Integration der Schiffsstatusdaten in den Testprozess durchzuführen. Die Regelungskomponente kann ihren Anforderungen ohne diese Daten offensichtlich nicht nachkommen.

	Erfolgreich	Nicht Erfolgreich
Phase I	Interne Verarbeitung des Schiffszustands korrekt	Schiffszustand wird nicht richtig erkannt / Fehler in Dateninterpretation
Phase I + II	Grundsätzliche Funktionalität zur Regelung des Schiffes vorhanden	Falsche Berechnung der Steuerbefehle
Phase I + II + III	Regelungskomponente funktionsfähig	Parameterfindung für die Zuse nicht möglich

Tabelle 7.1.: Testphasen und Bedeutung ihrer Resultate

Die Einteilung der Komponententests in die drei Phasen lässt sich folgendermaßen begründen: Mit den *Phase-I-Tests* können Fehler identifiziert werden, die in der Datenübertragung und Verarbeitung innerhalb der Regelungskomponente entstehen. Wohingegen durch die *Phase-II-Tests* Fehler im gesamten Regelungsverhalten identifiziert werden können. Erfolgreiche *Phase-III-Tests* stellen sicher, dass es auch in der realen Welt möglich ist Parameter zu finden, die eine zufriedenstellende Regelung ermöglichen. Tabelle 7.1 liefert eine Übersicht über die Testphasen und ihre Bedeutung.

Evaluation

Schon bei den Phase-I-Tests stellte sich heraus, dass irgendwo ein Fehler vorhanden war. Hier konnte das Endergebnis von MATE I reproduziert werden. Selbst bei sich verändernder Position und Ausrichtung regelte die Regelungskomponente meist ausschließlich das Ruder mit vollem Einschlag in eine Richtung. Nach einigen Nachforschungen stellte sich heraus, dass der Controller seine Berechnungen basierend auf der NMEA2000-Nachricht mit PGN 129026 (Course Angle) durchführte. Jedoch wurde diese Nachricht nie übermittelt, wodurch die Regelungskomponente immer mit einem Kurs von 0 Grad rechnete. Dieser Fehler wurde behoben, indem das Heading des Schiffes als Referenz für die Berechnungen genutzt wurde. Nach der Beseitigung dieses Fehlers waren die Tests erfolgreich, somit wurde die Anforderung F-CT-1 erfüllt.

Für die Phase-II-Tests wurde die Regelungskomponente über eine eigens entwickelte Schnittstelle verbunden. Die Simulation zeige grundsätzlich gute Ergebnisse. Es wurde jedoch davon abgesehen eine genaue Parameterschätzung für die Simulation durchzuführen, da das Dynamikverhalten des simulierten Schiffes und das der Zuse deutlich voneinander abwichen.

Aufgrund von diversen technischen Problemen und undokumentierten Konfigurationen zögerte sich der Start der Phase-III-Tests leider unerwartet lange hinaus. Nachdem alles eingerichtet war, stellte sich zunächst heraus, dass die Regelungskomponente viel zu häufig Steuerbefehle sendete. Es wurden alle 100 ms Steuerbefehle geschickt, die Hardware auf der Zuse benötigte aber deutlich länger um das Ruder auf eine bestimmte Stellung einzustellen. Die Regelungszeit wurde also auf zwei Sekunden verlängert. Diese Zeit erzielte in mehreren Testläufen die besten Ergebnisse. Nach der Parameterschätzung konnte festgestellt werden, dass der Unterregler zur Einstellung des richtigen Headings problemlos funktionierte. Der PID-Unterregler zur Minimierung des XTE allerdings regelte zwar zunächst in die richtige Richtung, schwang dann aber über und schaffte es nicht rechtzeitig zurückzuregeln. Auch verschiedenste Parameterkonfigurationen konnten dieses Problem nicht lösen. Daher wurde dieser Unterregler durch eine parametrisierte Funktion ersetzt, die je nach Schiffssituation ein Heading-Setpoint berechnet. Nach dieser Ersetzung funktionierte die Ruderregelung ohne weitere Probleme und erfüllt somit die Anforderung F-CT-5.
Da der Geschwindigkeitsregler nur ein normaler PID-Regler ist, schien hier nur eine Parameterschätzung von Nöten zu sein. Es stellte sich allerdings heraus, dass die Drehzahl des Motors der Zuse zwar sehr schnell heruntergesetzt werden konnte, dafür aber nur sehr langsam erhöht werden konnte. Das Schiff beschleunigte also langsam bis zur gewünschten Geschwindigkeit, bremste dann aber stark ab, als die Geschwindigkeit überschritten wurde, und der PID-Regler darauf reagierte. Dieses Problem wurde durch eine Fallunterscheidung gelöst: Wenn das Schiff zu schnell ist, wird die Differenz mit einem Faktor < 1 multipliziert. Dies führt dazu, dass die Antwort des Reglers viel kleiner ausfällt, als zuvor. Mit dieser Änderung ist die Komponente fähig die Anforderung F-CT-4 zu erfüllen.

7.1.4. Objektdetektion

Im Folgenden wird der Versuchsaufbau und Evaluation des Komponententests der Komponente Objektdetektion beschrieben.

Versuchsaufbau

Um die Erfüllung der Komponentenanforderungen der Objektdetektion zu prüfen, muss getestet werden, ob die Komponente beim Verarbeiten des Bildmaterials Schiffe die vor dem eigenen Schiff liegen erkennt, um diese anschließend an die Pfadplannungskomponente zu senden.
Die Komponente Objektdetektion analysiert den mit einer beliebigen Kamera aufgenommenen Videostream in Echtzeit. Durch das neuronale Netz erfolgt eine binäre Klassifikation der auf dem

Bild befindlichen Objekte. Hierbei werden nur die Objekte, welche als Schiffe klassifiziert werden betrachtet. Diese binäre Klassifikation kann in Bezug auf die Realität (hier: bekannt) wahr und falsch sein. Diese Ergebnisse lassen sich in eine Konfusionsmatrix (Wahrheitsmatrix) abbilden (siehe Tabelle 7.2). Die vier Felder (mögliche Kombinationen aus Klassifikationsergebnis und

	Schiff	kein Schiff
Schiff erkannt	true positiv	false positiv
kein Schiff erkannt	false negativ	true negativ

Tabelle 7.2.: Konfusionsmatrix

Realität) sind „true positive“ (TP), „true negative“ (TN), „false positive“ (FP) und „false negative“ (FN). [Loh11]

Um ein aussagekräftiges Ergebnis über die Erfüllung der Anforderungen zu erzielen wurden insgesamt 400 Bilder im physikalischen Testfeld aufgenommen und anschließend analysiert. Hierbei wurden 200 Bilder mit der Logitech C920 und 200 Bilder mit der Blackmagic aufgenommen. Die Aufnahmen beinhalten das Kamerabild mit markierten Schiffen (rechteckig umrandet). Diese Aufnahmen wurden manuell durchgegangen, wobei die Anzahl der nicht erkannten Schiffe sowie der erkannten Schiffe gezählt und die Klassifikation des Algorithmus (richtig/falsch) beurteilt wurde. Ergebnisse dieser Analysen sind in der Tabelle 7.3 und Tabelle 7.4 zu sehen.

	Schiff	kein Schiff
Schiff erkannt	75	9
kein Schiff erkannt	112	-

Tabelle 7.3.: Ergebnisse mit der Logitech C920 HD Pro

	Schiff	kein Schiff
Schiff erkannt	217	11
kein Schiff erkannt	126	-

Tabelle 7.4.: Ergebnisse mit der Blackmagic Micro Studio Camera 4K

Es wurde hier deutlich, dass die Anzahl der unscharf zu sehenden bzw. nicht komplett zu sehenden Schiffe auf den Bildern der Logitech C920 deutlich höher sind (Logitech C920: 222 von insgesamt 248 [89.52%], Blackmagic: 107 von insgesamt 343 [31.20%]).
Mit den gemessenen Zahlen lassen sich verschiedene Maße berechnen. Wichtige Kennzahlen

sind die Trefferquote und die Genauigkeit. [Loh11]
Die Trefferquote ist die Richtig-Positiv-Rate, d.h die Relation von richtig positiven (TP) Entscheidungen zur Gesamtzahl der wirklichen positiven Beobachtungen (TP und FN) [Loh11]. In diesem Fall gibt die Trefferquote also den Anteil an Schiffen, der erkannt wurde, von der Gesamtzahl an Schiffen, welche sich auf dem Bildmaterial befinden, an.
Die Genauigkeit, auch Relevanz, Wirksamkeit oder positiver Vorhersagewert, ist das Verhältnis von den richtig-positiven Entscheidungen (TP) zu der Anzahl aller positiver Vorhersagen (TP + NP) [Loh11]. Sie beschreibt somit die Anzahl der richtig markierten Schiffe im Verhältnis zur Gesamtanzahl der als Schiff markierten Objekte.

Evaluation

Die Berechnung der Gütemaße (siehe Tabelle 7.5) zeigt deutliche Unterschiede in Bezug zu den verwendeten Kameras. Dieser signifikante Unterschied von ca. 20% in der Trefferquote ist durch die Kameraauflösung zu erklären. Auch in der Genauigkeit der Klassifizierung ist die Logitech C920 im Vergleich um ca. 6% schlechter. Hierbei wird allerdings deutlich, dass die Genauigkeit bei beiden verwendeten Kameras sehr gut ist und im Durchschnitt bei 92,24% liegt.
Die Trefferquote liegt bei der Kamera mit der höheren Auflösung bei 63,27%, welches bereits ein gutes Ergebnis darstellt und wodurch die Anforderung an die Komponente erfüllt werden. Um diesen Wert weiter zu erhöhen, kann die Hardware weiter verbessert werden (31,20% der Schiffe waren beim Test unscharf). Eine weitere Möglichkeit ist die Verwendung eines anderen Models welches eine höhere Genauigkeit aufweist. Dies würde aber negative Auswirkungen auf die Verarbeitungsdauer haben, wodurch dies nicht zu empfehlen ist.

Maßzahl	Berechnung	**1. Test**	**2. Test**
Trefferquote	TP / (TP + FN)	0.4011	0.6327
Genauigkeit/Relevanz	TP / (TP + FP)	0.8929	0.9518

Tabelle 7.5.: Gütemaße der Klassifikation

7.1.5. Dynamische Objektidentifizierung

Dieses Unterkapitel befasst sich mit dem Versuchsaufbau und Evaluation des Modul- sowie des Komponententests der Komponente Dynamische Objektidentifizierung.

Versuchsaufbau

Um zu überprüfen, ob die Komponente die komponentenspezifischen Anforderungen erfüllt, muss Folgendes getestet werden:

- Identifiziert die Komponente die Schiffe, welche sich in einem bestimmten Umkreis um das eigene Schiff befinden?
- Erkennt die Komponente die Größe, Position, Geschwindigkeit und den Kurs von Objekten?
- Erkennt die Komponente, dass verschiedene AIS Nachrichten vom selben Schiff sind?
- Kann die Komponente verschiedene AIS- und Radar-Nachrichten eines Schiffes diesem zuordnen?
- Kann die Komponente die Parameter Position, Größe, Geschwindigkeit und Kurs von fusionierten Schiffen so exakt wie möglich schätzen?

Bevor die Komponententests durchgeführt wurden, wurde sichergestellt, dass die einzelnen Funktionen der Komponente funktionieren. Hierfür wurden verschiedene Modultests durchgeführt. Diese werden im Folgenden zusammen mit den Ergebnissen näher beschrieben werden.

Um sicherzustellen, dass nur Schiffe in einem bestimmten Umkreis des eigenen Schiffs identifiziert werden, gibt es eine Klasse, die sich speziell um die Schiffsumgebung kümmert (ShipEnvironmentHandler). Diese kennt die eigene Position des Schiffs, berechnet die Umgebung, in der Schiffe identifiziert werden sollen, und überprüft ob Schiffe, dessen Nachrichten in der Komponente ankommen, in dieser Umgebung liegen oder nicht. Um zu testen, ob diese Umgebungsberechnung korrekt funktioniert wurde ein JUnit-Test erstellt. Dieser erzeugt zunächst zu einer beliebigen Position mittels Schiffsumgebungs-Klasse den Umkreis und prüft anschließend, ob verschiedene gewählte Positionen, die innerhalb bzw. außerhalb des Umkreises liegen, auch als innerhalb oder außerhalb liegend betrachtet werden. Es wurden sieben verschiedene Positionen getestet: die eigene Position (also der Mittelpunkt des Umkreises), drei verschiedene Positionen die direkt auf dem Umkreis liegen und somit noch zum Umkreis gehören, eine willkürliche Position innerhalb des Umkreises und zwei willkürliche Positionen außerhalb des Umkreises. Das Ergebnis des Tests ist positiv verlaufen.
Die Größe, Position, Geschwindigkeit und Kurs eines Objektes wird in der Klasse ReadInformation erkannt. Hier werden die ankommenden Nachrichten der Objekte ausgelesen und die relevanten Werte abgespeichert. Um zu testen, ob die Komponente die korrekten Werte erkennt und abspeichert, wurden zuerst drei verschiedene NMEA2000-Nachrichten (Radar, AIS, statische AIS) erstellt und als XML abgespeichert. Der ReadInformationTest kann diese Nachrichten nach

dem selben Prinzip wie der InputListener des UDPAdapters einlesen. Nachdem die Klasse ReadInformation die Informationen verarbeitet und abgespeichert hat, prüft der Test, ob im Speicher die korrekten Daten abgespeichert sind. Dieser Test ist positiv verlaufen.
Die Tests zur Assoziation beinhalten die Überprüfung, ob zwei mittels AIS erkannten Schiffe als das selbe Schiff erkannt werden, sowie ob ein Radar und ein AIS Schiff als eins erkannt wird, wenn dies jeweils der Fall ist. Hierfür wurden vom Simulationsadapter Radar und AIS Nachrichten in die Komponente eingespeist, diese verarbeitet und abgespeichert, und im Anschluss assoziiert. Zur Überprüfung wurden zum einen Konsolenausgaben ausgewertet, und zum anderen auf die EPD zurückgegriffen. Die Ergebnisse wurden an die EPD gesendet, die die Schiffe visualisiert hat. Hierdurch war sehr deutlich zu sehen, dass zum einen nur ein Schiff mit einer ID zu sehen war und zum anderen ein assoziiertes Schiff auf dem Radar und dem AIS Schiff lag. Das bedeutet, dass die Tests positiv verlaufen sind.
Da für die möglichst exakte Schätzung der relevanten Schiffsinformationen von fusionierten Schiffen die Regression der Datenfusion zuständig ist, wurde zu dessen Überprüfung der JUnit-Test RegressionsTest implementiert. Hierfür wurden weitere NMEA2000-Nachrichten der eigenen Position, des Radars und des AIS erstellt. Hierbei besitzen alle Radar bzw. alle AIS Nachrichten die selbe Schiffs ID, das heißt alle Radarnachrichten gehören zu einem Schiff und alle AIS-Nachrichten gehören zu einem Schiff. Des Weiteren wurden die verschiedenen Werte so gewählt, dass die Schiffe fusioniert werden können, um so die Informationen eines fusionierten Schiffs testen zu können. Es können drei verschiedene Fälle bei der Ermittlung der Schiffsinformationen auftreten: Es existiert zu dem Schiff nur AIS Daten, es existiert zu dem Schiff nur Radar Daten, es existieren zu dem Schiff Radar und AIS Daten, das heißt es wurde fusioniert. Der Test deckt durch drei verschiedene Testfälle alle drei Fälle ab. Der Ablauf dieser Testfälle ist jeweils sehr ähnlich. Zuerst werden die NMEA2000-Nachrichten der entsprechenden Daten eingelesen und durch die Klasse ReadInformation verarbeitet und abgespeichert. Im Falle der Fusionierung werden im Anschluss die Schiffe assoziiert. Hiernach startet in allen drei Fällen die Regression und der Testfall speichert sich das Ergebnis dieser ab. Zum Schluss werden die Daten des Regressionsergebnis mit den korrekten Ergebnissen verglichen. Hierbei sind alle Tests positiv verlaufen.

Nachdem in den Modultests die Korrektheit der einzelnen Module getestet wurden, wurden Komponententests zur Überprüfung der gesamten Komponente durchgeführt. Um die Erfüllung der Komponentenanforderungen zu prüfen, müssen folgende Tests positiv verlaufen:

1. Sende AIS- und Radarnachrichten des selben Schiffes mit exakt den selben Werten für Position, Geschwindigkeit und Kurs an die DOI und überprüfe, ob erkannt wird, dass es sich um das selbe Schiff handelt.

2. Sende AIS- und Radarnachrichten des selben Schiffes mit einem Unterschied von 69 Metern in den einzelnen befahrenen Positionen an die DOI und überprüfe, ob erkannt wird, dass es sich um das selbe Schiff handelt.
3. Sende AIS- und Radarnachrichten von unterschiedlichen Schiffen, die in den einzelnen Positionen jeweils 71 Meter unterschied haben, an die DOI und überprüfe, ob erkannt wird, dass es sich nicht um das selbe Schiff handelt.
4. Sende Radar-Nachrichten von einem Schiff und AIS-Nachrichten von einem anderen Schiff, die die selben Positionen abfahren, aber zu unterschiedlichen Zeitpunkten, an die DOI und überprüfe, ob erkannt wird, dass die Nachrichten nicht zu dem selben Schiff gehören.
5. Sende Radar-Nachrichten von einem Schiff und AIS-Nachrichten von zwei Schiffen (mit zwei unterschiedlichen IDs) an die DOI und überprüfe, ob erkannt wird, dass die ID der Radar-Nachrichten und eine ID der zwei IDs der AIS-Nachrichten zu dem selben Schiff gehören.
6. Sende AIS-Nachrichten von einem Schiff und Radarnachrichten von zwei Schiffen (mit zwei unterschiedlichen IDs) an die DOI und überprüfe, ob erkannt wird, dass die ID der AIS-Nachrichten und eine ID der zwei IDs der Radarnachrichten zu dem selben Schiff gehören.

Alle Tests wurden nach dem selben Schema durchgeführt. Hierfür sind folgende Schritte notwendig:

- Eingabe der entsprechenden Schiffsdaten in HAGGIS.
- Starte Polymorphic-Interface, Hub und Dynamic Object Identification.
- Verschicke Radar- und AIS-Nachrichten mithilfe von HAGGIS.
- Überprüfe die Ausgabe der Assoziationsliste.

Als Ergebnis wird folgendes erwartet: Beim ersten und zweiten Test werden die Schiffe assoziiert, beim dritten und vierten Test werden die Schiffe nicht assoziiert und beim fünften und sechsten Schiff werden jeweils ein Schiff assoziiert und das andere nicht.

Evaluation

Es sind alle oben genannten Testfälle des Modultests positiv verlaufen. Das bedeutet, dass die Komponente an den entsprechenden Stellen jeweils die korrekten Daten abgespeichert, assoziiert oder berechnet hat.

Auch die erwarteten Ergebnisse des Komponententests sind alle eingetreten.
Hierdurch ist abgesichert, dass die funktionalen Anforderungen F-DOI-4 bis F-DOI-8 erfüllt sind.

7.1.6. Simulationsadapter

Im Folgenden wird der Versuchsaufbau und Evaluation des Komponententests der Komponente „Simulationsadapter“ beschrieben.

Versuchsaufbau

Um die Erfüllung der Anforderungen an die Komponente Simulationsadapter zu prüfen, muss getestet werden, ob der Simulationsadapter die von HAGGIS versandten XML Dateien empfangen, verarbeiten und weiter senden kann.
Der Simulationsadapter empfängt über UDP XML Dateien mit Informationen zu den in HAGGIS angelegten Szenarien, diese müssen vor der weiteren Verwendung deserialisiert werden. Diese werden mittels einer XSL Transformation zu static und dynamic AIS-, Radar- und Positionsnachrichten verarbeitet. Im Anschluss an die Verarbeitung werden die erstellten Nachrichten serialisiert und via RabbitMQ versendet.
Für jeden dieser Schritte wurden JUnit-Tests erstellt. Eine genaue Aufschlüsselung der Tests ist im Anhang unter Abschnitt B.1 ersichtlich.

Evaluation

Alle erstellten JUnit-Tests konnten erfolgreich durchlaufen werden. Hierdurch kann eine korrekte Funktionsweise des Simulationsadapters für die Funktionen des Daten empfangen, verarbeiten und weiter versenden bestätigt werden. Durch diese Testabdeckung ist abgesichert, dass die funktionalen Anforderungen F-SA-2 bis F-SA-6 erfüllt sind.

7.1.7. Kollisionsverhütung

Dieses Unterkapitel befasst sich mit der Methodik und Evaluation des Komponententests der Komponente „Kollisionsverhütung “.

Methodik

Um zu überprüfen, ob die Komponente die komponentenspezifischen Anforderungen erfüllt, muss Folgendes getestet werden:

- Ist die Komponente in der Lage über eine interne Repräsentation, das Gebiet um die eigene Position herum bis zum jeweils nächsten Wegpunkt, inklusive der darin enthaltenen Hindernisse, darzustellen? (F-CA-RP-5)
- Verfügt die Komponente über einen Algorithmus, welcher in der Lage ist einen Pfad zwischen jeweils zwei aufeinanderfolgende Wegpunkten zu berechnen, die durch keines der Hindernisse hindurch führt? (F-CA-RP-6)
- Ist der berechnete valide Pfad relativ effizient? (NF-CA-PP-1)
- Weist die Route einen ausreichenden Mindestabstand zu den Hindernissen auf? (NF-CA-PP-2)

Hierfür wurden verschiedene Testarten durchgeführt, die im Folgenden zusammen mit den Ergebnissen näher beschrieben werden.
Um zu überprüfen ob die Komponente das Gebiet um die eigene Position herum korrekt abbildet, wurde die bereits im Rahmen der Evaluation des Küstenleitstand erstellte Debug-Schnittstelle benutzt. Hier zeigte sich, dass die statischen Hindernisse, erkennen in der Seekarte, zu eine Blockierung der darüber abgebildeten Zelles des Grids führt. Eine Verkleinerung der Zellen-Größe führte zudem zu einer Verfeinerung der Hindernis-Nachbildung im Grid.
Für die Kontrolle der Abbildung, der dynamischen Hindernisse (andere Schiffe), wurde eigens eine neue Simulationsumgebung geschaffen. Diese ist in der Lage ein Grid grafisch darzustellen, sowie übergeben Schiffe und die dementsprechend blockierten Zellen.

Es zeigte sich, dass die Schiffe bzw. deren über sie gelegten Goodwinshapes korrekt ausgerichtet waren, die korrekte Größe besaßen und die umliegenden Zellen korrekt blockierten. Das ausführliche Testen der Voraus-Projektierung der anderen Schiffspositionen mittels CPA war nicht nötig, da diese Funktion aus MTCAS kommt und dort schon ausreichend evaluiert wurde.
Innerhalb der Simulationsumgebung wurde auch die Pfad-Berechnung mittels A*-Algorithmus, unter Einsatz verschiedensten Hindernissen, erprobt. Dabei führte keiner der Pfade durch eine blockierte Zelle und somit durch kein Hindernis.
Da ein A*-Algorithmus verwendet wurde, dessen Heuristik-Funktion optimistisch ausgelegt und somit immer kleiner oder gleich den realen Kosten ist, wird immer garantiert der Kürzeste und somit ein relativ effizienter Pfad gefunden. [HEUb]

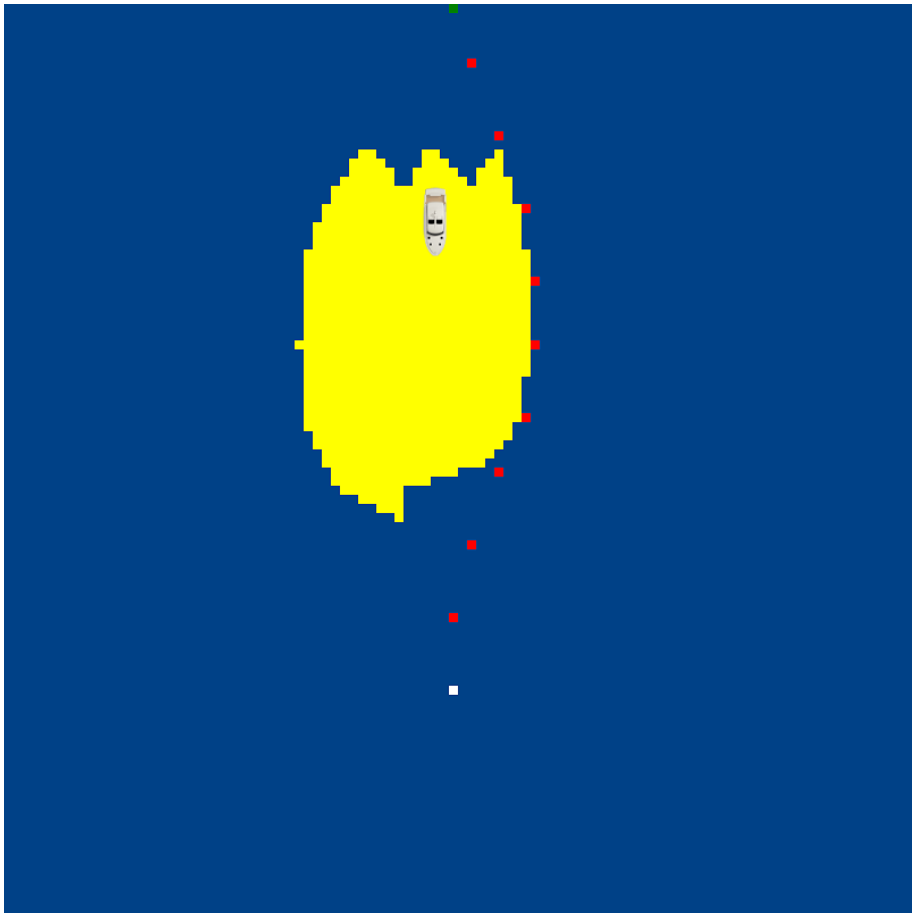


Abbildung 7.2.: Simulator

Im Rahmen der Simulationen war zudem zu beobachten, dass immer ein ausreichender Abstand zu den Hindernissen eingehalten wurde. Dies lag zu Einem daran, dass eine Zelle bereits als blockiert gilt, wenn ein Hindernis sie auch nur an kleinerster Ecke schneidet.

7.2. Integrationstests

Im folgenden Abschnitt werden die Integrationstests der einzelnen Komponenten des Gesamtsystems von PG MATE II beschrieben. Zudem werden die jeweiligen Ergebnisse erläutert.

7.2.1. Controller

In diesem Kapitel werden die Integrationstests des Controllers vorgestellt.

Versuchsaufbau

Durch die Komponententests kann bei Erfolg sichergestellt werden, dass die Regelungskomponente dazu in der Lage ist zwei konstante Wegpunkte anzufahren sowie manuelle Steuerbefehle anzunehmen und an die Hardware weiterzugeben. Die Integration von Schiffsstatusdaten wurde aus dort diskutieren Gründen bereits in den Modultests betrachtet. Die Wegpunkte, die gewünschte Geschwindigkeit und den Routenstatus erhält die Regelungskomponente von der Pfadplanung. Manuelle Steuerbefehle werden von der EPD gesendet. Die Integrationstests lassen sich hier in drei Schritte einteilen:

- **Syntaktische Korrektheit** der Daten verifizieren. Hier wird mithilfe der MATLAB/Simulink Analysetools geprüft ob die Syntax des NMEA2000 Protokoll eingehalten wird.
- **Semantische Korrektheit** der Daten verifizieren. Hier wird mithilfe der MATLAB/Simulink Analysetools geprüft ob zwischen den Komponenten sinnvolle Werte übermittelt werden (Koordinaten richtig, keine negative Drehzahlanfrage, etc.).
- **Latenzmessung, Kontinuierliches Verhalten überprüfen**. Die Latenzen vom Senden einer Nachricht bis zur Umsetzung in Form von Steuerbefehlen werden geprüft. Wie verhält sich die Regelungskomponente, wenn über die Zeit neue Wegpunkte gesetzt werden, bzw. sich die Wegpunkte mit hoher Frequenz ändern?

Evaluation

Aufbauend auf den bereits von MATE I entworfenen und implementierten Kommunikationsschnittstellen konnte die syntaktische Korrektheit der gesendeten Daten schnell verifiziert werden. Hier mussten keine weiteren Änderungen vorgenommen werden.

Auch bei der semantischen Korrektheit mussten nur kleine Details geändert werden. So gab es beim Routenstatus (Pfadplanung aktiv oder inaktiv) einige Unklarheiten, die jedoch schnell behoben werden konnten. Somit gelten die Anforderungen F-CT-2, F-CT-4 und F-CT-6 als erfüllt.

Im letzten Schritt wurde zunächst die Latenz gemessen, die aber deutlich unter der Regelungszeit von zwei Sekunden liegt und daher vernachlässigt werden kann (erfüllt die Anforderung F-CT-07). In den Tests, in denen die Regelungskomponente einem von der Pfad-Planung berechneten Pfad folgen sollte, kam es zu erheblichen Abweichungen vom erwarteten Soll-Pfad. Die Ursache hierfür war die Tatsache, dass die Pfad-Planung in einem bestimmten Zeitintervall neue anzufahrende Wegpunkte an die Regelungskomponente geschickt hatte. Diese konnte jedoch in der Zeit zwischen zwei Pfadberechnungen das Schiff nicht ausreichend schnell auf die alten gewünschten Wegpunkte bewegen bevor neue berechnet wurden. Weiterhin basiert die Pfad-Planung ihre Berechnungen u.a. der aktuellen Position des Schiffes und berechnet einen Pfad von dort zum nächsten Routenwegpunkt. Da nun die Regelungskomponente das Schiff nun nicht schnell genug manövrieren konnte, wurde folglich jedes mal ein neuer Pfad von der aktuellen Position berechnet. Dies interpretierte die Regelungskomponente nun so, dass sie sich bereits auf dem richtigen Pfad befand und regelte nur wenig in die eigentlich gewünschte Richtung. So wich der gefahrene Pfad also immer weiter vom Soll-Pfad ab. Dieses Problem konnte nach einigen Tests dadurch behoben werden, dass erst neue Wegpunkte an die Regelungskomponente geschickt werden, wenn die letzten gesendeten Wegpunkte erreicht wurden und der Abstand der Wegpunkte erhöht wurde, indem jeweils der 5. und 9. Wegpunkt des berechneten Pfades als Ziel gewählt wurden.

7.2.2. Hindernisidentifikation

Durch die Integrationstests wird sichergestellt, dass die Dynamische Objektidentifizierung dazu in der Lage ist, alle nötigen Nachrichten vom Hub und der Objektdetektions-Komponente zu empfangen und nach der Datenfusion alle relevanten Schiffsinformationen an die EPD und die Pfad-Planung weiterzuleiten.

Hub-Entgegennahme Zum Evaluieren der Anforderungen F-DOI-1 und F-DOI-2, dass die Komponente AIS und Radar Sensordaten empfangen muss, müssen folgende Schritte vollzogen werden:

- Starte die RabbitMQ Verbindung zum Wilkinson Server
- Starte das Polymorphic Interface
- Starte den Hub (Polymorphic Interface Instance)
- Starte die Dynamic Object Identification

Als Ergebnis sollte der Logger der Dynamischen Objektidentifizierung für jede empfangene und benötigte Nachricht eine Ausgabe der Form *PGN: xxx* mit der jeweiligen PGN der NMEA2000 Nachricht beinhalten. Da die PGNs die von Radar, AIS (statisch und dynamisch), eigene Position und eigene Kurs sind, wurden im Rahmen des Integrationstests die Sensordaten erfolgreich empfangen.

Schiffsparameter Versand EPD Zum Testen wird eine Anzeige der Schiffe in der EPD benötigt. Um die Funktion dieser Anzeige zu testen, müssen folgende Schritte vollzogen werden:

- Starte die RabbitMQ Verbindung zum Wilkinson Server
- Starte das Polymorphic Interface
- Starte den Hub (Polymorphic Interface Instance)
- Starte die Dynamic Object Identification
- Starte die EPD

Als Ergebnis sollte in der Ausgabe der Dynamischen Objektidentifizierung deutlich werden, dass die Schiffsinformationen versendet wurden, sowie diese Schiffe in der EPD visuell angezeigt werden. Im Rahmen des Integrationstests wurden die Schiffsparameter erfolgreich zur EPD versendet.

Schiffsparameter Versand Pfad-Planung Zum Evaluieren der Anforderung F-DOI-9 (*Die Komponente muss alle in einem vorgegebenen Umkreis erkannte dynamische Objekte mit den Parametern Position, Geschwindigkeit, Kurs und Größe an die Pfad-Planungs-Komponente senden.*) müssen folgende Schritte vollzogen werden:

- Starte die RabbitMQ Verbindung zum Wilkinson Server
- Starte das Polymorphic Interface
- Starte den Hub (Polymorphic Interface Instance)
- Starte die Dynamic Object Identification
- Starte das Path Planning

Als Ergebnis sollte in der Ausgabe der Dynamische Objektidentifizierung deutlich werden, dass die Schiffsinformationen versendet wurden, sowie in der Pfad-Planungs-Komponente, dass die Nachrichten angekommen sind. Im Rahmen des Integrationstests wurden die Schiffsparameter erfolgreich zur Pfad-Planungs-Komponente versendet.

Kamerainformation Versand und Entgegennahme Zum Evaluieren der Anforderung F-OD-4 (*Das System muss die Position* [*rechts/links*] *der erkannten und relevanten* [*unter dem Horizont*] *Schiffe an die Dynamische Objektidentifizierung übertragen können.*) und dem Empfangen der Position von Seiten der Dynamischen Objektidentifizierung, müssen folgende Schritte vollzogen werden:

- Starte die Object Detection
- Wähle die Kamera aus
- Starte die Dynamic Object Identification

Als Ergebnis sollte der Logger der Objektdetektion eine Ausgabe der Form *Sending detected Ship to DOI.* beinhalten und der Logger der Dynamischen Objektidentifizierung eine Ausgabe der Form *Camerainput.* Im Rahmen des Integrationstests wurde die Position der Objektdetektion erfolgreich an die Dynamische Objektidentifizierung versendet und von dieser erfolgreich empfangen.

7.2.3. Küstenleitstand

Durch die Integrationstest wird sichergestellt, dass der Küstenleitstand dazu in der Lage ist eine vom Benutzer definierte Route entgegenzunehmen und mit Hilfe des NoGoSolvers diese so anzupassen, dass die Route, ohne das Durchkreuzen statischer Hindernisse, befahrbar ist.

Integrationstests

Im folgenden Abschnitt werden die durchgeführten Integrationstests des Küstenleitstandes beschrieben sowie die Ergebnisse vorgestellt.

NoGoSolver-Anfrage Zum Evaluieren des ersten Teiles der Anforderung F-CA-RP-1 *Die Komponente soll statische Hindernisse (Land & Untiefen) von dem NoGoSolver anfragen [...] können*, müssen folgende Schritte vollzogen werden:

- Starte die Routen-Planungs-Komponente
- Starte die glsepd-Shore
- Starte den NoGoSolver

- Konfiguriere Schiffsparameter und eine Route in der glsepd-Shore und drücke auf versenden

Als Ergebnis sollte der Logger des NoGoSolvers eine Ausgabe der Form *Now generated Json* anzeigen, was bedeutet, dass die Hindernisse erfolgreich angefragt wurden. Im Rahmen des Integrationstests wurden die Hindernisse erfolgreich angefragt.

NoGoSolver-Entgegennahme Zum Evaluieren des zweiten Teiles der Anforderung F-CA-RP-1 *Die Komponente soll statische Hindernisse (Land & Untiefen) von dem NoGoSolver [...], entgegennehmen und decodieren können*, müssen folgende Schritte vollzogen werden:

- Starte die Routen-Planungs-Komponente
- Starte die glsepd-Shore
- Starte den NoGoSolver
- Konfiguriere Schiffsparameter und eine Route in der glsepd-Shore und drücke auf versenden

Als Ergebnis sollte der Logger des NoGoSolvers eine Ausgabe der Form *New Obstacles [...]* anzeigen, was bedeutet, dass die Hindernisse erfolgreich entgegengenommen, decodiert und zwischengespeichert wurden. Im Rahmen des Integrationstests wurden die Hindernisse erfolgreich entgegengenommen, decodiert und zwischengespeichert.

Routen-Entgegennahme Zum Evaluieren des ersten Teiles der Anforderung F-CA-RP-2 *Die globale Route [...] soll von der glsepd entgegengenommen und decodiert werden*, müssen folgende Schritte vollzogen werden:

- Starte die Routen-Planungs-Komponente
- Starte die glsepd-Shore
- Konfiguriere eine Route in der glsepd-Shore und drücke auf versenden

Als Ergebnis sollte der Logger eine Ausgabe der Form *New active route: die übergebenen Route in RTZ-Form* anzeigen, was bedeutet, dass die Route erfolgreich entgegengenommen, decodiert und zwischengespeichert wurde. Im Rahmen des Integrationstests wurde die Route erfolgreich entgegengenommen, decodiert und zwischengespeichert. Durch den erfolgreichen Empfang konnte auch der erfolgreiche Versand durch die glsepd (Anforderung F-EPD-4) evaluiert werden.

Schiffsparameter-Entgegennahme Zum Evaluieren des zweiten Teiles der Anforderung F-CA-RP-2 *[...] die Schiffsparameter sollen von der glsepd entgegengenommen und decodiert werden*, müssen folgende Schritte vollzogen werden:

- Starte die Routen-Planungs-Komponente
- Starte die glsepd-Shore
- Konfiguriere die Schiffsparameter in der glsepd-Shore und drücke auf versenden

Als Ergebnis sollte der Logger eine Ausgabe der Form *New shipparameters: die übergebenen Schiffsparameter* anzeigen, was bedeutet, dass die Schiffsparameter erfolgreich entgegengenommen, decodiert und zwischengespeichert wurden. Im Rahmen des Integrationstests wurden die Schiffsparameter erfolgreich entgegengenommen, decodiert und zwischengespeichert. Durch den erfolgreichen Empfang konnte auch der erfolgreiche Versand der Schiffsparameter durch die glsepd (Anforderung F-EPD-4) evaluiert werden.

Routen-Versand Zum Evaluieren der Anforderung F-CA-RP-5 *Die valide, globale Route soll an die glsepd zurückgeschickt werden*, müssen folgende Schritte vollzogen werden:

- Starte die Routen-Planungs-Komponente
- Starte die glsepd-Shore
- Starte den NoGoSolver
- Konfiguriere Schiffsparameter und eine Route in der glsepd-Shore und drücke auf versenden

Als Ergebnis sollte in der EPD nach wenigen Sekunden eine valide Route angezeigt werden, was bedeutet, dass die neu berechnete Route an die glsepd versandt wurde. Im Rahmen des Integrationstests wurde die valide, globale Route erfolgreich an die glsepd versandt. Somit konnte auch Anforderung F-EPD-5 *Das System muss eine optimierte Route vom Routeplanning empfangen und auf der Karte darstellen können.* evaluiert werden.

Darstellung von DOI-Rohdaten (AIS-, Radar- und fusionierte Schiffe) im Küstenleitstand Zum Evaluieren der Anforderungen F-EPD-1.2 *Das System muss Radar-Daten auf der Karte anzeigen können.* und F-EPD-1.1 *Das System muss AIS-Daten auf der Karte anzeigen können.*, müssen folgende Schritte vollzogen werden:

- Starte die RabbitMQ Verbindung zum Wilkinson Server
- Starte das Polymorphic Interface
- Starte den Hub (Polymorphic Interface Instance)
- Starte die Dynamische Objektidentifizierung
- Starte die glsepd-Shore
- Aktiviere den DOIDataLayer im MapView-Plugin

Als Ergebnis sollten im DOIDataLayer drei verschieden farbige Symbole auf der Karte angezeigt werden. Grüne (Radar), Lilane (AIS) und rote (fusionierte Schiffe). Im Rahmen des Integrationstests wurden die von der DOI erkannten Schiffe erfolgreich empfangen und angezeigt.

Darstellung von Debug-Informationen der Pfadplanung im Küstenleitstand Zum Evaluieren der Anforderungen F-EPD-1.3 *Das System muss Grid-Daten auf der Karte anzeigen können.* und F-EPD-1.4 *Das System muss Polygone auf der Karte anzeigen können.* müssen folgende Schritte vollzogen werden:

- Starte den Hub
- Starte den NoGoSolver
- Starte die Dynamische Objektidentifizierung
- Starte die Kollisionverhütungskomponente
- Starte den Controller
- Starte den Küstenleitstand
- Starte HAGGIS und innerhalb dessen ein beliebiges Testszenario
- Konfiguriere eine Route und die Schiffsparameter im Küstenleitstand und sende sie an die Kollisionverhütungskomponente
- Aktiviere den PathPlanningDataLayer im MapView-Plugin

Als Ergebnis sollten im PathPlanningDataLayer GoodWinShapes erkannter und berücksichtigter Schiffe der Pfadplanungskomponente sowie die berücksichtigten statischen Hindernisse erscheinen. Im Rahmen des Integrationstests wurden die von der Pfadplanungskomponente gesendeten Hindernisdaten erfolgreich empfangen und angezeigt.

Darstellung der eigenen Schiffsposition im Küstenleitstand Zum Evaluieren der Anforderungen F-EPD-1.5 *Das System muss die Schiffsposition auf der Karte anzeigen können.* müssen folgende Schritte vollzogen werden:

- Starte den Hub
- Starte den Küstenleitstand (inkl. aktiviertem NMEA0183-Sensor)
- Starte HAGGIS und innerhalb dessen ein beliebiges Testszenario
- Aktiviere den ownShip-Layer im MapView-Plugin

Als Ergebnis sollte im ownShipLayer ein ECDIS-konformes OwnShip visualisiert werden. Im Rahmen des Integrationstests wurden die im Szenario hinterlegten ownship-Positionen erfolgreich empfangen und angezeigt.

7.2.4. Simulationsadapter

Mittels der Integrationstests wird sichergestellt, dass der Simualtionsadapter dazu in der Lage ist, alle benötigten Nachrichten aus HAGGIS zu empfangen und die erstellten Nachrichten via RabbitMQ zu versenden. Um die versendeten Nachrichten zu überprüfen wurden diese durch die *Dynamische Objektidentifizierung* verifiziert.

Empfangen der Daten aus HAGGIS Zum Evaluieren der Anforderung F-SA-1, dass der Simulationsadapter XML Daten aus HAGGIS empfängt, müssen folgende Schritte vollzogen werden:

- Starte den Simulationsadapter
- Starte HAGGIS.

Um das Ergebnis zu überprüfen wurde zu Testzwecken eine Ausgabe im Simulationsadapter mit der Größe der empfangenen Nachricht über die Konsole erzeugt. Diese Ausgabe entsprach den erwarteten Zeitabständen der empfangenen Nachrichten von einer Nachricht pro 0,1 Sekunden, dies wurde mittels einer Latenzmessung bestätigt.

Versenden der erstellten Nachrichten via RabbitMQ Zum Überprüfen des Versendens der erstellten Nachrichten via RabbitMQ wurden die Nachrichten mittels der *Dynamischen Objektidentifizierung* vom Hub abgerufen und weiter verarbeitet. Die Weiterverarbeitung der Daten umfasste hierbei eine Anzeige der simulierten Schiffe in der EPD.

- Starte den Simulationsadapter
- Starte HAGGIS
- Starte die RabbitMQ Verbindung zum Wilkinson Server
- Starte das Polymorphic Interface
- Starte den Hub (Polymorphic Interface Instance)
- Starte die Dynamic Object Identification
- Starte die EPD

Als Ergebnis sollte deutlich werden, dass die Nachrichten durch den Simulationsadapter via RabbitMQ versendet wurden, sowie diese Nachrichten von der *Dynamischen Objektidentifizierung* verarbeitet werden können. Im Rahmen des Integrationstests wurden die AIS und Radar Nachrichten erfolgreich via RabbitMQ versendet und konnten durch die *Dynamische Objektidentifizierung* in der EPD visualisiert werden. Eine genaue Aufschlüsselung der Integrationstests ist im Anhang unter Abschnitt B.3 ersichtlich.

7.2.5. Kollisionverhütung

Durch den Komponententest wird zu einem sichergestellt, dass die Kollisionverhütungskomponente in der Lage ist alle benötigten Informationen (statische & dynamische Hindernisse, Position & Heading, Schiffsparameter & zuverfolgende Route) anzufragen, entgegenzunehmen und zu decodieren. Zum anderem wird kontrolliert ob der errechnete und vom Controller zuverfolgende Pfad sowohl an den Controller als auch an den Küstenleitstand erfolgreich übermittelt wird.

Integrationstests

Im folgenden Abschnitt werden die durchgeführten Integrationstests der Kollisionverhütung beschrieben sowie die Ergebnisse vorgestellt.

NoGoSolver-Anfrage Zum Evaluieren des ersten Teiles der Anforderung F-CA-PP-1 *Die Komponente soll statische Hindernisse (Land & Untiefen) von dem NoGoSolver anfragen ... können*, müssen folgende Schritte vollzogen werden:

- Starte die Kollisionverhütungskomponente
- Starte den Küstenleitstand
- Starte den NoGoSolver
- Starte den Hub
- Konfiguriere Schiffsparameter und eine Route im Küstenleitstand und drücke auf versenden
- Starte HAGGIS und innerhalb diesen ein beliebiges Testszenario

Als Ergebnis sollte der Logger des NoGoSolvers eine Ausgabe der Form *Now generated Json* anzeigen, was bedeutet, dass die Hindernisse erfolgreich angefragt wurden. Im Rahmen des Integrationstests wurden die Hindernisse erfolgreich angefragt.

NoGoSolver-Entgegennahme Zum Evaluieren des zweiten Teiles der Anforderung F-CA-PP-1 *Die Komponente soll statische Hindernisse (Land & Untiefen) von dem NoGoSolver ..., entgegennehmen und decodieren können*, müssen folgende Schritte vollzogen werden:

- Starte die Kollisionverhütungskomponente
- Starte den Küstenleitstand
- Starte den NoGoSolver
- Starte den Hub
- Konfiguriere Schiffsparameter und eine Route im Küstenleitstand und drücke auf versenden
- Starte HAGGIS und innerhalb diesen ein beliebiges Testszenario

Als Ergebnis sollte der Logger eine Ausgabe der Form *New Obstacles …* anzeigen, was bedeutet, dass die Hindernisse erfolgreich entgegengenommen, decodiert und zwischengespeichert wurden. Im Rahmen des Integrationstests wurden die Hindernisse erfolgreich entgegengenommen, decodiert und zwischengespeichert.

Dynamische Hindernisse-Entgegenahme Zum Evaluieren der Anforderung F-CA-PP-2 *Die Komponente soll dynamische Hindernisse (andere Schiffe) von der Dynamischen Objektidentifizierung entgegennehmen und decodieren können*, müssen folgende Schritte vollzogen werden:

- Starte den Hub und verbinde diesen per RabbitMQ auf die NaviBox in Wilhelmshaven
- Starte die Dynamischen Objektidentifizierung
- Starte die Kollisionverhütungskomponente

Als Ergebnis sollte der Logger eine Ausgabe der Form *New dynamic obstacles …* anzeigen, was bedeutet, dass die dynamische Hindernisse erfolgreich entgegengenommen, decodiert und zwischengespeichert wurden. Im Rahmen des Integrationstests wurden die dynamische Hindernisse erfolgreich entgegengenommen, decodiert und zwischengespeichert.

Routen-Entgegennahme Zum Evaluieren des ersten Teiles der Anforderung F-CA-PP-3 *Die globale Route … soll von der EPD entgegengenommen und decodiert werden*, müssen folgende Schritte vollzogen werden:

- Starte die Kollisionverhütungskomponente
- Starte den Küstenleitstand
- Konfiguriere eine Route im Küstenleitstand und drücke auf versenden

Als Ergebnis sollte der Logger eine Ausgabe der Form *New active route: die übergebenen Route in RTZ-Form* anzeigen, was bedeutet, dass die Route erfolgreich entgegengenommen, decodiert und zwischengespeichert wurde. Im Rahmen des Integrationstests wurde die Route erfolgreich entgegengenommen, decodiert und zwischengespeichert. Durch den erfolgreichen Empfang konnte auch der erfolgreiche Versand durch die EPD (Anforderung F-EPD-6) evaluiert werden.

Schiffsparameter-Entgegennahme Zum Evaluieren des zweiten Teiles der Anforderung F-CA-PP-3 *... die Schiffsparameter sollen von der EPD entgegengenommen und decodiert werden*, müssen folgende Schritte vollzogen werden:

- Starte die Kollisionverhütungskomponente
- Starte den Küstenleitstand
- Konfiguriere die Schiffsparameter im Küstenleitstand und drücke auf versenden

Als Ergebnis sollte der Logger eine Ausgabe der Form *New shipparameters ...* anzeigen, was bedeutet, dass die Schiffsparameter erfolgreich entgegengenommen, decodiert und zwischengespeichert wurden. Im Rahmen des Integrationstests wurden die Schiffsparameter erfolgreich entgegengenommen, decodiert und zwischengespeichert. Durch den erfolgreichen Empfang konnte auch der erfolgreiche Versand durch die EPD (Anforderung F-EPD-6) evaluiert werden.

Positions-Entgegennahme Zum Evaluieren des ersten Teiles der Anforderung F-CA-PP-4 *Die Position ... solle vom Hub entgegengenommen und decodiert werden.*
, müssen folgende Schritte vollzogen werden:

- Starte den Hub
- Starte HAGGIS und innerhalb diesen ein beliebiges Testszenario
- Starte die Kollisionverhütungskomponente

Als Ergebnis sollte der Logger eine Ausgabe der Form *New postion ...* anzeigen, was bedeutet, dass die Postion erfolgreich entgegengenommen, decodiert und zwischengespeichert wurde. Im Rahmen des Integrationstests wurde die Postion erfolgreich entgegengenommen, decodiert und zwischengespeichert.

Heading-Entgegennahme Zum Evaluieren des ersten Teiles der Anforderung F-CA-PP-4 *... das Heading solle vom Hub entgegengenommen und decodiert werden.*
, müssen folgende Schritte vollzogen werden:

- Starte denEPD Hub
- Starte HAGGIS und innerhalb diesen ein beliebiges Testszenario
- Starte die Kollisionverhütungskomponente

Als Ergebnis sollte der Logger eine Ausgabe der Form *New heading …* anzeigen, was bedeutet, dass das Heading erfolgreich entgegengenommen, decodiert und zwischengespeichert wurde. Im Rahmen des Integrationstests wurde das Heading erfolgreich entgegengenommen, decodiert und zwischengespeichert.

Pfad-Versand an den Controller Zum Evaluieren der Anforderung F-CA-PP-9 *Der valide Pfad soll über das Distribution System an den Controller geschickt werden*, müssen folgende Schritte vollzogen werden:

- Starte den Hub
- Starte den NoGoSolver
- Starte die Dynamischen Objektidentifizierung
- Starte die Kollisionverhütungskomponente
- Starte den Controller
- Starte den Küstenleitstand
- Starte HAGGIS und innerhalb diesen ein beliebiges Testszenario
- Konfiguriere eine Route und die Schiffsparameter im Küstenleitstand und sende sie an die Kollisionverhütungskomponente

Als Ergebnis sollten zum Controller die nächsten zwei anzufahrenden Wegpunkte gesendet werden. Im Rahmen des Integrationstests wurden die nächsten zwei anzufahrenden Wegpunkte erfolgreich an den Controller versandt.

Pfad-Versand an den Küstenleitstand Zum Evaluieren der Anforderung F-CA-PP-10 *Der valide Pfad soll über das Distribution System an die EPD geschickt werden*, müssen folgende Schritte vollzogen werden:

- Starte den Hub
- Starte den NoGoSolver
- Starte die Dynamischen Objektidentifizierung
- Starte die Kollisionverhütungskomponente
- Starte den Controller
- Starte den Küstenleitstand
- Starte HAGGIS und innerhalb diesen ein beliebiges Testszenario

- Konfiguriere eine Route und die Schiffsparameter im Küstenleitstand und sende sie an die Kollisionverhütungskomponente

Als Ergebnis sollte im Küstenleitstand der jeweils anzusteuernde Pfad grafisch dargestellt werden. Im Rahmen des Integrationstests wurde der jeweils anzusteuernde Pfad grafisch dargestellt. Durch die erfolgreiche Darstellung konnte F-EPD-6 positiv evaluiert werden.

7.3. Systemtests

Für die Umsetzung der Systemtests wurde eine Szenario-basierte Testmethode genutzt, welche mittels des im vorangegangenen Abschnitt evaluierten Testkatalogs durchgeführt wurde.

7.3.1. Durchführung des Systemtests

Der Systemtest betrachtet das gesamte Softwaresystem und testet dieses mit Bezug zu den erhobenen Anforderungen. Demnach erfolgt im Systemtest eine Validierung, ob das entwickelte Softwaresystem die gestellten Anforderungen erfüllt. Da ein Systemtest in einer Testumgebung durchgeführt werden sollte, die der Umgebung für den späteren Produktiveinsatz sehr nahe kommt, sollte möglichst diese Hard- und Software auch für den Systemtest genutzt werden. [Spi10]
Die PG MATE II hat wiederholt Systemtests des entwickelten Softwaresystems durchgeführt. Diese wurden im Rahmen des physikalischen Testfelds auf der Zuse durchgeführt. Der Ablauf dieser Systemtests wurde nach jedem Systemtest mit Auswertung der aufgetretenen Schwierigkeiten angepasst. So wurde der zeitliche Ablauf für den Testtag nach jedem Durchlauf weiter optimiert.
Der Ablauf eines Testtags skizziert sich in einer Übersicht durch die folgenden Punkte:

- Inbetriebnahme der Infrastruktur
- Inbetriebnahme und Konfiguration der einzelnen Komponenten
- Inbetriebnahme der Systeme
- Basistest 0-4
- Testszenarien 1-5
- Fehlerinjektion
- Abbau und Abschlussbesprechung

Für eine genaue Ansicht des Ablaufs der Systemtests können die Ablaufpläne im Anhang unter Anhang E betrachtet werden. Für die Erstellung der Testszenarien wurde das simulative Testfeld HAGGIS genutzt. Um die erstellten Testszenarien im physikalischen Testfeld nutzen zu können wurde eine Verknüpfung mittels des Simulationsadapters hergestellt.

7.3.2. Inhalt der Systemtests

Der Aufbau des Testkatalogs für die Durchführung der Systemtests besteht aus verschiedenen Testszenarien, um die gestellten Anforderungen an das System vollständig überprüfen zu können. Der Testkatalog durchlief eine evolutionäre Entwicklung, bei der Artefakte in eigene Testfälle ausgelagert oder neue Szenarien für weitere Fahrmanöver erstellt wurden. Im Folgenden werden die durchgeführten Testszenarien des finalen Testkatalogs kurz beschrieben, eine genaue Darstellung inklusive der zugehörigen Testfälle befindet sich im Anhang unter Anhang F:

- Basistest 0 - Abfahrt fester Wegpunkte
  Das Schiff kann in Richtung eines konstanten, in der Regelungstechnik hinzugefügten, Wegpunkts fahren.
- Basistest 1 - Abfahrt einer optimierten Route
  Nachdem eine Route durch die Routen-Planung optimiert wurde, wird diese durch die Pfad-Planung mittels der Regelungstechnik abgefahren. Dabei werden statische Hindernisse berücksichtigt.
- Basistest 2 - Steuerung der Zuse mittels VirtualHandles
  Anwendung der Virtual Handles zur Steuerung der Zuse über ein webbasiertes Interface (Ruder und Drehzahl).
  Die Regelungstechnik erhält manuelle Steuerbefehle und gibt diese weiter an die Aktuatoren des physikalischen Testfelds.
- Basistest 3 - Erkennen von Schiffen mittels Kamera
  Die Objektdetektions-Komponente muss kontinuierlich die Umgebung in Fahrtrichtung mittels einer Kamera erfassen und Schiffe erkennen. Für erkannte Schiffe, die unter dem Horizon liegen, wird die Position (rechts oder links vom eigenen Schiff) and die DOI als JSON-Objekt gesendet.
  Verarbeitung der von der Objektdetektion empfangenen JSON-Objekte über die Position der erfassten Schiffe mit der Kamera. Es wird geprüft, ob die erkannten Schiffe bereits mit AIS und/oder Radar erkannt wurden. Ist dies nicht der Fall wird ein „Kameraschiff “

in einem 45° Winkel zur eigenen Fahrtrichtung mit einer Geschwindigkeit von 0 Knoten erstellt. Dieses Schiff wird an den Küstenleitstand und an die Pfad-Planung gesendet.

- Testszenario 1 - Kreuzendes Schiff von links (Ausweichen) (COLREG 15, 16, 17)
  Das eigene Schiff soll das von links kreuzende simulierte Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren.

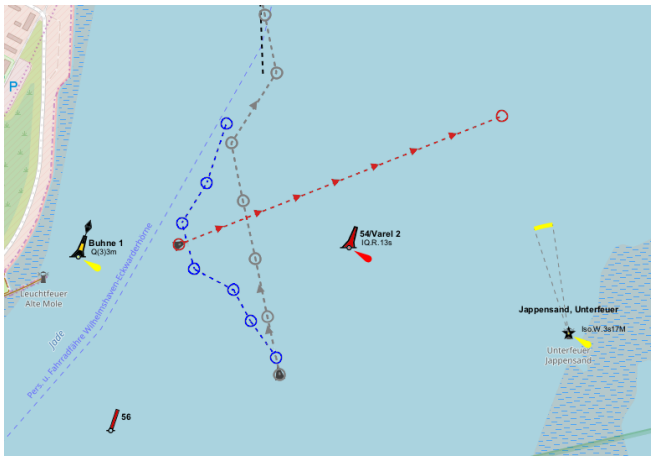


Abbildung 7.3.: Versuchsaufbau Szenario 1 (Route Fremdschiff (rot), erwartete Route autonomes Schiff (blau))

- Testszenario 2 - Kreuzendes Schiff von rechts (COLREG 15, 16, 17)
  Das eigene Schiff soll das von rechts kreuzende simulierte Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren.

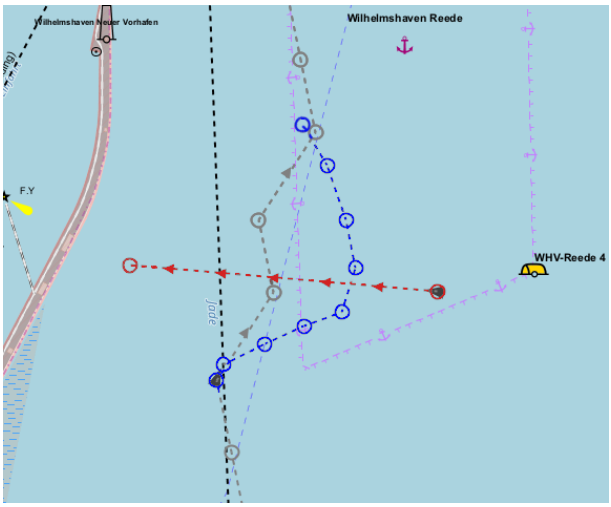


Abbildung 7.4.: Versuchsaufbau Szenario 2 (Route Fremdschiff (rot), erwartete Route autonomes Schiff (blau))

- Testszenario 3 - Entgegenkommendes Schiff (COLREG 14, 16, 17)
  Das eigene Schiff soll das entgegenkommende simulierte Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren.

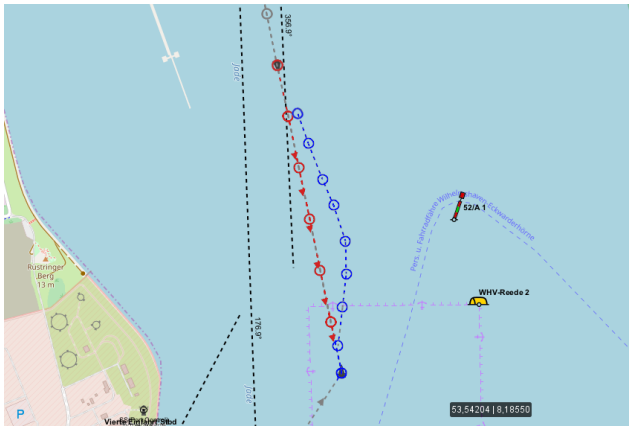


Abbildung 7.5.: Versuchsaufbau Szenario 3 (Route Fremdschiff (rot), erwartete Route autonomes Schiff (blau))

- Testszenario 4 - Ausweichen eines dynamischen und statischen Hindernisses
  Das eigene Schiff soll das simulierte Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. Dabei soll die Kollision verhindert werden. Zusätzlich soll ein im Ausweichpfad liegendes statisches Hindernis bei der Ausweichplanung berücksichtigt werden.

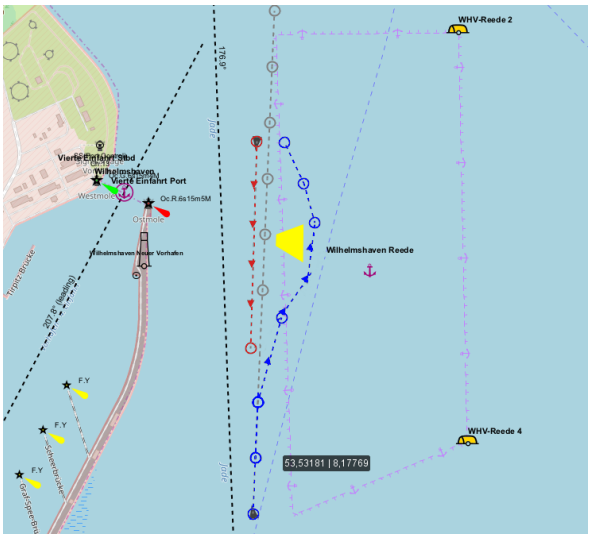


Abbildung 7.6.: Versuchsaufbau Szenario 4 (Route Fremdschiff (rot), erwartete Route autonomes Schiff (blau))

- Testszenario 5 - Überholmanöver (COLREG 13)
  Das eigene Schiff soll im Autopiloten das simulierte Schiff erkennen, überholen und auf die ursprünglich geplante Route zurückkehren.

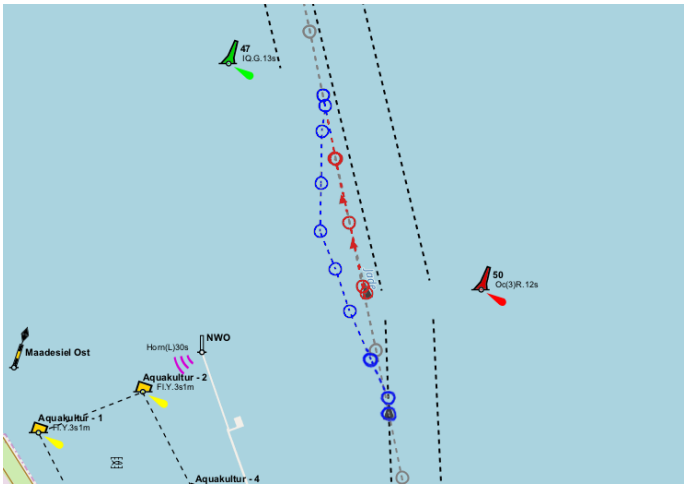


Abbildung 7.7.: Versuchsaufbau Szenario 5 (Route Fremdschiff (rot), erwartete Route autonomes Schiff (blau))

- Fehlerinjektion 1 - Ein Schiff wechselt auf unsere Position
  Ein anderes, bereits vorher über AIS identifiziertes Schiff wechselt die Position auf die Position des eigenen Schiffes.
- Fehlerinjektion 2 - Ein Schiff verschwindet
  Ein existierendes, über AIS erkanntes Schiff verschwindet von seiner bekannten Position.
- Fehlerinjektion 3 - Ein Schiff taucht wieder auf
  Ein zuvor verschwundenes, über AIS erkanntes Schiff erscheint wieder auf seiner echten Position.
- Fehlerinjektion 4 - Wechsel der MMSI
  Ein mit AIS und/oder Radar identifiziertes Schiff in der Umgebung des eigenen Schiffes wechselt die MMSI.
- Fehlerinjektion 5 - Ungültiger Ruderwinkel
  Der Controller empfängt einen außerhalb der spezifizierten Grenzen definierten Ruderwinkel.

Aufgrund von zeitlichen Engpässen bei den Systemtests wurden die Fehlerinjektionen nachträglich im Labor durchgeführt. Alle Tests wurden mit dem gesamten System durchgeführt. Die Sensordaten der Zuse wurden bei Bedarf von HAGGIS simuliert oder als frühere Aufzeichnung von originalen Daten in das System eingespeist.

Bei Fehlerinjektion 1 konnte beobachtet werden, dass sich nach einer kurzen Reaktionszeit der dynamischen Objekterkennung das identifizierte Schiff auf die Position des Eigenschiffes bewegte. Daraufhin befand sich das eigene Schiff im Goodwin-Shape des Fremdschiffes. Die Pfadplanung deaktivierte darauf die Pfadberechnung und der Controller regelte die Geschwindigkeit des Eigenschiffes langsam herunter.
In Fehlerinjektion 2 wurde beobachtet, dass das verschwundene Schiff noch mehrere Sekunden von der dynamischen Objektidentifizierung als Schiff identifiziert wurde, danach verschwand es.
Ähnlich wie in Fehlerinjektion 1 wurde die neue Position des Schiffes nach einer kurzen Reaktionszeit in der Fehlerinjektion 3 identifiziert und das System funktionierte weiterhin ohne Ausfälle.
Bei Fehlerinjektion 4 wurde jedes mal beim Wechsel der MMSI ein neues Schiff erkannt. Die Objektidentifizierung erkannte hier also mehrere Schiffe und blockierte somit einen größeren Bereich als nur ein Schiff blockiert hätte.
In der Fehlerinjektion 5 konnte kein fehlerhaftes Verhalten beobachtet werden. Der Controller funktionierte während des gesamten Tests ordnungsgemäß.

Das gezeigte Verhalten des Systems während der Fehlerinjektionen ist vertretbar. So kann das System beim Ausfall von AIS-Nachrichten in Fehlerinjektion 2 nicht davon ausgehen, dass das erkannte Schiff an der zuletzt empfangenen Position verweilt. In der realen Anwendung wäre ein Ausfall der AIS-Nachrichten außerdem durch die Kamera kompensiert werden können. Falls ein AIS-Schiff nun aber auf der eigenen Schiffsposition liegt ist es durchaus sinnvoll dieses Fehlerzustand bewusst zu erkennen und das Schiff langsam anzuhalten. Bei Fehlerinjektion 4 tritt eine Situation auf, die vom System nicht als Problem identifiziert wird. Trotzdem ist durch die eingebauten Sicherheitsmechanismen der dynamischen Objekterkennung eine Kollisionsgefahr ausgeschlossen. Der betroffene Bereich wird durch diese Mechanismen sogar weiträumiger als sonst umfahren.

Während der Systemtests wurden die aufgelisteten Testszenarien strukturiert durchgeführt. Die Ergebnisse wurden während der Testdurchführung protokolliert und im Anschluss ausgewertet.

7.3.3. Auswertung der Systemtests

Anhand des im Vorangegangenen beschriebenen Testkatalogs und der während der Systemtests dokumentierten Ergebnisse konnte im Anschluss eine Auswertung der Systemtests erfolgen. Im Folgenden werden die Ergebnisse der Systemtests aufgeführt. Hierbei wird zunächst aufgelistet, wie viele Testfälle pro Testzenario erfolgreich getestet werden konnten, danach erfolgt eine

Darstellung der Testabdeckung in Prozent. Bei der Prozentualen Darstellung gibt es jeweils ein Ergebnis, das die gesamten durchgeführten Testfälle berücksichtigt und jeweils ein Ergebnis, das die in den Testszenarien mehrfach vorkommenden Basistests nur einfach für den gesamten Test wertet.

Systemtest 15.08.2018

Basistests	2/2
TS-1	21/22
TS-2	25/28
TS-3	27/31
TS-4	26/32
TS-5	26/31
Gesamt	127/146
Prozentual Gesamt	86.39 %
Prozentual einfache Basistests	76.82 %

Insgesamt ist die Testabdeckung von fast 77 % positiv zu bewerten. Der detaillierte Testkataloge befindet sich im Anhang unter Abschnitt F. Im Folgenden wird detaillierter auf die Ergebnisse des Systemtest eingegangen.

Kommunikation Nach erfolgter Konfiguration der Komponenten konnten in sämtlichen Tests Daten fehlerfrei zwischen den Komponenten übertragen werden.

Routenmonitoring Die durch die Komponenten generierten Informationen (Routen, Debug-Informationen, wie Polygone, und AIS- und Radar-Schiffe) wurden stets im Küstenleitstand dargestellt.

Regelungssystem Die zu dem Zeitpunkt implementierte Regelungstechnik regelte die Geschwindigkeit sehr langsam nach oben, während Geschwindigkeitsabsenkungen abrupt erfolgten. Ein „sanftes“ Gleiten durch das Wasser war somit nicht möglich. Aus diesem Grund wurde entschieden, mit einer konstant konfigurierten Geschwindigkeit die Tests durchzuführen. Das Einstellen von Geschwindigkeiten in der globalen Route war demnach wirkungslos.

