CARL VON OSSIETZKY universität OLDENBURG

Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften

Department für Informatik

Dissertation

Simulative Überprüfung von Sensordatenverarbeitungssystemen

Dissertation zur Erlangung des Grades eines
Doktor der Ingenieurswissenschaften

vorgelegt von
Dipl.-Inform. Sören Schweigert

Gutachter
Prof. Dr.-Ing. Axel Hahn
Prof. Dr. Martin Fränzle

Tag der Disputation:
10 Juni 2016

Zusammenfassung

Sensordatenverarbeitende Systeme, lassen sich aus dem heutigen Alltag kaum noch wegdenken. Sie werden in nahezu allen Bereichen des privaten und beruflichen Lebens eingesetzt. Eine besondere Herausforderung für die Entwicklung und Qualitätssicherung stellen sensordatenverarbeitende Systeme dar, die im Rahmen von Assistenz- und Automatisierungssystemen eine Umfelderkennung vornehmen. Insbesondere dann, wenn auf Grundlage der Sensordatenverarbeitung sicherheitskritische Entscheidungen getroffen werden, muss man sich auf die Korrektheit der ermittelten Daten verlassen, bzw. die Verlässlichkeit der Ergebnisse einschätzen können.
Herkömmliche Methoden zur Qualitätssicherung, wie beispielsweise formale Sicherheitsuntersuchungen oder vollständige funktionale Tests, lassen sich aufgrund der hohen Komplexität der untersuchten Systeme sowie ihrer Umgebungen nicht oder nur mit sehr hohem Aufwand realisieren.

Um dennoch Aussagen über die Güte eines sensordatenverarbeitenden Systems machen zu können, wird in der vorliegenden Arbeit eine werkzeuggestützte Methodik zur simulativen Überprüfung von sensordatenverarbeitenden Systemen vorgestellt. Die vorgestellte Methodik kann dabei sowohl während der Entwicklung als Entwicklungsunterstützung, als auch zu einer nachträglichen Überprüfung von sensordatenverarbeitenden Systemen angewendet werden.
Zu diesem Zweck wird die reale, von dem zu überprüfenden System beobachtete Umgebung durch eine virtuelle Umgebung ersetzt. Der Zustand dieser virtuellen Umgebung kann dann genutzt werden, um zum Einen Sensorbeobachtungen zu erzeugen, die als Eingabe für das sensordatenverarbeitende System verwendet werden, zum Anderen lässt sich mit Hilfe des aktuellen Zustandes der simulierten Umgebung, ein Abgleich mit den Ergebnissen der Sensordatenverarbeitung herstellen.
Zum Einsatz kommt dabei ein modellbasierter Ansatz, bei dem zunächst die Sensoren inklusive ihres Fehlverhaltens in Bezug auf ihre Messungenauigkeiten und Sensorfehler beschrieben und anschließend durch ein komponentenbasiertes Simulationssystem simuliert werden. Weiterhin kann das Simulationssystem in eine bestehende Ko-Simulationsumgebung eingebunden werden, um ein realistisches Verhalten der beobachteten Objekte zu simulieren.

Der Ansatz wurde prototypisch implementiert und anhand von zwei Anwendungsfällen aus der maritimen Domäne evaluiert und demonstriert.

Abstract

Sensor data processing systems have become hardly indispensable in almost all areas of our private and professional life. A special challenge for the development and quality assurance of sensor data processing systems are those systems that perform an environment monitoring as part of an autonomous or assistance system. Especially when safety-critical decisions are made based on the results of the sensor data processing. In this case we have to rely on the correctness of the data processing or at least be able to estimate the reliability of the sensor processing results.
Conventional methods for quality assurance, such as formal safety analyses or a complete functional testing can hardly be realized or require a very high expenditure.

However to still make statements about the quality of the sensor data processing, a tool-based methodology for simulative validation of sensor data processing systems, is presented within this work. The presented methodology can then be used both during development, as development assistance as well as for a subsequent validation of existing sensor data processing systems.
For this purpose the system to be checked will be placed in a virtual environment, replacing the real environment in which it normally operates. The current state of this virtual environment can then be used in order to produce sensor observations as input for the sensor processing system. On the other hand the same state could also be used to perform a comparison between the results of the sensor processing system and the virtual environment.
A model-based approach is used, to describe the sensors including their abnormal behavior in respect to their measurement uncertainties and sensor errors. These modeled sensors can be simulated with the help of a developed component-based simulation system which can also be connected to other simulation systems into a co-simulation environment.

The proposed tool-based methodology has been prototypically implemented, evaluated and demonstrated with two maritime use cases.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Sensordatenverarbeitende Systeme spielen in vielen Bereichen des täglichen Lebens eine große Rolle. Sie werden in den verschiedensten Bereichen eingesetzt, sei es zur erleichterten Bedienung von Smartphones oder zur Steuerung von autonomen Systemen. Dabei werden unterschiedliche Qualitätsanforderungen an diese Systeme gestellt.
Während sich ein Fehler innerhalb der Sensordatenverarbeitung eines Smartphones ggf. auf die Akzeptanz eines solchen Systems auswirkt, sind keine Personen- oder Materialschäden zu erwarten. Anders sieht dies bei der Verwendung von sensordatenverarbeitenden Systemen in Assistenz- und Automatisierungssystemen aus, wie sie exemplarisch in Abbildung 1.1 dargestellt sind.

Abbildung 1.1: Exemplarische Auswahl von assistenz- (oben) und automatisierten Systemen (unten); Bildquellen[1]

[1]Die obere rechte Abbildung wurde aus der Arbeit Simulation Augmented Manoeuvring Design and Monitoring–a New Method for Advanced Ship Handling [BKG+14] entnommen, Das Bild unten links entstammt der Präsentation [TE14].

In solchen Systemen wird die Sensordatenverarbeitung verwendet, um das Umfeld in dem sich das System befindet wahrzunehmen, und aufbauend darauf teils sicherheitsrelevante Entscheidungen zu treffen. Während die Umfelderkennung im Zusammenhang von Assistenzsystemen noch durch einen menschlichen Operator bestätigt wird, stellen die Sensorwahrnehmungen bei autonomen Systemen häufig die einzige Informationsquelle über die Umgebung dar. Dementsprechend können Fehler, die sich aus einer falschen Interpretation von Sensordaten ergeben, nicht oder nur zu einem kleinen Teil in den nachfolgenden Systemen ausgeglichen werden.

In dieser Arbeit werden vornehmlich sensordatenverarbeitende Systeme betrachtet, die für die Umfelderkennung von autonomen oder teil-autonomen Fahrzeugen benötigt werden. Dabei findet zunächst keine direkte Unterscheidung zwischen verschiedenen Domänen statt, allerdings werden Beispiele aus der maritimen Domäne zur Verdeutlichung verwendet.
Ziel der betrachteten sensordatenverarbeitenden Systeme ist es, ein konsistentes Umgebungsmodell auf Grundlage von Sensormesswerten aufzubauen. Dieses Umgebungsmodell soll dabei die relevanten Aspekte der beobachteten Umgebung so genau wie möglich abbilden. Dies geschieht wiederum vornehmlich aus Sicherheitsüberlegungen, bei denen eine genaue Abbildung der Umgebung für die Kollisionsvermeidung unerlässlich ist, wie es zum Beispiel in dem Forschungsprojekt SaLsA [RN11] der Fall ist.

1.1 Herausforderungen bei der Entwicklung von sensordatenverarbeitenden Systemen

Bei der Entwicklung von sensordatenverarbeitenden Systemen, ergeben sich neben den inhaltlichen Schwierigkeiten weitere Herausforderungen.

1. Die Qualität der entwickelten Systeme in Bezug auf die Sicherheit und Effizienz muss sichergestellt und überprüft werden können.
2. Die Aufwände zum bereitstellen von Testdaten, die für die Entwicklung und Überprüfung genutzt werden können, sollten so gering wie möglich gehalten werden um die verfügbaren Ressourcen verstärkt in die Entwicklung und Qualitätssicherung des eigentlichen, sensordatenverarbeitenden, Systems investieren zu können.

Gerade der erste Punkt bekommt eine besondere Bedeutung, wenn das sensordatenverarbeitende System in seinem Systemkontext betrachtet wird. Dieser lässt sich in den

betrachteten Robotik-, Automotive und maritimen Domänen i.d.R. gut durch das sogenannte Sense-Think-Act Paradigma (vgl. Abbildung 1.2) beschreiben [Sie03].

Bei diesem Paradigma werden zunächst mit Hilfe von Sensoren Messwerte aufgenommen und zu höherwertigen Daten verdichtet (Sense). Dabei wird z.B. aus Entfernungsmessungen eines Laserscanners sowie Aufnahmen einer Kamera die Information abgeleitet, dass sich vor dem eigenen Fahrzeug ein weiteres Fahrzeug befindet. Durch eine zeitliche Verknüpfung der Ergebnisse lassen sich zudem ggf. die Geschwindigkeit des vorausfahrenden Fahrzeug bestimmen bzw. die relative Geschwindigkeit zwischen dem eigenen und dem vorausfahrenden Fahrzeug.

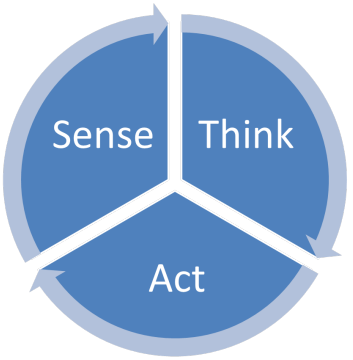

Abbildung 1.2: Sense-Think-Act Zyklus als grundlegender Aufbau von (teil-) autonomen Systemen.

Ausgehend von dem ermittelten Lagebild werden Entscheidungen getroffen, wie sich das System in der nahen Zukunft verhalten soll (Think). Das angefangene Beispiel fortführend würde an dieser Stelle ggf. entschieden werden, dass die eigene Geschwindigkeit reduziert werden muss, wenn die relative Geschwindigkeit zwischen dem eigenem und vorausfahrendem Fahrzeug zu hoch ist sowie der Abstand der beiden Fahrzeuge einen gewissen Schwellenwert unterschreitet.
Das Ergebnis des “Think“ Schrittes wird im letzten Teil des Paradigmas, durch Aktuatoren umgesetzt (Act), indem z.B. weniger Gas gegeben, bzw. die Bremse betätigt wird.

Die Anwendung dieses häufig vorzufindenden Paradigmas führt zu einer weiteren Herausforderung bei der Entwicklung von sensordatenverarbeitenden Systemen, welche in den Naturwissenschaften, insbesondere der Physik, als Fehlerfortpflanzung[2] bekannt ist. Dabei handelt es sich um den in [CG07] beschriebenen Effekt, dass sich Fehler über benachbarte Systeme hinweg ausbreiten können. Auf das Sense-Think-Akt Paradigma übertragen, bedeutet dies zum Beispiel das übersehene Hindernisse in der Sensorwahrnehmung durch die nachfolgenden Teilsysteme nicht weiter behandelt werden können, da sie für diese Systeme nicht existieren.
Generell kann davon ausgegangen werden, dass frühzeitig auftretende Fehler, welche

[2] In der Physik und insbesondere in der DIN 1319 wird der Begriff der Fehlerfortpflanzung im Zusammenhang mit Messungenauigkeiten verwendet. An dieser Stelle wird eine etwas weiter ausgelegte Interpretation des Begriffes verwendet, die die Propagierung von Fehlern über verschiedene Systeme hinweg behandelt.

nicht erkannt und behandelt werden, sich über die verschiedenen Teilsysteme ausbreiten. Dabei erhöht sich die potenzielle Auswirkung eines Fehlers mit zunehmender Entfernung zu seinem Ursprung. Analog steigt die Schwierigkeit, die Ursache eines Fehlers zu identifizieren mit der Entfernung zwischen seinem Ursprung und seiner Entdeckung.

1.2 Problemstellung und Forschungsfrage

Ausgehend von den beschriebenen Herausforderungen bei der Entwicklung von neuen sensordatenverarbeitenden Systemen lässt sich folgende Forschungsfrage formulieren:

Wie kann die korrekte Funktionsweise eines sensordatenverarbeitenden Systems überprüft und gleichzeitig die Entwicklung der Sensordatenverarbeitung in ihrem Systemkontext unterstützt werden?

Diese Fragestellung setzt sich aus zwei Aspekten zusammen: Zum einen der Frage nach der Qualität der Sensordatenverarbeitung in Bezug auf die Sicherheit und Effizienz des entwickelten Gesamtsystems.
Sowie der Frage, wie durch den Einsatz der vorgestellten Methode der Entwicklungsprozess der Sensordatenverarbeitung aber auch der daran anschließenden Systeme (Think & Act) vereinfacht bzw. beschleunigt werden kann.

Die grundlegende Idee dieser Arbeit basiert dabei auf einer Kombination von konstruktiven und analytischen Ansätzen zur Qualitätssicherung von sensordatenverarbeitenden Systemen, die in den Entwicklungsprozess von sensordatenverarbeitenden Systemen eingegliedert werden können.

Die Unterstützung des Entwicklungsprozesses erfolgt ebenfalls durch zwei Ansätze. Zum einen unterstützt das Vorgehen die Spezifikation von Sensoren und sensordatenverarbeitenden Systemen, indem diese strukturiert beschrieben werden. Dabei ist neben der Beschreibung des normativen Verhaltens von Sensoren auch eine Spezifikation ihres Fehlverhaltens in Form von Messungenauigkeiten und Sensorfehlern vorgesehen (vgl. Abschnitt 2.2.2).
Diese strukturierte Beschreibung kann im weiteren Verlauf nicht nur zur Dokumentation, sondern auch für die Durchführung von analytischen Qualitätssicherungsverfahren, wie beispielsweise Unit-Tests oder Integrationstests verwendet werden.
Zum anderen können die so spezifizierten Sensoren bereits in frühen Phasen der Entwicklung simuliert werden und den anschließenden Systemen des Sense-Think-Act Paradigmas, Ergebnisse zur Verfügung stellen. Dies ermöglicht eine parallele Entwicklung von

Sensordatenverarbeitung (Sense) und Entscheidungslogik (Think), indem beispielsweise Teile der noch zu entwickelnden Sensordatenverarbeitung simuliert werden.

Aufbauend auf der zuvor durchgeführten Spezifikation der einzelnen Sensoren sowie ihres normativen und ggf. Fehlverhalten, lässt sich eine automatisierbare Überprüfung von sensordatenverarbeitenden Systemen umsetzen, was dem analytischen Ansatz zur Qualitätssicherung entspricht. Hierzu wird das Entwicklungssystem in eine bestehende Ko-Simulationsumgebung eingebunden, mit der ein realistisches Verhalten von komplexen Umgebungen abgebildet werden kann. Die dabei resultierende Umgebung wird für das sensordatenverarbeitende System aufbereitet und in Form von Sensorbeobachtungen zur Verfügung gestellt. Anschließend kann das durch die Sensordatenverarbeitung hergeleitete Umgebungsmodell mit dem durch die Ko-Simulation abgebildetem Modell verglichen werden.

1.2.1 Qualitätskriterien von sensordatenverarbeitenden Systemen

In der oben genannten Fragestellung, wird von der korrekten Funktionsweise der Sensordatenverarbeitung gesprochen, wobei es sich dabei um eine bewusst unpräzise Formulierung handelt. Dies liegt darin begründet, dass der hier vorgestellte Ansatz soweit möglich, unabhängig von der zu testenden Anwendung entwickelt werden soll.
Dennoch lassen sich allgemeine Qualitätskriterien für sensordatenverarbeitende Systeme bestimmen, die im Folgenden diskutiert werden und die Grundlage für Anforderungen an die entwickelte Methode darstellen.

Die hier vorgestellten Kriterien an sensordatenverarbeitende Systeme können als Fragen interpretiert werden, die ein Entwickler von sensordatenverarbeitenden Systemen, an die im Folgenden vorgestellte Methode und die daraus resultierende Toolunterstützung stellen kann.

SDV1 - Korrektheit des erkannten Umgebungsmodells.

Die Korrektheit des erkannten Umgebungsmodells entspricht im Wesentlichen der Abwesenheit von False positive und False negative Ergebnissen innerhalb des Umgebungsmodells.

Dabei können sich die False positive und False negative sowohl auf die Existenz von Objekten beziehen, wie beispielsweise das Nichterkennen eines Hindernisses im Fahrweg eines autonomen Fahrzeuges, als auch auf die richtige Klassifikation von erkannten Objekten.

In der Realität ist es i.d.R. nicht möglich ein sensordatenverarbeitendes System zu

erstellen, welches komplett ohne Fehler arbeitet, Entsprechend dem oben genannten Ziel, die Methode unabhängig von einer konkreten Anwendung zu entwickeln, obliegt es dem Anwender der Methode, regeln für das Auftreten von False positive bzw. False negative zu spezifizieren.

SDV2 - Fehlertoleranz der Sensordatenverarbeitung.

Ein fehlertolerantes System (z.B. die Sensordatenverarbeitung) sollte in der Lage sein, den (teilweisen) Ausfall von Subsystemen erkennen und ggf. behandeln zu können.

Im Rahmen dieser Arbeit wird die Sensordatenverarbeitung als ein Blackbox System angesehen, dessen interner Aufbau nicht bekannt ist. Somit beschränkt sich die zu überprüfende Fehlertoleranz aus Sicht der Methode auf das Erkennen und ggf. Behandeln von Fehlern innerhalb der verwendeten Sensoren und nicht der einzelnen Komponenten des zu testenden Systems.

Ähnlich wie zuvor bei der Korrektheit des ermittelten Umgebungsmodells obliegt es dem Anwender der Methode, konkrete Gütemerkmale für die Fehlertoleranz festzulegen. So kann je nach untersuchter Anwendung das Erkennen eines Sensorfehlers und die daraus resultierende Änderung der Ergebnisqualität bereits als ausreichend angesehen werden. Andere Anwendungen benötigen ggf. eine stärkere Fehlertoleranz, bei der sichergestellt wird, dass dennoch eine entsprechende Aussagekraft des erkannten Umgebungsmodells vorhanden ist.

SDV3 - Robustheit der Sensordatenverarbeitung.

Unter Robustheit der Sensordatenverarbeitung wird in diesem Zusammenhang die Robustheit gegenüber Messunsicherheiten der Sensoren verstanden .

Reale Sensoren liefern in der Regel keine exakten Messwerte, sondern unterliegen immer einer Messunsicherheit (vgl. auch Abschnitt 2.3), welche sich i.d.R. auf die Genauigkeit des erkannten Umgebungsmodells auswirkt.

Auch hier gilt: Welche Genauigkeit der Ergebnisse als akzeptabel angesehen wird, hängt von der jeweilig untersuchten Anwendung ab. So kann beispielsweise für die Navigation eines (autonomen) Schiffes auf offener See eine Positionsungenauigkeit von bis zu 10 Metern ausreichend sein. In stark befahrenen Gewässern oder bei der Navigation innerhalb eines Hafens können dagegen Abweichungen von mehr als einem Meter nicht mehr akzeptiert werden. [IMO01]

SDV4 - Antwortzeit der Sensordatenverarbeitung.

Unter Antwortzeit der Sensordatenverarbeitung wird die Zeit verstanden, die die Sensordatenverarbeitung benötigt, um aus den eingehenden Sensormessungen das

Umgebungsmodell zu erzeugen und ggf. an die Anwendung weiterzugeben. Insbesondere beinhaltet dies die Zeit, die für die Kommunikation zwischen den verschiedenen Systemen (Sensoren, Sensordatenverarbeitung, Anwendung) benötigt wird. Wie auch in den zuvor genannten Gütekriterien obliegt die Quantifizierung der erlaubten Antwortzeiten dem Anwendungsexperten. Unter Umständen kann sie auch vollständig ignoriert werden, wenn z.B. eine Offline Analyse der Sensormessungen durchgeführt wird. Andere Anwendungen benötigen hingegen eine verhältnismäßig kurze Antwortzeit, da sich die betrachtete Umgebung i.d.R. schnell verändert und bei einer zu großen Antwortzeit, dass erzeugte Umgebungsmodell bereits veraltet ist, wie es schematisch in Abbildung 1.3 dargestellt ist.

Aufbauend auf den zuvor genannten Gütekriterien für sensordatenverarbeitende Systeme sowie die benötigte Expertise zur Konkretisierung der Kriterien in Hinblick auf die Anwendung, lassen sich nun Anforderungen an die entwickelte Methode sowie das eingesetzte Tooling ableiten. Bevor dies geschieht, wird jedoch im nächsten Abschnitt ein Überblick über die verwendete, konzeptionelle Architektur sowie die Methode gegeben.

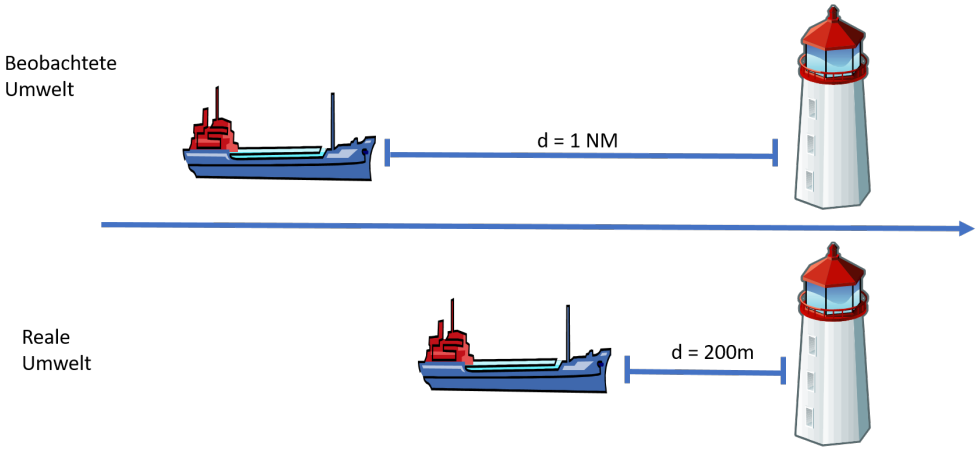

Abbildung 1.3: Auswirkungen einer zu großen Antwortzeit eines Sensordatenverarbeitenden Systems

1.3 Lösungsansatz zur Bewertung von sensordatenverarbeitenden Systemen

Dieser Abschnitt stellt die generelle Idee bzw. den generellen Lösungsansatz zur Bewertung von sensordatenverarbeitenden Systemen vor. Dabei handelt es sich um eine Vorschau auf das vierte Kapitel, bzw. eine Übersicht über den gewählten Ansatz und dient der Einordnung nachfolgender Kapitel in den Gesamtkontext der Arbeit.

Das Ziel dieser Arbeit ist, eine Methode zu finden, mit der sensordatenverarbeitende Systeme auf ihre Korrektheit in Bezug auf die oben definierten Qualitätskriterien überprüft werden können. Dabei sollen so wenige Anforderungen wie möglich an das zu überprüfende System gestellt werden, um mit der hier vorgestellten Methode eine breite Auswahl an sensordatenverarbeitenden Systemen testen zu können.