Schiffserkennung mittels Kamera Die Schiffserkennung der Objektidentifizierung hat Schiffe mittels der eingesetzten Webcam erkannt, jedoch waren diese sehr schlecht auf den Bildern sichtbar. Entschieden wurde, dass eine Kamera mit höherer Auflösung eingesetzt werden soll. Die Reaktion der Pfad-Planungs-Komponente konnte zu dem Zeitpunkt noch nicht getestet werden. In einigen Testszenarien standen zudem keine realen Schiffe in der näheren Umgebung zur Verfügung, sodass die Funktion nicht konkret getestet werden konnte. Dies wurde in späteren Labortests mit aufgezeichnetem Videomaterial erfolgreich nachgeholt.

TS-1 - Route automatisiert abfahren Nachdem die Route erfolgreich im Küstenleitstand erstellt, durch die Routen-Planung optimiert und durch Weiterleiten an die Pfad-Planung gestartet wurde, konnte diese durch das Regelungssystem fehlerfrei abgefahren werden (jedoch mit konstanter Geschwindigkeit).

TS-2 - Überholmanöver COLREG 13 Das Überholmanöver gelang größtenteils. Der von der Pfad-Planung berechnete Pfad wurde abgefahren, bis der Überholvorgang durch Rückführung auf den globalen Pfad beendet werden sollte. Die Rückführung auf den globalen Pfad schlug fehl, weil ein Wegpunkt nicht als erreicht markiert wurde. Dadurch wendete das Schiff nach erfolgreichem Überholen und fuhr zu einem bereits abgefahrenen Wegpunkt zurück.

TS-3- kreuzendes Schiff von rechts COLREG 15, 16, 17 Dem von rechts kreuzenden Schiff konnte erfolgreich (d.h. kollisionsfrei) ausgewichen werden. Die Rückführung auf den globalen Pfad schlug fehl, weil ein Wegpunkt nicht als erreicht markiert wurde. Dadurch wendete das Schiff nach erfolgreichem Überholen und fuhr zu einem bereits abgefahrenen Wegpunkt zurück.

TS-4 - kreuzendes Schiff von links COLREG 15, 16, 17 Während der Pfadverfolgung wurde der abzufahrene Pfad zu „sprunghaft“ berechnet, d.h. das Schiff fuhr mehrfach von der einen in die andere Fahrtrichtung, sodass der berechnete Pfad mehrfach den Kurs des kreuzenden Schiffs kreuzte und eine Kollision drohte. Die Rückführung auf den globalen Pfad erfolgte jedoch problemlos.

TS-5 - entgegenkommendes Schiff COLREG 14, 16, 17 Während der Pfadverfolgung wurden mehrfach zu wenig Wegpunkte an das Regelungssystem gesendet, sodass die Steuerung nicht fehlerfrei funktioniert hat und das autonome Schiff zu stark vom berechneten Pfad abwich. Dies führte zu einem Testabbruch, der Test wurde anschließend erneut durchgeführt. Beim zweiten

Testablauf wurden die Rohdaten des simulierten Schiffs (AIS- und Radar-Daten) durch die Dynamische Objektidentifizierung nicht fusioniert, wodurch mehrere Schiffe der Pfad-Planung übergeben und so fehlerhafte Pfade berechnet wurden. Dies führte zu einem Testabbruch, der Test wurde anschließend erneut durchgeführt. Im dritten Testlauf konnte dem entgegenkommenden Schiff erfolgreich ausgewichen werden.

Systemtest 29.08.2018
Im Gegensatz zum vorherigen Systemtest wurde ein neues Szenario entwickelt, die Fehlerinjektion ausgearbeitet und allgemeine Tests, die nicht konkret in den Szenarien überprüft werden konnten, als „Basistests“ in den Katalog aufgenommen. Basistests wurden bei den folgenden Tests immer zu Beginn durchgeführt, um anschließend mit einer getesteten Infrastruktur mit den Szenarien fortzufahren.

Basistests	2/6
TS-1	24/24
TS-2	24/24
TS-3	0/24
TS-4	0/24
TS-5	0/24
Gesamt	50/126
Prozentual Gesamt	39.68 %
Prozentual einfache Basistests	54.84 %

Insgesamt ist die Testabdeckung von fast 40 % allgemein negativ zu bewerten. Der detaillierte Testkataloge befindet sich im Anhang unter Abschnitt F. Während des Tests kam es zu einem Ausfall des internen Bordstromnetzes, sodass die Komponenten des Systems von MATE II nicht mehr betrieben werden konnten. Infolgedessen musste der Systemtest abgebrochen werden. Berücksichtigt man jedoch, dass die einzig durchgeführten Testszenarien sehr erfolgreich getestet werden konnten, wären bessere Ergebnisse möglich gewesen.
Im Folgenden wird detaillierter auf die Ergebnisse des Systemtest eingegangen.

Kommunikation Nach erfolgter Konfiguration der Komponenten konnten in sämtlichen Tests Daten fehlerfrei zwischen den Komponenten übertragen werden. Zwischenzeitlich musste jedoch der RaspberryPI neugestartet werden, damit Steuerkommandos durch das Regelungssystem verarbeitet werden konnten.

Routenmonitoring Die durch die Komponenten generierten Informationen (Routen, Debug-Informationen, wie Polygone, und AIS- und Radar-Schiffe) wurden stets im Küstenleitstand dargestellt.

Regelungssystem Die implementierte Regelungstechnik konnte nach dem bereits beschriebenen Systemtest korrigiert werden, sodass nun ein „sanftes“ Gleiten durch das Wasser möglich war. Dies führte dazu, dass Geschwindigkeiten aus Wegpunkten der globalen Route bei der autonomen Fahrt berücksichtigt werden konnten.

Basistest 0 - Abfahrt fester Wegpunkte Der fest im Regelungssystem hinterlegte Wegpunkt wurde mit der konfigurierten Zielgeschwindigkeit erfolgreich angefahren. Anzumerken ist, dass das autonome Schiff bei autonomer Regelung oszilliert und die abgefahrene Strecke leichte Kurven enthielt.

Basistest 1 - Abfahrt einer optimierten Route Nachdem die Route erfolgreich im Küstenleitstand erstellt, durch die Routen-Planung optimiert und durch Weiterleiten an die Pfad-Planung gestartet wurde, konnte diese durch das Regelungssystem fehlerfrei abgefahren werden.

Basistest 2 - Steuerung der Zuse mittels VirtualHandles Als von MATE I übernommenes bzw. portiertes Artefakt konnte die Steuerung insoweit getestet werden, dass die Steuereinheit („VirtualHandles“) korrekt die Ruder- und Motordrehzahl-Befehle an den Controller übertragen konnte. Dies konnte durch spätere Labortests bestätigt werden. Ein manuelles Steuern des autonomen Schiffes als Notfallsteuerung ist demnach möglich.

Basistest 3 - Erkennen von Schiffen mittels Kamera Die Schiffserkennung mittels Kamera konnte nicht durchgeführt werden, da zum Testzeitpunkt keine Kamera zur Verfügung stand. Die Tests konnten anschließend in späteren Labortests erfolgreich durchgeführt werden.

TS-1 - kreuzendes Schiff von links COLREG 15, 16, 17 Das autonome Schiff konnte dem berechneten Pfad fehlerfrei folgen. Eine Kollision durch das kreuzende Schiff konnte verhindert werden, das Ausweichmanöver wurde durchgeführt. Die Rückführung auf den globalen Pfad erfolgte ebenfalls problemlos.

TS-2 - kreuzendes Schiff von rechts COLREG 15, 16, 17 Im ersten Testablauf kam es zu einer Kollision zwischen dem autonomen und dem virtuellen Schiff. Grund hierfür war eine Asynchronität zwischen der Realität und der Simulation (die Simulation wurde zu früh oder zu spät gestartet). Im zweiten Testablauf konnte dem von rechts kreuzenden Schiff erfolgreich (d.h. kollisionsfrei) ausgewichen werden. Die Rückführung auf den globalen Pfad gelang ebenfalls. Das Ausweichmanöver entsprach dem erwarteten Ergebnis.

TS-3 - entgegenkommendes Schiff COLREG 14, 16, 17 Infolge eines Ausfalls der Bordelektronik konnte das Szenario nicht getestet werden.

TS-4 - Ausweichen eines dynamischen und statischen Hindernisses Infolge eines Ausfalls der Bordelektronik konnte das Szenario nicht getestet werden.

TS-5 - Überholmanöver COLREG 13 Infolge eines Ausfalls der Bordelektronik konnte das Szenario nicht getestet werden.

Fehlerinjektion 1-5 Infolge eines Ausfalls der Bordelektronik konnte das Szenario nicht getestet werden. Es wurde beschlossen, die Fehlerinjektion in Labortests durchzuführen.

Systemtest 12.09.2018

Basistests	2/6
TS-1	24/24
TS-2	24/24
TS-3	23/24
TS-4	23/24
TS-5	24/24
Gesamt	120/126
Prozentual Gesamt	95.23 %
Prozentual einfache Basistests	90 %

Insgesamt ist die Testabdeckung von 90 % positiv zu bewerten. Der detaillierte Testkataloge befindet sich im Anhang unter Abschnitt F. Im Folgenden wird detaillierter auf die Ergebnisse des Systemtest eingegangen.

Kommunikation Nach erfolgter Konfiguration der Komponenten konnten in sämtlichen Tests Daten fehlerfrei zwischen den Komponenten übertragen werden. Zwischenzeitlich musste jedoch der RaspberryPI neugestartet werden, damit Steuerkommandos durch das Regelungssystem verarbeitet werden konnten. Während der Testszenarien fiel der eingesetzte RabbitMQ-Server aus, sodass Tests wiederholt werden mussten. Hierfür wurde ein zweiter RabbitMQ-Server auf einem anderen Rechner nachinstalliert und konfiguriert.

Routenmonitoring Die durch die Komponenten generierten Informationen (Routen, Debug-Informationen, wie Polygone, und AIS- und Radar-Schiffe) wurden stets im Küstenleitstand dargestellt.

Regelungssystem Die Regelung erfolgte fehlerfrei, wobei die Geschwindigkeit bei ca. zehn Knoten gehalten wurde. Die Pfadverfolgung erfolgte ebenfalls fehlerfrei.

Basistest 0 - Abfahrt fester Wegpunkte Der fest im Regelungssystem hinterlegte Wegpunkt wurde mit der konfigurierten Zielgeschwindigkeit erfolgreich angefahren. Anzumerken ist, dass das autonome Schiff bei autonomer Regelung oszilliert und die abgefahrene Strecke leichte Kurven enthielt. Aufgrund des (im Vergleich zu den anderen Systemstests) schlechteren Wetters wurde das autonome Schiff einer starken Strömung ausgesetzt, was die Regelung jedoch ohne größere Probleme ausgeglichen hat.

Basistest 1 - Abfahrt einer optimierten Route Nachdem die Route erfolgreich im Küstenleitstand erstellt, durch die Routen-Planung optimiert und durch Weiterleiten an die Pfad-Planung gestartet wurde, konnte diese durch das Regelungssystem fehlerfrei abgefahren werden.

Basistest 2 - Steuerung der Zuse mittels VirtualHandles Die manuelle Steuerung wurde nicht getestet, da die korrekte Funktion in vorherigen Tests bereits getestet werden konnte.

Basistest 3 - Erkennen von Schiffen mittels Kamera Auf der autonom abzufahrenen Route befanden sich keinerlei Schiffe, die zur Erkennung mittels Kamera notwendig waren. Aus diesem Grund wurde der Test abgebrochen. Die Funktion konnte jedoch bereits in vorherigen Labortests erfolgreich getestet werden.

TS-1 - kreuzendes Schiff von links COLREG 15, 16, 17 Das autonome Schiff konnte dem berechneten Pfad fehlerfrei folgen. Eine Kollision durch das kreuzende Schiff konnte verhindert werden, das Ausweichmanöver wurde durchgeführt. Der berechnete Soll-Pfad wurde dabei für ungefähr zehn Meter verlassen, was auch durch eine systembedingte Abweichung (GPS) begünstigt wurde. Die Rückführung auf den globalen Pfad erfolgte anschließend problemlos. Die im Test tatsächlich gefahrene Strecke des autonomen Schiffes im Szenario ist in der Abbildung 7.8 zu sehen.

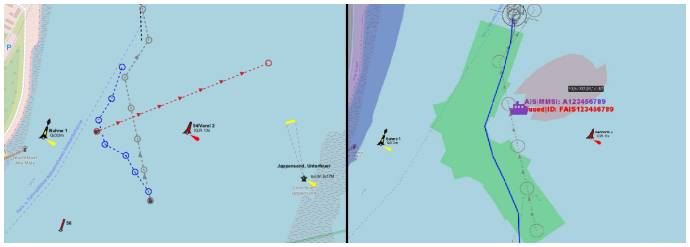


Abbildung 7.8.: Versuchsaufbau des Szenarios 1 (links) und Testergebnisse (rechts)

TS-2 - kreuzendes Schiff von rechts COLREG 15, 16, 17 Das autonome Schiff konnte dem berechneten Pfad fehlerfrei folgen. Eine Kollision durch das kreuzende Schiff konnte verhindert werden, das Ausweichmanöver wurde durchgeführt. Die Rückführung auf den globalen Pfad erfolgte anschließend problemlos. Die im Test tatsächlich gefahrene Strecke des autonomen Schiffes im Szenario ist in der Abbildung 7.9 zu sehen.

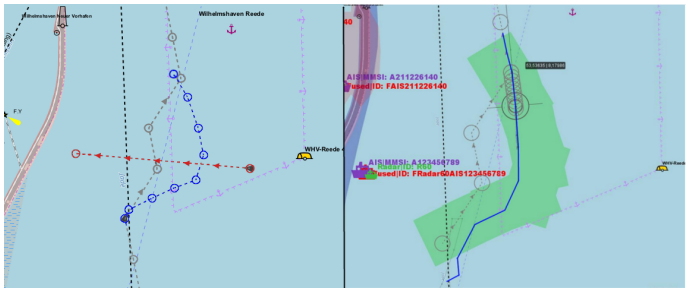


Abbildung 7.9.: Versuchsaufbau des Szenarios 2 (links) und Testergebnisse (rechts)

TS-3 - entgegenkommendes Schiff COLREG 14, 16, 17 Das autonome Schiff konnte dem berechneten Pfad fehlerfrei folgen. Eine Kollision durch das kreuzende Schiff konnte verhindert werden, das Ausweichmanöver wurde durchgeführt. Die Rückführung auf den globalen Pfad war jedoch fehlerhaft, da das autonome Schiff in das GoodWin-Shape gelang und die Fahrt somit

vom Soll-Pfad abwich. Nachdem das GoodWin-Shape verlassen wurde, erfolgt eine planmäßige Abfahrt des Soll-Pfades. Die im Test tatsächlich gefahrene Strecke des autonomen Schiffes im Szenario ist in der Abbildung 7.10 zu sehen.

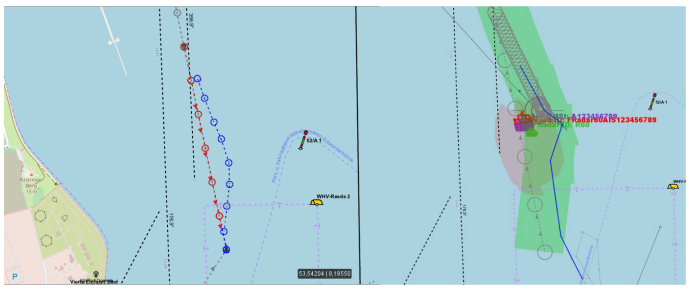


Abbildung 7.10.: Versuchsaufbau des Szenarios 3 (links) und Testergebnisse (rechts)

TS-4 - Ausweichen eines dynamischen und statischen Hindernisses Das autonome Schiff konnte dem berechneten Pfad größtenteils fehlerfrei folgen. Eine Kollision durch das kreuzende Schiff konnte verhindert werden, das Überholmanöver wurde durchgeführt. Die Rückführung auf den globalen Pfad war jedoch fehlerhaft, da das autonome Schiff in das GoodWin-Shape gelang und die Fahrt somit vom Soll-Pfad abwich. Nachdem das GoodWin-Shape verlassen wurde, erfolgt eine planmäßige Abfahrt des Soll-Pfades. Die im Test tatsächlich gefahrene Strecke des autonomen Schiffes im Szenario ist in der Abbildung 7.11 zu sehen.

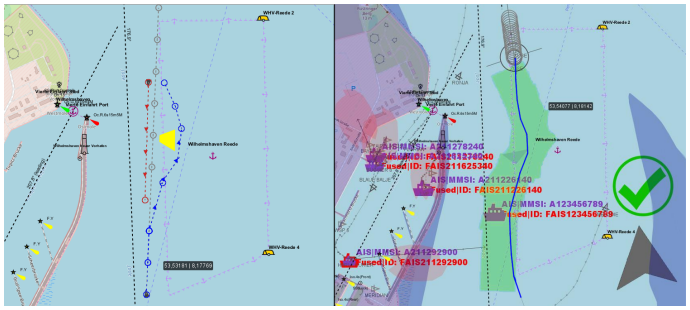


Abbildung 7.11.: Versuchsaufbau des Szenarios 4 (links) und Testergebnisse (rechts)

TS-5 - Überholmanöver COLREG 13 Das autonome Schiff konnte dem berechneten Pfad fehlerfrei folgen. Eine Kollision durch das entgegenkommende Schiff konnte verhindert werden, das Ausweichmanöver wurde durchgeführt. Zudem wurde erfolgreich der virtuellen Untiefe ausgewichen. Die Rückführung auf den globalen Pfad erfolgte fehlerfrei. Die im Test tatsächlich gefahrene Strecke des autonomen Schiffes im Szenario ist in der Abbildung 7.12 zu sehen.

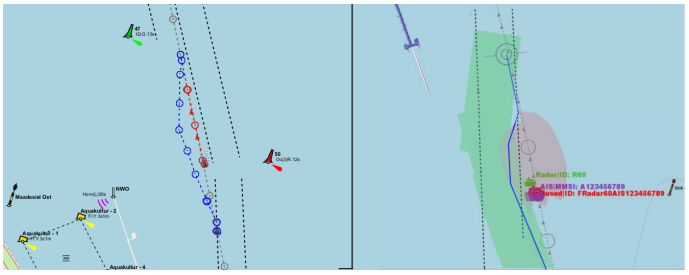


Abbildung 7.12.: Versuchsaufbau des Szenarios 5 (links) und Testergebnisse (rechts)

7.3.4. Veränderungen zwischen den Systemstests

Nach Auswertung des Systemtests vom 15.08.2018 wurden einige Anpassungen durchgeführt, um die aufgetretenen Probleme zu lösen und die Ergebnisse beim nächsten Systemtest, bzw. dem Abnahmetest zu verbessern.
Bei der Komponente der Pfad-Planung wurden die für die kollisionsfreie autonome Fahrt verwendeten GoodWinShapes in ihrer Form optimiert. Ergebnis dieser Änderung war eine stabilere Berechnung des abzufahrenden Pfades (weniger „Sprünge“ des Pfades) und ein daraus resultierendes realistischeres Fahrverhalten des autonomen Schiffes, was im Abnahmetest vom 29.08. sowie 12.09. positiv getestet wurde.
Zudem wurde die Wahl des nächsten Wegpunktes der Pfad-Planung optimiert, sodass eine verbesserte Wegberechnung erfolgen kann und das autonome Schiff ruhiger und kontrollierter die Wegpunkte abfährt.
Zudem wurden die vom Anwender im Küstenleitstand eingegebenen Wegpunkte der abzufahrenden Route als „User-Wegpunkte“ markiert, sodass diese bei der Pfad-Planung und Erreichung von Wegpunkt immer als erreicht gelten müssen. Andere, von der Pfad-Planung dynamisch berechneten Wegpunkte, werden weiterhin während der Fahrt dynamisch gewählt.
Der zum Systemtest vom 15.08. implementierte CPA-Algorithmus in der Pfad-Planung war nicht funktionsfähig, sodass alle Tests ohne den CPA durchgeführt wurde. Hin zu den beiden Abnahmetests wurde der von MTCAS-entwickelte CPA übernommen und konnte in den Abschlusstests erfolgreich eingesetzt werden und so das Ausweichverhalten weiter optimieren.

Die Dynamische Objektidentifizierung wurde zwischen den Tests dahingehend optimiert, dass durch Kamera erkannte Schiffe mit einem größeren Abstand zur eigenen Position des autonomen Schiffes positioniert werden, damit ein rechtzeitiges Ausweichen während der Fahrt möglich wird. Zudem wurde die Geschwindigkeit dieser Schiffe auf Null gesetzt, damit die GoodWinShapes der Pfad-Planung auf die Stelle des Schiffes gesetzt werden, wodurch das Ausweichmanöver des autonomen Schiffes rechtzeitig eingeleitet werden kann. Zur Optimierung der Datenfusion wurde der implementierte Alpha-Beta-Filter für verarbeitete Radar-Daten optimiert. So konnte die Stabilität der Fusionierung der Daten verbessert werden.
Die durchgeführten Anpassungen der Regelungstechnik können im Unterkapitel Evaluation nachgelesen werden.

8. Projektabschluss

Im folgenden Abschnitt wird der Projektabschluss der PG MATE II beschrieben.

8.1. Fazit

In dieser Arbeit hat sich die PG MATE II mit der Entwicklung eines autonom fahrenden Schiffes beschäftigt. Einige der für die Erfüllung dieses Ziels benötigten Komponenten gingen aus der früher durchgeführten PG MATE I oder anderen Arbeiten hervor.
Viele der Komponenten, die für die autonome Fahrt essentiell sind, fehlten jedoch zu Beginn der Arbeit oder wurden den hohen Anforderungen der PG MATE II nicht gerecht.

Um überhaupt eine autonome Reise des Schiffes zu ermöglichen, benötigt man ein Werkzeug um Routen zu planen und die Fahrt beginnen zu können. Die PG MATE II entschied sich hierbei für das vom OFFIS entwickelte ECDIS ähnliche Produkt EPD. Diese Software ist bereits in der Lage, Seekarten standardkonform anzuzeigen, verschiedene Routen anzulegen und zu verwalten. Außerdem beinhaltet sie für diese Arbeit weitere weniger relevante Funktionen.

Nun da eine Route vom Nutzer erstellt werden kann, wird eine Komponente benötigt, die die eingegebene Route auf Fehler prüfen kann. Ein solcher Fehler könnte beispielsweise ein Routenabschnitt sein, der über Land, also nicht befahrbares Gebiet, führt. Diese Aufgabe übernimmt die Routen-Planung. Sie ist in der Lage, eine Route vom Küstenleitstand entgegenzunehmen und anschließend mithilfe verschiedener Algorithmen einen Pfad vom aktuell betrachteten Wegpunkt zum nächsten Wegpunkt zu finden. Sollte dies nicht der Fall sein, gibt die Komponente eine Fehlermeldung an die EPD zurück, die diese anzeigt. Die Routen-Planung stützt sich dabei auch auf den sogenannten NoGoSolver. Diese Software ging aus einer weiteren am OFFIS durchgeführten Projektarbeit hervor. Sie gibt auf Anfrage alle nicht befahrbaren Gebiete zurück.

Befindet der Nutzer die von der Routen-Planung optimierte Route als gut, kann die Navigation begonnen werden. Dazu wird die optimierte Route an die Pfad-Planung geschickt. Diese ist mithilfe der Dynamischen Objektidentifizierung in der Lage, den Pfad dynamisch als Reaktion auf

nicht statische Hindernisse anzupassen. Außerdem schickt die Pfad-Planung die anzufahrenden Wegpunkte an die Regelungskomponente.

Die Dynamische Objektidentifizierung kann mithilfe der auf dem Forschungsboot bereits verbauten und weiteren Sensoren die Umgebung ebendieses erkennen und die Pfad-Planung über eventuelle Hindernisse in Kenntnis setzen. Hierzu wird eine Datenfusion genutzt, die auf bewährte Konzepte aufbaut. Sie ist in der Lage, AIS- und Radarsignale gegebenenfalls dem gleichen Schiff zuzuordnen. So wird die Präzision der Sensoren verfeinert.
Zusätzlich verfügt die Dynamische Objektidentifizierung über die Möglichkeit, Schiffe mithilfe einer Kamera zu erkennen. Somit können Schiffe erfasst werden, die weder einen AIS-Sender haben, noch vom Radar erkannt werden. Die Erkennung findet mithilfe von Machine Learning Algorithmen statt.

Die Regelungskomponente ist in der Lage, mithilfe der aktuell anzufahrenden Wegpunkte, der gewünschten Sollgeschwindigkeit und verschiedenen weiteren Parametern, wie beispielsweise die aktuelle Position und Ist-Geschwindigkeit, einem Pfad mit der entsprechenden Geschwindigkeit zu folgen. Das System besteht bereits seit MATE I, dort wies es jedoch ein starkes Fehlverhalten auf und konnte den Kurs oder die Geschwindigkeit nicht korrekt halten. Nach der Fehlersuche und der anschließenden Substitution bzw. Korrektur der fehlerhaften Komponenten wies das Regelungssystem bereits ein korrekteres Verhalten auf. Abschließend wurden die Parameter des PID-Reglers angepasst, um das System endgültig einsatzfähig zu machen.
Die Steuerbefehle der Regelungskomponente werden von einer aus einer Masterarbeit hervorgegangenen Steuerungseinheit umgesetzt.

Mithilfe der vom OFFIS entwickelten Simulationssoftware HAGGIS wurde das System abschließend getestet. Es ist bereits in der Lage, AIS- und Radarsignale verschiedener Schiffe zu erzeugen. Auch ist es möglich, verschiedene Szenarien, welche Fahrsituationen des Alltags widerspiegeln, zu erstellen. Um diese Form der Tests zu ermöglichen und so eine Brücke zwischen Realität und Simulation zu schlagen, musste jedoch erst ein Adapter implementiert werden, der die simulierten Daten in die Infrastruktur des Forschungsschiffes einspielen kann.

Eine zuletzt im Rahmen der PG MATE II entwickelte Komponente ist Mate Control. Sie dient zur Überwachung des Systems und aller seiner Teilkomponenten während der durchgeführten Testfahrten. So ermöglicht Mate Control den Einblick in die ausgetauschten Daten und den Status der verschiedenen Komponenten. Dabei müssen nur minimale Änderungen am Code der anderen Komponenten vorgenommen werden. Durch diese Kapselung bleibt das System intakt und es stellt keinen Unterschied dar, ob Mate Control zur Laufzeit des Gesamtsystems betrieben wird oder nicht.

In den durchgeführten szenariobasierten Tests erwies sich nicht nur, dass das von der PG MATE II entwickelte Produkt in Lage ist einem Pfad zu folgen. Es wurde auch gezeigt, dass Kollisionen mit Schiffen und Objekten verhindert und Gefahrenstellen wie beispielsweise Untiefen umfahren werden können. Es ist also unter Einsatz moderner Technologien und Konzepte gelungen, die autonome Fahrt im maritimen Bereich zu entwickeln und anschließend erfolgreich zu testen. Auch wurde gezeigt, dass ein solches System zu einer erhöhten Sicherheit auf hoher See und in Hafengebieten führen kann.
So wurde ein weiterer Grundpfeiler für weiterführende Forschung in diesem Bereich gelegt. Weitere Projekte können nun das bereits bestehende System für ihre Forschung nutzen oder die Entwicklung der Experimentierplattform weiter vorantreiben.

8.2. Reflexion

Im Laufe der Entwicklung und Überprüfung der Integrität des Systems traten immer wieder Schwierigkeiten auf. Schon zu Beginn der Projektgruppe gab es leider immer wieder Probleme mit der Infrastruktur auf der Zuse. Hier fehlten Zugangsdaten zu installierter Hardware wie etwa dem RaspberryPI oder der Navibox. Außerdem gab es bis zum Ende Probleme mit dem Radar, die nicht behoben werden konnten. An vielen Stellen musste hier erst eine eigene Herangehensweise entwickelt werden, um die Probleme lösen zu können. Ein Zugang zu einer ausführlichen Dokumentation der Infrastruktur wäre hier sehr hilfreich gewesen. Auch bei den vom OFFIS entwickelten Komponenten, welche von der Projektgruppe genutzt wurden hätte eine bessere Dokumentation die Arbeit erleichtert. An dieser Stelle muss jedoch sehr positiv angemerkt werden, dass die Entwickler dieser Software der Projektgruppe zu jeder Zeit unterstützend zur Verfügung standen. Deshalb möchte sich die Projektgruppe insbesondere bei Sören Schweigert, Florian Klein, Arne Lamm, Stefan Behrensen und Dr.-Ing. Christian Denker für diese Unterstützung bedanken.

Des Weiteren sind im Entwicklungsprozess selbst einige Dinge nicht optimal gelaufen. So hat die Projektgruppe verhältnismäßig spät mit Labortests begonnen, obwohl vom Entwicklungsstand der einzelnen Komponenten her schon früher die Möglichkeit dazu bestanden hätte. Es wurde erst spät klar, dass ebenfalls schon in früheren Entwicklungsphasen Visualisierungen der vom System berechneten Größen (bspw. berechneter Pfad, CPA, fusionierte Objekte) vonnöten gewesen wären. Dies führte dazu, dass visuell leicht identifizierbare Fehler erst spät erkannt werden konnten. Ergänzend lässt sich erkennen, dass es zusätzlich notwendig gewesen wäre eher mit realen Daten zu testen.

Trotz alledem konnte die Projektgruppe diese Probleme durch Unterstützung der Betreuer, weiterer OFFIS Mitarbeiter und vor allem dem guten Gruppenzusammenhalt kompensieren und ein funktionsfähiges Endprodukt abliefern.

8.3. Ausblick

Im Laufe des Projektes entstanden weitere Ansätze für zukünftige Verbesserungen des Systems. Im folgenden Abschnitt werden deshalb potenzielle Weiterentwicklungsmöglichkeiten des Gesamtsystems und den einzelnen Hauptkomponenten beschrieben.

Beginnend bei dem Gesamtsystem fiel beim Wechsel von Labor- zu Systemtests auf der Zuse auf, dass die Konfiguration und das Eintragen der IPs für den Verbindungsaufbau zwischen den Komponenten fehleranfällig und langwierig ist. Hier bietet es sich an, dass die Komponenten um Funktionalitäten wie einer Autodiscovery und Statusanzeige bezüglich der erfolgreichen Verbindungen erweitert werden.

Bezüglich der autonomen Fahrt lässt sich sowohl in der Regelungstechnik als auch in der Pfad-Planung ansetzen, um einerseits das Verfolgen eines Pfades „ruhiger“ zu gestalten, als auch die berechneten Pfade nicht zu stark springen zu lassen.

Das Erkennen von Hindernissen kann durch die Integration weiterer Sensoren in die Datenfusion verbessert werden. Auch wurde an dieser Stelle aus Zeitgründen kein Vergleich von erkannten Hindernissen und statischen Objekten mehr vorgenommen.

Möglichkeiten für Funktionserweiterungen über das aktuelle Produkt hinaus sind beispielsweise das Auswählen mehrerer Routen in der Routen-Planung, eine Voyage-Planung (Wetter, Tankstopps und Abfahrts-/Ankunftszeiten) und dynamische Filter des Datenstroms in Mate Control.

Abbildungsverzeichnis

Literatur

[A08] Milstein A. „Occupancy Grid Maps for Localization and Mapping“. In: *University of Waterloo, Canada* (2008).

[AE15] Daniel Aarno und Jakob Engblom. „Chapter 1 - Introduction“. In: *Full-System Simulation with Simics*. Hrsg. von Daniel Aarno und Jakob Engblom. Boston: Morgan Kaufmann, 2015, S. 1–19. ISBN: 978-0-12-800725-9. DOI: https://doi.org/10.1016/B978-0-12-800725-9.00001-9.

[AS10] *A-Stern Algorithmus*. URL: http://www.geosimulation.de/methoden/a_stern_algorithmus.htm (besucht am 10. 09. 2018).

[asi18] asimshankar. *DetectObjects.java*. 2018]. URL: https://github.com/tensorflow/models/blob/master/samples/languages/java/object_detection/src/main/java/DetectObjects.java (besucht am 15. 06. 2018).

[Bal] Paul Balzer. *Fahrzeugumfeldsensorik: Überblick und Vergleich zwischen Lidar, Radar, Video*. License CC-BY-SA2.0 https://creativecommons.org/licenses/by-sa/2.0/de/. URL: http://www.cbcity.de/fahrzeugumfeldsensorik-ueberblick-und-vergleich-zwischen-lidar-radar-video (besucht am 07. 09. 2018).

[Bal14] Paul Balzer. *Fahrzeugumfeldsensorik: Überblick und Vergleich zwischen Lidar, Radar, Video*. 2014. URL: http://www.cbcity.de/fahrzeugumfeldsensorik-ueberblick-und-vergleich-zwischen-lidar-radar-video (besucht am 11. 09. 2018).

[BH10] B. Berking und W. Huth. „Navigatorische Schiffsführung“. In: *Hamburg: Seehafen-Verlag, DVV Media Group* (2010).

[BH17] Marius Brinkmann und Axel Hahn. *Physical Testbed for Highly Automated and Autonomous Vessels*. Mai 2017.

[Bla12] Michael Blaich. „Extended Grid Based Collision Avoidance Considering COLREGs for Vessels“. In: *9th IFAC Conference on Manoeuvring and Control of Marine Craft* (2012).

[Bla16] Michael Blaich. „Path Planning and Collision Avoidance for Safe Autonomous Vessel Navigation in Dynamic Environments“. Carl von Ossietzky Universitaet Oldenburg, 2016.

[Boc12] Thomas Bock. „Bewertung von Fahrerassistenzsystemen mittels der Vehicle in the Loop-Simulation“. In: *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort.* Hrsg. von Hermann Winner, Stephan Hakuli und Gabriele Wolf. Wiesbaden: Vieweg+Teubner Verlag, 2012, S. 76–83. ISBN: 978-3-8348-8619-4. DOI: 10.1007/978-3-8348-8619-4_9.

[Brine] *3D Simulation.* URL: https://www.emaritime.de/services/haggis/3d-simulation/ (besucht am 15. 09. 2018).

[BWN14] A. G. Bole, Alan Wall und Andy Norris. „Radar and ARPA manual: radar, AIS and target tracking for marine radar users“. In: *Butterworth-Heinemann, Oxford* 3 (2014). [ISBN: 978-0-08-097752-2].

[C13] Pieringer C. „Modellierung des Fahrzeugumfelds mit Occupancy Grids“. In: *Masterarbeit, Universität Passau* (2013).

[Coc11] A. Cockcroft. „A guide to the collision avoidance rules : International Regulations for Preventing Collisions “. In: *Butterworth-Heinemann, Kidlington* 7 (2011). [ISBN 978-0-08-09717-1].

[CPA8] *MI Simulators. Some radar theory.* URL: http : / / www . misimulators . co . uk / ONLINE/documents/theory/theory.html (besucht am 10. 09. 2018).

[CX09] Liu Chang und Shi Xiaofei. „Study of data fusion of AIS and radar“. In: *Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of.* IEEE. 2009, S. 674–677.

[DIF] *Difference Between GPS and DGPS.* URL: https://techdifferences.com/difference-between-gps-and-dgps.html (besucht am 07. 09. 2018).

[ECI] *About ECDIS.* URL: http://www.ecdis-info.com/about_ecdis.html (besucht am 19. 09. 2018).

[ENC] *ELECTRONIC NAVIGATIONAL CHARTS (ENCs)"PRODUCTION, MAINTENANCE AND DISTRIBUTION GUIDANCE".* 2012. URL: https : / / www . iho . int / iho _ pubs / standard/S-65/S-65_ed2.0.0_Apr12.pdf (besucht am 23. 09. 2018).

[F+14] Duchoň F u. a. „Path Planning with Modified a Star Algorithm for a Mobile Robot“. In: *In Procedia Engineering* 96 (2014).

[F06] McNally M. F. „Walking the Grid: Robotics in CS 2“. In: *Alma College* (2006).

[FBS03] Thor I. Fossen, Morten Breivik und Roger Skjetne. „Line-of-sight path following of underactuated marine craft“. In: *IFAC Proceedings Volumes* 36.21 (Sep. 2003), S. 211–216. DOI: 10.1016/s1474-6670(17)37809-6.

[Fos11] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, 12. Apr. 2011. 596 S.

[Fra17] Justin Francis. *Object detection with TensorFlow. How to create your own custom object detection model.* 2017. URL: https://www.oreilly.com/ideas/object-detection-with-tensorflow (besucht am 13. 09. 2018).

[GEF18] *Entwicklungsprozess.* URL: http://www.gefaz.de/Entwicklungsprozess.html (besucht am 14. 09. 2018).

[GML] *Geography Markup Language.* URL: http://www.opengeospatial.org/standards/gml (besucht am 21. 09. 2018).

[Gns] *ECDIS Standards IEC 61174 – The Facts.* URL: https://www.gnsworldwide.com/ecdis-standards-iec-61174-facts/ (besucht am 18. 09. 2018).

[Gre17] Rudolph Greub. „Target Association von Schiffstracks unterschiedlicher maritimer Sensoren zur Vorbereitung auf die Datenfusion heterogener Beobachtungsdatensätze“. Bachelor's Thesis. Carl von Ossietzky Universität, 2017.

[HEUa] Axel Hahn und André Bolles. „Save Maritime Systems Testbed“. In: *Annual of Navigation Gdynia : De Gruyter* 21 (2014), S. 19–34.

[HEUb] *Mathematische Grundlagen Heuristik A*-Algorithmus.* URL: http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html (besucht am 20. 09. 2018).

[Hop] Dipl.-Ing. (FH) Michael Hoppe. *Technische Details und Systemparameter der DGPS-Referenzstationen nach IALA Standard.* URL: http://www.fvt.wsv.de/dgps/pdf/technische%5C_daten.pdf (besucht am 07. 09. 2018).

[Hub] Stefan Hubert. *Hough-Transformation.* URL: http://page.mi.fu-berlin.de/alt/vorlesungen/sem04/10_6Die%20Hough.doc (besucht am 16. 09. 2018).

[IA01] *IALA AIDS TO NAVIGATION GUIDE (Navguide).* 2001. URL: http://www.eskema.eu/DownloadFile.aspx?tableName=tblSubjectArticles&field=PDF%20Filename&idField=subjectArticleID&id=157 (besucht am 30. 09. 2018).

[IHO] *S-57, S-63 and S-52: The latest IHO Standards and what they mean.* URL: https://www.admiralty.co.uk/news/blogs/s-57-and-the-latest-iho-standards (besucht am 23. 09. 2018).

[IMO] *Convention on the International Regulations for Preventing Collisions at Sea, 1972 (COLREGs)*. URL: http://www.imo.org/en/About/Conventions/ListOfConventions/Pages/COLREG.aspx (besucht am 07. 09. 2018).

[Int] MUNIN – Maritime Unmanned Navigation through Intelligence in Networks. *The Autonomous Ship*. URL: http://www.unmanned-ship.org/munin/about/the-autonomus-ship/ (besucht am 13. 09. 2018).

[Ise10] Rolf Isermann. „Elektronisches Management motorischer Fahrzeugantriebe Elektronik, Modellbildung, Regelung und Diagnose für Verbrennungsmotoren, Getriebe und Elektroantriebe". In: *Vieweg+Teubner, Wiesbaden* (2010). [ISBN 978-3-8348-0855-4].

[ISS18] Projektgruppe ISS. „Abschlussdokumentation Intelligente Schiffssimulation". In: (2017-2018]). [Letzter Zugriff: 12.09.2019].

[IV16] Projektgruppe MOPS IV. „Abschlussbericht der Projektgruppe Marine Observation Platform for Surfaces IV". In: (2015-2016). [Letzter Zugriff: 12.09.2019].

[Joh] Prof. Dr. Christian Johner. *V-Modell vs. Wasserfallmodell für Hardware- und Softwareentwicklung*. [Letzter Zugriff: 14.09.2018]. URL: https://www.johner-institut.de/blog/iec-62304-medizinische-software/v-modell/ (besucht am 14. 09. 2018).

[Kaz13] Witold Kazimierski. „Problems of data fusion of tracking radar and AIS for the needs of integrated navigation systems at sea". In: *Radar Symposium (IRS), 2013 14th International*. Bd. 1. IEEE. 2013, S. 270–275.

[KBL13] Univ-Prof. Dr. Hermann Krallmann, Dr. Annette Bobrik und Dr. Olga Levina. „Systemanalyse im Unternehmen Prozessorientierte Methoden der Wirtschaftsinformatik". In: *Oldenbourg Verlag München* 6 (2013). [ISBN 978-3-486-71768-6].

[Kla06] Ferdinand Klaus. „Einführung in Techniken und Methoden der Multisensor-Datenfusion". In: (2006). URL: http://dokumentix.ub.uni-siegen.de/opus/volltexte/2006/57/pdf/klaus.pdf.

[Lan] Norddeutsche Landesbank. *Anzahl der Schiffe in der globalen Containerschiffsflotte in den Jahren 2000 bis 2017*. URL: https://de.statista.com/statistik/daten/studie/28570/umfrage/anzahl-der-schiffe-der-globalen-containerschiffsflotte/ (besucht am 11. 09. 2018).

[LMS1] *LMS151-10100 LMS1xx*. URL: https://cdn.sick.com/media/pdf/0/40/840/dataSheet_LMS151-10100_1047607_en.pdf (besucht am 30. 09. 2018).

[Loh11] Hans Lohninger. *Gütemaße für Klassifikatoren*. 2011. URL: http://statistics4u.com/fundstat_germ/ee_classifier_performance_metrics.html (besucht am 15. 09. 2018).

[LZ+16] Sang L-Z u. a. „CPA Calculation Method based on AIS Position Prediction“. In: *The Journal of Navigation* 69 (2016).

[Man09] W. Mansfeld. „Satellitenortung und Navigation: Grundlagen und Anwendung globaler Satellitennavigationssysteme“. In: *Vieweg Verlag* (1998/2009). [ISBN 3-528-06886-8].

[Mat] Mathworks. *Design and implement PID controllers*. URL: https://de.mathworks.com/discovery/pid-control.html (besucht am 21. 11. 2017).

[MAT17] Projektgruppe – MATE. „Maritime Test- und Experimentierplattform Abschlussdokumentation“. In: (2016-2017). [Letzter Zugriff: 12.09.2019].

[Mau+15] Markus Maurer u. a. „Autonomes Fahren; Technische, rechtliche und gesellschaftliche Aspekte“. In: *Daimler und Benz Stiftung* (2015). [Letzter Zugriff: ISBN: 978-3-662-45853-2].

[MS] Hiroki Mizumoto und Nobutaka Suzuki. *A Simple XSLT Processor for Distributed XML*. Springer Berlin Heidelberg,

[Nic] Birger Nicolai. *Autonomes Fahren: Die Geisterschiffe sind schon startklar*. URL: https://www.welt.de/wirtschaft/article170160411/Die-Geisterschiffe-sind-schon-startklar.html (besucht am 13. 09. 2018).

[NM01] *NMEA 0183 Standard*. URL: https://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp (besucht am 14. 09. 2018).

[NM20] *NMEA 2000 Standard*. URL: https://www.nmea.org/content/nmea_standards/nmea_2000_ed3_10.asp (besucht am 14. 09. 2018).

[Opea] OpenCV. *Hough Line Transform*. URL: https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html (besucht am 16. 09. 2018).

[Opeb] OpenCV. *Hough Line Transform*. URL: https://docs.opencv.org/3.4.0/d9/db0/tutorial_hough_lines.html (besucht am 17. 09. 2018).

[Orl13] Peter F. Orlowski. *Praktische Regeltechnik*. Springer Berlin Heidelberg, 4. Dez. 2013.

[Ove16] H. Overschmidt. „Sportbootführerschein See. mit amtlichem Fragenkatalog“. In: *Daimler und Benz Stiftung* 34 (2016).

[Paw] Dr. Boris Pawlowski. *Autonome Schifffahrt: Transdisziplinäre Forschung an der Uni Kiel.* URL: https://www.uni-kiel.de/pressemeldungen/index.php?pmid=2018-154-autonome-schifffahrt (besucht am 13. 09. 2018).

[Ran95] Thomas Ranney. „Models of driving behavior: A review of their evolution". In: 26 (Jan. 1995), S. 733–50.

[RB] Ward Robert und Greenslade Barrie. *S-100 - Universal Hydrographic Data Model – Info paper.* URL: https://www.iho.int/mtg_docs/com_wg/TSMAD/TSMAD_Misc/S-100InfoPaper_FinalJan2011.pdf (besucht am 19. 09. 2018).

[RB12] Ø. J. Rødseth und H-C. Burmeister. „Developments toward the unmanned ship". In: *Proceedings International Symposium Information on Ships (ISIS 2012)* (2012).

[Rei14] Konrad Reif. *Automobilelektronik. Eine Einführung für Ingenieure.* Springer Vieweg, 2014.

[S100] *S-100 Information.* URL: https://www.iho.int/iho_pubs/standard/S-100/S-100_Info.htm (besucht am 14. 09. 2018).

[SAa] Logitech Europe S.A. *C920 HD Pro Webcam.* URL: https://www.logitech.com/de-de/product/hd-pro-webcam-c920#specification-tabular (besucht am 16. 09. 2018).

[SAb] A. Singh und Amit. *Rudder angle indicator.* URL: http://marineinfobox.blogspot.de/2017/05/rudder-angle-indicator-definition.html (besucht am 09. 09. 2018).

[SB08] Alan N Steinberg und Christopher L Bowman. „Revisions to the JDL data fusion model". In: *Handbook of multisensor data fusion.* CRC Press, 2008, S. 65–88.

[Sch+10] Alexander Schatten u. a. *Best Practice Software-Engineering.* Spektrum-Akademischer Vlg, 19. Feb. 2010. ISBN: 978-3-8274-2486-0. URL: https://www.ebook.de/de/product/9469541/alexander_schatten_stefan_biffl_markus_demolsky_erik_gostischa_franta_thomas_oestreicher_best_practice_software_engineering.html.

[Sch16] K. Schlösser. „Ukw-funkzeugnisse src und ubi". In: *Bielefeld: Delius Klasing Verlag* 10 (2016).

[See18] Bundestelle für Seeunfallunteruschung. *Jahresbericht 2017.* Jan. 2018]. URL: https://www.bsu-bund.de/SharedDocs/pdf/DE/Jahresstatistik/Jahresbericht_2017.pdf?__blob=publicationFile&v=3 (besucht am 30. 09. 2018).

[SENC] *system electronic navigational chart.* URL: https://www.spektrum.de/lexikon/kartographie-geomatik/system-electronic-navigational-chart/4793 (besucht am 19. 09. 2018).

[SL06] Andrzej Stateczny und Andrzej Lisaj. „Radar and AIS data fusion for the needs of the maritime navigation". In: *Radar Symposium, 2006. IRS 2006. International.* IEEE. 2006, S. 1–4.

[SL07] Andreas Spillner und Tilo Linz. *Basiswissen Softwaretest : Aus- und Weiterbildung zum Certified Tester.* Bd. 3. 2007.

[SOA] *Data Model Transformation.* URL: http://soapatterns.org/design%5C_patterns/data%5C_model%5C_transformation (besucht am 08. 04. 2018).

[Som12] Ian Sommerville. „Software Engineering". In: *Pearson* 9 (2012). [ISBN 978-3-86894-099-2].

[Spi10] Andreas Spillner. *Basiswissen Softwaretest : Aus- und Weiterbildung zum Certified Tester ; Foundation Level nach ISTQB-Standard.* ger. Heidelberg, 2010.

[Tan17] Peter Tank. „Entwicklung einer sicheren physischen Basisplattform zur echtzeitfähigen Steuerung für die Autonomisierung von hochseetauglichen Schiffen". Magisterarb. Carl von Ossietzky Universität Oldenburg, 2017.

[Ten] Tensorflow. *Tensorflow detection model zoo.* URL: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md (besucht am 16. 09. 2018).

[TR15] Karl-Dieter Tieste und Oliver Romberg. *Keine Panik vor Regelungstechnik!* Springer Fachmedien Wiesbaden, 13. Apr. 2015.

[Tra] *Transas ECDIS Data Services Navi-Planner 4000.* [Letzter Zugriff: 18.09.2018]. URL: http://marinov.com.sg/Transas%20Seminar%20presentation.pdf (besucht am 18. 09. 2018).

[WHW09] Hermann Winner, Stephan Hakuli und Gabriele Wolf. *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort.* Springer-Verlag, 2009.

[Zan09] Justyna Zander-Nowicka. *Model-based testing of real-time embedded systems in the automotive domain.* 2009. URL: https://depositonce.tu-berlin.de/bitstream/11303/2423/1/Dokument_46.pdf (besucht am 28. 09. 2018).

[ZEH] JONAS ZEH. *AUTONOME SCHIFFFAHRT : Fahren die Schiffe bald ohne Steuermann?* URL: http://www.faz.net/aktuell/wirtschaft/massterly-treibt-die-entwicklung-autonomer-schiffe-voran-15533354.html (besucht am 13. 09. 2018).

[Zop+17] Barret Zoph u. a. *AutoML for large scale image classification and object detection.* 2017. URL: https://ai.googleblog.com/2017/11/automl-for-large-scale-image.html (besucht am 16. 09. 2018).

Glossar

Automatisches Identifikationssystem ist ein Funksystem, das durch den Datenaustausch die Schiffsnavigation vereinfacht. 18, 82, 205

Capture Card beschreibt Computerzubehör, das erlaubt Kameras bspw. per HDMI an einen Computer anzuschließen. 126

Controller Area Network beschreibt ein serielles Bussystem. 29, 205

Convention on the International Regulations for Preventing Collisions at Sea ist eine Konvention, die Kollisionen in der Schifffahrt verhindern soll. 17, 205

Cross Track Error beschreibt die Abweichung der aktuellen Position vom gewünschten Pfad. 131, 206

Ecore Ist ein Metamodell zur Beschreibung und Laufzeitunterstützung von Modellen im Eclipse Modeling Framework. 20, 62, 63, 92, 93, 136, 210

Electronic Chart Display and Information System ist ein elektronisches Navigationsinformationssystem. 8, 205

e-Martime Integrated Reference Platform ist eine Entwicklungs- und Testplattform für autonome Schiffe. 8, 48, 205

Extensible Markup Language ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Dateien. 206

HAGGIS ist ein simulatives Testfeld. 20, 48, 49, 65, 91, 92, 93, 136, 154, 155, 165, 166, 167, 169, 170, 171, 175, 188, 243, 245

HTTP ist ein Protokoll zur Übertragung von Daten. 69, 70, 71, 72, 77, 95, 106, 108, 110, 124

International Association of Marine Aids to Navigation and Lighthouse Authorities ist eine internationale Organisation, die sich mit der Seezeichenverwaltung befasst. 122, 205

JSON ist ein in Textform lesbares Datenformat zum Datenaustausch. 71, 72, 76, 89, 99, 100, 101, 103, 104, 108, 110, 112, 124, 135, 139, 144

künstliches neuronales Netz bezeichnet eine miteinander verknüpfte Anzahl von künstlichen Neuronen, die einen funktionalen Zusammenhang bilden. 82

LABSKAUS ist eine physische Testplattform zur Technologiebewertung im Maritimen Bereich. 6, 8, 48, 49, 91, 147

MATLAB ist eine Software zum Lösen mathematischer Probleme und zur grafischen Darstellung der Ergebnisse. 46, 47, 129, 131, 132, 158

MMSI ist eine weltweit gültige Rufnummer des Seefunkdienst. 37, 71, 98, 175

National Marine Electronics Association 0183 ist der Vorgänger des NMEA2000-Standarts. Zum Datenaustausch, vorwiegend in der Schiffahrt verwendet. 28, 29, 206

National Marine Electronics Association 2000 ist der Nachfolger des NMEA0183-Standarts. Zum Datenaustausch, vorwiegend in der Schiffahrt verwendet. 18, 29, 206

OFFIS ist ein Forschungsinstitut, das eng mit der Carl von Ossietzky Universität Oldenburg zusammenarbeitet. 26, 67, 76, 97, 134, 187, 188, 189

OpenCV ist eine quelloffene Programmbibliothek mit Algorithmen für die Bildverarbeitung und maschinelles Sehen. 244

RabbitMQ ist eine Software zum asynchronen Nachrichtenaustausch. 21, 70, 93, 95, 138, 155, 160, 165, 166, 243

RTP ist ein Protokoll zur fortlaufenden Übertragung audiovisueller Daten. 95

S-57 ist der Vorgänger von S-100. 28, 30, 31, 76

S-100 ist ein hydrographischer Datenstandart für die maritime Welt. 28, 62, 63, 92, 93, 136, 141, 203, 211, 212

Simulink ist eine Software zum Modellieren technischer Systeme. 46, 47, 129, 131, 132, 158

Singleton ist ein Entwurfsmuster, das sicherstellt, das nur eine Instanz eines Objekts besteht. 99, 103

Speed over Ground ist die Geschwindigkeit relativ zum Boden. 206

Speed over Ground ist der Kurs relativ zum Boden. 205

TCP ist zuverlässiges Netzwerkprotokoll. 68

Tensorflow ist ein Open-Source-Framework für maschinelles Lernen. 82

Ucore Ist ein Metamodell zur Beschreibung von Modellen, angelehnt an Ecore des Eclipse Modeling Frameworks. 20, 92, 93, 136

UDP ist unzuverlässiges minimales Netzwerkprotokoll. 20, 68, 71, 86, 89, 93, 95, 97, 107, 111, 118, 129, 130, 136, 139, 152, 155

Vessel Traffic Service ist ein elektronisches Überwachungssystem, im maritimen Bereich. 48, 206

WKT ist eine Spezifikation zur textlichen Darstellung von geometrischen Formen auf bspw. einer Seekarte. 76, 109, 114

X in the Loop ist ein Sammelbegriff für verschiedene in the Loop Testmethoden. 47, 206

Abkürzungsverzeichnis

AIS Automatisches Identifikationssystem. *Glossar:* Automatisches Identifikationssystem, 18, 19, 20, 21, 30, 37, 48, 57, 61, 62, 71, 72, 73, 82, 84, 85, 86, 87, 88, 89, 97, 98, 103, 116, 117, 118, 119, 120, 121, 123, 124, 125, 136, 143, 144, 152, 153, 154, 155, 172, 175, 176, 188, 205

CAN Controller Area Network. *Glossar:* Controller Area Network, 29, 205

COG Speed over Ground. *Glossar:* Speed over Ground, 98, 104, 205

COLREGs Convention on the International Regulations for Preventing Collisions at Sea. *Glossar:* Convention on the International Regulations for Preventing Collisions at Sea, 17, 29, 35, 55, 59, 79, 205

ECDIS Electronic Chart Display and Information System. *Glossar:* Electronic Chart Display and Information System, 8, 27, 30, 31, 49, 66, 67, 71, 97, 98, 165, 187, 205

eMIR e-Martime Integrated Reference Platform. *Glossar:* e-Martime Integrated Reference Platform, 8, 48, 205

ENC Electronic Navigational Chart. 28, 30, 31

EPD e-Navigation Prototype Display. 16, 18, 19, 49, 58, 59, 68, 69, 70, 71, 72, 73, 76, 94, 97, 98, 99, 100, 101, 103, 104, 105, 107, 108, 109, 110, 113, 115, 117, 124, 130, 141, 144, 145, 152, 158, 159, 160, 163, 166, 168, 169, 170, 187, 217, 218, 219, 220, 240, 243, 244, 250

GML Geography Markup Language. 31, 76

HiL Hardware in the Loop. 47, 49, 91

IALA International Association of Marine Aids to Navigation and Lighthouse Authorities. *Glossar:* International Association of Marine Aids to Navigation and Lighthouse Authorities, 122, 205

IHO International Hydrographic Organisation. 28, 30, 31

IMO International Maritime Organization. 30, 31, 37

Lidar Light Detection and Ranging. 40, 82, 191

MiL Model in the Loop. 47

NMEA0183 National Marine Electronics Association 0183. *Glossar:* National Marine Electronics Association 0183, 28, 29, 68, 97, 111, 144, 206

NMEA2000 National Marine Electronics Association 2000. *Glossar:* National Marine Electronics Association 2000, 18, 24, 29, 63, 70, 87, 117, 118, 130, 132, 135, 136, 148, 152, 158, 159, 206, 221, 222

PGN Parameter Gruppennummer. 118, 148, 159

RTZ Route plan exchange format. 27, 69, 70, 99, 102, 108, 110, 111, 140, 144, 218, 219, 220, 238, 239, 255, 256, 257, 258

SOG Speed over Ground. *Glossar:* Speed over Ground, 206

ViL Vehicle in the Loop. 47, 48

VTS Vessel Traffic Service. *Glossar:* Vessel Traffic Service, 48, 206

XiL X in the Loop. *Glossar:* X in the Loop, 47, 206

XML Extensible Markup Language. *Glossar:* Extensible Markup Language, 20, 21, 31, 62, 63, 92, 93, 118, 136, 152, 155, 165, 206

XTE Cross Track Error. *Glossar:* Cross Track Error, 131, 132, 148, 206

A. Anhang

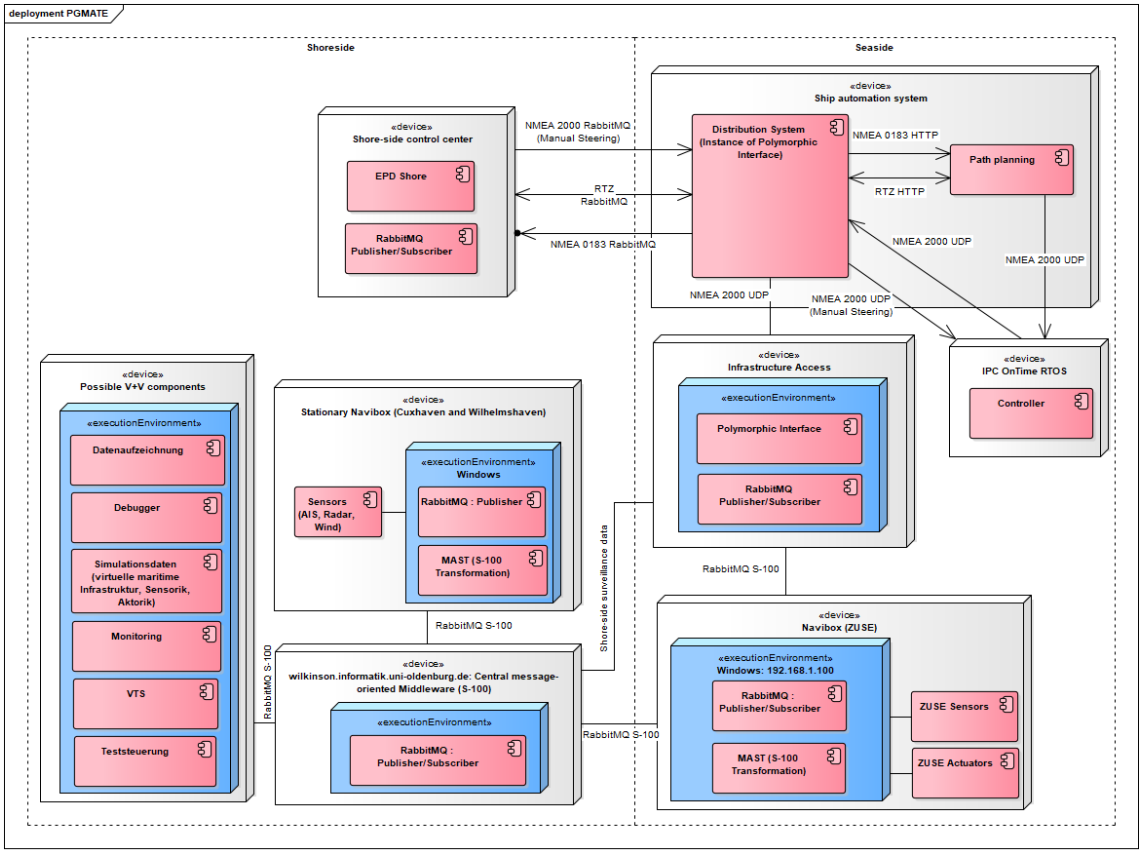


Abbildung A.1.: Gesamtsystem MATE I

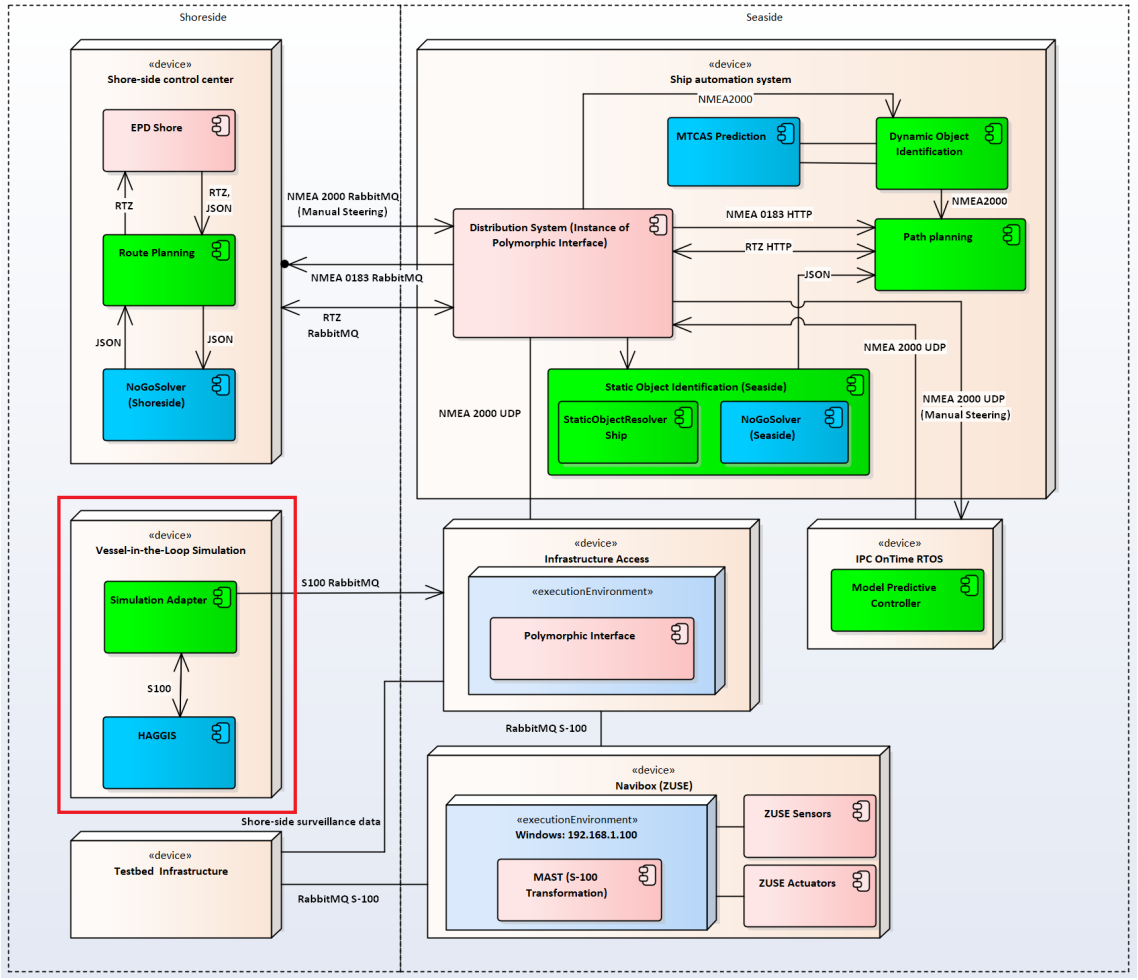


Abbildung A.2.: Einbettung des Simulations-Adapters in die Gesamtarchitektur

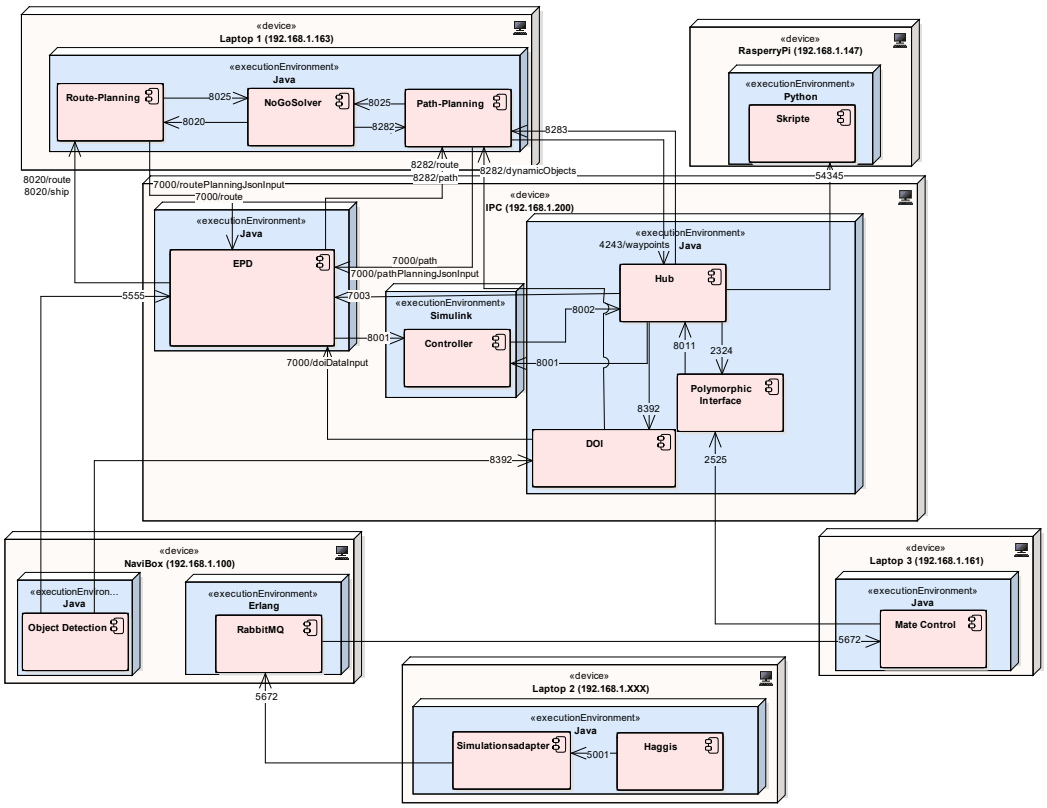


Abbildung A.3.: Aufbau der Verteilung der Komponenten auf den verschiedenen Rechnern auf der Zuse. Angegeben sind auch die verwendeten IPs und die jeweiligen zur Kommunikation benötigten Ports

B. Testfälle

B.1. Modultests

Simulations-Adapter

Test Objectives		
Objective ID	Name	Description
TOb-SA-Internal-1	Kommunikation HAGGIS	Ucore Nachrichten werden von HAGGIS via UDP korrekt vom Simulations-Adapter empfangen.
TOb-SA-Internal-2	Radar Abweichung	Die in HAGGIS nicht vorhandene Radar Ungenauigkeits-Abweichung wird vom Simulations-Adapter simuliert.
TOb-SA-Internal-3	Test des xsl Transformers	Die nicht vorhandene Radar Ungenauigkeits-Abweichung wird vom Simulations-Adapter simuliert.
TOb-SA-Internal-4	Kommunikation via RabbitMQ	Es wird eine Nachricht via RabbitMQ versendet und kann empfangen werden.

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-SA-Internal-1	TOb-SA-Internal-1	Sende und empfange Nachrichten via UDPGZIP und überprüfe, ob sie korrekt sind	Drei String Nachrichten
Test steps	**TC-SA-Internal-1-0** Starte den EntryPoint des Simulationsadapters **TC-SA-Internal-1-1** Starte den UDPGZIPInputAdapterTest		
Expected results	Die gesendeten Elemente entsprechen den empfangenen Elementen		
Actual results			
Tested functions	UDPGZIPInputAdapter recieving, UDPGZIPOutputadapter sending		
Tested compon.	UDPGZIPInputadapter, UDPGZIPOutputadapter		
Test Method		**Pass/Fail**	
JUnit Komponenten-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-SA-Internal-2-0	TOb-SA-Internal-2	Überprüfe, ob nicht gewollte Abweichungen auch keine Änderungen in den longtitude- und latitude-Werten verursachen	Ein Test S100-Ecore-Radar String
Test steps	**TC-SA-Internal-2-0-0** Führe den RadarDeviationTest aus		
Expected results	keine Abweichungen, bei einer Abweichung von 0 Metern		
Actual results	Java-Rundungsfehler in manchen Testausführungen		
Tested functions	RadarDeviation		
Tested compon.	RadarDeviation		
Test Method		**Pass/Fail**	
JUnit Komponenten-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-SA-Internal-2-1	TOb-SA-Internal-2	Überprüfe, ob die gewollten Abweichungen Änderungen in den longtitude- und latitude-Werten verursachen	Ein Test S100-eCore-Radar String
Test steps	**TC-SA-Internal-2-1-0** Führe den RadarDeviationTest aus		
Expected results	Abweichungen, bei gewollten Abweichungen		
Actual results	es wird eine Ungenauigkeit berechnet		
Tested functions	RadarDeviation		
Tested compon.	RadarDeviation		
Test Method		**Pass/Fail**	
JUnit Komponenten-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-SA-Internal-3-0	TOb-SA-Internal-3	Überprüfe, ob eine Ucore-S100-Nachricht in die gewollten S100-Ecore Nachrichten transformiert werden	S100-Ucore Nachricht als String, S-100 Ecore static-AIS-Nachricht als String, S-100 Ecore dynamic-AIS-Nachricht als String, S-100 Ecore Radar-Nachricht als String, S-100 Ecore Positions-Nachricht als String
Test steps	**TC-SA-Internal-3-0-0** Führe den TransformerTest aus		
Expected results	Die Ucore-S100-Nachricht wird in die gewollten Ecore Nachrichten Transformiert und entspricht dem gewollten S100-Ecore Schema		
Actual results	Der Timestamp wird aktualisiert, sonst entspricht die erwartete Nachricht, der transformierten Nachricht		
Tested functions	Transformer.transform		
Tested compon.	Transform		
Test Method		**Pass/Fail**	
JUnit Komponenten-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-SA-Internal-3-1	TOb-SA-Internal-3	Überprüfe, ob die S100-Ecore Nachrichten serialisiert werden	S-100 Ecore static-AIS-Nachricht als String, S-100 Ecore dynamic-AIS-Nachricht als String, S-100 Ecore Radar-Nachricht als String, S-100 Ecore Positions-Nachricht als String
Test steps	**TC-SA-Internal-3-1-0** Führe den TransformerTest aus		
Expected results	Die Ecore-S100-Nachrichten werden Serialisiert, ohne einen Fehler zu werfen und sind anschließend Bytes		
Actual results	Die Nachrichten werden zu Bytes serialisiert		
Tested functions	Transformer.serialize		
Tested compon.	Transformer		
Test Method		**Pass/Fail**	
JUnit Komponenten-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-SA-Internal-4-0	TOb-SA-Internal-4	Überprüfe, ob eine S100-Ucore Nachricht transformiert, serialisiert, via RabbitMQ verschickt und empfangen wird	S-100 Ucore Nachricht als String, S-100 Ecore static-AIS-Nachricht als String, S-100 Ecore dynamic-AIS-Nachricht als String, S-100 Ecore Radar-Nachricht als String, S-100 Ecore Positions-Nachricht als String
Test steps	**TC-SA-Internal-3-1-0** Wähle die lokale RabbitMQ Konfiguration in der RabbitMQConfig-Klasse **TC-SA-Internal-3-1-0** Führe den SimulationAdapterTest aus		
Expected results	Die S100-Ucore Nachricht wird transformiert, serialisiert, via RabbitMQ verschickt und empfangen. Die empfangenen S-100 Ecore Nachrichten entsprechen den erwarteten S-100 Ecore Nachrichten		
Actual results	Die deserialisierten S100-Ecore Nachrichten entsprachen den erwarteten S 100-Ecore Nachrichten, es gab aber leichte Änderungen in der Formatierung.		
Tested functions	SimulationAdapter.onMessage		
Tested compon.	SimulationAdapter, Transformer, RabbitMQConfig, RadarDeviation		
Test Method		**Pass/Fail**	
JUnit Komponenten-Test		Fail	

Dynamische Objektidentifizierung

Test Objectives		
Objective ID	Name	Description
TOb-DOI-Assoziation-1	Assoziation	AIS- und Radarnachrichten mit gleicher Position werden von der Dynamischen Objektidentifizierung korrekt assoziiert.
TOb-DOI-Assoziation-2	Assoziation	AIS- und Radarnachrichten mit einem Abstand von bis zu 70 Meter werden von der Dynamischen Objektidentifizierung korrekt assoziiert.
TOb-DOI-Assoziation-3	Assoziation	Nicht zusammengehörige AIS- und Radarnachrichten (Position zu weit auseinander) werden von der Dynamischen Objektidentifizierung nicht assoziiert.
TOb-DOI-Assoziation-4	Assoziation	Nicht zusammengehörige AIS- und Radarnachrichten (gleiche Position aber zu unterschiedlichen Zeitpunkten) werden von der Dynamischen Objektidentifizierung nicht assoziiert.
TOb-DOI-Assoziation-5	Assoziation	Radartracks werden von der Dynamischen Objektidentifizierung nicht mehrfach assoziiert.
TOb-DOI-Assoziation-6	Assoziation	AIS-Tracks werden von der Dynamischen Objektidentifizierung nicht mehrfach assoziiert.

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-DOI-Assoziation-1	TOb-DOI-Assoziation-1	Sende AIS- und Radarnachrichten des selben Schiffes mit exakt den selben Werten für Position, Geschwindigkeit und Kurs an die Dynamic Object Identification und überprüfe, ob erkannt wird, dass es sich um das selbe Schiff handelt.	AIS- und Radarnachrichten in NMEA2000
Test steps	**TC-DOI-Assoziation-1-1** Eingabe der Daten in HAGGIS. **TC-DOI-Assoziation-1-2** Starte Polymorphic-Interface, Hub und Dynamic Object Identification. **TC-DOI-Assoziation-1-3** Verschicke Radar- und AIS-Nachrichten mithilfe von HAGGIS. **TC-DOI-Assoziation-1-4** Ausgabe der Assoziationsliste.		
Expected results	Die IDs der AIS- und Radarnachrichten werden als ein Schiff assoziiert.		
Actual results	Die IDs der AIS- und Radarnachrichten wurden als ein Schiff assoziiert.		
Tested functions	Assoziation der Datenfusion		
Tested compon.	Dynamic Object Identification		
Test Method		**Pass/Fail**	
Spezifikationsbasiertes Testen		pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-DOI-Assoziation-2	TOb-DOI-Assoziation-1	Sende AIS- und Radarnachrichten des selben Schiffes mit einem Unterschied von 69 Metern in den einzelnen befahrenen Positionen an die Dynamic Object Identification und überprüfe, ob erkannt wird, dass es sich um das selbe Schiff handelt.	AIS- und Radarnachrichten in NMEA2000
Test steps	**TC-DOI-Assoziation-2-1** Eingabe der Daten in HAGGIS. **TC-DOI-Assoziation-2-2** Starte Polymorphic-Interface, Hub und Dynamic Object Identification. **TC-DOI-Assoziation-2-3** Verschicke Radar- und AIS-Nachrichten mithilfe von HAGGIS. **TC-DOI-Assoziation-2-4** Ausgabe der Assoziationsliste.		
Expected results	Die IDs der AIS- und Radarnachrichten werden als ein Schiff assoziiert.		
Actual results	Die IDs der AIS- und Radarnachrichten wurden als ein Schiff assoziiert.		
Tested functions	Assoziation der Datenfusion		
Tested compon.	Dynamic Object Identification		
Test Method		**Pass/Fail**	
Spezifikationsbasiertes Testen		pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-DOI-Assoziation-3	TOb-DOI-Assoziation-1	Sende AIS- und Radarnachrichten von unterschiedlichen Schiffen, die in den einzelnen Positionen jeweils 71 Meter unterschied haben, an die Dynamic Object Identification und überprüfe, ob erkannt wird, dass es sich nicht um das selbe Schiff handelt.	AIS- und Radarnachrichten in NMEA2000
Test steps	**TC-DOI-Assoziation-7-1** Eingabe der Daten in HAGGIS. **TC-DOI-Assoziation-7-2** Starte Polymorphic-Interface, Hub und Dynamic Object Identification. **TC-DOI-Assoziation-7-3** Verschicke Radar- und AIS-Nachrichten mithilfe von HAGGIS. **TC-DOI-Assoziation-7-4** Ausgabe der Assoziationsliste.		
Expected results	Die IDs der AIS- und Radarnachrichten werden nicht als ein Schiff assoziiert.		
Actual results	Die IDs der AIS- und Radarnachrichten wurden nicht als ein Schiff assoziiert.		
Tested functions	Assoziation der Datenfusion		
Tested compon.	Dynamic Object Identification		
Test Method		**Pass/Fail**	
Spezifikationsbasiertes Testen		pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-DOI-Assoziation-4	TOb-DOI-Assoziation-1	Sende Radar-Nachrichten von einem Schiff und AIS-Nachrichten von einem anderen Schiff, die die selben Positionen abfahren, aber zu unterschiedlichen Zeitpunkten, an die DOI und überprüfe, ob erkannt wird, dass die Nachrichten nicht zu dem selben Schiff gehören.	AIS- und Radarnachrichten in NMEA2000
Test steps	**TC-DOI-Assoziation-5-1** Eingabe der Daten in HAGGIS. **TC-DOI-Assoziation-5-2** Starte Polymorphic-Interface, Hub und Dynamic Object Identification. **TC-DOI-Assoziation-5-3** Verschicke Radar- und AIS-Nachrichten mithilfe von HAGGIS. **TC-DOI-Assoziation-5-4** Ausgabe der Assoziationsliste.		
Expected results	Die IDs der AIS- und Radarnachrichten werden nicht als ein Schiff assoziiert.		
Actual results	Die IDs der AIS- und Radarnachrichten werden nicht als ein Schiff assoziiert.		
Tested functions	Assoziation der Datenfusion		
Tested compon.	Dynamic Object Identification		
Test Method		**Pass/Fail**	
Spezifikationsbasiertes Testen		pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-DOI-Assoziation-5	TOb-DOI-Assoziation-1	Sende Radar-Nachrichten von einem Schiff und AIS-Nachrichten von zwei Schiffen (mit zwei unterschiedlichen IDs) an die Dynamic Object Identification und überprüfe, ob erkannt wird, dass die ID der Radar-Nachrichten und eine ID der zwei IDs der AIS-Nachrichten zu dem selben Schiff gehören.	AIS- und Radarnachrichten in NMEA2000
Test steps	**TC-DOI-Assoziation-5-1** Eingabe der Daten in HAGGIS. **TC-DOI-Assoziation-5-2** Starte Polymorphic-Interface, Hub und Dynamic Object Identification. **TC-DOI-Assoziation-5-3** Verschicke Radar- und AIS-Nachrichten mithilfe von HAGGIS. **TC-DOI-Assoziation-5-4** Ausgabe der Assoziationsliste.		
Expected results	Die ID der Radarnachrichten und der zugehörigen ID der AIS-Nachrichten werden als ein Schiff assoziiert. Die ID der anderen AIS-Nachrichten wird nicht assoziiert.		
Actual results	Die ID der Radarnachrichten und der zugehörigen ID der AIS-Nachrichten werden als ein Schiff assoziiert. Die ID der anderen AIS-Nachrichten wird nicht assoziiert.		
Tested functions	Assoziation der Datenfusion		
Tested compon.	Dynamic Object Identification		
Test Method		**Pass/Fail**	
Spezifikationsbasiertes Testen		pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-DOI-Assoziation-6	TOb-DOI-Assoziation-1	Sende AIS-Nachrichten von einem Schiff und Radarnachrichten von zwei Schiffen (mit zwei unterschiedlichen IDs) an die Dynamic Object Identification und überprüfe, ob erkannt wird, dass die ID der AIS-Nachrichten und eine ID der zwei IDs der Radarnachrichten zu dem selben Schiff gehören.	AIS- und Radarnachrichten in NMEA2000
Test steps	**TC-DOI-Assoziation-6-1** Eingabe der Daten in HAGGIS. **TC-DOI-Assoziation-6-2** Starte Polymorphic-Interface, Hub und Dynamic Object Identification. **TC-DOI-Assoziation-6-3** Verschicke Radar- und AIS-Nachrichten mithilfe von HAGGIS. **TC-DOI-Assoziation-6-4** Ausgabe der Assoziationsliste.		
Expected results	Die ID der AIS-Nachrichten und der zugehörigen ID der Radarnachrichten werden als ein Schiff assoziiert. Die ID der anderen Radar-Nachricht wird nicht assoziiert.		
Actual results	Die ID der AIS-Nachrichten und der zugehörigen ID der Radarnachrichten werden als ein Schiff assoziiert. Die ID der anderen Radar-Nachricht wird nicht assoziiert.		
Tested functions	Assoziation der Datenfusion		
Tested compon.	Dynamic Object Identification		
Test Method		**Pass/Fail**	
Spezifikationsbasiertes Testen		pass	

Routenplanung

Route Planing Komponententests

Test Objectives		
Objective ID	Name	Description
TOb-RP-Internal-1	Schiffsparameter-Update	Bei Empfang neuer Schiffsparamter, werden die alten zwischengespeicherten ersetzt.
TOb-RP-Internal-2	Routen-Akkumulation	Solange keine Schiffsparameter zur Verfügung stehen, werden alle bisher empfangenen Routen akkumuliert.
TOb-RP-Internal-3	Routen-Teilung	Routen mit mehr als zwei Wegpunkten werden in korrekte Teilrouten zerlegt.
TOb-RP-Internal-4	Grid-Blockierung	Die Zellen des Grids werden ordnungsgemäß entprechen der NoGoAreas blockiert.
TOb-RP-Internal-5	Teilrouten-Berechnung	Teilrouten können und werden vollständig und unabhängig zueinander errechnet.
TOb-RP-Internal-6	Routen-Berechnung	Eventuelle Teilrouten werden ordnungsgemäß wieder zusammen zu einer gesamt Route vereint. Routen sind immer valide und befahrbar.

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Internal-1	TOb-RP-Internal-1	Bei Empfang neuer Schiffsparameter, werden die alten zwischengespeicherten ersetzt.	Schiffsparameter in JSON
Test steps	**TC-RP-Internal-1-0** Starte die Route-Planning-Komponente. **TC-RP-Internal-1-1** Starte die EPD-Shore. **TC-RP-Internal-1-2** Stelle die Schiffsparameter in der EPD-Shore ein und drücke auf versenden. **TC-RP-Internal-1-3** Stelle die neuen Schiffsparameter in der EPD-Shore ein und drücke auf versenden. **TC-RP-Internal-1-3** Überprüfe die aktuellen Schiffsparameter-Belegung in der Route-Planning-Komponente mittels des Loggers.		
Expected results	Die aktuellen Schiffsparameter entsprechen den zuletzt von der EPD-Shore gesendeten Schiffsparametern.		
Actual results	Die aktuellen Schiffsparameter entsprechen den zuletzt von der EPD-Shore gesendeten Schiffsparametern.		
Tested functions	Schiffsparameter-Update		
Tested compon.	Route-Planning, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test.		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Internal-2	TOb-RP-Internal-2	Solange keine Schiffsparameter zur Verfügung stehen, werden alle bisher empfangenen Routen akkumuliert.	Routen im RTZ-Format, Schiffsparameter und Obstacles in JSON
Test steps	**TC-RP-Internal-2-0** Starte den NoGoSolver. **TC-RP-Internal-2-1** Starte die Route-Planning-Komponente. **TC-RP-Internal-2-2** Starte die EPD-Shore. **TC-RP-Internal-2-3** Konfiguriere eine Route in der EPD-Shore und sende sie an die Route-Planning-Komponente. **TC-RP-Internal-2-4** Wiederhole dies mit einer weiteren Route. **TC-RP-Internal-2-5** Stelle die neuen Schiffsparameter in der EPD-Shore ein und drücke auf versenden.		
Expected results	Beide Routen werden einzeln nach gewisser Zeit in fertig berechneter Form in der EPD-Shore angezeigt.		
Actual results	Beide Routen werden einzeln nach gewisser Zeit in fertig berechneter Form in der EPD-Shore angezeigt.		
Tested functions	Routen-Akkumulation		
Tested compon.	Route-Planning, EPD-Shore, NoGoSolver		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Internal-3	TOb-RP-Internal-3	Routen mit mehr als zwei Wegpunkten werden in korrekte Teilrouten zerlegt.	Routen im RTZ-Format, Schiffsparameter und Obstacles in JSON
Test steps	**TC-RP-Internal-3-0** Starte den NoGoSolver. **TC-RP-Internal-3-1** Starte die Route-Planning-Komponente. **TC-RP-Internal-3-2** Starte die EPD-Shore. **TC-RP-Internal-3-3** Konfiguriere eine Route in der EPD-Shore mit mehr als zwei Wegpunkten und sende sie an die Route-Planning-Komponente. **TC-RP-Internal-3-4** Vergleiche den Zielpunkt der ersten berechneten (Teil-)Route in der Route-Planning-Komponente mit dem Startpunkt der zweiten berechneten (Teil-)Route mittels des Loggers.		
Expected results	Der erste Zielpunkt entspricht dem zweiten Startpunkt.		
Actual results	Der erste Zielpunkt entspricht dem zweiten Startpunkt.		
Tested functions	Routen-Teilung		
Tested compon.	Route-Planning, EPD-Shore, NoGoSolver		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Internal-4	TOb-RP-Internal-4	Die Zellen des Grids werden ordnungsgemäß entprechen der NoGoAreas blockiert. data	PLEASE SET
Test steps	**TC-RP-Internal-4-0** Starte den NoGoSolver. **TC-RP-Internal-4-1** Starte die Route-Planning-Komponente. **TC-RP-Internal-4-2** Starte die EPD-Shore. **TC-RP-Internal-4-3** Konfiguriere eine Route in der EPD-Shore und sende sie an die Route-Planning-Komponente. **TC-RP-Internal-4-4** Manipuliere die setObstacles-Methode der RouteCalculation-Klasse dahingehend, dass anstatt der berechneten Route die blockierten Zellen als eine Route sowie alle NoGoAreas als Routen an die EPD-Shore gesendet werden.		
Expected results	Die NoGoAreas und die blockierten Zellen sind deckungsgleich.		
Actual results	Die NoGoAreas und die blockierten Zellen sind deckungsgleich.		
Tested functions	Grid-Blockierung		
Tested compon.	Route-Planning, EPD-Shore, NoGoSolver		
Test Method		**Pass/Fail**	
Whitebox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Internal-5	TOb-RP-Internal-5	Teilrouten können und werden vollständig und unabhängig zueinander errechnet.	Routen im RTZ-Format, Schiffsparameter in JSON
Test steps	**TC-RP-Internal-5-0** Starte den NoGoSolver. **TC-RP-Internal-5-1** Starte die Route-Planning-Komponente. **TC-RP-Internal-5-2** Starte die EPD-Shore. **TC-RP-Internal-5-3** Konfiguriere eine Route in der EPD-Shore und sende sie an die Route-Planning-Komponente. **TC-RP-Internal-5-4** Manipuliere die setObstacles-Methode der RouteCalculation-Klasse dahingehend, dass nach der Berechnung der ersten Teilroute die Routen-Berechnung abgeschlossen ist.		
Expected results	Nur die erste Teilroute wird in der EPD-Shore angezeigt.		
Actual results	Nur die erste Teilroute wird in der EPD-Shore angezeigt.		
Tested functions	Teilrouten-Berechnung		
Tested compon.	Route-Planning, EPD-Shore, NoGoSolver		
Test Method		**Pass/Fail**	
Blackbox-Test, Whitebox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Internal-6	TOb-RP-Internal-6	Eventuelle Teilrouten werden ordnungsgemäß wieder zusammen zu einer gesamt Route vereint. Routen sind immer valide und befahrbar.	Routen im RTZ-Format, Schiffsparameter in JSON
Test steps	**TC-RP-Internal-6-0** Starte den NoGoSolver. **TC-RP-Internal-6-1** Starte die Route-Planning-Komponente. **TC-RP-Internal-6-2** Starte die EPD-Shore. **TC-RP-Internal-6-3** Konfiguriere eine Route in der EPD-Shore mit mehr als zwei Wegpunkten und sende sie an die Route-Planning-Komponente.		
Expected results	Die in der EPD-Shore angezeigte Route ist vollständig und valide.		
Actual results	Die in der EPD-Shore angezeigte Route ist vollständig und valide.		
Tested functions	Routen-Berechnung		
Tested compon.	Route-Planning, EPD-Shore, NoGoSolver		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

PID-Controller

Test Objectives		
Objective ID	Name	Description
TOb-Cont-Internal-1	Kommunikation NMEA2000	NMEA2000-Nachrichten werden vom Controller korrekt geparst und versendet
TOb-Cont-internal-2	Heading-Controller	Der Heading-Controller berechnet korrekte Werte.
TOb-Cont-Internal-3	Speed-Controller	Der Speed-Controller berechnet korrekte Werte.
TOb-Cont-Internal-4	Manual-Steering	Überbrückung des Autopiloten bei Verwendung der virtual Handles auf der EPD.
TOb-Cont-Sim-1	Controller-Simulation	Das Controller-Model wird in die Simulation eingebunden.
TOb-Cont-Integration-1	Controller-Integrationstests	Integriere den Controller in bestehende weitere Komponenten
TOb-Cont-Zuse-1	Zuse on Shore	Überprüfe die Kommunikation der realen Hardwarekomponenten mit dem Controller
TOb-Cont-Zuse-2	Zuse off Shore	Abschlusstest des Controllers im realen Testbed

Kommunikation NMEA2000

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Internal-1	TOb-Cont-Internal-1	Sende Heading-Nachrichten in NMEA2000 an den Controller und überprüfe, ob diese korrekt interpretiert werden.	Heading in NMEA2000
Test steps	**TC-Cont-Internal-1-0** Starte die Hub-Instanz des Polymorphic-Interfaces. **TC-Cont-Internal-1-1** Stelle in der Controller-Testschnittstelle das Senden von Heading-Nachrichten mit bestimmten Werten ein. **TC-Cont-Internal-1-2** Stelle im Controller das Logging der Heading-Werte nach dem NMEA2000-Parser ein und starte den Controller. **TC-Cont-Internal-1-3** Starte die Controller-Testschnittstelle.		
Expected results	Die gesendeten Werte entsprechen den geloggten Werten nach dem Parsing.		
Actual results	Die gesendeten Werte entsprechen den geloggten Werten nach dem Parsing.		
Tested functions	NMEA2000 Parser		
Tested compon.	Controller, Hub		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test.		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Internal-2	TOb-Cont-Internal-1	Sende Positions-Nachrichten in NMEA2000 an den Controller und überprüfe, ob diese korrekt interpretiert werden.	Positionsdaten in NMEA2000
Test steps	**TC-Cont-Internal-2-0** Starte die Hub-Instanz des Polymorphic-Interfaces **TC-Cont-Internal-2-1** Stelle in der Controller-Testschnittstelle das Senden von Positions-Nachrichten mit bestimmten Werten ein. **TC-Cont-Internal-2-2** Stelle im Controller das Logging der Positions-Werte nach dem NMEA2000-Parser ein und starte den Controller. **TC-Cont-Internal-2-3** Starte die Controller-Testschnittstelle		
Expected results	Die gesendeten Werte entsprechen den geloggten Werten nach dem Parsing.		
Actual results	Die gesendeten Werte entsprechen den geloggten Werten nach dem Parsing.		
Tested functions	NMEA2000 Parser		
Tested compon.	Controller, Hub		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Internal-3	TOb-Cont-Internal-1	Sende Speed-Nachrichten in NMEA2000 an den Controller und überprüfe, ob diese korrekt interpretiert werden.	Speed in NMEA2000
Test steps	**TC-Cont-Internal-3-0** Starte die Hub-Instanz des Polymorphic-Interfaces **TC-Cont-Internal-3-1** Stelle in der Controller-Testschnittstelle das Senden von Speed-Nachrichten mit bestimmten Werten ein. **TC-Cont-Internal-3-2** Stelle im Controller das Logging der Speed-Werte nach dem NMEA2000-Parser ein und starte den Controller. **TC-Cont-Internal-3-3** Starte die Controller-Testschnittstelle		
Expected results	Die gesendeten Werte entsprechen den geloggten Werten nach dem Parsing.		
Actual results	Die gesendeten Werte entsprechen den geloggten Werten nach dem Parsing.		
Tested functions	NMEA2000 Parser		
Tested compon.	Controller, Hub		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Internal-4	TOb-Cont-Internal-1	Sende RPM-Nachrichten aus dem Controller und überprüfe ob diese korrekte Werte enthalten.	Konstante Testwerte für RPM
Test steps	**TC-Cont-Internal-4-0** Starte die Hub-Instanz des Polymorphic-Interfaces **TC-Cont-Internal-4-1** Stelle in der Controller-Testschnittstelle das Empfangen von NMEA2000-Nachrichten ein. **TC-Cont-Internal-4-2** Starte die Controller-Testschnittstelle **TC-Cont-Internal-4-3** Stelle im Controller als Eingabe des ParserInterpreter konstante RPM-Testwerte ein. **TC-Cont-Internal-4-4** Starte den Controller.		
Expected results	Die gesendeten Werte des Controllers entsprechen den Empfangengen der Testschnittstelle		
Actual results			
Tested functions	NMEA2000 Interpreter		
Tested compon.	Controller, Hub		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Internal-5	TOb-Cont-Internal-1	Sende Rudder-Nachrichten aus dem Controller und überprüfe ob diese korrekte Werte enthalten.	Konstante Testwerte für Rudder
Test steps	**TC-Cont-Internal-5-0** Starte die Hub-Instanz des Polymorphic-Interfaces **TC-Cont-Internal-5-1** Stelle in der Controller-Testschnittstelle das Empfangen von NMEA2000-Nachrichten ein. **TC-Cont-Internal-5-2** Starte die Controller-Testschnittstelle **TC-Cont-Internal-5-3** Stelle im Controller als Eingabe des ParserInterpreter konstante Rudder-Testwerte ein. **TC-Cont-Internal-5-4** Starte den Controller.		
Expected results	Die gesendeten Werte des Controllers entsprechen den Empfangengen der Testschnittstelle.		
Actual results	Die gesendeten Werte des Controllers entsprechen den Empfangengen der Testschnittstelle.		
Tested functions	NMEA2000 Interpreter		
Tested compon.	Controller, Hub		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test		Pass	

Heading-Controller

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Internal-6	TOb-Cont-Internal-2	Manuelles Einspeisen von Testwerten ind die Berechnung und Verifikation der Ergebnisse durch manuelles Prüfen	Menge an Testdaten (Waypoint 1, Waypoint 2, Position)
Test steps	**TC-Cont-Internal-6-0** Manuelle Berechnung des Bearings und des Cross-Track-Errors mit entsprechender Formel. **TC-Cont-Internal-6-1** Einspeisung der Testdaten in den Simulink Block durch Konstanten. Logging des Ergebnisses **TC-Cont-Internal-6-2** Verifikation des Ergebnisses durch vergleichen des manuell berechneten Wertes und der Ergebnisse aus dem Block.		
Expected results	Manuelle Berechnung stimmt mit dem Resultat des Blockes überein.		
Actual results			
Tested functions	Block: Bearing to dest.		
Tested compon.	Controller: Autopilot		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Internal-7	TOb-Cont-Internal-2	Manuelles Einspeisen von Testwerten in den PID-Controller. Überprüfen der Ausgaben	Menge an Testdaten (XTE, Bearing)
Test steps	**TC-Cont-Internal-7-0** Bereitstellung mehrere Testsets aus (XTE, Bearing) mit verschiedenen erwarteten Regelungsantworten (Kurssetpoint links oder rechts vom aktuellen Heading des Schiffes). **TC-Cont-Internal-7-1** Einspeisung der Testdaten in den Simulink Block durch Konstanten. Logging des Ergebnisses. **TC-Cont-Internal-7-2** Verifikation des Ergebnisses durch vergleichen des erwarteten Wertes und der Ergebnisse aus dem Block.		
Expected results	Erwartetes Resultat stimmt mit dem Resultat des Blockes überein.		
Actual results			
Tested functions	Block: LOS-Controller		
Tested compon.	Controller: Autopilot		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Internal-8	TOb-Cont-Internal-2	Manuelles Einspeisen von Testwerten in den PID-Controller. Überprüfen der Ausgaben	Menge an Testdaten (desired course, Heading)
Test steps	**TC-Cont-Internal-8-0** Bereitstellung mehrerer Testsets aus (desired course, Heading) mit verschiedenen erwarteten Regelungsantworten (Desired Course links oder rechts vom aktuellen Heading des Schiffes). **TC-Cont-Internal-8-1** Einspeisung der Testdaten in den Simulink Block durch Konstanten. Logging des Ergebnisses. **TC-Cont-Internal-8-2** Verifikation des Ergebnisses durch vergleichen des erwarteten Wertes und der Ergebnisse aus dem Block.		
Expected results	Erwartetes Resultat stimmt mit dem Resultat des Blockes überein.		
Actual results			
Tested functions	Block: Rudder-Controller		
Tested compon.	Controller: Autopilot		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

Speed-Controller

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Internal-9	TOb-Cont-Internal-3	Setzte Eingabewerte der Geschwindigkeit für den Controller manuell, überprüfe Reaktion des Controllers auf diese Werte	Feste Eingabewerte an entsprechnenden Input Schnittstelle
Test steps	**TC-Cont-Internal-8-0** Setzte Eingabewerte an Eingängen des Controllers **TC-Cont-Internal-8-1** Überprüfe Sprungantwort, vergleiche diese mit erwarteter Antwort		
Expected results	Erwartetes Resultat stimmt mit dem Resultat des Blockes überein.		
Actual results			
Tested functions			
Tested compon.	Controller: Speedcontroller		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Internal-10	TOb-Cont-Internal-3	Setze Eingabewerte des Controllers auf Ausgabewerte des Virtuellen Testbeds. Überprüfe Antwort des Controllers auf die Eingabewerte der Simulation	Ausgabewerte des virtuellen Testfelds.
Test steps	**TC-Cont-Internal-9-0** Verbinde Controller mit virtuellem Testbed **TC-Cont-Internal-9-1** Starte Simulation und Controller **TC-Cont-Internal-9-2** Überwache Reaktion des Controllers auf die generierten Eingabewerte		
Expected results	Der Controller reagiert angemessen auf die von der Simulation generierten Eingabewerte.		
Actual results			
Tested functions			
Tested compon.	Controller: Speedcontroller		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

Manual Steering

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Internal-11	TOb-Cont-Internal-4	Konstante, manuelle Rudder und Speed Commands im Controller anlegen und Überbrückung des Autopiloten testen.	Rudder und Speed Command (Manual Steering)
Test steps	**TC-Cont-Internal-10-0** Anlegen der konstanten Rudder und Speed Commands an den Controller **TC-Cont-Internal-10-1** tarten der Simulation. **TC-Cont-Internal-10-2** Überprüfen ob der Controller den Autopilot entsprechend überbrückt.		
Expected results	Die Ausgabe des Controllers entspricht den gesendeten Rudder und Speed-Commands.		
Actual results			
Tested functions	Block: ForwardRPMRudderCommand		
Tested compon.	Controller: Main		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

Simulationstests

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Sim-1	TOb-Cont-Sim-1	Nutze das virtuelle Testbed um Wegpunkte in verschiedenen Himmelsrichtungen anzufahren	Acht verschiedene Himmelsrichtungen innerhalb des virtuellen Testbeds
Test steps	**TC-Cont-Sim-1-0** Verbinde Controller mit virtuellem Testbed **TC-Cont-Sim-1-1** Stelle verschiedene Himmelsrichtungen im Virtuellen Testbed ein **TC-Cont-Sim-1-2** Starte Simulation		
Expected results	Das simulierte Schiff nähert sich dem aktuellen Wegpunkt bis auf X Metern an.		
Actual results			
Tested functions	Heading- und RPM-Berechnung		
Tested compon.	Controller		
Test Method		**Pass/Fail**	
Model-in-the-Loop Test			

B.2. Komponententests

PID-Controller

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Component-1	TC-Cont-Component-1	Überprüfen des inneren Zustands mithilfe aufgezeichneter Daten	Aufgezeichnete Daten der Navibox
Test steps	**TC-Cont-Component-1-0** Starten des Polymorphic Interface **TC-Cont-Component-1-1** Starten des RabbitMQs **TC-Cont-Component-1-2** Starten MateControl **TC-Cont-Component-1-3** Starten des Controllers **TC-Cont-Component-1-4** Starten des Navibox-Simulators **TC-Cont-Component-1-5** Inneren Zustand des Controllers mithilfe der Matlab Simulink Werkzeuge		
Expected results	Schiffstatus (Heading, Geschwindigkeit etc) des Controllers entspricht den aufgezeichneten Daten		
Actual results			
Tested functions			
Tested compon.			
Test Method		**Pass/Fail**	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Component-2	TC-Cont-Component-2	Überprüfen der allgemeinen Funktionstüchtigkeit des Controllers mithilfe einer 3D-Simulation (Brine)	Ein- und Ausgaben der 3D-Simulation (Brine)
Test steps	**TC-Cont-Component-2-0** Matjes Handles starten **TC-Cont-Component-2-1** BrineMaster starten **TC-Cont-Component-2-2** Controller Debug Adapter starten **TC-Cont-Component-2-3** Brine3D starten **TC-Cont-Component-2-4** Controller konfigurieren **TC-Cont-Component-2-5** Controller starten **TC-Cont-Component-2-6** Reaktion des Controllers, sowohl mithilfe der Matlab Simulink Werkzeuge, als auch in der Simulation beobachten		
Expected results	Das simulierte Schiff folgt dem definierten Pfad		
Actual results			
Tested functions			
Tested compon.			
Test Method		**Pass/Fail**	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Component-3	TC-Cont-Component-3	Überprüfen der Funktionstüchtigkeit des Controllers im realen Testfeld (LABSKAUS)	Alle physikalischen Einflüsse der Umwelt auf die Zuse
Test steps	**TC-Cont-Component-3-** Navibox starten **TC-Cont-Component-3-** Konfiguration der benötigten Systeme **TC-Cont-Component-3-** Netzwerkkommunikation herstellen **TC-Cont-Component-3-** Konfiguration des Controllers **TC-Cont-Component-3-** Starten des Controllers **TC-Cont-Component-3-** Starten der Steuerskripte **TC-Cont-Component-3-** Reaktion des Controllers auf die Einflüsse der Umwelt beobachten **TC-Cont-Component-3-** Eventuell Parameter Einstellungen vornehmen, anschließend erneutes Testen		
Expected results	Das regelungssystem ist in der Lage trotz der Einflüsse der Umwelt dem gewünschten Pfad zu folgen.		
Actual results			
Tested functions			
Tested compon.			
Test Method		**Pass/Fail**	

B.3. Integrationstests

Simulations-Adapter

Test Objectives		
Objective ID	Name	Description
TOb-SA-Integration-1	Kommunikation HAGGIS	Ucore Nachrichten werden von HAGGIS korrekt versendet und vom Simulations-Adapter empfangen.
TOb-SA-Integration-2	Kommunikation Polymorphic Interface	Ecore Nachrichten werden vom Simulations-Adapter korrekt versendet und können mithilfe des Polymorphen Interfaces empfangen werden.

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-SA-Integration-1	TOb-SA-Integration-1	Empfange Nachrichten via UDP-GZIP und überprüfe, ob sie korrekt sind	Drei String Nachrichten
Test steps	**TC-SA-Integration-1-0** Starte den EntryPoint des Simulationsadapters **TC-SA-Integration-1-1** Starte HAGGIS **TC-SA-Integration-1-2** Starte das Szenario in HAGGIS		
Expected results	Die gesendeten Elemente aus HAGGIS entsprechen den empfangenen Elementen		
Actual results	Die gesendeten Elemente aus HAGGIS konnten vom Simulationsadapter empfangen werden und entsprachen den gesendeten Elementen.		
Tested functions	UDPGZIPInputAdapter recieving		
Tested compon.	UDPGZIPInputadapter, HAGGIS TransportHandler		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-SA-Integration-2	TOb-SA-Integration-2	Empfange Nachrichten via UDP-GZIP und überprüfe, ob sie korrekt sind	Drei String Nachrichten
Test steps	**TC-SA-Integration-2-0** Starte den EntryPoint des Simulationsadapters **TC-SA-Integration-2-1** Starte HAGGIS **TC-SA-Integration-2-1** Starte das Szenario in HAGGIS **TC-SA-Integration-2-2** Starte den HUB (Polymorphic Interface Instance)		
Expected results	Die gesendeten Elemente können über den HUB abgerufen werden		
Actual results	Die mit dem Simulationsadapter über RMQ versendeten Elemente konnten über den HUB abgerufen werden.		
Tested functions	Korrektes Versenden der Ecore Nachrichten über RMQ		
Tested compon.	HUB, HAGGIS, Simulationsadapter RMQOutput		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Dynamische Objektidentifikation

Test Objectives		
Objective ID	Name	Description
TOb-DOI-Integration-1	Hub-Entgegennahme	Die relevanten Nachrichten seitens Hub sind entgegennehmbar und werden vorverarbeitet.
TOb-DOI-Integration-2	Schiffsparameter Versand EPD	Die relevanten Schiffsparameter zu den fusionierten Schiffen sowie den reinkommenden Schiffen werden korrekt an die EPD gesandt.
TOb-DOI-Integration-3	Schiffsparameter Versand Path Planning	Die relevanten Schiffsparameter zu den fusionierten Schiffen werden korrekt an das Path Planning gesandt.
TOb-DOI-Integration-4	Kamerainformation Versand und Entgegennahme	Die mittels Kamera ermittelte Schiffsposition wird korrekt an die Dynamische Objektidentifikation gesendet und von dieser entgegengenommen.

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-DOI-Integration-1	TOb-DOI-Integration-1	Die relevanten Nachrichten seitens Hub sind entgegennehmbar und werden vorverarbeitet.	AIS-, Radar-, eigene Position- und eigener Kurs-Nachrichten in NMEA2000
Test steps	**TC-DOI-Integration-1-0** Starte die RabbitMQ Verbindung zum Wilkinson Server. **TC-DOI-Integration-1-1** Starte das Polymorphic Interface. **TC-DOI-Integration-1-2** Starte den Hub (Polymorphic Interface Instance. **TC-DOI-Integration-1-3** Starte die Dynamic Object Identification.		
Expected results	Die Logger-Ausgabe der Dynamic Object Identification-Komponente enthält für jede empfangene benötigte Nachricht die Ausgabe "PGN: xxx " mit der jeweiligen PGN.		
Actual results	Die Logger-Ausgabe der Dynamic Object Identification-Komponente enthält für jede empfangene benötigte Nachricht die Ausgabe "PGN: xxx " mit der jeweiligen PGN.		
Tested functions	relevante AIS-, Radar-, Position- und Kurs-Nachrichten-Entgegennahme		
Tested compon.	Polymorphic Interface, Polymorphic Interface Instance, Dynamic Object Identification		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-DOI-Integration-2	TOb-DOI-Integration-2	Die relevanten Schiffsparameter zu den fusionierten Schiffen sowie den reinkommenden Schiffen werden korrekt an die EPD gesandt.	Json-Objekte mit Arraylisten die alle relevanten Schiffsparameter zu allen erkannten Schiffen beinhalten
Test steps	**TC-DOI-Integration-2-0** Starte die RabbitMQ Verbindung zum Wilkinson Server. **TC-DOI-Integration-2-1** Starte das Polymorphic Interface. **TC-DOI-Integration-2-2** Starte den Hub (Polymorphic Interface Instance. **TC-DOI-Integration-2-3** Starte die Dynamic Object Identification. **TC-DOI-Integration-2-4** Starte die EPD.		
Expected results	In der Ausgabe der Dynamic Object Identification wird deutlich, dass die Schiffsinformationen versendet wurden und in der EPD werden die Schiffe visuell angezeigt.		
Actual results	In der Ausgabe der Dynamic Object Identification wird deutlich, dass die Schiffsinformationen versendet wurden und in der EPD werden die Schiffe visuell angezeigt.		
Tested functions	Versenden der relevanten Schiffsparameter zu allen erkannten Schiffen		
Tested compon.	Polymorphic Interface, Polymorphic Interface Instance, Dynamic Object Identification, EPD		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-DOI-Integration-3	TOb-DOI-Integration-3	Die relevanten Schiffsparameter zu den fusionierten Schiffen sowie den reinkommenden Schiffen werden korrekt an das Path Planning gesandt.	Json-Objekte mit Arraylisten die alle relevanten Schiffsparameter zu allen fusionierten Schiffen beinhalten
Test steps	**TC-DOI-Integration-3-0** Starte die RabbitMQ Verbindung zum Wilkinson Server. **TC-DOI-Integration-3-1** Starte das Polymorphic Interface. **TC-DOI-Integration-3-2** Starte den Hub (Polymorphic Interface Instance. **TC-DOI-Integration-3-3** Starte die Dynamic Object Identification. **TC-DOI-Integration-3-4** Starte das Path Planning.		
Expected results	In der Ausgabe der Dynamic Object Identification wird deutlich, dass die Schiffsinformationen versendet wurden und in der Path Planning Komponente werden diese Informationen entgegengenommen.		
Actual results	In der Ausgabe der Dynamic Object Identification wird deutlich, dass die Schiffsinformationen versendet wurden und in der Path Planning Komponente werden diese Informationen entgegengenommen.		
Tested functions	Versenden der relevanten Schiffsparameter zu allen fusionierten Schiffen		
Tested compon.	Polymorphic Interface, Polymorphic Interface Instance, Dynamic Object Identification, Path Planning		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-DOI-Integration-4	TOb-DOI-Integration-4	Die mittels Kamera erkannten Schiffspositionen werden von der Object Detection an die Dynamic Object Identification gesandt und von dieser entgegengenommen	Schiffsposition (Left, Right, Both, Nothing)
Test steps	**TC-DOI-Integration-4-0** Starte die Object Detection. **TC-DOI-Integration-4-1** Wähle die Kamera aus. **TC-DOI-Integration-4-2** Starte die Dynamic Object Identification.		
Expected results	Die Object Detection hat in der Logger-Ausgabe Einträge der Form "Sending detected Ship to DOI " und die Dynamic Object Identification hat Einträge der Form "Camerainput ".		
Actual results	Die Object Detection hat in der Logger-Ausgabe Einträge der Form "Sending detected Ship to DOI " und die Dynamic Object Identification hat Einträge der Form "Camerainput ".		
Tested functions	Versenden der Schiffsposition der mittels Kamera erkannten Schiffe sowie Entgegennahme dieser		
Tested compon.	Object Detection, Dynamic Object Identification		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

EPD

EPD Integrationstests

Test Objectives		
Objective ID	Name	Description
TOb-EPD-Integration-1	Schiffsparameter-Senden1	Die Schiffsparameter in JSON seitens der EPD sind an die Routen-Planung übertragbar.
TOb-EPD-Integration-2	Schiffsparameter-Senden2	Die Schiffsparameter in JSON seitens der EPD sind an die Pfad-Planung übertragbar.
TOb-EPD-Integration-3	Routen-Senden1	Eine Route in RTZ seitens EPD ist an die Routen-Planung übertragbar.
TOb-EPD-Integration-4	Routen-Senden2	Eine Route in RTZ seitens EPD ist an die Pfad-Planung übertragbar.
TOb-EPD-Integration-5	Routen-Empfang1	Eine Route in RTZ seitens Routen-Planung ist von der EPD empfangbar, speicherbar und anzeigbar.
TOb-EPD-Integration-6	Routen-Empfang2	Eine Route in RTZ seitens Pfad-Planung ist von der EPD empfangbar, speicherbar und anzeigbar.
TOb-EPD-Integration-7	Debug-Informationen-Empfang und Darstellung - DOI	Die von der DOI gesendeten AIS-, Radar- und Fusionsergebnisse werden von der EPD empfangen und angezeigt.
TOb-EPD-Integration-8	Debug-Informationen-Empfang und Darstellung - RP	Die von der Routen-Planung gesendeten Polygone werden von der EPD empfangen und angezeigt.
TOb-EPD-Integration-9	Debug-Informationen-Empfang und Darstellung - PP	Die von der Pfad-Planung gesendeten Polygone werden von der EPD empfangen und angezeigt.

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-EPD-Integration-1	TOb-EPD-Integration-1	Schiffsparameter-Senden1Die Schiffsparameter in JSON seitens der EPD sind an die Routen-Planung übertragbar.	Schiffsparameter als JSON-String
Test steps	**TC-EPD-Integration-1-0** Starte die Route-Planning-Komponente. **TC-EPD-Integration-1-1** Starte die EPD-Shore. **TC-EPD-Integration-1-2** Konfiguriere die Schiffsparameter in der EPD-Shore und drücke auf versenden.		
Expected results	Die Logger-Ausgabe der EPD zeigt an, dass Schiffsparameter im als JSON-String gesendet wurden.		
Actual results	Die Logger-Ausgabe der EPD zeigt an, dass Schiffsparameter im als JSON-String gesendet wurden.		
Tested functions	Schiffsparameter-Senden1		
Tested compon.	Routen-Planung, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Integration-2	TOb-EPD-Integration-2	Schiffsparameter-Senden1Die Schiffsparameter in JSON seitens der EPD sind an die Pfad-Planung übertragbar.	Schiffsparameter als JSON-String
Test steps	**TC-EPD-Integration-2-0** Starte die Pfad-Planungs-Komponente. **TC-EPD-Integration-2-1** Starte die EPD-Shore. **TC-EPD-Integration-2-2** Konfiguriere die Schiffsparameter in der EPD-Shore und drücke auf versenden.		
Expected results	Die Logger-Ausgabe der EPD zeigt an, dass Schiffsparameter im als JSON-String gesendet wurden.		
Actual results	Die Logger-Ausgabe der EPD zeigt an, dass eine Schiffsparameter im als JSON-String gesendet wurden.		
Tested functions	Schiffsparameter-Senden2		
Tested compon.	Pfad-Planung, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-EPD-Integration-3	TOb-EPD-Integration-3	Eine Route in RTZ seitens EPD ist an die Routen-Planung übertragbar.	Route im RTZ-Format
Test steps	**TC-EPD-Integration-3-0** Starte die Routen-Planungs-Komponente. **TC-EPD-Integration-3-1** Starte die EPD-Shore. **TC-EPD-Integration-3-2** Wähle eine beliebige Route aus und klicke auf "Route optimieren "		
Expected results	Die Logger-Ausgabe der EPD zeigt an, dass eine Route im RTZ-Format gesendet wurde.		
Actual results	Die Logger-Ausgabe der EPD zeigt an, dass eine Route im RTZ-Format gesendet wurde.		
Tested functions	Routen-Senden1		
Tested compon.	Routen-Planung, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-EPD-Integration-4	TOb-EPD-Integration-4	Eine Route in RTZ seitens EPD ist an die Pfad-Planung übertragbar.	Route im RTZ-Format
Test steps	**TC-EPD-Integration-4-0** Starte die Pfad-Planungs-Komponente. **TC-EPD-Integration-4-1** Starte die EPD-Shore. **TC-EPD-Integration-4-2** Wähle eine beliebige Route aus und klicke auf "Route starten "		
Expected results	Die Logger-Ausgabe der EPD zeigt an, dass eine Route im RTZ-Format gesendet wurde.		
Actual results	Die Logger-Ausgabe der EPD zeigt an, dass eine Route im RTZ-Format gesendet wurde.		
Tested functions	Routen-Senden2		
Tested compon.	Pfad-Planung, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-EPD-Integration-5	TOb-EPD-Integration-5	Eine Route in RTZ seitens Routen-Planung ist von der EPD empfangbar, speicherbar und anzeigbar.	Route im RTZ-Format
Test steps	**TC-EPD-Integration-5-0** Starte die Routen-Planungs-Komponente. **TC-EPD-Integration-5-1** Starte die EPD-Shore. **TC-EPD-Integration-5-2** Wähle eine beliebige Route aus und klicke auf "Route optimieren "		
Expected results	Die Logger-Ausgabe der EPD zeigt an, dass eine Route im RTZ-Format gesendet wurde, im Anschluss wird angezeigt, dass eine neue Route empfangen und gespeichert wurde. In der EPD wird eine neue Route im RouteManager-Plugin angezeigt.		
Actual results	Die Logger-Ausgabe der EPD zeigt an, dass eine Route im RTZ-Format gesendet wurde, im Anschluss wird angezeigt, dass eine neue Route empfangen und gespeichert wurde. In der EPD wird eine neue Route im RouteManager-Plugin angezeigt.		
Tested functions	Routen-Empfang1		
Tested compon.	Routen-Planung, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-EPD-Integration-6	TOb-EPD-Integration-6	Eine Route in RTZ seitens Pfad-Planung ist von der EPD empfangbar, speicherbar und anzeigbar.	Route im RTZ-Format
Test steps	**TC-EPD-Integration-6-0** Starte die Pfad-Planning-Komponente. **TC-EPD-Integration-6-1** Starte die EPD-Shore. **TC-EPD-Integration-6-2** Wähle eine beliebige Route aus und klicke auf "Route starten "		
Expected results	Die Logger-Ausgabe der EPD zeigt an, dass eine Route im RTZ-Format gesendet wurde, im Anschluss wird angezeigt, dass eine neue Route empfangen und gespeichert wurde. In der EPD wird eine neue Route im RouteManager-Plugin angezeigt.		
Actual results	Die Logger-Ausgabe der EPD zeigt an, dass eine Route im RTZ-Format gesendet wurde, im Anschluss wird angezeigt, dass eine neue Route empfangen und gespeichert wurde. In der EPD wird eine neue Route im RouteManager-Plugin angezeigt.		
Tested functions	Routen-Empfang2		
Tested compon.	Routen-Planung, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-EPD-Integration-7	TOb-EPD-Integration-7	Debug-Informationen-Empfang und Darstellung - DOIDie von der DOI gesendeten AIS-, Radar- und Fusionsergebnisse werden von der EPD empfangen und angezeigt.	NMEA2000-Daten als JSON-String
Test steps	**TC-EPD-Integration-7-0** Starte die DOI-Komponente. **TC-EPD-Integration-7-1** Starte die EPD-Shore. **TC-EPD-Integration-7-2** Aktiviere den DOIDataLayer		
Expected results	Die Logger-Ausgabe der EPD zeigt an, von der DOI kontinuierlich JSON-Daten empfangen und gespeichert werden. In der EPD-Karte werden Schiffe in grün, rot und lila angezeigt.		
Actual results	Die Logger-Ausgabe der EPD zeigt an, von der DOI kontinuierlich JSON-Daten empfangen und gespeichert werden. In der EPD-Karte werden Schiffe in grün, rot und lila angezeigt.		
Tested functions	Debug-Informationen-Empfang und Darstellung - DOI		
Tested compon.	DOI, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-EPD-Integration-8	TOb-EPD-Integration-8	Die von der Routen-Planung gesendeten Polygone werden von der EPD empfangen und angezeigt.	Polygone als JSON-String
Test steps	**TC-EPD-Integration-8-0** Starte die Routen-Komponente. **TC-EPD-Integration-8-1** Starte die EPD-Shore. **TC-EPD-Integration-8-2** Aktiviere den SOIDataLayer		
Expected results	Die Logger-Ausgabe der EPD zeigt an, von der Routen-Planung kontinuierlich JSON-Daten empfangen und gespeichert werden. In der EPD-Karte werden Polygone der statischen Hindernisse sowie Grids (blockierte und nicht blockierte Zellen) dargestellt.		
Actual results	Die Logger-Ausgabe der EPD zeigt an, von der Routen-Planung kontinuierlich JSON-Daten empfangen und gespeichert werden. In der EPD-Karte werden Polygone der statischen Hindernisse sowie Grids (blockierte und nicht blockierte Zellen) dargestellt.		
Tested functions	Debug-Informationen-Empfang und Darstellung - RP		
Tested compon.	Routen-Planung, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-EPD-Integration-9	TOb-EPD-Integration-9	Die von der Pfad-Planung gesendeten Polygone werden von der EPD empfangen und angezeigt.	Polygone als JSON-String
Test steps	**TC-EPD-Integration-9-0** Starte die Pfad-Komponente. **TC-EPD-Integration-9-1** Starte die EPD-Shore. **TC-EPD-Integration-9-2** Aktiviere den PathPlanningDataLayer		
Expected results	Die Logger-Ausgabe der EPD zeigt an, von der Pfad-Planung kontinuierlich JSON-Daten empfangen und gespeichert werden. In der EPD-Karte werden Polygone der statischen Hindernisse sowie GoodWin-Shapes dargestellt.		
Actual results	Die Logger-Ausgabe der EPD zeigt an, von der Pfad-Planung kontinuierlich JSON-Daten empfangen und gespeichert werden. In der EPD-Karte werden Polygone der statischen Hindernisse sowie GoodWin-Shapes dargestellt.		
Tested functions	Debug-Informationen-Empfang und Darstellung - PP		
Tested compon.	Pfad-Planung, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Route-Planing Komponente

Route Planing Integrationstests

Test Objectives		
Objective ID	Name	Description
TOb-RP-Integration-1	Schiffsparameter-Entgegennahme	Die Schiffsparameter in JSON seitens EPD-Shore sind entgegennehmbar und werden zwischengespeichert.
TOb-RP-Integration-2	Routen-Entgegennahme	Eine Route in RTZ seitens EPD ist entgegennehmbar und wird zwischengespeichert.
TOb-RP-Integration-3	NoGoSolver-Anfrage	Bei vorliegen von Schiffsparametern und einer oder mehreren zu berechnenden Routen, wird der NoGoSolver ordnungsgemäß angefragt.
TOb-RP-Integration-4	NoGoSolver-Entgegennahme	Die vom NoGoSolver berechneten NoGoAreas werden entgegen genommen und korrekt geparst.
TOb-RP-Integration-5	Routen-Versand	Vollständig berechnete Routen werden zur EPD versandt.

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Integration-1	TOb-RP-Integration-1	Die Schiffsparameter in JSON seitens EPD-Shore sind entgegennehmbar und werden zwischengespeichert.	Schiffsparameter in JSON
Test steps	**TC-RP-Integration-1-0** Starte die Route-Planning-Komponente. **TC-RP-Integration-1-1** Starte die EPD-Shore. **TC-RP-Integration-1-2** Konfiguriere die Schiffsparameter in der EPD-Shore und drücke auf versenden.		
Expected results	Die Logger-Ausgabe der Route-Planning-Komponente enthält die Ausgabe "New shipparameters: " sowie die übergebenen Schiffsparameter.		
Actual results	Die Logger-Ausgabe der Route-Planning-Komponente enthält die Ausgabe "New shipparameters: " sowie die übergebenen Schiffsparameter.		
Tested functions	Schiffsparameter-Entgegennahme		
Tested compon.	Route-Planning, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Integration-2	RP-Integration-2	Eine Route in RTZ seitens EPD ist entgegennehmbar und wird zwischengespeichert.	Routen im RTZ-Format
Test steps	**TC-RP-Integration-2-0** Starte die Route-Planning-Komponente. **TC-RP-Integration-2-1** Starte die EPD-Shore. **TC-RP-Integration-2-2** Konfiguriere eine Route in der EPD-Shore und sende sie an die Route-Planning-Komponente.		
Expected results	Die Logger-Ausgabe der Route-Planning-Komponente enthält die Ausgabe "New active route " sowie die übergebene Route.		
Actual results	Die Logger-Ausgabe der Route-Planning-Komponente enthält die Ausgabe "New active route " sowie die übergebene Route.		
Tested functions	Routen-Entgegennahme.		
Tested compon.	Route-Planning, EPD-Shore		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Integration-3	TC-RP-Integration-3	Bei vorliegen von Schiffsparametern und einer oder mehreren zu berechnenden Routen, wird der NoGoSolver ordnungsgemäß angefragt.	Routen im RTZ-Format, Schiffsparameter in JSON
Test steps	**TC-RP-Integration-3-0** Starte die Route-Planning-Komponente. **TC-RP-Integration-3-1** Starte die EPD-Shore. **TC-RP-Integration-3-2** Konfiguriere eine Route in der EPD-Shore und sende sie an die Route-Planning-Komponente.		
Expected results	In der Ausgabe des NoGoSolvers ist die Ausgabe "Now generated Json. " zu finden.		
Actual results	In der Ausgabe des NoGoSolvers ist die Ausgabe "Now generated Json. " zu finden.		
Tested functions	NoGoSolver-Anfrage		
Tested compon.	Route-Planning, EPD-Shore, NoGoSolver		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Integration-4	TC-RP-Integration-4	Die vom NoGoSolver berechneten NoGoAreas werden entgegen genommen und korrekt geparst.	Routen im RTZ-Format, NoGoAreas und Schiffsparameter in JSON
Test steps	**TC-RP-Integration-4-0** Starte den NoGoSolver. **TC-RP-Integration-4-1** Starte die Route-Planning-Komponente. **TC-RP-Integration-4-2** Starte die EPD-Shore. **TC-RP-Integration-4-3** Konfiguriere eine Route und die Schiffsparameter in der EPD-Shore und sende sie an die Route-Planning-Komponente.		
Expected results	Die Logger-Ausgabe der Route-Planning-Komponente enthält die Ausgabe "New obstacles: " sowie die vom NoGoSolver übergebenen Obstacles.		
Actual results	Die Logger-Ausgabe der Route-Planning-Komponente enthält die Ausgabe "New obstacles: " sowie die vom NoGoSolver übergebenen Obstacles.		
Tested functions	NoGoSolver-Entgegennahme		
Tested compon.	Route-Planning, EPD-Shore, NoGoSolver		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-RP-Integration-5	TC-RP-Integration-5	Vollständig berechnete Routen werden zur EPD versandt.	Routen im RTZ-Format, Schiffsparameter in JSON
Test steps	**TC-RP-Integration-5-0** Starte den NoGoSolver. **TC-RP-Integration-5-1** Starte die Route-Planning-Komponente. **TC-RP-Integration-5-2** Starte die EPD-Shore. **TC-RP-Integration-5-3** Konfiguriere eine Route und die Schiffsparameter in der EPD-Shoree und sende sie an die Route-Planning-Komponente.		
Expected results	In der EPD-Shore wird nach einer gewissen Berechnungszeit eine valide Route angezeigt.		
Actual results	In der EPD-Shore wird nach einer gewissen Berechnungszeit eine valide Route angezeigt.		
Tested functions	Routen-Versand		
Tested compon.	Route-Planning, EPD-Shore, NoGoSolver		
Test Method		**Pass/Fail**	
Blackbox-Test		Pass	

PID-Controller

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Integration-1	TOb-Cont-Integration-1	Überprüfe Verlustfreie Kommunikation zwischen Navibox und Controller	Reale Navibox-Aufnahmen
Test steps	**TC-Cont-Integration-1-0** Stelle im Controller das Logging der geparsten NMEA2000-Werte für Position, Heading und Speed ein. **TC-Cont-Integration-1-1** Starte Polymorphic-Interface und Hub. **TC-Cont-Integration-1-2** Starte den S100-Logger **TC-Cont-Integration-1-3** Starte den Controller **TC-Cont-Integration-1-4** Starte den Navibox-Simulator		
Expected results	Die Werte des S100-Loggers entsprechen den geloggten Werten die vom Controller empfangen werden.		
Actual results			
Tested functions			
Tested compon.	Controller, Hub, Polymorphic Interface		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Integration-2	TOb-Cont-Integration-1	Testen des Manual Steerings mithilfe der EPD	Benutzereingabe: Manuelles Steuern
Test steps	**TC-Cont-Integration-2-0** Starten des Controllers, der EPD, des Distributionsystems. **TC-Cont-Integration-2-1** Starten der VirtualHandles. Veränderung der Geschwindigkeit und des Ruderwinkels **TC-Cont-Integration-2-2** Logging der Ausgaben des Controllers.		
Expected results	Der Controller gibt die Befehle der Virtual Handles an das Schiff raus.		
Actual results			
Tested functions	Virtual Handles, Controller: RPMRudderCommandForward		
Tested compon.	Controller, EPD, Distribution System		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Integration-3	TOb-Cont-Integration-1	Überprüfe die Kommunikation zwischen Pathplanning und Controller	Routendaten
Test steps	**TC-Cont-Integration-3-0** Verschicke Routendaten mithilfe der Pathplanning Komponente **TC-Cont-Integration-3-1** Empfange die Routendaten mithilfe des Controllers **TC-Cont-Integration-3-2** Vergleiche die verschickten Daten mit den empfangenen Daten		
Expected results	Die von der Pathplanning Komponente verschickten Daten stimmen mit den vom Controller empfangenen Daten überein.		
Actual results			
Tested functions			
Tested compon.	Pathplanning, Controller		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

Zuse on Shore

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Zuse-1	TOb-Cont-Zuse-1	Überprüfe Verlustfreie Kommunikation zwischen Navibox und Controller	Reale Naviboxdaten
Test steps	**TC-Cont-Zuse-1-0** Stelle im Controller das Logging der geparsten NMEA2000-Werte für Position, Heading und Speed ein. **TC-Cont-Zuse-1-1** Starte Polymorphic-Interface und Hub. **TC-Cont-Zuse-1-2** Starte den S100-Logger **TC-Cont-Zuse-1-3** Starte den Controller **TC-Cont-Zuse-1-4** Starte den Navibox-Simulator		
Expected results	Die Werte des S100-Loggers entsprechen den geloggten Werten die vom Controller empfangen werden.		
Actual results			
Tested functions			
Tested compon.	Controller, Hub, Polymorphic Interface		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

Zuse off Shore

Test Case			
Test case ID	Test objectives	Test scenario	Test/Input data
TC-Cont-Zuse-OffShore-1	TOb-Cont-Zuse-OffShore-1	Abschlusstest des Controllers im realen Testbed (Zuse)	Alle durch die Umwelt entstehenden Einflüsse auf die Zuse.
Test steps	**TC-Cont-Zuse-OffShore-1-0** Starte Zuse. **TC-Cont-Zuse-OffShore-1-1** Beobachte das Verhalten der Zuse im Wasser.		
Expected results	Die Zuse folgt selbstständig einem vorgegebenen Kurs, fährt vorher spezifiezierte Wegpunkte mit einer genauigkeit von X Metern an.		
Actual results			
Tested functions			
Tested compon.	Controller		
Test Method		**Pass/Fail**	
Spezifikationsbas. Blackbox-Test			

C. Anleitungen

C.1. Simulationsadapter Anleitungen und Konfigurationen

In diesem Kapitel werden die Vorbereitung und die genauen Schritte zur Ausführung des Simulationsadapters beschrieben.
Um den Simulationsadapter nutzen zu können müssen HAGGIS, ein RabbitMQ-Server und ein Consumer eingerichtet werden. Anleitungen dazu sind bei den jeweiligen Komponenten zu finden.

Simulationsadapter

Die Einstellungen des Simulationsadapters sind in der ReadMe genau beschrieben.

HAGGIS

Um die Schnittstelle zum Simulationsadapter nutzen zu können muss in HAGGIS der ProtocolHandler UCoreXMLProtocolHandler eingebunden sein. Dieser muss in der Simulation entsprechend der im Simulationsadapter vorgenommenen Einstellungen (Voreingestellt ist der Port 5001) konfiguriert werden.

C.2. Route-Planing Anleitung

In diesem Kapitel werden die Vorbereitung und die genauen Schritte zur Ausführung der Route-Planing-Komponente beschrieben.
Um die Route-Planing-Komponente nutzen zu können, muss zunächst eine Konfigurationsdatei als Programm Argument übergeben werden. Hierzu kann die Beispiel-Datei unter dem Pfad "src

\main \resources \configuration.properties" verwendet werden. Zudem müssen die EPD und der NoGoSolver gestartet werden. Anleitungen dazu sind bei den jeweiligen Komponenten dazu zu finden.

Beispielhafter Ablauf für Routen-Berechnung

Sind Route-Planing-Komponente, EPD und NoGoSolver erfolgreich gestartet kann zunächst eine beliebige Route über den Routen-Manager in der EPD erstellt werden. Diese kann nun über den Button "Route senden" an die Route-Planing-Komponente gesendet werden. Diese nimmt die Route entgegen und berechnet eine neue valide Route, wobei sie automatisch den NoGoSolver anfragt. Ist die Berechnung abgeschlossen wird ebenfalls automatisch die neue valide Route an die EPD zurückgesendet. Diese kann dann über den Routen-Manager ausgewählt und auf der Karte der EPD visualisiert werden. Je nach Größe der erstellten Route und Version des NoGoSolvers kann dieser Ablauf mehrere Minuten in Anspruch nehmen.

C.3. Pfad-Planung Anleitung

In diesem Kapitel werden die Vorbereitung und die genauen Schritte zur Ausführung der Pfad-Planungs-Komponente beschrieben.
Um die Pfad-Planungs-Komponente nutzen zu können, muss zunächst eine Konfigurationsdatei als Programm Argument übergeben werden. Hierzu kann die Beispiel-Datei unter dem Pfad "src \main \resources \configuration.properties" verwendet werden. Zudem müssen die EPD , die Dynamische-Objektidentifizierung, der Hub und der NoGoSolver gestartet werden. Anleitungen dazu sind bei den jeweiligen Komponenten dazu zu finden.

C.4. Objektdetektion Anleitung und Konfiguration

In diesem Abschnitt werden die genauen Schritte und Konfigurationsmöglichkeiten zur Ausführung der Objektdetektions-Komponente beschrieben.
Um die Objektdetektions-Komponente nutzen zu können, muss die OpenCV-Bibliothek eingebunden werden. Hierfür kann wie folgt vorgegangen werden:
`medium.com/@aadimator/how-to-set-up-opencv-in-intellij-idea-6eb103c1d45c`. Die benötigten OpenCV Dateien liegen im Resources-Ordner unter dem Pfad „`src/main/resources`“.

Unter src/main/resources befindet sich auch die Konfigurationsdatei (configurations.properties) In dieser Konfigurationsdatei können neben den Schnittstellenkonfigurationen, auch ausgewählt werden, ob und wenn ja wie viele Aufnahmen zur Auswertung gespeichert werden sollen. Die gespeicherten Daten sind dann unter C://recordedData zu finden. Des Weiteren kann die Position des eigenen Schiffes in dem Bildmaterial angegeben werden. Wenn das eigene Schiff auf den Ausschnitt nicht zu sehen ist, wird 1.00 angegeben.
Alle Einstellungen der Objektdetektion sind in der ReadMe genau beschrieben.

C.5. Dynamische Objektidentifizierung Anleitung und Konfiguration

In diesem Abschnitt werden die wichtigsten Konfigurationsmöglichkeiten zur Ausführung der Dynamischen Objektidentifizierungs-Komponente beschrieben.
Unter src/main/ressources befindet sich die Konfigurationsdatei (configurations.properties).
Es können alle Schnittstellenkonfigurationen vorgenommen werden. Zudem kann hier ausgewählt werden, welche Daten die Komponente verarbeitet:

- *real*: reale Daten von der stationären Navibox und Daten von der Navibox auf der Zuse
- *simulated*: simulierten Daten aus HAGGIS
- *both*: Daten von allen Quellen

Alle Einstellungen der Dynamischen Objektidentifzierung sind in der ReadMe genau beschrieben.

C.6. EPD in Betrieb nehmen

Der PG MATE II wurde vom Offis eine neue EPD zur Verfügung gestellt. Im Folgenden wird beschrieben, welche Schritte nötig sind, um sie starten zu können.

1. Hinterlegen der settings.xml (siehe Dependency Problem: Konfigurieren der settings.xml) im eigenen Benutzerverzeichnis (${lokale Festplattenpartition}/Users/${Benutzername}/.m2/)
   a) wird für den Zugriff auf die Offis-Maven-Repository-Server benötigt
2. Speichern des privaten Schlüssels vom Offis als lokalen privaten Schlüssel (z. B. durch Ersetzen des bisherigen (id_rsa) vom Uni-Gitlab, in dem Fall muss der Public Key (extrahierbar aus der Putty-Datei vom Offis) im Uni-Gitlab hinterlegt werden!

   a) dadurch wird erst das Pullen/Pushen in das Offis-Repository ermöglicht
3. Klonen der drei Offis-Repositories
   a) git clone ssh://${VTECH_USER}@vscm.offis.de/var/lib/gitroot/emir/emirmodel.git
   b) git clone ssh://${VTECH_USER}@vscm.offis.de/var/lib/gitroot/emir/emirruntime.git
   c) git clone ssh://${VTECH_USER}@vscm.offis.de/var/lib/gitroot/emir/emirepd.git
   d) git clone git@gitlab.uni-oldenburg.de:mate2/EPDMate-Plugins.git
4. Öffnen des Repository „emirepd“ mit Eclipse (das Offis nutzt Eclipse-Entwicklungsmodule, daher ist Eclipse hier die erste Wahl)
   a) Rechte Maustaste → Import → Maven → Existing Maven Projects → das „emirepd“-Repository auswählen
   b) nun werden alle Unterprojekte geladen, welches mit „Finish“ bestätigt werden muss
5. Öffnen des Repository „emirruntime“ mit Eclipse
   a) Rechte Maustaste → Import → Maven → Existing Maven Projects → das „emirruntime“-Repository auswählen
   b) nun werden alle Unterprojekte geladen, welches mit „Finish“ bestätigt werden muss
6. Öffnen des Repository „emirmodel“ mit Eclipse
   a) Rechte Maustaste → Import → Maven → Existing Maven Projects → das „emirmodel“-Repository auswählen
   b) nun werden alle Unterprojekte geladen, welches mit „Finish“ bestätigt werden muss
7. Öffnen des Repository „EPDMate-Plugins“ mit Eclipse
   a) Rechte Maustaste → Import → Maven → Existing Maven Projects → das „PDMate-Plugins“-Repository auswählen → „EPDMate“ - Projekt auswählen!
   b) nun werden alle Unterprojekte geladen, welches mit „Finish“ bestätigt werden muss
8. wenn keine Ordner mehr rot sind (das Bauen des Workspaces kann ein wenig dauern), kann die EPD gestartet werden
   a) dazu das Projekt „EPDShoreProduct“ öffnen und in die Hauptklasse unter „src“ navigieren, diese anschließend ausführen
9. beim ersten Start müssen die Plugins der EPD konfiguriert werden (diese besteht quasi vollständig aus einzelnen Plugins)
   a) dazu im sich öffnen Fenster zunächst die pom.xml des Projektes „EPDModules“ via + hinzufügen
   b) analog dazu die pom.xml des Projektes „EPDCore“ hinzufügen
   c) Und auch die pom.xml des Projektes „EPDMate“!
10. Anschließend via „Apply&Exit“ die Konfiguration übernehmen (Die EPD sollte nun auch automatisch neustarten.)