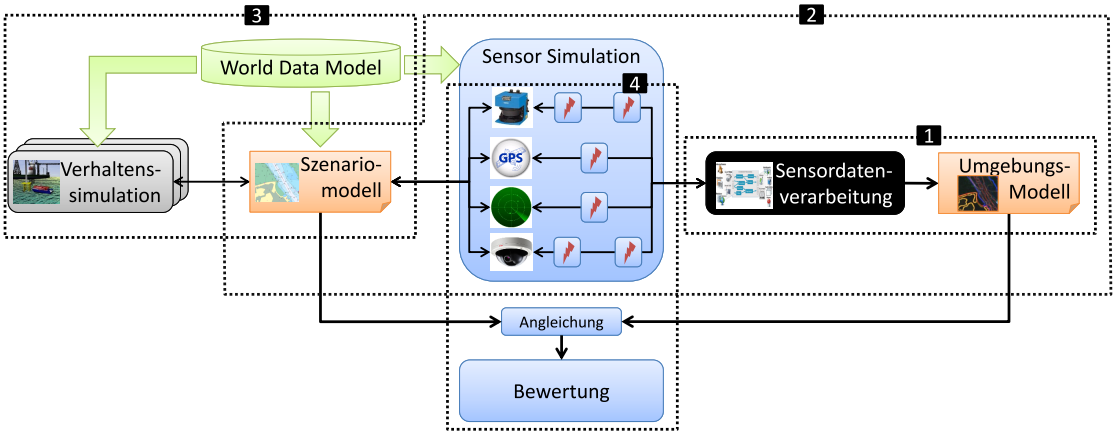

Abbildung 1.4: Konzeptionelle Architektur zur simulativen Überprüfung von sensordatenverarbeitenden Systemen

Abbildung 1.4 stellt die konzeptionelle Architektur des vorgeschlagenen Systems dar. Dieses kann in vier, sich teilweise überschneidende Sichten eingeteilt werden:

1. Das sensordatenverarbeitende System, dass sich mit dem zu überprüfenden System beschäftigt.
2. Die Datenerzeugung beschäftigt sich mit der Bereitstellung benötigter Daten für das zu überprüfende System.
3. Die Verhaltenssimulation beschreibt das dynamische Verhalten der Umgebung.
4. Die Bewertung der Sensordatenverarbeitung bewertet das zu überprüfenden Systems unter Berücksichtigung der zur Verfügung stehenden Informationen durch.

Diese verschiedenen Sichtweisen auf die konzeptionelle Architektur werden im Folgenden näher erläutert.

Das sensordatenverarbeitende System

Der schwarz gepunktete Kasten in Abbildung 1.4 mit der Nummer 1 zeigt das zu testende System in dem die Sensordatenverarbeitung erfolgt. Antwort der Sensordatenverarbeitung auf einen gegebenen Stimulus ist das sogenannte erkannte Umgebungsmodell.

Dieses beinhaltet das ermittelte Wissen des Systems über die Umgebung, in der es sich zu befinden glaubt und dient gemäß des Sense-Think-Act Paradigmas als Eingabe für die nachfolgenden Systeme.
Gleichermaßen handelt es sich bei dem Umgebungsmodell um das zu überprüfende Ergebnis der Sensordatenverarbeitung.

Wie bereits erwähnt, soll mit dem entwickelten Ansatz eine möglichst große Bandbreite an sensordatenverarbeitenden Systemen untersucht werden können. Aus diesem Grund wird davon ausgegangen, dass kein zusätzliches Wissen über die konkrete Sensordatenverarbeitung zur Verfügung steht.
Aus Sicht der entwickelten Methode handelt es sich bei der Sensordatenverarbeitung um ein Blackbox System, welches von einem oder mehreren Sensoren Daten erhält und ein aktuelles, Umgebungsmodell liefert.

Datenerzeugung

In einem realen System erhält das zu überprüfende System seine Eingangswerte durch reale Sensoren, die die Umgebung des Systems beobachten. Dementsprechend stellt der Zustand der realen Umgebung, den Erwartungswert für die Ausgabe des zu testenden Systems dar.
Dieser Erwartungswert lässt sich aber in einer realen Umgebung nicht genau bestimmen. Um dies bewerkstelligen zu können, müsste das System in einer vollständig kontrollierten Umgebung operieren, was im Allgemeinen nicht oder nur mit sehr großen finanziellen und zeitlichem Aufwand möglich ist.
Umgangen werden kann dieses Problem durch den Einsatz von Simulation, indem die reale Umgebung, wie sie von den Sensoren beobachtet wird, durch eine virtuelle Umgebung ersetzt wird. In dieser virtuellen Umgebung, kann der aktuelle Zustand und damit der Erwartungswert, zu jedem Zeitpunkt genau ermittelt werden. Im gleichen Rahmen müssen auch die realen Sensoren durch virtuelle bzw. simulierte Sensoren ersetzt werden.

In Abbildung 1.4 stellt das Szenariomodell eine virtuelle Abstraktion der realen Umgebung dar. Sie enthält alle Objekte, die von den Sensoren beobachtet werden können. Dazu gehören neben den Objekten, welche direkt von der Sensordatenverarbeitung erkannt werden sollen auch Hintergrundobjekte, die im Umgebungsmodell der Sensordatenverarbeitung nicht abgebildet werden.
Während das Szenariomodell in Abbildung 1.4 konkrete Instanzen von Objekten beinhaltet, kann das World Data Model als ein Objektkatalog betrachtet werden, in dem die Struktur der Objekte beschrieben wird.

Die Struktur der einzelnen Objekte bzw. ihre konkreten Instanziierungen werden von den simulierten Sensoren für die Messwertsynthese verwendet. Im Gegensatz zu ihren realen Vorbildern, sind simulierte Sensoren in der Lage, exakte bzw. ideale Werte eines beobachteten Phänomens zu erzeugen. Um Messungenauigkeiten und Sensorfehler bewerten zu können, werden im Nachgang an die Messwertsynthese Ungenauigkeiten und Fehler in die simulierten Sensormesswerte eingebracht.
Durch die Verwendung entsprechender Fehlermodelle lassen sich auf diese Weise, sowohl das reale Fehlerverhalten der Sensoren, als auch experimentelle Fehlerkonfigurationen wie beispielsweise höherwertigere oder niederwertigere Sensorik, nachbilden.

Verhaltenssimulation

Ein weiterer wichtiger Aspekt bei der Überprüfung sensordatenverarbeitender Systeme im Umfeld der autonomen Systeme, ist die Berücksichtigung der dynamischen Effekte der Umwelt. Diese werden in einigen Verarbeitungsalgorithmen der Sensordatenverarbeitung verwendet, um nicht direkt messbare Eigenschaften beobachteter Objekte herzuleiten.
Um das dynamische Verhalten der Objekte in der Sensorsimulation zu berücksichtigen, können verschieden Simulationen in das System integriert werden. Diese Simulationen werden im Folgenden als Verhaltenssimulationen bezeichnet. Die Verhaltenssimulationen manipulieren während der Laufzeit Teilaspekte des Szenariomodells.

Bewertung der Sensordatenverarbeitung

Die Bewertung der Sensordatenverarbeitung erfolgt durch den Vergleich zwischen dem von der Sensordatenverarbeitung ermittelten Umgebungsmodell, mit dem zur Sensorwertsynthese verwendeten Szenariomodell.
Es wird davon ausgegangen, dass das von der Sensordatenverarbeitung erstellte Umgebungsmodell ein Teilmodell des Szenariomodells repräsentiert. In der Regel werden diese beiden Abstraktionen der betrachteten Umgebung nicht über die gleichen Methoden beschrieben. Zum Zweck des Vergleiches, muss ggf. eine Angleichung der Repräsentationen zugunsten eines der Modelle (Szenariomodell oder Umgebungsmodell aus Abbildung 1.4) vorgenommen werden. Für die eigentliche Bewertung können dann sowohl Informationen aus dem Szenariomodell als auch von den idealen Sensormesswerten herangezogen werden.

1.4 Beitrag der Arbeit

Um die in Abschnitt 1.2 beschriebene Fragestellung beantworten zu können, wird in dieser Arbeit ein werkzeugbasiertes Vorgehen vorgestellt, mit dem eine simulative Überprüfung von sensordatenverarbeitenden Systemen ermöglicht wird. Diese kann sowohl während der Entwicklung neuer sensordatenverarbeitender Systeme eingesetzt werden, indem während der Entwicklung Testdatensätze zur Verfügung gestellt werden, als auch zu deren nachträglicher Überprüfung. Bei der Überprüfung wird dabei insbesondere auf die in Abschnitt 1.2.1 vorgestellten Gütekriterien für sensordatenverarbeitende Systeme eingegangen.
Zusammengefasst liefert die Arbeit dabei folgende Beiträge:

- Eine strukturierte Beschreibung von Sensoren innerhalb ihres Systemkontextes, d.h. ihrer Umgebung, inklusive ihres normativen und ggf. Fehlverhaltens.
  Dabei wird besonders auf eine Trennung zwischen der Generierung von Sensorbeobachtungen und der Anwendung von Sensorfehlern und Messungenauigkeiten eingegangen. Diese führt dazu, dass 1) die Entwicklung von Sensor- und Fehlermodellen vereinfacht werden und 2) eine Wiederverwendbarkeit der Modelle sowie konkreter Simulationsergebnisse, z.B. für eine nachträgliche Bewertung der Sensordatenverarbeitung, ermöglicht werden kann.

- Die Möglichkeit verschiedene Fehlermodelle für einen Sensor zu kombinieren und individuell zu konfigurieren. Durch diese Möglichkeit können existierende Fehlermodelle einfach nachgebildet aber auch neue Fehlermodelle z.B. für zukünftige Sensoren, erprobt werden. Diese Fähigkeit des vorgestellten Vorgehens bzw. Werkzeugs, kann beispielsweise zur Überprüfung von hypothetischen Fehlermodellen genutzt werden.

- Die Möglichkeit den Systemkontext eines Sensors oder mehrerer Sensoren nicht nur für die Erzeugung der Sensordaten heranzuziehen, sondern diese auch für die Bewertung des sensordatenverarbeitenden Systems zu verwenden.

Des Weiteren wird im Verlauf dieser Arbeit gezeigt, wie die Sensordatengenerierung sowie Bewertung von sensordatenverarbeitenden Systemen durch den Einsatz von Ko-Simulationstechniken profitieren können, indem die Komplexität der beteiligten Simulatoren reduziert wird.
Nicht zuletzt kann das entwickelte Werkzeug bei der Entwicklung von neuen sensordatenverarbeitenden Systemen genutzt werden, um frühzeitig Testdaten für das System bereitzustellen.

1.5 Aufbau der Arbeit

Die Arbeit gliedert sich neben der Einleitung und dem Fazit in vier Hauptabschnitte, in denen zunächst grundlegenden Systeme und Ansätze betrachtet werden (Kapitel 2 & 3). Dabei beschäftigt sich das zweite Kapitel hauptsächlich mit den zu untersuchenden Systemen sowie seinen Komponenten. Hier werden grundlegende Begrifflichkeiten und Zusammenhänge definiert bzw. diskutiert, die in der späteren Umsetzung berücksichtigt werden müssen.
Das Kapitel "Ansätze zur Sensordatengenerierung & Überprüfung" (Kapitel 3) beschäftigt sich mit verschiedenen Ansätzen zur Sensordatengenerierung. Weiterhin werden bestehende Lösungen zur Sensordatengenerierung und Überprüfung von robotischen Systemen, auf ihre Eignung überprüft, die gegebene Fragestellung zu beantworten.

Das vierte Kapitel stellt den Hauptteil der vorliegenden Arbeit dar und beschäftigt sich mit der Konkretisierung des eigenen Ansatzes. Innerhalb dieses Kapitels wird zunächst auf die Modellierung der Sensoren sowie ihres Verhaltens bzw. Fehlverhaltens eingegangen. Weiterhin wird die Bewertung von sensordatenverarbeitenden Systemen betrachtet.

Aufbauend auf der im vierten Kapitel beschriebenen Methode, wird im fünften Kapitel eine prototypische Umsetzung des Ansatzes vorgestellt. Hierbei kommen Techniken aus der modellgetriebenen Softwareentwicklung zum Einsatz.

Im sechsten Kapitel (Kapitel 6) werden die vorgestellten Ansätze sowie die prototypische Umsetzung in Hinblick auf die Erfüllung der gestellten Anforderungen evaluiert. Dies geschieht anhand von Anwendungsfällen aus der maritimen Domäne.

Kapitel 2

Sensoren und sensordatenverarbeitende Systeme

Das zweite Kapitel dieser Arbeit beschäftigt sich grundlegend mit dem zu testenden System, der Sensordatenverarbeitung sowie den verwendeten Sensoren und Möglichkeiten ihrer Modellierung.
Dazu wird zunächst eine allgemeine Architektur zur Sensordatenverarbeitung vorgestellt und hinsichtlich der zuvor genannten Gütekriterien diskutiert. Anschließend wird auf die verwendeten Sensoren eingegangen. Dies beinhaltet neben einer Klassifizierung von Sensoren insbesondere die Modellierung von Sensoren sowie eine Betrachtung ihres möglichen Fehlverhaltens. Das Modellierung des Fehlverhalten von Sensoren wiederum bildet die Grundlage für die im Kapitel 3 untersuchten Simulation.

2.1 Sensordatenverarbeitung

Dieser Abschnitt beschreibt grundlegende Techniken und Vorgehensmodelle aus dem Bereich der Sensordatenverarbeitung. Er dient der Abgrenzung von Möglichkeiten, die mit dem zu entwickelnden Lösungsansatz unterstützt werden sollen, bzw. zur Einordnung der in Abschnitt 1.2.1 beschriebenen Gütekriterien.
Da für den gewählten Lösungsansatz, dem funktionalen Testen der Sensordatenverarbeitung mittels Black Box Methoden, i.d.R. kein detailliertes Wissen über die innere Funktionsweise des zu testenden Systems benötigt wird, beschränkt sich dieser Abschnitt auf das generelle Vorgehensmodell bei der Sensordatenverarbeitung.

Die folgende Abbildung 2.1 zeigt eine generelle Drei-Schichten-Architektur zur Sensordatenfusion von Umfeldsensoren für Assistenzsysteme, wie sie von M. Darms in seiner Dissertation [Dar07] entwickelt wurde.

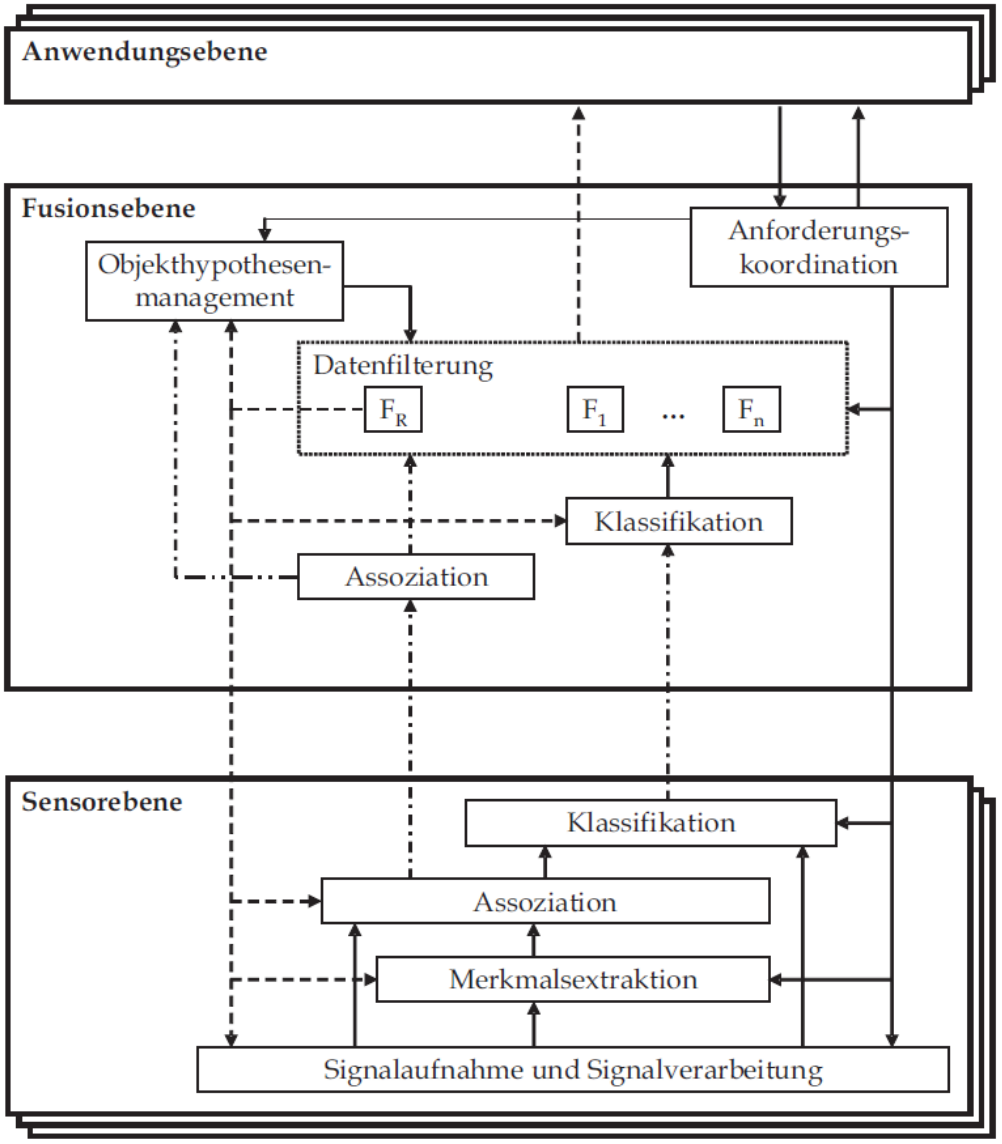

Abbildung 2.1: Drei-Schichten-Architektur für Sensorfusionssysteme nach [Dar07]

Die Architektur gliedert sich in die drei Ebenen: Sensoren, Fusion und Anwendung. Für diese Arbeit sind dabei insbesondere die Sensor- und Fusionsebene von Bedeutung. Die Anwendungsebene, bei der es sich bei M. Darms um die Realisierung der Assistenzsysteme handelt, wird hier nicht weiter betrachtet.

Die Sensorebene als die unterste Schicht der Architektur stellt in dem Zusammenhang dieser Arbeit die Schnittstelle mit der zu entwickelnden Sensorsimulation dar.

In ihr werden zunächst die von den Sensoren gelieferten Rohdaten wie beispielsweise Bilder, Entfernungen oder Helligkeiten aufgenommen und auf der Signalebene vor verarbeitet. Bei dieser Vorverarbeitung handelt es sich häufig um eine Filterung bzw. Rauschunterdrückung der eingehenden Daten oder um eine Angleichung der verwendeten Einheiten und Referenzsysteme. In der Arbeit "Introduction to multisensor data fusion" [HL97] liefern Hall und Llinas einen guten Überblick über Techniken, die in diesem sowie den folgenden Prozessschritten angewendet werden können.

Im nächsten Schritt der Sensorebene werden in den vor verarbeiteten Rohdaten Merkmale identifiziert (Merkmalsextraktion in Abbildung 2.1). Merkmale in diesem Zusammenhang können ggf. Linienzüge oder Polygone innerhalb einer Laserscanner- oder Radarmessung sein aber auch auffällige Elemente innerhalb eines Bildes. In der Regel handelt es sich bei diesen Merkmalen um die Entitäten, die in den weiteren Schritten näher analysiert werden.

Bei der anschließenden Datenassoziation werden die erkannten Rohdaten bzw. die extrahierten Merkmale, bekannten Objekten oder Objekthypothesen zugeordnet. Diese kann ggf. auf Grundlage des messenden Sensors erfolgen, z.B. kann die von einem GPS-Sensor gemessene Position, genau einem Sensor zugeordnet werden. Im Fall von Sensoren zur Umfelderkennung werden unter Umständen verschiedene, zuvor extrahierte Merkmale zu einem neuen Objekt zusammengefasst und stehen fortan als Objekthypothesen im System zur Verfügung.

Auf Grundlage der ermittelten Rohdaten, Merkmale und Objekthypothesen kann optional eine zusätzliche erste Klassifizierung der erkannten Merkmale und Objekte vorgenommen werden. Eine beispielhafte Klassifizierung, wie sie auf dieser Ebene vorgenommen werden kann, ist in Abbildung 2.2 für einen Laserscanner visualisiert. Dabei werden nach Darms auf dieser Ebene ausschließlich Messwerte aus einer Sensormessung verwendet und zum Beispiel durch Übereinstimmung mit Referenzgeometrien verglichen [FDL02].

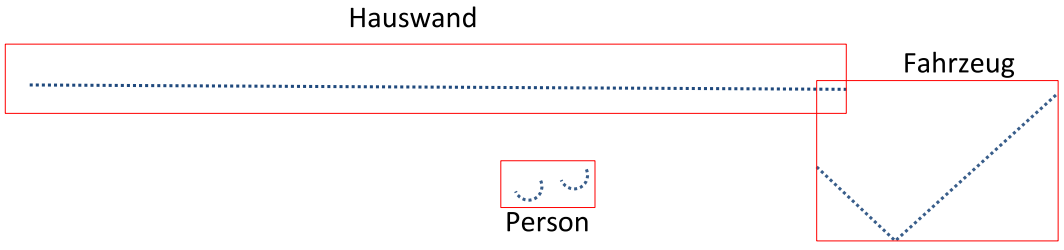

Abbildung 2.2: Beispiel für Klassifikation von Merkmalen aufgrund der Merkmalsgeometrien

Die bisher beschriebenen Schritte werden für alle an die Sensordatenverarbeitung angeschlossenen Sensoren durchgeführt, wobei je nach verwendeten Sensor und Aufgabenstellung, nicht jeder Teilschritt zwingend durchgeführt werden muss.

Die Fusionsebene in der von Darms vorgeschlagenen Architektur beschäftigt sich, mit der Fusion verschiedener und gegebenenfalls heterogener Sensoren.
Wie schon auf der Sensorebene kann in der Fusionsebene eine Datenassoziation stattfinden. Diese arbeitet nicht mehr auf den Rohdaten, sondern auf den bereits vor verarbeiteten Objekten bzw. Objekthypothesen. Dabei wird insbesondere das Problem angegangen, dass verschiedene Sensoren dasselbe Objekt beobachten und ggf. mit unterschiedlichen Identitäten an die Fusionsebene weiterleiten.