11. Nun sollte sich die EPD öffnen. Unter „Layout“ → „Open View...“ können Views ergänzt werden (z. B. die Map, Routes o.ä.)

Falls die Einstellungen der EPD zurückgesetzt werden sollen, in dem Benutzerverzeichnis eurers PCs den EPDShoreProduct-Ordner löschen.

D. Interface Control Document

Dynamische Objektidentifizierung

Datenströme

Eingehend:

Atribut	Wert
Name | ID	Radar | DOI-IN-RADAR
Beschreibung	Radartracks, die dazu dienen dynamische Objekte zu identifizieren.
Protokoll	NMEA2000-Datenstrom via UDP
Server	localhost
Port	8392

Atribut	Wert
Name | ID	AIS Dynamisch | DOI-IN-AIS-DYN
Beschreibung	AIS Positionsinformationen, die dazu dienen dynamische Objekte zu identifizieren.
Protokoll	NMEA2000-Datenstrom via UDP
Server	localhost
Port	8392

Atribut	Wert
Name | ID	AIS Statisch | DOI-IN-AIS-STAT
Beschreibung	Statische AIS Informationen, die dazu dienen dynamische Objekte zu identifizieren.
Protokoll	NMEA2000-Datenstrom via UDP
Server	localhost
Port	8392

Atribut	Wert
Name | ID	Das Heading der Zuse | DOI-IN-HEAD
Beschreibung	Das aktuelle Heading der Zuse.
Protokoll	NMEA2000-Datenstrom via UDP
Server	localhost
Port	8392

Atribut	Wert
Name | ID	Die Position der Zuse | DOI-IN-POS
Beschreibung	Die aktuelle Position der Zuse.
Protokoll	NMEA2000-Datenstrom via UDP
Server	localhost
Port	8392

Atribut	Wert
Name | ID	Kamera | DOI-IN-KAMERA
Beschreibung	Position von Schiffen, welche von der Objektdetektion erkannt wurden, die dazu dienen dynamische Objekte zu identifizieren.
Protokoll	UDP
Server	localhost
Port	8392

Ausgehend:

Atribut	Wert
Name | ID	Identifizierte, dynamische Objekte | DOI-OUT-FUSED
Beschreibung	Objekte, welche die DOI durch Assoziation und Regression verschiedener Datenströme identifiziert hat.
Protokoll	JSON via http
Server	path-planning.server @ '/dynamicObjects'
Port	8282

Atribut	Wert
Name | ID	Identifizierte, dynamische Objekte | DOI-OUT-EPD
Beschreibung	AIS- und Radar-Schiffe (nicht fusioniert) sowie fusionierte Schiffe, welche durch verschiedenen Datenströmen identifiziert wurden.
Protokoll	JSON via HTTP
Server	localhost @ '/doiDataInput'
Port	7000

Funktionalität: Dynamische Hindernisse können der Pfadplanung und der EPD durch eine http-POST Anfrage als JSON-Objekt im Request-Body mitgeteilt werden. Dazu wird die url `http://pathplanning.ip:8282/dynamicObjects` verwendet. Momentan wird nur eine Operation unterstützt. Sie teilt der Pfadplannung und der EPD die dynamischen Hindernisse mit und ist wie folgt aufgebaut:

Attribut	Wert
action	POST
contentType	fused, ais, radar
ship[]	JSON-Objekte, welche andere Schiffe beschreiben (siehe unten))

Das *ship*-Objekt ist definiert durch:

JSON-Attribut	Wert
shipidentifier	ID des Schiffs
latitude	Latitude
logitude	Longitude
ship Length	Länge
ship Beam	Breite
cog	Course over ground
sog	Speed over ground

Beispiel:

```
{
    "action":"POST",
    "contentType":"fused",
    "content":[
        {
            "ShipID":"FR1AIS211286490",
            "Latitude":"53.5247087",
```

```
            "Longitude":"8.1821687",
            "Ship Length":"40.0",
            "Ship Beam":"20.0",
            "COG":"6.97",
            "SOG":"7.41"
        }
    ]
}
```

Objektdetektion

Datenströme

Eingehend:

Atribut	Wert
Name | ID	Kamerastream | OD-IN-KAMERA
Beschreibung	Kamerastream, um Objekte hierauf zu detektieren
Protokoll	?
Server	localhost
Port	5555

Ausgehend:

Atribut	Wert
Name | ID	Position der Objekte | OD-OUT-POS
Beschreibung	Position der Schiffe, die von der Objektdetektion auf dem Kamerastream erkannt und klassifiziert wurde
Protokoll	UDP
Server	localhost
Port	8392

Statische Objekterkennung (NoGoSolver)

Datenströme

Statisch:

Atribut	Wert
Name | ID	Kartendaten | CHARTS
Beschreibung	Digitale Kartendaten eines maritimen Bereiches.
Protokoll	S-57 im gml-Format
Server	Lokale Datei
Port	-

Eingehend:

Atribut	Wert
Name | ID	Schiffsinformationen | SOI-IN
Beschreibung	Informationen über das Schiff, für das NoGo-Areas untersucht werden sollen (z.B: Tiefgang).
Protokoll	im json-Format via Websockets
Server	localhost
Port	8025

Ausgehend:

Atribut	Wert
Name | ID	NoGo-Areas | SOI-OUT
Beschreibung	Statische Hindernisse, die von dem spezifizierten Schiff nicht passiert werden können
Protokoll	WKT-Geometrien gruppiert im json-Format via Websockets
Server	soi.server
Port	8025

Controller

Datenströme

Eingehend:

Atribut	Wert
Name | ID	Wegpunkte | CONT-IN-AUTO-WP
Beschreibung	Der vorherige und der nächste Wegpunkt, die angefahren werden sollen.
Protokoll	NMEA2000-Datenstrom via UDP
Server	localhost
Port	8001

Atribut	Wert
Name | ID	Statusinformationen der Zuse | CONT-IN-AUTO-STATUS
Beschreibung	Der aktuelle Status des Schiffes, zur Berechnung der Steuerbefehle.
Protokoll	NMEA2000-Datenstrom via UDP
Server	localhost
Port	8001

Atribut	Wert
Name | ID	Manual Steering | CONT-IN-MANUAL
Beschreibung	Manuelle Steuerbefehle, die vom Controller geforwarded werden.
Protokoll	NMEA2000-Datenstrom via UDP
Server	localhost
Port	8001

Ausgehend:

Atribut	Wert
Name | ID	Steuerbefehle | CONT-OUT
Beschreibung	Vom Controller berechnete (bzw. weitergeleitete) Steuerbefehle
Protokoll	NMEA2000-Datenstrom via UDP
Server	polymorphic-hub.server
Port	8002

Routenplanung

Datenströme

Eingehend:

Atribut	Wert
Name | ID	Schiffsinformationen | RP-IN-INFO
Beschreibung	Informationen über das Schiff, für das eine Route geplant werden soll.
Protokoll	json via HTTP
Server	localhost
Port	8020

Funktionalität: Schiffsinformationen können dem Routeplanner durch eine HTTP-POST Anfrage als json-Objekt im Request-Body mitgeteilt werden. Dazu wird die URL `http://routeplanner.ip:8020/ship` verwendet. Momentan wird nur eine Operation unterstützt. Sie teilt dem Route-Planner die Schiffsinformationen mit und ist wie folgt aufgebaut:

Attribut	Wert
action	...
contentType	...
ship	JSON-Objekt, welches das Schiff beschreibt (siehe unten)

Das *ship*-Objekt ist definiert durch:

JSON-Attribut	Wert
draft	Tiefgang
width	Breite
height	Höhe
length	Länge

Beispiel:

```
{
    "contentType":"",
    "action":"",
    "ship": {
        "width":"10.2",
        "height":"3.0",
        "length":"20.0",
        "draft":"1.9"
```

```
9       }
10  }
```

Atribut	Wert
Name | ID	User Route | RP-IN-ROUTE
Beschreibung	Route, die ein Benutzer geplant hat (nicht optimiert).
Protokoll	RTZ via HTTP
Server	localhost
Port	8020

Ausgehend:

Atribut	Wert
Name | ID	NoGoSolver-Anfrage | RP-OUT-NOGO
Beschreibung	Anfrage an NoGoSolver, um statische Hindernisse zu erhaltent.
Protokoll	JSON via HTTP
Server	localhost
Port	8025

Funktionalität: Statische Hindernisse für eine angegebenen Bereich können vom NoGoSolver durch eine http-GET Anfrage als json-Objekt im Request-Body angefragt werden. Dazu wird die URL `http://localhost:8025` verwendet. Sie ist wie folgt aufgebaut:

Attribut	Wert
action	GET
contentType	NoGoInformation
request	JSON-Objekt, welches das Schiff und den Anfrage-Bereich beschreibt (siehe unten)

Das *request*-Objekt ist definiert durch:

JSON-Attribut	Wert
shipidentifier	ID des Schiffs
draft	Tiefgang
width	Breite
height	Höhe
length	Länge
minLat	Kleinster latitude-Wert des Abfragebereichs
maxLat	Größter latitude-Wert des Abfragebereichs
minLon	Kleinster Längengrad-Wert des Abfragebereichs
maxLon	Größter Längengrad-Wert des Abfragebereichs

Beispiel:

```
{
    "contentType":"NoGoInformation",
    "action":"GET",
    "request": {
        "shipidentifier" : "id"
        "width":"10.2",
        "height":"3.0",
        "length":"20.0",
        "draft":"1.9",
        "minLat":"2.004",
        "maxLat":"2.013",
        "minLon":"40.002",
        "maxLon":"40.016"
    }
}
```

Atribut	Wert
Name | ID	Optimierte Route | RP-OUT-ROUTE
Beschreibung	Optimierte Route, die statische Objekte (bspw. Land) umfährt.
Protokoll	RTZ via HTTP
Server	route-planning.server
Port	7000

Pfadplanung

Datenströme

Eingehend:

Atribut	Wert
Name | ID	Schiffsinformationen | PP-IN-INFO
Beschreibung	Informationen über das Schiff, für das Pfade geplant werden soll.
Protokoll	json via HTTP
Server	localhost
Port	8282

Atribut	Wert
Name | ID	Optimierte Route | PP-IN-ROUTE
Beschreibung	Optimierte Route, die statische Objekte (bspw. Land) umfährt, und verfolgt werden soll.
Protokoll	RTZ via HTTP
Server	localhost
Port	8282

Atribut	Wert
Name | ID	Identifizierte, dynamische Objekte | PP-IN-DOI
Beschreibung	Objekte, welche die DOI durch Assoziation (und Regression) verschiedener Datenströme identifiziert hat.
Protokoll	json-Datenstrom via HTTP
Server	localhost
Port	8282

Atribut	Wert
Name | ID	Die Position der ZUSE | PP-IN-POS
Beschreibung	Die aktuelle Position der ZUSE.
Protokoll	NMEA0183-Datenstrom via UDP
Server	localhost
Port	8283

Atribut	Wert
Name | ID	Das Heading der ZUSE | PP-IN-HEAD
Beschreibung	Das aktuelle Heading der ZUSE.
Protokoll	NMEA0183-Datenstrom via UDP
Server	localhost
Port	8283

Atribut	Wert
Name | ID	NoGo-Areas | PP-IN-SOI
Beschreibung	Statische Hindernisse, die von dem spezifizierten Schiff nicht passiert werden können.
Protokoll	WKT-Geometrien gruppiert im json-Format via Websockets
Server	localhost
Port	8025

Ausgehend:

Atribut	Wert
Name | ID	Optimierte Route | PP-OUT-PFAD
Beschreibung	Pfad, der dynamische und statische Objekte umfährt.
Protokoll	RTZ via HTTP
Server	epd.server
Port	7000

Atribut	Wert
Name | ID	Wegpunkte | PP-OUT-WPS
Beschreibung	Der erste und der fünfte Wegpunkt des Pfades, die angefahren werden sollen.
Protokoll	RTZ via HTTP
Server	polymorphic-hub.server
Port	4243

Atribut	Wert
Name | ID	NoGoSolver-Anfrage | PP-OUT-NOGO
Beschreibung	Anfrage an NoGoSolver, um statische Hindernisse zu erhaltent.
Protokoll	JSON via HTTP
Server	localhost
Port	8025

EPD

Datenströme

Eingehend:

Atribut	Wert
Name | ID	Identifizierte Dynamische Objekte | EPD-IN-DOI
Beschreibung	AIS-, Radar- oder fusionierte Schiffe / Verarbeitete Daten der dyn. Objektidentifizierung
Protokoll	json via HTTP
Server	localhost @ '/doiDataInput'
Port	7000

Beispiel:

```
{
    "action":"POST",
    "contentType":"fused",
    "content":[
        {
            "Ship Length":"40.0",
            "Latitude":"53.5247087",
            "Ship Beam":"20.0",
            "COG":"6.97",
            "SOG":"0.0",
            "ShipID":"FAIS211286490",
            "Longitude":"8.1821687"
        },
        {
            "Ship Length":"20.0",
            "Latitude":"53.5240655",
            "Ship Beam":"10.0",
            "COG":"6.93",
            "SOG":"0.0",
            "ShipID":"FRadar1",
            "Longitude":"8.1823527"
        }
```

```
    ]
}
```

Beispiel:

```
{
    "action":"POST",
    "contentType":"ais",
    "content":[
        {
            "Ship Length":"0.0",
            "Latitude":"53.5247087",
            "Ship Beam":"0.0",
            "COG":"6.97",
            "SOG":"0.0",
            "ShipID":"A211286490",
            "Longitude":"8.1821687"
        }
    ]
}
```

Beispiel:

```
{
    "action":"POST",
    "contentType":"radar",
    "content":[
        {
            "Ship Length":"0.0",
            "Latitude":"53.5240655",
            "Ship Beam":"0.0",
            "COG":"6.93",
            "SOG":"0.0",
            "ShipID":"R1",
            "Longitude":"8.1823527"
        }
    ]
}
```

Eingehend:

Atribut	Wert
Name | ID	Path-Planning Pfad | EPD-IN-PP
Beschreibung	Vom Path-Planning geplanter Pfad im RTZ-Format.
Protokoll	RTZ via HTTP
Server	localhost @ '/path'
Port	7000

Beispiel:

```
<?xml version="1.0"?>
<route xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="
       1.0"
    xmlns="http://www.cirm.org/\gls{RTZ}/1/0">
  <routeInfo routeName="Example Route" />
  <waypoints>
    <waypoint id="0" radius="0.008">
      <position lat="53.518166666666666" lon="
         8.176416666666666"/>
      <leg speedMax="5.144444444444445"/>
    </waypoint>
    <waypoint id="1" radius="0.008">
      <position lat="53.51928586954941" lon="
         8.175095103625296"/>
      <leg speedMax="5.144444444444445"/>
    </waypoint>
    <waypoint id="2" radius="0.008">
      <position lat="53.520226333086356" lon="
         8.174214419160476"/>
      <leg speedMax="5.144444444444445"/>
    </waypoint>
  </waypoints>
</route>
```

Eingehend:

Atribut	Wert
Name | ID	Debuginformationen Path-Planning | EPD-IN-PP-DBG
Beschreibung	Goodwin-Shapes oder NoGoAreas als Debug-Informationen für die Pfadplanung zum Anzeigen auf der EPD-Karte
Protokoll	json via HTTP
Server	localhost @ '/pathPlanningJsonInput'
Port	7000

Eingehend:

Atribut	Wert
Name | ID	Optimierte Route | EPD-IN-RP
Beschreibung	Vom Route-Planning optimierte Route.
Protokoll	json via HTTP
Server	localhost @ '/route'
Port	7000

Beispiel:

```
<?xml version="1.0"?>
<route xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="
       1.0"
    xmlns="http://www.cirm.org/\gls{RTZ}/1/0">
  <routeInfo routeName="Example Route" />
  <waypoints>
    <waypoint id="0" radius="0.008">
      <position lat="53.518166666666666" lon="
         8.176416666666666"/>
      <leg speedMax="5.144444444444445"/>
    </waypoint>
    <waypoint id="1" radius="0.008">
      <position lat="53.51928586954941" lon="
         8.175095103625296"/>
      <leg speedMax="5.144444444444445"/>
    </waypoint>
    <waypoint id="2" radius="0.008">
      <position lat="53.520226333086356" lon="
         8.174214419160476"/>
      <leg speedMax="5.144444444444445"/>
    </waypoint>
  </waypoints>
</route>
```

Eingehend:

Atribut	Wert
Name | ID	Debuginformationen und Statusmeldungen vom Route-Planning | EPD-IN-RP-DBG
Beschreibung	NoGo-Areas, Grid-Belegungen, Statusinformationen: Timeout oder NoRouteFindable
Protokoll	json via HTTP
Server	localhost @ '/routePlanningJsonInput'
Port	7000

Funktionalität: Diverse Informationen die visualisiert werden, können der EPD durch eine http-POST Anfrage als json-Objekt im Request-Body mitgeteilt werden. Dazu wird die URL `http://epd.ip:7000/apiURL` verwendet. Momentan werden 6 apiURLs unterstützt.

Ausgehend:

Atribut	Wert
Name | ID	Manuelle Steuerbefehle | EPD-OUT-CONT
Beschreibung	Manuelle Steuerbefehle (Ruderwinkel, RPM) die an den Controller gesendet werden
Protokoll	NMEA2000 via UDP
Server	controller.ip
Port	8001

Ausgehend:

Atribut	Wert
Name | ID	Zu optimierende Route | EPD-OUT-RP
Beschreibung	Eine benutzerdefinierte Route, die zur Routenplanung geschickt wird um diese zu optimieren.
Protokoll	RTZ via HTTP
Server	route-planning.ip @ '/route'
Port	benutzerdefiniert

Ausgehend:

Atribut	Wert
Name | ID	Zu fahrende Route | EPD-OUT-PP
Beschreibung	Eine von der Routenplanung bereits optimierte Route, die vom Schiff gefahren werden soll.
Protokoll	RTZ via HTTP
Server	path-planning.ip @ '/route'
Port	benutzerdefiniert

Ausgehend:

Atribut	Wert
Name | ID	Schiffsinformationen RP / PP | EPD-OUT-SI
Beschreibung	Schiffsinformationen zur Berechnung der optimierten Route benötigt.
Protokoll	json via HTTP
Server	path-planning.ip bzw. route-planning.ip @ '/ship'
Port	benutzerdefiniert

Simulationsadapter

Datenströme

Eingehend:

Atribut	Wert
Name | ID	Schiffsbeschreibende Objektdaten | SIM-IN
Beschreibung	Informationen über Schiffe, die Live in einem HAGGIS-Szenario simuliert werden.
Protokoll	uCoreXML/S100 via UDP komprimiert mit GZIP
Server	localhost
Port	5001

Ausgehend:

Atribut	Wert
Name | ID	AIS Statisch | SIM-AIS-STAT-OUT
Beschreibung	Von HAGGIS übersetzte AIS Daten zu statischen eCore AIS Daten.
Protokoll	eCoreXML/S100 via RabbitMQ
Server	mate.rabbitmq
Port	5672

Atribut	Wert
Name | ID	AIS Dynamisch | SIM-AIS-DYN-OUT
Beschreibung	Von HAGGIS übersetzte AIS Daten zu dynamischen eCore AIS Daten.
Protokoll	eCoreXML/S100 via RabbitMQ
Server	mate.rabbitmq
Port	5672

Atribut	Wert
Name | ID	Positionsdaten | SIM-POS-OUT
Beschreibung	Von HAGGIS übersetzte Schiffspositionsdaten in eCore Schiffsposiotionsdaten.
Protokoll	eCoreXML/S100 via RabbitMQ
Server	mate.rabbitmq
Port	5672

Atribut	Wert
Name | ID	Radar | SIM-RADAR-OUT
Beschreibung	Von HAGGIS gesendetes symbolisches Radar in eCore übersetzt und mit simulierten Daten ergänzt.
Protokoll	eCoreXML/S100 via RabbitMQ
Server	mate.rabbitmq
Port	5672

E. Ablaufpläne der Systemtests

Systemtest vom 15.08.2018

Systemtest am Mittwoch, 15.08.

Akt 0: Anfahrt (1 Stunde)

Abfahrt **7:00** Uhr am Offis

1. Anfahrt + Parken spätestens um 8 Uhr

Akt 1: Vorbereitungen (maximal 2 Stunden)

8-10:00 Uhr im Hafen

2. Begrüßung und Einstimmung (Was machen wir heute, Teams vorstellen)
3. Zuse in Betrieb nehmen (Tanken, Landstrom)
4. Infrastruktur in Betrieb nehmen
   - LTE-Modul
   - Switch
   - Rechner verkabeln
   - Kamera anbringen (Welche Kamera wird verwendet?)
5. Komponenten in Betrieb nehmen + Konfiguration (Übersicht ausdrucken und in Zuse aufhängen)
   a. Navibox:
      i. 192.168.1.100
   b. IPC
      i. Controller, PI, HUB, EPD, DOI
      ii. IP: 192.168.1.200
         1. EPD konfigurieren (Output-URL für Controller, RP, PP)
         2. xml-Dateien für hub und PI anpassen
         3. DOI konfigurieren (Konfig: Input-Adresse, HTTPRequestService -> PP, EPD)
   c. Lea
      i. SA und Haggis
      ii. IP: auto-DHCP
   d. Christian
      i. Object Detection muss auf Navibox laufen
      ii. Output-URL für EPD (Video-Stream) und DOI
      iii. IP: 192.168.1.165
   e. Julian
      i. PP, NoGoSolver, RP+NoGoSolver
      ii. z.B. Output-URLs für EPD (inkl. Debug) und Hub/Controller, Server-Bindings
      iii. IP: 192.168.1.163
   f. Julius
      i. MateControl + EPD-Anzeige via RDP
      ii. IP: 192.168.1.161
6. Systeme in Betrieb nehmen
   a. MateControl
   b. PI/HUB
   c. DOI
   d. Controller
   e. EPD + NMEA-Sensor + VirtualHandles
   f. RP + NoGoSolver

g. PP + NoGoSolver
h. SA + Haggis
i. Object Detection
j. Radar (zuletzt, Anleitung beachten)

7. Netzwerkverbindungen via Wireshark testen
    a. EPD -> PP, RP, Controller
    b. RP -> EPD
    c. DOI -> EPD, PP
    d. Object Detection -> DOI
    e. PP -> EPD, Hub (für Controller)
    f. PI -> Hub
    g. Hub -> EPD, PP, DOI, SA, Controller
    h. SA -> RabbitMQ/IPC

Akt 2: Testszenario 0 + 1 10:00-12:00 Uhr

1. **Testszenario 0 (TS-0) -> 0,5 Stunden**
    a. Komponenten/Team
        i. EPD, PP, RP, Controller
        ii. Ole, Julian, Hilko + Julius, Christian
    b. Aufwand
        i. Aus dem Hafen rausfahren
        ii. Wegpunkt x und y in Controller fest einstellen
        iii. Skripte der Zuse (Raspberry PI) einschalten
        iv. Controller-Steuerung (Ruder und RPM) aktivieren
        v. Fahrt überwachen
2. **Testszenario 1 (TS-1) -> maximal 1,5 Stunden (direkt im Anschluss an TS-0)**
   -- Während Aufwands-Schritt i.-xi. erfolgt ein Parametertuning des Controllers --
    a. Komponenten/Team
        i. EPD, PP, RP, Controller
        ii. Ole, Julian, Hilko + Julius, Christian
    b. Aufwand (30 Min + 1 Stunde)
        i. Route auf Land setzen (ungültige Route) (Referenz: RouteUngueltig.rtz)
        ii. Zufalls-Route in EPD erstellen
        iii. Route ans RP senden
        iv. NoGoSolver + Berechnung des RP
        v. Neue Route wird in Epd gespeichert und angezeigt
        vi. Polygone, Grid usw. werden in EPD angezeigt
        vii. ersten optimierten Wegpunkt anfahren (Umkreis max. 600m)
        viii. Skripte starten
        ix. Controller-Steuerung (Ruder und RPM) aktivieren
        x. EPD -> PP Route starten
        xi. Fahrt überwachen

-- Einlaufen in den Hafen --

- Mittagspause 12:00 -> 12:30 Uhr -

Akt 3: Testszenarien 2-5 (TS2-5) 12:30-17:00

3. **Testszenario 2 (TS-2)** Überholmanöver Schiff (COLREG 13)
   maximal 1,5 Stunden
   -> 30 Min Vorbereitung im Hafen
   -> 1 Stunde auf dem Wasser
   a. Komponenten/Team
      i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
      ii. Ole, Dennis, Hilko oder Julius, Linda, Lea
   b. Aufwand
      i. Fahrt zum ersten Wegpunkt (50-100m) und Anfahren mit konstanter Geschwindigkeit
      ii. Optimierte Route in EPD ans Path-Planning (Geschwindigkeit: 7 Knoten!)

-- ggfs. Einlaufen in den Hafen (je nach Situation und Zeitpuffer) –

4. **Testszenario 3 (TS-3)** Kreuzendes Schiff von rechts (wir müssen ausweichen) (COLREG 15, 16, 17)
   maximal 1 Stunde
   a. Komponenten/Team
      i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
      ii. Ole, Dennis, Hilko oder Julius, Yvonne oder Linda, Lea
   b. Aufwand
      i. Fahrt zum nächsten Szenario-Wegpunkt (50-100m) und Anfahren mit konstanter Geschwindigkeit
      ii. Optimierte Route in EPD ans Path-Planning (Geschwindigkeit: 7 Knoten!)
5. **Testszenario 4 (TS-4)** Kreuzendes Schiff von links (COLREG 15, 16, 17)
   maximal 1 Stunde
   a. Komponenten/Team
      i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
      ii. Ole, Dennis, Hilko oder Julius, Yvonne oder Linda, Lea
   b. Aufwand
      i. Fahrt zum nächsten Szenario-Wegpunkt (50-100m) und Anfahren mit konstanter Geschwindigkeit
      ii. Optimierte Route in EPD ans Path-Planning (Geschwindigkeit: 7 Knoten!)
6. **Testszenario 5 (TS-5)** Entgegenkommendes Schiff (COLREG 14, 16, 17)
   maximal 1 Stunde
   a. Komponenten/Team
      i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
      ii. Ole, Dennis, Hilko oder Julius, Yvonne oder Linda, Lea
   b. Aufwand
      i. Fahrt zum nächsten Szenario-Wegpunkt (50-100m) und Anfahren mit konstanter Geschwindigkeit
      ii. Optimierte Route in EPD ans Path-Planning (Geschwindigkeit: 7 Knoten!)

Backup-Akt 3b: Fehlerinjektion (Zeit variabel, je nach Restzeit)

1. TC-FI-2 Verbindungsabbruch Navibox
2. TC-FI-3 Gemessener Ruderwinkel außerhalb der Grenzen
3. TC-FI-4 Gemessene negative Drehzahl
4. TC-FI-7 Nahes Schiff verschwindet
5. TC-FI-10 Bekanntes Schiff springt auf eigene Position
6. TC-FI-12 Schiff wechselt AIS-MMSI
7. Weitere: 1,5,6,8,9,11,13,14,15

Akt 4: Abbau, Aufräumen und Abschlussbesprechung 17:00 (15 Minuten)

Ende: max. 17:15 Uhr

Rückfahrt max. 17:30 Uhr, Ankunft spätestens 18:30 Uhr in Oldenburg

Systemtest vom 29.08.2018

Abnahmetest am Mittwoch, 29.08.

Akt 0: Anfahrt (1 Stunde)

Abfahrt **7:00** Uhr am Offis

1. Anfahrt + Parken spätestens um 8 Uhr

Akt 1: Vorbereitungen (maximal 2 Stunden)

8-10:00 Uhr im Hafen

2. Begrüßung und Einstimmung
3. Zuse in Betrieb nehmen (Tanken, Landstrom)
4. Infrastruktur in Betrieb nehmen
   - LTE-Modul
   - Switch
   - Rechner verkabeln
   - Kamera aufbauen
5. Komponenten in Betrieb nehmen + Konfiguration (Deployment-Diagramm beachten!)
   a. Navibox:
      i. RMQ + Object Detection
      ii. 192.168.1.100
   b. IPC
      i. Controller, PI, HUB, EPD, DOI
      ii. IP: 192.168.1.200
   c. Lea
      i. SA und Haggis
      ii. IP: auto-DHCP
   d. Julian
      i. PP+NoGoSolver, RP+NoGoSolver
      ii. IP: 192.168.1.163
   e. Julius
      i. MateControl + EPD-Anzeige via RDP
      ii. IP: 192.168.1.161
6. Systeme in Betrieb nehmen
   a. MateControl
   b. PI/HUB
   c. DOI
   d. Controller
   e. EPD + NMEA-Sensor + VirtualHandles
   f. RP + NoGoSolver, PP + NoGoSolver
   g. SA + Haggis
   h. Object Detection
   i. Radar (zuletzt, Anleitung beachten)
7. Netzwerkverbindungen anhand des Deployment-Diagramms prüfen!
   a. EPD -> PP, RP, Controller
   b. RP -> EPD+NoGoSolver
   c. DOI -> EPD, PP
   d. Object Detection -> DOI, EPD (für Kamerabild)
   e. PP -> EPD, Hub (für Controller)
   f. PI -> Hub
   g. Hub -> PI, EPD, PP, DOI, SA, Controller
   h. SA -> RabbitMQ/Navibox

Akt 2: Basistest, Parametertuning Controller + Basis-Test-1 10:00-12:00 Uhr

Auslaufen 10:00 Uhr

30 Minuten Puffer, während Basis-Test-0 wird ein Parametertuning des Controllers vorgenommen!

1. **Basis-Test-0 -> 0,75 Stunden – „Delmenhorst“ –** Abfahrt fester Wegpunkte
    a. Komponenten/Team
        i. EPD, PP, RP, Controller, IPC
        ii. Ole, Julian, Hilko, Julius, Christian
2. **Basis-Test-1 -> 0,75 Stunden – „Oldenburg“ –** Abfahrt einer optimierten Route
    a. Komponenten/Team
        i. EPD, PP, RP, Controller, IPC
        ii. Ole, Julian, Hilko, Julius, Christian
3. **Basis-Test-2a+b -> 0,37 Stunden – „Bremen“ –** Steuerung der Zuse durch VirtualHandles
    a. Komponenten/Team
        i. EPD, PP, RP, Controller, IPC
        ii. Ole, Julian, Hilko, Julius, Christian
4. **Basis-Test-3a+b -> 0,37 Stunden – „Berlin“ –** Erkennen von Schiffen mittels Kamera
    a. Komponenten/Team
        i. EPD, PP, RP, Controller, IPC
        ii. Ole, Julian, Hilko, Julius, Christian

- Einlaufen in Hafen um 12:00 -

- Mittagspause 12:00 -> 12:30 Uhr -

Akt 3: Testszenarien 1-6 (TS1-6) 12:30-16:30

1. **Testszenario 1 (TS-1) „Europa“ -** Kreuzendes Schiff von links (Ausweichen) (COLREG 15, 16, 17)
    a. maximal 30 Minuten
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Dennis, Hilko, Yvonne, Lea
2. **Testszenario 2 (TS-2) „Asien“ -** Kreuzendes Schiff von rechts (COLREG 15, 16, 17)
    a. maximal 30 Minuten
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Dennis, Hilko, Yvonne, Lea
3. **Testszenario 3 (TS-3) „America“ -** Entgegenkommendes Schiff (COLREG 14, 16, 17)
    a. maximal 30 Minuten
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Dennis, Hilko, Yvonne, Lea

- 14:00 Uhr: Einlaufen in den Hafen, wenn innerhalb der Pufferzeit – Pause bis 15:00 Uhr -

4. **Testszenario 4 (TS-4) „Australien“ -** Überholmanöver Schiff (COLREG 13)
    a. maximal 30 Minuten
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Dennis, Hilko, Julius, Lea
5. **Testszenario 5 (TS-5) „Helgoland“ -** Kreuzendes Schiff von links (Vorfahrt) (COLREG 15, 16, 17)
    a. maximal 30 Minuten
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Dennis, Hilko, Julius, Lea

6. **Testszenario 6 (TS-6) „Brocken“ -** Ausweichen eines dynamischen und statischen Hindernisses
    a. **maximal 30 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Dennis, Hilko, Julius, Lea

Akt 4: Fehlerinjektion 16:30-17:30

1. **E1: „Wilhelmshaven“ – ein Schiff wechselt auf unsere Position**
    a. **maximal 10 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Julian, Julius, Yvonne, Lea
2. **TS-E1: „Africa 1“ – ein Schiff verschwindet**
    a. **maximal 10 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Julian, Julius, Yvonne, Lea
3. **TS-E2: „Africa 2“ – ein Schiff taucht wieder auf**
    a. **maximal 10 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Julian, Julius, Yvonne, Lea
4. **E4: „Bielefeld“ – Wechsel der MMSI**
    a. **maximal 15 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Julian, Julius, Yvonne, Lea
5. **E5: „Wolfsburg“ – Ungültiger Ruderwinkel**
    a. **maximal 15 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Julian, Julius, Yvonne, Lea

- Einlaufen in den Hafen -

Akt 5: Abbau, Aufräumen und Abschlussbesprechung (15 Minuten)

Ende: max. 17:45 Uhr

Rückfahrt max. 17:45 Uhr, Ankunft spätestens 18:45 Uhr in Oldenburg

Systemtest vom 12.09.2018

Abnahmetest am Mittwoch, 12.09.

Akt 0: Anfahrt (1 Stunde)

Abfahrt **8:00** Uhr am Offis

1. Anfahrt + Parken spätestens um 9 Uhr

Akt 1: Vorbereitungen (maximal 2 Stunden)

Schlüssel besorgen, Boot in Betrieb nehmen

09-11:00 Uhr im Hafen

2. Begrüßung und Einstimmung
3. Zuse in Betrieb nehmen (Tanken, Landstrom)
4. Infrastruktur in Betrieb nehmen
   - LTE-Modul
   - Switch
   - Rechner verkabeln
   - Kamera aufbauen
5. Komponenten in Betrieb nehmen + Konfiguration (Deployment-Diagramm beachten!)
   a. Navibox:
      i. RMQ + Object Detection
      ii. 192.168.1.100
   b. IPC
      i. Controller, PI, HUB, EPD, DOI
      ii. IP: 192.168.1.200
   c. Lea
      i. SA und Haggis
      ii. IP: auto-DHCP
   d. Hilko
      i. EPD-Anzeige via RDP
      ii. IP: 192.168.1.163
   e. Julius
      i. MateControl + PP+NoGoSolver
      ii. IP: 192.168.1.161
6. Systeme in Betrieb nehmen
   a. MateControl
   b. PI/HUB
   c. DOI
   d. Controller
   e. EPD + NMEA-Sensor (+ VirtualHandles)
   f. (RP + NoGoSolver), PP + NoGoSolver
   g. SA + Haggis
   h. Object Detection
   i. (Radar (zuletzt, Anleitung beachten))
7. Netzwerkverbindungen anhand des Deployment-Diagramms prüfen!
   a. EPD -> PP, RP, Controller
   b. RP -> EPD+NoGoSolver
   c. DOI -> EPD, PP
   d. Object Detection -> DOI, EPD (für Kamerabild)
   e. PP -> EPD, Hub (für Controller)
   f. PI -> Hub
   g. Hub -> PI, EPD, PP, DOI, SA, Controller
   h. SA -> RabbitMQ/Navibox

Abnahmetest am Mittwoch, 12.09.

Akt 2: Basistests 11:00-12:30 Uhr

30 Minuten Puffer

Auslaufen spätestens 11:30 Uhr

1. **Basis-Test-0 -> 0,25 Stunden – „Delmenhorst“ –** Abfahrt fester Wegpunkte
    a. Komponenten/Team
        i. EPD, PP, RP, Controller, IPC
        ii. Ole, Hilko, Julius, Christian, Lea
2. **Basis-Test-1 -> 0,25 Stunden – „Oldenburg“ –** Abfahrt einer optimierten Route
    a. Komponenten/Team
        i. EPD, PP, RP, Controller, IPC
        ii. Ole, Hilko, Julius, Christian, Lea
3. **Basis-Test-2a+b -> 0,25 Stunden – „Bremen“ –** Steuerung der Zuse durch VirtualHandles
    a. Komponenten/Team
        i. EPD, PP, RP, Controller, IPC
        ii. Ole, Hilko, Julius, Christian, Lea
4. **Basis-Test-3a+b -> 0,25 Stunden – „Berlin“ –** Erkennen von Schiffen mittels Kamera
    a. Komponenten/Team
        i. EPD, PP, RP, Controller, IPC
        ii. Ole, Hilko, Julius, Christian, Lea
5. **Basis-Test-4 (10.1) -> 0,25 Stunden – „Meppen“ –** Datenfusion (ohne AIS)
    a. Komponenten/Team
        i. EPD, PP, RP, Controller, IPC
        ii. Ole, Hilko, Julius, Christian, Lea

- Mittagspause spätestens 12:30 Uhr -> 13:00 Uhr –

Akt 3: Testszenarien 1-6 (TS1-6) 13:00-16:00

1. **Testszenario 1 (TS-1) „Europa“ -** Kreuzendes Schiff von links (Ausweichen) (COLREG 15, 16, 17)
    a. maximal 20 Minuten
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Hilko, Julius, Christian, Lea

2. **Testszenario 2 (TS-2) „Asien“ -** Kreuzendes Schiff von rechts (COLREG 15, 16, 17)
    a. maximal 20 Minuten
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Hilko, Julius, Christian, Lea

3. **Testszenario 3 (TS-3) „Amerika“ -** Entgegenkommendes Schiff (COLREG 14, 16, 17)
    a. maximal 20 Minuten
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Hilko, Julius, Christian, Lea

-- spätestens 14:00 Uhr - Einlaufen in den Hafen -> Dokumentation der Testergebnisse –

Abnahmetest am Mittwoch, 12.09.

4. **Testszenario 4 (TS-4) „Helgoland“ -** Kreuzendes Schiff von links (Vorfahrt) (COLREG 15, 16, 17)
    a. **maximal 20 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Hilko, Julius, Christian, Lea
5. **Testszenario 5 (TS-5) „Brocken“ -** Ausweichen eines dynamischen und statischen Hindernisses
    a. **maximal 20 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Hilko, Julius, Christian, Lea
6. **Testszenario 6 (TS-6) „Australien“ -** Überholmanöver Schiff (COLREG 13)
    a. **maximal 20 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Hilko, Julius, Christian, Lea

Akt 4: Fehlerinjektion (Je nach Zeitpuffer) – maximal Ende 16:00 Uhr

1. **E1: „Wilhelmshaven“ – ein Schiff wechselt auf unsere Position**
    a. **maximal 10 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Hilko, Julius, Christian, Lea
2. **TS-E1: „Afrika 1“ – ein Schiff verschwindet**
    a. **maximal 10 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Hilko, Julius, Christian, Lea
3. **TS-E2: „Afrika 2“ – ein Schiff taucht wieder auf**
    a. **maximal 10 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Hilko, Julius, Christian, Lea
4. **E4: „Bielefeld“ – Wechsel der MMSI**
    a. **maximal 15 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Hilko, Julius, Christian, Lea
5. **E5: „Wolfsburg“ – Ungültiger Ruderwinkel**
    a. **maximal 15 Minuten**
    b. Komponenten/Team
        i. EPD, PP, Controller, DOI, Object Detection, SA, Haggis
        ii. Ole, Hilko, Julius, Christian, Lea

- Einlaufen in den Hafen -

Akt 5: Abbau, Aufräumen und Abschlussbesprechung (15 Minuten)

Ende: max. 16:15 Uhr

Rückfahrt max. 16:30 Uhr, Ankunft spätestens 17:30 Uhr in Oldenburg

F. Testkataloge der Systemtests

Systemtest vom 12.07.2018

Labskaus-Integrationstest:

Methode.
Zunächst erfolgt ein Systemintegrationstest des PG-MATE Systems und LABSKAUS. Der Systemintegrationstest verifiziert die korrekte Funktionsweise der polymorphen Schnittstelle hinsichtlich einer erfolgten Kommunikationsverbindung, sowie der syntaktischen Korrektheit der Daten. Anschließend beurteilt ein Systemexperte des PG-MATE Systems die semantische Korrektheit einiger empfangener Live-Daten der mobilen Sensorbox, die dem PG-MATE Systems durch LABSKAUS bereitgestellt und ein plausibles Ergebnis anhand der vorliegenden Situation der Realwelt validiert wird.

Testdurchführung.

Testspezifikation:

Zur Vorbereitung des Systemtests werden die Komponenten des Systems, EPD, Route Planning, Path Planning, DOI und der Controller gestartet. Zusätzlich werden Haggis und der NoGoSolver gestartet und wie die Testumgebung mit dem System verbunden. Mit den geplanten Testszenarien werden die meisten funktionalen Anforderungen an das System verifiziert und zusätzlich werden die COLREG Regeln 13-17 abgedeckt. Die in den Szenarien benötigten Schiffe werden mit Hilfe von Haggis simulativ in die Testumgebung eingespeist.

Testkonfiguration:

Für die Szenarien des Systemtests erhält das zu testende Schiff eine Route in der jedes Szenario durchgespielt wird. Diese Route wurde im Vorfeld des Systemtest bereits zusammengestellt EPD abgespeichert. Im Folgenden werden die Abschnitte für jedes Szenario aufgelistet.

Testszenario	Abschnitt

TS-1	- WP-1:
	  - Lat: 52.697
	  - Long: 8.16988
	- WP-2:
	  - Lat: 51.577
	  - Long: 8.16988

TS-2	- WP-1:
	  - Lat: 51.577
	  - Long: 8.16988
	- WP-2:
	  - Lat: 52.697
	  - Long: 8.16988

TS-3	- WP-1:
	  - Lat: 52.697
	  - Long: 8.16988

	- WP-2:
	  - Lat: 51.577
	  - Long: 8.16988

TS-4	- WP-1:
	  - Lat: 51.577
	  - Long: 8.16988
	- WP-2:
	  - Lat: 52.697
	  - Long: 8.16988

TS-5	- WP-1:
	  - Lat: 52.697
	  - Long: 8.16988
	- WP-2:
	  - Lat: 51.577
	  - Long: 8.16988

Desweiteren müssen für die Szenarien 2-5 die Routen der simulierten Schiffe konfiguriert werden, diese wurden in Abhängigkeit zur Route des zu testenden Schiffes entwickelt..

Testszenario	Route

TS-2	- WP-1:
	  - Lat: 52.88
	  - Long: 8.17
	- WP-2:
	  - Lat: 51.577
	  - Long: 8.17

TS-3	- WP-1:
	  - Lat: 52.697
	  - Long: 8.17
	- WP-2:
	  - Lat: 51.577
	  - Long: 8.143

TS-4	- WP-1:
	  - Lat: 51,22
	  - Long: 8.22
	- WP-2:
	  - Lat: 53.33
	  - Long: 8.13

TS-5	- WP-1:
	  - Lat: 52.632
	  - Long: 8.1564
	- WP-2:
	  - Lat: 51.322
	  - Long: 8.248

Testszenarien:

ID: TS-1	Route automatisiert abfahren
Beschreibung:	Start im Jadebusen in Wilhelmshaven. Über den Küstenleitstand wird eine Route zum Jade-Weser Port generiert. Diese soll dann vom Schiff im Autopilot gefahren werden, während der eigentliche Steuermann die Fahrt über bereitgestellte Monitore mit Schiffs- und Umweltinformationen überwacht.
Testfälle:	TC-3-Ungültige Route optimieren TC-1-Route mit Schiffsparametern erstellen TC-2-Route optimieren und senden TC-4-Controller Geschwindigkeit TC-5-Controller Wegpunkt TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-11-Path Planning Kommunikation TC-13-Path-Planning berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes

ID: TS-2	Überholmanöver Schiff (COLREG 13)
Beschreibung:	Das Schiff soll im Autopiloten das Schiff erkennen, überholen und auf die ursprünglich geplante Route zurückkehren. <u>COLREG 13:</u> a) Das Schiff, das überholt wird, muss von anderen Schiffen gemieden werden b) Ein Schiff gilt als überholt, sobald das überholte Schiff deren Seitenlichter sieht c) Falls es irgendwelche bedenken gibt, darf nicht überholt werden
Testfälle:	TC-1-Route mit Schiffsparametern erstellen TC-2-Route optimieren und senden TC-4-Controller Geschwindigkeit TC-5-Controller Wegpunkt TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-8.1-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-8.2-Objekterkennung (vor dem eigenen Schiff) bei schlechter Sicht TC-9-Objekterkennung (neben bzw. hinter dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m) TC-10.2-Datenfusion (Größe des Schiffs 20m) TC-11-Path Planning Kommunikation TC-12-Path-Planning berücksichtigung dynamischer und statischer Hindernisse TC-13-Path-Planning berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.1-Path-Planning Überholen

ID: TS-3	Kreuzendes Schiff von rechts (wir müssen ausweichen) (COLREG 15, 16, 17)
Beschreibung:	Das Schiff soll das Fremdschiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. <u>COLREG 15:</u> Falls zwei Schiffe sich kreuzen muss das Schiff, welches sein Gegenüber auf der Steuerbord-Seite hat reagieren und ausweichen
Testfälle:	TC-1-Route mit Schiffsparametern erstellen TC-2-Route optimieren und senden TC-4-Controller Geschwindigkeit TC-5-Controller Wegpunkt TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-8.1-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-8.2-Objekterkennung (vor dem eigenen Schiff) bei schlechter Sicht TC-9-Objekterkennung (neben bzw. hinter dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m) TC-10.2-Datenfusion (Größe des Schiffs 20m) TC-11-Path Planning Kommunikation TC-12-Path-Planning berücksichtigung dynamischer und statischer Hindernisse TC-13-Path-Planning berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.3-Path-Planning Kreuzen von rechts

ID: TS-4	Kreuzendes Schiff von links (COLREG 15, 16, 17)
Beschreibung:	Das Schiff soll das simulierte Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. <u>COLREG 17:</u> a) Ein Schiff verhält sich nicht COLREG-Konform b) Falls das gegenüberliegende Schiff, das ausweichen muss, nicht reagiert, muss das eigene Schiff trotzdem entsprechend reagieren und ausweichen d) Diese Regel befreit Schiffe nicht von ihrer Pflicht, sich von solchen Situationen fern zu halten
Testfälle:	TC-1-Route mit Schiffsparametern erstellen TC-2-Route optimieren und senden TC-4-Controller Geschwindigkeit TC-5-Controller Wegpunkt TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-8.1-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-8.2-Objekterkennung (vor dem eigenen Schiff) bei schlechter Sicht TC-9-Objekterkennung (neben bzw. hinter dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m) TC-10.2-Datenfusion (Größe des Schiffs 20m) TC-11-Path Planning Kommunikation TC-12-Path-Planning berücksichtigung dynamischer und statischer Hindernisse TC-13-Path-Planning berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes

	TC-14.4.1-Path-Planning Kreuzen von links (Kurs beibehalten) TC-14.4.2-Path-Planning Kreuzen von links (Ausweichen)

ID: TS-5	Entgegenkommendes Schiff (COLREG 14, 16, 17)
Beschreibung:	Das Schiff soll das entgegenkommende Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. Danach übernimmt der Kapitän kurzzeitig die Kontrolle mithilfe der Virtual Handles. <u>COLREG 14:</u> a) Beide Schiffe weichen auf der Steuerbordseite aus b) Diese Situation gilt falls vermutet wird, dass ein gegenüberliegendes Schiff auf das Eigene zufährt oder sich annähert c) Falls die Vermutung besteht, dass diese Situation gilt, muss ausgewichen werden
Testfälle:	TC-1-Route mit Schiffsparametern erstellen TC-2-Route optimieren und senden TC-4-Controller Geschwindigkeit TC-5-Controller Wegpunkt TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-7-Virtual Handles TC-8.1-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht

	TC-8.2-Objekterkennung (vor dem eigenen Schiff) bei schlechter Sicht TC-9-Objekterkennung (neben bzw. hinter dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m) TC-10.2-Datenfusion (Größe des Schiffs 20m) TC-11-Path Planning Kommunikation TC-12-Path-Planning berücksichtigung dynamischer und statischer Hindernisse TC-13-Path-Planning berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.2-Path-Planning Entgegenkommend

Testfälle:

ID:	TC-1
Test Titel:	Route mit Schiffsparametern erstellen
Beschreibung:	Zusammenstellung einer Route in der EPD mit Schiffsparametern.
Funktionale Anforderungen:	F-EPD-2 F-EPD-5
Eingehende-Daten:	-
Testschritte:	TCS-01: Wegpunktwerkzeug in der EPD auswählen und eine Route auf der Karte zusammenstellen TCS-01a: Route im Routenmanager der EPD per Doppelklick öffnen und ggfs. Geschwindigkeiten (max) der einzelnen Wegpunkte setzen TCS-02: Schiffsparameter in der “PG Mate”-Einstellungsseite der EPD hinterlegen und speichern
Erwartetes Ergebnis:	Verbindet die gesetzten Wegpunkte zu einer Route und nimmt die Schiffsparameter entgegen.
Eingetretenes Ergebnis:	Die Route wurde erstellt
Erfolg/Misserfolg:	Erfolg
Testmethode:	Anforderungstest

ID:	TC-2
Test Titel:	Route optimieren und senden
Beschreibung:	Auswahl einer Route, die anschließend optimiert und an das Schiff gesendet wird.
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5 F-EPD-6 F-EPD-7
Eingehende-Daten:	1. in Route-Planning eingehend: - RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit zwischen den Wegpunkten 2. in EPD eingehend (im Anschluss an 1): - RTZ-Route vom Route-Planning eingehend via HTTP in die EPD, bestehend aus berechneten Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit zwischen den Wegpunkten - “NoGoAreas” als JSON-Objekte via HTTP vom Route-Planning, die die nicht zu befahrenden Bereiche darstellen (ermittelt vom NoGoSolver) - Informationen des “Grids” via HTTP des Route-Plannings als JSON, beinhaltet alle dynamisch berechneten Zellen (blockiert oder nicht blockiert; dient als Basis zur Routenberechnung) 3. in Path-Planning eingehend (im Anschluss an 2):

	- RTZ-Route von der EPD via HTTP, beinhaltet alle zuvor vom Route-Planning berechneten Wegpunkte inklusive der Geschwindigkeiten zwischen den Wegpunkten
Testschritte:	TCS-01: Sicher gehen, dass Route ausgewählt ist und „Route optimieren“ klicken TCS-02: Nach Erhalt der optimierten Route, diese mit Hilfe des Kartenmaterials, den vom Route-Planning berechneten Grid-Zellen und Polygonen des NoGoSolvers auf Fehler überprüfen TCS-03: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und auf “Route starten” klicken
Erwartetes Ergebnis:	Das Path-Planning erhält eine vom Route-Planning optimierte Route als RTZ-Route via HTTP, die die gleichen Start- und Zielpunkte wie die Originalroute hat und Hindernisse, visualisiert durch “NoGo-Areas”, meidet.
Eingetretenes Ergebnis:	Die optimierte Route wurde erstellt
Erfolg/Misserfolg:	Erfolg
Testmethode:	Anforderungstest

ID:	TC-3
Test Titel:	Ungültige Route optimieren
Beschreibung:	Zusammenstellung einer Route die einen Wegpunkt auf Land bzw. einer NoGoArea besitzt. Anschließend soll der Versuch gestartet werden, diese zu optimieren.
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5
Eingehende-Daten:	1. in Route-Planning eingehend: - RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit zwischen den Wegpunkten 2. in EPD eingehend (im Anschluss an 1): - Fehlermeldung als JSON-Objekt via HTTP, welches in der EPD das Darstellen einer Fehlermeldung auslöst
Testschritte:	TCS-01: Wegpunktwerkzeug auswählen und auf der Karte eine Route zusammenstellen, wobei ein Wegpunkt auf Land zu setzen ist (Vorgabe Start-Ziel, genau sagen wo und wie viele Wegpunkte zu setzen sind) TCS-02: Sichergehen, dass Route ausgewählt ist und „Route optimieren“ auswählen TCS-03: Fehlermeldung mit “OK” bestätigen
Erwartetes Ergebnis:	Rückmeldung als Hinweisfenster der EPD: “Es wurde keine Route gefunden!”
Eingetretenes Ergebnis:	Fehlermeldung wurde angezeigt

Erfolg/Misserfolg:	Erfolg
Testmethode:	Anforderungstest

ID:	TC-4
Test Titel:	Controller Geschwindigkeit
Beschreibung:	Überwachung der vorgegebenen Geschwindigkeit des Controllers
Funktionale Anforderungen:	F-CA-PP-9
Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - Rudder - Angle - Shipspeed Die nächsten 2 Wegpunkte vom Path-Planning
Testschritte:	TCS-01:Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen der Geschwindigkeit des Schiffes mit MATE Control.
Erwartetes Ergebnis:	Schiff fährt mit der vorgegebenen Geschwindigkeit
Eingetretenes Ergebnis:	Das Schiff pendelt sich in der vorgegebenen Geschwindigkeit ein
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-5
Test Titel:	Controller Wegpunkt
Beschreibung:	Der Controller empfängt vom Path-Planning einen Wegpunkt. Das Schiff fährt daraufhin in die Richtung des Wegpunktes.
Funktionale Anforderungen:	F-CA-PP-9
Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - Rudder - Angle - Shipspeed Die nächsten 2 Wegpunkte vom Path-Planning
Testschritte:	TCS-01:Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen ob der Controller den empfangenen Wegpunkt anfährt mit MATE Control.
Erwartetes Ergebnis:	Schiff fährt in die Richtung des vom Controller empfangenen Wegpunktes
Eingetretenes Ergebnis:	Da das Path Planning nicht funktionsfähig war, wurden die Wegpunkte manuell hinzugefügt. Das Schiff fuhr mit einer Genauigkeit von 10 Metern auf den Wegpunkt zu
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-6.1
Test Titel:	Anzeige AIS-Daten
Beschreibung:	Anzeige der AIS-Daten in der EPD.
Funktionale Anforderungen:	F-DOI-1 F-DOI-6 F-EPD-1 F-EPD-1.1
Eingehende-Daten:	- Sensor-Daten (kontinuierlich) - Koordinaten erkannter Schiffe/Objekte (AIS, Radar und fusionierte Daten) als JSON via HTTP von der DOI
Testschritte:	TCS-01: Überprüfung von AIS Daten anhand der EPD-NMEA0183-Sensor-Funktionalität und MATE Control.
Erwartetes Ergebnis:	Anzeige der AIS-Daten in einem Layer der EPD.
Eingetretenes Ergebnis:	Die AIS Daten wurden in dem DOI-Layer der EPD angezeigt
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-6.2
Test Titel:	Anzeige Radar-Daten
Beschreibung:	Anzeige der Radar-Daten in der EPD.
Funktionale Anforderungen:	F-DOI-2 F-EPD-1 F-EPD-1.2
Eingehende-Daten:	- Sensor-Daten (kontinuierlich) - Koordinaten erkannter Schiffe/Objekte (AIS, Radar und fusionierte Daten) als JSON via HTTP von der DOI
Testschritte:	TCS-01: Überprüfung der Radar-Daten anhand von Live-Internet-Daten, des EPD-DOI-Layers und MATE Control
Erwartetes Ergebnis:	Anzeige der Radar-Daten in einem Layer der EPD
Eingetretenes Ergebnis:	Das Radar der Zuse war nicht funktionsfähig
Erfolg/Misserfolg:	Misserfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-6.3
Test Titel:	Anzeige Grid-Daten

Beschreibung:	Anzeige der Grid-Daten in der EPD.
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.3
Eingehende-Daten:	- Routendaten - NoGo-Daten und Griddaten als JSON via HTTP vom Route-Planning (bereits vom Routenberechnen vorhanden)
Testschritte:	TCS-01: Überprüfung der Grid-Daten anhand der EPD Karte und MATE Control
Erwartetes Ergebnis:	Anzeige der Grid-Daten in einem Layer der EPD
Eingetretenes Ergebnis:	Nicht explizit getestet, on shore aber ein Erfolg
Erfolg/Misserfolg:	
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-6.4
Test Titel:	Anzeige Polygone
Beschreibung:	Anzeige der Polygone in der EPD.

Funktionale Anforderungen:	F-EPD-1 F-EPD-1.4
Eingehende-Daten:	- Routendaten - NoGo-Daten und Griddaten als JSON via HTTP vom Route-Planning (bereits vom Routenberechnen vorhanden)
Testschritte:	TCS-01: Überprüfung der Polygone anhand der EPD Karte und MATE Control
Erwartetes Ergebnis:	Anzeige der Polygone in einem Layer der EPD
Eingetretenes Ergebnis:	Nicht explizit getestet, on shore aber ein Erfolg
Erfolg/Misserfolg:	
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-6.5
Test Titel:	Anzeige Schiffsposition
Beschreibung:	Anzeige der Schiffsposition in der EPD.
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.5
Eingehende-Daten:	- Sensor-Daten (kontinuierlich)

Testschritte:	TCS-01: Überprüfung der eigenen Schiffsposition anhand der EPD Karte und MATE Control
Erwartetes Ergebnis:	Anzeige der Schiffsposition in einem Layer der EPD
Eingetretenes Ergebnis:	Das eigene Schiff wurde während der Fahrt in der EPD angezeigt
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-7
Test Titel:	Virtual Handles
Beschreibung:	Anwendung der Virtual Handles während der Fahrt im Autopiloten.
Funktionale Anforderungen:	F-EPD-9 F-EPD-10 F-EPD-11
Eingehende-Daten:	
Testschritte:	TCS-01: Virtual Handles Interface in der EPD öffnen TCS-02: Durch die Nutzung der VirtualHandles Steuerbefehle die Steuerung des Schiffs übernehmen TCS-03: Autopiloten wieder einschalten

Erwartetes Ergebnis:	- Schiff lässt sich per Virtual Handles steuern - Nach Abschalten der Virtual Handles übernimmt der Autopilot wieder das Schiff und fährt die Route weiter
Eingetretenes Ergebnis:	Waren nicht bereit getestet zu werden
Erfolg/Misserfolg:	Misserfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-8.1
Test Titel:	Objekterkennung (vor dem eigenen Schiff) bei guter Sicht
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (vor, neben und hinter dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen. Zusätzlich werden die Schiffe vor unserem Schiff mittels Kamera erkannt
Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.1 F-DOI-4 F-DOI-5.2
Eingehende-Daten:	- Sensordaten des AIS in NMEA-2000 Format, mit erkanntem Schiff welches 3500m Entfernt von der eigenen Position ist. - Sensordaten des Radars in NMEA-2000 konformen Format, mit erkanntem Schiff welches 3500m Entfernt von der eigenen Position ist. - JSON-Objekt über die Position (rechts, links, beides) welche durch Bildverarbeitung erkannt wurde

Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layer an TCS-04: Zeige den Kamerastream in der EPD mit erkannten Schiffe an.
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente korrekt empfangen - Alle Informationen aus den eingehenden Sensordaten von Radar werden von der DOI-Komponente korrekt empfangen - Schiff wird von der Kamera erkannt
Eingetretenes Ergebnis:	Schiffe wurden bei einer Entfernung von circa 400-700 Metern (Augenmaß) erkannt.
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-9
Test Titel:	Objekterkennung (neben bzw. hinter dem eigenen Schiff)
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (vor, neben und hinter dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen. Zusätzlich werden die Schiffe vor unserem Schiff mittels Kamera erkannt
Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.2 F-DOI-3.3

Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format, mit erkanntem Schiff, welches sich in einer Entfernung von bis zu 2sm neben bzw. hinter dem eigenen Schiff befindet - Sensordaten des Radars im NMEA-2000 konformen Format, mit erkanntem Schiff welches sich in einer Entfernung von bis zu 2sm neben bzw. hinter dem eigenen Schiff befindet
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente korrekt empfangen - Alle Informationen aus den eingehenden Sensordaten vom Radar werden von der DOI-Komponente korrekt empfangen
Eingetretenes Ergebnis:	AIS Daten wurden empfangen und auf der EPD angezeigt Radar war technisch bedingt nicht vorhanden
Erfolg/Misserfolg:	Erfolg mit AIS und ungetestet mit RADAR
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-10.1
Test Titel:	Datenfusion (Größe des Schiffs 5m)
Beschreibung:	Die DOI-Komponente muss empfangene AIS-Nachrichten (statisch und dynamisch) und empfangene Radar Tracks untereinander fusionieren (AIS mit AIS und AIS mit Radar)
Anforderung	F-DOI-6 F-DOI-7

Eingehende-Daten:	- Sensordaten des Radars im NMEA-2000 konformen Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten) - Sensordaten des GPS in NMEA-2000 - JSON-Objekt über die Position (rechts, links) welche durch Bildverarbeitung erkannt wurde
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Fusioniere die Schiffe (automatisch) TCS-04: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an TCS-05: Zeige den Kamerastream in der EPD mit erkannten Schiffen an
Erwartetes Ergebnis:	Da Schiff kein AIS besitzt, keine Datenfusion. Von Kamera erkanntes Schiff wird bereits vom Radar aufgezeichnet.
Eingetretenes Ergebnis:	Simulierte Schiffe wurden fusioniert für echte fehlte das Radar
Erfolg/Misserfolg:	Erfolg mit simiulierten Schiffen und ungetestet mit echten Schiffen
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-10.2
Test Titel:	Datenfusion (Größe des Schiffs ab 20m)
Beschreibung:	Die DOI-Komponente muss empfangene AIS-Nachrichten (statisch und dynamisch) und empfangene Radar Tracks untereinander fusionieren (AIS mit AIS und AIS mit Radar). Die DOI-Komponente muss bei allen assoziierten Schiffen in der Umgebung (2 sm Umkreis ums eigene Schiff) die objektspezifischen Parameter Größe, Geschwindigkeit und Position neu berechnen.

Anforderung	F-DOI-6 F-DOI-7 F-DOI-8
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten) - Sensordaten des Radars im NMEA-2000 konformen Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten) - Sensordaten des GPS in NMEA-2000
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Positionen der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Fusioniere die Schiffe (automatisch) TCS-04: Zeige die Rohdaten der Schiffe über den EPD-Layer an TCS-05: Zeige den Kamera-Stream in der EPD mit erkannten Schiffe an
Erwartetes Ergebnis:	AIS-Nachrichten von Schiffen, die sich in einem 2 sm Umkreis vom eigenen Schiff befinden, werden bei gleicher User ID einander zugeordnet. AIS-Nachrichten und Radar-Tracks eines Schiffes im Umkreis von 2sm werden einander zugeordnet, wenn es sich um ein Schiff handelt.
Eingetretenes Ergebnis:	Simulierte Schiffe wurden fusioniert für echte fehlte das Radar
Erfolg/Misserfolg:	Erfolg mit simiulierten Schiffen und ungetestet mit echten Schiffen
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-11
Test Titel:	Kommunikation Path-Planning
Beschreibung:	Entgegennahme der validen globalen Route von der EPD. Anschließend muss die Path-Planning-Komponente kontinuierlich Informationen über alle statischen Hindernisse von dem NoGoSolver anfragen, entgegennehmen und decodieren können. Sowie kontinuierlich Informationen über alle dynamischen Hindernisse (Schiffe) von der DOI-Komponente entgegennehmen und decodieren können. Der anschließend berechnete valide Pfad wird über das Distribution System an den Controller und die EPD geschickt.
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und EPD durch MATE Control TCS-03: Überprüfung der Verbindung zwischen Path-Planning und NoGoSolver durch MATE Control TCS-04: Überprüfung der Verbindung zwischen Path-Planning und DOI-Komponente durch MATE Control TCS-05: Überprüfung der Verbindung zwischen Path-Planning und Controller über das Distribution System durch MATE Control
Erwartetes Ergebnis:	- Die Path-Planning-Komponente sendet und empfängt Daten vom NoGoSolver - Die Path-Planning-Komponente empfängt Daten von der DOI-Komponente - Die Path-Planning-Komponente sendet Daten an den Controller und die EPD über das Distribution System
Eingetretenes Ergebnis:	Solange die Route “optimized” im Namen hat funktioniert die Kommunikation mit der EPD Kommunikation mit dem Controller und dem HUB konnte nicht getestet werden

Erfolg/Misserfolg:	Erfolg mit EPD und ungetestet mit Controller und HUB
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-12
Test Titel:	Path-Planning berücksichtigung dynamischer und statischer Hindernisse
Beschreibung:	Ausweichen eines dynamischen Objektes (Schiff) unter berücksichtigung eines nahen statischen Hindernisses.
Anforderung	F-CA-PP-5
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Wegpunktwerkzeug in der EPD auswählen und auf der Karte eine Route zusammenstellen die entlang statischer Hindernisse führt TCS-02: Schiffsparameter(minimal_turn_radius, maximum_turning_angle) in den configuration.properties einstellen TCS-03: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-04: Den berechneten Pfad der Path-Planning überwachen
Erwartetes Ergebnis:	- Das System weicht dem Dynamischen Objekt aus ohne dabei mit dem statischen Hinderniss zu kolidieren
Eingetretenes Ergebnis:	

Erfolg/Misserfolg:	
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-13
Test Titel:	Path-Planning berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes
Beschreibung:	Ausweichen eines dynamischen Objektes (Schiff) unter berücksichtigung der physikalischen Eigenschaften für das im Test eingesetzte Schiff (ZUSE).
Anforderung	F-CA-PP-6
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Schiffsparameter(minimal_turn_radius, maximum_turning_angle) in den configuration.properties einstellen TCS-02: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-03: Überwachen des berechneten Pfades auf Einhaltung der Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes
Erwartetes Ergebnis:	- Der berechnete Pfad steht nicht im Konflikt mit den physikalischen Eigenschaften des im Test eingesetzten Schiffs
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	

Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-14.1
Test Titel:	Path-Planning Überholen
Beschreibung:	Überholen eines dynamischen Objektes (Schiff) unter berücksichtigung der COLREG Regel 13.
Anforderung	F-CA-PP-8
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen (Überholen, COLREG 13) TCS-05b: Das Ausweichen des Schiffes überwachen (Entgegenkommen, COLREGs 14, 16) TCS-05c: Das Ausweichen des Schiffes überwachen (Kreuzen, COLREGs 15, 16) TCS-05d: Das Ausweichen des Schiffes überwachen (Zwangsweises Ausweichen, COLREGs 15, 17)
Erwartetes Ergebnis:	- Die COLREG Regel 13 wurde eingehalten
Eingetretenes Ergebnis:	

Erfolg/Misserfolg:	
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-14.2
Test Titel:	Path-Planning Entgegenkommend
Beschreibung:	Ausweichen eines entgegenkommenden dynamischen Objektes (Schiff) unter berücksichtigung der COLREGs Regeln 14, 16, 17.
Anforderung	F-CA-PP-8
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen
Erwartetes Ergebnis:	- Die COLREGs Regeln 14, 16, 17 wurden eingehalten
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-14.3
Test Titel:	Path-Planning Kreuzen von rechts
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von rechts gekreuzt unter berücksichtigung der COLREGs Regeln 15, 16, 17.
Anforderung	F-CA-PP-8
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen (Kreuzen, COLREGs 15, 16)
Erwartetes Ergebnis:	- Die COLREG Regeln 15, 16, 17 wurden eingehalten
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-14.4.1
Test Titel:	Path-Planning Kreuzen von links (Kurs beibehalten)
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von links gekreuzt unter berücksichtigung der COLREGs Regeln 15, 16, 17.
Anforderung	F-CA-PP-8
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Überwachung des Kreuz-Manövers
Erwartetes Ergebnis:	- Die COLREGs Regeln 15,16,17 wurde eingehalten
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-14.4.2
Test Titel:	Path-Planning Kreuzen von links (Ausweichen)

Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von links gekreuzt, das andere Schiff verletzt die COLREGs Regeln und weicht nicht aus, daraufhin muss das zu testende Schiff zwangsweise Ausweichen unter berücksichtigung der COLREGs Regeln 15, 16, 17.
Anforderung	F-CA-PP-8
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Überwachung des erzwungenen Ausweichens während des Kreuz-Manövers
Erwartetes Ergebnis:	- Die COLREGs Regeln 15,16,17 wurde eingehalten
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	Szenariobasiertes Blackbox-Testing

Fehlerinjektion:

Geplant für den Systemtest:

ID:	TC-FI-2

Test Titel:	Verbindungsabbruch Navibox
Beschreibung:	Die Verbindung zwischen dem zu testenden System und der Navibox wird einmalig für ein längeres Intervall unterbrochen.