Genau wie die Datenassoziation kann die Klassifikation auf der Fusionsebene weitere Informationen zur Erfüllung ihrer Aufgabe verwenden. Insbesondere können in dieser Schicht historische Informationen verwendet werden, um Beispielsweise eine Klassifikationshypothese zu überprüfen. Die Arbeiten [FW01, FDL02] verwenden beispielsweise typische Bewegungsmuster von Menschen zur Überprüfung einer durch Laserscannermessungen erhobenen Klassifikationshypothese.
Andere Verfahren, wie beispielsweise das von A. Bolles entwickelte datenstrombasierte Framework zur Objektverfolgung [Bol11], benutzen historische Daten für eine verlässliche Objektverfolgung, indem zunächst ausgehend von historischen Daten ein Dynamikmodell geschätzt wird. Mithilfe des geschätzten Modells, kann die Position des Objektes in die Zukunft prädiziert werden und ggf. bei der nächsten Messung als Referenz für die Datenassoziation dienen.

2.1.1 Anknüpfungspunkte zur Bewertung von sensordatenverarbeitenden Systemen

Aus der Drei-Schichten-Architektur von M. Darms wird deutlich, dass die Überprüfung der Sensordatenverarbeitung bezüglich der in Abschnitt 1.2.1 aufgestellten Gütekriterien, an drei Stellen ansetzen kann:

1. Nach der Sensorebene: In diesem Fall werden die Sensorrohwerte bzw. Sensorsignale vom Testsystem erzeugt. Wohingegen das Szenariomodell die Erwartungswerte für die, von der Sensordatenverarbeitung ermittelten Objekthypothesen darstellt.

2. Nach der Fusionsebene: Hierbei werden Objekthypothesen für verschiedene Sensoren durch das Testsystem erzeugt.

3. Nach der Fusionsebene: Wobei das Testsystem Sensorrohwerte bzw. Sensorsignale erzeugt.
   Bei dieser Überprüfungsmethode muss darauf geachtet werden, dass eine Bewertung hinsichtlich der Gütekritierien: Fehlertoleranz gegenüber Sensorfehlern (SDV2) und Robustheit bzgl. Messunsicherheiten (SDV3) nur indirekt durchgeführt werden kann, da ggf. auftretende Fehler und Unsicherheiten durch die Fusionsebene ausgeglichen werden.

Aus diesen drei Anknüpfungspunkten ergibt sich ein weiterer Vorteil einer strukturierten Beschreibung des normativen Verhaltens von Sensoren und ihrem Verhalten.
Für eine Überprüfung der Sensorebene müssen aus dem Szenariomodell Objekthypothesen gebildet werden, die mit den ermittelten Objekthypothesen verglichen werden können. Diese gebildeten Objekthypothesen können gleichzeitig als Eingabe für die Überprüfung der Fusionsebene verwendet werden. Die sich daraus ergebende Trennung der beiden Ebenen ermöglicht eine parallele Entwicklung von Sensorebene und Fusionsebene.

2.2 Sensoren und Sensortypen

Der Begriff des Sensors nimmt in dieser Arbeit einen besonderen Stellenwert ein. Dieser Abschnitt beschreibt zunächst das Verständnis eines Sensors im allgemeinen sowie verschiedene Ausprägungen des Sensorbegriffes. Dazu gehören insbesondere die Spezialformen des virtuellen und simulierten Sensors.

Die Aufgabe eines Sensors ist die Bestimmung der Werte bestimmter Eigenschaften eines Objektes. In der Regel wird dies über eine indirekte Messung und anschließendes folgern der Eigenschaften erreicht.
Ein Laserscanner zum Beispiel misst über die Laufzeit von Lichtimpulsen vom Aussenden bis zur Detektion der Reflexion die Entfernung zwischen zwei Punkten. Aus dieser Information lässt sich die Position des beobachteten Objektes bezüglich der eigenen Position des Laserscanners bestimmen. Über eine wiederholte Messung lässt sich die Form eines Objektes bzw. seine Oberfläche ermitteln.

2.2.1 Aktive & Passive Sensoren

Klassischerweise lassen sich Sensoren in zwei Kategorien einteilen; in passive und aktive Sensoren [DRRB07]. Diese Unterteilung bezieht sich dabei auf die Interaktion der Sensoren mit ihrer Umwelt.

Passive Sensoren

Passive Sensoren zeichnen sich dadurch aus, dass sie ihre Umgebung aufnehmen, ohne aktiv in sie einzugreifen. Die Aufnahme von Messwerten geschieht durch das Ändern von Eigenschaften innerhalb des Sensors durch von außen hinzugefügte Energie. Klassische Vertreter von passiven Sensoren sind:

Optische Sensoren Optische Sensoren, zu denen unter anderem Farb- oder Temperaturkameras gehören, zählen zu den bekanntesten Sensoren. Sie nehmen ihre Umgebung wahr, indem sie das Licht welches von externen Quellen emittiert oder reflektiert wird, in elektrische Signale umwandeln.
Optische Sensoren eignen sich im Rahmen einer Sensordatenfusion insbesondere für das Erkennen von Mustern und Objekten.

(Klassische) Thermometer Viele Thermometer messen nach wie vor die Temperatur mithilfe der Ausdehnung von Materialien, deren Ausdehnungskoeffizient bekannt ist.

Magnetkompass Bei dem ursprünglichen Magnetkompass wird die Kompassnadel (als interner Bestandteil des Sensors) anhand des Magnetfeldes der Erde ausgerichtet.

Passive Sensoren verfügen im Gegensatz zu aktiven Sensoren über den Vorteil, dass sie sich i.d.R. nicht gegenseitig beeinflussen und somit ohne große Einschränkungen eingesetzt werden können.

Aktive Sensoren

Aktive Sensoren verändern ihre Umgebung und messen die Reaktion der Umgebung auf diese Veränderung. Dies kann zum Beispiel durch das Emittieren von elektromagnetischen Wellen (Radar, Licht, ...) oder Schallwellen geschehen. In diesem Fall, wird von deren Reflexion auf die entsprechende Eigenschaft (zum Beispiel Entfernung) geschlossen.
Klassische Vertreter von aktiven Sensoren zur Umfelderkennung sind:

Laserscanner senden kurze Lichtimpulse einer bestimmten Wellenlänge aus und messen anschließend ihre Reflexion. Aus den dadurch gewonnenen Informationen lässt sich neben der Entfernungsermittlung auch eine Abschätzung der reflektierenden Materialien vornehmen.

Kinect Der verhältnismäßig neue Kinect Sensor von Microsoft (2010) fällt unter die Klasse der *structured light* Sensoren. Bei dieser Technik werden strukturierte Lichtimpulse ausgesendet. Die Ermittlung der zu messenden Eigenschaften z.B. bei dem Kinect Sensor erfolgt durch die Analyse der veränderten Struktur des emittierten Lichtes. Da es sich bei dem vom Sensor emittieren Licht um infrarotes Licht handelt, wird die Kinect sehr stark von der Sonneneinstrahlung beeinflusst [KE12].

Radar Bei Radarsensoren werden, ähnlich wie bei Laserscannern, gerichtete elektromagnetische Impulse ausgesendet und die Laufzeit zwischen dem Aussenden und Auffangen der Reflexion gemessen. Neben der Entfernungsermittlung sind Radarsensoren i.d.R. in der Lage, mithilfe des Doppler Effekts die relativen Geschwindigkeiten zwischen sich selbst und den beobachteten Objekten zu ermitteln [Sko81]. Weitere Vorteile von Radarsensoren gegenüber passiven Sensoren sind, dass sie nicht so stark von den äußeren Faktoren, wie beispielsweise der Helligkeit abhängen. Zudem verfügen sie i.d.R. über eine sehr hohe Reichweite, was sie insbesondere für den maritimen Bereich interessant macht.

Im Gegensatz zu den passiven Sensoren können aktive Sensoren des gleichen Wirkprinzips (z.B. Laser, strukturiertes Licht, usw.) nicht beliebig miteinander kombiniert werden, da sie sich ggf. gegenseitig beeinflussen. Insbesondere in dynamischen Umgebungen oder Sensoren, die auf einer mobilen Sensorplattform angebracht sind, kann dies zu unerwarteten Verhalten führen.
Auf der anderen Seite sind aktive Sensoren i.d.R. nicht so stark von dem Umwelteinflüssen abhängig wie passive Sensoren.

2.2.2 Simulierte, virtuelle und reale Sensoren

Neben der Unterteilung in aktive und passive Sensoren wird in dieser Arbeit eine zusätzliche Unterteilung in reale, virtuelle und simulierte Sensoren vorgenommen. Da die Interpretation von virtuellen Sensoren und ihre Abgrenzung in der Literatur nicht eindeutig ist, wird an dieser Stelle die für diese Arbeit verwendete Interpretation vorgestellt.

Realer Sensor Unter einem realen Sensor wird in dieser Arbeit ein Sensor verstanden, wie er aktuell in physischer Form erworben werden kann. Er besteht immer aus einer Hardwarekomponente mit bekannten Ausmaßen, sowie Schnittstellen über die die Messwerte in digitaler Form an eine Sensordatenverarbeitung weitergegeben werden können. Die Schnittstelle kann unter anderem auch eine Treibersoftware umfassen, mit der analoge in digitale Signale umgewandelt werden.

Smarter Sensor Ein smarter Sensor ist ein Sensor der mithilfe einer internen Logik zusätzliche Mehrwerte gegenüber den reinen Sensorsignalen erzeugen kann [Gia86]. In Hinblick auf die zuvor vorgestellte Drei-Schichten-Architektur zur Sensorfusion können eine oder mehrere Instanziierungen der Sensorebene durch einen smarten Sensor ersetzt werden, welcher bereits Objekthypothesen als Messergebnis zur Verfügung stellt.

Virtueller Sensor Unter einem virtuellen Sensor wird in dieser Arbeit ein Sensor verstanden, der ausschließlich aus Softwarekomponenten besteht. Die Software kann ggf. andere virtuelle oder reale Sensoren als Datenquelle verwenden und mittels Fusion höherwertige Interpretationen als Messwerte ausgeben. Dieses Verständnis schließt sich der Definition von Muir [Mui90] an, der insbesondere den Begriff der virtuellen Sensorwerte als eine Kombination von physikalischen Sensorwerten sowie ihrem Mapping in ein geeignetes Format versteht. Dadurch sieht es für den Benutzer eines virtuellen Sensors so aus, als würden die Messwerte von dem virtuellen Sensor erzeugt werden.

Simulierter Sensor Ein simulierter Sensor ist eine Softwarekomponente, die aus der Beobachtung einer virtuellen Umgebung (neue) Sensormesswerte erzeugt. Im Gegensatz zu virtuellen Sensoren, die Messwerte verschiedener realer (oder simulierter) Sensoren zu einer neuen Interpretation zusammenfassen, werden die Messwerte eines simulierten Sensors aus der Beobachtung der virtuellen Umgebung erzeugt.

2.2.3 Modellierung von Sensoren

Für die Modellierung von Sensoren, lassen sich in der Literatur verschiedene Modelle, wie beispielsweise die *Sensor Model Language* - SensorML [BR07], das *Observations and Measurements* - O&M Modell [Cox07] oder die *Sensor Semantic Network* (SSN) Ontologie [CBB+12, LHT+11] finden.
Dabei stellt die SSN Ontologie den erfolgversprechendsten Ansatz dar, da mit ihr beispielsweise eine Unterscheidung zwischen aktiven und passiven Sensoren vorgenommen werden kann.

2.2.3.1 Sensor Semantic Network Ontologie

Erstellt wurde die SSN Ontologie im Jahr 2010 von der W3C Sensor Network Incubator Group[1], mit dem Ziel, die bisher verfügbaren Modelle zur Beschreibung von Sensoren

[1] `http://www.w3.org/2005/Incubator/ssn/`

(vgl. *Sensor Model Language* - SensorML [BR07]) und ihren Beobachtungen (vgl. *Observations and Measurements* - O&M [Cox07]), um eine semantische Ebene zu erweitern.
Die SSN ist in die in Abbildung 2.3 dargestellten zehn konzeptionellen Modelle un-

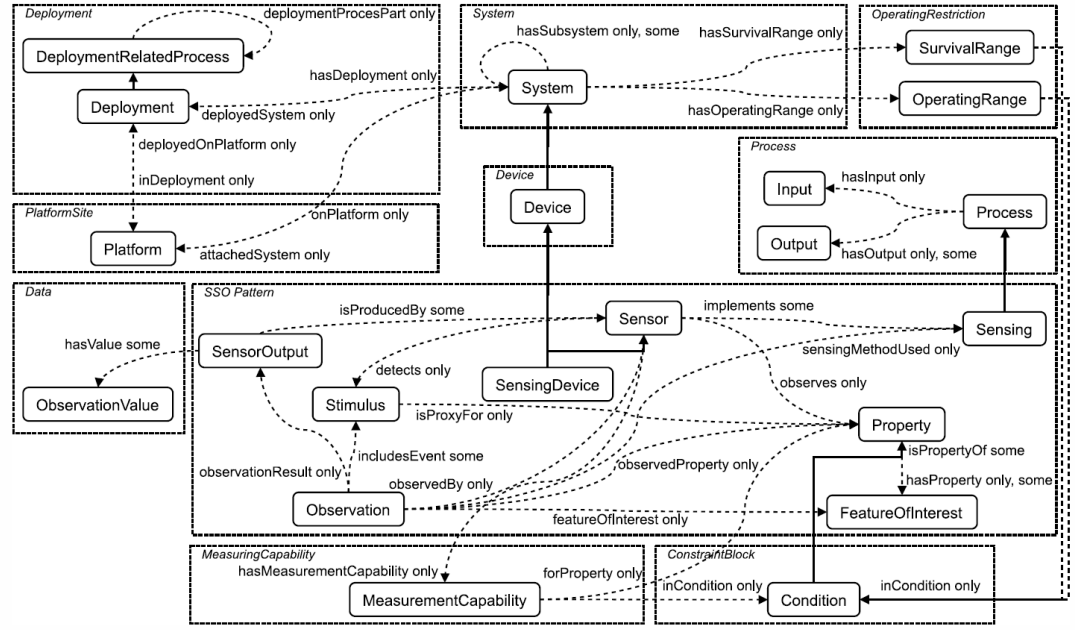

Abbildung 2.3: Konzepte und Relationen der Sensor Semantic Network Ontologie (SSN), unterteilt in konzeptionelle Module [CBB+12].

terteilt, welche nach Auffassung ihrer Autoren (vgl. [CBB+12]) aus vier Perspektiven betrachtet werden können:
Die **Sensorperspektive** beschreibt was ein Sensor beobachtet und in welcher Form die Beobachtung durchgeführt wird. Die **Beobachtungsperspektive**, die sich auf die vom Sensor erstellte Beobachtung konzentriert und dabei ggf. beobachtungsrelevante Metadaten betrachtet. Sowie eine **Eigenschaftenebene** in der die beobachteten Eigenschaften von Objekten betrachtet werden und was der Sensor aus diesen Eigenschaften abgeleitet hat. Die vierte und letzte Perspektive ist die **Systemperspektive** in der Systeme von Sensoren organisiert und ihre Bereitstellung betrachtet wird.

Im Rahmen einer Simulation von Sensoren, werden abgesehen von der Systemperspektive alle anderen Sichtweisen auf die SSN Ontologie berücksichtigt. Diese Zusammenhänge werden in dem sogenannten Stimulus - Sensor - Observation (SSO) Pattern zusammengefasst, welches gleichzeitig die Grundlage für die Sensorperspektive der SSN darstellt. Eine detaillierte Beschreibung des SSO-Patterns kann der Abhandlung “The Stimulus-Sensor-Observation Ontology Design Pattern and it Integration into the Semantic Sensor Network Ontology“ [JC10] entnommen werden, die während der Entwicklung der SSN entstanden ist. Eine Darstellung des SSO-Patterns ist in Abbildung 2.4 dargestellt.

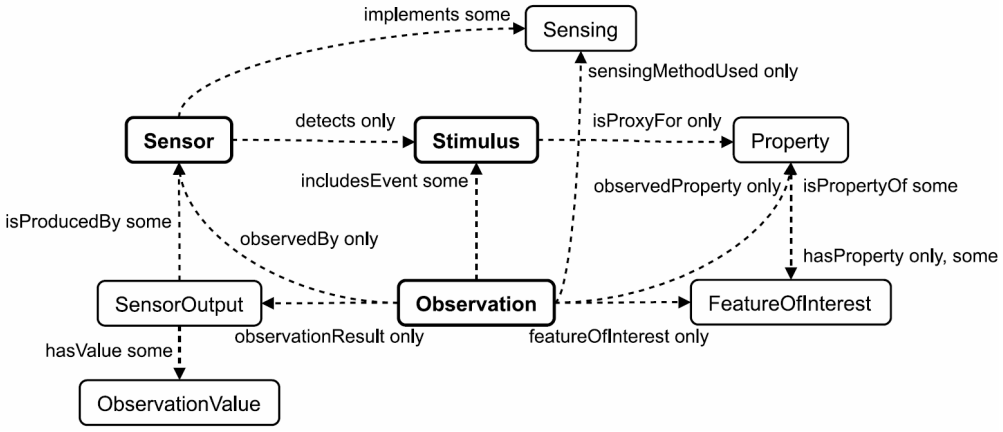

Abbildung 2.4: Darstellung des Stimulus - Sensor - Observation Pattern aus [CBB+12].

Bei dem SSO-Pattern transformieren Sensoren einen eingehenden Stimulus in eine (meist) digitale Repräsentation der beobachteten Werte, den *SensorOutput*. Dieser *SensorOutput* wiederum besteht aus einer Menge von beobachteten Werten (*ObservationValue*). Bei dem Stimulus handelt es sich um einen Stellvertreter für eine oder mehrere beobachtbare Eigenschaften (*Property*), eines Elementes aus der beobachteten Umwelt (*FeatureOfInterest*).
Die Beobachtung (*Observation*) stellt in diesem Pattern die Verknüpfung der Elemente Sensor, *Stimulus*, *Property* und *FeatureOfInterest* dar.

Neben der Beschreibung des Stimulus - Sensor - Observation Patterns beinhaltet die Sensorperspektive der SSN eine Betrachtung der Messmöglichkeiten (*MeasurementCapability* in Abbildung 2.3) eines Sensors. Diese Messmöglichkeiten stellen eine Verknüpfung zwischen einer beobachtbaren Eigenschaft (*Property*) und einer Bedingung (*Condition*) dar und können diese Kombination mit einer Messeigenschaft verknüpfen. Bei den Messeigenschaften wiederum kann es sich um Qualitätsdimensionen (z.B. Genauigkeit, Drift, Präzision, usw.) der Sensorwahrnehmung handeln. Eine detailliertere Beschreibung der Qualitätsdimensionen kann in der Doktorarbeit von C. Kuka gefunden werden [Kuk15].

2.2.3.2 Diskussion der SSN Ontologie

Das sich die Sensor Semantic Network Ontologie für die Modellierung von Sensorsystemen eignet, wurde bereits in verschiedenen Arbeiten gezeigt.
In der “Swiss Experiment“ Plattform [CJCA11] wird die SSN für die Repräsentation von Sensoren und ihren Beobachtungen in einem Sensornetzwerk verwendet. Innerhalb des Projektes wird die generische Sensor Semantic Network Ontologie mit der

NASA SWEET (Semantic Web for Earth and Environmental Terminology)[2] Ontologie ([Ras06]) verknüpft, die ein spezialisiertes Vokabular für die beobachteten Typen bereitstellt.

Im Projekt SemSorGrid4Env[3] wird eine serviceorientierte Architektur und Middleware für den Austausch von Sensordaten entwickelt. Die SSN Ontologie wird in diesem Zusammenhang ebenfalls als Datenmodell für die Sensoren und ihre Beobachtungen verwendet. Ähnlich wie auch bei der "Swiss Experiment" Plattform wird innerhalb des SemSorGrid4Env Projekt die SWEET Ontologie mit der SSN kombiniert.

In seiner Doktorarbeit verwendet C. Kuka [Kuk15] die Sensor Semantic Network Ontologie zur indirekten Bestimmung von Qualitätsinformationen von Sensoren bzw. deren Beobachtungen. Diese können bei der Verarbeitung in einem Datenstrommanagementsystem berücksichtigt werden. Die SSN Ontologie wird dabei aus der Sensorperspektive mit besonderem Augenmerk auf die Messmöglichkeiten und ihre Qualitätsdimensionen betrachtet. Dabei wird über die Ontologie eine Messmöglichkeit eines Sensors, sowie Bedingungen die sich auf bestimmte Eigenschaften (*Property*) der Umgebung beziehen, eine Verknüpfung zu der erwarteten Qualitätsdimension hergestellt.
Auf diese Weise lässt sich zum Beispiel ausdrücken, dass ein Positionssensor über eine niedrigere Genauigkeit verfügt, wenn die Umgebungstemperatur einen kritischen Schwellwert unterschreitet.

Die SSN Ontologie ist mit Absicht sehr generisch gehalten wurden. Dies schlägt sich zum Beispiel in der Verwendung einer Upper-Ontologie wie SWEET nieder. Gleichzeitig ermöglicht es jedoch viele Freiheiten bei der Interpretation und Anwendung der Ontologie.
Durch die direkte Verknüpfung zwischen einer Eigenschaft und ggf. dem beinhaltendem Element innerhalb einer Beobachtung, eignet sich die SSN gut für die Beschreibung eines Messergebnisses. Die entsprechenden Beziehungen werden durch den Prozess des Messens vom Sensor hergestellt.
Die Beziehung, in der sich Sensoren, Eigenschaften und ggf. die sie beinhaltenden Objekte befanden, bevor eine Messung stattgefunden hat, lässt sich mit Hilfe der Sensor Semantic Network Ontologie jedoch nicht direkt beschreiben.
Für die Generierung von Sensormesswerten innerhalb einer Sensorsimulation sind es aber gerade diese Beziehungen, die von besonderer Bedeutung sind.

[2] `https://sweet.jpl.nasa.gov/`
[3] `http://www.semsorgrid4env.eu/`

2.3 Sensorfehler und Messungenauigkeiten

Eine besondere Eigenschaft von realen Sensoren, die bei der Sensordatenverarbeitung berücksichtigt werden muss, ist die Tatsache, dass sie immer einem gewissen Fehler unterliegen. Dieser kann sich zum einen auf die Messungenauigkeit der Ergebnisse beziehen, was in der Messtechnik auch als Messabweichung oder Messunsicherheit bezeichnet wird. Zudem können Sensoren über Fehler verfügen, die sich auf das System des Sensors an sich auswirken, wie beispielsweise der Ausfall des Sensors. In Bezug auf die zuvor vorgestellte Architektur, werden diese Messunsicherheiten und Sensorfehler hauptsächlich in der Sensorebene behandelt. Können sich aber unter Umständen auch in den oberen Ebenen Fortsetzen.
Die folgenden Abschnitte gehen genauer auf die, im weiteren Verlauf betrachteten, Sensorfehler ein.

2.3.1 Messtechnische Fehler

In der klassischen Messtechnik wird davon ausgegangen, dass eine Sensormessung immer über eine Messunsicherheit verfügt. Diese Messunsicherheit setzt sich dabei aus zwei Komponenten zusammen, dem zufälligen und dem systematischen Fehler.
Das Internationale Wörterbuch der Metrologie (VIM) [VIM08] definiert diese wie folgt[4]:

Systematischer Fehler Die Komponente der Messunsicherheit, die in wiederholenden Messungen konstant bleibt oder sich vorhersagbar verhält[5].

Zufälliger Fehler Die Komponente der Messunsicherheit, die sich in wiederholenden Messungen unvorhersehbar verhält[6].

Die Messunsicherheit (F) wird in diesem Fall durch die Abweichung von einem Referenzwert (R) angegeben[7].

$$F = R - (F_{systematisch} + F_{zufaellig}) \tag{2.1}$$

Dabei handelt es sich bei $F_{systematisch}$ um die systematischen Komponente und bei $F_{zufaellig}$ um die zufällige Komponente des Fehlers.
Abbildung 2.5 zeigt die beiden Fehlermodelle am Beispiel einer Positionsbestimmung.

[4] frei übersetzt
[5] Definiert in VIM 2.17
[6] Definiert in VIM 2.19
[7] Definiert in VIM 2.16

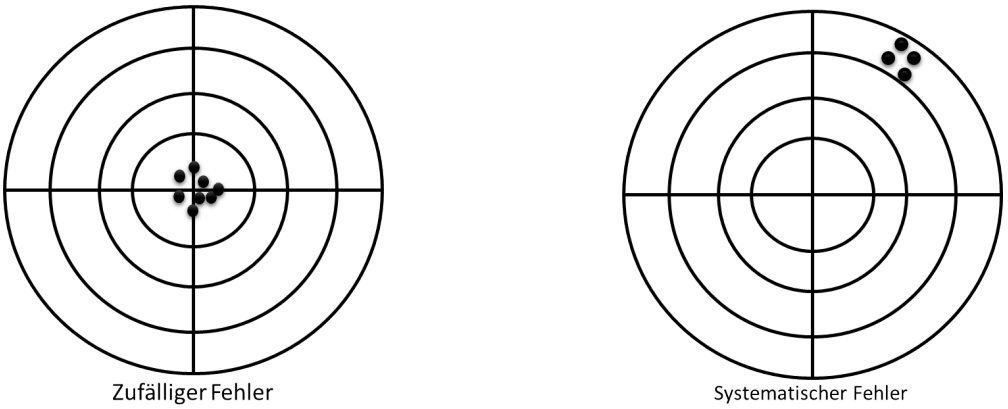

Abbildung 2.5: Darstellung von zufälligen (links) und systematischen (rechts) Fehlermodellen, am Beispiel einer Positionsbestimmung. Das Zentrum des jeweiligen Kreises gibt den tatsächlichen Wert an, die Punkte sind konkrete Messwerte.

In der Abbildung stellt das Zentrum des Kreises die tatsächliche Position des betrachteten Objektes dar, den Referenzwert. Die Punkte stellen jeweils das Ergebnis eines Messvorganges, zum Beispiel eines GPS Sensors, dar.

Die Ursachen für die beiden vorgestellten Fehlermodelle können vielseitig sein. Für systematische Fehler werden in [RJRV08] zum Beispiel falsch kalibrierte oder fehlerhafte Messgeräte genannt. Aber auch Umwelteinflüsse wie eine Temperaturabhängigkeit oder Alterungseffekte des Messgerätes können sich in Form eines systematischen Fehlers auf die Messungenauigkeit auswirken [KN12].
Als Ursachen für zufällige Fehler werden natürliche Prozesse angegeben, die sich auf die Messinstrumente auswirken.

Die Messunsicherheit, insbesondere der zufälligen Fehler wird in der Regel durch eine Wahrscheinlichkeitsverteilung der Messwerte um den Erwartungswert angegeben. Die Gauß'sche Normalverteilung ist hier eine häufig verwendete Wahrscheinlichkeitsverteilung, da sie verhältnismäßig einfach zu behandeln ist.
Es werden zunehmend auch nicht normalverteilte Wahrscheinlichkeiten bei der Fehlerbehandlung betrachtet. Im Umfeld von maritimen Radargeräten bzw. von SAR (Synthetic Aperture Radar) hat sich beispielsweise die K-Verteilung als Standardverteilung zur Beschreibung des Clutters etabliert [WWT06, Boc11].
Und auch für die Beschreibung des Positionsbestimmungsfehlers bei GPS Sensoren eigenen sich nicht normalverteilte Wahrescheinlichkeiten besser. Zu diesem Schluss kommt zum Beispiel die Untersuchung von Ted Driver [Dri07], in der verschiedene Wahrscheinlichkeitsverteilungen für die Positionsabweichung miteinander verglichen werden.

2.3.1.1 Auswirkung auf die Sensordatenverarbeitung

Ein sensordatenverarbeitendes System kann nicht davon ausgehen, dass es sich bei den von den Sensoren gelieferten Messwerten um die tatsächlich korrekten Werte handelt. Stattdessen muss er davon ausgehen, dass sie einer Messunsicherheit unterliegen.
Dabei stellt insbesondere die Erkennung der systematischen Komponente der Messungenauigkeit eine Herausforderung der Sensordatenverarbeitung dar. Im Gegensatz zu der zufälligen Komponente, die sich bei genügend häufiger Wiederholung der Messung theoretisch aufhebt, kann die systematische Messunsicherheit nicht ohne zusätzliches Wissen erkannt werden. Erst mit der Zuhilfenahme weiterer Sensoren oder statischen Wissens lassen sich solche systematischen Messfehler in den Messwerten erkennen.

Die Anwendung des zusätzlichen Wissens, bzw. die Behandlung der Messunsicherheiten findet bei der zuvor beschriebenen Architektur für Sensorfusionssysteme (vgl. Abschnitt 2.1), in der Regel innerhalb der Sensorebene statt.
Sie wird häufig unterstützt durch umfangreiche Messkampagnen, bei denen die Rauschcharakteristiken der verwendeten Sensoren, ermittelt werden [YB02, KE12, MMH05].
Andere Verfahren zur Rauschschätzung verwenden Online Verfahren um die Messwerte anhand der eingehenden Sensormesswerte zu bestimmen. So werden beispielsweise in der Arbeit von Liu et al. Verfahren beschrieben, wie das Rauschlevel anhand eines einzelnen Bildes ermittelt werden kann [LFSK06]. Christian Kuka beschreibt in seiner Dissertationsschrift [Kuk15] eine lernende Methode, welche auf Grundlage der eingehenden Datenströme in dem Datenstrom Management System Odysseus die statistische Verteilung der Messwerte lernt.

2.3.2 Sensorfehler

Ein Sensorfehler bezieht sich im Gegensatz zu den messtechnischen Fehlermodellen nicht auf einzelne Messwerte einer Messung, sondern auf die gesamte Messung.
Zu diesen Fehlern gehören Fehlermodelle, wie beispielsweise der Totalausfall eines Sensors durch einen technischen Defekt. Während dieser Fehler in der Sensordatenverarbeitung i.d.R. leicht erkannt werden kann, z.B. durch die Abwesenheit von erwarteten Messwerten, stellen andere Fehlermodelle dieser Klasse eine besondere Herausforderung für die Sensordatenverarbeitung dar.
Zu diesen Fehlermodellen gehört zum Beispiel die Fehlerklasse *Delay*, bei der der Sensor die Messergebnisse mit einer gewissen Verzögerung an die Sensordatenverarbeitung weiterleitet. Je nach Anwendungsfall und Zeitpunkt zu dem dieser Fehler auftritt, ist

er nur schwer von der Sensordatenverarbeitung zu erkennen, da nach wie vor Sensormesswerte geliefert werden, diese ggf. aber nicht den aktuellen Zustand der Umgebung repräsentieren.

Ein weiterer Fehler aus dieser Klasse der Sensorfehler ist der *Stuck-At* Fehler. Dieses Fehlermodell wird häufig im Zusammenhang mit der Entwicklung von eingebetteten Systemen betrachtet, wobei dort zwischen dem Stuck-at-1 und Stuck-at-0 Fehler unterschieden wird. Bei einem solchen Fehler verbleibt der logische Wert eines Gatters entweder bei einer logischen 0 oder einer logischen 1 obwohl sich eine Änderung ergeben müsste [HTI97].
Übertragen auf die Simulation von Sensoren werden bei diesem Fehler neue Messwerte des Sensors verworfen und stattdessen ein historischer Wert versendet. Ähnlich wie beim *Delay* Fehler ist dieser unter Umständen schwer durch die Sensordatenverarbeitung zu erkennen, da nach wie vor Messwerte kommuniziert werden.

2.3.2.1 Auswirkungen auf die Sensordatenverarbeitung

Die Auswirkungen von Sensorfehlern auf die Sensordatenverarbeitung unterscheiden sich stark von der Art des Sensorfehlers. Technische Defekte, wie beispielsweise ein zeitlich begrenzter oder ggf. auch Totalausfall, lassen sich i.d.R. einfach detektieren.
Delay bzw. *Stuck-At* Fehler dagegen stellen eine größere Herausforderung für das Sensordatenverarbeitende System dar, da nur mit zusätzlichen Wissen über die Umgebung gefolgert werden kann, dass beispielsweise gleichbleibende Sensorwerte eine Folge eines Fehlers und nicht einer entsprechenden Umgebungssituation sind.

2.3.3 Kontextsensitive Fehler

Die bisher betrachteten Sensorfehler und Sensormessungenauigkeiten treten i.d.R. zu jedem Zeitpunkt gleichermaßen auf, bzw. im Fall der Sensorfehler haben sie zu jedem Zeitpunkt eine gewisse Wahrscheinlichkeit, dass sie eintreten. Dabei wurde nicht berücksichtigt, dass Sensoren unter Umständen in bestimmten Situationen verstärkt zu Fehlern neigen oder ein verändertes Verhalten zeigen. Die folgende Definition beschreibt den hier verwendeten Terminus eines kontextsensitiven Fehlers zur Beschreibung dieses Sachverhaltes.

Kontextsensitiver Fehler. *Ein Sensorfehler ist kontextsensitiv, wenn seine Auftrittswahrscheinlichkeit oder seine Ausprägung von seinem internen oder externen Zustand abhängig ist.*

Die Definition setzt sich aus zwei Abschnitten zusammen. Zum Einen wird auf die Ursache des Fehlers eingegangen, zum Anderen auf dessen Auswirkungen.
Bei den Ursachen können sowohl interne als auch externe Zustände einen kontextsensitiven Fehler verursachen. Ein Beispiel für eine interne Ursache liefern Cang Ye und Johan Borenstein in ihrer Untersuchung des SICK LMS 200 Laserscanners [YB02]. Bei den Messungen des Laserscanners konnte beobachtet werden, dass er innerhalb der ersten drei Stunden nach dem Anschalten eine andere Messungenauigkeit bei der Entfernungsmessung aufwies, als in den Messungen nach diesem Zeitraum. Diese ca. 3-stündige Phase wird in der Arbeit als "Warm-up" Zeit beschrieben, wobei keine direkte Zuordnung zu der Betriebstemperatur des Sensors gemacht wurde.
In der gleichen Arbeit beschreiben Sie, einen weiteren kontextabhängigen / kontextsensitiven Fehler im Zusammenhang mit Laserscannern, der einer externen Ursache zugeschrieben werden kann.
Dieser Fehler wird als *"Mixed Pixel"* Problem bezeichnet und beschreibt, dass sich der Messwert an sehr scharfen Kanten eines vom Laserscanner beobachteten Objektes, aus der Entfernung des Objektes sowie der Entfernung des Hintergrundes zusammensetzt (vgl. Abbildung 2.6). In diesem Fall bezieht sich der Kontext des Sensorfehlers sowohl

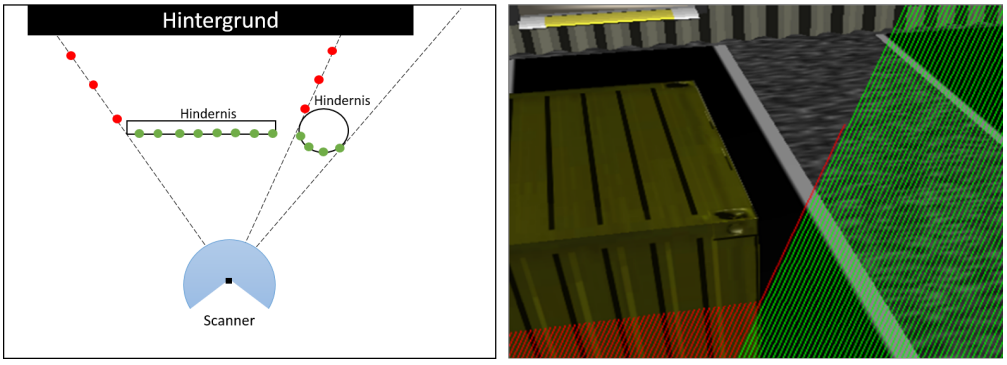

Abbildung 2.6: Darstellung des Mixed Pixel Messfehlers bei Laserscannern; Links: Schematische Darstellung mit korrekten Messungen (grün) und Fehlmessungen (rot) (Angelehnt an [TVH05]); Rechts: Simulierter Mixed Pixel Fehler an den Kanten eines Containers.

auf die Form des Objektes, als auch die Position des Sensors in Relation zum Objekt, da dieser Fehler nur auftritt, wenn eine scharfe Kante des Objektes beobachtet wird. Eine genauere Betrachtung dieses Effektes, auch mit dem Hintergrund der automatischen Elemination kann in der Untersuchung "Analysis and Removal of Artifacts in 3-D LIDAR Data" von J. Tuley et. al. gefunden werden [TVH05].
Ein weiterer kontextabhängiger Effekte im Zusammenhang mit Laserscannern ist die Änderung der Reflektivität einer Oberfläche, wenn diese feucht ist oder über einen ungünstigen Reflexionswinkel verfügt [RHE11].

In vielen Fällen, u.a. auch bei dem *Mixed Pixel* Problem, lassen sich die kontextsensitiven Fehler auf ein Zusammenspiel des vom Sensor verwendeten Messverfahrens sowie Eigenschaften des beobachteten Objektes zurückführen.
Andere kontextabhängige Sensorfehler müssen dagegen unabhängig von dem beobachteten Objekt betrachtet werden. Ein Beispiel für einen solchen kontextsensitiven Fehler bietet der sogenannte Mehrwegefehler bei GPS Messungen (vgl. Abbildung 2.7). In diesem Fall interagiert das zu messende Signal mit Objekten in der Umgebung des Sensors.

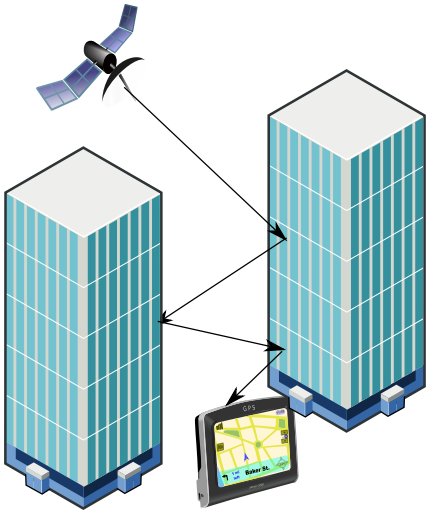

Abbildung 2.7: Schematische Darstellung des GPS Mehrwegefehlers in Urbanen Gebieten.

Genau wie die Ursachen für kontextabhängige Fehler verschiedene Gründe haben kann, lassen sich auch die Auswirkungen dieser Fehler mittels verschiedener Methoden beschreiben. Im Fall der zuvor beschriebenen “Aufwärmphase“ des SICK LMS 200 Laserscanners verstärkt sich die zufällige Komponente der Messunsicherheit.
Eine andere mögliche Auswirkungen kann zum Beispiel die Erhöhung des Ausfallrisikos darstellen, beispielsweise wenn ein Sensor an den Grenzen seiner Betriebstemperatur arbeitet oder generell in einer unwirtlichen Umgebung eingesetzt wird.

2.3.3.1 Auswirkung auf die Sensordatenverarbeitung

Für die Sensordatenverarbeitung stellen kontextabhängige Fehler eine Herausforderung dar, da sie nicht in jeder Situation auftreten und somit gegebenenfalls schwer zu reproduzieren sind. Zudem können sie einen großen Einfluss auf die Qualität der Messwerte ausüben.
Ihren Einfluss auf die Qualität der Sensordatenverarbeitung lässt sich unter bestimmten Bedingungen vorhersagen. C. Kuka stellt zum Beispiel in seiner Promotion [Kuk15, KN12] neben der lernenden Methode zur Fehlerschätzung einen Ansatz vor, bei dem die Qualität von Sensormessungen mit Hilfe verschiedener anderer Sensoren bestimmt werden kann.
Dabei werden zusätzliche Sensoren verwendet, um den Kontext eines Hauptsensors in seiner Umgebung zu detektieren. Auf dieser Grundlage kann die Vertrauenswürdigkeit des Hauptsensors und damit die Qualität der darauf aufbauenden Sensorfusion genauer abgeschätzt werden.

2.3.3.2 Modellierungsansätze

Im Abschnitt 3.3 werden verschiedene Simulationsframeworks vorgestellt, die zum Teil kontextabhängige Fehler betrachten. Dabei wird i.d.R. die Betrachtung des Fehlers direkt in das Messverfahren des Sensors integriert. Zum Beispiel indem direkt auf die veränderten Reflexionseigenschaften einer feuchten Oberfläche reagiert wird.
Die Integration der Fehler in die Sensormesswertsynthese bietet den Vorteil, dass sich die kontextsensitiven Fehler direkt in den Sensormesswerten widerspiegeln. Allerdings resultiert diese Integration der Fehler im Allgemeinen in einem komplexeren Prozess bei der Erzeugung von Messwerte, während der Sensorsimulation. Weiterhin kann durch die korrekte Abbildung des kontextabhängigen Fehlers im Sinne einer physikalisch korrekten Simulation die Laufzeitkomplexität der Sensormesswertgenerierung stark ansteigen.
Dieser Effekt kann zu einem gewissen Anteil umgangen werden, wenn die kontextabhängigen Fehler in eine Nachbearbeitung ausgelagert werden. Dies erfordert die Fähigkeit der Sensorsimulation, den Zustand des Sensors während der Laufzeit zu erfassen.

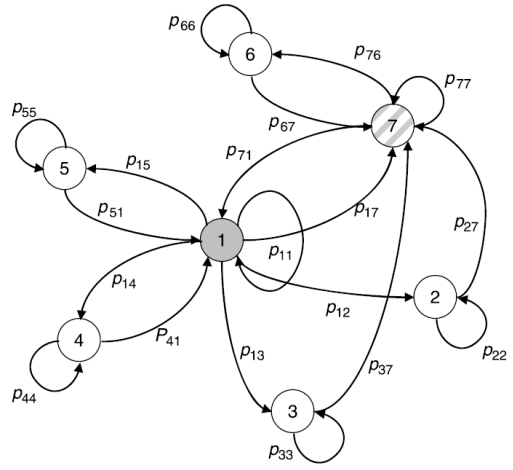

Abbildung 2.8: Markov Kette zur Modellierung von Fehlern in einer Abwasserreinigungsanlage. (aus [RJRV08])

Das Paper "Adding realism to simulated sensors and actuators" von Rosen et. al. beschreibt die Modellierung kontextabhängiger Fehler in Form von Markov Ketten, wie sie in Abbildung 2.8 dargestellt sind [RJRV08]. Dazu werden verschiedene interne Zustände betrachtet in denen sich die Sensoren, in diesem Fall Sensoren zur Überwachung und Steuerung einer Abwasserreinigungsanlage, befinden können. Jeder dieser Zustände beschreibt eine Ausprägung einer der zuvor genannten Fehlerklassen. Zum Beispiel wird der normale Operationsmodus mit einem zufälligen und einem systematischen Fehler beschrieben. Ausgehend vom normalen Operationsmodus kann der Sensor mit einer vordefinierten Wahrscheinlichkeit in einen Zustand übergehen, der in dem von Rosen et. al. verwendeten Beispiel als "Excessive Drift" (übermäßige Abweichung) bezeichnet wird. Innerhalb dieses Zustandes werden ebenfalls zufällige Messungenauigkeiten auf die Messwerte angewendet, jedoch mit einer veränderten Stärke. Weitere Zustände, die

aus dem normalen Operationsmodus erreicht werden können, beschreiben den kompletten Ausfall oder einen Stuck-At Fehler[8], wie sie hier bereits als Sensorfehler eingeführt worden sind.

Gemäß der Definition einer Markov Kette werden in der Arbeit von Rosen et. al., ausschließlich Wahrscheinlichkeiten für den Wechsel zwischen zwei Fehlerzuständen modelliert. Eine Erweiterung dieses Modells zu einem kontextsensitiven Fehler könnte so aussehen, dass neben den Wahrscheinlichkeiten auch Informationen aus dem Kontext des Sensors für einen Zustandsübergang verwendet werden.

2.4 Zusammenfassung

Dieses Kapitel hat sich allgemein mit Sensorsystemen beschäftigt. Dabei wurde zunächst ein kurzer Überblick über das zu überprüfende System, die Sensordatenverarbeitung gegeben. Anhand einer generellen Drei-Schichten-Architektur für sensordatenverarbeite Systeme wurden mögliche Aufgaben einer Sensordatenverarbeitung beschrieben, die der hier vorgestellten Ansatz überprüfen können soll.
Weiterhin wurde im Abschnitt 2.3 auf das Problem der Messunsicherheiten und Sensorfehler eingegangen, welche eine besondere Herausforderung für die Entwicklung von, zuverlässigen und sicheren, sensordatenverarbeitenden Systemen darstellen und somit ebenfalls von dem Ansatz unterstützt werden muss.
Zudem wurde mit der Sensor Semantic Network (SSN) Ontologie ein Modellierungsansatz für Sensorsysteme vorgestellt, mit dem eine größere Interoperabilität zwischen verschiedenen Sensordatenverarbeitenden Systemen hergestellt werden soll. Gleichzeitig wurden Schwachpunkte in dem vorgestellten Modellierungsansatz identifiziert, die eine Anpassung für die Modellierung von simulierten Sensoren notwendig macht. Das in dem Absatz 2.2.3 eingeführte *Stimulus - Sensor - Observation* Pattern stellt die Grundlage für die Erweiterung zu einem Simulationssystem dar.

[8] in dem Paper [RJRV08] wird der Stuck-At Fehler als “Fixed Value“ Fehler bezeichnet

Kapitel 3

Ansätze zur Sensordatengenerierung & Überprüfung

Nachdem im vorhergegangenen Kapitel Sensoren und sensordatenverarbeitende Systeme vorgestellt worden sind, beschäftigt sich dieses Kapitel mit existierenden Ansätzen zur Erzeugung von Sensormesswerten, sowie der Bewertung von sensordatenverarbeitenden Systemen. Die dabei erzeugten Sensordaten sollen gemäß des in Abschnitt 1.3 vorgestellten Vorgehens die realen Sensormessungen während der Entwicklung von neuen sensordatenverarbeitenden Systemen bzw. bei deren Überprüfung ersetzen.