Eingehende-Daten:	- Schiffssensorinformationen
	  - Schiffsposition
	  - Schiffsrichtung
	  - AIS
	  - (Radar)

Testschritte:	TCS-01: Verbindung zwischen dem zu testenden System und Labskaus abbrechen TCS-02: Überwachung des zu testenden Systems mit MATE-Control TCS-03: Verbindung zwischen dem zu testenden System und Labskaus nach 2 Minuten wieder aufbauen
Erwartetes Ergebnis:	- Das Schiff drosselt die Geschwindigkeit auf null
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-3
Test Titel:	Gemessener Ruderwinkel außerhalb der Grenzen
Beschreibung:	Der Controller empfängt den Ruderwinkel außerhalb der spezifizierten Grenzen

Eingehende-Daten:	- Ruderwinkel per NMEA 2000 (an Controller)
Testschritte:	TCS-01: Abfangen der Ruderwinkel-Nachrichten mit MATE-Control TCS-02: Ändern und senden der Ruderwinkel-Nachricht zu einem Wert außerhalb der spezifizierten Grenzen (180° und -180°) TCS-03: Überwachung des Systems mit MATE-Control
Erwartetes Ergebnis:	- Ist uns unbekannt
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-4
Test Titel:	Gemessene negative Drehzahl
Beschreibung:	Der Controller erhält negative Drehzahlen.
Eingehende-Daten:	- Drehzahl per NMEA 2000
Testschritte:	TCS-01: Abfangen der Drehzahl-Nachrichten TCS-02: Ändern und senden der Drehzahl-Nachrichten (negative Werte) TCS-03: Überwachung des Systems mittels MATE-Control

Erwartetes Ergebnis:	- Schiff beschleunigt
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-7
Test Titel:	Nahes Schiff verschwindet
Beschreibung:	Ein mit AIS und/oder Radar identifiziertes Schiff in unmittelbarer Nähe zum eigenen Schiff verschwindet aus dem Grid der Path-Planning-Komponente
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten)
Testschritte:	TCS-01: In MATE-Control den Error-Injector öffnen und das nahe Schiff löschen TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Schiff fährt weiter
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-10
Test Titel:	Bekanntes Schiff springt auf eigene Position
Beschreibung:	Ein mit AIS und oder Radar identifiziertes Schiff in der Umgebung der eigenen Schiffes springt auf die Position des eigenen Schiffes
Eingehende-Daten:	
Testschritte:	TCS-01: Mittels MATE-Control Error-Injection die Schiffsposition eines anderen Schiffes auf die eigene Position setzen TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Das Schiff drosselt die Geschwindigkeit auf null
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-12
Test Titel:	Schiff wechselt AIS-MMSI
Beschreibung:	Ein mit AIS identifiziertes Schiff ändert dauernd seine AIS-MMSI

Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten)
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die MMSI des Schiffes laufend ändern TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Schiffe werden nicht gefused
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

Weitere Fehlerinjektionen:

ID:	TC-FI-1
Test Titel:	Verbindungsabbrüche Navibox
Beschreibung:	Die Verbindung zwischen dem zu testenden System und der Navibox wird in kurzen Abständen abgebrochen und wieder aufgebaut.
Eingehende-Daten:	- Was kommt alles aus der Navibox in unser System?
Testschritte:	TCS-01: Verbindung zwischen dem zu testenden System und Labskaus abbrechen TCS-02: Verbindung zwischen dem zu testenden System und Labskaus wieder aufbauen TCS-03: Zehnfache Wiederholung der TCS-01 und TCS-02 in schneller Abfolge

Erwartetes Ergebnis:	- Schiff drosselt Geschwindigkeit auf null
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-5
Test Titel:	Unvollständige AIS-Daten
Beschreibung:	Einspeisung unvollständiger AIS-Nachrichten die an die DOI-Komponente gesendet werden.
Eingehende-Daten:	- Sensordaten des AIS in NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm
Testschritte:	TCS-01: Einspeisung unvollständiger AIS-Nachrichten mittels MATE-Control TCS-02: TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	-
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	

Testmethode:	

ID:	TC-FI-6
Test Titel:	Unvollständige Radar-Daten
Beschreibung:	Einspeisung unvollständiger Radar-Nachrichten die an die DOI-Komponente gesendet werden.
Eingehende-Daten:	- Sensordaten des Radars im NMEA-2000 konformen Format in einem Umkreis ums eigenen Schiff von bis zu 2sm
Testschritte:	TCS-01: Einspeisung unvollständiger Radar-Nachrichten mittels MATE-Control TCS-02: TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	-
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-8
Test Titel:	Entferntes Schiff springt in unmittelbare Nähe

Beschreibung:	Ein mit AIS und oder Radar identifiziertes Schiff in einiger Entfernung zum eigenen Schiff springt plötzlich in unmittelbare Nähe vor das eigene Schiff
Eingehende-Daten:	- Sensordaten des AIS in NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten)
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die Position eines entfernten Schiffes vor die eigene Position setzen TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Schiff leitet Ausweichen ein
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-9
Test Titel:	Schiff im Umkreis wechselt ständig die Position
Beschreibung:	Ein mit AIS und oder Radar identifiziertes Schiff in der Umgebung zum eigenen Schiff springt zwischen mehreren Positionen um das Schiff umher
Eingehende-Daten:	- Sensordaten des AIS in NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten)
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die Position eines entfernten Schiffes dauern ändern TCS-02: Überwachung des Systems mittels MATE-Control

Erwartetes Ergebnis:	- Schiff fährt normal seine Route weiter
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-11
Test Titel:	Unbekanntes Schiff springt auf eigene Position
Beschreibung:	Ein nicht vorher identifiziertes Schiff springt auf die Position des eigenen Schiffes
Eingehende-Daten:	
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die Position eines unbekannten Schiffes vor die eigene Position setzen TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Das Schiff drosselt die Geschwindigkeit auf null
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

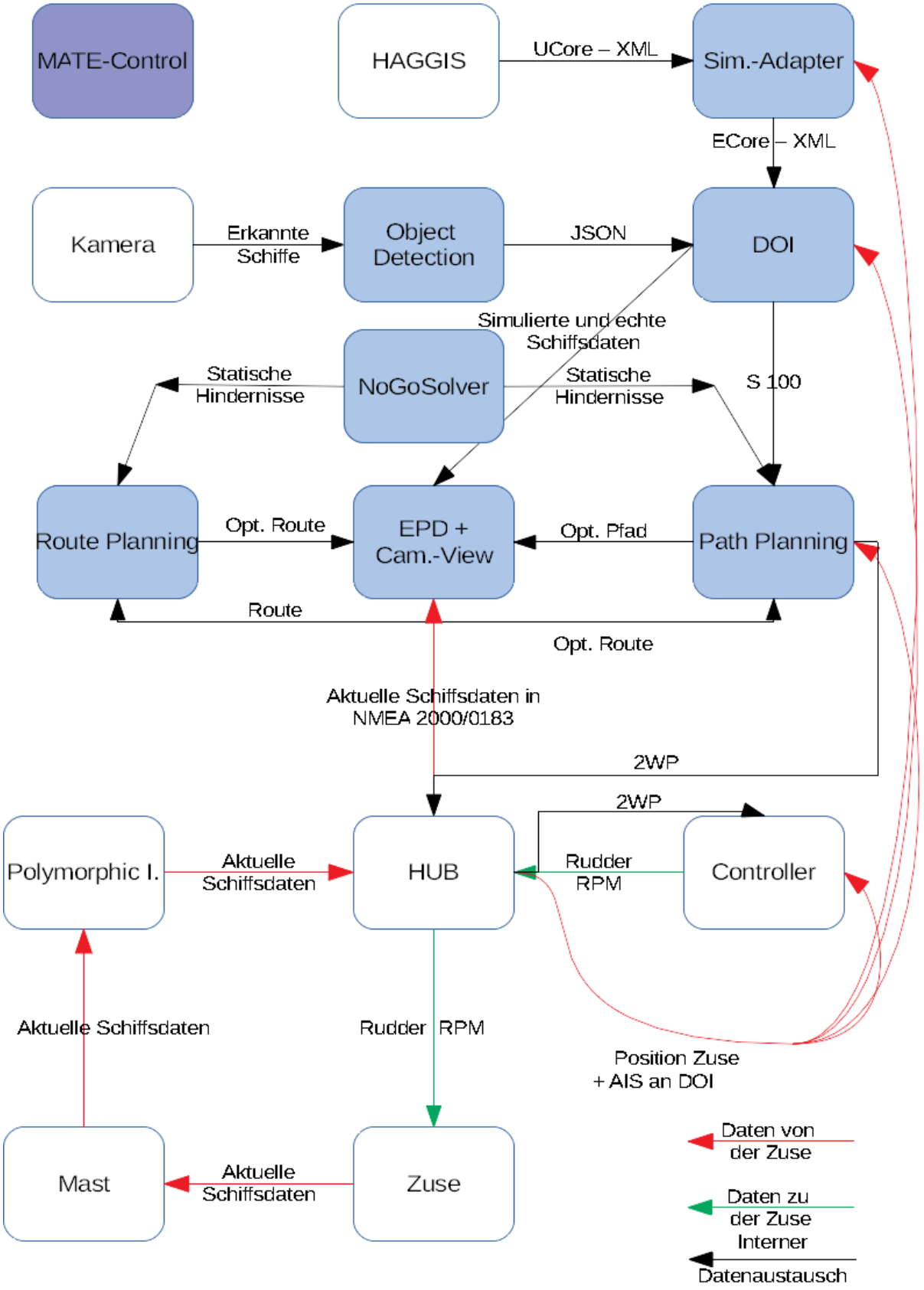
ID:	TC-FI-13
Test Titel:	Statisches Hindernis taucht auf eigener Position auf
Beschreibung:	Während der Fahrt taucht ein statisches Hindernis auf der Position des eigenen Schiffes auf.
Eingehende-Daten:	- Polygone vom NoGoSolver per JSON
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die Position eines statischen Hindernisses auf die eigene Position setzen TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Schiff drosselt Geschwindigkeit auf Null
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-14
Test Titel:	Statisches Hindernis taucht vor eigener Position auf
Beschreibung:	Während der Fahrt taucht ein statisches Hindernis unmittelbar vor der Position des eigenen Schiffes auf.
Eingehende-Daten:	- Polygone vom NoGoSolver per JSON
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die Position eines statischen Hindernis vor die eigene Position setzen TCS-02: Überwachung des Systems mittels MATE-Control

Erwartetes Ergebnis:	- Schiff weicht dem Hindernis aus
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-15
Test Titel:	Ausfall RabbitMQ
Beschreibung:	RabbitMQ fällt während der Fahrt aus.
Eingehende-Daten:	
Testschritte:	TCS-01: Verbindungen des zu testenden Systems zu RabbitMQ unterbrechen
Erwartetes Ergebnis:	- System arbeitet nicht mehr Ordnungsgemäß
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

Gesamtübersicht:



Systemtest vom 05.08.2018

Labskaus-Integrationstest:

Methode.
Zunächst erfolgt ein Systemintegrationstest des PG-MATE Systems und LABSKAUS. Der Systemintegrationstest verifiziert die korrekte Funktionsweise der polymorphen Schnittstelle hinsichtlich einer erfolgten Kommunikationsverbindung, sowie der syntaktischen Korrektheit der Daten. Anschließend beurteilt ein Systemexperte des PG-MATE Systems die semantische Korrektheit einiger empfangener Live-Daten der mobilen Sensorbox, die dem PG-MATE Systems durch LABSKAUS bereitgestellt und ein plausibles Ergebnis anhand der vorliegenden Situation der Realwelt validiert wird.

Testdurchführung.

Testspezifikation:

Zur Vorbereitung des Systemtests werden die Komponenten des Systems, EPD, Route Planning, Path Planning, DOI und der Controller gestartet. Zusätzlich werden Haggis, der Simulations Adapter und der NoGoSolver gestartet und wie die Testumgebung mit dem System verbunden. Mit den geplanten Testszenarien werden die meisten funktionalen Anforderungen an das System verifiziert und zusätzlich werden die COLREG Regeln 13-17 abgedeckt. Die in den Szenarien benötigten Schiffe werden mit Hilfe von Haggis simulativ über den Simulations Adapter in die Testumgebung eingespeist.

Testkonfiguration:

Für die Szenarien des Systemtests erhält das zu testende Schiff eine Route in der jedes Szenario durchgespielt wird. Diese Route wurde im Vorfeld des Systemtest bereits zusammengestellt und in der EPD abgespeichert. Im Folgenden werden die Abschnitte für jedes Szenario aufgelistet.

Testszenario	Abschnitt

TS-1	- WP-1:
	  - Lat: 52.697
	  - Long: 8.16988
	- WP-2:
	  - Lat: 51.577
	  - Long: 8.16988

TS-2	- WP-1:
	  - Lat: 51.577
	  - Long: 8.16988
	- WP-2:
	  - Lat: 52.697
	  - Long: 8.16988

TS-3	- WP-1:
	  - Lat: 52.697
	  - Long: 8.16988

	- WP-2:
	  - Lat: 51.577
	  - Long: 8.16988

TS-4	- WP-1:
	  - Lat: 51.577
	  - Long: 8.16988
	- WP-2:
	  - Lat: 52.697
	  - Long: 8.16988

TS-5	- WP-1:
	  - Lat: 52.697
	  - Long: 8.16988
	- WP-2:
	  - Lat: 51.577
	  - Long: 8.16988

Des Weiteren müssen für die Szenarien 2-5 die Routen der simulierten Schiffe konfiguriert werden, diese wurden in Abhängigkeit zur Route des zu testenden Schiffes entwickelt.

Testszenario	Route

TS-2	- WP-1:
	  - Lat: 52.88
	  - Long: 8.17
	- WP-2:
	  - Lat: 51.577
	  - Long: 8.17

TS-3	- WP-1:
	  - Lat: 52.697
	  - Long: 8.17
	- WP-2:
	  - Lat: 51.577
	  - Long: 8.143

TS-4	- WP-1:
	  - Lat: 51,22
	  - Long: 8.22
	- WP-2:
	  - Lat: 53.33
	  - Long: 8.13

TS-5	- WP-1:
	  - Lat: 52.632
	  - Long: 8.1564
	- WP-2:
	  - Lat: 51.322
	  - Long: 8.248

Testszenarien:

ID: TS-1	Route automatisiert abfahren
Beschreibung:	Start im Jadebusen in Wilhelmshaven. Über den Küstenleitstand wird eine Route zum Jade-Weser Port generiert. Diese soll dann vom Schiff im Autopilot gefahren werden, während der eigentliche Steuermann die Fahrt über bereitgestellte Monitore mit Schiffs- und Umweltinformationen überwacht.
Testfälle:	TC-1-Route mit Schiffsparametern erstellen TC-2.1-Route-Planning Route empfangen und optimieren TC-2.2- EPD-seitiger Empfang einer vom Path-Planning optimierten Route TC-2.3- Empfang einer optimierten Route ausgehend von der EPD an das Path-Planning TC-3-Ungültige Route optimieren TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-11-Path Planning Kommunikation TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes

ID: TS-2	Überholmanöver Schiff (COLREG 13)
Beschreibung:	Das Schiff soll im Autopiloten das Schiff erkennen, überholen und auf die ursprünglich geplante Route zurückkehren. <u>COLREG 13:</u> a) Das Schiff, das überholt wird, muss von anderen Schiffen gemieden werden b) Ein Schiff gilt als überholt, sobald das überholte Schiff deren Seitenlichter sieht c) Falls es irgendwelche bedenken gibt, darf nicht überholt werden
Testfälle:	TC-1-Route mit Schiffsparametern erstellen TC-2.1-Route-Planning Route empfangen und optimieren TC-2.2- EPD-seitiger Empfang einer vom Path-Planning optimierten Route TC-2.3- Empfang einer nicht-optimierten Route ausgehend von der EPD an das Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-15- Schiffserkennung mittels Kamera TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-9-Objekterkennung (neben bzw. hinter dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m) TC-10.2-Datenfusion (Größe des Schiffs 20m) TC-11-Path-Planning Kommunikation TC-12-Path-Planning Berücksichtigung dynamischer und statischer Hindernisse

	TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.1-Path-Planning Überholen

ID: TS-3	Kreuzendes Schiff von rechts (wir müssen ausweichen) (COLREG 15, 16, 17)
Beschreibung:	Das Schiff soll das Fremdschiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. <u>COLREG 15:</u> Falls zwei Schiffe sich kreuzen muss das Schiff, welches sein Gegenüber auf der Steuerbord-Seite hat reagieren und ausweichen
Testfälle:	TC-1-Route mit Schiffsparametern erstellen TC-2.1-Route-Planning Route empfangen und optimieren TC-2.2- EPD-seitiger Empfang einer vom Path-Planning optimierten Route TC-2.3- Empfang einer nicht-optimierten Route ausgehend von der EPD an das Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-15- Schiffserkennung mittels Kamera TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-9-Objekterkennung (neben bzw. hinter dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m) TC-10.2-Datenfusion (Größe des Schiffs 20m)

	TC-11-Path-Planning Kommunikation TC-12-Path-Planning Berücksichtigung dynamischer und statischer Hindernisse TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.3-Path-Planning Kreuzen von rechts

ID: TS-4	Kreuzendes Schiff von links (COLREG 15, 16, 17)
Beschreibung:	Das Schiff soll das simulierte Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. <u>COLREG 17:</u> a) Ein Schiff verhält sich nicht COLREG-Konform b) Falls das gegenüberliegende Schiff, das Ausweichen muss, nicht reagiert, muss das eigene Schiff trotzdem entsprechend reagieren und ausweichen d) Diese Regel befreit Schiffe nicht von ihrer Pflicht, sich von solchen Situationen fern zu halten
Testfälle:	TC-1-Route mit Schiffsparametern erstellen TC-2.1-Route-Planning Route empfangen und optimieren TC-2.2- EPD-seitiger Empfang einer vom Path-Planning optimierten Route TC-2.3- Empfang einer nicht-optimierten Route ausgehend von der EPD an das Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen

	TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-15- Schiffserkennung mittels Kamera TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-9-Objekterkennung (neben bzw. hinter dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m) TC-10.2-Datenfusion (Größe des Schiffs 20m) TC-11-Path-Planning Kommunikation TC-12-Path-Planning Berücksichtigung dynamischer und statischer Hindernisse TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.4.1-Path-Planning Kreuzen von links (Kurs beibehalten) TC-14.4.2-Path-Planning Kreuzen von links (Ausweichen)

ID: TS-5	Entgegenkommendes Schiff (COLREG 14, 16, 17)
Beschreibung:	Das Schiff soll das entgegenkommende Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. Danach übernimmt der Kapitän kurzzeitig die Kontrolle mithilfe der Virtual Handles. <u>COLREG 14:</u> a) Beide Schiffe weichen auf der Steuerbordseite aus b) Diese Situation gilt falls vermutet wird, dass ein gegenüberliegendes Schiff auf das Eigene zufährt oder sich annähert c) Falls die Vermutung besteht, dass diese Situation gilt, muss ausgewichen werden

Testfälle:	TC-1-Route mit Schiffsparametern erstellen TC-2.1-Route-Planning Route empfangen und optimieren TC-2.2- EPD-seitiger Empfang einer vom Path-Planning optimierten Route TC-2.3- Empfang einer nicht-optimierten Route ausgehend von der EPD an das Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-7-Virtual Handles TC-15-Schiffserkennung mittels Kamera TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-9-Objekterkennung (neben bzw. hinter dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m) TC-10.2-Datenfusion (Größe des Schiffs 20m) TC-11-Path-Planning Kommunikation TC-12-Path-Planning Berücksichtigung dynamischer und statischer Hindernisse TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.2-Path-Planning Entgegenkommend

Testfälle:

ID:	TC-1
Test Titel:	Route mit Schiffsparametern erstellen
Beschreibung:	Zusammenstellung einer Route in der EPD mit Schiffsparametern.
Funktionale Anforderungen:	F-EPD-2 F-EPD-5
Eingehende-Daten:	-
Testschritte:	TCS-01: Wegpunktwerkzeug in der EPD auswählen und eine Route, bestehend aus zwei Wegpunkten, auf der Karte erstellen Optional: TCS-02a: Route im Routenmanager der EPD per Doppelklick öffnen und ggfs. Geschwindigkeiten (max) der einzelnen Wegpunkte setzen TCS-02: Schiffsparameter in der “PG Mate”-Einstellungsseite der EPD hinterlegen und speichern
Erwartetes Ergebnis:	Verbindet die gesetzten Wegpunkte zu einer Route und nimmt die Schiffsparameter entgegen.
Eingetretenes Ergebnis:	Die Route wurde erstellt
Erfolg/Misserfolg:	Erfolg
Testmethode:	Anforderungstest

ID:	TC-2.1
Test Titel:	Route-Planning Route empfangen und optimieren
Beschreibung:	Auswahl einer Route, die anschließend optimiert wird
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5
Eingehende-Daten:	in Route-Planning eingehend: - RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit zwischen den Wegpunkten
Testschritte:	TCS-01: nicht-optimierte Route von der EPD an das Route Planning senden TCS-02: optimierte Route in der EPD empfangen
Erwartetes Ergebnis:	Das Route-Planning sendet eine optimierte Route an die EPD.
Eingetretenes Ergebnis:	Die optimierte Route wurde erstellt und an die EPD gesendet
Erfolg/Misserfolg:	Erfolg
Testmethode:	Anforderungstest

ID:	TC-2.2
Test Titel:	EPD-seitiger Empfang einer vom Route-Planning optimierten Route

Beschreibung:	Empfang einer optimierten Route des Path-Planning auf Basis einer Auswahl einer Route
Funktionale Anforderungen:	F-EPD-6 F-EPD-7
Eingehende-Daten:	1. in EPD eingehend: - RTZ-Route vom Route-Planning eingehend via HTTP in die EPD, bestehend aus berechneten Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit zwischen den Wegpunkten
Testschritte:	TCS-01: Sicher gehen, dass Route ausgewählt ist und „Route optimieren“ klicken TCS-02: Nach Erhalt der optimierten Route, überprüfen, ob diese mit einem „optimized“ im Routenname im Routenmanager der EPD abgespeichert wird und sichtbar in der Karte dargestellt wird.
Erwartetes Ergebnis:	Das Path-Planning erhält eine vom Route-Planning optimierte Route als RTZ-Route via HTTP, die die gleichen Start- und Zielpunkte wie die Originalroute hat und Hindernisse, visualisiert durch “NoGo-Areas”, meidet.
Eingetretenes Ergebnis:	Die optimierte Route wurde empfangen und in der EPD gespeichert.
Erfolg/Misserfolg:	Erfolg
Testmethode:	Anforderungstest

ID:	TC-2.3
Test Titel:	Empfang einer optimierten Route ausgehend von der EPD an das Path-Planning
Beschreibung:	Eine in der EPD gespeicherte optimierte Route wird zum Start der Fahrt an das Path-Planning gesendet.
Funktionale Anforderungen:	F-EPD-6 F-EPD-7
Eingehende-Daten:	in Path-Planning eingehend - RTZ-Route von der EPD via HTTP, beinhaltet alle zuvor vom Route-Planning berechneten Wegpunkte inklusive der Geschwindigkeiten zwischen den Wegpunkten - Konfigurierte Schiffsinformationen aus der EPD als JSON-Objekte via HTTP
Testschritte:	TCS-01: Schiffsinformationen in der Einstellungsmaske für Schiffsinformationen in der EPD setzen. TCS-02: Sicher gehen, dass Route ausgewählt ist und in der EPD „Route optimieren“ klicken. TCS-03: Prüfen, ob Route im Path-Planning eingeht.
Erwartetes Ergebnis:	Das Path-Planning erhält eine vom Route-Planning optimierte Route als RTZ-Route via HTTP, die die gleichen Start- und Zielpunkte wie die Originalroute hat und Hindernisse, visualisiert durch “NoGo-Areas”, meidet.
Eingetretenes Ergebnis:	Das PP hat eine nicht optimierte Route von der EPD als RTZ-Route empfangen.
Erfolg/Misserfolg:	Erfolg

Testmethode:	Anforderungstest

ID:	TC-3
Test Titel:	Ungültige Route optimieren
Beschreibung:	Zusammenstellung einer Route die einen Wegpunkt auf Land bzw. einer NoGoArea besitzt. Anschließend soll der Versuch gestartet werden, diese zu optimieren.
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5
Eingehende-Daten:	1. in Route-Planning eingehend: - RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit zwischen den Wegpunkten 2. in EPD eingehend (im Anschluss an 1): - Fehlermeldung als JSON-Objekt via HTTP, welches in der EPD das Darstellen einer Fehlermeldung auslöst
Testschritte:	TCS-01: Wegpunktwerkzeug auswählen und auf der Karte eine Route zusammenstellen, wobei ein Wegpunkt auf Land zu setzen ist (Vorgabe Start-Ziel, genau sagen wo und wie viele Wegpunkte zu setzen sind) TCS-02: Sichergehen, dass Route ausgewählt ist und „Route optimieren“ auswählen TCS-03: Fehlermeldung mit “OK” bestätigen

Erwartetes Ergebnis:	Rückmeldung als Hinweisfenster der EPD: “Es wurde keine Route gefunden!”
Eingetretenes Ergebnis:	Fehlermeldung wurde angezeigt
Erfolg/Misserfolg:	Erfolg
Testmethode:	Anforderungstest

ID:	TC-4
Test Titel:	Controller Geschwindigkeit
Beschreibung:	Überwachung der vorgegebenen Geschwindigkeit des Controllers
Funktionale Anforderungen:	F-CA-PP-9
Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - Rudder - Angle - Speed Die nächsten 2 Wegpunkte vom Path-Planning
Testschritte:	TCS-01:Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen der Geschwindigkeit des Schiffes mit MATE Control.
Erwartetes Ergebnis:	Schiff fährt mit der vorgegebenen Geschwindigkeit
Eingetretenes Ergebnis:	Controller hat nicht funktioniert

Erfolg/Misserfolg:	Misserfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-5.1
Test Titel:	Controller Wegpunkt vom Path-Planning empfangen
Beschreibung:	Der Controller empfängt vom Path-Planning einen Wegpunkt.
Funktionale Anforderungen:	F-CA-PP-9
Eingehende-Daten:	Die nächsten 2 Wegpunkte vom Path-Planning
Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen ob der Controller den gesendeten Wegpunkt von Path-Planning empfängt mit MATE Control.
Erwartetes Ergebnis:	Controller empfängt nächsten anzufahrenden Wegpunkt von der Path-Planning Komponente
Eingetretenes Ergebnis:	Controller hat nicht funktioniert
Erfolg/Misserfolg:	Misserfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-5.2
Test Titel:	Controller Wegpunkt abfahren
Beschreibung:	Das Schiff kann in Richtung eines im Controller hinzugefügten Wegpunkts fahren.
Funktionale Anforderungen:	
Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - Rudder - Angle - Speed Die nächsten 2 Wegpunkte
Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen ob der Controller den eingegebenen Wegpunkt anfährt mit MATE Control.
Erwartetes Ergebnis:	Schiff fährt in die Richtung des im Controller gegebenen Wegpunktes
Eingetretenes Ergebnis:	Controller hat nicht funktioniert
Erfolg/Misserfolg:	Misserfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-6.1
Test Titel:	Anzeige AIS-Daten

Beschreibung:	Anzeige der AIS-Daten in der EPD.
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.1
Eingehende-Daten:	- Sensor-Daten (kontinuierlich) - Koordinaten erkannter Schiffe/Objekte (AIS, Radar und fusionierte Daten) als JSON via HTTP von der DOI
Testschritte:	TCS-01: Überprüfung von AIS Daten anhand der EPD-NMEA0183-Sensor-Funktionalität und MATE Control.
Erwartetes Ergebnis:	Anzeige der AIS-Daten in einem Layer der EPD.
Eingetretenes Ergebnis:	Die AIS Daten wurden in dem DOI-Layer der EPD angezeigt
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-6.2
Test Titel:	Anzeige Radar-Daten
Beschreibung:	Anzeige der Radar-Daten in der EPD.
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.2

Eingehende-Daten:	- Sensor-Daten (kontinuierlich) - Koordinaten erkannter Schiffe/Objekte (AIS, Radar und fusionierte Daten) als JSON via HTTP von der DOI
Testschritte:	TCS-01: Überprüfung der Radar-Daten anhand von Live-Internet-Daten, des EPD-DOI-Layers und MATE Control
Erwartetes Ergebnis:	Anzeige der Radar-Daten in einem Layer der EPD
Eingetretenes Ergebnis:	Die Radar Daten wurden in dem DOI-Layer der EPD angezeigt
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-6.3
Test Titel:	Anzeige Grid-Daten
Beschreibung:	Anzeige der Grid-Daten in der EPD.
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.3
Eingehende-Daten:	- Routendaten - NoGo-Daten und Griddaten als JSON via HTTP vom Route-Planning (bereits vom Routenberechnen vorhanden)

Testschritte:	TCS-01: Überprüfung der Grid-Daten anhand der EPD Karte
Erwartetes Ergebnis:	Anzeige der Grid-Daten in einem Layer der EPD
Eingetretenes Ergebnis:	Die Grid-Daten wurden in der EPD angezeigt
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-6.4
Test Titel:	Anzeige Polygone
Beschreibung:	Anzeige der Polygone in der EPD.
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.4
Eingehende-Daten:	- Routendaten - NoGo-Daten und Griddaten als JSON via HTTP vom Route-Planning (bereits vom Routenberechnen vorhanden)
Testschritte:	TCS-01: Überprüfung der Polygone anhand der EPD Karte

Erwartetes Ergebnis:	Anzeige der Polygone in einem Layer der EPD
Eingetretenes Ergebnis:	Die Polygone wurden in der EPD angezeigt
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-6.5
Test Titel:	Anzeige Schiffsposition
Beschreibung:	Anzeige der Schiffsposition in der EPD.
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.5
Eingehende-Daten:	- Sensor-Daten (kontinuierlich)
Testschritte:	TCS-01: Überprüfung der eigenen Schiffsposition anhand der EPD Karte und MATE Control
Erwartetes Ergebnis:	Anzeige der Schiffsposition in einem Layer der EPD
Eingetretenes Ergebnis:	Das eigene Schiff wurde während der Fahrt in der EPD angezeigt
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-7
Test Titel:	Virtual Handles
Beschreibung:	Anwendung der Virtual Handles während der Fahrt im Autopiloten.
Funktionale Anforderungen:	F-EPD-9 F-EPD-10 F-EPD-11
Eingehende-Daten:	Steuerkommandos (RPM und Ruder-Winkel) über die Webbasierte Oberfläche der VirtualHandles (View in der EPD)
Testschritte:	TCS-01: Virtual Handles Interface in der EPD öffnen TCS-02: Virtual Handles-Steuerung über die „Power“-Schaltfläche (oben rechts) einschalten. TCS-03: Durch die Nutzung der VirtualHandles Steuerbefehle die Steuerung des Schiffs übernehmen TCS-04: Virtual Handles-Steuerung über erneutes Klicken der „Power“-Schaltfläche (oben rechts) ausschalten.
Erwartetes Ergebnis:	- VirtualHandles senden Steuerbefehle (zyklisch und bei Änderung der Einstellgrößen) - Schiff lässt sich per Virtual Handles steuern - Nach Abschalten der Virtual Handles wird die Steuerung wieder abgegeben.
Eingetretenes Ergebnis:	Der Controller konnte die gesendeten Steuerbefehle nicht verarbeiten
Erfolg/Misserfolg:	Misserfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-8
Test Titel:	Objekterkennung (vor dem eigenen Schiff) bei guter Sicht
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (vor, neben und hinter dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen. Zusätzlich werden JSON-Objekte über http von erkannten Schiffe mittels Bilderkennung von der DOI empfangen
Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.1 F-DOI-4 F-DOI-5.2
Eingehende-Daten:	- Sensordaten des AIS in NMEA-2000 Format, mit erkanntem Schiff welches 3500m entfernt von der eigenen Position ist. - Sensordaten des Radars in NMEA-2000 konformen Format, mit erkanntem Schiff welches 3500m entfernt von der eigenen Position ist. - JSON-Objekt über die Position (rechts, links, beides) welche durch Bildverarbeitung erkannt wurde
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layer an und prüfe, ob umliegenden Schiffe korrekt angezeigt werden.

Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente korrekt empfangen - Alle Informationen aus den eingehenden Sensordaten von Radar werden von der DOI-Komponente korrekt empfangen - Erkannte Schiffe von der Object-Detection-Komponente werden korrekt empfangen
Eingetretenes Ergebnis:	AIS- und Radarnachrichten in NMEA 2000 wurden über UDP korrekt empfangen. Erkannte Schiffe von der Object-Detection-Komponente wurden korrekt empfangen.
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-9
Test Titel:	Objekterkennung (neben bzw. hinter dem eigenen Schiff)
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (vor, neben und hinter dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen. Zusätzlich werden die Schiffe vor unserem Schiff mittels Kamera erkannt
Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.2 F-DOI-3.3
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format, mit erkanntem Schiff, welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet - Sensordaten des Radars im NMEA-2000 konformen Format, mit erkanntem Schiff welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet

Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente korrekt empfangen - Alle Informationen aus den eingehenden Sensordaten vom Radar werden von der DOI-Komponente korrekt empfangen
Eingetretenes Ergebnis:	AIS und Radar Daten wurden korrekt empfangen
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-10.1
Test Titel:	Datenfusion (Größe des Schiffs bis 20m)
Beschreibung:	Die DOI-Komponente muss empfangene AIS-Nachrichten (statisch und dynamisch) und empfangene Radar Tracks untereinander fusionieren (AIS mit AIS und AIS mit Radar)
Anforderung	F-DOI-6 F-DOI-7
Eingehende-Daten:	- Sensordaten des Radars im NMEA-2000 konformen Format in einem Umkreis ums eigenen Schiff von bis zu 2 sm (mind. 4 Nachrichten) - Sensordaten des GPS in NMEA-2000 - Sensordaten des Headings in NMEA-2000 - JSON-Objekt über die Position (rechts, links) welche durch Bildverarbeitung erkannt wurde

Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Fusioniere die Schiffe (automatisch) TCS-04: Zeige die fusionierten Schiffe über den EPD-DOI-Layers an
Erwartetes Ergebnis:	Da Schiff kein AIS besitzt, keine Datenfusion. Von Kamera erkanntes Schiff wird bereits vom Radar aufgezeichnet.
Eingetretenes Ergebnis:	Simulierte und echte Schiffe wurden fusioniert
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-10.2
Test Titel:	Datenfusion (Größe des Schiffs ab 20m)
Beschreibung:	Die DOI-Komponente muss empfangene AIS-Nachrichten (statisch und dynamisch) und empfangene Radar Tracks untereinander fusionieren (AIS mit AIS und AIS mit Radar). Die DOI-Komponente muss bei allen assoziierten Schiffen in der Umgebung (2 sm Umkreis ums eigene Schiff) die objektspezifischen Parameter Größe, Geschwindigkeit und Position neu berechnen.
Anforderung	F-DOI-6 F-DOI-7 F-DOI-8

Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Positionsnachrichten und mind. 1 x statische AIS-Nachricht von einem Schiff) - Sensordaten des Radars im NMEA-2000 konformen Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten) - Sensordaten des GPS in NMEA-2000
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Positionen der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Fusioniere die Schiffe (automatisch) TCS-04: Zeige die Rohdaten der Schiffe über den EPD-Layer an TCS-05: Zeige den Kamera-Stream in der EPD mit erkannten Schiffe an
Erwartetes Ergebnis:	AIS-Nachrichten von Schiffen, die sich in einem 2 sm Umkreis vom eigenen Schiff befinden, werden bei gleicher User ID einander zugeordnet. AIS-Nachrichten und Radar-Tracks eines Schiffes im Umkreis von 2sm werden einander zugeordnet, wenn es sich um ein Schiff handelt.
Eingetretenes Ergebnis:	Simulierte und echte Schiffe wurden fusioniert
Erfolg/Misserfolg:	Erfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-11
Test Titel:	Kommunikation Path-Planning
Beschreibung:	Entgegennahme der validen globalen Route von der EPD. Anschließend muss die Path-Planning-Komponente kontinuierlich Informationen über alle statischen Hindernisse von dem NoGoSolver anfragen, entgegennehmen und decodieren können. Sowie kontinuierlich Informationen über alle dynamischen Hindernisse (Schiffe) von der DOI-Komponente entgegennehmen und decodieren können. Der anschließend berechnete valide Pfad wird über das Distribution System an den Controller und die EPD geschickt.
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und EPD TCS-03: Überprüfung der Verbindung zwischen Path-Planning und NoGoSolver TCS-04: Überprüfung der Verbindung zwischen Path-Planning und DOI-Komponente TCS-05: Überprüfung der Verbindung zwischen Path-Planning und Controller über das Distribution System
Erwartetes Ergebnis:	- Die Path-Planning-Komponente sendet und empfängt Daten vom NoGoSolver - Die Path-Planning-Komponente empfängt Daten von der DOI-Komponente - Die Path-Planning-Komponente sendet Daten an den Controller und die EPD über das Distribution System
Eingetretenes Ergebnis:	Kommunikation der Path-Planning-Komponente mit dem NoGo-Solver, der DOI-Komponente, des Controllers und der EPD über das Distribution System funktioniert
Erfolg/Misserfolg:	Erfolg

Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-12
Test Titel:	Path-Planning berücksichtigung dynamischer und statischer Hindernisse
Beschreibung:	Ausweichen eines dynamischen Objektes (Schiff) unter berücksichtigung eines nahen statischen Hindernisses.
Anforderung	F-CA-PP-5
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Wegpunktwerkzeug in der EPD auswählen und auf der Karte eine Route zusammenstellen die entlang statischer Hindernisse führt TCS-02: Schiffsparameter(minimal_turn_radius, maximum_turning_angle) in den configuration.properties einstellen TCS-03: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-04: Den berechneten Pfad der Path-Planning überwachen
Erwartetes Ergebnis:	- Das System weicht dem Dynamischen Objekt aus ohne dabei mit dem statischen Hinderniss zu kolidieren
Eingetretenes Ergebnis:	Radius, der betrachtet wird, ob das Schiff ausweicht zu klein (25m) (Goodwinshapes)
Erfolg/Misserfolg:	Misserfolg

Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-13
Test Titel:	Path-Planning berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes
Beschreibung:	Ausweichen eines dynamischen Objektes (Schiff) unter berücksichtigung der physikalischen Eigenschaften für das im Test eingesetzte Schiff (ZUSE).
Anforderung	F-CA-PP-6
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Schiffsparameter(minimal_turn_radius, maximum_turning_angle) in den configuration.properties einstellen TCS-02: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-03: Überwachen des berechneten Pfades auf Einhaltung der Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes
Erwartetes Ergebnis:	- Der berechnete Pfad steht nicht im Konflikt mit den physikalischen Eigenschaften des im Test eingesetzten Schiffs
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	Nicht getestet
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-14.1
Test Titel:	Path-Planning Überholen
Beschreibung:	Überholen eines dynamischen Objektes (Schiff) unter berücksichtigung der COLREG Regel 13.
Anforderung	F-CA-PP-8
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen (Überholen, COLREG 13) TCS-05b: Das Ausweichen des Schiffes überwachen (Entgegenkommen, COLREGs 14, 16) TCS-05c: Das Ausweichen des Schiffes überwachen (Kreuzen, COLREGs 15, 16) TCS-05d: Das Ausweichen des Schiffes überwachen (Zwangsweises Ausweichen, COLREGs 15, 17)
Erwartetes Ergebnis:	- Die COLREG Regel 13 wurde eingehalten
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	Nicht getestet
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-14.2
Test Titel:	Path-Planning Entgegenkommend
Beschreibung:	Ausweichen eines entgegenkommenden dynamischen Objektes (Schiff) unter berücksichtigung der COLREGs Regeln 14, 16, 17.
Anforderung	F-CA-PP-8
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen
Erwartetes Ergebnis:	- Die COLREGs Regeln 14, 16, 17 wurden eingehalten
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	Nicht getestet
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-14.3
Test Titel:	Path-Planning Kreuzen von rechts
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von rechts gekreuzt unter berücksichtigung der COLREGs Regeln 15, 16, 17.
Anforderung	F-CA-PP-8
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen (Kreuzen, COLREGs 15, 16)
Erwartetes Ergebnis:	- Die COLREG Regeln 15, 16, 17 wurden eingehalten
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	Nicht getestet
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-14.4.1
Test Titel:	Path-Planning Kreuzen von links (Kurs beibehalten)
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von links gekreuzt unter berücksichtigung der COLREGs Regeln 15, 16, 17.
Anforderung	F-CA-PP-8
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Überwachung des Kreuz-Manövers
Erwartetes Ergebnis:	- Die COLREGs Regeln 15,16,17 wurde eingehalten
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	Nicht getestet
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-14.4.2
Test Titel:	Path-Planning Kreuzen von links (Ausweichen)
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von links gekreuzt, das andere Schiff verletzt die COLREGs Regeln und weicht nicht aus, daraufhin muss das zu testende Schiff zwangsweise Ausweichen unter berücksichtigung der COLREGs Regeln 15, 16, 17.
Anforderung	F-CA-PP-8
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Überwachung des erzwungenen Ausweichens während des Kreuz-Manövers
Erwartetes Ergebnis:	- Die COLREGs Regeln 15,16,17 wurde eingehalten
Eingetretenes Ergebnis:	Radius, der betrachtet wird, ob das Schiff ausweicht zu klein (25m) (Goodwinshapes)
Erfolg/Misserfolg:	Misserfolg
Testmethode:	Szenariobasiertes Blackbox-Testing

ID:	TC-15
Test Titel:	Schiffserkennung mittels Kamera

Beschreibung:	Die Object Detection Komponente muss kontinuierlich die Umgebung in Fahrtrichtung mittels einer Kamera erfassen und Schiffe erkennen. Für erkannte Schiffe, die unter dem Horizont liegen, wird die Position (rechts oder links vom eigenen Schiff) an die DOI als JSON-Objekt gesendet.
Anforderung	F-OD-1 F-OD-2 F-OD-3 F-OD-4 F-OD-5
Eingehende-Daten:	Echtzeit-Videoaufnahmen einer Webcam von der Umgebung vor dem Schiff
Testschritte:	TCS-01: „Webcam Selection“ öffnen (durch starten der Komponente), die Webcam „HD Pro Webcam C920 0“ auswählen und auf „Select Camera“ klicken TCS-02: Überprüfen der gesendeten Schiffe an die DOI mit MATE Control.
Erwartetes Ergebnis:	● Schiffe, die sich in Fahrtrichtung befinden, werden korrekt erkannt. ● Die Position (RIGHT, LEFT, BOTH oder NOTHING) der mittels Bildverarbeitung erkannten Schiffe, die sich unter dem Horizont befinden, werden als JSON-Objekt an die DOI gesendet.
Eingetretenes Ergebnis:	Schiffe werden erkannt erkannt, außer sie befinden sich zu weit weg, dann werden auf dem Kamerabild zu klein angezeigt und können nicht erkannt werden. Wenn Schiffe erkannt werden, wird die Position als JSON-Objekt an die DOI gesendet.
Erfolg/Misserfolg:	Erfolg (sollte aber weiter verbessert werden)
Testmethode:	Szenariobasiertes Blackbox-Testing

Fehlerinjektion:

Geplant für den Systemtest:

ID:	TC-FI-2
Test Titel:	Verbindungsabbruch Navibox
Beschreibung:	Die Verbindung zwischen dem zu testenden System und der Navibox wird einmalig für ein längeres Intervall unterbrochen.

Eingehende-Daten:	- Schiffssensorinformationen
	  - Schiffsposition
	  - Schiffsrichtung
	  - AIS
	  - (Radar)

Testschritte:	TCS-01: Verbindung zwischen dem zu testenden System und Labskaus abbrechen TCS-02: Überwachung des zu testenden Systems mit MATE-Control TCS-03: Verbindung zwischen dem zu testenden System und Labskaus nach 2 Minuten wieder aufbauen
Erwartetes Ergebnis:	- Das Schiff drosselt die Geschwindigkeit auf null
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-3
Test Titel:	Gemessener Ruderwinkel außerhalb der Grenzen
Beschreibung:	Der Controller empfängt den Ruderwinkel außerhalb der spezifizierten Grenzen
Eingehende-Daten:	- Ruderwinkel per NMEA 2000 (an Controller)
Testschritte:	TCS-01: Abfangen der Ruderwinkel-Nachrichten mit MATE-Control TCS-02: Ändern und senden der Ruderwinkel-Nachricht zu einem Wert außerhalb der spezifizierten Grenzen (180° und -180°) TCS-03: Überwachung des Systems mit MATE-Control
Erwartetes Ergebnis:	- Ist uns unbekannt
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-4
Test Titel:	Gemessene negative Drehzahl
Beschreibung:	Der Controller erhält negative Drehzahlen.

Eingehende-Daten:	- Drehzahl per NMEA 2000
Testschritte:	TCS-01: Abfangen der Drehzahl-Nachrichten TCS-02: Ändern und senden der Drehzahl-Nachrichten (negative Werte) TCS-03: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Schiff beschleunigt
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-7
Test Titel:	Nahes Schiff verschwindet
Beschreibung:	Ein mit AIS und/oder Radar identifiziertes Schiff in unmittelbarer Nähe zum eigenen Schiff verschwindet aus dem Grid der Path-Planning-Komponente
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten)
Testschritte:	TCS-01: In MATE-Control den Error-Injector öffnen und das nahe Schiff löschen TCS-02: Überwachung des Systems mittels MATE-Control

Erwartetes Ergebnis:	- Schiff fährt weiter
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-10
Test Titel:	Bekanntes Schiff springt auf eigene Position
Beschreibung:	Ein mit AIS und oder Radar identifiziertes Schiff in der Umgebung der eigenen Schiffes springt auf die Position des eigenen Schiffes
Eingehende-Daten:	
Testschritte:	TCS-01: Mittels MATE-Control Error-Injection die Schiffsposition eines anderen Schiffes auf die eigene Position setzen TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Das Schiff drosselt die Geschwindigkeit auf null
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-12
Test Titel:	Schiff wechselt AIS-MMSI
Beschreibung:	Ein mit AIS identifiziertes Schiff ändert dauernd seine AIS-MMSI
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten)
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die MMSI des Schiffes laufend ändern TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Schiffe werden nicht gefused
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

Weitere Fehlerinjektionen:

ID:	TC-FI-1
Test Titel:	Verbindungsabbrüche Navibox
Beschreibung:	Die Verbindung zwischen dem zu testenden System und der Navibox wird in kurzen Abständen abgebrochen und wieder aufgebaut.

Eingehende-Daten:	- Was kommt alles aus der Navibox in unser System?
Testschritte:	TCS-01: Verbindung zwischen dem zu testenden System und Labskaus abbrechen TCS-02: Verbindung zwischen dem zu testenden System und Labskaus wieder aufbauen TCS-03: Zehnfache Wiederholung der TCS-01 und TCS-02 in schneller Abfolge
Erwartetes Ergebnis:	- Schiff drosselt Geschwindigkeit auf null
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-5
Test Titel:	Unvollständige AIS-Daten
Beschreibung:	Einspeisung unvollständiger AIS-Nachrichten die an die DOI-Komponente gesendet werden.
Eingehende-Daten:	- Sensordaten des AIS in NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm
Testschritte:	TCS-01: Einspeisung unvollständiger AIS-Nachrichten mittels MATE-Control TCS-02: TCS-02: Überwachung des Systems mittels MATE-Control

Erwartetes Ergebnis:	-
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-6
Test Titel:	Unvollständige Radar-Daten
Beschreibung:	Einspeisung unvollständiger Radar-Nachrichten die an die DOI-Komponente gesendet werden.
Eingehende-Daten:	- Sensordaten des Radars im NMEA-2000 konformen Format in einem Umkreis ums eigenen Schiff von bis zu 2sm
Testschritte:	TCS-01: Einspeisung unvollständiger Radar-Nachrichten mittels MATE-Control TCS-02: TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	-
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-8
Test Titel:	Entferntes Schiff springt in unmittelbare Nähe
Beschreibung:	Ein mit AIS und oder Radar identifiziertes Schiff in einiger Entfernung zum eigenen Schiff springt plötzlich in unmittelbare Nähe vor das eigene Schiff
Eingehende-Daten:	- Sensordaten des AIS in NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten)
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die Position eines entfernten Schiffes vor die eigene Position setzen TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Schiff leitet Ausweichen ein
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-9
Test Titel:	Schiff im Umkreis wechselt ständig die Position
Beschreibung:	Ein mit AIS und oder Radar identifiziertes Schiff in der Umgebung zum eigenen Schiff springt zwischen mehreren Positionen um das Schiff umher

Eingehende-Daten:	- Sensordaten des AIS in NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Nachrichten)
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die Position eines entfernten Schiffes dauern ändern TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Schiff fährt normal seine Route weiter
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-11
Test Titel:	Unbekanntes Schiff springt auf eigene Position
Beschreibung:	Ein nicht vorher identifiziertes Schiff springt auf die Position des eigenen Schiffes
Eingehende-Daten:	
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die Position eines unbekannten Schiffes vor die eigene Position setzen TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Das Schiff drosselt die Geschwindigkeit auf null
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	

Testmethode:	

ID:	TC-FI-13
Test Titel:	Statisches Hindernis taucht auf eigener Position auf
Beschreibung:	Während der Fahrt taucht ein statisches Hindernis auf der Position des eigenen Schiffes auf.
Eingehende-Daten:	- Polygone vom NoGoSolver per JSON
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die Position eines statischen Hindernisses auf die eigene Position setzen TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Schiff drosselt Geschwindigkeit auf Null
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

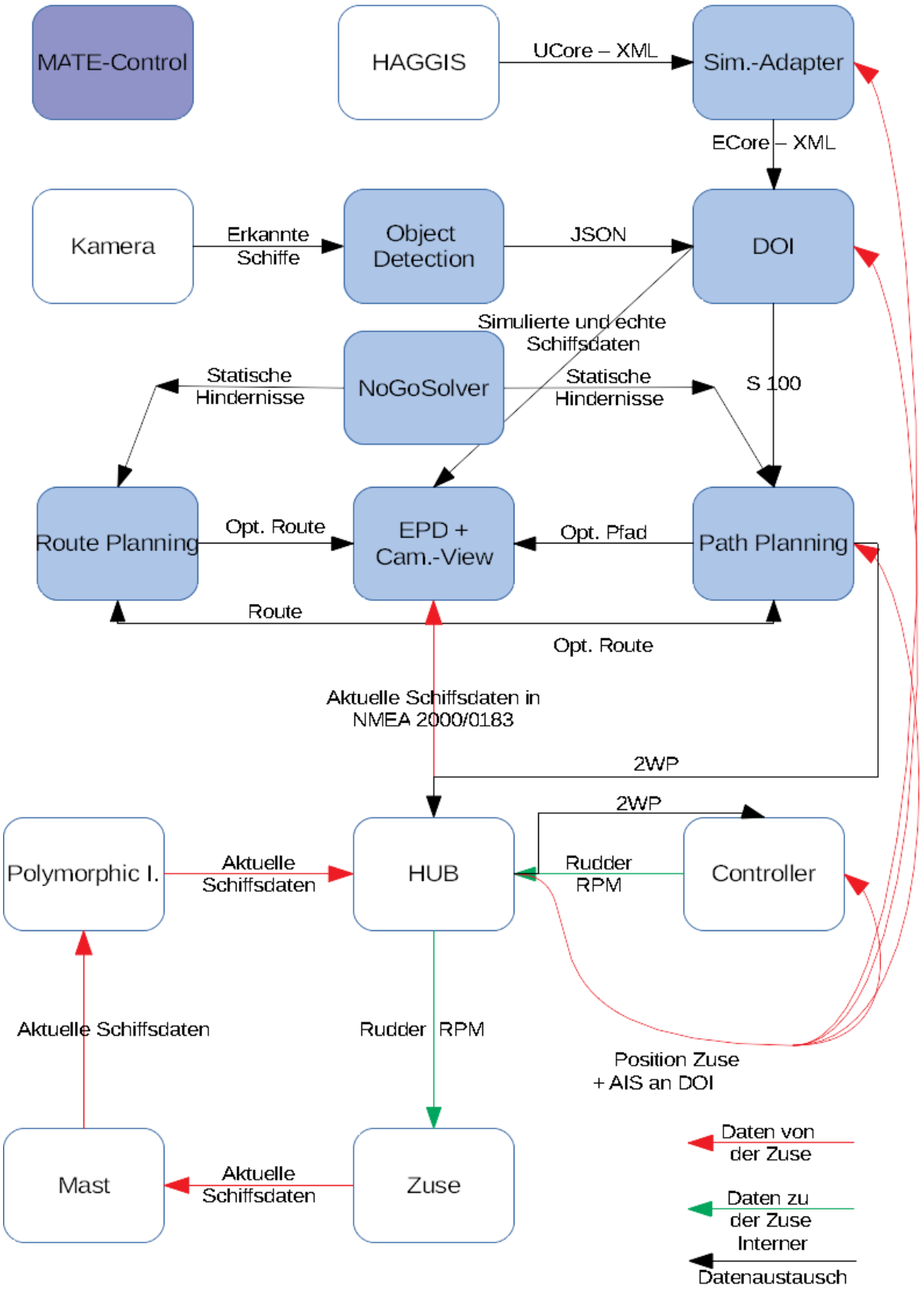
ID:	TC-FI-14
Test Titel:	Statisches Hindernis taucht vor eigener Position auf
Beschreibung:	Während der Fahrt taucht ein statisches Hindernis unmittelbar vor der Position des eigenen Schiffes auf.

Eingehende-Daten:	- Polygone vom NoGoSolver per JSON
Testschritte:	TCS-01: Mittels MATE-Control Error Injection die Position eines statischen Hindernis vor die eigene Position setzen TCS-02: Überwachung des Systems mittels MATE-Control
Erwartetes Ergebnis:	- Schiff weicht dem Hindernis aus
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	
Testmethode:	

ID:	TC-FI-15
Test Titel:	Ausfall RabbitMQ
Beschreibung:	RabbitMQ fällt während der Fahrt aus.
Eingehende-Daten:	
Testschritte:	TCS-01: Verbindungen des zu testenden Systems zu RabbitMQ unterbrechen
Erwartetes Ergebnis:	- System arbeitet nicht mehr Ordnungsgemäß
Eingetretenes Ergebnis:	
Erfolg/Misserfolg:	

Testmethode:	

Gesamtübersicht:


MATE-Control
HAGGIS
UCore – XML
Sim.-Adapter
ECore – XML
Kamera
Erkannte Schiffe
Object Detection
JSON
DOI
Simulierte und echte Schiffsdaten
Statische Hindernisse
NoGoSolver
Statische Hindernisse
S 100
Route Planning
Opt. Route
EPD + Cam.-View
Opt. Pfad
Path Planning
Route
Opt. Route
Aktuelle Schiffsdaten in NMEA 2000/0183
2WP
2WP
Polymorphic I.
Aktuelle Schiffsdaten
HUB
Rudder RPM
Controller
Aktuelle Schiffsdaten
Rudder RPM
Position Zuse + AIS an DOI
Mast
Aktuelle Schiffsdaten
Zuse
Daten von der Zuse
Daten zu der Zuse
Interner Datenaustausch

Systemtest vom 15.08.2018

Labskaus-Integrationstest:

Methode.
Zunächst erfolgt ein Systemintegrationstest des PG-MATE Systems und LABSKAUS. Der Systemintegrationstest verifiziert die korrekte Funktionsweise der polymorphen Schnittstelle hinsichtlich einer erfolgten Kommunikationsverbindung, sowie der syntaktischen Korrektheit der Daten. Anschließend beurteilt ein Systemexperte des PG-MATE Systems die semantische Korrektheit einiger empfangener Live-Daten der mobilen Sensorbox, die dem PG-MATE Systems durch LABSKAUS bereitgestellt und ein plausibles Ergebnis anhand der vorliegenden Situation der Realwelt validiert wird.

Testdurchführung.

Testspezifikation:

Zur Vorbereitung des Systemtests werden die Komponenten des Systems, EPD, Route Planning, Path Planning, DOI, Object Detection und der Controller gestartet. Zusätzlich werden Haggis, der Simulations Adapter und der NoGoSolver gestartet und wie die Testumgebung mit dem System verbunden. Mit den geplanten Testszenarien werden die meisten funktionalen Anforderungen an das System verifiziert und zusätzlich werden die COLREG Regeln 13-17 abgedeckt. Die in den Szenarien benötigten Schiffe werden mit Hilfe von Haggis simulativ über den Simulations Adapter in die Testumgebung eingespeist.

Testkonfiguration:

Für die Szenarien des Systemtests erhält das zu testende Schiff eine Route, in der jedes Szenario durchgespielt wird. Diese Route wurde im Vorfeld des Systemtests bereits zusammengestellt und in der EPD abgespeichert. Im Folgenden werden die Abschnitte für jedes Szenario aufgelistet.

Testszenario	Abschnitt

TS-0	- WP-1:
	  - Lat: 53.5169404273528
	  - Long: 8.171167373657227
	- WP-2:
	  - Lat: 53.52477336808653
	  - Long: 8.181166648864746

TS-1	- WP-1:
	  - Lat: 52.697
	  - Long: 8.16988
	- WP-2:
	  - Lat: 51.577
	  - Long: 8.16988

TS-2	- WP-1:
	  - Lat: 53.5630989074707
	  - Long: 8.172971725463867
	- WP-2:
	  - Lat: 53.56486129760742
	  - Long: 8.172354698181152
	- WP-3:
	  - Lat: 53.566619873046875
	  - Long: 8.171737670898438
	- WP-4:
	  - Lat: 53.56837844848633
	  - Long: 8.171119689941406
	- WP-5:
	  - Lat: 53.57013702392578
	  - Long: 8.170502662658691
	- WP-6:
	  - Lat: 53.571895599365234
	  - Long: 8.169885635375977
	- WP-7:
	  - Lat: 53.57365417480469
	  - Long: 8.169268608093262

TS-3	- WP-1:
	  - Lat: 53.53246307373047
	  - Long: 8.1775484085083
	- WP-2:
	  - Lat: 53.53422164916992
	  - Long: 8.17693233489
	- WP-3:
	  - Lat: 53.536346435546875
	  - Long: 8.179267883300781
	- WP-4:
	  - Lat: 53.53986740112305
	  - Long: 8.178035736083984
	- WP-5:
	  - Lat: 53.5416259765625
	  - Long: 8.17741870880127
	- WP-6:
	  - Lat: 53.54375076293945
	  - Long: 8.179755210876465
	- WP-7:
	  - Lat: 53.545509338378906
	  - Long: 8.179139137268066

TS-4	- WP-1:
	  - Lat: 53.519412994384766
	  - Long: 8.175958633422852
	- WP-2:
	  - Lat: 53.52117156982422
	  - Long: 8.175342559814453
	- WP-3:
	  - Lat: 53.52293395996094
	  - Long: 8.174725532531738
	- WP-4:
	  - Lat: 53.52469253540039
	  - Long: 8.17410945892334
	- WP-5:
	  - Lat: 53.526817321777344
	  - Long: 8.176445007324219

	- WP-6:
	  - Lat: 53.5285758972168
	  - Long: 8.17582893371582
	- WP-7:
	  - Lat: 53.53033447265625
	  - Long: 8.175211906433105

TS-5	- WP-1:
	  - Lat: 53.545509338378906
	  - Long: 8.179139137268066
	- WP-2:
	  - Lat: 53.54726791381836
	  - Long: 8.178523063659668
	- WP-3:
	  - Lat: 53.54902648925781
	  - Long: 8.177906036376953
	- WP-4:
	  - Lat: 53.55078887939453
	  - Long: 8.177289009094238
	- WP-5:
	  - Lat: 53.552547454833984
	  - Long: 8.1766729354858
	- WP-6:
	  - Lat: 53.55430603027344
	  - Long: 8.176055908203125
	- WP-7:
	  - Lat: 53.55606460571289
	  - Long: 8.17543888092041

Des Weiteren müssen für die Szenarien 2-5 die Routen der simulierten Schiffe konfiguriert werden, diese wurden in Abhängigkeit zur Route des zu testenden Schiffes entwickelt.

Testszenario	Route

TS-2	- WP-1:
	  - Lat: 53.568378448
	  - Long: 8.171119689941
	- WP-2:
	  - Lat: 53.57365417
	  - Long: 8.1692686080

TS-3	- WP-1:
	  - Lat: 53.53416
	  - Long: 8.18306
	- WP-2:
	  - Lat: 53.53449
	  - Long: 8.17168

TS-4	- WP-1:
	  - Lat: 53.52179
	  - Long: 8.17245
	- WP-2:
	  - Lat: 53.52345
	  - Long: 8.17934
	- WP-3:
	  - Lat: 53.52551
	  - Long: 8.18808

TS-5	- WP-1:
	  - Lat: 53.552547
	  - Long: 8.1766729
	- WP-2:
	  - Lat: 53.54375076
	  - Long: 8.17975521

Testszenarien:

ID: Basis-0	Controllergesteuerte Fahrt von einem Wegpunkt zum anderen.	Bemerkungen: - Fehler in der RPM-Regelung: Der Controller hebt die Geschwindigkeit zu langsam an, ist die Zielgeschwindigkeit erreicht, fällt sie zu stark ab. - die RPM-Regelung des Controllers funktioniert nicht, sodass eine Konstante (ca. 1300 RPM) fest konfiguriert wird. Alle in diesem Test angefahrenen Wegpunkte werden mit der konstanten Geschwindigkeit angefahren. - später wurde die Konstante auf 1700 RPM erhöht. - Wegpunkte werden abgefahren.
Beschreibung:	Start im Jadebusen in Wilhelmshaven im Umkreis von 100m zum Startpunkt des ausgewählten Wegpunktpaars. Die Fahrt vom ersten zum zweiten Wegpunkt wird dabei kontinuierlich durch den Steuermann über bereitgestellte Monitore mit Schiffs- und Umweltinformationen überwacht.	
Vorbedingungen:	□ Kommunikation zwischen den Komponenten muss gegeben sein □ Eigene Position im Umkreis von max. 100m zum Startpunkt der Route	Beim ersten Versuch ist der Mate-Control-Adapter abgestürzt, sodass die Kommunikation zwischen den Komponenten unterbrochen wurde. Anschließend wurden alle Vorbedingungen erfüllt.
Testfälle:	TC-5.2-Controller Wegpunkt abfahren	

ID: TS-1	Route automatisiert abfahren	Bemerkungen: - die Regelung der Geschwindigkeit war nicht funktionsfähig - sämtliche Tests werden mit einer konstanten, fest konfigurierten Geschwindigkeit abgefahren
Beschreibung:	Start im Jadebusen in Wilhelmshaven im Umkreis von maximal 600m zum Startpunkt der berechneten globalen Route. Über den Küstenleitstand wird eine Route zum Jade-Weser Port generiert. Diese soll dann vom Schiff automatisiert abgefahren werden, während der eigentliche Steuermann die Fahrt über bereitgestellte Monitore mit Schiffs- und Umweltinformationen überwacht.	
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Eigene Position im Umkreis von max. 600m zum Startpunkt der Route	- Konfiguration des RP und des PP mussten angepasst werden. - anschließend wurden alle Vorbedingungen erfüllt.

Testfälle:	TC-3-Ungültige Route optimieren TC-1.1-Route mit Schiffsparametern erstellen TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning TC-2.1-Route-Planning Route empfangen TC-2.2-Route-Planning Route optimieren TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD) TC-11.2-Path Planning Kommunikation mit dem NoGoSolver TC-11.3-Path Planning Kommunikation mit der DOI TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller)	- alle erfolgreich, außer: ▪ TC-4-Controller Geschwindigkeit

	TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD) TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	

ID: TS-2	Überholmanöver Schiff (COLREG 13)	Bemerkungen:
Beschreibung:	Das Schiff soll im Autopiloten das Schiff erkennen, überholen und auf die ursprünglich geplante Route zurückkehren. <u>COLREG 13:</u> a) Das Schiff, das überholt wird, muss von anderen Schiffen gemieden werden b) Ein Schiff gilt als überholt, sobald das überholte Schiff deren Seitenlichter sieht c) Falls es irgendwelche bedenken gibt, darf nicht überholt werden	
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Startpunkt für das Szenario in Haggis muss erreicht werden ☒ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (7 Knoten) ☒ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	- alle Vorbedingungen wurden erfüllt.

Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning TC-2.1-Route-Planning Route empfangen TC-2.2-Route-Planning Route optimieren TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-15.1- Schiffserkennung mittels Kamera TC-15.2- Senden der erkannten Schiffe der Object Detection an PP und EPD TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-9.1-Objekterkennung (hinter dem eigenen Schiff) TC-9.2-Objekterkennung (links neben dem eigenen Schiff) TC-9.3-Objekterkennung (rechts neben dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m)	- alle erfolgreich, außer: ▪ TC-4-Controller Geschwindigkeit (siehe Basis-0) ▪ TC15-Schiffserkennung mittels Kamera: o Schiffe wurden mit der Kamera erkannt, die Signalgebung an das Path-Planning konnte nicht getestet werden. ▪ Eigener Testfall mit Schiff ohne AIS/Radar ▪ Reaktion des PP ist noch zu testen ▪ TC-14.1-Path-Planning Überholen: o Pfad verändert sich zu oft o weitläufigeres Ausweichen notwendig o Rückführung auf globalen Pfad: Nach erfolgreichem Überholvorgang wird ein falscher Wegpunkt ausgewählt, sodass die Zuse umdrehen muss.

	TC-10.2-Datenfusion (Größe des Schiffs 20m) TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD) TC-11.2-Path Planning Kommunikation mit dem NoGoSolver TC-11.3-Path Planning Kommunikation mit der DOI TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller) TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD) TC-12-Path-Planning Berücksichtigung statischer Hindernisse TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.1-Path-Planning Überholen	

ID: TS-3	Kreuzendes Schiff von rechts (wir müssen ausweichen) (COLREG 15, 16, 17)	Bemerkungen:
Beschreibung:	Das Schiff soll das Fremdschiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. <u>COLREG 15:</u> Falls zwei Schiffe sich kreuzen muss das Schiff, welches sein Gegenüber auf der Steuerbord-Seite hat reagieren und ausweichen	
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Startpunkt für das Szenario in Haggis muss erreicht werden ☒ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (7 Knoten) ☒ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	- alle Vorbedingungen wurden erfüllt.

Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning TC-2.1-Route-Planning Route empfangen TC-2.2-Route-Planning Route optimieren TC-2.2- EPD-seitiger Empfang einer vom Path-Planning optimierten Route TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-15.1- Schiffserkennung mittels Kamera TC-15.2- Senden der erkannten Schiffe der Object Detection an PP und EPD TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-9.1-Objekterkennung (hinter dem eigenen Schiff) TC-9.2-Objekterkennung (links neben dem eigenen Schiff)	- alle erfolgreich, außer: ▪ TC-4-Controller Geschwindigkeit (siehe Basis-0) ▪ TC-15-Schiffserkennung mittels Kamera: o Schiffe wurden mit der Kamera erkannt, jedoch kamen im Szenario nur virtuelle Schiffe vor, sodass die Auswirkung auf das konkrete Fahrmanöver nicht getestet werden konnte. o Die Reling der Zuse war die meiste Zeit auf Höhe der zu erkennenden Schiffe ▪ TC-14.3-Path-Planning Kreuzen von rechts o Pfad verändert sich zu oft o weitläufigeres Ausweichen notwendig o Rückführung auf globalen Pfad: Nach erfolgreichem Überholvorgang wird ein falscher Wegpunkt ausgewählt, sodass die Zuse umdrehen muss. ▪ TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes o fehlende Validierungsindikatoren

	TC-9.3-Objekterkennung (rechts neben dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m) TC-10.2-Datenfusion (Größe des Schiffs 20m) TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD) TC-11.2-Path Planning Kommunikation mit dem NoGoSolver TC-11.3-Path Planning Kommunikation mit der DOI TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller) TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD) TC-12-Path-Planning Berücksichtigung dynamischer und statischer Hindernisse TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.3-Path-Planning Kreuzen von rechts	

ID: TS-4	Kreuzendes Schiff von links (COLREG 15, 16, 17)	Bemerkungen:
Beschreibung:	Das Schiff soll das simulierte Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. <u>COLREG 17:</u> a) Ein Schiff verhält sich nicht COLREG-Konform b) Falls das gegenüberliegende Schiff, das Ausweichen muss, nicht reagiert, muss das eigene Schiff trotzdem entsprechend reagieren und ausweichen d) Diese Regel befreit Schiffe nicht von ihrer Pflicht, sich von solchen Situationen fern zu halten	
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Startpunkt für das Szenario in Haggis muss erreicht werden ☒ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (7 Knoten) ☒ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	- Konfiguration des RP und des PP mussten angepasst werden. - anschließend wurden alle Vorbedingungen erfüllt.

Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning TC-2.1-Route-Planning Route empfangen TC-2.2-Route-Planning Route optimieren TC-2.2- EPD-seitiger Empfang einer vom Path-Planning optimierten Route TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-15.1- Schiffserkennung mittels Kamera TC-15.2- Senden der erkannten Schiffe der Object Detection an PP und EPD TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-9.1-Objekterkennung (hinter dem eigenen Schiff) TC-9.2-Objekterkennung (links neben dem eigenen Schiff)	- alle erfolgreich, außer: ▪ TC-2.2-Route-Planning Route optimieren: o Das RP musste einmal neugestartet warden, da die Kommunikation zwischen EPD und RP nicht funktioniert hat. ▪ TC-4-Controller Geschwindigkeit (siehe Basis-0) ▪ TC-12-Path-Planning Berücksichtigung dynamischer und statischer Hindernisse: o die „NoGo-Zones“, die nicht befahren werden dürfen und vom PP in der Pfadberechnung berücksichtigt werden, müssen größer o berechneter Pfad springt zu stark ▪ TC-14.4.2-Path-Planning Kreuzen von links (Ausweichen): o Ausweichrichtung kreuzt den Kurs des anderen Schiffes o Pfad verändert sich zu oft o weitläufigeres Ausweichen notwendig o Rückführung auf globalen Pfad: Nach erfolgreichem Überholvorgang wird ein falscher Wegpunkt ausgewählt, sodass die Zuse umdrehen muss. ▪ TC-14.4.1-Path-Planning Kreuzen von links (Kurs beibehalten) o Zum Testen fehlt ein Szenario in Haggis ▪ TC-15-Schiffserkennung mittels Kamera: o Schiffe wurden mit der Kamera erkannt, jedoch kamen im Szenario nur virtuelle Schiffe vor, sodass die Auswirkung auf das konkrete Fahrmanöver nicht getestet werden konnte.

	TC-9.3-Objekterkennung (rechts neben dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m) TC-10.2-Datenfusion (Größe des Schiffs 20m) TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD) TC-11.2-Path Planning Kommunikation mit dem NoGoSolver TC-11.3-Path Planning Kommunikation mit der DOI TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller) TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD) TC-12-Path-Planning Berücksichtigung statischer Hindernisse TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.4.1-Path-Planning Kreuzen von links (Kurs beibehalten) TC-14.4.2-Path-Planning Kreuzen von links (Ausweichen)	○ Die Reling der Zuse war die meiste Zeit auf Höhe der zu erkennenden Schiffe ▪ TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes ○ fehlende Validierungsindikatoren

ID: TS-5	Entgegenkommendes Schiff (COLREG 14, 16, 17)	Bemerkungen: - 1. Testabbruch: ▪ Path-Planning sendet zu wenig Wegpunkte an den Controller, sodass eine Steuerung nicht möglich ist (Zuse weicht zu stark vom abzufahrenden Pfad ab) - 2. Testabbruch: ▪ Die DOI fusioniert das Radar-Objekt nicht mit dem AIS-Target, sodass zwei Schiffe, bzw. GoodWin-Shapes in die Pfad-Berechnung des Path-Plannings eingehen.
Beschreibung:	Das Schiff soll das entgegenkommende Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. Danach übernimmt der Kapitän kurzzeitig die Kontrolle mithilfe der Virtual Handles. <u>COLREG 14:</u> a) Beide Schiffe weichen auf der Steuerbordseite aus b) Diese Situation gilt falls vermutet wird, dass ein gegenüberliegendes Schiff auf das Eigene zufährt oder sich annähert c) Falls die Vermutung besteht, dass diese Situation gilt, muss ausgewichen werden	
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Startpunkt für das Szenario in Haggis muss erreicht werden ☒ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (7 Knoten) ☒ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	- alle Vorbedingungen wurden erfüllt.

Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning TC-2.1-Route-Planning Route empfangen TC-2.2-Route-Planning Route optimieren TC-2.2- EPD-seitiger Empfang einer vom Path-Planning optimierten Route TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-15.1- Schiffserkennung mittels Kamera TC-15.2- Senden der erkannten Schiffe der Object Detection an PP und EPD TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-9.1-Objekterkennung (hinter dem eigenen Schiff) TC-9.2-Objekterkennung (links neben dem eigenen Schiff)	- alle erfolgreich, außer: ▪ TC-4-Controller Geschwindigkeit (siehe Basis-0) ▪ TC-15-Schiffserkennung mittels Kamera: o Schiffe wurden mit der Kamera erkannt, jedoch kamen im Szenario nur virtuelle Schiffe vor, sodass die Auswirkung auf das konkrete Fahrmanöver nicht getestet werden konnte. o Die Reling der Zuse war die meiste Zeit auf Höhe der zu erkennenden Schiffe ▪ TC-14.2-Path-Planning Entgegenkommend: o Pfad verändert sich zu oft o weitläufigeres Ausweichen notwendig o Rückführung auf globalen Pfad: Nach erfolgreichem Ausweichvorgang wird ein falscher Wegpunkt ausgewählt, sodass die Zuse umdrehen muss. ▪ TC-10.2-Datenfusion (Größe des Schiffs 20m) o teilweise keine Fusion der AIS-Targets und Radar-Objekte o Auswirkungen aufs Path-Planning, da mehrere GoodWin-Shapes berechnet werden und in die Fahrtplanung eingehen.

	TC-9.3-Objekterkennung (rechts neben dem eigenen Schiff) TC-10.1-Datenfusion (Größe des Schiffs 5m) TC-10.2-Datenfusion (Größe des Schiffs 20m) TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD) TC-11.2-Path Planning Kommunikation mit dem NoGoSolver TC-11.3-Path Planning Kommunikation mit der DOI TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller) TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD) TC-12-Path-Planning Berücksichtigung dynamischer und statischer Hindernisse TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.2-Path-Planning Entgegenkommend	

Testfälle:

ID:	TC-1.1	Bemerkungen:
Test Titel:	Route mit Schiffsparametern erstellen	
Beschreibung:	Zusammenstellung einer Route in der EPD mit Schiffsparametern.	
Funktionale Anforderungen:	F-EPD-2 F-EPD-5	
Eingehende-Daten:	-	
Testschritte:	TCS-01: Wegpunktwerkzeug in der EPD auswählen und eine Route, bestehend aus zwei Wegpunkten, auf der Karte erstellen Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 Optional: TCS-02a: Route im Routenmanager der EPD per Doppelklick öffnen und ggfs. Geschwindigkeiten (max) der einzelnen Wegpunkte setzen (auf 7 Knoten)	

	TCS-02: Schiffsparameter in der “PG Mate”-Einstellungsseite der EPD hinterlegen und speichern Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Erwartetes Ergebnis:	Verbindet die gesetzten Wegpunkte zu einer Route, bestehend auf WP 1 und 2 mit einer maximalen WP-Leg-Geschwindigkeit von 7 Knoten und speichert die folgenden Schiffsparameter. Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 Schiffsparameter: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-1.2	Bemerkungen:
Test Titel:	Senden einer Route und Schiffsparameter an das Route-Planning	
Beschreibung:	Die zusammengestellte RTZ-Route, bestehend aus: Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 7 Knoten) wird mit den hinterlegten Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5 der Zuse per HTTP-Request an das Route-Planning gesendet.	
Funktionale Anforderungen:	F-EPD-2 F-EPD-5	
Eingehende-Daten:		

Testschritte:	TCS-01: Route im Route-Manager der EPD auswählen und über die Schaltfläche „Route optimieren“ an das Route-Planning senden.	
Erwartetes Ergebnis:	Die ausgewählte RTZ-Route und die hinterlegten Schiffsparameter wird mit den hinterlegten Geschwindigkeiten der einzelnen Wegpunkte an das Route-Planning gesendet: Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 7 Knoten) wird mit den hinterlegten Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.1	Bemerkungen:
Test Titel:	Route-Planning Route empfangen	
Beschreibung:	Eine Route, bestehend aus den beiden Wegpunkten: Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 7 Knoten) soll zusammen mit den hinterlegten Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5 vom Route-Planning empfangen werden.	
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5	

Eingehende-Daten:	in Route-Planning eingehend: - RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit (7 Knoten) zwischen den Wegpunkten: o Wegpunkt 1: o Lat: 53.51765489239272 o Lon: 8.17657470703125 o Wegpunkt 2: o Lat: 53.667163323271666 o Lon: 8.124010935557715 - Schiffsinformationen o Länge: 8.0 o Breite: 2.5 o Höhe 2.5 o Tiefgang: 0.5	
Testschritte:	TCS-01: nicht-optimierte Route von der EPD an das Route Planning senden: o Wegpunkt 1: o Lat: 53.51765489239272 o Lon: 8.17657470703125 o Wegpunkt 2: o Lat: 53.667163323271666 o Lon: 8.124010935557715	

Erwartetes Ergebnis:	Das Route-Planning erhält eine Route von der EPD und die Schiffsinformationen Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 7 Knoten) Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.2	Bemerkungen:
Test Titel:	Route-Planning Route optimieren	
Beschreibung:	Optimieren einer ausgewählten Route	
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5	
Eingehende-Daten:	in Route-Planning eingehend: - RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit (7 Knoten) zwischen den Wegpunkten o Wegpunkt 1: o Lat: 53.51765489239272 o Lon: 8.17657470703125 o Wegpunkt 2: o Lat: 53.667163323271666 o Lon: 8.124010935557715	
Testschritte:	TCS-01: eingehende Route optimieren	
Erwartetes Ergebnis:	Das Route-Planning optimiert eine ausgewählte Route, diese wird via HTTP zurück an die EPD übertragen. Diese berücksichtigt in ihrer Gestalt statische Hindernisse, geliefert vom NoGoSolver.	

	Enthaltene Wegpunkte: Siehe: Anlage 1: optimierte RTZ-Route	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.3	Bemerkungen:
Test Titel:	EPD-seitiger Empfang einer vom Route-Planning optimierten Route	
Beschreibung:	Empfang einer optimierten Route des Route-Planning auf Basis einer ausgewählten Route	
Funktionale Anforderungen:	F-EPD-6 F-EPD-7	
Eingehende-Daten:	1. in EPD eingehend: - RTZ-Route vom Route-Planning eingehend via HTTP in die EPD, bestehend aus berechneten Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit von 7 Knoten zwischen den Wegpunkten: Siehe:	

	Anlage 1: optimierte RTZ-Route	
Testschritte:	TCS-01: Nach Erhalt der optimierten Route (Siehe: Anlage 1: optimierte RTZ-Route), überprüfen, ob diese mit einem „optimized“ im Routenname im Routenmanager der EPD abgespeichert wird und sichtbar in der Karte dargestellt wird.	
Erwartetes Ergebnis:	Die EPD erhält eine vom Route-Planning optimierte Route via HTTP, die die gleichen Start- und Zielpunkte wie die Originalroute hat und Hindernisse, visualisiert durch “NoGo-Areas”, meidet. Die Geschwindigkeiten der Wegpunktverbindungen sind auf 7 Knoten gesetzt. Enthaltene Wegpunkte sind hier aufgelistet: Anlage 1: optimierte RTZ-Route	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.4	Bemerkungen:
Test Titel:	Senden einer optimierten Route von der EPD zum Path-Planning	
Beschreibung:	Eine in der EPD gespeicherte optimierte Route wird zum Start der Fahrt an das Path-Planning gesendet. Die optimierte Route (Siehe: Anlage 1: optimierte RTZ-Route) enthält dabei Abfahrgeschwindigkeiten von 7 Knoten.	
Funktionale Anforderungen:	F-EPD-6 F-EPD-7	
Eingehende-Daten:	in Path-Planning eingehend - RTZ-Route von der EPD via HTTP, beinhaltet alle zuvor vom Route-Planning berechneten Wegpunkte inklusive der Geschwindigkeiten zwischen den Wegpunkten Siehe: Anlage 1: optimierte RTZ-Route - Konfigurierte Schiffsinformationen aus der EPD als JSON-Objekte via HTTP	

Testschritte:	TCS-01: Schiffsinformationen in der Einstellungsmaske für Schiffsinformationen in der EPD setzen. TCS-02: Sicher gehen, dass Route ausgewählt (Anlage 1: optimierte RTZ-Route) ist und in der EPD „Route senden“ klicken. TCS-03: Prüfen, ob Route im Path-Planning eingeht.	
Erwartetes Ergebnis:	Das Path-Planning erhält eine vom Route-Planning optimierte Route (Anlage 1: optimierte RTZ-Route) als RTZ-Route via HTTP, die die gleichen Start- und Zielpunkte wie die Originalroute hat und statische Hindernisse(“NoGo-Areas”)meidet.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-3	Bemerkungen:
Test Titel:	Ungültige Route optimieren	
Beschreibung:	Zusammenstellung einer Route die einen Wegpunkt auf Land bzw. einer NoGoArea besitzt. Anschließend soll der Versuch gestartet werden, diese zu optimieren.	
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5	
Eingehende-Daten:	1. in Route-Planning eingehend: - RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten o WP1: o Lat: 53.56396732809957 o Lon: 8.124732971191406 o WP2: o Lat: 53.5792576599901 o Lon: 8.178634643554688 2. in EPD eingehend (im Anschluss an 1): - Fehlermeldung als JSON-Objekt via HTTP, welches in	

	der EPD das Darstellen einer Fehlermeldung auslöst	
Testschritte:	TCS-01: Wegpunktwerkzeug auswählen und auf der Karte eine Route zusammenstellen, wobei ein Wegpunkt auf Land zu setzen ist WP1: Lat: 53.56396732809957 Lon: 8.124732971191406 WP2: Lat: 53.5792576599901 Lon: 8.178634643554688 TCS-02: Sichergehen, dass Route ausgewählt ist und „Route optimieren“ auswählen TCS-03: Fehlermeldung mit “OK” bestätigen	
Erwartetes Ergebnis:	Rückmeldung als Hinweisfenster der EPD: “Es wurde keine Route gefunden!”	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-4	Bemerkungen:
Test Titel:	Controller Geschwindigkeit	
Beschreibung:	Überwachung der vorgegebenen Geschwindigkeit des Controllers: 3.60 m/s. Wird im Controller als Konstante angegeben.	
Funktionale Anforderungen:	F-CA-PP-9	
Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - Rudder - Angle - Speed Die nächsten 2 Wegpunkte vom Path-Planning	
Testschritte:	TCS-01:Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Im Controller die konstante Geschwindigkeit konfigurieren. TCS-03: Überprüfen der Geschwindigkeit des Schiffes mit MATE Control.	

Erwartetes Ergebnis:	Schiff fährt mit der vorgegebenen Geschwindigkeit von 3.60 m/s (7.00 Knoten).	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-5.1	Bemerkungen:
Test Titel:	Controller Wegpunktpaar vom Path-Planning empfangen	
Beschreibung:	Der Controller empfängt vom Path-Planning ein Wegpunktpaar.	
Funktionale Anforderungen:	F-CA-PP-9	
Eingehende-Daten:	Die nächsten 2 Wegpunkte vom Path-Planning	

Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen ob der Controller den gesendeten Wegpunkt von Path-Planning empfängt mit MATLAB Simulink Überwachungstools.	
Erwartetes Ergebnis:	Controller empfängt die nächsten anzufahrenden Wegpunkte von der Path-Planning Komponente	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-5.2	Bemerkungen:
Test Titel:	Controller Wegpunkt abfahren	
Beschreibung:	Das Schiff kann in Richtung eines im Controller hinzugefügten Wegpunkts fahren.	
Funktionale Anforderungen:		
Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - Rudder - Angle - Speed Die nächsten 2 Wegpunkte	
Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen ob der Controller in Richtung des eingegebenen Wegpunkts fährt mit MATE Control.	

Erwartetes Ergebnis:	Schiff fährt in die Richtung des im Controller gegebenen Wegpunktes mit einer erlaubten Abweichung von 60 m.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.1	Bemerkungen:
Test Titel:	Anzeige AIS-Daten	
Beschreibung:	Anzeige der AIS-Daten in der EPD.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.1	
Eingehende-Daten:	- Sensor-Daten (kontinuierlich) vom HUB über den NMEA-Sensor - Koordinaten erkannter Schiffe/Objekte (AIS, Radar und	

	fusionierte Daten) als JSON via HTTP von der DOI	
Testschritte:	TCS-01: Empfang von AIS-Daten per UDP vom HUB und Darstellung auf der EPD-Karte TCS-02: Überprüfung von AIS-Roh-Daten von der DOI (Layer: DOIDataLayer)	
Erwartetes Ergebnis:	Anzeige der von der DOI übermittelten AIS-Daten im Layer „DOIDataLayer“ der EPD.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.2	Bemerkungen:
Test Titel:	Anzeige Radar-Daten	
Beschreibung:	Anzeige der Radar-Daten in der EPD.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.2	
Eingehende-Daten:	- Sensor-Daten (kontinuierlich) - Koordinaten erkannter Schiffe/Objekte (AIS, Radar und fusionierte Daten) als JSON via HTTP von der DOI	
Testschritte:	TCS-01: Überprüfung von Radar-Roh-Daten von der DOI (Layer: DOIDataLayer)	
Erwartetes Ergebnis:	Anzeige der von der DOI übermittelten Radar-Daten im Layer „DOIDataLayer“ der EPD	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.3	Bemerkungen:
Test Titel:	Anzeige Grid-Daten	
Beschreibung:	Anzeige der Grid-Daten in der EPD. Diese werden als JSON vom Route-Planning nach Optimierung einer nicht-optimierten Route an die EPD übertragen.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.3	
Eingehende-Daten:	- Routendaten - NoGo-Daten und Griddaten als JSON via HTTP vom Route-Planning (bereits vom Routenberechnen vorhanden)	
Testschritte:	TCS-01: Überprüfung der Grid-Daten anhand der EPD Karte (Layer: SOIDataLayer)	

Erwartetes Ergebnis:	Anzeige der vom Route-Planning übermittelten Grid-Daten im Layer „SOIDataLayer“ der EPD	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.4	Bemerkungen:
Test Titel:	Anzeige Polygone	
Beschreibung:	Anzeige der Polygone in der EPD. Diese werden als JSON vom Route-Planning nach Optimierung einer nicht-optimierten Route an die EPD übertragen.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.4	
Eingehende-Daten:	- Routendaten	

	- NoGo-Daten und Griddaten als JSON via HTTP vom Route-Planning (bereits vom Routenberechnen vorhanden)	
Testschritte:	TCS-01: Überprüfung der Polygone anhand der EPD Karte (Layer: SOIDataLayer)	
Erwartetes Ergebnis:	Anzeige der vom Route-Planning übermittelten Polygone im Layer „SOIDataLayer“ der EPD	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.5	Bemerkungen:
Test Titel:	Anzeige der eigenen Schiffsposition	
Beschreibung:	Anzeige der eigenen Schiffsposition in der EPD.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.5	
Eingehende-Daten:	- Sensor-Daten (kontinuierlich)	
Testschritte:	TCS-01: Überprüfung der eigenen Schiffsposition anhand der EPD Karte und MATE Control	
Erwartetes Ergebnis:	Anzeige der korrekten, eigenen Schiffsposition auf der EPD-Seekarte, mit korrektem Heading. Position aus MATE Control und der EPD stimmt überein.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-7	Bemerkungen:
Test Titel:	Virtual Handles	Wurden nicht getestet!
Beschreibung:	Anwendung der Virtual Handles zur Steuerung der Zuse über ein webbasiertes Interface (Ruder und RPM).	
Funktionale Anforderungen:	F-EPD-9 F-EPD-10 F-EPD-11	
Eingehende-Daten:	Steuerkommandos (RPM und Ruder-Winkel) über die webbasierte Oberfläche der VirtualHandles (View in der EPD)	
Testschritte:	TCS-01: Virtual Handles Interface in der EPD öffnen TCS-02: Virtual Handles-Steuerung über die „Power“-Schaltfläche (oben rechts) einschalten. TCS-03: Durch die Nutzung der VirtualHandles Steuerbefehle die Steuerung des Schiffs übernehmen ○ Ruder-Winkel: maximal-Einschlag von 20°	

	○ RPM: maximal mögliche Drehzahl: 2000 RPM TCS-04: Virtual Handles-Steuerung über erneutes Klicken der „Power“-Schaltfläche (oben rechts) ausschalten.	
Erwartetes Ergebnis:	- VirtualHandles senden Steuerbefehle (zyklisch und bei Änderung der Einstellgrößen) ○ Ruder-Winkel: maximal-Einschlag von 20° Steuerbord ○ RPM: maximal mögliche Drehzahl: 2000 RPM - Schiff lässt sich per Virtual Handles steuern ○ Ruder-Winkel: das Schiff steuert nach rechts ○ RPM: das Schiff beschleunigt auf maximal 2000 RPM - Nach Abschalten der Virtual Handles wird die Steuerung wieder abgegeben. Der Controller deaktiviert dabei den manuellen Fahrmodus. ○ Der Ruder-Winkel verstellt sich in den Normalzustand	

	○ Die Geschwindigkeit wird auf 700 RPM gedrosselt	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-8	Bemerkungen:
Test Titel:	Objekterkennung (vor dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (vor dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen. Zusätzlich werden JSON-Objekte über http von erkannten Schiffe mittels Bilderkennung von der Object Detection empfangen	

Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.1 F-DOI-4	
Eingehende-Daten:	- Sensordaten des AIS in NMEA-2000 Format, mit erkanntem Schiff welches bis zu 3500m von der eigenen Position entfernt ist. - Sensordaten des Radars in NMEA-2000 konformen Format, mit erkanntem Schiff welches bis zu 3500m von der eigenen Position entfernt ist. - JSON-Objekt über die Position (rechts, links, beides) welches von der Object Detection erkannt wurde	
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layer an und prüfe, ob umliegenden Schiffe von Radar und AIS angezeigt werden.	

Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten von Radar werden von der DOI-Komponente empfangen - Erkannte Schiffe von der Object-Detection-Komponente werden empfangen	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-9.1	Bemerkungen:
Test Titel:	Objekterkennung (hinter dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (hinter dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen.	
Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.3	

Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format, mit erkanntem Schiff, welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet - Sensordaten des Radars im NMEA-2000 konformen Format, mit erkanntem Schiff welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet	
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe, welche von der Object-Detection-Komponente erkannt wurden, (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten vom	-

	Radar werden von der DOI-Komponente empfangen	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-9.2	Bemerkungen:
Test Titel:	Objekterkennung (links neben dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (links neben dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen.	

Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.2	
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format, mit erkanntem Schiff, welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet - Sensordaten des Radars im NMEA-2000 konformen Format, mit erkanntem Schiff welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet	
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe, welche von der Object-Detection-Komponente erkannt wurden, (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an	

Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten vom Radar werden von der DOI-Komponente empfangen	-
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-9.3	Bemerkungen:
Test Titel:	Objekterkennung (rechts neben dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (rechts neben dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen.	

Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.2	
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format, mit erkanntem Schiff, welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet - Sensordaten des Radars im NMEA-2000 konformen Format, mit erkanntem Schiff welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet	
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe, welche von der Object-Detection-Komponente erkannt wurden, (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an	

Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten vom Radar werden von der DOI-Komponente empfangen	-
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-10.1	Bemerkungen:
Test Titel:	Datenfusion (Größe des Schiffs bis 20m)	
Beschreibung:	Die DOI-Komponente muss empfangene AIS-Nachrichten (statisch und dynamisch) und	

	empfangene Radar Tracks untereinander fusionieren (AIS mit AIS und AIS mit Radar)	
Anforderung	F-DOI-6 F-DOI-7	
Eingehende-Daten:	- Sensordaten des Radars im NMEA-2000 konformen Format in einem Umkreis ums eigenen Schiff von bis zu 2 sm (mind. 15 Nachrichten) - Sensordaten des GPS in NMEA-2000 - Sensordaten des Headings in NMEA-2000 - JSON-Objekt über die Position (rechts, links) welche durch Bildverarbeitung erkannt wurde	-
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Fusioniere die Schiffe (automatisch)	

	TCS-04: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an TCS-05: Zeige die fusionierten Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	Schiffe unter 20 m müssen kein AIS besitzen. Daher wird das Schiff nur mit Radar erkannt. Der empfangene Radartrack wird nicht mit einem AIS-Track fusioniert. D.h. der EPD-DOI-Layer zeigt ein Schiff, welches von Radar erkannt wurde, an aber nicht mit AIS fusioniert wurde (FRadarXXX).	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-10.2	Bemerkungen:
Test Titel:	Datenfusion (Größe des Schiffs ab 20m)	
Beschreibung:	Die DOI-Komponente muss empfangene AIS-Nachrichten (statisch und dynamisch) und empfangene Radar Tracks untereinander fusionieren (AIS mit AIS und AIS mit Radar). Die DOI-Komponente muss bei allen assoziierten Schiffen in der Umgebung (2 sm Umkreis ums eigene Schiff) die objektspezifischen Parameter Größe, Geschwindigkeit und Position neu berechnen.	
Anforderung	F-DOI-6 F-DOI-7 F-DOI-8	
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 4 Positionsnachrichten und mind. 1 x statische AIS-Nachricht von einem Schiff)	

	- Sensordaten des Radars im NMEA-2000 konformen Format in einem Umkreis ums eigenen Schiff von bis zu 2sm (mind. 15 Nachrichten) - Sensordaten des GPS in NMEA-2000 Sensordaten des Headings in NMEA-2000	
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Positionen der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Fusioniere die Schiffe (automatisch) TCS-04: Gleiche bei den fusionierten Schiffe, die Geschwindigkeit, Position und Größe an (automatisch) TCS-05: Zeige die Rohdaten der Schiffe über den EPD-Layer an TCS-06: Zeige die fusionierten Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	AIS-Nachrichten von Schiffen, die sich in einem 2 sm Umkreis vom eigenen Schiff befinden, werden bei gleicher User ID einander	

	zugeordnet. Radartracks von Schiffen, die sich in einem 2 sm Umkreis vom eigenen Schiff befinden, werden bei gleicher User ID einander zugeordnet. AIS-Nachrichten und Radar-Tracks eines Schiffes im Umkreis von 2 sm werden einander zugeordnet, wenn es sich um dasselbe Schiff handelt. - Bei der Simulation werden die AIS-Nachrichten mit der MMSI 123456789 und den Radarnachrichten mit der MMSI 60 ab 15 empfangenen Radarnachrichten und 4 AIS-Nachrichten durchgehend fusioniert. Im EPD-DOI-Layer ist ein Schiff mit der Bezeichnung FRadar60AIS123456789 zu sehen.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.1	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit der EPD (empfangen von Daten von der EPD)	
Beschreibung:	Entgegennahme der validen globalen Route von der EPD.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	

Eingehende-Daten:	- Route von der EPD	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und EPD	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente empfängt eine valide, globale Route von der EPD	-
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.2	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit dem NoGoSolver	

Beschreibung:	Die Path-Planning-Komponente muss kontinuierlich Informationen über alle statischen Hindernisse vom NoGoSolver anfragen, entgegennehmen und decodieren können.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- JSON-Objekte vom NoGoSolver	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und NoGoSolver	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente fragt statische Hindernisse beim NoGoSolver an und empfängt von diesem die statischen Hindernisse.	

Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.3	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit der DOI	
Beschreibung:	Das Path-Planning muss kontinuierlich Informationen über alle dynamischen Hindernisse (Schiffe) von der DOI-Komponente entgegennehmen und decodieren können.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- JSON-Objekte von der DOI-Komponente	

Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und DOI-Komponente	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente empfängt dynamische, zum Teil fusionierte Daten von der DOI-Komponente, welche alle Schiffe in einem 2 sm Radius um das eigene Schiff repräsentieren.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.4	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit dem Controller (senden von Daten an den Controller)	

Beschreibung:	Der berechnete valide Pfad wird über das Distribution System an den Controller geschickt.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Distribution System	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und Controller über das Distribution System	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente sendet Daten (nächster Wegpunkt, RoutenStatus, wished ship speed) an den Controller über das Distribution System	-

Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.5	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit der EPD (senden von Daten an die EPD)	
Beschreibung:	Der berechnete valide Pfad wird über das Distribution System an die EPD geschickt.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Distribution System	

Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-05: Überprüfung der Verbindung zwischen Path-Planning und EPD über das Distribution System	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente sendet den aktuellen Pfad an die EPD über das Distribution System (optional: Debug-Ausgaben, wie Goodwin-Shapes / statische Hindernisse)	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

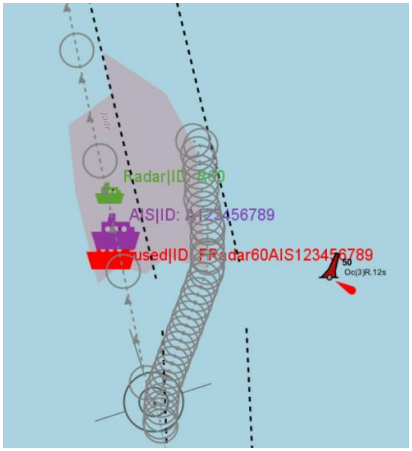
ID:	TC-12	Bemerkungen:
Test Titel:	Path-Planning Berücksichtigung dynamischer und statischer Hindernisse	Statisches Hindernis: 53.51218; 8.16608 Wurde nicht explizit getestet.
Beschreibung:	Ausweichen eines dynamischen Objektes (Schiff) unter Berücksichtigung eines nahen statischen Hindernisses.	
Anforderung	F-CA-PP-5	
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub	
Testschritte:	TCS-01: Wegpunktwerkzeug in der EPD auswählen und auf der Karte eine Route zusammenstellen die entlang statischer Hindernisse führt	

	TCS-02: Schiffsparameter(minimal_turn_radius, maximum_turning_angle) in den configuration.properties einstellen TCS-03: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-04: Den berechneten Pfad der Path-Planning überwachen	
Erwartetes Ergebnis:	- Das System weicht dem dynamischen Objekt aus ohne dabei mit den statischen Hindernissen zu kollidieren	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-13	Bemerkungen:
Test Titel:	Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	
Beschreibung:	Ausweichen eines dynamischen Objektes (Schiff) unter Berücksichtigung der physikalischen Eigenschaften für das im Test eingesetzte Schiff (ZUSE).	
Anforderung	F-CA-PP-6	
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub	
Testschritte:	TCS-01: Schiffsparameter(minimal_turn_radius, maximum_turning_angle) in den configuration.properties einstellen TCS-02: Senden der ausgewählten und optimierten Route an das Path-Planning	

	durch Auswählen der Route und Klicken auf “Route starten” TCS-03: Überwachen des berechneten Pfades auf Einhaltung der Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	
Erwartetes Ergebnis:	- Der berechnete Pfad steht nicht im Konflikt mit den physikalischen Eigenschaften (Wendewinkel der Zuse) des im Test eingesetzten Schiffs, d.h. der Wendewinkel beträgt maximal 30 Grad)	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-14.1	Bemerkungen:
Test Titel:	Path-Planning Überholen	
Beschreibung:	Überholen eines dynamischen Objektes (Schiff) unter Berücksichtigung der COLREG Regel 13.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen (Überholen, COLREG 13) TCS-05b: Das Ausweichen des Schiffes überwachen (Entgegenkommen, COLREGs 14, 16) TCS-05c: Das Ausweichen des Schiffes überwachen (Kreuzen, COLREGs 15, 16)	

	TCS-05d: Das Ausweichen des Schiffes überwachen (Zwangsweises Ausweichen, COLREGs 15, 17)	
Erwartetes Ergebnis:	Die Zuse überholt das Fremdschiff auf der linken oder rechten Seite mit einem ausreichenden Mindestabstand von 100 Metern und folgt nach dem Überholmanöver wieder der zuvor definierten Route. 	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-14.2	Bemerkungen:
Test Titel:	Path-Planning entgegenkommend	
Beschreibung:	Ausweichen eines entgegenkommenden dynamischen Objektes (Schiff) unter Berücksichtigung der COLREGs Regeln 14, 16, 17.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen	
Erwartetes Ergebnis:	Die Zuse weicht dem entgegenkommenden Fremdschiff nach rechts hin mit einem ausreichenden Mindestabstand von 100 Metern aus und folgt nach dem	

	Ausweichmanöver wieder der zuvor definierten Route.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-14.3	Bemerkungen:
Test Titel:	Path-Planning Kreuzen von rechts	
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von rechts gekreuzt unter Berücksichtigung der COLREGs Regeln 15, 16, 17.	
Anforderung	F-CA-PP-8	

Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen (Kreuzen, COLREGs 15, 16)	
Erwartetes Ergebnis:	Die Zuse weicht dem Fremdschiff nach links hin mit einem ausreichenden Mindestabstand von 100 Metern aus und folgt nach dem Ausweichmanöver wieder der zuvor definierten Route.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-14.4.1	Bemerkungen:
Test Titel:	Path-Planning Kreuzen von links (Kurs beibehalten)	
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von links gekreuzt unter Berücksichtigung der COLREGs Regeln 15, 16, 17.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Überwachung des Kreuz-Manövers	

Erwartetes Ergebnis:	Die Zuse verfolgt weiterhin den aktuell vorgegebenen Pfad.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-14.4.2	Bemerkungen:
Test Titel:	Path-Planning Kreuzen von links (Ausweichen)	
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von links gekreuzt, das andere Schiff verletzt die COLREGs Regeln und weicht nicht aus, daraufhin muss das zu testende Schiff zwangsweise Ausweichen unter Berücksichtigung der COLREGs Regeln 15, 16, 17.	

Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route von der EPD - JSON-Objekte vom NoGoSolver - JSON-Objekte von der DOI-Komponente - Position vom Hub	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Überwachung des erzwungenen Ausweichens während des Kreuz-Manövers	
Erwartetes Ergebnis:	Die Zuse weicht dem Fremdschiff nach links oder rechts hin mit einem ausreichenden Mindestabstand von 100 Metern aus und folgt nach dem Ausweichmanöver wieder der zuvor definierten Route.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-15.1	Bemerkungen:
Test Titel:	Schiffserkennung mittels Kamera	
Beschreibung:	Die Object Detection Komponente muss kontinuierlich die Umgebung in Fahrtrichtung mittels einer Kamera erfassen und Schiffe erkennen. Für erkannte Schiffe, die unter dem Horizont liegen, wird die Position (rechts oder links vom eigenen Schiff) an die DOI als JSON-Objekt gesendet.	
Anforderung	F-OD-1 F-OD-2 F-OD-3 F-OD-4 F-OD-5	
Eingehende-Daten:	Echtzeit-Videoaufnahmen einer Webcam von der Umgebung vor dem Schiff	

Testschritte:	TCS-01: „Webcam Selection“ öffnen (durch starten der Komponente), die Webcam „HD Pro Webcam C920 0“ auswählen und auf „Select Camera“ klicken TCS-02: Überprüfen der gesendeten Schiffe an die DOI mit MATE Control. TCS-03: (nach der Testfahrt) 200 aufgenommenen Bilder mit Boxen um erkannte Schiffe auswerten.	
Erwartetes Ergebnis:	● Schiffe, die sich in Fahrtrichtung befinden, werden als Schiff markiert: Die nachträgliche Auswertung der aufgenommen Bildern ergibt, dass 50% der auf den 200 Bildern zu sehenden Schiffe von der Object Detection erkannt wurde. ● Die Position (RIGHT, LEFT, BOTH oder NOTHING) der mittels Bildverarbeitung erkannten Schiffe, die sich unter dem Horizont befinden, werden als JSON-Objekt an die DOI gesendet.	
Eingetretenes Ergebnis:		

Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-15.2	Bemerkungen:
Test Titel:	Erkannte Schiffe der Object Detection an PP und EPD senden.	
Beschreibung:	Verarbeitung der von der Object Detection Komponente empfangenen JSON-Objekte über die Position der erfassten Schiffe mit der Kamera. Es wird geprüft, ob die erkannten Schiffe bereits mit AIS und/oder Radar erkannt wurden. Ist dies nicht der Fall wird ein „Kameraschiff“ in einem 45° Winkel zur eigenen Fahrrichtung mit einer Geschwindigkeit von 0 Knoten erstellt. Dieses Schiff wird an die EPD und an das Path Planning gesendet.	

Anforderung	F-DOI-9	
Eingehende-Daten:	JSON-Objekte von der Object Detection über die Position (RIGHT, LEFT, BOTH oder NOTHING) der mittels Bildverarbeitung erkannten Schiffe, die sich unter dem Horizont befinden.	
Testschritte:	TCS-01: „Webcam Selection“ öffnen (durch starten der Komponente), die Webcam „HD Pro Webcam C920 0“ auswählen und auf „Select Camera“ klicken TCS-03: In der Konfiguration (configuration.properties) der DOI einstellen, dass nur simulierte Schiffe verarbeitet werden sollen (simulatedOrRealData=simulated) TCS-03: Zeige die fusionierten Schiffe über den EPD-DOI-Layers an	

Erwartetes Ergebnis:	● Die von der Object Detection erkannten Schiffe werden nicht mit Radar oder AIS erkannt, da keine AIS- und Radardaten verarbeitet werden. ● Wenn sich ein Schiff vor dem eigenen Schiff befinden, ist dieses in dem EPD-DOI-Layer mit der Bezeichnung „Camera“ zu sehen.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

Fehlerinjektion:

ID:	TC-FI-1	Bemerkungen:
Test Titel:	ownPosition-Injection	
Beschreibung:	Ein anderes, bereits vorher über AIS identifiziertes Schiff wechselt die Position auf die Position des eigenen Schiffes.	

Eingehende-Daten:	- Schiffssensorinformationen
	  - Schiffsposition
	  - Schiffsrichtung
	  - AIS
	  - (Radar)	

Testschritte:	TCS-01: TCS-02: TCS-03:	
Erwartetes Ergebnis:		

Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:		

ID:	TC-FI-2	Bemerkungen:
Test Titel:	Ungültiger Ruderwinkel	
Beschreibung:	Der Controller empfängt einen außerhalb der spezifizierten Grenzen definierten Ruderwinkel.	
Eingehende-Daten:	- Ruderwinkel per NMEA 2000 (an Controller)	
Testschritte:	TCS-01: Abfangen der Ruderwinkel-Nachrichten mit MATE-Control TCS-02: Ändern und senden der Ruderwinkel-Nachricht zu einem Wert	

	außerhalb der spezifizierten Grenzen (180°) TCS-03: Überwachung des Systems mit MATE-Control	
Erwartetes Ergebnis:	Das System verhält sich unverändert.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:		

ID:	TC-FI-3	Bemerkungen:
Test Titel:	Schiff verschwindet (Afrika-Injektion)	
Beschreibung:	Ein existierendes (real oder simuliert), über AIS erkanntes Schiff verschwindet von seiner bekannten Position.	

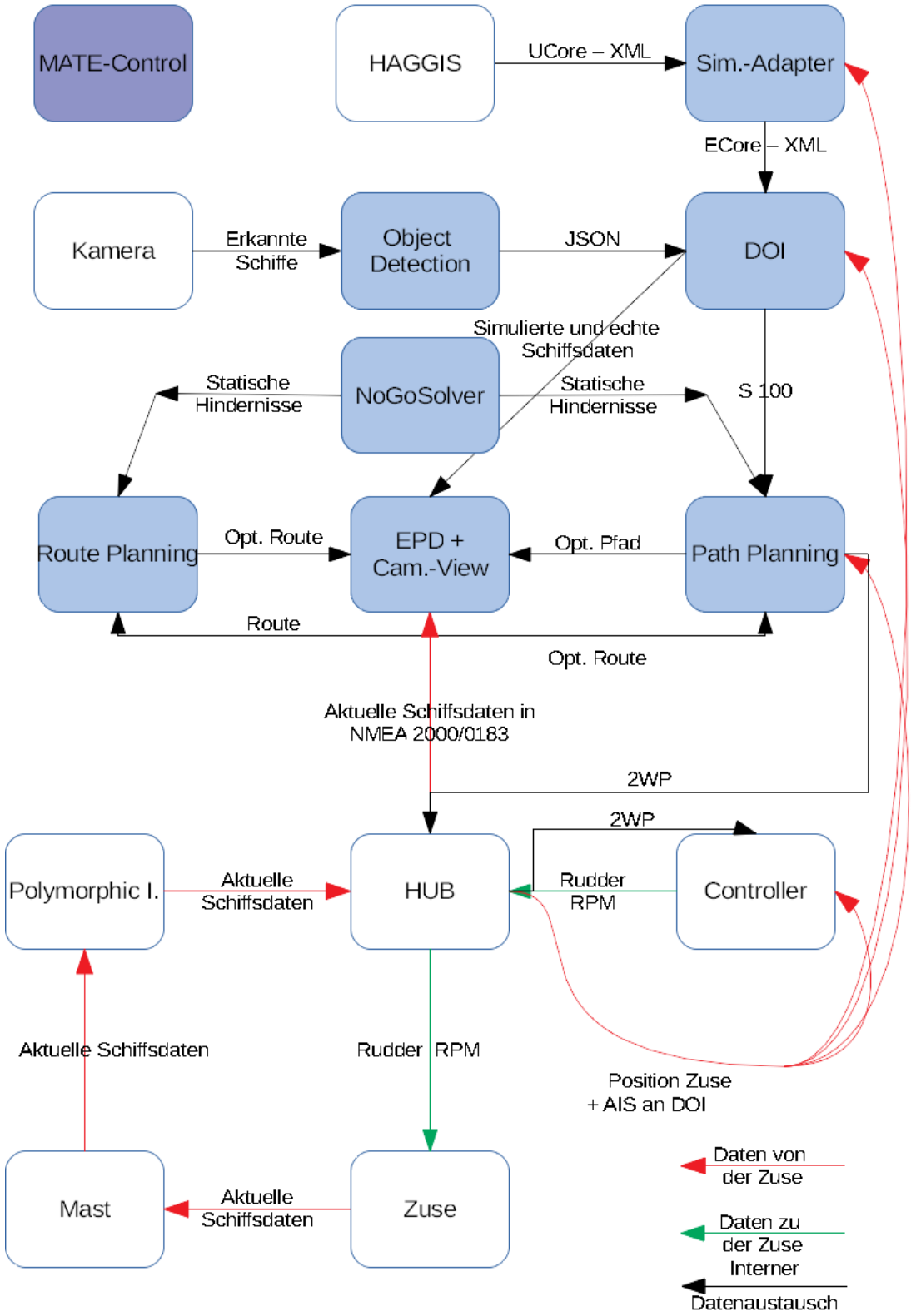
Eingehende-Daten:		
Testschritte:	TCS-01: TCS-02: TCS-03:	
Erwartetes Ergebnis:	Das über AIS erkannte Schiff wird noch weitere 30 Sekunden im Speicher gehalten und in dem Zeitraum auch an die EPD und an das PP gesendet. Nach 30 Sekunden wird das Schiff in der DOI gelöscht und nicht mehr versendet. In dem EPD-DOI-Layer ist die Repräsentation des Schiffes mit der Bezeichnung FAISXXX noch 30 Sekunden nach Erhalt der letzten Nachricht dieses Schiffes zu sehen.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:		

ID:	TC-FI-4	Bemerkungen:
Test Titel:	Schiff taucht auf (Afrika-Injektion)	
Beschreibung:	Ein zuvor verschwundenes (real oder simuliert), über AIS erkanntes Schiff erscheint wieder auf seiner echten Position.	
Eingehende-Daten:		
Testschritte:	TCS-01: TCS-02:	
Erwartetes Ergebnis:	Das erkannte Schiff wird wieder erfasst und an die EPD und an das PP gesendet. In dem EPD-DOI-Layer ist das Schiff mit der Bezeichnung FAISXXX zu sehen.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		

Testmethode:		

ID:	TC-FI-05	Bemerkungen:
Test Titel:	Wechsel der MMSI	
Beschreibung:	Ein mit AIS und oder Radar identifiziertes Schiff in der Umgebung des eigenen Schiffs wechselt seine MMSI.	
Eingehende-Daten:		
Testschritte:	TCS-01: TCS-02:	
Erwartetes Ergebnis:	Für jede neue MMSI wird ein Schiff in der DOI gespeichert. Diese Schiffe werden an die EPD und an das PP gesendet. In dem EPD-DOI-Layer werden somit mehrere Repräsentation des Schiffes angezeigt.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:		

Gesamtübersicht:


MATE-Control
HAGGIS
UCore – XML
Sim.-Adapter
ECore – XML
Kamera
Erkannte Schiffe
Object Detection
JSON
DOI
Simulierte und echte Schiffsdaten
Statische Hindernisse
NoGoSolver
Statische Hindernisse
S 100
Route Planning
Opt. Route
EPD + Cam.-View
Opt. Pfad
Path Planning
Route
Opt. Route
Aktuelle Schiffsdaten in NMEA 2000/0183
2WP
2WP
Polymorphic I.
Aktuelle Schiffsdaten
HUB
Rudder RPM
Controller
Aktuelle Schiffsdaten
Rudder RPM
Position Zuse + AIS an DOI
Mast
Aktuelle Schiffsdaten
Zuse
Daten von der Zuse
Daten zu der Zuse
Interner Datenaustausch

Anlage 1: optimierte RTZ-Route

WP1:
lat: 53.51765489239272
lon: 8.17657470703125
WP2:
lat: 53.51941397091744
lon: 8.175958462627543
WP3:
lat: 53.52117304574505
lon: 8.175342167169969
WP4:
lat: 53.52293211687508
lon: 8.174725820651123
WP5:
lat: 53.524691184307095
lon: 8.174109423063586
WP6:
lat: 53.526817548437876
lon: 8.176445103099207
WP7:
lat: 53.52857662358174
lon: 8.175828720333746
WP8:
lat: 53.53033569502767
lon: 8.175212286487849
WP9:
lat: 53.53246203559427
lon: 8.177548332323932
WP10:
lat: 53.53422111475336
lon: 8.176931913315126
WP11:
lat: 53.53634742033682
lon: 8.17926825913435
WP12:
lat: 53.53810650720982
lon: 8.178651854982
WP13:
lat: 53.5398655903855
lon: 8.17803539973313
WP14:
lat: 53.541624669863445
lon: 8.177418893380318
WP15:
lat: 53.543750963280054
lon: 8.17975567121135
WP16:
lat: 53.54551005047374
lon: 8.1791391797257
WP17:
lat: 53.547269133969756
lon: 8.178522637124349
WP18:
lat: 53.54902821376765
lon: 8.177906043399876
WP19:
lat: 53.55078728986697
lon: 8.177289398544849
WP20:
lat: 53.552546362267265
lon: 8.17667270255184
WP21:
lat: 53.55430543096807
lon: 8.176055955413423
WP22:
lat: 53.55606449596894
lon: 8.175439157122161
WP23:
lat: 53.55782355726942
lon: 8.174822307670626
WP24:
lat: 53.55958261486906
lon: 8.17420540705138
WP25:
lat: 53.561341668767376
lon: 8.173588455256988
WP26:
lat: 53.563100718963945
lon: 8.172971452280017
WP27:
lat: 53.5648597654583
lon: 8.172354398113024
WP28:
lat: 53.566618808249984
lon: 8.17173729274857
WP29:
lat: 53.56837784733853
lon: 8.171120136179217
WP30:
lat: 53.57013688272353
lon: 8.17050292839752
WP31:
lat: 53.57189591440448
lon: 8.169885669396034
WP32:
lat: 53.573654942380934
lon: 8.169268359167317
WP33:
lat: 53.57541396665244
lon: 8.168650997703919
WP34:
lat: 53.57717298721855
lon: 8.168033584998396
WP35:
lat: 53.57893200407879
lon: 8.167416121043292
WP36:
lat: 53.58069101723272
lon: 8.166798605831163
WP37:
lat: 53.582450026679886
lon: 8.166181039354553
WP38:
lat: 53.584209032419814
lon: 8.16556342160601
WP39:
lat: 53.58596803445207
lon: 8.16494575257808
WP40:
lat: 53.58772703277619
lon: 8.164328032263299
WP41:
lat: 53.58948602739171
lon: 8.16371026065422
WP42:
lat: 53.59124501829817
lon: 8.163092437743378
WP43:
lat: 53.59300400549512
lon: 8.162474563523311
WP44:
lat: 53.59476298898212
lon: 8.161856637986562
WP45:
lat: 53.59652196875869
lon: 8.161238661125662
WP46:
lat: 53.59828094482437
lon: 8.160620632933151
WP47:
lat: 53.60003991717871
lon: 8.160002553401561
WP48:
lat: 53.60179888582128
lon: 8.159384422523424
WP49:
lat: 53.603557850751585
lon: 8.158766240291273
WP50:
lat: 53.60531681196919
lon: 8.158148006697635
WP51:
lat: 53.60707576947363
lon: 8.157529721735038
WP52:
lat: 53.60883472326444
lon: 8.156911385396013
WP53:
lat: 53.61059367334118
lon: 8.156292997673082
WP54:
lat: 53.61235261970337
lon: 8.155674558558768
WP55:
lat: 53.61411156235057
lon: 8.155056068045596
WP56:
lat: 53.615870501282316
lon: 8.154437526126088
WP57:
lat: 53.61762943649817
lon: 8.153818932792761
WP58:
lat: 53.619388367997644
lon: 8.153200288038136
WP59:
lat: 53.621147295780275
lon: 8.152581591854732
WP60:
lat: 53.62290621984563
lon: 8.15196284423506
WP61:
lat: 53.62466514019325
lon: 8.151344045171633
WP62:
lat: 53.62642405682268
lon: 8.15072519465697
WP63:
lat: 53.62818296973344
lon: 8.15010629268358
WP64:
lat: 53.62994187892507
lon: 8.149487339243972
WP65:
lat: 53.631700784397125
lon: 8.148868334330656
WP66:
lat: 53.63345968614916
lon: 8.148249277936138
WP67:
lat: 53.63521858418069
lon: 8.147630170052926
WP68:
lat: 53.636977478491275
lon: 8.147011010673522
WP69:
lat: 53.638736369080455
lon: 8.14639179979043
WP70:
lat: 53.640495255947755
lon: 8.145772537396152
WP71:
lat: 53.64225413909272
lon: 8.14515322348319
WP72:
lat: 53.64401301851491
lon: 8.144533858044039
WP73:
lat: 53.64577189421386
lon: 8.143914441071196
WP74:
lat: 53.647530766189085
lon: 8.143294972557163
WP75:
lat: 53.649289634440144
lon: 8.142675452494432
WP76:
lat: 53.651048498966574
lon: 8.142055880875498
WP77:
lat: 53.6524392023931
lon: 8.138475642429457
WP78:
lat: 53.65419804426581
lon: 8.137855849923817
WP79:
lat: 53.65595688241105
lon: 8.13723600582885
WP80:
lat: 53.657715716828356
lon: 8.136616110137037
WP81:
lat: 53.65910625617243
lon: 8.13303512809199
WP82:
lat: 53.660865067920284
lon: 8.132415011396885
WP83:
lat: 53.662255481013496
lon: 8.128833624346225
WP84:
lat: 53.66401427008092
lon: 8.128213286582469
WP85:
lat: 53.66540455689565
lon: 8.124631494455508
WP86:
lat: 53.667163323271666
lon: 8.124010935557715

Systemtest vom 29.08.2018

Integrationstest:

Methode.
Zunächst erfolgt ein Systemintegrationstest des PG-MATE Systems und LABSKAUS. Der Systemintegrationstest verifiziert die korrekte Funktionsweise der polymorphen Schnittstelle hinsichtlich einer erfolgten Kommunikationsverbindung, sowie der syntaktischen Korrektheit der Daten. Anschließend beurteilt ein Systemexperte des PG-MATE Systems die semantische Korrektheit einiger empfangener Live-Daten der mobilen Sensorbox, die dem PG-MATE Systems durch LABSKAUS bereitgestellt und ein plausibles Ergebnis anhand der vorliegenden Situation der Realwelt validiert wird.

Testdurchführung.

Testspezifikation:

Zur Vorbereitung des Systemtests werden die Komponenten des Systems EPD, Route Planning, Path Planning, DOI, Object Detection und der Controller gestartet. Zusätzlich werden Haggis, der Simulations Adapter und der NoGoSolver gestartet und wie die Testumgebung mit dem System verbunden. Mit den geplanten Testszenarien werden die meisten funktionalen Anforderungen an das System verifiziert und zusätzlich werden die COLREG Regeln 13-17 abgedeckt. Die in den Szenarien benötigten Schiffe werden mit Hilfe von Haggis simulativ über den Simulations Adapter in die Testumgebung eingespeist.

Testkonfiguration:

Für die Szenarien des Systemtests erhält das zu testende Schiff eine Route, in der jedes Szenario durchgespielt wird. Diese Route wurde im Vorfeld des Systemtests bereits zusammengestellt und in der EPD abgespeichert. Im Folgenden werden die Abschnitte für jedes Szenario aufgelistet.

Testszenario	Abschnitt

TS-2	- WP-1:
	  - Lat: 53.5630989074707
	  - Long: 8.172971725463867
	- WP-2:
	  - Lat: 53.56486129760742
	  - Long: 8.172354698181152
	- WP-3:
	  - Lat: 53.566619873046875
	  - Long: 8.171737670898438
	- WP-4:
	  - Lat: 53.56837844848633
	  - Long: 8.171119689941406
	- WP-5:
	  - Lat: 53.57013702392578
	  - Long: 8.170502662658691
	- WP-6:
	  - Lat: 53.571895599365234
	  - Long: 8.169885635375977
	- WP-7:

	- Lat: 53.57365417480469 - Long: 8.169268608093262

TS-3	- WP-1:
	  - Lat: 53.53246307373047
	  - Long: 8.1775484085083
	- WP-2:
	  - Lat: 53.53422164916992
	  - Long: 8.17693233489
	- WP-3:
	  - Lat: 53.536346435546875
	  - Long: 8.179267883300781
	- WP-4:
	  - Lat: 53.53986740112305
	  - Long: 8.178035736083984
	- WP-5:
	  - Lat: 53.5416259765625
	  - Long: 8.17741870880127
	- WP-6:
	  - Lat: 53.54375076293945
	  - Long: 8.179755210876465
	- WP-7:
	  - Lat: 53.545509338378906
	  - Long: 8.179139137268066

TS-5	- WP-1:
	  - Lat: 53.545509338378906
	  - Long: 8.179139137268066
	- WP-2:
	  - Lat: 53.54726791381836
	  - Long: 8.178523063659668
	- WP-3:
	  - Lat: 53.54902648925781
	  - Long: 8.177906036376953
	- WP-4:
	  - Lat: 53.55078887939453
	  - Long: 8.177289009094238
	- WP-5:
	  - Lat: 53.552547454833984
	  - Long: 8.1766729354858
	- WP-6:
	  - Lat: 53.55430603027344
	  - Long: 8.176055908203125
	- WP-7:
	  - Lat: 53.55606460571289
	  - Long: 8.17543888092041

Des Weiteren müssen für die Szenarien 2-5 die Routen der simulierten Schiffe konfiguriert werden, diese wurden in Abhängigkeit zur Route des zu testenden Schiffes entwickelt.

Testszenario	Route

TS-2	- WP-1:
	  - Lat: 53.568378448
	  - Long: 8.171119689941
	- WP-2:
	  - Lat: 53.57365417
	  - Long: 8.1692686080

TS-3	- WP-1:
	  - Lat: 53.53416
	  - Long: 8.18306
	- WP-2:
	  - Lat: 53.53449
	  - Long: 8.17168

TS-5	- WP-1:
	  - Lat: 53.552547
	  - Long: 8.1766729
	- WP-2:
	  - Lat: 53.54375076
	  - Long: 8.17975521

Basis-Testfälle

ID:	Basis-Test 0 (TC-4, Delmenhorst)	Bemerkungen:
Test Titel:	Controller Wegpunkt abfahren	- Ruderregelung funktioniert ordnungsgemäß. - Dem gesetzten Kurs wird erfolgreich gefolgt - Geschwindigkeitsregelung funktioniert ordnungsgemäß. - Regelung erfolgt entsprechend der erwarteten Ergebnisse
Beschreibung:	Das Schiff kann in Richtung eines konstanten, im Controller hinzugefügten, Wegpunkts fahren.	
Funktionale Anforderungen:	F-CA-CT-1, F-CA-CT-4, F-CA-CT-5	
Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - SoG Zwei konstante Wegpunkte. Geschwindigkeit 10 Knoten	
Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen ob der Controller in Richtung des eingegebenen Wegpunkts fährt (mit MATE Control und der EPD). (EPD-Route BT-0)	

Erwartetes Ergebnis:	Schiff fährt in die Richtung des im Controller gegebenen zweiten Wegpunktes mit einer maximalen Abweichung von 60 m (XTE, ablesbar in MatLAB).	
Eingetretenes Ergebnis:	Der in der EPD-Route „BT-0“ hinterlegte Zielwegpunkt wird erfolgreich abgefahren.	
Erfolg/Misserfolg:	Erfolg	

ID:	Basis-Test-1 (Oldenburg)	Bemerkungen:
Test Titel:	Abfahrt einer optimierten Route	- Wegpunkte werden erfolgreich an den Controller gesendet - Route wird wie erwartet abgefahren - Als nächster Wegpunkt wird jeweils der 7. Pfadwegpunkt gesendet
Beschreibung:	Nachdem eine Route durch das Route-Planning optimiert wurde, wird diese durch das Path-Planning mittels des Controllers abgefahren. Dabei werden statische Hindernisse berücksichtigt.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	

Eingehende-Daten:	- optimierte Route im RTZ-Format von der EPD (Basis-Test-1_optimized) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - Position vom Distribution System	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und Controller über das Distribution System TCS-03: Überprüfung auf Korrektheit der Daten, durch MATLAB Simulink Analysetools	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente sendet Daten (nächster Wegpunkt, RoutenStatus, wished ship speed) an den Controller über das Distribution System	

Eingetretenes Ergebnis:	Ergebnis entspricht dem erwarteten erwarteten Ergebnis. Die definierte Route wird abgefahren.	
Erfolg/Misserfolg:	Erfolg	
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	Basis-Test-2a (TC-7.1) (Bremen)	Bemerkungen:
Test Titel:	Virtual Handles	- Test wurde nicht durchgeführt, da er bereits in vorherigen Tests durchgeführt wurde.
Beschreibung:	Anwendung der Virtual Handles zur Steuerung der Zuse über ein webbasiertes Interface (Ruder und RPM).	
Funktionale Anforderungen:	F-EPD-9 F-EPD-10 F-EPD-11	
Eingehende-Daten:	Steuerkommandos (RPM und Ruder-Winkel) über die webbasierte Oberfläche der VirtualHandles (View in der EPD)	

Testschritte:	TCS-01: Virtual Handles Interface in der EPD öffnen TCS-02: Virtual Handles-Steuerung über die „Power“-Schaltfläche (oben rechts) einschalten. TCS-03: Durch die Nutzung der VirtualHandles Steuerbefehle die Steuerung des Schiffs übernehmen ○ Ruder-Winkel: maximal-Einschlag von 20° ○ RPM: maximal mögliche Drehzahl: 2000 RPM TCS-04: Virtual Handles-Steuerung über erneutes Klicken der „Power“-Schaltfläche (oben rechts) ausschalten.	
Erwartetes Ergebnis:	- VirtualHandles senden Steuerbefehle (zyklisch und bei Änderung der Einstellgrößen) ○ Ruder-Winkel: maximal-Einschlag von 20° Steuerbord ○ RPM: maximal mögliche Drehzahl: 2000 RPM - Schiff lässt sich per Virtual Handles steuern ○ Ruder-Winkel: das Schiff steuert nach rechts	

	○ RPM: das Schiff beschleunigt auf maximal 2000 RPM - Nach Abschalten der Virtual Handles wird die Steuerung wieder abgegeben. Der Controller deaktiviert dabei den manuellen Fahrmodus. ○ Der Ruder-Winkel verstellt sich in den Normalzustand ○ Die Geschwindigkeit wird auf 700 RPM gedrosselt	
Eingetretenes Ergebnis:	-	
Erfolg/Misserfolg:	nicht getestet.	
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	Basis-Test-2b (TC-7.2) (Bremen)	Bemerkungen:
Test Titel:	Manuelle Steuerung	- Test wurde nicht durchgeführt, da er bereits in vorherigen Tests durchgeführt wurde.
Beschreibung:	Der Controller erhält manuelle Steuerbefehle und gibt diese weiter an die Aktuatoren des physikalischen Testbeds	
Funktionale Anforderungen:	F-CA-CT-3, F-CA-CT-6	
Eingehende-Daten:	Steuerkommandos (RPM und Ruder-Winkel) an den UDP-Input des Controllers, beispielsweise erzeugt durch die Virtual Handles	
Testschritte:	TCS-01: Controller starten TCS-02: Manuelle Steuerbefehle erzeugen und an den Controller senden (Änderung des Ruderwinkels, Geschwindigkeit, RouteStatus: inaktiv) TCS-03: Überprüfung der weitergeleiteten Steuerbefehle durch Reaktion des Schiffes und MATLAB Simulink Analysetools.	

Erwartetes Ergebnis:	- Schiff lässt sich durch die Steuerbefehle Steuern:
	  o Ruder-Winkel: das Schiff steuert nach rechts oder links
	  o RPM: das Schiff beschleunigt oder bremst ab.
	- Nach Änderung des RouteStatus auf aktiv wird die Steuerung wieder abgegeben. Der Controller deaktiviert dabei den manuellen Fahrmodus.	

Eingetretenes Ergebnis:	-	
Erfolg/Misserfolg:	nicht getestet.	
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	Basis-Test-3a (TC-15.1) (Berlin)	Bemerkungen:
Test Titel:	Schiffserkennung mittels Kamera	- Konnte nicht getestet werden, da keine Kamera zur Verfügung stand
Beschreibung:	Die Object Detection Komponente muss kontinuierlich die Umgebung in Fahrtrichtung mittels einer Kamera erfassen und Schiffe erkennen. Für erkannte Schiffe, die unter dem Horizont liegen, wird die Position (rechts oder links vom eigenen Schiff) an die DOI als JSON-Objekt gesendet.	
Anforderung	F-OD-1 F-OD-2 F-OD-3 F-OD-4 F-OD-5	
Eingehende-Daten:	Echtzeit-Videoaufnahmen einer Webcam von der Umgebung vor dem Schiff	
Testschritte:	TCS-01: „Webcam Selection“ öffnen (durch starten der Komponente), die Webcam „HD Pro Webcam C920 0“ auswählen und auf „Select Camera“ klicken TCS-02: Überprüfen der gesendeten Schiffe an die DOI mit MATE Control.	

	TCS-03: (nach der Testfahrt) 200 aufgenommenen Bilder mit Boxen um erkannte Schiffe auswerten.	
Erwartetes Ergebnis:	● Schiffe, die sich in Fahrtrichtung befinden, werden als Schiff markiert: Die nachträgliche Auswertung der aufgenommen Bildern ergibt, dass 50% der auf den 200 Bildern zu sehenden Schiffe von der Object Detection erkannt wurde. ● Die Position (RIGHT, LEFT, BOTH oder NOTHING) der mittels Bildverarbeitung erkannten Schiffe, die sich unter dem Horizont befinden, werden als JSON-Objekt an die DOI gesendet.	
Eingetretenes Ergebnis:	Konnte nicht getestet werden.	
Erfolg/Misserfolg:	Misserfolg; Labortest erfolgreich	
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	Basis-Test-3b (TC-15.2) (Berlin)	
Test Titel:	Erkannte Schiffe der Object Detection an PP und EPD senden.	
Beschreibung:	Verarbeitung der von der Object Detection Komponente empfangenen JSON-Objekte über die Position der erfassten Schiffe mit der Kamera. Es wird geprüft, ob die erkannten Schiffe bereits mit AIS und/oder Radar erkannt wurden. Ist dies nicht der Fall wird ein „Kameraschiff“ in einem 45° Winkel zur eigenen Fahrrichtung mit einer Geschwindigkeit von 0 Knoten erstellt. Dieses Schiff wird an die EPD und an das Path Planning gesendet.	
Anforderung	F-DOI-9	
Eingehende-Daten:	JSON-Objekte von der Object Detection über die Position (RIGHT, LEFT, BOTH oder NOTHING) der mittels Bildverarbeitung erkannten Schiffe, die sich unter dem Horizont befinden.	