Zum Zweck der Bereitstellung von Sensormesswerten werden im Folgenden zunächst zwei unterschiedliche Ansätze vorgestellt: das Bereitstellen von Sensormessungen aus historischen Daten sowie das Bereitstellen von Werten aus einer Simulation. Für beide Varianten werden jeweils die Vor- und Nachteile gegeneinander abgewogen.
Anschließend wird auf die verschiedenen möglichen Abstraktionslevel eingegangen, die bei der Verwendung von Simulation zur Verfügung stehen. Diese werden jeweils, in Hinblick auf das in Abschnitt 1.2, beschriebene Ziel, dieser Arbeit, bewertet.
Den Abschluss bildet ein Abschnitt über existierende Systeme, die in der Lage sind Sensordaten zu simulieren.

3.1 Manipulation historischer Messwerte

Die erste Methode zur Messwertsynthese basiert auf der Manipulation von historischen Daten.

Dafür müssen zunächst Messwerte mit dem nachzubildenden Sensor aufgezeichnet werden. Im Rahmen des eigentlichen Experimentes werden die aufgezeichneten Messwerte wieder abgespielt. Leichte Abwandlungen der ursprünglichen Messwerte, zum Beispiel durch das Hinzufügen von Zufallszahlen, können den Eindruck neuer Datensätze vermitteln.

Die wesentliche Stärke dieses Ansatzes liegt in seiner einfachen Umsetzung. In der Regel genügt ein einfacher Datenrekorder mit dem die originalen Messwerte aufgenommen werden. In der Regel lassen sich Teile der späteren Sensordatenverarbeitung als initialer Datenrekorder verwenden und der Realisierungsaufwand auf die Implementierung eines entsprechenden Encoder und Decoder für ein zuvor festgelegtes Datenformat beschränken.
Ein weiterer Vorteil dieses Verfahrens ist der Realismus der aufgenommenen Daten. Da es sich um Aufnahmen von realen Sensoren handelt, die ggf. unter realistischen Bedingungen getätigt wurden, sind etwaige Messungenauigkeiten bereits in den Sensordaten vorhanden.
Unter anderem aus diesen beiden Gründen handelt es sich bei der Verwendung von historischen Messwerten um eine gerne verwendete Methode zum Entwickeln und Testen von neuen sensordatenverarbeitenden Systemen.

Dem Vorteil der einfachen Realisierung dieser Methode stehen verschiedene Nachteile gegenüber: Zum einen lässt sich mit diesem Verfahren nur eine begrenzte Anzahl an Testszenarien realisieren. Für jede (inhaltliche) Variation des Szenarios müssen entweder neue Messkampagnen gestartet oder die vorhandenen Daten manipuliert werden. Zudem lassen sich nicht für alle zu überprüfenden Situationen entsprechende Messkampagnen realisieren, insbesondere wenn es sich um für Mensch und/oder Material kritische Situationen handelt.
Zum anderen ist diese Methode auf die Verfügbarkeit der verwendeten Sensoren beschränkt. Neue bzw. in der Entwicklung befindliche Sensoren lassen sich mit dieser Methode nicht nachbilden.
Der dritte und letzte hier behandelte Nachteil der Erzeugung von Sensormesswerten aus historischen Daten betrachtet die in den Sensormessungen vorhandenen Messungenauigkeiten. Da die Sensorwerte durch Messkampagnen mit realen Sensoren ermittelt wurden, unterliegen die aufgenommenen Werte den Messungenauigkeiten der verwendeten Sensoren (vgl. Abschnitt 2.3). Zwar handelt es sich im Bezug auf die Realitätstreue der Sensoren um einen Vorteil, kann aber als Nachteil für die Bewertung der Sensordatenverarbeitung angesehen werden, da die Daten zunächst von den Messungenauigkeiten bereinigt werden müssen. Dieser Prozess erfolgt häufig manuell unter Zuhilfenahme zusätzlicher Sensormesswerte, zum Beispiel von Video- und Fotomaterial.

Da es sich bei der Bereinigung der Datensätze i.d.R. um einen sehr aufwendigen Prozess handelt, haben sich in einigen Bereichen der Sensordatenverarbeitung Communities gebildet, von denen vorverarbeitete Datensätze bereitgestellt werden. Einer der bekannteren Datensätze (vgl. Abbildung 3.1) wird vom Middlebury College[1] bereitgestellt [SS02, SHK+14].

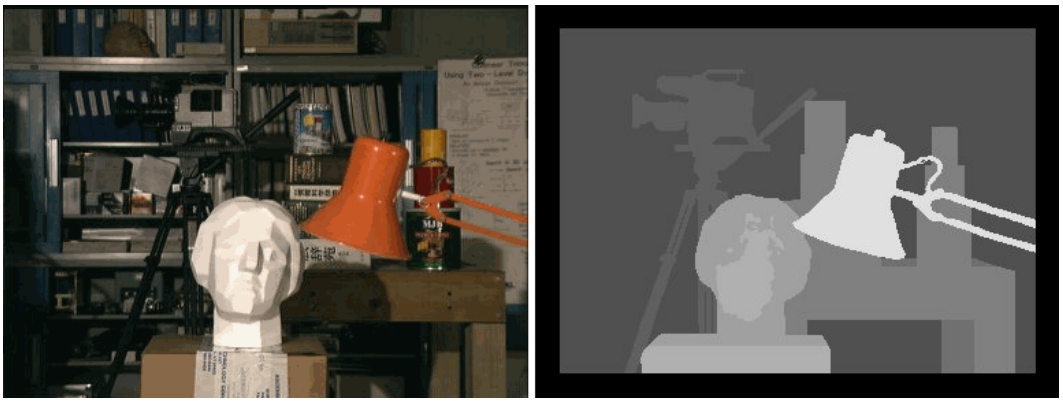

Abbildung 3.1: Beispieldatensatz für die Stereorekonstruktion (links) linkes Kamera Bild; (rechts) Bereinigte Tiefeninformationen [SS02].

Bei den bereitgestellten Daten handelt es sich um Stereokameraaufnahmen (linkes und rechtes Teilbild) sowie eine bereinigte Disparitätskarte (Tiefenkarte). Insbesondere die Disparitätskarte kann verwendet werden, um die Güte neuer Stereoalgorithmen zu bewerten. Dies gilt sowohl in Bezug auf die Genauigkeit der ermittelten Disparitäten, als auch in Bezug auf die Verarbeitungsgeschwindigkeit.

Ähnliche Datensätze existieren u.a. für die Analyse des optischen Flusses [GLSU13, MG15] oder die Analyse von Laserscanner Messungen [BMG09, BMGJ14].

3.2 Beobachtung einer simulierten Umgebung

Die zweite Methode zur Bereitstellung von Sensormesswerten beruht auf der Beobachtung einer virtuellen Umgebung.
Damit orientiert sie sich im Ansatz an der Funktionsweise von realen Sensoren, die ebenfalls durch Beobachtung physikalischer Effekte in einer realen Umgebung ihre Messwerte ermitteln.

Simulierte Sensoren greifen dabei auf die beschreibenden Eigenschaften von in der Umgebung vorkommenden Objekten zu und wandeln die ermittelten Eigenschaften in Sensormesswerte um. Weiterhin lässt sich durch die Eigenschaften von Objekten beziehungsweise ihre räumliche Beziehung untereinander der aktuelle Kontext eines simulierten

[1] `http://vision.middlebury.edu/stereo/`

Sensors ermitteln.
Soll beispielsweise die Position eines Objektes mittels GPS Sensor ermittelt werden, würde eine entsprechende Sensorsimulation die Position des Objektes auslesen und wenn notwendig in eine GPS Koordinate umwandeln.
Der Kontext der in dem zuvor genannten Beispiel zu einer Mehrfachreflexion führen kann, kann dabei ebenfalls durch eine räumliche Anfrage der unmittelbaren Umgebung des beobachteten Objektes ermittelt werden. Werden in unmittelbarer Nähe des Objekts große Objekte erkannt, kann darauf geschlossen werden, dass diese zu einer Mehrfachreflexion führen würden.

Der Vorteil dieser Methode liegt in der konzeptionellen Nähe zu der tatsächlichen Funktionsweise realer Sensoren, die wie in den Abschnitten 2.2 & 2.2.3 beschrieben, ebenfalls Eigenschaften von Objekten und Phänomenen bestimmen. Zudem lässt sich der Kontext eines Sensors verhältnismäßig leicht bestimmen und somit bei der Generierung der Messwerte bzw. deren Messungenauigkeiten berücksichtigen. Dies führt insbesondere in der Kombination von verschiedenen Sensoren, die das gleiche Phänomen beobachten, zu einer konsistenten Sichtweise auf die Umgebung.
Durch das Beobachten einer vollständig virtuellen Umgebung werden keine realen Messkampagnen benötigt. Somit können die generellen Vorteile von Simulationen ebenfalls zu den Vorteilen dieser Methode gezählt werden. Es lassen sich Szenarien überprüfen bzw. simulieren, die ein Risiko für Mensch oder Maschine darstellen und es können neue Sensoren und Sensorkonzepte nachgebildet werden.

Gegenüber der Verwendung von historischen oder mathematisch erzeugten Sensorwerten verfügt diese Methode über verschiedene Nachteile. So muss neben den Sensoren eine komplette Umgebung modelliert und simuliert werden, die alle beobachtbaren Elemente beinhaltet. Abhängig von dem geforderten Detailgrad der Simulation müssen dabei unter Umständen sehr komplexe Umgebungen simuliert werden, sowie ein hoher Detailgrad bei den verwendeten Elementen berücksichtigt werden.
Die entsprechenden Elemente müssen weiterhin über beobachtbare Eigenschaften verfügen, denen eine eindeutige Semantik zugeordnet werden kann. Um auf Grundlage dieser Eigenschaften Sensormesswerte generieren zu können, muss der simulierte Sensor weiterhin in die Lage versetzt werden, die konkreten Ausprägungen der Eigenschaften während der Simulationslaufzeit abzufragen.
Ein weiterer Faktor bei der Anwendung dieser Methode ist die Tatsache, dass die simulierten Sensoren dazu neigen, perfekte Messwerte zu erzeugen. Die dadurch idealisierten Messwerte müssen entweder im Nachhinein mit einer Messungenauigkeit belegt werden oder es muss die vereinfachte Verarbeitung der Sensormesswerte durch die Sensordatenverarbeitung in Kauf genommen werden.

3.2.1 Detailgrad der Simulation

Die Simulation von Sensormesswerten kann grob in drei Bereiche aufgeteilt werden, die sich hauptsächlich in dem Detailgrad der gelieferten Ergebnisse und damit in ihrem Einsatzzweck unterscheiden. Diese Einteilung ist angelehnt an die Einteilung, welche von M. Brooker [Bro08] in seiner Doktorarbeit zur Einteilung von Radarsimulatoren verwendet wurde. Sie lässt sich aber ohne weiteres auf allgemeine Sensorsimulationen ausweiten.

3.2.1.1 Signal Level Simulationen

Signal Level Simulationen versuchen, die vom Sensor gemessenen physikalischen Größen nachzubilden.
Dabei kann es sich zum Beispiel um die an einer Empfangsantenne gemessene Leistung in Folge einer Radarreflexion handeln, wie sie in der bereits erwähnten Dissertation von M. Brooker (vgl. [Bro08]) simuliert wird. In der Regel werden für diese Art der Simulation die physikalischen Zusammenhänge sehr genau nachgebildet.

Mit Hilfe der Signal Level Simulationen können Vorhersagen über das Verhalten sowie die Qualität von Sensoren unter bestimmten Umgebungsbedingungen abgeleitet werden. Durch diese Eigenschaft spielen Signal Level Simulationen eine große Rolle in der Entwicklung neuer Sensorsysteme. Insbesondere bei der Entwicklung von intelligenten bzw. smarten Sensoren, bei denen intern eine Vorverarbeitung der Daten vorgenommen wird. Diese lässt sich mithilfe einer Signal Level Simulation sowohl leichter entwickeln, als auch überprüfen.
Auf der anderen Seite führt die Nachbildung komplexer physikalischer Zusammenhänge häufig zu einer hohen Laufzeitkomplexität, wodurch die Simulationsergebnisse häufig nicht in Realzeit[2] berechnet werden, bzw. nur eine geringe Anzahl an Sensoren gleichzeitig simuliert werden können.

3.2.1.2 Statistische Simulation

Statistische Simulationen beschränken sich auf die Beschreibung der statistischen Zusammenhänge von Sensoren und ihrer Umgebung.

Mit Hilfe einer statistischen Simulation kann zum Beispiel die Wahrscheinlichkeit beschrieben werden, dass ein Hindernis von einem Sensor übersehen wird.

[2] In diesem Zusammenhang wird unter dem Begriff Realzeit die Zeit verstanden, wie sie auch in der Realität verstrichen ist. Teilweise wird diese Zeit auch mit dem englischen Begriff “Wall-clock time“ bezeichnet.

Diese Form der Simulation kann sehr effizient durchgeführt werden, wodurch sie sich für die Simulation von großen Sensorgruppen eignet. Die Genauigkeit der Simulation hängt dabei stark von den verwendeten Wahrscheinlichkeitsmodellen ab. Dabei bleiben physikalische Effekte in der Regel unberücksichtigt und auch die Interaktion mit der betrachteten Umgebung lässt sich i.d.R. nicht einfach modellieren, bzw. ist auf eine Klasse von Umgebungen beschränkt, für das die Wahrscheinlichkeitsmodelle ermittelt wurden.

3.2.1.3 Ergebnisorientierte Simulation

Ergebnisorientierte Simulationen versuchen, das von einem Sensor zu erwartende bzw. kommunizierte Messergebnis zu reproduzieren.
Dabei kann es sich im Falle von intelligenten Sensoren unter Umständen um bereits aggregierte Informationen handeln. Im Fall einer Radarsimulation würde eine ergebnisorientierte Sensorsimulation anstelle der eingehenden Leistung die Entfernung und ggf. den Winkel zu einem Hindernis bereitstellen. Eine ergebnisorientierte Simulation ist nicht darauf angewiesen, die Messwerte auf die gleiche Weise zu bestimmen, wie es von einer physikalischen Simulation erwartet werden kann. Stattdessen kann sie auf symbolische Informationen zurückgreifen.

Aufgrund des geringeren Detaillevels kann eine ergebnisorientierte Simulation in der Regel sehr effizient durchgeführt werden. Dadurch eignet sie sich insbesondere für das Bereitstellen von Sensordaten im Rahmen einer automatisierten Überprüfung von sensordatenverarbeitenden Systemen, bei der eine größere Anzahl an Sensoren verwendet werden soll.

Die ergebnisorientierte Sensorsimulation kann vom Abstraktionsgrad zwischen den Signal Level Simulationen und den statistischen Simulationen angeordnet werden. Sie profitiert dabei stark von der Übernahme einzelner Aspekte der beiden anderen Simulationslevel.
Durch die Berücksichtigung physikalischer Effekte können realistischere Messergebnisse produziert werden. Ein einfaches Beispiel wäre hier die Abstandsbestimmung zwischen einem Laserscanner, Radar oder Ultraschallsensor und einem großen Objekt, wie beispielsweise einem Schiff. Durch die Berücksichtigung der Geometrie des Objektes kann der geringste Abstand zwischen dem Sensor und der Oberfläche des Objektes bestimmt werden. Dabei handelt es sich bereits in erster Näherung um eine Berücksichtigung der auf der Reflexion von Wellen basierenden physikalischen Messprinzipien.
Auf der anderen Seite können physikalische Effekte, die einen geringen Einfluss auf die zu erwartenden Messergebnisse ausüben, durch statistische Modelle angenähert werden. Ein Beispiel für eine solche statistische Annäherung kann eine geringe Ungenauigkeit in

der Zeitmessung des Sensors sein. Die daraus resultierende Abweichung der errechneten Entfernung lässt sich durch ein nachträglich angewendetes Fehlermodell annähern.

3.3 Analyse bestehender Ansätze zur Sensordatengenerierung

Dieser Abschnitt beschäftigt sich mit der Analyse bestehender Lösungsansätze welche in Teilen die in Kapitel 1 genannten Anforderungen und Ziele dieser Arbeit unterstützten. Dabei konzentriert sich die Analyse zu einem großen Teil auf die Domäne der Robotik, in der sich die Verwendung simulationsbasierter Werkzeuge spätestens nach der Veröffentlichung des Player / Stage / Gazebo -Projektes [GVS+01], etabliert hat.

Der Abschnitt ist in drei Unterabschnitte unterteilt. Zunächst werden die Bewertungskriterien festgelegt. Anschließend werden Simulationswerkzeuge aus der Robotik Domäne hinsichtlich ihrer Zielerfüllung untersucht. Die untersuchten Robotikframeworks verfolgen i.d.R. das Ziel, die Entwicklung neuer robotischer Systeme zu vereinfachen. Zu diesen Systemen gehören gemäß dem in der Motivation vorgestellten Sense-Think-Act Paradigmas auch sensordatenverarbeitende Systeme, mit denen ein Abbild der Umgebung erstellt wird. Dementsprechend verfügen die untersuchten Frameworks über die Möglichkeit Sensorwerte im Sinne einer ergebnisorientierten Sensordatengenerierung bereitzustellen.
Neben diesen Werkzeugen werden einzelne Simulationssysteme betrachtet, welche das direkte oder indirekte Ziel verfolgen, eine Bewertung von sensordatenverarbeitenden Systemen durchführen zu können.

3.3.1 Bewertungskriterien

Die untersuchten Lösungsansätze und Simulationswerkzeuge werden anhand der folgenden Kriterien untersucht und bewertet, welche sich an den in der Motivation angegebenen Anforderungen und Zielen orientieren.

Unterstützte Sensoren Bei diesem Bewertungskriterium wird zunächst auf die Anzahl und Diversität der betrachteten bzw. simulierten Sensoren geachtet. Dies geschieht vor dem Hintergrund, dass im Bereich der Sensordatenverarbeitung / Sensordatenfusion ein immer stärker werdender Trend zur Verwendung von heterogenen Sensoren beobachtet werden kann [Stü04, KBF+13]. Bei diesen Ansätzen sollen die jeweiligen Schwächen einzelner Sensoren durch die Stärken anderer Sensoren ausgeglichen werden. Beispiele für solche sensordatenverarbeitenden Systeme lassen

sich wiederum in verschiedenen Domänen finden. So wurden im Forschungsprojekt SaLsA ([RN11]), Laserscanner, 2D und 3D Kameras, Lichtschranken, Ultraschall und Bewegungsmelder verwendet, um die sichere Steuerung von autonomen Fahrzeugen bei verhältnismäßig hohen Geschwindigkeiten zu erlauben. Dabei wurden die verschiedenen Sensoren sowohl mobil als auch stationär eingesetzt, was wiederum zu einer großen Anzahl an verwendeten Sensoren führte [KBF+13, EKR+11]. Neben der puren Anzahl an unterstützten Sensoren wird bei diesem Kriterium auch auf die Art der Sensoren geachtet. Beispielsweise behandeln viele der vorgestellten Ansätze Sensoren aus der Automotivedomäne, wie Laserscanner, Odometer und Ultraschall, wohingegen die maritime Domäne sowie die dort gebräuchlichen Sensoren, wie beispielsweise maritime Radare, Sonar und AIS (Automatic Identification System) größtenteils vernachlässigt werden.

Simulationslevel Im vorherigen Abschnitt 3.2.1 wurden drei Simulationslevel vorgestellt, die sich auf die Granularität beziehen ,mit der die Simulation durchgeführt wird. Unterschieden wurde zwischen den Leveln: stochastische Simulation, ergebnisorientierte und physikalische Simulation. Ebenfalls in Abschnitt 3.2.1 wurde festgestellt, dass sich für die Simulation und Bewertung von sensordatenverarbeitenden Systemen, vornehmlich die beiden letzten Simulationslevels eignen, weshalb in diesem Kapitel keine stochastischen Simulationen untersucht werden. In diesem Abschnitt wird auf die Simulationslevel geachtet, da sich aus ihnen i.d.R. die Größe der zu simulierenden Sensorpopulation ergibt, wobei das physikalische Simulationslevel i.A. auf eine kleine Anzahl an gleichzeitig zu simulierenden Sensoren schließen lässt.

Fehlermodelle Ein wichtiger Aspekt bei der Überprüfung von sensordatenverarbeitenden Systemen ist, deren Verhalten auf unterschiedliche Fehlermodelle von Sensoren überprüfen zu können, da wie in Abschnitt 2.3 beschrieben, in der Realität normalerweise keine idealen Sensormesswerte zur Verfügung stehen. Auf der anderen Seite kann es während der Entwicklung neuer Systeme von Vorteil sein, temporär auf die Anwendung von Fehlern zu verzichten oder diese sehr gering zu halten, um die generelle Arbeitsweise eines neuen Fusionsalgorithmus überprüfen zu können. Dementsprechend werden die untersuchten Systeme auf ihre Eignung hin untersucht, verschiedene Fehlermodelle abbilden und konfigurieren zu können um die durch sie herbeigeführten Effekte gezielt verstärken oder verringern zu können.

Kommunikationsfähigkeit Ein weiterer Aspekt bei der Entwicklung und Überprüfung von sensordatenverarbeitenden Systemen ist die Fähigkeit, mit dem zu testenden System zu kommunizieren. Dies beinhaltet hauptsächlich die Form der Kommunikation der Sensordaten mit der Sensordatenverarbeitung.

Dies gilt insbesondere, wenn Hardware In the Loop (HIL) Systeme oder Systeme von externen Partnern überprüft werden sollen. In diesen Fällen muss die Sensorsimulation an die Kommunikationsbedürfnisse des sensordatenverarbeitenden Systems angepasst werden können, bzw. als kleinsten gemeinsamen Nenner, die Kommunikationsschnittstellen der realen Sensorik nachbilden können.

Umgebungsmodellierung Ein Bestandteil der in Abschnitt 3.2.1.3 beschriebenen Ergebnisorientierten Simulation ist die Modellierung der Umgebung, in der die Experimente durchgeführt werden können.
Diese sollte soweit an das zu untersuchende Szenario angepasst werden können, dass aussagekräftige Simulationen durchgeführt werden können. Dies beinhaltet unter anderem die Fähigkeit, Rückschlüsse auf die beobachteten Elemente zu ermöglichen, was wiederum für die Bewertung der Sensordatenverarbeitung von Bedeutung ist.

Bewertungsfähigkeit Bei diesem Kritierium wird überprüft, ob mithilfe des untersuchten Systems eine Bewertung der Sensordatenverarbeitung durchgeführt werden kann. Ist dies nicht der Fall, wird überprüft, inwieweit die benötigten Informationen zur Verfügung stehen um eine entsprechende Lösung ggf. um diese Fähigkeit zu erweitern.

Intention Die Intention bzw. Motivation, mit der eine Lösung entwickelt wird, spielt eine große Rolle in Bezug auf die umgesetzten bzw. unterstützten Fähigkeiten. Somit kann die Intention einer der im Folgenden vorgestellten Lösungen, als ein Gewichtungsfaktor für die vorherigen Kriterien angesehen werden. Beispielsweise ist von einem Simulationssystem das sich auf die Simulation von Radarsensoren spezialisiert hat, keine große Sensordiversität zu erwarten.