Testschritte:	TCS-01: „Webcam Selection“ öffnen (durch starten der Komponente), die Webcam „HD Pro Webcam C920 0“ auswählen und auf „Select Camera“ klicken TCS-03: In der Konfiguration (configuration.properties) der DOI einstellen, dass nur simulierte Schiffe verarbeitet werden sollen (simulatedOrRealData=simulated) TCS-03: Zeige die fusionierten Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	● Die von der Object Detection erkannten Schiffe werden nicht mit Radar oder AIS erkannt, da keine AIS- und Radardaten verarbeitet werden. ● Wenn sich ein Schiff vor dem eigenen Schiff befinden, ist dieses in dem EPD-DOI-Layer mit der Bezeichnung „Camera“ zu sehen.	
Eingetretenes Ergebnis:	-	
Erfolg/Misserfolg:	Konnte nicht getestet werden; Labortests erfolgreich.	
Testmethode:	Szenariobasiertes Blackbox-Testing	

Testszenarien

ID: TS-1 (Europa)	Kreuzendes Schiff von links (COLREG 15, 16, 17)	Bemerkungen: - erster Testversuch gescheitert, da der Controller nicht die Fahrt nicht mit der erforderlichen Geschwindigkeit regelt - nachdem der RPI und der Controller neugestartet wurden, erfolgte die Regelung erfolgreich - Soll-Pfad wird für ca. 10 m Fahrweg verlassen (gelber Bereich)
Beschreibung:	Das Schiff soll das simulierte Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. COLREG 17: a) Ein Schiff verhält sich nicht COLREG-Konform b) Falls das gegenüberliegende Schiff, das Ausweichen muss, nicht reagiert, muss das eigene Schiff trotzdem entsprechend reagieren und ausweichen d) Diese Regel befreit Schiffe nicht von ihrer Pflicht, sich von solchen Situationen fern zu halten	
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Startpunkt für das Szenario in Haggis muss erreicht werden ☒ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (10 Knoten) ☒ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	

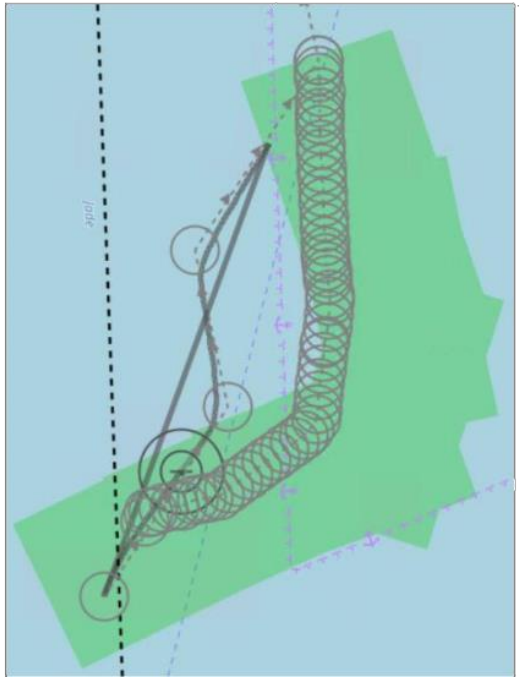
Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen	- TC-1.1 Erfolg
	TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning	- TC-1.2 Erfolg
	TC-2.1-Route-Planning Route empfangen	- TC-2.1 Erfolg
	TC-2.2-Route-Planning Route optimieren	- TC-2.2 Erfolg
	TC-2.2- EPD-seitiger Empfang einer vom Path-Planning optimierten Route	- TC-2.2 Erfolg
	TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route	- TC-2.3 Erfolg
	TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning	- TC-2.4 Erfolg
	TC-4-Controller Geschwindigkeit	- TC-4 Erfolg (im zweiten Versuch, siehe Bemerkungen)
	TC-5.1-Controller Wegpunkt vom Path-Planning empfangen	- TC-5.1 Erfolg
	TC-5.2-Controller Wegpunkt abfahren	- TC-5.2 Erfolg
	TC-6.1-AIS-Daten anzeigen	- TC-6.1 Erfolg
	TC-6.2-Radar-Daten anzeigen	- TC-6.2 Erfolg
	TC-6.3-Grid-Daten anzeigen	- TC-6.3 Erfolg
	TC-6.4-Polygone anzeigen	- TC-6.4 Erfolg
	TC-6.5-Schiffsposition anzeigen	- TC-6.5 Erfolg
	TC-9.2-Objekterkennung (links neben dem eigenen Schiff)	- TC-9.2 Erfolg
	TC-10-Datenfusion	- TC-10 Erfolg
	TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD)	- TC-11.1 Erfolg
	TC-11.2-Path Planning Kommunikation mit dem NoGoSolver	- TC-11.2 Erfolg

	TC-11.3-Path Planning Kommunikation mit der DOI	- TC-11.3 Erfolg
	TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller)	- TC-11.4 Erfolg
	TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD)	- TC-11.5 Erfolg
	TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	- TC-13 Erfolg
	TC-14.4.2-Path-Planning Kreuzen von links (Ausweichen)	- TC-14.4.2 Erfolg

ID: TS-2 (Asien)	Kreuzendes Schiff von rechts (wir müssen ausweichen) (COLREG 15, 16, 17)	Bemerkungen: - im ersten Versuch kam es zu einer Kollision, da nicht früh genug ausgewichen wurde. Möglicherweise wurde das Szenario nicht passend genug gestartet, sodass das kreuzende Schiff bereits den erwarteten Kollisionspunkt verlassen hatte und demnach kein ausreichender Ausweichpfad abgefahren wurde. - zweiter Versuch war erfolgreich. Geschwindigkeit und Pfadverfolgung funktioniert erfolgreich. - Ausweichmanöver entspricht dem erwarteten Ergebnis
Beschreibung:	Das Schiff soll das Fremdschiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. COLREG 15: Falls zwei Schiffe sich kreuzen muss das Schiff, welches sein Gegenüber auf der Steuerbord-Seite hat reagieren und ausweichen	
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Startpunkt für das Szenario in Haggis muss erreicht werden ☒ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (10 Knoten) ☒ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	

Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen	- TC-1.1 Erfolg
	TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning	- TC-1.2 Erfolg
	TC-2.1-Route-Planning Route empfangen	- TC-2.1 Erfolg
	TC-2.2-Route-Planning Route optimieren	- TC-2.2 Erfolg
	TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route	- TC-2.3 Erfolg
	TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning	- TC-2.4 Erfolg
	TC-4-Controller Geschwindigkeit	- TC-4 Erfolg
	TC-5.1-Controller Wegpunkt vom Path-Planning empfangen	- TC-5.1 Erfolg
	TC-5.2-Controller Wegpunkt abfahren	- TC-5.2 Erfolg
	TC-6.1-AIS-Daten anzeigen	- TC-6.1 Erfolg
	TC-6.2-Radar-Daten anzeigen	- TC-6.2 Erfolg
	TC-6.3-Grid-Daten anzeigen	- TC-6.3 Erfolg
	TC-6.4-Polygone anzeigen	- TC-6.4 Erfolg
	TC-6.5-Schiffsposition anzeigen	- TC-6.5 Erfolg
	TC-9.3-Objekterkennung (rechts neben dem eigenen Schiff)	- TC-9.3 Erfolg
	TC-10-Datenfusion	- TC-10 Erfolg
	TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD)	- TC-11.1 Erfolg
	TC-11.2-Path Planning Kommunikation mit dem NoGoSolver	- TC-11.2 Erfolg

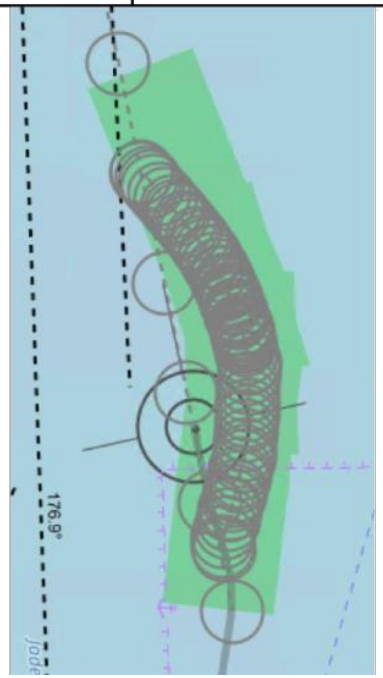
	TC-11.3-Path Planning Kommunikation mit der DOI	- TC-11.3 Erfolg
	TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller)	- TC-11.4 Erfolg
	TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD)	- TC-11.5 Erfolg
	TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	- TC-13 Erfolg
	TC-14.3-Path-Planning Kreuzen von rechts	- TC-14.3 Erfolg (im zweiten Versuch)



ID: TS-3 (America)	Entgegenkommendes Schiff (COLREG 14, 16, 17)	Bemerkungen:
Beschreibung:	Das Schiff soll das entgegenkommende Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. Danach übernimmt der Kapitän kurzzeitig die Kontrolle mithilfe der Virtual Handles. COLREG 14: a) Beide Schiffe weichen auf der Steuerbordseite aus b) Diese Situation gilt falls vermutet wird, dass ein gegenüberliegendes Schiff auf das Eigene zufährt oder sich annähert c) Falls die Vermutung besteht, dass diese Situation gilt, muss ausgewichen werden	- beim Anfahren des ersten Wegpunktes des Szenarios erfasst eine Welle das autonome Schiff und zerstörte die Bordelektronik. - Ergebnis: Systemausfall und Testabbruch
Vorbedingungen:	☐ Kommunikation zwischen den Komponenten muss gegeben sein ☐ Startpunkt für das Szenario in Haggis muss erreicht werden ☐ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (10 Knoten) ☐ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	

Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning TC-2.1-Route-Planning Route empfangen TC-2.2-Route-Planning Route optimieren TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-10-Datenfusion TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD)	

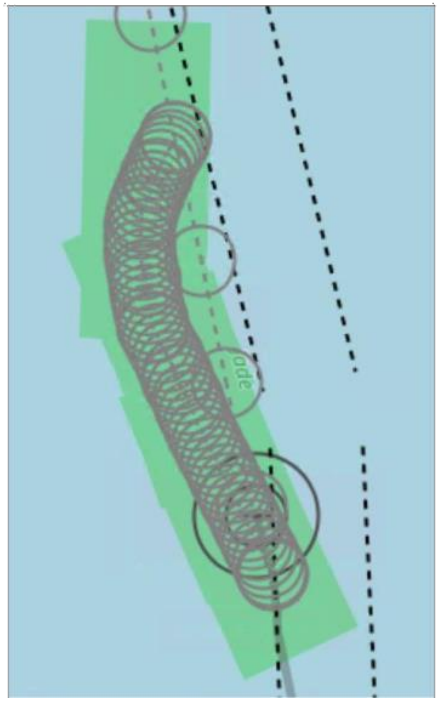
	TC-11.2-Path Planning Kommunikation mit dem NoGoSolver TC-11.3-Path Planning Kommunikation mit der DOI TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller) TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD) TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.2-Path-Planning entgegenkommend	



ID: TS-6 (Australien)	Überholmanöver Schiff (COLREG 13)	Bemerkungen:
Beschreibung:	Das Schiff soll im Autopiloten das Schiff erkennen, überholen und auf die ursprünglich geplante Route zurückkehren. COLREG 13: a) Das Schiff, das überholt wird, muss von anderen Schiffen gemieden werden b) Ein Schiff gilt als überholt, sobald das überholte Schiff deren Seitenlichter sieht c) Falls es irgendwelche bedenken gibt, darf nicht überholt werden	- konnte nicht mehr getestet werden, da die Systeme aufgrund des Defekts nicht mehr zur Verfügung standen.
Vorbedingungen:	☐ Kommunikation zwischen den Komponenten muss gegeben sein ☐ Startpunkt für das Szenario in Haggis muss erreicht werden ☐ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (10 Knoten) ☐ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	

Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning TC-2.1-Route-Planning Route empfangen TC-2.2-Route-Planning Route optimieren TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-10-Datenfusion TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD) TC-11.2-Path Planning Kommunikation mit dem NoGoSolver	

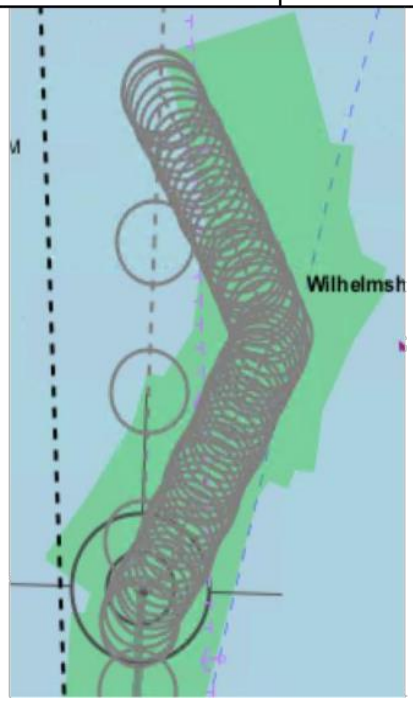
	TC-11.3-Path Planning Kommunikation mit der DOI TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller) TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD) TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.1-Path-Planning Überholen	



ID: TS-5 (Brocken)	Ausweichen eines dynamischen und statischen Hindernisses	Bemerkungen:
Beschreibung:	Das Schiff soll das simulierte Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. Dabei soll eine Kollision verhindert werden. Zusätzlich soll ein im Ausweichpfad liegendes statisches Hindernis bei der Ausweichplanung berücksichtigt werden. a) Ein Schiff befindet sich auf Kollisionskurs b) Falls das gegenüberliegende Schiff, das Ausweichen muss, nicht reagiert, muss das eigene Schiff trotzdem entsprechend reagieren und ausweichen d) Diese Regel befreit Schiffe nicht von ihrer Pflicht, sich von solchen Situationen fern zu halten e) statische Hindernisse in der Ausweichpfadplanung müssen ebenfalls berücksichtigt werden, um eine Kollision zu vermeiden.	- konnte nicht mehr getestet werden, da die Systeme aufgrund des Defekts nicht mehr zur Verfügung standen.
Vorbedingungen:	☐ Kommunikation zwischen den Komponenten muss gegeben sein ☐ Startpunkt für das Szenario in Haggis muss erreicht werden ☐ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (10 Knoten) ☐ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in	

	das Szenario starten zu können (fliegender Start)	
Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning TC-2.1-Route-Planning Route empfangen TC-2.2-Route-Planning Route optimieren TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-8-Objekterkennung (vor und links neben dem eigenen Schiff) TC-10-Datenfusion TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD)	

	TC-11.2-Path Planning Kommunikation mit dem NoGoSolver TC-11.3-Path Planning Kommunikation mit der DOI TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller) TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD) TC-12-Path-Planning Berücksichtigung statischer Hindernisse TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes TC-14.2-Path-Planning entgegenkommend	



Testfälle:

ID:	TC-1.1	Bemerkungen:
Test Titel:	Route mit Schiffsparametern erstellen	
Beschreibung:	Zusammenstellung einer Route in der EPD mit Schiffsparametern.	
Funktionale Anforderungen:	F-EPD-2 F-EPD-5	
Eingehende-Daten:		
Testschritte:	TCS-01: Wegpunktwerkzeug in der EPD auswählen und eine Route, bestehend aus zwei Wegpunkten, auf der Karte erstellen Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 Optional: TCS-02a: Route im Routenmanager der EPD per Doppelklick öffnen und ggfs. Geschwindigkeiten (max) der einzelnen Wegpunkte setzen (auf 10 Knoten)	

	TCS-02: Schiffsparameter in der “PG Mate”-Einstellungsseite der EPD hinterlegen und speichern Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Erwartetes Ergebnis:	Verbindet die gesetzten Wegpunkte zu einer Route, bestehend auf WP 1 und 2 mit einer maximalen WP-Leg-Geschwindigkeit von 10 Knoten und speichert die folgenden Schiffsparameter. Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 Schiffsparameter: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-1.2	Bemerkungen:
Test Titel:	Senden einer Route und Schiffsparameter an das Route-Planning	
Beschreibung:	Die zusammengestellte RTZ-Route, bestehend aus: Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 10 Knoten) wird mit den hinterlegten Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5 der Zuse per HTTP-Request an das Route-Planning gesendet.	
Funktionale Anforderungen:	F-EPD-2 F-EPD-5	
Eingehende-Daten:		

Testschritte:	TCS-01: Route im Route-Manager der EPD auswählen und über die Schaltfläche „Route optimieren“ an das Route-Planning senden.	
Erwartetes Ergebnis:	Die ausgewählte RTZ-Route und die hinterlegten Schiffsparameter wird mit den hinterlegten Geschwindigkeiten der einzelnen Wegpunkte an das Route-Planning gesendet: Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 10 Knoten) wird mit den hinterlegten Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.1	Bemerkungen:
Test Titel:	Route-Planning Route empfangen	
Beschreibung:	Eine Route, bestehend aus den beiden Wegpunkten: Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 10 Knoten) soll zusammen mit den hinterlegten Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5 vom Route-Planning empfangen werden.	
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5	
Eingehende-Daten:	in Route-Planning eingehend:	

	- RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit (10 Knoten) zwischen den Wegpunkten: ○ Wegpunkt 1: ○ Lat: 53.51765489239272 ○ Lon: 8.17657470703125 ○ Wegpunkt 2: ○ Lat: 53.667163323271666 ○ Lon: 8.124010935557715 - Schiffsinformationen ○ Länge: 8.0 ○ Breite: 2.5 ○ Höhe 2.5 ○ Tiefgang: 0.5	
Testschritte:	TCS-01: nicht-optimierte Route von der EPD an das Route Planning senden: ○ Wegpunkt 1: ○ Lat: 53.51765489239272 ○ Lon: 8.17657470703125 ○ Wegpunkt 2: ○ Lat: 53.667163323271666 ○ Lon: 8.124010935557715	

Erwartetes Ergebnis:	Das Route-Planning erhält eine Route von der EPD und die Schiffsinformationen Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 10 Knoten) Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.2	Bemerkungen:
Test Titel:	Route-Planning Route optimieren	
Beschreibung:	Optimieren einer ausgewählten Route	
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5	
Eingehende-Daten:	in Route-Planning eingehend: - RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit (10 Knoten) zwischen den Wegpunkten ○ Wegpunkt 1: ○ Lat: 53.51765489239272 ○ Lon: 8.17657470703125 ○ Wegpunkt 2: ○ Lat: 53.667163323271666 ○ Lon: 8.124010935557715	
Testschritte:	TCS-01: eingehende Route optimieren	
Erwartetes Ergebnis:	Das Route-Planning optimiert eine ausgewählte Route, diese wird via HTTP zurück an die EPD übertragen. Diese berücksichtigt in ihrer Gestalt statische Hindernisse, geliefert vom NoGoSolver. Enthaltene Wegpunkte: Siehe:	

	Anlage 1: optimierte RTZ-Route	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.3	Bemerkungen:
Test Titel:	EPD-seitiger Empfang einer vom Route-Planning optimierten Route	
Beschreibung:	Empfang einer optimierten Route des Route-Planning auf Basis einer ausgewählten Route	
Funktionale Anforderungen:	F-EPD-6 F-EPD-7	
Eingehende-Daten:	1. in EPD eingehend: - RTZ-Route vom Route-Planning eingehend via HTTP in die EPD, bestehend aus berechneten Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit von 10 Knoten zwischen den Wegpunkten: Siehe: Anlage 1: optimierte RTZ-Route	

Testschritte:	TCS-01: Nach Erhalt der optimierten Route (Siehe: Anlage 1: optimierte RTZ-Route), überprüfen, ob diese mit einem „optimized“ im Routenname im Routenmanager der EPD abgespeichert wird und sichtbar in der Karte dargestellt wird.	
Erwartetes Ergebnis:	Die EPD erhält eine vom Route-Planning optimierte Route via HTTP, die die gleichen Start- und Zielpunkte wie die Originalroute hat und Hindernisse, visualisiert durch “NoGo-Areas”, meidet. Die Geschwindigkeiten der Wegpunktverbindungen sind auf 10 Knoten gesetzt. Enthaltene Wegpunkte sind hier aufgelistet: Anlage 1: optimierte RTZ-Route	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.4	Bemerkungen:
Test Titel:	Senden einer optimierten Route von der EPD zum Path-Planning	
Beschreibung:	Eine in der EPD gespeicherte optimierte Route wird zum Start der Fahrt an das Path-Planning gesendet. Die optimierte Route (Siehe: Anlage 1: optimierte RTZ-Route) enthält dabei Abfahrgeschwindigkeiten von 10 Knoten.	
Funktionale Anforderungen:	F-EPD-6 F-EPD-7	
Eingehende-Daten:	in Path-Planning eingehend - RTZ-Route von der EPD via HTTP, beinhaltet alle zuvor vom Route-Planning berechneten Wegpunkte inklusive der Geschwindigkeiten zwischen den Wegpunkten Siehe: Anlage 1: optimierte RTZ-Route - Konfigurierte Schiffsinformationen aus der EPD als JSON-Objekte via HTTP	

Testschritte:	TCS-01: Schiffsinformationen in der Einstellungsmaske für Schiffsinformationen in der EPD setzen. TCS-02: Sicher gehen, dass Route ausgewählt (Anlage 1: optimierte RTZ-Route) ist und in der EPD „Route senden“ klicken. TCS-03: Prüfen, ob Route im Path-Planning eingeht.	
Erwartetes Ergebnis:	Das Path-Planning erhält eine vom Route-Planning optimierte Route (Anlage 1: optimierte RTZ-Route) als RTZ-Route via HTTP, die die gleichen Start- und Zielpunkte wie die Originalroute hat und statische Hindernisse(“NoGo-Areas”)meidet.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-3	Bemerkungen:
Test Titel:	Ungültige Route optimieren	
Beschreibung:	Zusammenstellung einer Route die einen Wegpunkt auf Land bzw. einer NoGoArea besitzt. Anschließend soll der Versuch gestartet werden, diese zu optimieren.	
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5	
Eingehende-Daten:	1. in Route-Planning eingehend: - RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten ○ WP1: ○ Lat: 53.56396732809957 ○ Lon: 8.124732971191406 ○ WP2: ○ Lat: 53.5792576599901 ○ Lon: 8.178634643554688 2. in EPD eingehend (im Anschluss an 1): - Fehlermeldung als JSON-Objekt via HTTP, welches in	

	der EPD das Darstellen einer Fehlermeldung auslöst	
Testschritte:	TCS-01: Wegpunktwerkzeug auswählen und auf der Karte eine Route zusammenstellen, wobei ein Wegpunkt auf Land zu setzen ist WP1: Lat: 53.56396732809957 Lon: 8.124732971191406 WP2: Lat: 53.5792576599901 Lon: 8.178634643554688 TCS-02: Sichergehen, dass Route ausgewählt ist und „Route optimieren“ auswählen TCS-03: Fehlermeldung mit “OK” bestätigen	
Erwartetes Ergebnis:	Rückmeldung als Hinweisfenster der EPD: “Es wurde keine Route gefunden!”	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-4	Bemerkungen:
Test Titel:	Controller Geschwindigkeit	
Beschreibung:	Eine in der EPD vorgegebene, optimierte Route, bestehend aus 2 Wegpunkten, wird an das Path-Planning gesendet. Die Wegpunktgeschwindigkeit wird dabei jeweils auf 10 Knoten gesetzt.	
Funktionale Anforderungen:	F-CA-CT-1, F-CA-CT-2, F-CA-CT-4	
Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - Current Speed Die nächsten 2 Wegpunkte vom Path-Planning. Die gewünschte Wegpunktgeschwindigkeit.	
Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: In der EPD eine Route aus 2 Wegpunkten erstellen, die nicht durch statische Hindernisse führen.	

	TCS-03: Geschwindigkeit der Wegpunkte auf 10 Knoten einstellen. TCS-04: Die Route ans Path-Planning schicken. TCS-05: Überprüfen der Geschwindigkeit des Schiffes mit MATE Control.	
Erwartetes Ergebnis:	Das Schiff fährt mit der vorgegebenen Geschwindigkeit (zwischen 9 und 11, optimal 10.00 Knoten).	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-5.1	Bemerkungen:
Test Titel:	Controller Wegpunktpaar vom Path-Planning empfangen	
Beschreibung:	Der Controller empfängt vom Path-Planning ein Wegpunktpaar im NMEA2000 Standard. Dabei wird das Path-Planning mit einer beliebigen Route gestartet und die anliegenden Werte (Positionsdaten der Wegpunkte) im Controller mit den Logs im Path-Planning verglichen.	
Funktionale Anforderungen:	F-CA-PP-9, F-CA-CT-1, F-CA-CT-2, F-CA-CT-5	
Eingehende-Daten:	Die nächsten 2 Wegpunkte vom Path-Planning	
Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen, ob der Controller den gesendeten Wegpunkt vom Path-Planning empfängt (mit MATLAB Simulink Überwachungstools).	

Erwartetes Ergebnis:	Controller empfängt die nächsten anzufahrenden Wegpunkte von der Path-Planning-Komponente, die in der Path-Planning Komponente herausgesendet wurden.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-5.2	Bemerkungen:
Test Titel:	Controller Wegpunkt abfahren	
Beschreibung:	Das Schiff kann in Richtung eines im Controller hinzugefügten Wegpunkts fahren.	
Funktionale Anforderungen:	F-CA-CT-1, F-CA-CT-4, F-CA-CT-5	

Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - SoG Die nächsten 2 Wegpunkte. Geschwindigkeit 10 Knoten	
Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen ob der Controller in Richtung des eingegebenen Wegpunkts fährt (mit MATE Control und der EPD).	
Erwartetes Ergebnis:	Schiff fährt in die Richtung des im Controller gegebenen zweiten Wegpunktes mit einer maximalen Abweichung von 60 m (XTE, ablesbar in MatLAB).	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.1	Bemerkungen:
Test Titel:	Anzeige AIS-Daten	
Beschreibung:	Anzeige der AIS-Daten in der EPD.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.1	
Eingehende-Daten:	- Sensor-Daten (kontinuierlich) vom HUB über den NMEA-Sensor - Koordinaten erkannter Schiffe/Objekte (AIS, Radar und fusionierte Daten) als JSON via HTTP von der DOI	
Testschritte:	TCS-01: Empfang von AIS-Daten per UDP vom HUB und Darstellung auf der EPD-Karte TCS-02: Überprüfung von AIS-Roh-Daten von der DOI (Layer: DOIDataLayer)	
Erwartetes Ergebnis:	Anzeige der von der DOI übermittelten AIS-Daten im Layer „DOIDataLayer“ der EPD.	

Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.2	Bemerkungen:
Test Titel:	Anzeige Radar-Daten	
Beschreibung:	Anzeige der Radar-Daten in der EPD.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.2	
Eingehende-Daten:	- Koordinaten erkannter Schiffe/Objekte (AIS, Radar und fusionierte Daten) als JSON via HTTP von der DOI	
Testschritte:	TCS-01: Überprüfung von Radar-Roh-Daten von der DOI (Layer: DOIDataLayer)	

Erwartetes Ergebnis:	Anzeige der von der DOI übermittelten Radar-Daten im Layer „DOIDataLayer“ der EPD	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.3	Bemerkungen:
Test Titel:	Anzeige Grid-Daten	
Beschreibung:	Anzeige der Grid-Daten in der EPD. Diese werden als JSON vom Route-Planning nach Optimierung einer nicht-optimierten Route an die EPD übertragen.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.3	
Eingehende-Daten:	- Routendaten - NoGo-Daten und Griddaten als JSON via HTTP vom Route-Planning	

	(bereits vom Routenberechnen vorhanden)	
Testschritte:	TCS-01: Überprüfung der Grid-Daten anhand der EPD Karte (Layer: SOIDataLayer)	
Erwartetes Ergebnis:	Anzeige der vom Route-Planning übermittelten Grid-Daten im Layer „SOIDataLayer“ der EPD	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.4	Bemerkungen:
Test Titel:	Anzeige Polygone	
Beschreibung:	Anzeige der Polygone in der EPD. Diese werden als JSON vom Route-Planning nach Optimierung einer nicht-optimierten Route an die EPD übertragen.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.4	
Eingehende-Daten:	- Routendaten vom Route-Planning - NoGo-Daten und Griddaten als JSON via HTTP vom Route-Planning (bereits vom Routenberechnen vorhanden)	
Testschritte:	TCS-01: Überprüfung der Polygone anhand der EPD Karte (Layer: SOIDataLayer)	

Erwartetes Ergebnis:	Anzeige der vom Route-Planning übermittelten Polygone im Layer „SOIDataLayer“ der EPD	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.5	Bemerkungen:
Test Titel:	Anzeige der eigenen Schiffsposition	
Beschreibung:	Anzeige der eigenen Schiffsposition in der EPD.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.5	
Eingehende-Daten:	- Sensor-Daten (kontinuierlich) vom Polymorphic Interface (NMEA0183)	
Testschritte:	TCS-01: Überprüfung der eigenen Schiffsposition anhand der EPD Karte und MATE Control	

Erwartetes Ergebnis:	Anzeige der korrekten, eigenen Schiffsposition auf der EPD-Seekarte, mit korrektem Heading. Position aus MATE Control und der EPD stimmt überein.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-8	Bemerkungen:
Test Titel:	Objekterkennung (vor dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (vor dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen. Zusätzlich werden JSON-Objekte über http von erkannten Schiffe mittels Bilderkennung von der Object Detection empfangen	

Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.1 F-DOI-4 F-DOI-5	
Eingehende-Daten:	- Sensordaten des AIS in NMEA-2000 Format, mit erkanntem Schiff welches bis zu 3500m von der eigenen Position entfernt ist. - Sensordaten des Radars in NMEA-2000 konformen Format, mit erkanntem Schiff welches bis zu 3500m von der eigenen Position entfernt ist. - JSON-Objekt über die Position (rechts, links, beides) welches von der Object Detection erkannt wurde	
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layer an und prüfe, ob umliegenden Schiffe	

	von Radar und AIS angezeigt werden.	
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten von Radar werden von der DOI-Komponente empfangen - Erkannte Schiffe von der Object-Detection-Komponente werden empfangen	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-9.1	Bemerkungen:
Test Titel:	Objekterkennung (hinter dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (hinter dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen.	
Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.3 F-DOI-5	
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format, mit erkanntem Schiff, welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet - Sensordaten des Radars im NMEA-2000 konformen Format, mit erkanntem Schiff welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet	

Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe, welche von der Object-Detection-Komponente erkannt wurden, (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten vom Radar werden von der DOI-Komponente empfangen	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-9.2	Bemerkungen:
Test Titel:	Objekterkennung (links neben dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (links neben dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen.	
Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.2 F-DOI-5	
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format, mit erkanntem Schiff, welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet - Sensordaten des Radars im NMEA-2000 konformen Format, mit erkanntem Schiff welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet	

Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe, welche von der Object-Detection-Komponente erkannt wurden, (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten vom Radar werden von der DOI-Komponente empfangen	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-9.3	Bemerkungen:
Test Titel:	Objekterkennung (rechts neben dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (rechts neben dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen.	
Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.2 F-DOI-5	
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format, mit erkanntem Schiff, welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet - Sensordaten des Radars im NMEA-2000 konformen Format, mit erkanntem Schiff welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet	

Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe, welche von der Object-Detection-Komponente erkannt wurden, (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten vom Radar werden von der DOI-Komponente empfangen	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-10	Bemerkungen:
Test Titel:	Datenfusion	
Beschreibung:	Die DOI-Komponente muss empfangene AIS-Nachrichten (statisch und dynamisch) und empfangene Radar Tracks untereinander fusionieren (AIS mit AIS und AIS mit Radar) Die DOI-Komponente muss bei allen assoziierten Schiffen in der Umgebung (2 sm Umkreis ums eigene Schiff) die objektspezifischen Parameter Größe, Geschwindigkeit, und Position neu berechnen.	
Anforderung	F-DOI-6 F-DOI-7 F-DOI-8	
Eingehende-Daten:	- Sensordaten des Radars im NMEA-2000 konformen Format in einem Umkreis ums eigenen Schiff von bis zu 2 sm (mind. 15 Nachrichten) - Sensordaten des GPS in NMEA-2000 - Sensordaten des Headings in NMEA-2000 - JSON-Objekt über die Position (rechts, links) welche durch Bildverarbeitung erkannt wurde	

Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Fusioniere die Schiffe (automatisch) TCS-04: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an TCS-05: Zeige die fusionierten Schiffe über den EPD-DOI-Layers an	
	AIS-Nachrichten von Schiffen, die sich in einem 2 sm Umkreis vom eigenen Schiff befinden, werden bei gleicher User ID einander zugeordnet. Radartracks von Schiffen, die sich in einem 2 sm Umkreis vom eigenen Schiff befinden, werden bei gleicher User ID einander zugeordnet. AIS-Nachrichten und Radar-Tracks eines Schiffes im Umkreis von 2 sm werden einander zugeordnet, wenn es sich um dasselbe Schiff handelt. Bei der Simulation werden die AIS-Nachrichten mit der MMSI 123456789 und den Radarnachrichten mit der MMSI 60 ab 15 empfangenen Radarnachrichten und 4 AIS-Nachrichten durchgehend fusioniert. Im EPD-DOI-Layer ist ein Schiff mit der Bezeichnung FRadar60AIS123456789 zu sehen.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-11.1	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit der EPD (empfangen von Daten von der EPD)	
Beschreibung:	Eine globale Route, die bereits optimiert wurde enthält mehrere Wegpunkte. Diese Route wird an das Path-Planning über eine http-Verbindung übermittelt. Eine erfolgreiche Übertragung wird in der EPD visualisiert.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route	

	TCS-02: Die EPD visualisiert den Erfolg der Übertragung durch einen Grünen Haken. TCS-03: Die Path-Planning Komponente gibt nach kurzer Berechnungszeit einen Pfad an die EPD zurück.	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente empfängt eine valide, globale Route von der EPD über die http-Verbindung - Das erfolgreiche Übertragen wird in der EPD visualisiert.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.2	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit dem NoGoSolver	
Beschreibung:	Die Path-Planning-Komponente muss kontinuierlich Informationen über alle statischen Hindernisse vom NoGoSolver anfragen, entgegennehmen und decodieren können.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- JSON-Objekte vom NoGoSolver: WKT Geometrien, die statische Hindernisse repräsentieren. Diese umfassen den durch das Path-Planning spezifizierten Koordinatenbereich und der angegebenen Schiffsinformationen.	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route	

	TCS-02: Überprüfung der Verbindung zwischen Path-Planning und NoGoSolver durch Log-Ausgaben (optional: Visualisierung durch die EPD)	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente fragt statische Hindernisse beim NoGoSolver an und empfängt von diesem die statischen Hindernisse. Diese Daten stimmen mit vorhanden Kartendaten überein (Verifizierbar durch Expertenwissen, Boardinstrumente der Zuse).	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.3	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit der DOI	
Beschreibung:	Das Path-Planning muss kontinuierlich Informationen über alle dynamischen Hindernisse (Schiffe) von der DOI-Komponente entgegennehmen und decodieren können.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- JSON-Objekte von der DOI-Komponente, enthalten identifizierte Objekte (nur fusionierte Objekte: Positionen, COG, SOG, Länge und Breite der Schiffe)	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und DOI-Komponente durch log-Ausgaben und Vergleich mit den Boardinstrumenten der Zuse	

Erwartetes Ergebnis:	- Die Path-Planning-Komponente empfängt dynamische, zum Teil fusionierte Daten von der DOI-Komponente, welche alle Schiffe in einem 2 sm Radius um das eigene Schiff repräsentieren.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.4	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit dem Controller (senden von Daten an den Controller)	
Beschreibung:	Der berechnete valide Pfad wird über das Distribution System an den Controller geschickt.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- optimierte Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente	

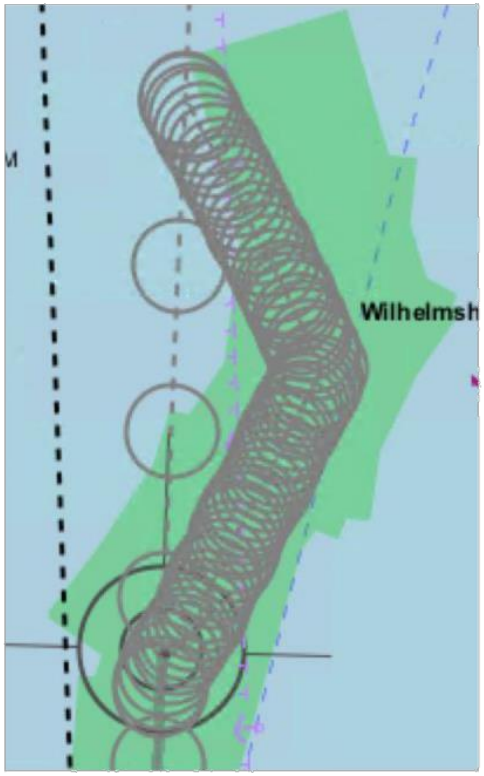
	- Position vom Distribution System	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und Controller über das Distribution System TCS-03: Überprüfung auf Korrektheit der Daten, durch MATLAB Simulink Analysetools	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente sendet Daten (nächster Wegpunkt, RoutenStatus, wished ship speed) an den Controller über das Distribution System	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.5	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit der EPD (senden von Daten an die EPD)	
Beschreibung:	Der berechnete valide Pfad wird an die EPD geschickt, dort gespeichert und angezeigt.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- Berechneter Pfad von Path-Planning (variable Anzahl) - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route	

	TCS-02: Überprüfung der Verbindung zwischen Path-Planning und EPD durch Log-Ausgaben	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente sendet den aktuellen Pfad an die EPD (optional: Debug-Ausgaben, wie Goodwin-Shapes / statische Hindernisse)	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-12	Bemerkungen:
Test Titel:	Berücksichtigung eines dynamischen und statischen Hindernisses bei der Pfadberechnung	
Beschreibung:	Ausweichen eines dynamischen Objektes (Schiff) unter Berücksichtigung eines nahen statischen Hindernisses (Untiefe).	
Anforderung	F-CA-PP-5	
Eingehende-Daten:	- Szenario-Route im RTZ-Format von der EPD (Szenario 6) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System	

	- Definierte Optimalroute (Sollpfad)	
Testschritte:	TCS-01: Optimierte Route für Szenario 6 in der EPD laden und zusammen mit den Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 durch Klick auf die Schaltfläche „Route starten“ an das Path-Planning senden. TCS-02: Den berechneten Pfad der Path-Planning überwachen TCS-03: In den Zuse-Tab in MATE-Control wechseln, dort die optimierte Route für Szenario 6 auswählen und Validierung durch Klick auf „Record“ starten (Polygone werden in EPD gezeichnet)	
Erwartetes Ergebnis:	Das System weicht dem dynamischen Objekt aus ohne dabei mit den statischen Hindernissen zu kollidieren	

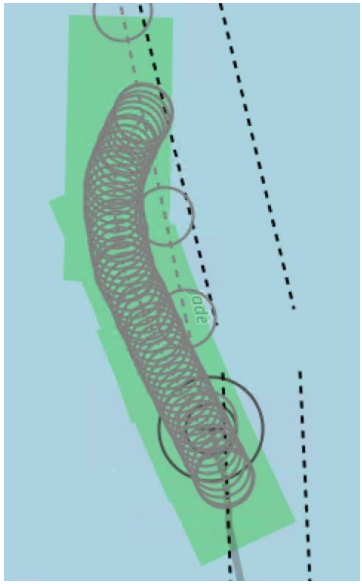
		
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-13	Bemerkungen:
Test Titel:	Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	
Beschreibung:	Ausweichen eines Objektes unter Berücksichtigung der physikalischen Eigenschaften für das im Test eingesetzte Schiff (ZUSE).	
Anforderung	F-CA-PP-6	
Eingehende-Daten:	- Szenario-Routen im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System	

Testschritte:	TCS-01: Optimierte Route für Basis-Test-1 in der EPD laden und zusammen mit den Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 durch Klick auf die Schaltfläche „Route starten“ an das Path-Planning senden. TCS-02: Den berechneten Pfad der Path-Planning überwachen (werden in EPD gezeichnet TCS-03: Statistische Auswertung der berechneten Pfade auf Einhaltung der Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes (Turning Angle darf nicht 20 Grad bei jedem Wegpunktübergang)	
Erwartetes Ergebnis:	- Der berechnete Pfad steht nicht im Konflikt mit den physikalischen Eigenschaften (Wendewinkel der Zuse) des im Test eingesetzten Schiffs, d.h. der Wendewinkel beträgt maximal 30 Grad). Verifizierbar, dadurch, dass der Controller dem berechneten	

	Pfad, bis auf einen Threshold, folgen kann.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

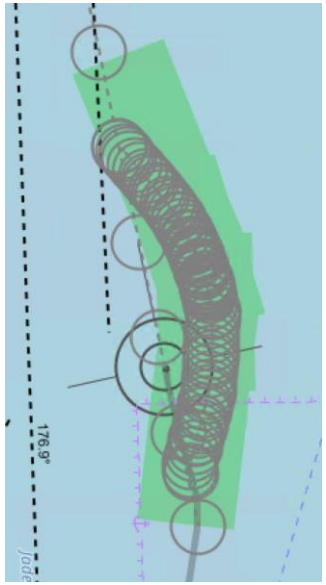
ID:	TC-14.1	Bemerkungen:
Test Titel:	Path-Planning Überholen	
Beschreibung:	Überholen eines dynamischen Objektes (Schiff) unter Berücksichtigung der COLREG Regel 13.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System - Definierte Optimalroute (Sollpfad)	

Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route (Szenario 4) an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen (Überholen, COLREG 13) TCS-03: Validierung durch Einhaltung des Threshold mit MateControl	
Erwartetes Ergebnis:	Die Zuse überholt das Fremdschiff auf der linken oder rechten Seite mit einem ausreichenden Mindestabstand von 100 Metern und folgt nach dem Überholmanöver wieder der zuvor definierten Route. 	

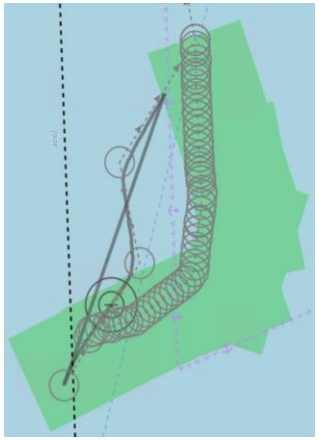
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-14.2	Bemerkungen:
Test Titel:	Path-Planning entgegenkommend	
Beschreibung:	Ausweichen eines entgegenkommenden dynamischen Objektes (Schiff) unter Berücksichtigung der COLREGs Regeln 14, 16, 17.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0	

	• Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System - Definierte Optimalroute (Sollpfad)	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route (Szenario 3) an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen TCS-03: Validierung durch Einhaltung des Threshold mit MateControl	
Erwartetes Ergebnis:	Die Zuse weicht dem entgegenkommenden Fremdschiff nach rechts hin mit einem ausreichenden Mindestabstand von 100 Metern aus und folgt nach dem	

	Ausweichmanöver wieder der zuvor definierten Route. Validierung durch Einhaltung des Threshold mit MateControl mit zuvor definierten Route. 	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

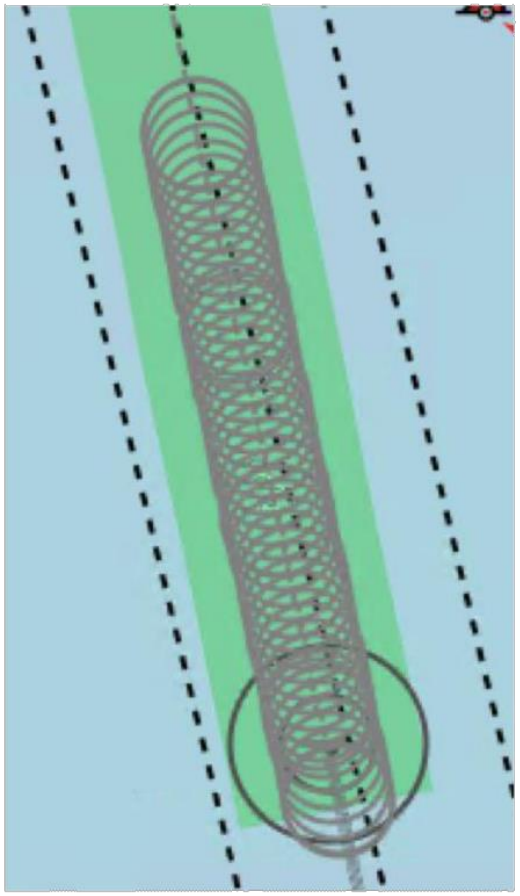
ID:	TC-14.3	Bemerkungen:
Test Titel:	Path-Planning Kreuzen von rechts	
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von rechts gekreuzt unter Berücksichtigung der COLREGs Regeln 15, 16, 17.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente	

	- Position, Heading (NMEA0183) vom Distribution System - Definierte Optimalroute (Sollpfad)	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen (Kreuzen, COLREGs 15, 16) TCS-03: Validierung durch Einhaltung des Threshold mit MateControl	
Erwartetes Ergebnis:	Die Zuse weicht dem Fremdschiff nach links hin mit einem ausreichenden Mindestabstand von 100 Metern aus und folgt nach dem Ausweichmanöver wieder der zuvor definierten Route. 	

Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-14.4.1	Bemerkungen:
Test Titel:	Path-Planning Kreuzen von links (Kurs beibehalten)	
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von links gekreuzt unter Berücksichtigung der COLREGs Regeln 15, 16, 17.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5	

	- JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System - Definierte Optimalroute (Sollpfad)	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route (Szenario 5) an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Überwachung des Kreuz-Manövers TCS-03: Validierung durch Einhaltung des Threshold mit MateControl	
Erwartetes Ergebnis:	Die Zuse verfolgt weiterhin den vorberechneten validen Pfad innerhalb der definierten Thresholds.	

		
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-14.4.2	Bemerkungen:
Test Titel:	Path-Planning Kreuzen von links (Ausweichen)	
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von links gekreuzt, das andere Schiff verletzt die COLREGs Regeln und weicht nicht aus, daraufhin muss das zu testende Schiff zwangsweise Ausweichen unter Berücksichtigung der COLREGs Regeln 15, 16, 17.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von	

	erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System - Definierte Optimalroute (Sollpfad)	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Überwachung des erzwungenen Ausweichens während des Kreuz-Manövers TCS-03: Validierung durch Einhaltung des Threshold mit MateControl	
Erwartetes Ergebnis:	Die Zuse weicht dem Fremdschiff nach links oder rechts hin mit einem ausreichenden Mindestabstand von 100 Metern aus und folgt nach dem Ausweichmanöver wieder der zuvor definierten Route.	

		
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

Fehlerinjektion:

ID:	TC-FI-1	Bemerkungen:
Test Titel:	ownPosition-Injection	
Beschreibung:	Ein anderes, bereits vorher über AIS identifiziertes Schiff wechselt die Position auf die Position des eigenen Schiffes.	
Eingehende-Daten:	- Route im RTZ-Format von der EPD (Szenario 1) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System	

Testschritte:	TCS-01: optimierte Route in der EPD auswählen (Szenario 1). TCS-02: Automatisierte Fahrt starten, in dem die optimierte Route an das Path-Planning gesendet wird. TCS-03: In den Error-Injection-Tab in MATE-Control wechseln, das simulierte Schiff auswählen, den Haken bei „ownPosition“ setzen und die Fehlerinjektion mit einem Klick auf „inject“ ausführen.	
Erwartetes Ergebnis:	Nachdem die Injektion durchgeführt wurde, berechnet das Path-Planning einen Pfad, der dazu führt, dass der Controller die Fahrt abbremst und die Fahrt gestoppt wird.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Fehlertest	

ID:	TC-FI-2	Bemerkungen:
Test Titel:	Ungültiger Ruderwinkel	
Beschreibung:	Der Controller empfängt einen außerhalb der spezifizierten Grenzen definierten Ruderwinkel.	
Eingehende-Daten:	- Ruderwinkel per NMEA 2000 (an Controller)	
Testschritte:	TCS-01: optimierte Route in der EPD auswählen (Szenario 1). TCS-02: Automatisierte Fahrt starten, in dem die optimierte Route an das Path-Planning gesendet wird. TCS-03: In den Error-Injection-Tab in MATE-Control wechseln, das simulierte Schiff auswählen, den Haken bei „rudder angle“ setzen und die Fehlerinjektion mit einem Klick auf „inject“ ausführen. TCS-04: Überwachung des Systems mit MATE-Control	
Erwartetes Ergebnis:	Das System verhält sich unverändert.	
Eingetretenes Ergebnis:		

Erfolg/Misserfolg:		
Testmethode:	Fehlertest	

ID:	TC-FI-3	Bemerkungen:
Test Titel:	Schiff verschwindet (Afrika-Injektion)	
Beschreibung:	Ein existierendes (real oder simuliert), über AIS erkanntes Schiff verschwindet von seiner bekannten Position.	
Eingehende-Daten:	- Route im RTZ-Format von der EPD (Szenario 1) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition)	

	- JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System TCS-04: Überwachung des Systems mit MATE-Control	
Testschritte:	TCS-01: optimierte Route in der EPD auswählen (Szenario 1). TCS-02: Automatisierte Fahrt starten, in dem die optimierte Route an das Path-Planning gesendet wird. TCS-03: In den Error-Injection-Tab in MATE-Control wechseln, das simulierte Schiff auswählen, den Haken bei „remove“ setzen und die Fehlerinjektion mit einem Klick auf „inject“ ausführen. TCS-04: Überwachung des Systems mit MATE-Control	
Erwartetes Ergebnis:	Das über AIS erkannte Schiff wird noch weitere ca. 30 Sekunden im Speicher gehalten und in dem Zeitraum auch an die EPD und an das PP gesendet. Nach ca. 30 Sekunden wird das Schiff in der DOI gelöscht und nicht mehr versendet.	

	In dem EPD-DOI-Layer ist die Repräsentation des Schiffes mit der Bezeichnung FAIS123456789 noch ca. 30 Sekunden nach Erhalt der letzten Nachricht dieses Schiffes zu sehen.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Fehlertest	

ID:	TC-FI-4	Bemerkungen:
Test Titel:	Schiff taucht auf (Afrika-Injektion)	
Beschreibung:	Ein zuvor verschwundenes (simuliert), über AIS erkanntes Schiff erscheint wieder auf seiner echten Position.	

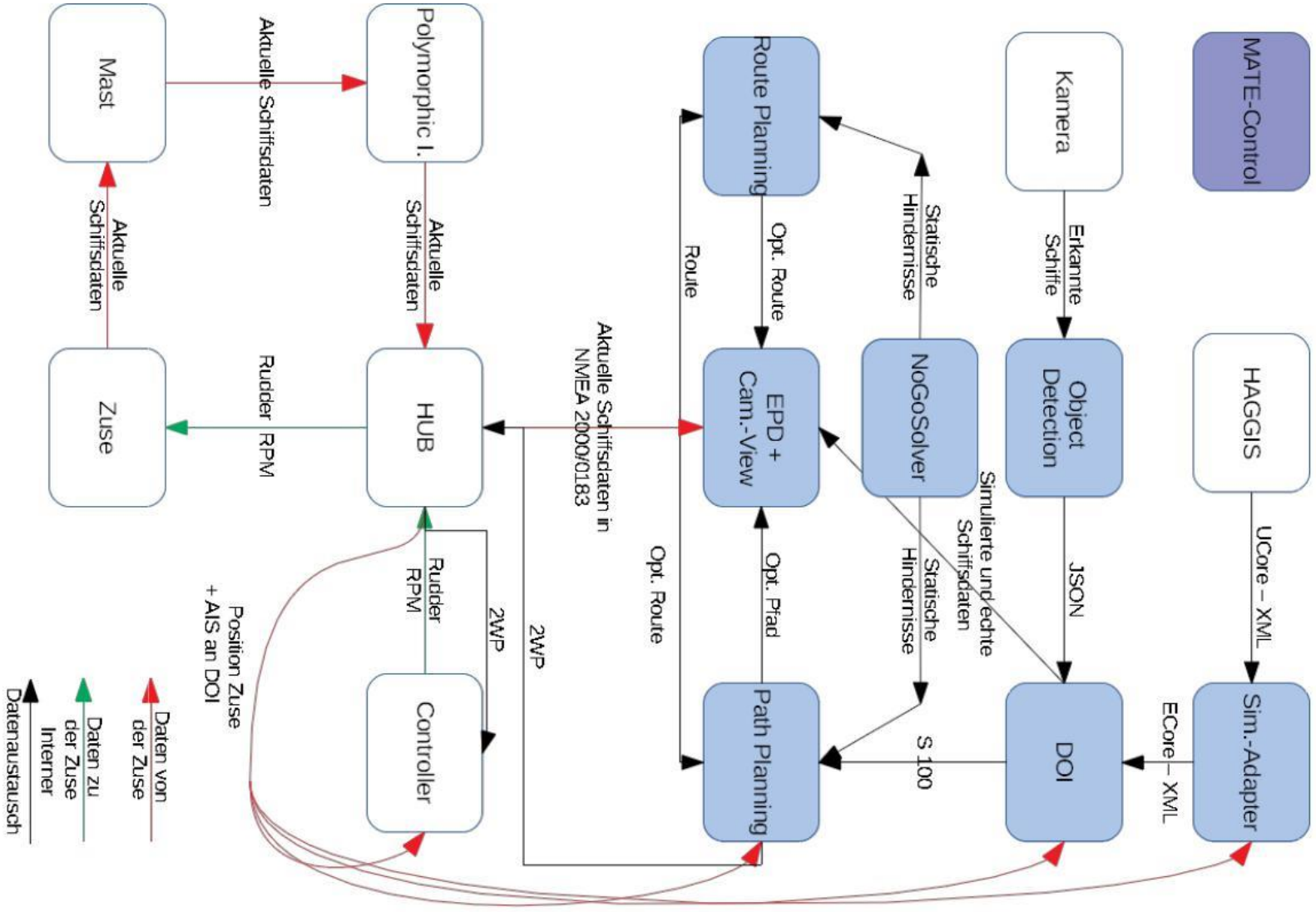
Eingehende-Daten:	- Route im RTZ-Format von der EPD (Szenario 1) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System TCS-04: Überwachung des Systems mit MATE-Control	
Testschritte:	TCS-01: optimierte Route in der EPD auswählen (Szenario 1). TCS-02: Automatisierte Fahrt starten, in dem die optimierte Route an das Path-Planning gesendet wird. TCS-03: In den Error-Injection-Tab in MATE-Control wechseln, das simulierte Schiff auswählen, den Haken bei „remove“ entfernen und die	

	Fehlerinjektion mit einem Klick auf „inject“ ausführen. TCS-04: Überwachung des Systems mit MATE-Control	
Erwartetes Ergebnis:	Das erkannte Schiff wird wieder erfasst und an die EPD und an das PP gesendet. In dem EPD-DOI-Layer ist das Schiff mit der Bezeichnung FAIS123456789 zu sehen.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Fehlertest	

ID:	TC-FI-05	Bemerkungen:
Test Titel:	Wechsel der MMSI	
Beschreibung:	Ein mit AIS und oder Radar identifiziertes Schiff in der Umgebung des eigenen Schiffs wechselt seine MMSI.	
Eingehende-Daten:	- Route im RTZ-Format von der EPD (Szenario 1) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System TCS-04: Überwachung des Systems mit MATE-Control	

Testschritte:	TCS-01: optimierte Route in der EPD auswählen (Szenario 1). TCS-02: Automatisierte Fahrt starten, in dem die optimierte Route an das Path-Planning gesendet wird. TCS-03: In den Error-Injection-Tab in MATE-Control wechseln, das simulierte Schiff auswählen, den Haken bei „MMSI“ setzen und die Fehlerinjektion mit einem Klick auf „inject“ ausführen. TCS-04: Überwachung des Systems mit MATE-Control	
Erwartetes Ergebnis:	Für jede neue MMSI wird ein Schiff in der DOI gespeichert. Diese Schiffe werden an die EPD und an das PP gesendet. In dem EPD-DOI-Layer werden somit mehrere Repräsentationen des Schiffes angezeigt.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Fehlertest	

Gesamtübersicht:



Anlage 1: optimierte RTZ-Route

WP1:
lat: 53.51765489239272
lon: 8.17657470703125
WP2:
lat: 53.51941397091744
lon: 8.175958462627543
WP3:
lat: 53.52117304574505
lon: 8.175342167169969
WP4:
lat: 53.52293211687508
lon: 8.174725820651123
WP5:
lat: 53.524691184307095
lon: 8.174109423063586
WP6:
lat: 53.526817548437876
lon: 8.176445103099207
WP7:
lat: 53.52857662358174
lon: 8.175828720333746
WP8:
lat: 53.53033569502767
lon: 8.175212286487849
WP9:
lat: 53.53246203559427
lon: 8.177548332323932
WP10:
lat: 53.53422111475336
lon: 8.176931913315126
WP11:
lat: 53.53634742033682
lon: 8.17926825913435
WP12:
lat: 53.53810650720982
lon: 8.178651854982
WP13:
lat: 53.5398655903855
lon: 8.17803539973313
WP14:
lat: 53.541624669863445
lon: 8.177418893380318
WP15:
lat: 53.543750963280054
lon: 8.17975567121135
WP16:
lat: 53.54551005047374
lon: 8.1791391797257
WP17:
lat: 53.547269133969756
lon: 8.178522637124349
WP18:
lat: 53.54902821376765
lon: 8.177906043399876
WP19:
lat: 53.55078728986697
lon: 8.177289398544849
WP20:
lat: 53.552546362267265
lon: 8.17667270255184
WP21:
lat: 53.55430543096807
lon: 8.176055955413423
WP22:
lat: 53.55606449596894
lon: 8.175439157122161
WP23:
lat: 53.55782355726942
lon: 8.174822307670626
WP24:
lat: 53.55958261486906
lon: 8.17420540705138
WP25:
lat: 53.561341668767376
lon: 8.173588455256988
WP26:
lat: 53.563100718963945
lon: 8.172971452280017
WP27:
lat: 53.5648597654583
lon: 8.172354398113024
WP28:
lat: 53.566618808249984
lon: 8.17173729274857
WP29:
lat: 53.56837784733853
lon: 8.171120136179217
WP30:
lat: 53.57013688272353
lon: 8.17050292839752
WP31:
lat: 53.57189591440448
lon: 8.169885669396034
WP32:
lat: 53.573654942380934
lon: 8.169268359167317
WP33:
lat: 53.57541396665244
lon: 8.168650997703919
WP34:
lat: 53.57717298721855
lon: 8.168033584998396
WP35:
lat: 53.57893200407879
lon: 8.167416121043292
WP36:
lat: 53.58069101723272
lon: 8.166798605831163
WP37:
lat: 53.582450026679886
lon: 8.166181039354553
WP38:
lat: 53.584209032419814
lon: 8.16556342160601
WP39:
lat: 53.58596803445207
lon: 8.16494575257808
WP40:
lat: 53.58772703277619
lon: 8.164328032263299
WP41:
lat: 53.58948602739171
lon: 8.16371026065422
WP42:
lat: 53.59124501829817
lon: 8.163092437743378
WP43:
lat: 53.59300400549512
lon: 8.162474563523311
WP44:
lat: 53.59476298898212
lon: 8.161856637986562
WP45:
lat: 53.59652196875869
lon: 8.161238661125662
WP46:
lat: 53.59828094482437
lon: 8.160620632933151
WP47:
lat: 53.60003991717871
lon: 8.160002553401561
WP48:
lat: 53.60179888582128
lon: 8.159384422523424
WP49:
lat: 53.603557850751585
lon: 8.158766240291273
WP50:
lat: 53.60531681196919
lon: 8.158148006697635
WP51:
lat: 53.60707576947363
lon: 8.157529721735038
WP52:
lat: 53.60883472326444
lon: 8.156911385396013
WP53:
lat: 53.61059367334118
lon: 8.156292997673082
WP54:
lat: 53.61235261970337
lon: 8.155674558558768
WP55:
lat: 53.61411156235057
lon: 8.155056068045596
WP56:
lat: 53.615870501282316
lon: 8.154437526126088
WP57:
lat: 53.61762943649817
lon: 8.153818932792761
WP58:
lat: 53.619388367997644
lon: 8.153200288038136
WP59:
lat: 53.621147295780275
lon: 8.152581591854732
WP60:
lat: 53.62290621984563
lon: 8.15196284423506
WP61:
lat: 53.62466514019325
lon: 8.151344045171633
WP62:
lat: 53.62642405682268
lon: 8.15072519465697
WP63:
lat: 53.62818296973344
lon: 8.15010629268358
WP64:
lat: 53.62994187892507
lon: 8.149487339243972
WP65:
lat: 53.631700784397125
lon: 8.148868334330656
WP66:
lat: 53.63345968614916
lon: 8.148249277936138
WP67:
lat: 53.635218584180 69
lon: 8.147630170052926
WP68:
lat: 53.636977478491275
lon: 8.147011010673522
WP69:
lat: 53.638736369080455
lon: 8.14639179979043
WP70:
lat: 53.640495255947755
lon: 8.145772537396152
WP71:
lat: 53.64225413909272
lon: 8.14515322348319
WP72:
lat: 53.64401301851491
lon: 8.144533858044039
WP73:
lat: 53.64577189421386
lon: 8.143914441071196
WP74:
lat: 53.647530766189085
lon: 8.143294972557163
WP75:
lat: 53.649289634440144
lon: 8.142675452494432
WP76:
lat: 53.651048498966574
lon: 8.142055880875498
WP77:
lat: 53.6524392023931
lon: 8.138475642429457
WP78:
lat: 53.65419804426581
lon: 8.137855849923817
WP79:
lat: 53.65595688241105
lon: 8.13723600582885
WP80:
lat: 53.657715716828356
lon: 8.136616110137037
WP81:
lat: 53.65910625617243
lon: 8.13303512809199
WP82:
lat: 53.660865067920284
lon: 8.132415011396885
WP83:
lat: 53.662255481013496
lon: 8.128833624346225
WP84:
lat: 53.66401427008092
lon: 8.128213286582469
WP85:
lat: 53.66540455689565
lon: 8.124631494455508
WP86:
lat: 53.667163323271666
lon: 8.124010935557715

Systemtest vom 12.09.2018

Integrationstest:

Methode.
Zunächst erfolgt ein Systemintegrationstest des PG-MATE Systems und LABSKAUS. Der Systemintegrationstest verifiziert die korrekte Funktionsweise der polymorphen Schnittstelle hinsichtlich einer erfolgten Kommunikationsverbindung, sowie der syntaktischen Korrektheit der Daten. Anschließend beurteilt ein Systemexperte des PG-MATE Systems die semantische Korrektheit einiger empfangener Live-Daten der mobilen Sensorbox, die dem PG-MATE Systems durch LABSKAUS bereitgestellt und ein plausibles Ergebnis anhand der vorliegenden Situation der Realwelt validiert wird.

Testdurchführung.

Testspezifikation:

Zur Vorbereitung des Systemtests werden die Komponenten des Systems EPD, Route Planning, Path Planning, DOI, Object Detection und der Controller gestartet. Zusätzlich werden Haggis, der Simulations Adapter und der NoGoSolver gestartet und wie die Testumgebung mit dem System verbunden. Mit den geplanten Testszenarien werden die meisten funktionalen Anforderungen an das System verifiziert und zusätzlich werden die COLREG Regeln 13-17 abgedeckt. Die in den Szenarien benötigten Schiffe werden mit Hilfe von Haggis simulativ über den Simulations Adapter in die Testumgebung eingespeist.

Testkonfiguration:

Für die Szenarien des Systemtests erhält das zu testende Schiff eine Route, in der jedes Szenario durchgespielt wird. Diese Route wurde im Vorfeld des Systemtests bereits zusammengestellt und in der EPD abgespeichert. Im Folgenden werden die Abschnitte für jedes Szenario aufgelistet.

Testszenario	Abschnitt

TS-0	- WP-1:
	  - Lat: 53.5169404273528
	  - Long: 8.171167373657227
	- WP-2:
	  - Lat: 53.52477336808653
	  - Long: 8.181166648864746

TS-1	- WP-1:
	  - Lat: 52.697
	  - Long: 8.16988
	- WP-2:
	  - Lat: 51.577
	  - Long: 8.16988

TS-2	- WP-1:
	  - Lat: 53.5630989074707
	  - Long: 8.172971725463867
	- WP-2:
	  - Lat: 53.56486129760742
	  - Long: 8.172354698181152
	- WP-3:
	  - Lat: 53.566619873046875
	  - Long: 8.171737670898438
	- WP-4:
	  - Lat: 53.56837844848633
	  - Long: 8.171119689941406
	- WP-5:
	  - Lat: 53.57013702392578
	  - Long: 8.170502662658691
	- WP-6:
	  - Lat: 53.571895599365234
	  - Long: 8.169885635375977
	- WP-7:

	- Lat: 53.57365417480469 - Long: 8.169268608093262

TS-3	- WP-1:
	  - Lat: 53.53246307373047
	  - Long: 8.1775484085083
	- WP-2:
	  - Lat: 53.53422164916992
	  - Long: 8.17693233489
	- WP-3:
	  - Lat: 53.536346435546875
	  - Long: 8.179267883300781
	- WP-4:
	  - Lat: 53.53986740112305
	  - Long: 8.178035736083984
	- WP-5:
	  - Lat: 53.5416259765625
	  - Long: 8.17741870880127
	- WP-6:
	  - Lat: 53.54375076293945
	  - Long: 8.179755210876465
	- WP-7:
	  - Lat: 53.545509338378906
	  - Long: 8.179139137268066

TS-4	- WP-1:
	  - Lat: 53.519412994384766
	  - Long: 8.175958633422852
	- WP-2:
	  - Lat: 53.52117156982422
	  - Long: 8.175342559814453
	- WP-3:
	  - Lat: 53.52293395996094
	  - Long: 8.174725532531738
	- WP-4:

	- Lat: 53.52469253540039 - Long: 8.17410945892334 - WP-5: - Lat: 53.526817321777344 - Long: 8.176445007324219 - WP-6: - Lat: 53.5285758972168 - Long: 8.17582893371582 - WP-7: - Lat: 53.53033447265625 - Long: 8.175211906433105
TS-5	- WP-1: - Lat: 53.545509338378906 - Long: 8.179139137268066 - WP-2: - Lat: 53.54726791381836 - Long: 8.178523063659668 - WP-3: - Lat: 53.54902648925781 - Long: 8.177906036376953 - WP-4: - Lat: 53.55078887939453 - Long: 8.177289009094238 - WP-5: - Lat: 53.552547454833984 - Long: 8.1766729354858 - WP-6: - Lat: 53.55430603027344 - Long: 8.176055908203125 - WP-7: - Lat: 53.55606460571289 - Long: 8.17543888092041

Des Weiteren müssen für die Szenarien 2-5 die Routen der simulierten Schiffe konfiguriert werden, diese wurden in Abhängigkeit zur Route des zu testenden Schiffes entwickelt.

Testszenario	Route
TS-2	- WP-1: - Lat: 53.568378448 - Long: 8.171119689941 - WP-2: - Lat: 53.57365417 - Long: 8.1692686080
TS-3	- WP-1: - Lat: 53.53416 - Long: 8.18306 - WP-2: - Lat: 53.53449 - Long: 8.17168
TS-4	- WP-1: - Lat: 53.52179 - Long: 8.17245 - WP-2: - Lat: 53.52345 - Long: 8.17934 - WP-3: - Lat: 53.52551 - Long: 8.18808
TS-5	- WP-1: - Lat: 53.552547 - Long: 8.1766729 - WP-2: - Lat: 53.54375076 - Long: 8.17975521

Basis-Testfälle

ID:	Basis-Test 0 (TC-4, Delmenhorst)	Bemerkungen:
Test Titel:	Controller Wegpunkt abfahren	- Ruderregelung funktioniert nicht ordnungsgemäß. - Die Konfiguration des RaspberryPIs wird angepasst - Ruderregelung funktioniert nach Anpassungen - Regelung erfolgt entsprechend der erwarteten Ergebnisse
Beschreibung:	Das Schiff kann in Richtung eines konstanten, im Controller hinzugefügten, Wegpunkts fahren.	
Funktionale Anforderungen:	F-CA-CT-1, F-CA-CT-4, F-CA-CT-5	
Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - SoG Zwei konstante Wegpunkte. Geschwindigkeit 10 Knoten	
Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen ob der Controller in Richtung des eingegebenen Wegpunkts fährt (mit MATE Control und der EPD). (EPD-Route BT-0)	

Erwartetes Ergebnis:	Schiff fährt in die Richtung des im Controller gegebenen zweiten Wegpunktes mit einer maximalen Abweichung von 60 m (XTE, ablesbar in MatLAB).	
Eingetretenes Ergebnis:	Der in der EPD-Route „BT-0“ hinterlegte Zielwegpunkt wird erfolgreich abgefahren.	
Erfolg/Misserfolg:	Erfolg	

ID:	Basis-Test-1 (Oldenburg)	Bemerkungen:
Test Titel:	Abfahrt einer optimierten Route	-
Beschreibung:	Nachdem eine Route durch das Route-Planning optimiert wurde, wird diese durch das Path-Planning mittels des Controllers abgefahren. Dabei werden statische Hindernisse berücksichtigt.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	

Eingehende-Daten:	- optimierte Route im RTZ-Format von der EPD (Basis-Test-1_optimized) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - Position vom Distribution System	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und Controller über das Distribution System TCS-03: Überprüfung auf Korrektheit der Daten, durch MATLAB Simulink Analysetools	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente sendet Daten (nächster Wegpunkt, RoutenStatus, wished ship speed) an den Controller über das Distribution System	

Eingetretenes Ergebnis:	Ergebnis entspricht dem erwarteten erwarteten Ergebnis. Route wird abgefahren.	
Erfolg/Misserfolg:	Erfolg	
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	Basis-Test-2a (TC-7.1) (Bremen)	Bemerkungen:
Test Titel:	Virtual Handles	- Test wurde nicht durchgeführt, da er bereits in vorherigen Tests durchgeführt wurde.
Beschreibung:	Anwendung der Virtual Handles zur Steuerung der Zuse über ein webbasiertes Interface (Ruder und RPM).	
Funktionale Anforderungen:	F-EPD-9 F-EPD-10 F-EPD-11	
Eingehende-Daten:	Steuerkommandos (RPM und Ruder-Winkel) über die webbasierte Oberfläche der VirtualHandles (View in der EPD)	

Testschritte:	TCS-01: Virtual Handles Interface in der EPD öffnen TCS-02: Virtual Handles-Steuerung über die „Power“-Schaltfläche (oben rechts) einschalten. TCS-03: Durch die Nutzung der VirtualHandles Steuerbefehle die Steuerung des Schiffs übernehmen ○ Ruder-Winkel: maximal-Einschlag von 20° ○ RPM: maximal mögliche Drehzahl: 2000 RPM TCS-04: Virtual Handles-Steuerung über erneutes Klicken der „Power“-Schaltfläche (oben rechts) ausschalten.	
Erwartetes Ergebnis:	- VirtualHandles senden Steuerbefehle (zyklisch und bei Änderung der Einstellgrößen) ○ Ruder-Winkel: maximal-Einschlag von 20° Steuerbord ○ RPM: maximal mögliche Drehzahl: 2000 RPM - Schiff lässt sich per Virtual Handles steuern ○ Ruder-Winkel: das Schiff steuert nach rechts	

	○ RPM: das Schiff beschleunigt auf maximal 2000 RPM - Nach Abschalten der Virtual Handles wird die Steuerung wieder abgegeben. Der Controller deaktiviert dabei den manuellen Fahrmodus. ○ Der Ruder-Winkel verstellt sich in den Normalzustand ○ Die Geschwindigkeit wird auf 700 RPM gedrosselt	
Eingetretenes Ergebnis:	-	
Erfolg/Misserfolg:	nicht getestet.	
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	Basis-Test-2b (TC-7.2) (Bremen)	Bemerkungen:
Test Titel:	Manuelle Steuerung	- Test wurde nicht durchgeführt, da er bereits in vorherigen Tests durchgeführt wurde.
Beschreibung:	Der Controller erhält manuelle Steuerbefehle und gibt diese weiter an die Aktuatoren des physikalischen Testbeds	
Funktionale Anforderungen:	F-CA-CT-3, F-CA-CT-6	
Eingehende-Daten:	Steuerkommandos (RPM und Ruder-Winkel) an den UDP-Input des Controllers, beispielsweise erzeugt durch die Virtual Handles	
Testschritte:	TCS-01: Controller starten TCS-02: Manuelle Steuerbefehle erzeugen und an den Controller senden (Änderung des Ruderwinkels, Geschwindigkeit, RouteStatus: inaktiv) TCS-03: Überprüfung der weitergeleiteten Steuerbefehle durch Reaktion des Schiffes und MATLAB Simulink Analysetools.	

Erwartetes Ergebnis:	- Schiff lässt sich durch die Steuerbefehle Steuern:
	  o Ruder-Winkel: das Schiff steuert nach rechts oder links
	  o RPM: das Schiff beschleunigt oder bremst ab.
	- Nach Änderung des RouteStatus auf aktiv wird die Steuerung wieder abgegeben. Der Controller deaktiviert dabei den manuellen Fahrmodus.	

Eingetretenes Ergebnis:	-	
Erfolg/Misserfolg:	nicht getestet.	
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	Basis-Test-3a (TC-15.1) (Berlin)	Bemerkungen:
Test Titel:	Schiffserkennung mittels Kamera	- AIS-Signale werden mittels MateControl gekappt - Schiffserkennung mittels Kamera funktioniert im Hafengebiet
Beschreibung:	Die Object Detection Komponente muss kontinuierlich die Umgebung in Fahrtrichtung mittels einer Kamera erfassen und Schiffe erkennen. Für erkannte Schiffe, die unter dem Horizont liegen, wird die Position (rechts oder links vom eigenen Schiff) an die DOI als JSON-Objekt gesendet.	- auf der autonom abgefahrenen Route befindet sich kein Entsprechend zu erkennendes Schiff - Test mit Handykamera schlägt fehl, da die Helligkeit des Display nicht ausreicht - Test wird abgebrochen - Test wurde bereits im Labor erfolgreich getestet
Anforderung	F-OD-1 F-OD-2 F-OD-3 F-OD-4 F-OD-5	
Eingehende-Daten:	Echtzeit-Videoaufnahmen einer Webcam von der Umgebung vor dem Schiff	
Testschritte:	TCS-01: „Webcam Selection“ öffnen (durch starten der Komponente), die Webcam „HD Pro Webcam C920 0“ auswählen und auf „Select Camera“ klicken TCS-02: Überprüfen der gesendeten Schiffe an die DOI mit MATE Control.	

	TCS-03: (nach der Testfahrt) 200 aufgenommenen Bilder mit Boxen um erkannte Schiffe auswerten.	
Erwartetes Ergebnis:	● Schiffe, die sich in Fahrtrichtung befinden, werden als Schiff markiert: Die nachträgliche Auswertung der aufgenommen Bildern ergibt, dass 50% der auf den 200 Bildern zu sehenden Schiffe von der Object Detection erkannt wurde. ● Die Position (RIGHT, LEFT, BOTH oder NOTHING) der mittels Bildverarbeitung erkannten Schiffe, die sich unter dem Horizont befinden, werden als JSON-Objekt an die DOI gesendet.	
Eingetretenes Ergebnis:	Test konnte mangels zu erkennender Schiffe nicht durchgeführt werden.	
Erfolg/Misserfolg:	Misserfolg; Labortest erfolgreich	
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	Basis-Test-3b (TC-15.2) (Berlin)	
Test Titel:	Erkannte Schiffe der Object Detection an PP und EPD senden.	
Beschreibung:	Verarbeitung der von der Object Detection Komponente empfangenen JSON-Objekte über die Position der erfassten Schiffe mit der Kamera. Es wird geprüft, ob die erkannten Schiffe bereits mit AIS und/oder Radar erkannt wurden. Ist dies nicht der Fall wird ein „Kameraschiff“ in einem 45° Winkel zur eigenen Fahrrichtung mit einer Geschwindigkeit von 0 Knoten erstellt. Dieses Schiff wird an die EPD und an das Path Planning gesendet.	
Anforderung	F-DOI-9	
Eingehende-Daten:	JSON-Objekte von der Object Detection über die Position (RIGHT, LEFT, BOTH oder NOTHING) der mittels Bildverarbeitung erkannten Schiffe, die sich unter dem Horizont befinden.	