3.3.2 Robotik Frameworks

Im Folgenden werden verschiedene Simulationsframeworks aus dem Bereich der Robotik betrachtet, mit denen die Simulation von Sensoren möglich ist. Der Bereich der Robotik setzt bereits seit mehreren Jahren erfolgreich auf den Einsatz von Simulation zur Unterstützung der Entwicklung und bei der Erprobung von neuen Systemen. Dabei liegt der Fokus nicht immer auf einer realitätsgetreuen Wiedergabe, sondern in der einfachen Bereitstellung von Sensordaten, mit deren Hilfe neue Steuerungsalgorithmen entwickelt werden können.

3.3.2.1 Player / Stage / Gazebo

Bei dem Tripel *Player/Stage/Gazebo (PSG)* [GVS+01, GVH03, CMG05] handelt es sich um einen verhältnismäßig alten Vertreter aus dem Bereich der Robotikframeworks. Begonnen wurde das Projekt um die Jahrtausendwende mit dem Ziel, die Entwicklung von robotischen Systemen voranzutreiben.
Dabei handelt es sich bei dem Projekt *Player* um eine Middleware, die den Zugriff auf verschiedene Hard- und Softwaresysteme vereinheitlichen soll. Durch dieses Vorgehen soll eine leichtere Entwicklung sowie eine höhere Übertragbarkeit von entwickelten Softwaremodulen ermöglicht werden.
Bei dem Teilprojekt *Stage* handelt es sich dagegen um einen zweidimensionalen Simulator, der unter anderem in der Lage ist, verschiedene Sensoren und Aktuatoren zu simulieren.
Das dritte Teilprojekt *Gazebo*, ist eine Erweiterung des *Stage* Simulators, der in die dritte Dimension einführt [KH04, AKC+15].
Trotz des hohen Alters wird das Player/Stage/Gazebo Tripel nach wie vor angewendet [KHS12]. Dies liegt unter anderem an der großen Community und der damit verbundenen Anzahl an Erweiterungen, die sich dadurch ergeben haben.

PSG unterstützt verschiedene Klassen von Sensoren, hauptsächlich aus dem klassischen Umfeld der Umgebungswahrnehmung. Zu den unterstützten Sensoren gehören Laserscanner, Hodometer, Lichtschranken, Ultraschall, Inertialsensorik und Kameras. Dabei konzentriert sich das *Gazebo* Teilprojekt zu einem großen Teil auf die Kamerasimulation, die in der 2D bzw. 2,5D[3] Simulation *Stage* eher vernachlässigt wird.
Die Unterstützung von Sensorfehlern liegt nicht im Hauptfokus des PSG Projektes, vielmehr sollen Sensordaten generiert werden, um eine einfache Erprobung neuer Algorithmen in einer virtuellen Umgebung zu ermöglichen, sowie den anschließenden Umstieg auf eine reale Sensorplattform zu erleichtern. Dies soll dadurch erreicht werden, dass das verwendete Kommunikationsinterface (Player) sowohl bei der virtuellen als auch bei der physikalischen Anwendung eingesetzt werden kann.

Eine Zusammenfassung der Bewertungskriterien kann in Tabelle 3.1 nachgelesen werden.

3.3.2.2 Robot Operation System

Bei dem Robot Operation System (ROS) handelt es sich wie bei PSG ebenfalls um eine Middlewarelösung mit der die Entwicklung von (autonomen) robotischen Systemen

[3] es wird von 2,5 Dimensionalen Geometrien gesprochen, wenn die dritte Dimension nicht vollständig berücksichtigt und stattdessen der gesamten Geometrie nur eine Höhe zugewiesen wird. [Tur97]

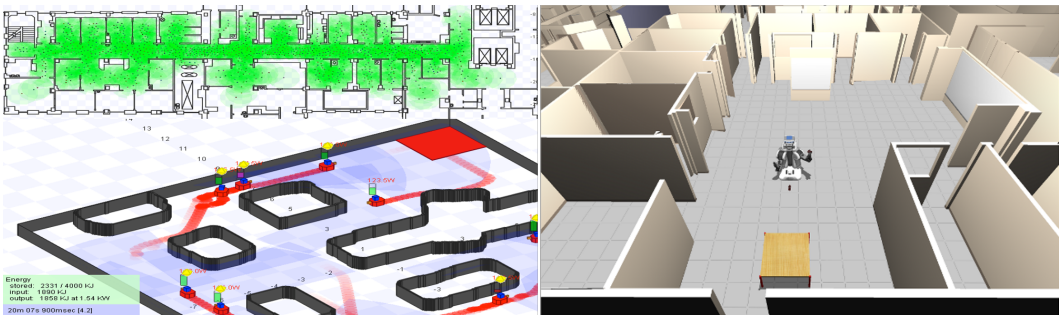

Abbildung 3.2: Umgebungsrepräsentation von *Stage* (links; Bildquelle: `http://playerstage.sourceforge.net/index.php?src=stage`) und *Gazebo* (rechts; Bildquelle: [KH12]).

unterstützt werden soll [QCG+09]. Dabei entwickelt sich ROS aktuell als ein Nachfolger von *Player/Stage/Gazebo* mit einer schnell ansteigenden Community. ROS verfolgt dabei einen etwas anderen Ansatz als PSG, indem es versucht, verschiedene existierende robotische Systeme unter einer Middleware zu vereinen. In diesem Sinne stellt ROS auch keinen eigenen Sensorsimulator zur Verfügung, sondern verwendet unter anderem *Stage* und *Gazebo* für die Generierung von Sensormesswerten.

3.3.2.3 Mobile Robot Programming Toolkit

Das Mobile Robot Programming Toolkit (MRPT)[4] versteht sich als eine Sammlung von Anwendungen und Bibliotheken, die die Entwicklung von robotischen Applikationen vereinfachen sollen [Cla08]. Dazu wird auch an dieser Stelle eine Abstraktion der verwendeten Hardware vorgenommen. Im Gegensatz zu PSG und ROS handelt es sich allerdings nicht um eine Middlewarelösung die zur Kommunikation zwischen verschiedenen Komponenten verwendet wird. Stattdessen stellt das MRPT eine API zur Verfügung mit der einzelne Komponenten entwickelt werden können.

Zum Testen von neuen Algorithmen stellt das MRPT ebenfalls einfache Simulatoren zur Verfügung, mit dessen Hilfe zum Beispiel Laserscannermesswerte synthetisiert werden können. Neben den Verarbeitungsbibliotheken stellt das MRPT Visualisierungskomponenten zur Verfügung über die die Ergebnisse einer Sensordatenverarbeitung (und auch die einer Sensorsimulation) angezeigt werden können.

Die Umgebungsmodellierung der Simulatoren beschränkt sich dabei im Wesentlichen auf die Angabe von Hindernissen in einer zweidimensionalen Umwelt, welche wiederum in Form einer Rasterkarte bzw. einem Graustufenbild angegeben werden. Neben der einfachen Rasterumgebung, wie sie in Abbildung 3.3 dargestellt ist, können verschiedene

[4] `http://www.mrpt.org/`

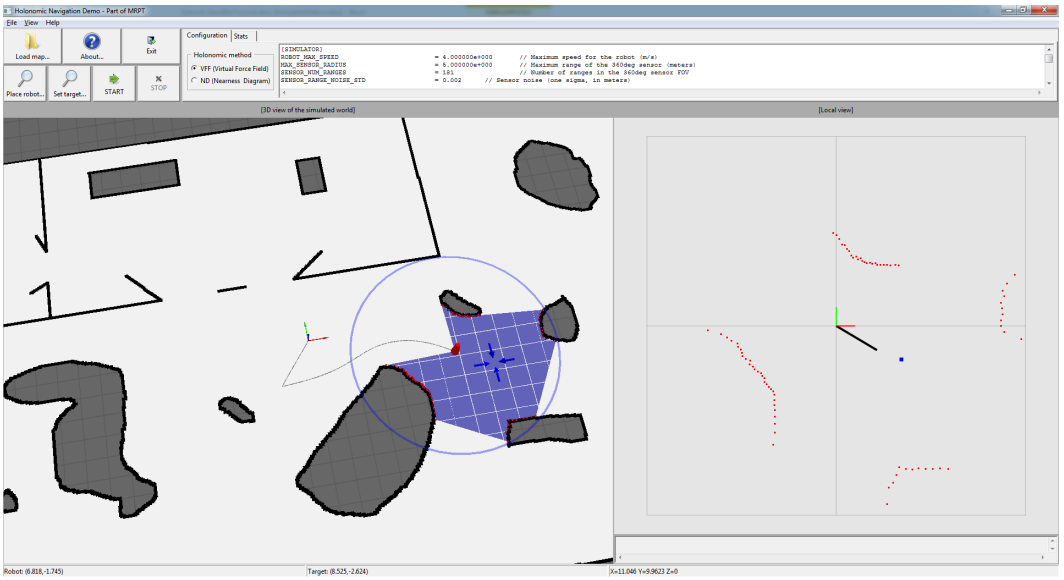

Abbildung 3.3: Visualisierung des MRPT Laserscanner Simulators. Die Umgebung wird durch eine Rasterkarte dargestellt. Linker Teil: Visualisierung des Roboters in der simulierten Umgebung; Rechter Teil: Visualisierung des aktuellen Scans.

Landmarken simuliert werden, die von einigen SLAM[5] Algorithmen (z.B. Hochdorfer und Schlegel [HS10]) für die Positionsbestimmung verwendet werden.

Eine Zusammenfassung der Bewertungskriterien für das Mobile Robot Programming Toolkit kann in der Tabelle 3.2 nachgelesen werden.

3.3.2.4 USARSim

Bei dem Simulationswerkzeug USARSim handelte es sich lange Zeit um das Simulationswerkzeug der RoboCup Rescue League[6], bei der autonome Roboter verschüttete Personen nach einer Naturkatastophe finden sollen [CLW+07, SV09]. Gleichzeitig ist USARSim ein hervorragendes Beispiel für die das Stichwort “Serious Gaming“, da dieser Simulator auf dem kommerziellen Spiel Unreal Tournament aufsetzt. Dabei handelte es sich in den frühen 2000 Jahren um ein beliebtes 3D Egoshooter- Spiel. Ein Vorteil, der sich aus der Verwendung eines kommerziell betriebenen 3D Spiels ergeben hat, ist die für die damalige Zeit sehr gute grafische Leistungsfähigkeit mit der auch optische Sensorsysteme, wie Kamerasysteme gut nachgebildet werden konnten. Neben den Kamerasystemen stehen Entfernungsmesser wie Laserscanner, Sonar oder Ultraschall, Kontaktsensoren und Positionssensoren zur Verfügung.
Abgesehen von einigen Ausnahmen lassen sich einfache Fehlermodelle auf die synthetisch

[5] Simultaneous Localization and Mapping
[6] http://www.robocuprescue.org/

hergestellten Sensormesswerte anwenden, wobei dieser Fehler i.d.R. einem einfachen Modell folgt [WB08]:

$$Output(X) = X + RNG(min, max) * X \tag{3.1}$$

Die Anwendung des Fehlers erfolgt direkt bei der Erzeugung der Daten und berücksichtigt, dass bestimmte Attribute der Beobachtung als nicht veränderlich angesehen werden. Als Beispiel wird in der Dokumentation von USARSim der Abstand zwischen zwei Winkeln eines Laserscans angegeben, der nicht durch die Fehlermodelle verändert werden kann.
Die simulierte Umgebung von USARSim (vgl. Abbildung 3.4) besteht im Allgemeinen aus einem Set an statischen Objekte, die die Umwelt (z.B.: Gebäude) repräsentieren, den simulierten Robotern und einem oder mehreren Verletzten, die während einer USAR (Urban Search And Rescue) Simulation gefunden werden sollen.

Die Tabelle 3.3 fasst die Bewertungskriterien für den USARSim Simulator nocheinmal zusammen.

Abbildung 3.4: Abbildung einer USAR Simulation in USARSim [CLW+07])

3.3.3 Sensorsimulationen

Neben den bisher behandelten Robotik Frameworks, welche i.d.R. in der Lage sind, Sensorwerte zum Zweck der Datenbereitstellung zu generieren, existieren verschiedene Simulationen, die sich auf die Generierung von Sensorwerten spezialisiert haben. Im Folgenden soll eine Auswahl dieser Sensorsimulationssysteme vorgestellt und gemäß der zuvor beschriebenen Kriterien analysiert werden.

3.3.3.1 VANE

VANE steht für "Virtual Autonomous Navigation Environment" und wurde von den US Streitkräften zur Vereinfachung der Entwicklung neuer Algorithmen in der Robotik entwickelt [RCT+09, GDG+10] Bei der Entwicklung von VANE wurde ein besonderer Fokus auf die Realitätsnähe der generierten Sensoren geachtet. Dies wird durch eine sehr starke Anlehnung an die physikalischen Messprinzipien der einzelnen Sensoren erreicht. Der dadurch benötigte Rechenaufwand wird von einem Supercomputer (Cray XT4) zur Verfügung gestellt, der eine extreme Parallelisierung ermöglicht. Diese Parallelisierung wird unter anderem für die Berechnung von Laserscannern sowie von CCD Kameras verwendet, wobei ein extra zu diesem Zweck Entwickelter Raycaster (Quick Caster) zum Einsatz kommt.
Die simulierten Laserscanner werden zum Beispiel durch eine Vielzahl von einzelnen Strahlen approximiert, wodurch sich Effekte wie die Aufweitung eines Laserstrahls mit zunehmender Entfernung, nachbilden lassen. Gleichzeitig werden Effekte wie der in Abschnitt 2.3.3 beschriebene Mixed Pixel Fehler bei der Berechnung der Laserscans berücksichtigt.

Für die Visualisierung der Ergebnisse wird eine vereinfachte Visualisierungssoftware verwendet, die auf einem handelsüblichen Computer ausgeführt werden kann (ANVEL - Autonomous Navigation Virtual Environment Laboratory).

Eine Zusammenfassung der Bewertungskriterien für die VANE Simulation kann in Tabelle 3.4 nachgelesen werden.

3.3.3.2 Tunnelsimulator

Der Tunnelsimulator [VHVS+11] (vgl. auch Tabelle 3.5) ist einer der wenigen Sensorsimulationen, die mit dem Ziel entwickelt werden, eine sensordatenverarbeitende Software (automatisch) zu überprüfen. Damit besitzt der Tunnelsimulator eine sehr ähnliche Zielsetzung wie auch die vorliegende Arbeit.

Gleichzeitig verfolgt der Tunnelsimulator einen vereinfachten Ansatz, indem die Sensordaten im Vorfeld der Analyse berechnet werden. Dadurch lassen sich die simulierten Szenarien auf eine deterministische Weise wiederholen. Auf der anderen Seite können keine Rückkopplungen von dem untersuchten System (wenn es sich zum Beispiel um ein Assistenzsystem handelt) vorgenommen werden. Durch dieses Vorgehen verbindet der Tunnelsimulator die zuvor vorgestellten Ansätze zur Sensordatengenerierung aus Erzeugung von neuen Sensordaten mit der Methode der historischen Aufnahmen. Abgesehen von der bereits erwähnten Einschränkung, dass während der Laufzeit der Simulation kein Einfluss auf die Erzeugung der Sensordaten genommen werden kann, versucht dieses Verfahren die Vorteile beider Methoden miteinander zu vereinen.

3.3.3.3 CViewLab

Im Gegensatz zu den bisher vorgestellten Frameworks und Sensorsimulationen, von denen sich die meisten mehr oder weniger stark auf die Simulation von Bodenfahrzeugen und deren Sensoren konzentriert haben, handelt es sich bei der CViewLab Simulation des Frauenhofer Institutes um eine Simulation in der maritimen Domäne [GPJ+09]. Simuliert werden autonome Unterwasserfahrzeuge. Das CViewLab wurde mit dem Ziel entwickelt, eine Testumgebung für die Entwicklung von Steuerungsalgorithmen von Unterwasserfahrzeugen zu ermöglichen.

Tabelle 3.6 betrachtet die einzelnen Bewertungskriterien für das CViewLab.

3.3.3.4 FERS

Bei FERS (The flexible extensible radar simulator) handelt es sich um einen physikalischen Signal Level Simulator, der mit sehr genauen Modellen die erwarteten Ergebnisse eines Radargerätes simulieren kann [Bro08].
Im Gegensatz zu den Simulatoren aus dem Bereich der Robotik werden bei FERS Fehlermodelle nicht auf den eigentlich zu untersuchenden Gegenstand angewendet, also den Radarstrahlen. Stattdessen werden Fehlermodelle auf einzelne Teile der Prozesskette angewendet, wie beispielsweise die Uhren oder Signalgeneratoren im Sender und Empfänger.

Die Eigenschaften der FERS Radarsimulation, in Bezug auf die in Abschnitt 3.3.1 vorgestellten Bewertungskriterien, werden in Tabelle 3.7 zusammengefasst.

3.3.4 Kommerzielle Simulationssysteme

Neben den bisher vorgestellten Frameworks- und Simulationssystemen existieren verschiedene kommerzielle Simulationssysteme, mit denen Sensordaten generiert werden können. Dabei ist die Nutzung von simulierten Sensormesswerten in verschiedenen Domänen unterschiedlich stark ausgeprägt.

In der maritimen Domäne, die in dieser Arbeit verstärkt betrachtet wird, werden Simulation hauptsächlich im Rahmen von Ausbildung und Trainingsmaßnahmen Eingesetzt. Hierfür stehen verschiedene Schiffsführungssimulatoren [AG17, Kon09] zur Verfügung, die neben dem dynamischen Verhalten des simulierten Schiffes ebenfalls in der Lage sind, Sensormesswerte zu generieren. Dabei handelt es sich i.d.R. um Sensoren wie Radar, Echolot, GPS, (AIS) und weitere domänenspezifische Sensoren. Die generierten Sensormesswerte werden während der Simulation den Probanden über die, auf der Brücke befindlichen Systeme dargestellt. Eine Nutzung zur Entwicklung und Überprüfung von Assistenzsystemen ist dagegen nicht vorgesehen.

Die Automotive Domäne setzt dagegen sehr stark auf den Einsatz von Simulationstechniken in verschiedenen Phasen des Entwicklungsprozesses. In der Regel kommen dabei verschiedene Werkzeuge bzw. Werkzeugketten zum Einsatz, die für ihren Einsatzzweck optimiert sind.
Für die Simulation des dynamischen Verhaltens von Fahrzeugen werden u.a. die Werkzeuge IPG-CarMaker [Gmb17] oder Virtual Test Drive der Firma VIRES [noa17] verwendet.
Bei der Simulation von Sensoren lassen sich aktuell zwei Ansätze erkennen. Zum einen werden physikalische Simulationen, mit einem sehr hohen Detailgrad verwendet, um realistische Sensormesswerte zu generieren. Hierbei kommen zum Beispiel Raytracingverfahren zum Einsatz mit denen sich die in Abschnitt 2.3.3 vorgestellten Mehrwegefehler in LIDAR oder Ultraschall Sensoren, gut nachbilden bzw. annähernd fotorealistische Kamerabilder erstellen lassen [MB16, RCH+12, GGDS12]. Der Nachteil dieser Methode ist der sehr hohe Modellierungsaufwand der betrieben werden muss, um realistische Messwerte zu erzeugen.
Der zweite Ansatz sieht eine nachträgliche Anwendung von Fehlern auf idealisierte Sensormesswerte vor. Dieser Ansatz wird zum Beispiel in der Arbeit Schubert et. Al. [SMB14] vorgestellt. Dabei handelt es sich um ein Joint-Venture der Firmen TASS International und ihrer Sensorsimulation PreScan[7], sowie der Deutschen Firma BASELABS[8]. Dabei profitiert dieses Verfahren von den Erfahrungen der Firma BASELABS, die in ihrem Hauptgeschäftsfeld Datenfusionsalgorithmen entwickelt. Bei dem Verfahren werden

[7] https://www.tassinternational.com/prescan
[8] https://www.baselabs.de/

die Fehlercharakteristiken, die ursprünglich zur Bereinigung realer Messwerte verwendet wurden, invertiert und auf die von PreScan gelieferten Sensorbeobachtungen angewendet.
Der Vorteil dieses Verfahrens liegt in einer einfacheren Modellierung der simulierten Szenarien, da probabilistische Fehler verwendet werden können. Auf der anderen Seite müssen zunächst die entsprechenden Fehlercharakteristiken der einzelnen Sensoren ermittelt werden, was in der Realität kein trivialer Prozess ist.

3.4 Zusammenfassung und Handlungsbedarf

Es existieren bereits eine Vielzahl von Werkzeugen, die den Entwickler von sensordatenverarbeitenden Systemen bei der Entwicklung unterstützen sollen. Insbesondere der Bereich der Robotik setzt dabei bereits seit längerem auf eine simulative Unterstützung bei der Entwicklung neuer Systeme. Viele der vorgestellten Werkzeuge unterstützen den Entwicklungsprozess der Sensordatenverarbeitung, indem sie kontextbezogene Sensordaten generieren, die als Eingabe für die zu entwickelnde Sensordatenverarbeitung verwendet werden können. Dabei wird häufig nur eine sehr einfache Modellierung der Umgebung durchgeführt. Der Stage Simulator für die Player Middleware verwendet zum Beispiel eine Schwarz-Weiß Bitmap zur Modellierung der statischen Hindernisse. Bei dieser Modellierungsmethode gehen die Informationen, um welche Objekte es sich handelt und welche zusätzlichen Eigenschaften sie besitzen, verloren.
Andere Simulatoren beschränken sich auf eine 3D Modellierung der Umgebung, indem sie für jedes Objekt ein Oberflächenmodell und eine Position und Ausrichtung im Raum bestimmen. Zusätzliche Informationen, wie Namen, Identifikationsnummern, Typ usw. werden entweder nicht modelliert oder werden in zusätzlichen Datenbanken abgelegt.
Eine ggf. vorhandene Bewertungsmetrik muss sich in diesem Fall die benötigten Informationen aus einer Vielzahl von unterschiedlichen Datenquellen zusammensuchen.
Inzwischen unterstützen die meisten Robotik-Simulationssysteme das Verrauschen der generierten Sensordaten. Bei den verwendeten Fehlermodellen handelt es sich aber i.d.R. um einfache Modelle, bei denen eine Zufallszahl auf einzelne Messwerte addiert wird. Nur die wenigsten unterstützen verschiedene Wahrscheinlichkeitsfunktionen. Stattdessen beschränken sie sich auf die am häufigsten verwendete Normalverteilung. Zeitabhängige Fehlermodelle, wie sie beispielsweise bei der Analyse der Rauschcharakteristik des SICK LMS 200 (vgl. Abschnitt 2.3.3) herausgekommen sind, werden nicht berücksichtigt. Kontextbezogene Fehlermodelle werden aktuell ohne ein explizites Modell nur für einzelne Sensoren implementiert oder sind als implizites Modell in die Implementierung der Sensorsimulation mit eingeflossen (vgl. z.B. VANE Simulation).