Testschritte:	TCS-01: „Webcam Selection“ öffnen (durch starten der Komponente), die Webcam „HD Pro Webcam C920 0“ auswählen und auf „Select Camera“ klicken TCS-03: In der Konfiguration (configuration.properties) der DOI einstellen, dass nur simulierte Schiffe verarbeitet werden sollen (simulatedOrRealData=simulated) TCS-03: Zeige die fusionierten Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	● Die von der Object Detection erkannten Schiffe werden nicht mit Radar oder AIS erkannt, da keine AIS- und Radardaten verarbeitet werden. ● Wenn sich ein Schiff vor dem eigenen Schiff befinden, ist dieses in dem EPD-DOI-Layer mit der Bezeichnung „Camera“ zu sehen.	
Eingetretenes Ergebnis:		Aufgrund des Misserfolgs von Basis-Test-3a kann 3b nicht erfolgreich getestet werden.
Erfolg/Misserfolg:	Konnte nicht getestet werden; Labortests erfolgreich.	
Testmethode:	Szenariobasiertes Blackbox-Testing	

Testszenarien

ID: TS-1 (Europa)	Kreuzendes Schiff von links (COLREG 15, 16, 17)	Bemerkungen: - fehlerhaft konfigurierter Geschwindigkeits-Faktor im Controller
Beschreibung:	Das Schiff soll das simulierte Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. COLREG 17: a) Ein Schiff verhält sich nicht COLREG-Konform b) Falls das gegenüberliegende Schiff, das Ausweichen muss, nicht reagiert, muss das eigene Schiff trotzdem entsprechend reagieren und ausweichen d) Diese Regel befreit Schiffe nicht von ihrer Pflicht, sich von solchen Situationen fern zu halten	- die Zuse fährt die anzufahrenden Wegpunkte mit einer zu geringen Geschwindigkeit ab. - Geschwindigkeit durch Anpassung des Faktors korrigiert - Soll-Pfad wird für ca. 10 m Fahrweg verlassen (gelber Bereich)
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Startpunkt für das Szenario in Haggis muss erreicht werden ☒ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (10 Knoten) ☒ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	

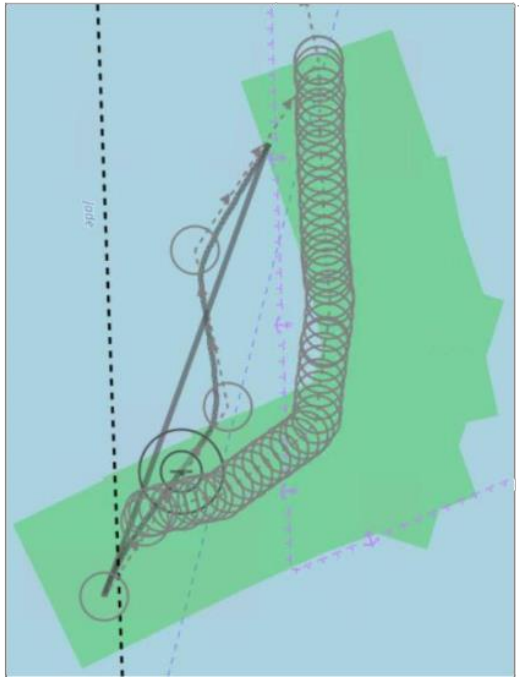
Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen	- TC-1.1 Erfolg
	TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning	- TC-1.2 Erfolg
	TC-2.1-Route-Planning Route empfangen	- TC-2.1 Erfolg
	TC-2.2-Route-Planning Route optimieren	- TC-2.2 Erfolg
	TC-2.2- EPD-seitiger Empfang einer vom Path-Planning optimierten Route	- TC-2.2 Erfolg
	TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route	- TC-2.3 Erfolg
	TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning	- TC-2.4 Erfolg
	TC-4-Controller Geschwindigkeit	- TC-4 Erfolg (im zweiten Versuch, siehe Bemerkungen)
	TC-5.1-Controller Wegpunkt vom Path-Planning empfangen	- TC-5.1 Erfolg
	TC-5.2-Controller Wegpunkt abfahren	- TC-5.2 Erfolg
	TC-6.1-AIS-Daten anzeigen	- TC-6.1 Erfolg
	TC-6.2-Radar-Daten anzeigen	- TC-6.2 Erfolg
	TC-6.3-Grid-Daten anzeigen	- TC-6.3 Erfolg
	TC-6.4-Polygone anzeigen	- TC-6.4 Erfolg
	TC-6.5-Schiffsposition anzeigen	- TC-6.5 Erfolg
	TC-9.2-Objekterkennung (links neben dem eigenen Schiff)	- TC-9.2 Erfolg
	TC-10-Datenfusion	- TC-10 Erfolg
	TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD)	- TC-11.1 Erfolg
	TC-11.2-Path Planning Kommunikation mit dem NoGoSolver	- TC-11.2 Erfolg

	TC-11.3-Path Planning Kommunikation mit der DOI	- TC-11.3 Erfolg
	TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller)	- TC-11.4 Erfolg
	TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD)	- TC-11.5 Erfolg
	TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	- TC-13 Erfolg
	TC-14.4.2-Path-Planning Kreuzen von links (Ausweichen)	- TC-14.4.2 Erfolg

ID: TS-2 (Asien)	Kreuzendes Schiff von rechts (wir müssen ausweichen) (COLREG 15, 16, 17)	Bemerkungen: - der berechnete Soll-Pfad wird erfolgreich abgefahren - es findet keine Kollision mit dem simulierten Schiff statt
Beschreibung:	Das Schiff soll das Fremdschiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. COLREG 15: Falls zwei Schiffe sich kreuzen muss das Schiff, welches sein Gegenüber auf der Steuerbord-Seite hat reagieren und ausweichen	
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Startpunkt für das Szenario in Haggis muss erreicht werden ☒ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (10 Knoten) ☒ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	

Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen	- TC-1.1 Erfolg
	TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning	- TC-1.2 Erfolg
	TC-2.1-Route-Planning Route empfangen	- TC-2.1 Erfolg
	TC-2.2-Route-Planning Route optimieren	- TC-2.2 Erfolg
	TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route	- TC-2.3 Erfolg
	TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning	- TC-2.4 Erfolg
	TC-4-Controller Geschwindigkeit	- TC-4 Erfolg
	TC-5.1-Controller Wegpunkt vom Path-Planning empfangen	- TC-5.1 Erfolg
	TC-5.2-Controller Wegpunkt abfahren	- TC-5.2 Erfolg
	TC-6.1-AIS-Daten anzeigen	- TC-6.1 Erfolg
	TC-6.2-Radar-Daten anzeigen	- TC-6.2 Erfolg
	TC-6.3-Grid-Daten anzeigen	- TC-6.3 Erfolg
	TC-6.4-Polygone anzeigen	- TC-6.4 Erfolg
	TC-6.5-Schiffsposition anzeigen	- TC-6.5 Erfolg
	TC-9.3-Objekterkennung (rechts neben dem eigenen Schiff)	- TC-9.3 Erfolg
	TC-10-Datenfusion	- TC-10 Erfolg
	TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD)	- TC-11.1 Erfolg
	TC-11.2-Path Planning Kommunikation mit dem NoGoSolver	- TC-11.2 Erfolg

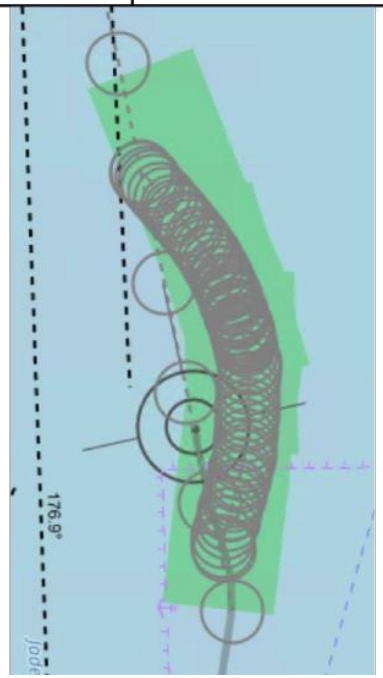
	TC-11.3-Path Planning Kommunikation mit der DOI	-	TC-11.3 Erfolg
	TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller)	-	TC-11.4 Erfolg
	TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD)	-	TC-11.5 Erfolg
	TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	-	TC-13 Erfolg
	TC-14.3-Path-Planning Kreuzen von rechts	-	TC-14.3 Erfolg



ID: TS-3 (America)	Entgegenkommendes Schiff (COLREG 14, 16, 17)	Bemerkungen:
Beschreibung:	Das Schiff soll das entgegenkommende Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. Danach übernimmt der Kapitän kurzzeitig die Kontrolle mithilfe der Virtual Handles. COLREG 14: a) Beide Schiffe weichen auf der Steuerbordseite aus b) Diese Situation gilt falls vermutet wird, dass ein gegenüberliegendes Schiff auf das Eigene zufährt oder sich annähert c) Falls die Vermutung besteht, dass diese Situation gilt, muss ausgewichen werden	- zwischenzeitlicher Ausfall des RabbitMQ (Navibox) - RabbitMQ auf IPC nachinstalliert - Testlauf konnte durchgeführt werden - Geschwindigkeitssteuerung unzuverlässig (6,5 Knoten) - Pfadverfolgung korrekt, bis Schiff überholt wurde - Keine Kollision mit anderem Schiff - Rückführung auf globalen Pfad fehlerhaft - Fehler auf Timeouts beim RabbitMQ zurückzuführen - (Komponenten funktionsfähig, daher Test erfolgreich)
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Startpunkt für das Szenario in Haggis muss erreicht werden ☒ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (10 Knoten) ☒ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	

Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen	- TC-1.1 Erfolg
	TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning	- TC-1.2 Erfolg
	TC-2.1-Route-Planning Route empfangen	- TC-2.1 Erfolg
	TC-2.2-Route-Planning Route optimieren	- TC-2.2 Erfolg
	TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route	- TC-2.3 Erfolg
	TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning	- TC-2.4 Erfolg
	TC-4-Controller Geschwindigkeit	- TC-4 kein Erfolg, siehe Bemerkungen
	TC-5.1-Controller Wegpunkt vom Path-Planning empfangen	- TC-5.1 Erfolg
	TC-5.2-Controller Wegpunkt abfahren	- TC-5.2 Erfolg
	TC-6.1-AIS-Daten anzeigen	- TC-6.1 Erfolg
	TC-6.2-Radar-Daten anzeigen	- TC-6.2 Erfolg
	TC-6.3-Grid-Daten anzeigen	- TC-6.3 Erfolg
	TC-6.4-Polygone anzeigen	- TC-6.4 Erfolg
	TC-6.5-Schiffsposition anzeigen	- TC-6.5 Erfolg
	TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht	- TC-8 Erfolg
	TC-10-Datenfusion	- TC-10 Erfolg
	TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD)	- TC-11.1 Erfolg

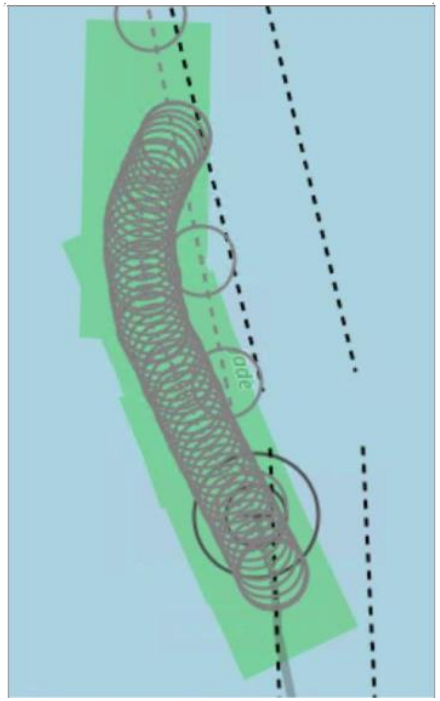
	TC-11.2-Path Planning Kommunikation mit dem NoGoSolver	- TC-11.2 Erfolg
	TC-11.3-Path Planning Kommunikation mit der DOI	- TC-11.3 Erfolg
	TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller)	- TC-11.4 Erfolg
	TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD)	- TC-11.5 Erfolg
	TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	- TC-13 Erfolg
	TC-14.2-Path-Planning entgegenkommend	- TC-14.2 Erfolg



ID: TS-6 (Australien)	Überholmanöver Schiff (COLREG 13)	Bemerkungen:
Beschreibung:	Das Schiff soll im Autopiloten das Schiff erkennen, überholen und auf die ursprünglich geplante Route zurückkehren. COLREG 13: a) Das Schiff, das überholt wird, muss von anderen Schiffen gemieden werden b) Ein Schiff gilt als überholt, sobald das überholte Schiff deren Seitenlichter sieht c) Falls es irgendwelche bedenken gibt, darf nicht überholt werden	- das Schiff fährt mehrfach in GoodWin-Shapes - Pfadberechnung zu nah am GoodWinShape - CPA-Berechnung lässt GoodWin-Shape zu stark varriieren - keine Kollision mit überholtem Schiff - Rückführung auf globalen Pfad fehlerhaft, da Fahrt innerhalb eines GoodWin-Shape („Blindflug“) - nachdem das Schiff aus dem GoodWin-Shape herausgefahren ist, findet eine erfolgreiche Rückführung auf den globalen Pfad statt
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Startpunkt für das Szenario in Haggis muss erreicht werden ☒ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (10 Knoten) ☒ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in das Szenario starten zu können (fliegender Start)	

Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning TC-2.1-Route-Planning Route empfangen TC-2.2-Route-Planning Route optimieren TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-8-Objekterkennung (vor dem eigenen Schiff) bei guter Sicht TC-10-Datenfusion TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD) TC-11.2-Path Planning Kommunikation mit dem NoGoSolver	- TC-1.1 Erfolg - TC-1.2 Erfolg - TC-2.1 Erfolg - TC-2.2 Erfolg - TC-2.3 Erfolg - TC-2.4 Erfolg - TC-4 Erfolg - TC-5.1 Erfolg - TC-5.2 Erfolg - TC-6.1 Erfolg - TC-6.2 Erfolg - TC-6.3 Erfolg - TC-6.4 Erfolg - TC-6.5 Erfolg - TC-8 Erfolg - TC-10 Erfolg - TC-11.1 Erfolg - TC-11.2 Erfolg

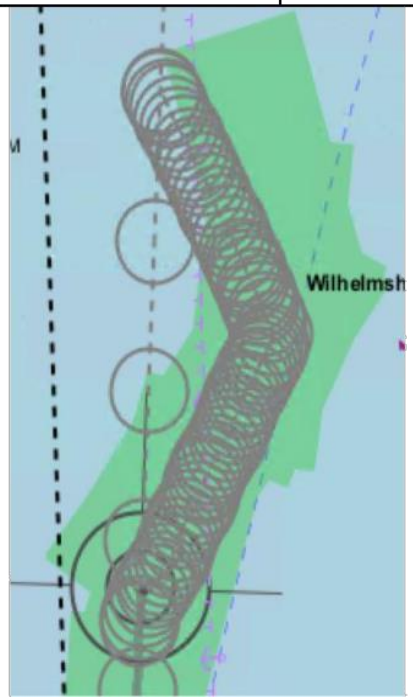
	TC-11.3-Path Planning Kommunikation mit der DOI	-	TC-11.3 Erfolg
	TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller)	-	TC-11.4 Erfolg
	TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD)	-	TC-11.5 Erfolg
	TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	-	TC-13 Erfolg
	TC-14.1-Path-Planning Überholen	-	TC-14.1 Erfolg mit Einschränkungen



ID: TS-5 (Brocken)	Ausweichen eines dynamischen und statischen Hindernisses	Bemerkungen:
Beschreibung:	Das Schiff soll das simulierte Schiff im Autopiloten erkennen, wenn nötig ausweichen und auf die ursprünglich geplante Route zurückkehren. Dabei soll eine Kollision verhindert werden. Zusätzlich soll ein im Ausweichpfad liegendes statisches Hindernis bei der Ausweichplanung berücksichtigt werden. a) Ein Schiff befindet sich auf Kollisionskurs b) Falls das gegenüberliegende Schiff, das Ausweichen muss, nicht reagiert, muss das eigene Schiff trotzdem entsprechend reagieren und ausweichen d) Diese Regel befreit Schiffe nicht von ihrer Pflicht, sich von solchen Situationen fern zu halten e) statische Hindernisse in der Ausweichpfadplanung müssen ebenfalls berücksichtigt werden, um eine Kollision zu vermeiden.	- dem entgegenkommenden Schiff wird nach rechts ausgewichen - der dort vorhandenen virtuellen Untiefe wird ebenfalls nach rechts ausgewichen - dabei findet keine Kollision statt - Rückführung auf globalen Pfad gelingt - Geschwindigkeit wird korrekt geregelt
Vorbedingungen:	☒ Kommunikation zwischen den Komponenten muss gegeben sein ☒ Startpunkt für das Szenario in Haggis muss erreicht werden ☒ Geschwindigkeit muss passend zur Geschwindigkeit im Szenario sein (10 Knoten) ☒ 50 – 100m vor Startpunkt passende Geschwindigkeit wählen/fahren um nahtlos in	

	das Szenario starten zu können (fliegender Start)	
Testfälle:	TC-1.1-Route mit Schiffsparametern erstellen TC-1.2 Senden einer Route und Schiffsparameter an das Route-Planning TC-2.1-Route-Planning Route empfangen TC-2.2-Route-Planning Route optimieren TC-2.3- EPD-seitiger Empfang einer vom Route-Planning optimierten Route TC-2.4-Senden einer optimierten Route von der EPD zum Path-Planning TC-4-Controller Geschwindigkeit TC-5.1-Controller Wegpunkt vom Path-Planning Empfangen TC-5.2-Controller Wegpunkt abfahren TC-6.1-AIS-Daten anzeigen TC-6.2-Radar-Daten anzeigen TC-6.3-Grid-Daten anzeigen TC-6.4-Polygone anzeigen TC-6.5-Schiffsposition anzeigen TC-8-Objekterkennung (vor und links neben dem eigenen Schiff) TC-10-Datenfusion TC-11.1-Path Planning Kommunikation mit der EPD (empfangen von Daten von der EPD)	- TC-1.1 Erfolg - TC-1.2 Erfolg - TC-2.1 Erfolg - TC-2.2 Erfolg - TC-2.3 Erfolg - TC-2.4 Erfolg - TC-4 Erfolg - TC-5.1 Erfolg - TC-5.2 Erfolg - TC-6.1 Erfolg - TC-6.2 Erfolg - TC-6.3 Erfolg - TC-6.4 Erfolg - TC-6.5 Erfolg - TC-8 Erfolg - TC-10 Erfolg - TC-11.1 Erfolg

	TC-11.2-Path Planning Kommunikation mit dem NoGoSolver	-	TC-11.2 Erfolg
	TC-11.3-Path Planning Kommunikation mit der DOI	-	TC-11.3 Erfolg
	TC-11.4-Path Planning Kommunikation mit dem Controller (senden von Daten an den Controller)	-	TC-11.4 Erfolg
	TC-11.5-Path Planning Kommunikation mit der EPD (senden von Daten an die EPD)	-	TC-11.5 Erfolg
	TC-12-Path-Planning Berücksichtigung statischer Hindernisse	-	TC-12 Erfolg
	TC-13-Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	-	TC-13 Erfolg
	TC-14.2-Path-Planning entgegenkommend	-	TC-14.2 Erfolg



Testfälle:

ID:	TC-1.1	Bemerkungen:
Test Titel:	Route mit Schiffsparametern erstellen	
Beschreibung:	Zusammenstellung einer Route in der EPD mit Schiffsparametern.	
Funktionale Anforderungen:	F-EPD-2 F-EPD-5	
Eingehende-Daten:		
Testschritte:	TCS-01: Wegpunktwerkzeug in der EPD auswählen und eine Route, bestehend aus zwei Wegpunkten, auf der Karte erstellen Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 Optional: TCS-02a: Route im Routenmanager der EPD per Doppelklick öffnen und ggfs. Geschwindigkeiten (max) der einzelnen Wegpunkte setzen (auf 10 Knoten)	

	TCS-02: Schiffsparameter in der “PG Mate”-Einstellungsseite der EPD hinterlegen und speichern Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Erwartetes Ergebnis:	Verbindet die gesetzten Wegpunkte zu einer Route, bestehend auf WP 1 und 2 mit einer maximalen WP-Leg-Geschwindigkeit von 10 Knoten und speichert die folgenden Schiffsparameter. Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 Schiffsparameter: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-1.2	Bemerkungen:
Test Titel:	Senden einer Route und Schiffsparameter an das Route-Planning	
Beschreibung:	Die zusammengestellte RTZ-Route, bestehend aus: Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 10 Knoten) wird mit den hinterlegten Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5 der Zuse per HTTP-Request an das Route-Planning gesendet.	
Funktionale Anforderungen:	F-EPD-2 F-EPD-5	
Eingehende-Daten:		

Testschritte:	TCS-01: Route im Route-Manager der EPD auswählen und über die Schaltfläche „Route optimieren“ an das Route-Planning senden.	
Erwartetes Ergebnis:	Die ausgewählte RTZ-Route und die hinterlegten Schiffsparameter wird mit den hinterlegten Geschwindigkeiten der einzelnen Wegpunkte an das Route-Planning gesendet: Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 10 Knoten) wird mit den hinterlegten Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.1	Bemerkungen:
Test Titel:	Route-Planning Route empfangen	
Beschreibung:	Eine Route, bestehend aus den beiden Wegpunkten: Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 10 Knoten) soll zusammen mit den hinterlegten Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5 vom Route-Planning empfangen werden.	
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5	
Eingehende-Daten:	in Route-Planning eingehend:	

	- RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit (10 Knoten) zwischen den Wegpunkten: ○ Wegpunkt 1: ○ Lat: 53.51765489239272 ○ Lon: 8.17657470703125 ○ Wegpunkt 2: ○ Lat: 53.667163323271666 ○ Lon: 8.124010935557715 - Schiffsinformationen ○ Länge: 8.0 ○ Breite: 2.5 ○ Höhe 2.5 ○ Tiefgang: 0.5	
Testschritte:	TCS-01: nicht-optimierte Route von der EPD an das Route Planning senden: ○ Wegpunkt 1: ○ Lat: 53.51765489239272 ○ Lon: 8.17657470703125 ○ Wegpunkt 2: ○ Lat: 53.667163323271666 ○ Lon: 8.124010935557715	

Erwartetes Ergebnis:	Das Route-Planning erhält eine Route von der EPD und die Schiffsinformationen Wegpunkt 1: Lat: 53.51765489239272 Lon: 8.17657470703125 Wegpunkt 2: Lat: 53.667163323271666 Lon: 8.124010935557715 und einer WP-Leg-Geschwindigkeit von 10 Knoten) Schiffsinformationen: Länge: 8.0 Breite: 2.5 Höhe 2.5 Tiefgang: 0.5	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.2	Bemerkungen:
Test Titel:	Route-Planning Route optimieren	
Beschreibung:	Optimieren einer ausgewählten Route	
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5	
Eingehende-Daten:	in Route-Planning eingehend: - RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit (10 Knoten) zwischen den Wegpunkten ○ Wegpunkt 1: ○ Lat: 53.51765489239272 ○ Lon: 8.17657470703125 ○ Wegpunkt 2: ○ Lat: 53.667163323271666 ○ Lon: 8.124010935557715	
Testschritte:	TCS-01: eingehende Route optimieren	
Erwartetes Ergebnis:	Das Route-Planning optimiert eine ausgewählte Route, diese wird via HTTP zurück an die EPD übertragen. Diese berücksichtigt in ihrer Gestalt statische Hindernisse, geliefert vom NoGoSolver. Enthaltene Wegpunkte: Siehe:	

	Anlage 1: optimierte RTZ-Route	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.3	Bemerkungen:
Test Titel:	EPD-seitiger Empfang einer vom Route-Planning optimierten Route	
Beschreibung:	Empfang einer optimierten Route des Route-Planning auf Basis einer ausgewählten Route	
Funktionale Anforderungen:	F-EPD-6 F-EPD-7	
Eingehende-Daten:	1. in EPD eingehend: - RTZ-Route vom Route-Planning eingehend via HTTP in die EPD, bestehend aus berechneten Wegpunkten und der jeweiligen maximal zu fahrenden Geschwindigkeit von 10 Knoten zwischen den Wegpunkten: Siehe: Anlage 1: optimierte RTZ-Route	

Testschritte:	TCS-01: Nach Erhalt der optimierten Route (Siehe: Anlage 1: optimierte RTZ-Route), überprüfen, ob diese mit einem „optimized“ im Routenname im Routenmanager der EPD abgespeichert wird und sichtbar in der Karte dargestellt wird.	
Erwartetes Ergebnis:	Die EPD erhält eine vom Route-Planning optimierte Route via HTTP, die die gleichen Start- und Zielpunkte wie die Originalroute hat und Hindernisse, visualisiert durch “NoGo-Areas”, meidet. Die Geschwindigkeiten der Wegpunktverbindungen sind auf 10 Knoten gesetzt. Enthaltene Wegpunkte sind hier aufgelistet: Anlage 1: optimierte RTZ-Route	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-2.4	Bemerkungen:
Test Titel:	Senden einer optimierten Route von der EPD zum Path-Planning	
Beschreibung:	Eine in der EPD gespeicherte optimierte Route wird zum Start der Fahrt an das Path-Planning gesendet. Die optimierte Route (Siehe: Anlage 1: optimierte RTZ-Route) enthält dabei Abfahrgeschwindigkeiten von 10 Knoten.	
Funktionale Anforderungen:	F-EPD-6 F-EPD-7	
Eingehende-Daten:	in Path-Planning eingehend - RTZ-Route von der EPD via HTTP, beinhaltet alle zuvor vom Route-Planning berechneten Wegpunkte inklusive der Geschwindigkeiten zwischen den Wegpunkten Siehe: Anlage 1: optimierte RTZ-Route - Konfigurierte Schiffsinformationen aus der EPD als JSON-Objekte via HTTP	

Testschritte:	TCS-01: Schiffsinformationen in der Einstellungsmaske für Schiffsinformationen in der EPD setzen. TCS-02: Sicher gehen, dass Route ausgewählt (Anlage 1: optimierte RTZ-Route) ist und in der EPD „Route senden“ klicken. TCS-03: Prüfen, ob Route im Path-Planning eingeht.	
Erwartetes Ergebnis:	Das Path-Planning erhält eine vom Route-Planning optimierte Route (Anlage 1: optimierte RTZ-Route) als RTZ-Route via HTTP, die die gleichen Start- und Zielpunkte wie die Originalroute hat und statische Hindernisse(“NoGo-Areas”)meidet.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-3	Bemerkungen:
Test Titel:	Ungültige Route optimieren	
Beschreibung:	Zusammenstellung einer Route die einen Wegpunkt auf Land bzw. einer NoGoArea besitzt. Anschließend soll der Versuch gestartet werden, diese zu optimieren.	
Funktionale Anforderungen:	F-CA-RP-1 F-CA-RP-2 F-CA-RP-4 F-CA-RP-5	
Eingehende-Daten:	1. in Route-Planning eingehend: - RTZ-Route eingehend via HTTP von der EPD, bestehend aus Wegpunkten ○ WP1: ○ Lat: 53.56396732809957 ○ Lon: 8.124732971191406 ○ WP2: ○ Lat: 53.5792576599901 ○ Lon: 8.178634643554688 2. in EPD eingehend (im Anschluss an 1): - Fehlermeldung als JSON-Objekt via HTTP, welches in	

	der EPD das Darstellen einer Fehlermeldung auslöst	
Testschritte:	TCS-01: Wegpunktwerkzeug auswählen und auf der Karte eine Route zusammenstellen, wobei ein Wegpunkt auf Land zu setzen ist WP1: Lat: 53.56396732809957 Lon: 8.124732971191406 WP2: Lat: 53.5792576599901 Lon: 8.178634643554688 TCS-02: Sichergehen, dass Route ausgewählt ist und „Route optimieren“ auswählen TCS-03: Fehlermeldung mit “OK” bestätigen	
Erwartetes Ergebnis:	Rückmeldung als Hinweisfenster der EPD: “Es wurde keine Route gefunden!”	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-4	Bemerkungen:
Test Titel:	Controller Geschwindigkeit	
Beschreibung:	Eine in der EPD vorgegebene, optimierte Route, bestehend aus 2 Wegpunkten, wird an das Path-Planning gesendet. Die Wegpunktgeschwindigkeit wird dabei jeweils auf 10 Knoten gesetzt.	
Funktionale Anforderungen:	F-CA-CT-1, F-CA-CT-2, F-CA-CT-4	
Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - Current Speed Die nächsten 2 Wegpunkte vom Path-Planning. Die gewünschte Wegpunktgeschwindigkeit.	
Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: In der EPD eine Route aus 2 Wegpunkten erstellen, die nicht durch statische Hindernisse führen.	

	TCS-03: Geschwindigkeit der Wegpunkte auf 10 Knoten einstellen. TCS-04: Die Route ans Path-Planning schicken. TCS-05: Überprüfen der Geschwindigkeit des Schiffes mit MATE Control.	
Erwartetes Ergebnis:	Das Schiff fährt mit der vorgegebenen Geschwindigkeit (zwischen 9 und 11, optimal 10.00 Knoten).	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-5.1	Bemerkungen:
Test Titel:	Controller Wegpunktpaar vom Path-Planning empfangen	
Beschreibung:	Der Controller empfängt vom Path-Planning ein Wegpunktpaar im NMEA2000 Standard. Dabei wird das Path-Planning mit einer beliebigen Route gestartet und die anliegenden Werte (Positionsdaten der Wegpunkte) im Controller mit den Logs im Path-Planning verglichen.	
Funktionale Anforderungen:	F-CA-PP-9, F-CA-CT-1, F-CA-CT-2, F-CA-CT-5	
Eingehende-Daten:	Die nächsten 2 Wegpunkte vom Path-Planning	
Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen, ob der Controller den gesendeten Wegpunkt vom Path-Planning empfängt (mit MATLAB Simulink Überwachungstools).	

Erwartetes Ergebnis:	Controller empfängt die nächsten anzufahrenden Wegpunkte von der Path-Planning-Komponente, die in der Path-Planning Komponente herausgesendet wurden.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-5.2	Bemerkungen:
Test Titel:	Controller Wegpunkt abfahren	
Beschreibung:	Das Schiff kann in Richtung eines im Controller hinzugefügten Wegpunkts fahren.	
Funktionale Anforderungen:	F-CA-CT-1, F-CA-CT-4, F-CA-CT-5	

Eingehende-Daten:	Sensor-Daten der Navibox: - Position - Heading - SoG Die nächsten 2 Wegpunkte. Geschwindigkeit 10 Knoten	
Testschritte:	TCS-01: Das Polymorphic Interface mit MATE Control-Adapter starten und im Zuse-Tab von Mate-Control arbeiten TCS-02: Überprüfen ob der Controller in Richtung des eingegebenen Wegpunkts fährt (mit MATE Control und der EPD).	
Erwartetes Ergebnis:	Schiff fährt in die Richtung des im Controller gegebenen zweiten Wegpunktes mit einer maximalen Abweichung von 60 m (XTE, ablesbar in MatLAB).	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.1	Bemerkungen:
Test Titel:	Anzeige AIS-Daten	
Beschreibung:	Anzeige der AIS-Daten in der EPD.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.1	
Eingehende-Daten:	- Sensor-Daten (kontinuierlich) vom HUB über den NMEA-Sensor - Koordinaten erkannter Schiffe/Objekte (AIS, Radar und fusionierte Daten) als JSON via HTTP von der DOI	
Testschritte:	TCS-01: Empfang von AIS-Daten per UDP vom HUB und Darstellung auf der EPD-Karte TCS-02: Überprüfung von AIS-Roh-Daten von der DOI (Layer: DOIDataLayer)	
Erwartetes Ergebnis:	Anzeige der von der DOI übermittelten AIS-Daten im Layer „DOIDataLayer“ der EPD.	

Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.2	Bemerkungen:
Test Titel:	Anzeige Radar-Daten	
Beschreibung:	Anzeige der Radar-Daten in der EPD.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.2	
Eingehende-Daten:	- Koordinaten erkannter Schiffe/Objekte (AIS, Radar und fusionierte Daten) als JSON via HTTP von der DOI	
Testschritte:	TCS-01: Überprüfung von Radar-Roh-Daten von der DOI (Layer: DOIDataLayer)	

Erwartetes Ergebnis:	Anzeige der von der DOI übermittelten Radar-Daten im Layer „DOIDataLayer“ der EPD	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.3	Bemerkungen:
Test Titel:	Anzeige Grid-Daten	
Beschreibung:	Anzeige der Grid-Daten in der EPD. Diese werden als JSON vom Route-Planning nach Optimierung einer nicht-optimierten Route an die EPD übertragen.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.3	
Eingehende-Daten:	- Routendaten - NoGo-Daten und Griddaten als JSON via HTTP vom Route-Planning	

	(bereits vom Routenberechnen vorhanden)	
Testschritte:	TCS-01: Überprüfung der Grid-Daten anhand der EPD Karte (Layer: SOIDataLayer)	
Erwartetes Ergebnis:	Anzeige der vom Route-Planning übermittelten Grid-Daten im Layer „SOIDataLayer“ der EPD	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.4	Bemerkungen:
Test Titel:	Anzeige Polygone	
Beschreibung:	Anzeige der Polygone in der EPD. Diese werden als JSON vom Route-Planning nach Optimierung einer nicht-optimierten Route an die EPD übertragen.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.4	
Eingehende-Daten:	- Routendaten vom Route-Planning - NoGo-Daten und Griddaten als JSON via HTTP vom Route-Planning (bereits vom Routenberechnen vorhanden)	
Testschritte:	TCS-01: Überprüfung der Polygone anhand der EPD Karte (Layer: SOIDataLayer)	

Erwartetes Ergebnis:	Anzeige der vom Route-Planning übermittelten Polygone im Layer „SOIDataLayer“ der EPD	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-6.5	Bemerkungen:
Test Titel:	Anzeige der eigenen Schiffsposition	
Beschreibung:	Anzeige der eigenen Schiffsposition in der EPD.	
Funktionale Anforderungen:	F-EPD-1 F-EPD-1.5	
Eingehende-Daten:	- Sensor-Daten (kontinuierlich) vom Polymorphic Interface (NMEA0183)	
Testschritte:	TCS-01: Überprüfung der eigenen Schiffsposition anhand der EPD Karte und MATE Control	

Erwartetes Ergebnis:	Anzeige der korrekten, eigenen Schiffsposition auf der EPD-Seekarte, mit korrektem Heading. Position aus MATE Control und der EPD stimmt überein.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-8	Bemerkungen:
Test Titel:	Objekterkennung (vor dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (vor dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen. Zusätzlich werden JSON-Objekte über http von erkannten Schiffe mittels Bilderkennung von der Object Detection empfangen	

Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.1 F-DOI-4 F-DOI-5	
Eingehende-Daten:	- Sensordaten des AIS in NMEA-2000 Format, mit erkanntem Schiff welches bis zu 3500m von der eigenen Position entfernt ist. - Sensordaten des Radars in NMEA-2000 konformen Format, mit erkanntem Schiff welches bis zu 3500m von der eigenen Position entfernt ist. - JSON-Objekt über die Position (rechts, links, beides) welches von der Object Detection erkannt wurde	
Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layer an und prüfe, ob umliegenden Schiffe	

	von Radar und AIS angezeigt werden.	
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten von Radar werden von der DOI-Komponente empfangen - Erkannte Schiffe von der Object-Detection-Komponente werden empfangen	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-9.1	Bemerkungen:
Test Titel:	Objekterkennung (hinter dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (hinter dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen.	
Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.3 F-DOI-5	
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format, mit erkanntem Schiff, welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet - Sensordaten des Radars im NMEA-2000 konformen Format, mit erkanntem Schiff welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet	

Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe, welche von der Object-Detection-Komponente erkannt wurden, (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten vom Radar werden von der DOI-Komponente empfangen	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-9.2	Bemerkungen:
Test Titel:	Objekterkennung (links neben dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (links neben dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen.	
Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.2 F-DOI-5	
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format, mit erkanntem Schiff, welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet - Sensordaten des Radars im NMEA-2000 konformen Format, mit erkanntem Schiff welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet	

Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe, welche von der Object-Detection-Komponente erkannt wurden, (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten vom Radar werden von der DOI-Komponente empfangen	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-9.3	Bemerkungen:
Test Titel:	Objekterkennung (rechts neben dem eigenen Schiff)	
Beschreibung:	Die DOI-Komponente muss kontinuierlich Informationen über alle umliegenden Schiffe (rechts neben dem eigenen Schiff) durch NMEA2000 Radar- und AIS-Nachrichten empfangen.	
Anforderung	F-DOI-1 F-DOI-2 F-DOI-3.2 F-DOI-5	
Eingehende-Daten:	- Sensordaten des AIS im NMEA-2000 Format, mit erkanntem Schiff, welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet - Sensordaten des Radars im NMEA-2000 konformen Format, mit erkanntem Schiff welches sich in einer Entfernung von bis zu 2 sm neben bzw. hinter dem eigenen Schiff befindet	

Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe, welche von der Object-Detection-Komponente erkannt wurden, (rechts/links) über http in JSON (automatisch) TCS-03: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	- Alle Informationen aus den eingehenden Sensordaten vom AIS werden von der DOI-Komponente empfangen - Alle Informationen aus den eingehenden Sensordaten vom Radar werden von der DOI-Komponente empfangen	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-10	Bemerkungen:
Test Titel:	Datenfusion	
Beschreibung:	Die DOI-Komponente muss empfangene AIS-Nachrichten (statisch und dynamisch) und empfangene Radar Tracks untereinander fusionieren (AIS mit AIS und AIS mit Radar) Die DOI-Komponente muss bei allen assoziierten Schiffen in der Umgebung (2 sm Umkreis ums eigene Schiff) die objektspezifischen Parameter Größe, Geschwindigkeit, und Position neu berechnen.	
Anforderung	F-DOI-6 F-DOI-7 F-DOI-8	
Eingehende-Daten:	- Sensordaten des Radars im NMEA-2000 konformen Format in einem Umkreis ums eigenen Schiff von bis zu 2 sm (mind. 15 Nachrichten) - Sensordaten des GPS in NMEA-2000 - Sensordaten des Headings in NMEA-2000 - JSON-Objekt über die Position (rechts, links) welche durch Bildverarbeitung erkannt wurde	

Testschritte:	TCS-01: Empfange AIS, Radar und Positionsdaten in NMEA2000 über UDP (automatisch) TCS-02: Empfange Position der Schiffe (rechts/links) über http in JSON (automatisch) TCS-03: Fusioniere die Schiffe (automatisch) TCS-04: Zeige die Rohdaten der Schiffe über den EPD-DOI-Layers an TCS-05: Zeige die fusionierten Schiffe über den EPD-DOI-Layers an	
Erwartetes Ergebnis:	Schiffe unter 20 m müssen kein AIS besitzen. Daher wird das Schiff nur mit Radar erkannt. Der empfangene Radartrack wird nicht mit einem AIS-Track fusioniert. D.h. der EPD-DOI-Layer zeigt ein Schiff, welches von Radar erkannt wurde, an aber nicht mit AIS fusioniert wurde (FRadarXXX).	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Anforderungstest	

ID:	TC-11.1	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit der EPD (empfangen von Daten von der EPD)	
Beschreibung:	Eine globale Route, die bereits optimiert wurde enthält mehrere Wegpunkte. Diese Route wird an das Path-Planning über eine http-Verbindung übermittelt. Eine erfolgreiche Übertragung wird in der EPD visualisiert.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route	

	TCS-02: Die EPD visualisiert den Erfolg der Übertragung durch einen Grünen Haken. TCS-03: Die Path-Planning Komponente gibt nach kurzer Berechnungszeit einen Pfad an die EPD zurück.	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente empfängt eine valide, globale Route von der EPD über die http-Verbindung - Das erfolgreiche Übertragen wird in der EPD visualisiert.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.2	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit dem NoGoSolver	
Beschreibung:	Die Path-Planning-Komponente muss kontinuierlich Informationen über alle statischen Hindernisse vom NoGoSolver anfragen, entgegennehmen und decodieren können.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- JSON-Objekte vom NoGoSolver: WKT Geometrien, die statische Hindernisse repräsentieren. Diese umfassen den durch das Path-Planning spezifizierten Koordinatenbereich und der angegebenen Schiffsinformationen.	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route	

	TCS-02: Überprüfung der Verbindung zwischen Path-Planning und NoGoSolver durch Log-Ausgaben (optional: Visualisierung durch die EPD)	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente fragt statische Hindernisse beim NoGoSolver an und empfängt von diesem die statischen Hindernisse. Diese Daten stimmen mit vorhanden Kartendaten überein (Verifizierbar durch Expertenwissen, Boardinstrumente der Zuse).	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.3	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit der DOI	
Beschreibung:	Das Path-Planning muss kontinuierlich Informationen über alle dynamischen Hindernisse (Schiffe) von der DOI-Komponente entgegennehmen und decodieren können.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- JSON-Objekte von der DOI-Komponente, enthalten identifizierte Objekte (nur fusionierte Objekte: Positionen, COG, SOG, Länge und Breite der Schiffe)	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und DOI-Komponente durch log-Ausgaben und Vergleich mit den Boardinstrumenten der Zuse	

Erwartetes Ergebnis:	- Die Path-Planning-Komponente empfängt dynamische, zum Teil fusionierte Daten von der DOI-Komponente, welche alle Schiffe in einem 2 sm Radius um das eigene Schiff repräsentieren.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.4	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit dem Controller (senden von Daten an den Controller)	
Beschreibung:	Der berechnete valide Pfad wird über das Distribution System an den Controller geschickt.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- optimierte Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente	

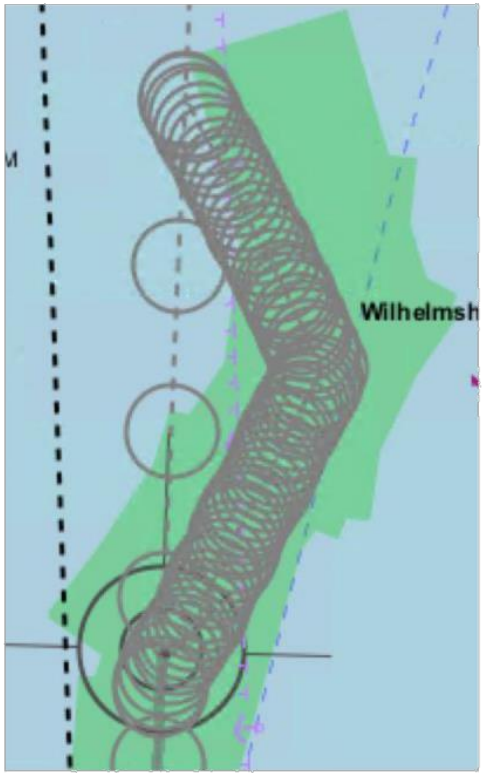
	- Position vom Distribution System	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route TCS-02: Überprüfung der Verbindung zwischen Path-Planning und Controller über das Distribution System TCS-03: Überprüfung auf Korrektheit der Daten, durch MATLAB Simulink Analysetools	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente sendet Daten (nächster Wegpunkt, RoutenStatus, wished ship speed) an den Controller über das Distribution System	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-11.5	Bemerkungen:
Test Titel:	Kommunikation Path-Planning mit der EPD (senden von Daten an die EPD)	
Beschreibung:	Der berechnete valide Pfad wird an die EPD geschickt, dort gespeichert und angezeigt.	
Anforderung	F-CA-PP-1 F-CA-PP-2 F-CA-PP-3 F-CA-PP-9	
Eingehende-Daten:	- Berechneter Pfad von Path-Planning (variable Anzahl) - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System	
Testschritte:	TCS-01: Starten des Autopiloten mit einer gültigen Route	

	TCS-02: Überprüfung der Verbindung zwischen Path-Planning und EPD durch Log-Ausgaben	
Erwartetes Ergebnis:	- Die Path-Planning-Komponente sendet den aktuellen Pfad an die EPD (optional: Debug-Ausgaben, wie Goodwin-Shapes / statische Hindernisse)	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-12	Bemerkungen:
Test Titel:	Berücksichtigung eines dynamischen und statischen Hindernisses bei der Pfadberechnung	
Beschreibung:	Ausweichen eines dynamischen Objektes (Schiff) unter Berücksichtigung eines nahen statischen Hindernisses (Untiefe).	
Anforderung	F-CA-PP-5	
Eingehende-Daten:	- Szenario-Route im RTZ-Format von der EPD (Szenario 6) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System	

	- Definierte Optimalroute (Sollpfad)	
Testschritte:	TCS-01: Optimierte Route für Szenario 6 in der EPD laden und zusammen mit den Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 durch Klick auf die Schaltfläche „Route starten“ an das Path-Planning senden. TCS-02: Den berechneten Pfad der Path-Planning überwachen TCS-03: In den Zuse-Tab in MATE-Control wechseln, dort die optimierte Route für Szenario 6 auswählen und Validierung durch Klick auf „Record“ starten (Polygone werden in EPD gezeichnet)	
Erwartetes Ergebnis:	Das System weicht dem dynamischen Objekt aus ohne dabei mit den statischen Hindernissen zu kollidieren	

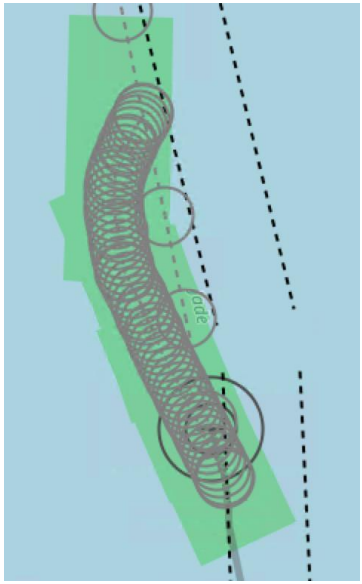
		
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-13	Bemerkungen:
Test Titel:	Path-Planning Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes	
Beschreibung:	Ausweichen eines Objektes unter Berücksichtigung der physikalischen Eigenschaften für das im Test eingesetzte Schiff (ZUSE).	
Anforderung	F-CA-PP-6	
Eingehende-Daten:	- Szenario-Routen im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System	

Testschritte:	TCS-01: Optimierte Route für Basis-Test-1 in der EPD laden und zusammen mit den Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 durch Klick auf die Schaltfläche „Route starten“ an das Path-Planning senden. TCS-02: Den berechneten Pfad der Path-Planning überwachen (werden in EPD gezeichnet TCS-03: Statistische Auswertung der berechneten Pfade auf Einhaltung der Berücksichtigung der physikalischen Eigenschaften des im Test eingesetzten Schiffes (Turning Angle darf nicht 20 Grad bei jedem Wegpunktübergang)	
Erwartetes Ergebnis:	- Der berechnete Pfad steht nicht im Konflikt mit den physikalischen Eigenschaften (Wendewinkel der Zuse) des im Test eingesetzten Schiffs, d.h. der Wendewinkel beträgt maximal 30 Grad). Verifizierbar, dadurch, dass der Controller dem berechneten	

	Pfad, bis auf einen Threshold, folgen kann.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

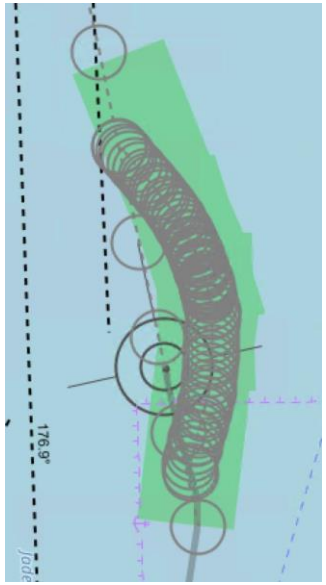
ID:	TC-14.1	Bemerkungen:
Test Titel:	Path-Planning Überholen	
Beschreibung:	Überholen eines dynamischen Objektes (Schiff) unter Berücksichtigung der COLREG Regel 13.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System - Definierte Optimalroute (Sollpfad)	

Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route (Szenario 4) an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen (Überholen, COLREG 13) TCS-03: Validierung durch Einhaltung des Threshold mit MateControl	
Erwartetes Ergebnis:	Die Zuse überholt das Fremdschiff auf der linken oder rechten Seite mit einem ausreichenden Mindestabstand von 100 Metern und folgt nach dem Überholmanöver wieder der zuvor definierten Route. 	

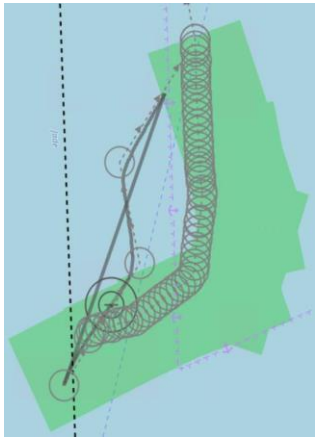
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-14.2	Bemerkungen:
Test Titel:	Path-Planning entgegenkommend	
Beschreibung:	Ausweichen eines entgegenkommenden dynamischen Objektes (Schiff) unter Berücksichtigung der COLREGs Regeln 14, 16, 17.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0	

	• Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System - Definierte Optimalroute (Sollpfad)	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route (Szenario 3) an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen TCS-03: Validierung durch Einhaltung des Threshold mit MateControl	
Erwartetes Ergebnis:	Die Zuse weicht dem entgegenkommenden Fremdschiff nach rechts hin mit einem ausreichenden Mindestabstand von 100 Metern aus und folgt nach dem	

	Ausweichmanöver wieder der zuvor definierten Route. Validierung durch Einhaltung des Threshold mit MateControl mit zuvor definierten Route. 	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

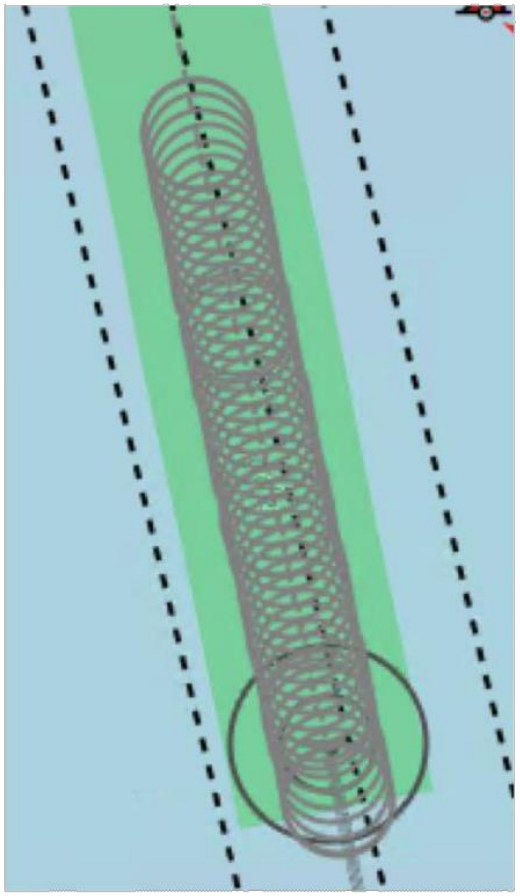
ID:	TC-14.3	Bemerkungen:
Test Titel:	Path-Planning Kreuzen von rechts	
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von rechts gekreuzt unter Berücksichtigung der COLREGs Regeln 15, 16, 17.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente	

	- Position, Heading (NMEA0183) vom Distribution System - Definierte Optimalroute (Sollpfad)	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Das Ausweichen des Schiffes überwachen (Kreuzen, COLREGs 15, 16) TCS-03: Validierung durch Einhaltung des Threshold mit MateControl	
Erwartetes Ergebnis:	Die Zuse weicht dem Fremdschiff nach links hin mit einem ausreichenden Mindestabstand von 100 Metern aus und folgt nach dem Ausweichmanöver wieder der zuvor definierten Route. 	

Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-14.4.1	Bemerkungen:
Test Titel:	Path-Planning Kreuzen von links (Kurs beibehalten)	
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von links gekreuzt unter Berücksichtigung der COLREGs Regeln 15, 16, 17.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5	

	- JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System - Definierte Optimalroute (Sollpfad)	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route (Szenario 5) an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Überwachung des Kreuz-Manövers TCS-03: Validierung durch Einhaltung des Threshold mit MateControl	
Erwartetes Ergebnis:	Die Zuse verfolgt weiterhin den vorberechneten validen Pfad innerhalb der definierten Thresholds.	

		
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

ID:	TC-14.4.2	Bemerkungen:
Test Titel:	Path-Planning Kreuzen von links (Ausweichen)	
Beschreibung:	Das zu testende Schiff wird von einem anderen Schiff von links gekreuzt, das andere Schiff verletzt die COLREGs Regeln und weicht nicht aus, daraufhin muss das zu testende Schiff zwangsweise Ausweichen unter Berücksichtigung der COLREGs Regeln 15, 16, 17.	
Anforderung	F-CA-PP-8	
Eingehende-Daten:	- Route im RTZ-Format von der EPD - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von	

	erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System - Definierte Optimalroute (Sollpfad)	
Testschritte:	TCS-01: Senden der ausgewählten und optimierten Route an das Path-Planning durch Auswählen der Route und Klicken auf “Route starten” TCS-02: Überwachung des erzwungenen Ausweichens während des Kreuz-Manövers TCS-03: Validierung durch Einhaltung des Threshold mit MateControl	
Erwartetes Ergebnis:	Die Zuse weicht dem Fremdschiff nach links oder rechts hin mit einem ausreichenden Mindestabstand von 100 Metern aus und folgt nach dem Ausweichmanöver wieder der zuvor definierten Route.	

		
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Szenariobasiertes Blackbox-Testing	

Fehlerinjektion:

ID:	TC-FI-1	Bemerkungen:
Test Titel:	ownPosition-Injection	
Beschreibung:	Ein anderes, bereits vorher über AIS identifiziertes Schiff wechselt die Position auf die Position des eigenen Schiffes.	
Eingehende-Daten:	- Route im RTZ-Format von der EPD (Szenario 1) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System	

Testschritte:	TCS-01: optimierte Route in der EPD auswählen (Szenario 1). TCS-02: Automatisierte Fahrt starten, in dem die optimierte Route an das Path-Planning gesendet wird. TCS-03: In den Error-Injection-Tab in MATE-Control wechseln, das simulierte Schiff auswählen, den Haken bei „ownPosition“ setzen und die Fehlerinjektion mit einem Klick auf „inject“ ausführen.	
Erwartetes Ergebnis:	Nachdem die Injektion durchgeführt wurde, berechnet das Path-Planning einen Pfad, der dazu führt, dass der Controller die Fahrt abbremst und die Fahrt gestoppt wird.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Fehlertest	

ID:	TC-FI-2	Bemerkungen:
Test Titel:	Ungültiger Ruderwinkel	
Beschreibung:	Der Controller empfängt einen außerhalb der spezifizierten Grenzen definierten Ruderwinkel.	
Eingehende-Daten:	- Ruderwinkel per NMEA 2000 (an Controller)	
Testschritte:	TCS-01: optimierte Route in der EPD auswählen (Szenario 1). TCS-02: Automatisierte Fahrt starten, in dem die optimierte Route an das Path-Planning gesendet wird. TCS-03: In den Error-Injection-Tab in MATE-Control wechseln, das simulierte Schiff auswählen, den Haken bei „rudder angle“ setzen und die Fehlerinjektion mit einem Klick auf „inject“ ausführen. TCS-04: Überwachung des Systems mit MATE-Control	
Erwartetes Ergebnis:	Das System verhält sich unverändert.	
Eingetretenes Ergebnis:		

Erfolg/Misserfolg:		
Testmethode:	Fehlertest	

ID:	TC-FI-3	Bemerkungen:
Test Titel:	Schiff verschwindet (Afrika-Injektion)	
Beschreibung:	Ein existierendes (real oder simuliert), über AIS erkanntes Schiff verschwindet von seiner bekannten Position.	
Eingehende-Daten:	- Route im RTZ-Format von der EPD (Szenario 1) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition)	

	- JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System TCS-04: Überwachung des Systems mit MATE-Control	
Testschritte:	TCS-01: optimierte Route in der EPD auswählen (Szenario 1). TCS-02: Automatisierte Fahrt starten, in dem die optimierte Route an das Path-Planning gesendet wird. TCS-03: In den Error-Injection-Tab in MATE-Control wechseln, das simulierte Schiff auswählen, den Haken bei „remove“ setzen und die Fehlerinjektion mit einem Klick auf „inject“ ausführen. TCS-04: Überwachung des Systems mit MATE-Control	
Erwartetes Ergebnis:	Das über AIS erkannte Schiff wird noch weitere ca. 30 Sekunden im Speicher gehalten und in dem Zeitraum auch an die EPD und an das PP gesendet. Nach ca. 30 Sekunden wird das Schiff in der DOI gelöscht und nicht mehr versendet.	

	In dem EPD-DOI-Layer ist die Repräsentation des Schiffes mit der Bezeichnung FAIS123456789 noch ca. 30 Sekunden nach Erhalt der letzten Nachricht dieses Schiffes zu sehen.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Fehlertest	

ID:	TC-FI-4	Bemerkungen:
Test Titel:	Schiff taucht auf (Afrika-Injektion)	
Beschreibung:	Ein zuvor verschwundenes (simuliert), über AIS erkanntes Schiff erscheint wieder auf seiner echten Position.	

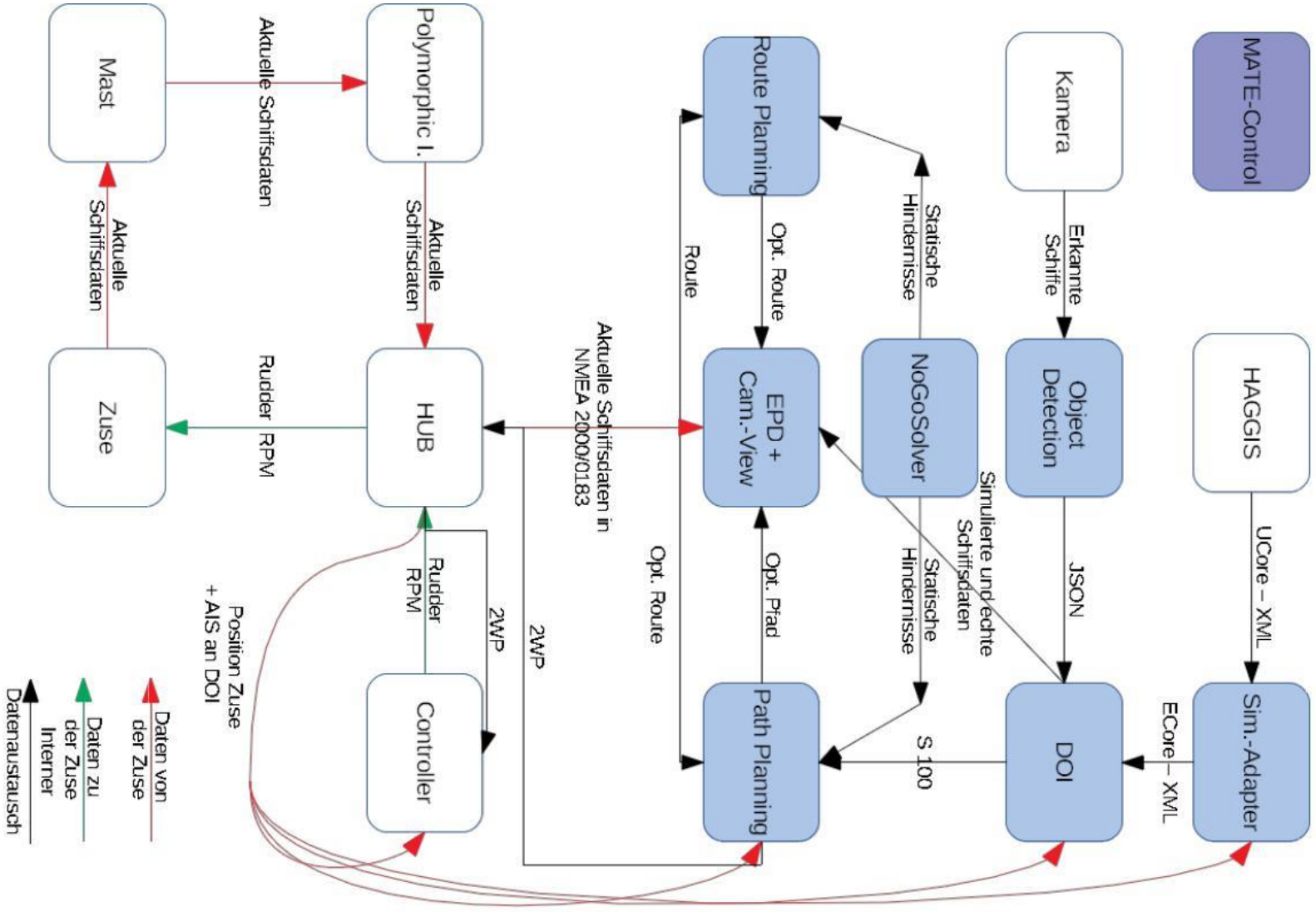
Eingehende-Daten:	- Route im RTZ-Format von der EPD (Szenario 1) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System TCS-04: Überwachung des Systems mit MATE-Control	
Testschritte:	TCS-01: optimierte Route in der EPD auswählen (Szenario 1). TCS-02: Automatisierte Fahrt starten, in dem die optimierte Route an das Path-Planning gesendet wird. TCS-03: In den Error-Injection-Tab in MATE-Control wechseln, das simulierte Schiff auswählen, den Haken bei „remove“ entfernen und die	

	Fehlerinjektion mit einem Klick auf „inject“ ausführen. TCS-04: Überwachung des Systems mit MATE-Control	
Erwartetes Ergebnis:	Das erkannte Schiff wird wieder erfasst und an die EPD und an das PP gesendet. In dem EPD-DOI-Layer ist das Schiff mit der Bezeichnung FAIS123456789 zu sehen.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Fehlertest	

ID:	TC-FI-05	Bemerkungen:
Test Titel:	Wechsel der MMSI	
Beschreibung:	Ein mit AIS und oder Radar identifiziertes Schiff in der Umgebung des eigenen Schiffs wechselt seine MMSI.	
Eingehende-Daten:	- Route im RTZ-Format von der EPD (Szenario 1) - Schiffsinformationen: • Länge: 8.0 • Breite: 2.5 • Höhe 2.5 • Tiefgang: 0.5 - JSON-Objekte (z. B. Polygone, Points) vom NoGoSolver (2000m Radius ausgehend von der eigenen Schiffsposition) - JSON-Objekte (Koordinaten, COG, SOG, Länge und Breite von erkannten Schiffen) von der DOI-Komponente - Position, Heading (NMEA0183) vom Distribution System TCS-04: Überwachung des Systems mit MATE-Control	

Testschritte:	TCS-01: optimierte Route in der EPD auswählen (Szenario 1). TCS-02: Automatisierte Fahrt starten, in dem die optimierte Route an das Path-Planning gesendet wird. TCS-03: In den Error-Injection-Tab in MATE-Control wechseln, das simulierte Schiff auswählen, den Haken bei „MMSI“ setzen und die Fehlerinjektion mit einem Klick auf „inject“ ausführen. TCS-04: Überwachung des Systems mit MATE-Control	
Erwartetes Ergebnis:	Für jede neue MMSI wird ein Schiff in der DOI gespeichert. Diese Schiffe werden an die EPD und an das PP gesendet. In dem EPD-DOI-Layer werden somit mehrere Repräsentationen des Schiffes angezeigt.	
Eingetretenes Ergebnis:		
Erfolg/Misserfolg:		
Testmethode:	Fehlertest	

Gesamtübersicht:



Anlage 1: optimierte RTZ-Route

WP1:
lat: 53.51765489239272
lon: 8.17657470703125
WP2:
lat: 53.51941397091744
lon: 8.175958462627543
WP3:
lat: 53.52117304574505
lon: 8.175342167169969
WP4:
lat: 53.52293211687508
lon: 8.174725820651123
WP5:
lat: 53.524691184307095
lon: 8.174109423063586
WP6:
lat: 53.526817548437876
lon: 8.176445103099207
WP7:
lat: 53.52857662358174
lon: 8.175828720333746
WP8:
lat: 53.53033569502767
lon: 8.175212286487849
WP9:
lat: 53.53246203559427
lon: 8.177548332323932
WP10:
lat: 53.53422111475336
lon: 8.176931913315126
WP11:
lat: 53.53634742033682
lon: 8.17926825913435
WP12:
lat: 53.53810650720982
lon: 8.178651854982
WP13:
lat: 53.5398655903855
lon: 8.17803539973313
WP14:
lat: 53.541624669863445
lon: 8.177418893380318
WP15:
lat: 53.543750963280054
lon: 8.17975567121135
WP16:
lat: 53.54551005047374
lon: 8.1791391797257
WP17:
lat: 53.547269133969756
lon: 8.178522637124349
WP18:
lat: 53.54902821376765
lon: 8.177906043399876
WP19:
lat: 53.55078728986697
lon: 8.177289398544849
WP20:
lat: 53.552546362267265
lon: 8.17667270255184
WP21:
lat: 53.55430543096807
lon: 8.176055955413423
WP22:
lat: 53.55606449596894
lon: 8.175439157122161
WP23:
lat: 53.55782355726942
lon: 8.174822307670626
WP24:
lat: 53.55958261486906
lon: 8.17420540705138
WP25:
lat: 53.561341668767376
lon: 8.173588455256988
WP26:
lat: 53.563100718963945
lon: 8.172971452280017
WP27:
lat: 53.5648597654583
lon: 8.172354398113024
WP28:
lat: 53.566618808249984
lon: 8.17173729274857
WP29:
lat: 53.56837784733853
lon: 8.171120136179217
WP30:
lat: 53.57013688272353
lon: 8.17050292839752
WP31:
lat: 53.57189591440448
lon: 8.169885669396034
WP32:
lat: 53.573654942380934
lon: 8.169268359167317
WP33:
lat: 53.57541396665244
lon: 8.168650997703919
WP34:
lat: 53.57717298721855
lon: 8.168033584998396
WP35:
lat: 53.57893200407879
lon: 8.167416121043292
WP36:
lat: 53.58069101723272
lon: 8.166798605831163
WP37:
lat: 53.582450026679886
lon: 8.166181039354553
WP38:
lat: 53.584209032419814
lon: 8.16556342160601
WP39:
lat: 53.58596803445207
lon: 8.16494575257808
WP40:
lat: 53.58772703277619
lon: 8.164328032263299
WP41:
lat: 53.58948602739171
lon: 8.16371026065422
WP42:
lat: 53.59124501829817
lon: 8.163092437743378
WP43:
lat: 53.59300400549512
lon: 8.162474563523311
WP44:
lat: 53.59476298898212
lon: 8.161856637986562
WP45:
lat: 53.59652196875869
lon: 8.161238661125662
WP46:
lat: 53.59828094482437
lon: 8.160620632933151
WP47:
lat: 53.60003991717871
lon: 8.160002553401561
WP48:
lat: 53.60179888582128
lon: 8.159384422523424
WP49:
lat: 53.603557850751585
lon: 8.158766240291273
WP50:
lat: 53.60531681196919
lon: 8.158148006697635
WP51:
lat: 53.60707576947363
lon: 8.157529721735038
WP52:
lat: 53.60883472326444
lon: 8.156911385396013
WP53:
lat: 53.61059367334118
lon: 8.156292997673082
WP54:
lat: 53.61235261970337
lon: 8.155674558558768
WP55:
lat: 53.61411156235057
lon: 8.155056068045596
WP56:
lat: 53.615870501282316
lon: 8.154437526126088
WP57:
lat: 53.61762943649817
lon: 8.153818932792761
WP58:
lat: 53.619388367997644
lon: 8.153200288038136
WP59:
lat: 53.621147295780275
lon: 8.152581591854732
WP60:
lat: 53.62290621984563
lon: 8.15196284423506
WP61:
lat: 53.62466514019325
lon: 8.151344045171633
WP62:
lat: 53.62642405682268
lon: 8.15072519465697
WP63:
lat: 53.62818296973344
lon: 8.15010629268358
WP64:
lat: 53.62994187892507
lon: 8.149487339243972
WP65:
lat: 53.631700784397125
lon: 8.148868334330656
WP66:
lat: 53.63345968614916
lon: 8.148249277936138
WP67:
lat: 53.635218584180 69
lon: 8.147630170052926
WP68:
lat: 53.636977478491275
lon: 8.147011010673522
WP69:
lat: 53.638736369080455
lon: 8.14639179979043
WP70:
lat: 53.640495255947755
lon: 8.145772537396152
WP71:
lat: 53.64225413909272
lon: 8.14515322348319
WP72:
lat: 53.64401301851491
lon: 8.144533858044039
WP73:
lat: 53.64577189421386
lon: 8.143914441071196
WP74:
lat: 53.647530766189085
lon: 8.143294972557163
WP75:
lat: 53.649289634440144
lon: 8.142675452494432
WP76:
lat: 53.651048498966574
lon: 8.142055880875498
WP77:
lat: 53.6524392023931
lon: 8.138475642429457
WP78:
lat: 53.65419804426581
lon: 8.137855849923817
WP79:
lat: 53.65595688241105
lon: 8.13723600582885
WP80:
lat: 53.657715716828356
lon: 8.136616110137037
WP81:
lat: 53.65910625617243
lon: 8.13303512809199
WP82:
lat: 53.660865067920284
lon: 8.132415011396885
WP83:
lat: 53.662255481013496
lon: 8.128833624346225
WP84:
lat: 53.66401427008092
lon: 8.128213286582469
WP85:
lat: 53.66540455689565
lon: 8.124631494455508
WP86:
lat: 53.667163323271666
lon: 8.124010935557715