Auffällig bei den untersuchten Robotik Frameworks und Sensorsimulationen ist, dass sie mit einer Ausnahme, die direkte Bewertung der Sensordatenverarbeitung vernachlässigen. Stattdessen wird eine indirekte Bewertungsmethode gewählt, bei der die Sensordatenverarbeitung in das Gesamtsystem integriert untersucht wird.
Dabei wirken sich Fehler in der Sensordatenverarbeitung, wie beispielsweise das Übersehen eines Hindernisses durch das Fehlverhalten des robotischen Systems aus, indem dieser zum Beispiel mit dem übersehenen Hindernis kollidiert oder ein gesetztes Ziel nicht erreicht. Der Nachteil dieses Vorgehens wurde bereits in der Motivation im Zusammenhang mit der Fehlerfortpflanzung beschrieben. Dabei kann sich der Entwickler nicht sicher sein, an welcher Stelle der erste Fehler aufgetreten ist.

Eine direkte Überprüfung der Sensordatenverarbeitung, wie sie mit dem im Folgenden beschriebenen Ansatz möglich ist, erlaubt eine schnellere Identifikation der Fehlerursache indem 1) ein Fehler in der Sensordatenverarbeitung schneller gefunden werden kann und somit eine zusätzliche Fehlersuche in der nachfolgenden Logik ausbleiben kann bzw. 2) ein Fehler in der Sensordatenverarbeitung ausgeschlossen werden kann und somit die Suche auf den Fehler innerhalb der Entscheidungslogik oder den Aktuatoren beschränkt werden kann.

3.4.1 Anforderungen

In den vorangegangenen Kapiteln wurde zunächst die in dieser Arbeit behandelte Fragestellung sowie eine erste Idee des Lösungsansatzes vorgestellt. Anschließend wurde im zweiten Kapitel das zu testende System, die Sensordatenverarbeitung näher betrachtet sowie im dritten Kapitel existierende Lösungsansätze auf ihre Eignung zur Beantwortung der Fragestellung untersucht.
Aufbauend auf diesen Informationen, sollen im Folgenden die Anforderungen an den eigenen Ansatz hergeleitet und diskutiert werden.

Die Anforderungen lassen sich grob nach den zwei definierten Zielen: Der Bewertung von sensordatenverarbeitenden Systemen gemäß der in Abschnitt 1.3 vorgestellten Qualitätskriterien, sowie der Unterstützung des Entwicklungsprozesses neuer sensordatenverarbeitender Systeme unterteilen. Dabei ist zu beachten, dass diese Trennung nicht immer eindeutig vorgenommen und einige Anforderungen beiden Kategorien zugeordnet werden können.

3.4.1.1 Anforderungen für die Bewertung von sensordatenverarbeitenden Systemen

Der erste Block von Anforderungen bezieht sich hauptsächlich auf das Ziel, eine Bewertung von sensordatenverarbeitenden Systemen zu ermöglichen.

A1 - Modellierbarkeit *Unter der Modellierbarkeit des Systems wird die Fähigkeit zur strukturierten Beschreibung der verwendeten Sensoren sowie ihres Einsatzraumes verstanden.*

Eine adäquate Beschreibung des Einsatzraumes für komplexe, simulierte Szenarien ist schwierig. Insbesondere im Bereich der physikalischen Sensorsimulation muss dabei viel Wissen über die Funktionsweise der Sensoren und ihre Interaktion mit der Umgebung bekannt sein. Dies führt dazu, dass die untersuchten speziellen Simulationssysteme über eine sehr komplexe und damit aufwändige Beschreibung der Umgebung verfügen (vgl. z.B. VANE oder FEARS).
Bei anderen Ansätzen, insbesondere aus dem Robotikbereich, wird dagegen auf eine einfache Beschreibung der Umgebung zurückgegriffen, bei der relevante Informationen wie beispielsweise die Semantik eines Objektes oder seine Materialeigenschaften nicht berücksichtigt werden können (vgl. z.B. Player/Stage oder MRPT).

Um verschiedene Komplexitätsstufen bei der Modellierung des Einsatzraumes unterstützen zu können, muss eine strukturierte Beschreibung eben jenes Einsatzraumes ermöglicht werden.
Auf Grundlage dieser Beschreibung soll anschließend das Verhalten der verwendeten Sensoren beschrieben werden können.
Um die verschiedenen Phasen des Entwicklungsprozesses unterstützen zu können, soll zudem für beide Teilbereiche die Beschreibung der Sensorik sowie die Beschreibung ihres Einsatzraumes, eine sukzessive Steigerung der Komplexität ermöglichen.

A2 - Konfigurierbarkeit *Unter Konfigurierbarkeit bzw. Kontrollierbarkeit wird die Fähigkeit des Systems verstanden, das Verhalten der Sensoren sowie von Objekten in der Umgebung durch die Änderung von Parametern und Eigenschaften zu kontrollieren.*

Unter einer Eigenschaft in diesem Sinne wird zum Beispiel die Position eines Objektes in der Umgebung verstanden. Dabei handelt es sich um eine Information bzw. einen Datensatz, mit dem der Zustand des betrachteten Objektes genauer beschrieben werden kann. Weiterhin ermöglichen es die Werte von Eigenschaften eine Unterscheidung zwischen zwei Objekten vom selben Typen vorzunehmen. Unter dem Begriff des Parameters wiederum wird eine Stellgröße verstanden, die sich

auf das Verhalten eines Objektes auswirkt und damit ggf. bestimmt, wie sich die Eigenschaften des Objektes über die Zeit verändern. Die Reichweite eines maritimen Radarsensors wäre ein Beispiel für einen Parameter in diesem Sinne.
Die Forderung nach konfigurierbaren Parametern trägt dabei zum einem dem Umstand Rechnung, dass dies auch bei realen Sensoren möglich ist. Die Reichweite des maritimen Radars lässt sich beispielsweise während des laufenden Betriebes ändern.
Zum anderen kann diese Fähigkeit für automatische Analysen wie beispielsweise der von Gollücke et. al. [GPL+12] vorgestellten systematische Parameterexploration verwenden. Die Kombination von konfigurierbaren Parametern und der systematischen Parameterexploration kann verwendet werden, um beispielsweise eine Sensitivitätsanalyse durchzuführen oder um automatisch die Parametergrenzen zu ermitteln, mit denen eine Erfüllung der Qualitätskriterien gewährleistet werden kann.

Die Konfigurierbarkeit von Eigenschaften hat dagegen einen höheren Wert bei der Unterstützung des Entwicklungsprozesses von sensordatenverarbeitenden Systemen. So ist sie eine notwendige Voraussetzung für das sogenannte Browsing. Dabei handelt es sich um eine strukturierte oder ggf. auch (scheinbar) unstrukturierte Form des Experimentierens mit verschiedenen Sensorkonfigurationen um beispielsweise das beobachtete System besser kennenzulernen oder Lösungsansätze abzuschätzen (vgl. [Hjø11]).

A3 - Beobachtbarkeit *Unter der Beobachtbarkeit des Simulationssystems wird die Fähigkeit verstanden, den Zustand eines beliebigen Objektes innerhalb der simulierten Umgebung zu jedem Zeitpunkt der Simulation genau bestimmen zu können.*

Bei dieser Anforderung handelt es sich um einen zentralen Aspekt der vorgestellten Methode zur Überprüfung von sensordatenverarbeitenden Systemen, da aus den beobachteten Simulationszuständen der Erwartungswert für die Überprüfung der Ergebnisse der Sensordatenverarbeitung gebildet werden kann.
Dabei ist wichtig, dass eine maschinelle Beobachtbarkeit gewährleistet werden kann, d.h. das ein externes System oder eine interne Komponente lesenden Zugriff auf die Eigenschaften der Objekte erhalten kann. Ein Beispiel für eine Eigenschaft kann die Position eines Objektes in der Umgebung sein.

Auf der anderen Seite soll eine manuelle Beobachtbarkeit gewährleistet werden, d.h. es muss einem menschlichen Benutzer die Möglichkeit gegeben werden, den aktuellen Zustand der Umgebung erfassen zu können. Diese Form der Beobachtbarkeit ist während der Entwicklung und ggf. der manuellen Durchführung von Tests von besonderer Bedeutung, da durch ihre Erfüllung die Plausibilität der vom testenden System generierten Ergebnisse abgeschätzt werden.

A4 - Zeitverhalten und Effizienz *Die Anforderung bezieht sich auf die Fähigkeit des Simulationssystems, die Sensormesswerte in den von der Sensordatenverarbeitung benötigten Zeitintervallen liefern zu können.*

Damit ist diese Forderung von zentraler Bedeutung für die Überprüfung von Antwortzeiten sensordatenverarbeitender Systeme, wie sie in Abschnitt 1.3 als eines der Ziele gefordert wird.
Dabei kann auch berücksichtigt werden, dass die Sensordatenverarbeitung Sensorwerte ggf. in einer geringeren Frequenz verarbeitet, als diese von einem realen Sensor geliefert werden. In diesem Fall wäre eine Erfüllung der Anforderung auch dann gegeben, wenn die Frequenz der Sensorsimulation mit der von der Sensordatenverarbeitung geforderten Frequenz übereinstimmt.

Als eine optionale Anforderung kann in diesem Zusammenhang die Forderung angesehen werden, dass das Simulationssystem verschiedene zeitliche Auflösungen unterstützen soll, d.h. eine beschleunigte oder verlangsamte Ausführung der Simulation.
Durch die Erfüllung dieser optionalen Anforderung lässt sich der vorgestellte Ansatz mit artverwandten Arbeiten, wie beispielsweise der in [Gol16] vorgestellten Rare Event Simulation kombinieren, bei der von einer Vielzahl von Simulationsläufen ausgegangen wird.

A5 - Fehlerfreie Sensormesswerte *Die Sensorsimulatoren müssen in der Lage sein, fehlerfreie Sensormesswerte zu generieren.*

Mit fehlerfrei ist in diesem Zusammenhang die Abwesenheit von Messungenauigkeiten und Sensorfehlern (vgl. Abschnitt 2.3) gemeint. Das heißt, wenn die Position eines Objektes bestimmt werden soll, zum Beispiel mit einem GPS Sensor und das Objekt ändert seine Position nicht, ändern sich auch die Sensormesswerte nicht; sie sind stabil.
Die Forderung nach einer fehlerfreien Sensormesswertgenerierung, ermöglicht den Einsatz von Sensoren zur Überprüfung des sensordatenverarbeitenden Systems, indem der fehlerfreie Messwert als Erwartungswert verwendet wird.

A6 - Konfigurierbare Fehlermodelle *Auf jeden Sensormesswert können Fehlermodelle angewendet werden, die die Messungenauigkeiten und Sensorfehler des Sensors berücksichtigen können.*

Reale Sensoren liefern keine fehlerfreien Messwerte. Um dies in der Simulation nachbilden zu können, sollen Messungenauigkeiten auf die generierten Sensormesswerte angewendet werden. Eine Liste der zu berücksichtigenden Fehlermodelle wurde in Abschnitt 2.3 diskutiert.

Mit dem Zusatz der konfigurierbaren Fehlermodelle werden zwei Absichten verfolgt:

Nachbildung der realen Fehlermodelle Mit Hilfe der beschriebenen Fehlermodelle soll eine Nachbildung der Messungenauigkeit und ggf. des Fehlverhaltens von realen Sensoren möglich sein.

Experimentelle Fehlermodelle Durch die Anwendung von nicht realitätsgetreuen Fehlermodellen lassen sich ggf. zukünftige Sensoren, die beispielsweise über eine geringere Messungenauigkeit verfügen, nachbilden. Diese Eigenschaft ist z.B. bei Forschungsprojekten sinnvoll, in denen neben einem sensordatenverarbeitenden System, auch die dazugehörige Sensorik entwickelt wird. Somit lässt sich die Sensordatenverarbeitung bereits vor der Fertigstellung der Sensoren entwickeln und überprüfen.
Auf der anderen Seite kann durch eine Erhöhung der Messungenauigkeit die Stabilität der verwendeten Algorithmen überprüft werden, zum Beispiel in Hinblick auf die Verwendung von kostengünstigeren Sensoren.

3.4.1.2 Anforderungen für die Unterstützung des Entwicklungsprozesses von sensordatenverarbeitenden Systemen

Der zweite Block von Anforderungen bezieht sich im Wesentlichen auf die Unterstützung bei der Entwicklung von sensordatenverarbeitenden Systemen, indem bereits in frühen Phasen der Entwicklung Messwerte zur Verfügung gestellt werden können.

A7 - Nachbildung existierender Sensorschnittstellen *Die Sensorsimulation soll in der Lage sein, die Schnittstellen real existierender Sensoren nachzubilden und die generierten Sensormesswerte sollen über diese Schnittstellen kommuniziert werden.*

Durch das Nachbilden der realen Schnittstellen lassen sich Integrationsfehler beim Umstieg von simulierten Sensordaten zu realen Sensordaten vermeiden, da keine Änderungen an den verwendeten Schnittstellen vorgenommen werden müssen.
Ein weiterer Effekt bei der Verwendung von realen Sensorschnittstellen, ist die Möglichkeit sensordatenverarbeitende Systeme zu überprüfen, die nicht in der Lage sind, eine Top-Level Middleware, wie beispielsweise Player [GVH03] oder ROS [QCG+09], zu verwenden. Dies kann unter anderem der Fall sein, wenn auf Grundlage von Überlegungen bezüglich der Sicherheit des Systems, auf eine zusätzliche Abstraktionsschicht verzichtet werden soll. Auch bei der Anwendung von HIL Verfahren ist die Verwendung einer komplexen Middleware nicht üblich.

A8 - Konsistenz *Zwei oder mehr Sensoren, die das gleiche Phänomen der Umgebung beobachten, müssen im Normalfall eine konsistente Interpretation des beobachteten Phänomens erlauben.*

Mit dem Begriff Konsistenz ist in diesem Zusammenhang die logische Interpretation des Begriffes gemeint, die besagt, dass eine Menge von Aussagen konsistent ist, wenn sich aus ihr keine Widersprüche ableiten lassen.
Im Zusammenhang mit einer Sensorsimulation bezieht sich die Forderung auf die Konsistenz der generierten Sensorbeobachtungen. Das heißt insbesondere, dass zwei Sensoren, die dasselbe Phänomen beobachten, die gleichen Rückschlüsse auf dieses Phänomen zulassen müssen.

Die Konsistenz der Sensormesswerte spielt eine besondere Rolle im Zusammenhang mit der Fusion verschiedener homogener und heterogener Sensoren, wie sie beispielsweise in [KBF+13] durchgeführt wird. Nur wenn die Sensormesswerte in sich konsistent sind, lässt sich ein entsprechend konsistentes Umgebungsmodell durch die Sensordatenverarbeitung erkennen.

Eine weitere Anwendung für die Verwendung konsistenter Daten liefert die Arbeit [KN12]. Dort werden verschiedene, verhältnismäßig einfache Sensoren verwendet, um eine Schätzung bezüglich der Qualität eines komplexeren Sensors abzugeben. In diese Schätzung geht das Wissen über das Verhalten des komplexen Sensors ein. Eine normale Überwachungskamera benötigt beispielsweise eine gewisse Lichtstärke, bei der sie qualitativ hochwertige Aufnahmen liefern kann. Durch die Verwendung eines einfachen Helligkeitssensors kann die Sensordatenverarbeitung ermitteln, ob die geforderte Lichtstärke in der Umgebung der Überwachungskamera gegeben ist oder nicht. Mit Hilfe dieses Wissens lassen sich, der Argumentation von [KN12] folgend, Aussagen über die Vertrauenswürdigkeit der Sensormesswerte machen. Gleichzeitig kann diese Technik verwendet werden, um Störungen in der Sensorik zu ermitteln.

A9 - Nachvollziehbarkeit *Das Verhalten der durch die Sensoren beobachteten Objekte muss für die Sensordatenverarbeitung nachvollziehbar sein.*

Verschiedene Verfahren im Rahmen der Sensordatenverarbeitung arbeiten mit Modellen des Verhaltens von Objekten, wie beispielsweise die Klassifikation oder das Tracking (vgl. Abschnitt 2.1). Um eine Überprüfung dieser Modelle auch bei den simulierten Sensormesswerten zu ermöglichen, muss das Verhalten der simulierten Objekte den realen Vorbildern nachempfunden werden.

Bei den beiden Anforderungen A8 und A9 ist zu beachten, dass diese sich auf die fehlerfrei simulierten Sensormesswerte (vgl. A5) beziehen. Mithilfe der konfigurierbaren

Fehlermodelle muss es aber möglich sein, genau diese beiden Anforderungen zu brechen um damit eine realistische Simulation der Sensormesswerte zu ermöglichen aber dennoch ein konsistentes Umgebungsmodell für die Bewertung der Sensordatenverarbeitung bereitstellen zu können.

Tabelle 3.1: Bewertungskriterien für das Player / Stage / Gazebo Framework

Kriterium:	**Analyse**
Sensor Klassen	PSG unterstützt verschiedene Klassen von Sensoren, hauptsächlich aus dem klassischen Umfeld der robotischen Umfelderkennung. Zu den hauptsächlich verwendeten Sensoren gehören Laserscanner, Odometer, Lichtschranken, Ultraschall und Inertialsensorik und Kameras.
Simulationslevel	Sowohl Stage als auch Gazebo erzeugen das erwartete Ergebnis in dem von der Sensordatenverarbeitung bzw. der Middleware erwarteten Format und fallen somit in die Klasse der ergebnisorientierten Simulationen.
Fehlermodelle	Die Unterstützung von Sensorfehlern liegt nicht im Hauptfokus des PSG Projektes, dennoch lassen sich einige der von Stage und Gazebo simulierten Sensoren durch statistische Fehler verrauschen. Hierbei werden verhältnismäßig einfache Modelle, wie normalverteilte Zufallszahlen verwendet und auf einzelne Eigenschaften der Beobachtung addiert. Dabei überlässt es die Stage Simulation der Verantwortung des Sensorentwicklers, ob ein entsprechender Fehler angewendet wird oder nicht. In der Folge unterstützen einige Sensoren eine stochastische Fehlermodellierung während andere dies nicht tun. Kontextsensitive Fehler werden nicht in dem Maße unterstützt, wie sie in dieser Arbeit verstanden werden. Allerdings berücksichtigen einige Sensoren den aktuellen Kontext in dem sich der Sensor gerade befindet, bei der Erzeugung der Sensorwerte.
Bewertung	Auf eine quantitative Analyse der Sensordatenverarbeitung wird verzichtet. Stattdessen setzt PSG auf die manuelle bzw. optische Überprüfung des Ergebnisses und bietet durch seinen Frameworkcharakter die Möglichkeit eigene Ansätze zu realisieren.
Umgebung	In der Umgebungsmodellierung unterscheiden sich Stage und Gazebo naturgemäß. Während Stage eine zweidimensionale Repräsentation verwendet, können in Gazebo komplexe 3D Modelle hinterlegt und für die Sensordatensynthese verwendet werden. Dabei werden die Objekte zu einem großen Teil auf ihre äußere Form (2D / 3D Geometrien) reduziert.
Kommunikation	PSG liefert seine eigene Middleware (Player) auf dessen Grundlage zwischen der Simulation und der Datenverarbeitung kommuniziert wird.
Intention	PSG wurde entwickelt um eine sichere Experimentierumgebung für die Entwickler von neuen robotischen System zu erschaffen, in der neue Fusions- und Steuerungsalgorithmen untersucht werden können.

Tabelle 3.2: Bewertungskriterien für das Mobile Robot Programming Toolkit (MRPT)

Kriterium:	**Analyse**
Sensor Klassen	Für den Bereich der Sensordatenverarbeitung bzw. die Anbindung real existierender Hardware unterstützt das MRPT verschiedene 2D und 3D Laserscanner, 3D Kameras wie die Kinect oder Stereokameras, GPS Sensoren und Inertialsensorik.
Simulationslevel	Ergebnisorientierte Sensorsimulation
Fehlermodelle	Für die simulierten Laser und Odometrie bzw. Positionsdaten lassen sich einfache Fehlermodelle anwenden. Der statistische Fehler wird über die Angabe der Standardabweichung in Metern angegeben. Für die Positionsbestimmung lassen sich neben dem normalverteiltem Rauschen ein Bias für die X und Y Komponente der Position angeben, um einen systematischen Fehler zu simulieren. Kontextsensitive Fehler werden nicht betrachtet.
Bewertung	Die Bewertung einer Sensordatenverarbeitung erfolgt über eine visuelle Bewertung der in der Benutzungsoberfläche dargestellten Ergebnisse. Eine quantitative Bewertung ist nicht vorgesehen.
Umgebung	Die Umgebungsmodellierung beschränkt sich im Wesentlichen auf die Angabe von Hindernissen in einer zweidimensionalen Umwelt, welche wiederum in Form einer Rasterkarte bzw. einem Graustufenbild angegeben wird. Neben der einfachen Rasterumgebung können verschiedene Landmarken modelliert werden.
Intention	Das MRPT versteht sich als eine Sammlung von Anwendungen und Bibliotheken die die Entwicklung von robotischen Applikationen vereinfachen soll.

Tabelle 3.3: Bewertungskriterien für die USARSim Simulation

Kriterium:	**Analyse**
Sensor Klassen	Kamera, Laserscanner, Ultraschall, Sonar, Kontakt, Position, RFID, Mikrofon
Simulationslevel	Ergebnisorientierte Simulation
Fehlermodelle	Einfache Fehlermodelle lassen sich während der Erzeugung der Sensorwerte auf die Messwerte anwenden. Dabei wird berücksichtigt, dass nicht jeder Wert einer Messung einem Fehler unterliegt.
Bewertung	Eine Bewertung der Sensordatenverarbeitung ist nicht direkt vorgesehen.
Umgebung	Unter anderem begründet in der kommerziellen Herkunft des Simulators, konzentriert sich die Umgebungsmodellierung auf die optischen Aspekte in Form von 3D Geometrien, Materialien und Texturen. Zusätzlich können die Verletzten als eine spezialisierte Klasse von Objekten in der Umgebung platziert und ggf. mit zusätzlichen Aktionen ausgestattet werden.
Kommunikation	USARSim verwendet intern eine eigene Script Sprache, mit der sowohl die Roboter als auch die Sensoren angesprochen werden können. Zusätzlich werden verschiedene Middlewarelösungen unterstützt, darunter auch die bereits vorgestellte Middelware Player (vgl. Abschnitt 3.3.2.1).
Intention	USARSim versteht sich selber als ein Robotiksimulator, der sowohl in der Forschung als auch in der Lehre eingesetzt werden kann. Zusätzlich wurde er als Testumgebung für die RoboCub Rescue League verwendet.

Tabelle 3.4: Bewertungskriterien für die VANE Simulation

Kriterium:	**Analyse**
Sensor Klassen	Vorgestellt werden GPS, Laserscanner und CCD Kameras
Simulationslevel	Nach der Beschreibung aus Coodin et.al [GDG+10] muss VANE in die Klasse der physikalischen Simulationen eingeordnet werden. Gerade die beschriebene Simulation der GPS Sensoren ähnelt eher einem Nachbau des eigentlichen Messprinzips, als eine Anwendung von Fehlermodellen auf die ursprünglich idealen Daten.
Fehlermodelle	Die VANE Simulationsumgebung versucht die Fehlermodelle auf ihre physikalischen Gegebenheiten zurückzuführen und damit soweit wie möglich zufällige Einflüsse zu unterbinden. Durch die Nachbildung der physikalischen Sensormodelle bzw. der physikalischen Arbeitsweise der Sensoren werden verschiedene kontextsensitive Fehler mit berücksichtigt, zum Beispiel die Mixed Pixel Fehler bei der Simulation von Laserscannern. Andere kontextsensitive Fehler wie beispielsweise die Mehrwegereflexion von GPS Signalen werden dagegen explizit durch eine Sichtbarkeitsprüfung bei der Auswahl der GPS-Satelliten durchgeführt.
Bewertung	Die Bewertung eines sensordatenverarbeitenden Systems erfolgt in VANE über eine optische Prüfung / Auswertung von Simulationsläufen, gekoppelt mit dem Verhalten der Sensorplattformen (z.B. Fahrzeug). Zu diesem Zweck kann auf die realzeitfähige Visualisierung ANVEL zurückgegriffen werden.
Umgebung	Die Umgebung der Simulatoren wird zu großen Teilen als 3D Polygone vorgehalten; jeweils mit verschiedenen Materialien für die unterschiedlichen Sensoren.
Intention	VANE wurde entwickelt, um die Entwicklung neuer robotischer Systeme zu vereinfachen, indem bereits unter Laborbedingungen genaue Sensormesswerte generiert werden können.

Tabelle 3.5: Bewertungskriterien für den Tunnel Simulator

Kriterium:	**Analyse**
Sensor Klassen	Der Tunnelsimulator konzentriert sich bei der Sensorsimulation auf genau einen Sensortyp, nämlich optische Kamerasysteme. Diese werden jedoch mit einer sehr hohen Genauigkeit simuliert.
Simulationslevel	Ergebnisorientierte Simulation, mit Ansätzen aus der Signal Level Simulation
Fehlermodelle	Es werden keine expliziten Fehlermodelle angegeben. Durch die verwendete Raytraceing Technik zur Erzeugung der Videobilder lässt sich der Detailgrad der Kamerabilder in der Theorie bis auf das fotorealistischen Level steigern. Dies würde eine explizite Modellierung von Fehlern Überflüssig machen.
Bewertung	Die Bewertung der Sensordatenverarbeitung besteht im Falle des Tunnelsimulators aus einer Überprüfung, ob die Sensordatenverarbeitung (SDV) ein zuvor definiertes signifikantes Ereignis, wie beispielsweise eine Kollision, erkennt. Zu diesem Zweck werden die zuvor festgelegten Ereignisse in einer OVF (Object Video File) gesammelt, um einen Abgleich mit den Ausgaben der Sensordatenverarbeitung vornehmen zu können.
Umgebung	Beim Tunnelsimulator erfolgt die Modellierung der Umgebung mit dem 3D Modellierungswerkzeug Blender. Sie besteht aus den, in der Kamera sichtbaren, 3D Objekten inklusive Texturen und Materialien. Gleichzeitig lassen sich die in Blender zur Verfügung stehenden Lichtquellen verwenden, um realistische Lichteffekte nachbilden zu können. Neben der Modellierung der visuellen Aspekte ermöglicht Blender auch die Modellierung des Verhaltens von Objekten, in Form von Animationen.
Kommunikation	Die Kommunikation zwischen der Datengenerierung und der Sensordatenverarbeitung erfolgt über offline gerenderte Videosequenzen.
Intention	Ziel des Tunnelsimulators ist es, die Güte eines kamerabasierten Assistenzsystems zur Verkehrsüberwachung in einem Tunnel bestimmen zu können.

Tabelle 3.6: Bewertungskriterien für die CViewLab Simulationsumgebung

Kriterium:	**Analyse**
Sensor Klassen	Das CViewLab hat sich auf die Simulation von Sonarsensoren von Unterwasserfahrzeugen spezialisiert.
Simulationslevel	Die CViewLab Simulation kann der ergebnisorientierten Sensorsimulation zugeordnet werden.
Fehlermodelle	Über mögliche Fehlermodelle der simulierten Sensoren wird in der Veröffentlichung keine Aussage gemacht.
Bewertung	Es wird keine explizite Bewertung erwähnt, stattdessen wird auf die visuelle Überprüfung zurückgegriffen. Dies liegt auch in der Intention begründet, bei der die Kontrollalgorithmen für Unterwasserfahrzeuge überprüft werden sollen und weniger die Funktionsweise der Sensordatenverarbeitung
Umgebung	CViewLab bzw. CViewVR hat einen starken Anteil in der Visualisierung, dementsprechend wird die Umgebung durch die zu visualisierenden 3D Modelle abgebildet
Intention	Die CView Umgebung stellt eine virtuelle Testumgebung für neue Regel- und Steueralgorithmen für autonome Wasserfahrzeuge bereit. Der Sensorsimulationsanteil in diesem Projekt dient hauptsächlich der Bereitstellung der für die Steuerung benötigten Daten.

Tabelle 3.7: Bewertungskriterien für den FERS Simulator

Kriterium:	**Analyse**
Sensor Klassen	Radar
Simulationslevel	Signal Level Simulation
Fehlermodelle	FERS fällt unter die Klasse der physikalischen Simulationen. Somit tritt eine Fehlermodellierung, wie sie in dieser Arbeit verwendet wird, in den Hintergrund. Stattdessen werden verschiedene physikalische Effekte, wie beispielsweise die Abschwächung des elektromagnetischen Signals oder die Mehrfachreflexion an Oberflächen explizit modelliert. Ganz ohne eine stochastische Modellierung kommt auch der physiklische Simulator FERS nicht aus. Im Gegensatz zu dem hier verwendeten Ansatz werden die stochastischen Fehlermodelle jedoch auf die Eingangsdaten der Simulation angewendet, anstatt auf die Ausgangsdaten, wie es unter anderem von CViewLab, MRPT oder Stage durchgeführt wird. Gleichwohl kann dies als eine nachträgliche Anwendung von Fehlermodellen auf die Oszillator / Frequenz Generierung angesehen werden. Kontextsensitive Fehler wie beispielsweise die Mehrfachreflexion eines Radarsignals werden in FERS explizit modelliert und in der Berechnung berücksichtigt. Dadurch muss keine nachträgliche Manipulation der Messergebnisse in bestimmten Situationen vorgenommen werden.
Bewertung	FERS enthält keine explizite Bewertung von sensordatenverarbeitenden Systemen, dies ist jedoch auch nicht das Ziel des Simulators.
Umgebung	Für die Umgebungsmodellierung verwendet FERS ein hierarchisches Modell, bestehend aus sechs Elementen: Das Radar, welches wiederum in Sender und Empfänger unterteilt wird. Ziele bei denen es sich um die durch das Radar beobachtbaren Elemente handelt. Eine Oberfläche zur Simulation von Mehrwegereflexionen sowie einer Plattform, mit der die zuvor genannten Umgebungselemente gruppiert werden können. Damit ist das Umgebungsmodell ähnlich einem Szenegraphen aus der 3D Welt aufgebaut.

Kapitel 4

Simulative Bewertung von sensordatenverarbeitenden Systemen (SenseSim)

In diesem Kapitel wird das Konzept der simulativen Bewertung von sensordatenverarbeitenden Systemen beschrieben. Es beginnt mit einer Übersicht und Einordnung der relevanten Konzepte. Dabei handelt es sich um eine Fortsetzung der in Abschnitt 1.3 formulierten Idee, die reale Umwelt durch eine simulierte Umwelt, inklusive simulierter Sensoren, zu ersetzen (vgl. Abbildung 4.1).

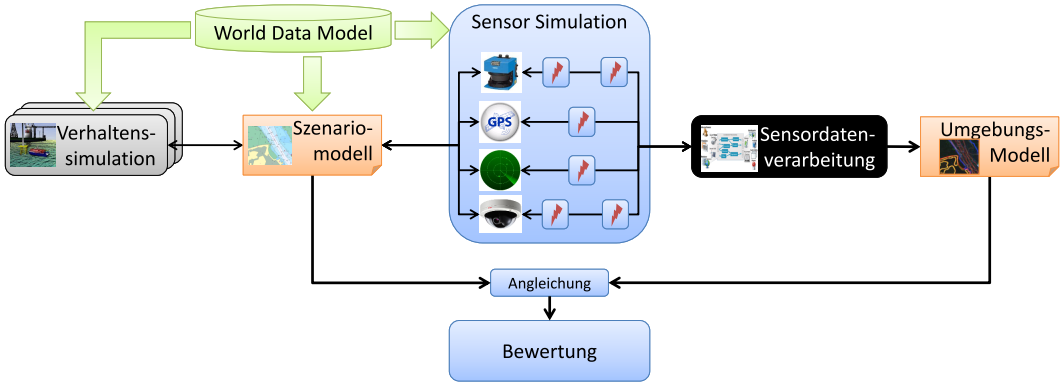

Abbildung 4.1: Konzeptionelle Idee

Weiterhin wurde im Abschnitt 1.3 beschrieben, dass die zur Messwertsynthese verwendeten Informationen gleichzeitig als Erwartungswert für die Bewertung des sensordatenverarbeitenden Systems verwendet werden können. Gerade die doppelte Verwendung des Szenariomodells zur Messwertsynthese und Bewertung der Sensordatenverarbeitung, unterscheidet diesen Ansatz von aktuellen Lösungsansätzen wie sie im dritten Kapitel (vgl. Abschnitt 3.3) untersucht wurden.

Bei der Betrachtung der existierenden Ansätzen zur Überprüfung von sensordatenverarbeitenden Systemen in Kapitel 3 konnten zudem drei Probleme identifiziert werden, auf dessen Behandlung durch den vorgestellten Ansatz, im Folgenden näher eingegangen wird.

1. Es findet keine direkte Bewertung der Sensordatenverarbeitung statt. Stattdessen wird i.d.R. eine indirekte Bewertung vorgenommen, indem geprüft wird, ob die entwickelte Entscheidungslogik korrekt arbeitet.

2. Wenn Fehlermodelle bei der Sensormesswertsynthese beachtet werden, dann werden diese direkt bei der Erzeugung der Messwerte angewendet. Zusätzlich handelt es sich gerade bei den public - Domain - Frameworks [GVS+01, CMG05, AKC+15, CLW+07, Cla08] um verhältnismäßig einfache Fehlermodelle, die von einer Sensordatenverarbeitung leicht erkannt und ausgeglichen werden können.

3. Es muss ein hoher Modellierungsaufwand für das Bereitstellen der Sensordaten betrieben werden. Dies beinhaltet das Bereitstellen einer beobachtbaren Umgebung, das Verhalten der beobachteten Objekte, das Verhalten der Sensoren sowie das Fehlverhalten der Sensoren.

Bewertung der Sensordatenverarbeitung Im dritten Kapitel dieser Arbeit wurden zwei unterschiedliche Ansätze zur Überprüfung von sensordatenverarbeitenden Systemen vorgestellt. Zum einen handelt es sich dabei um die Überprüfung durch reale Messwerte (vgl. Abschnitt 3.1) sowie die Überprüfung mithilfe von Sensorsimulationen (vgl. Abschnitt 3.2).
Bei der Überprüfung historischer Messwerte muss zunächst eine manuelle oder teilautomatisierte [Sta14, VHVS+11] Annotation der Erwartungswerte durchgeführt werden, bevor eine Überprüfung vorgenommen werden kann. Diese ist mit einem relativ hohen Aufwand verbunden. Auf der anderen Seite weißt J. Stallkamp [Sta14] darauf hin, dass bei der Verwendung von synthetischen Messwerten, diese für die Überprüfung der Sensordatenverarbeitung verwendet werden können, gibt aber kein strukturiertes Vorgehen hierfür an.
Bei den untersuchten Public - Domain - Frameworken [GVS+01, CMG05, AKC+15, CLW+07, Cla08] wird dagegen eine Überprüfung des Gesamtsystems, d.h. der Sensordatenverarbeitung incl. der Entscheidungslogik durchgeführt. In der Folge findet keine direkte sondern eine indirekte Bewertung der Sensordatenverarbeitung statt, indem festgestellt werden kann, das z.B. der Steuerungsalgorithmus eine fehlerhafte Entscheidung getroffen hat und dadurch sein Ziel nicht erreicht. Wie in der Motivation beschrieben, lässt sich an dieser Stelle nur durch manuellen Aufwand feststellen, ob es sich dabei um

ein durch die Sensordatenverarbeitung oder die Entscheidungslogik induziertes Problem handelt.

Um das Problem der fehlenden (direkten) Bewertung von sensordatenverarbeitenden Systemen, auf Grundlage von simulierten Sensormesswerten anzugehen, wird in dieser Arbeit ein Ansatz beschrieben, der eine strukturierte und toolunterstützte Methode vorstellt, mit der das zur Sensordatensynthese verwendete Szenariomodell und damit der Erwartungswert für die Bewertung nutzbar gemacht werden kann. Dabei bedient sich der Ansatz Techniken, die ursprünglich aus dem Bereich der Sensor- bzw. Signalverarbeitung stammen und sich durch eine hohe Flexibilität und Wiederverwendbarkeit auszeichnen.

Durch die Wiederverwendung des Simulationsmodells für die Bewertung einer Sensordatenverarbeitung, kann der Aufwand für das manuelle oder teilautomatische Erstellen des Erwartungswertes bzw. die manuelle Suche nach den Ursachen eines Fehlers aufgewendet werden musste eingespart werden.

Fehlermodelle bei der Sensormesswertsynthese Neben den Bewertungseigenschaften wurden die in Abschnitt 3.3 vorgestellten Frameworks und Sensorsimulationen auf ihre Fähigkeit untersucht, Fehlermodelle für Sensoren bereitzustellen, da sich diese im starken Maße auf die Fähigkeiten der Sensordatenverarbeitung und damit auf die Bewertung derselbigen auswirken [Sie01].
Dabei konnte festgestellt werden, dass gerade im Umfeld der Public-Domain-Frameworks nur eine sehr rudimentäre Umsetzung von Fehlermodellen stattfindet aber auch kommerzielle Systeme wie beispielsweise PreScan generieren zunächst idealisierte Messwerte (vgl. Abschnitt 3.3.4), die dann optional von einer externen Software verrauscht werden müssen [SMB14].
Zudem konnte festgestellt werden, dass die realisierten Fehlermodelle eng mit der Sensormesswertsynthese gekoppelt sind, d.h. direkt bei der Erzeugung der Messwerte angewendet werden [GDG+10]. In der Folge lassen sich die entwickelten Fehlermodelle nicht für andere ggf. ähnliche Sensoren wiederverwenden und eine Anpassung der Fehlermodelle bedingt die Anpassung des gesamten Sensormodells.

In dem, in dieser Arbeit vorgestellten Ansatz, werden die Sensordatenerzeugung und die Fehlerinjektion getrennt voneinander betrachtet (vgl. Anforderungen A5 & A6) und mithilfe eines Datenflussgraphen lose miteinander gekoppelt.
Dabei wird das, aus dem Bereich der Sensor- bzw. Signalverarbeitung stammende Konzept des Datenflussgraphen, mit einer strukturierten Beschreibung der Umgebung, der Sensoren und der Sensorbeobachtungen kombiniert, um die Anwendbarkeit von Fehlermodellen auf alle Sensorbeobachtungen ermöglichen zu können. Damit wird sein ursprünglicher Verwendungszweck, das Erkennen und Verarbeiten von Messunsicherheiten

ins Gegenteil gewandelt, indem Messunsicherheiten zu Sensorbeobachtungen hinzugefügt, anstatt entfernt werden.
Die durch den Datenflussgraphen erreichte lose Kopplung zwischen Messwertsynthese und Fehleranwendung ermöglicht zudem die Wiederverwendung und flexible Kombination von existierenden Fehlermodellen zu Fehlerketten. Innerhalb einer solchen Fehlerkette, werden die einzelnen Fehlermodelle einzeln konfiguriert und nacheinander auf die generierten Messwerte angewendet, um das komplexe Fehlverhalten von Sensoren abzubilden.

Durch den Einsatz der Fehlerketten wird es dem Anwender der Methode ermöglicht, neben der Bewertung von sensordatenverarbeitenden Systemen, auch die Evaluierung von Designentscheidungen zu untersuchen. Dabei kann es sich beispielsweise um eine Entscheidung für oder gegen die Verwendung von höherwertiger Sensorik handeln oder ob der Aufwand für die Entwicklung einer verbesserten Rauschunterdrückung gerechtfertigt ist. Im Rahmen der Entwicklungsunterstützung von neuen sensordatenverarbeitenden Systemen erlaubt die lose Kopplung zusätzlich den vollständigen Verzicht auf Sensormessfehler und Messungenauigkeiten, wenn z.B. in frühen Phasen der Entwicklung zunächst die generelle Funktionsweise eines neu entwickelten Algorithmus getestet werden soll (vgl. [Sta14]).

Modellierungsaufwand zum Bereitstellen von Sensormesswerten Die in Abschnitt 3.3.3 vorgestellten simulativen Ansätze zur Bewertung von sensordatenverarbeitenden Systemen benötigen i.d.R. einen hohen Modellierungsaufwand für das Bereitstellen einer beobachtbaren Umgebung sowie der Sensor und Fehlermodelle. Dies gilt insbesondere für Sensorsimulationen die unter die Gruppe der Signal - Level Sensorsimulationen fallen. In etwas eingeschränkter Form aber auch für die ergebnisorientierten Simulationsansätze wie sie in dieser Arbeit verwendet wird.

Neben der flexiblen Kombinierbarkeit vorhandener Fehlermodelle erlaubt der im Abschnitt 4.3.2.2 vorgestellte Datenflussgraph, die dynamische Konfiguration der verwendeten Fehlermodelle unter Berücksichtigung des aktuellen Sensorkontextes. Somit lassen sich Faktoren die nur einen geringen Einfluss auf die generierten Messwerte haben, durch kontextsensitive Fehlermodelle abbilden und müssen nicht mehr in den Sensormodellen berücksichtigt werden.
Ein Beispiel hierfür ist der in Abschnitt 6.1.3.2 modellierte entfernungsabhängige Fehler eines Differential GPS Sensors in Abhängigkeit zu der nächsten Referenzstation. Ist ein entsprechendes statistisches Fehlermodell verfügbar, kann während der Messwertsynthese auf die Berechnung der DGPS Korrektursignale verzichtet werden und stattdessen ein kontextsensitiver Fehler modelliert werden.

Eine zusätzliche Reduktion des Modellierungsaufwandes für die Sensorsimulation wird durch die Integration der Sensorsimulation in eine Ko-Simulationsumgebung erreicht, wie sie in ähnlicher Form auch bei kommerziellen Systemen (z.B. CarMaker und PreScan - vgl. Abschnitt 3.3.4) zum Einsatz kommt. Dabei übernehmen externe Simulatoren, die sogenannten Verhaltenssimulatoren, die Simulation von beobachtbaren Objekten, die in der Folge von der Sensorsimulation nicht weiter behandelt werden müssen. Hierzu kommen Interoperabilitätstechniken zum Einsatz, wie sie u.a. in Dibbern et. al. [DHS14] vorgestellt wurden.

Während die Abbildung 4.1 eine abstrakte Sicht auf das Konzept und die von ihm verwendeten Artefakte darstellt, wird im Folgenden Abschnitt eine Übersicht über die konkret auszuführenden Schritte zur Durchführung einer simulativen Überprüfung von sensordatenverarbeitenden Systemen gegeben.
Diese stellt gleichzeitig die Gliederung für das restliche Kapitel dar, bevor im nächsten Kapitel 5 eine modellbasierte prototypische Implementierung vorgestellt wird.

4.1 Genereller Ablauf zur Bewertung von sensordatenverarbeitenden Systemen

Der generelle Ablauf zur Bewertung eines sensordatenverarbeitenden Systems, hinsichtlich der in Abschnitt 1.2.1 vorgestellten Qualitätskriterien ist in Abbildung 4.2 dargestellt und gliedert sich in die drei Teilbereiche

1. Spezifikation der beteiligten Systeme
2. Beschreibung des Sensorverhaltens
3. Simulationsdurchführung.

Spezifikation der beteiligten Systeme

Beim Schritt der Spezifikation der beteiligten Systeme handelt es sich um einen Schritt, der von einem Experten im Vorfeld der hier vorgeschlagenen Methodik durchgeführt wird und die Basis für die anschließende Beschreibung des Sensorverhaltens darstellt.
In ihm werden zunächst die beteiligten Systeme, wie das zu testende sensordatenverarbeitende System (SUT), die verwendeten Sensoren inklusive ihrer Fehlermodelle, sowie die verwendeten Schnittstellen - zur als auch von dem SUT, identifiziert. Insbesondere wird an dieser Stelle eine Spezifikation bzw. Quantifizierung der in Abschnitt 1.2.1

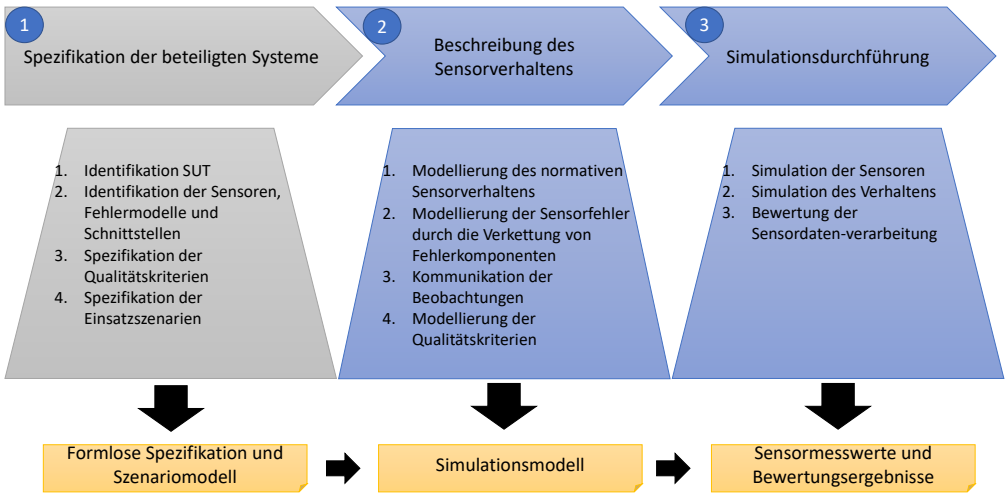

Abbildung 4.2: Generelles Vorgehen zur simulativen Bewertung von sensordatenverarbeitenden Systemen

vorgestellten Qualitätskriterien für die Sensordatenverarbeitung von dem Experten vorgenommen. Die Spezifikation dieser Systeme kann dabei zunächst formlos geschehen und wird im nächsten Prozessschritt formalisiert.

Neben der Identifikation der beteiligten Systeme beinhaltet der erste Schritt der Methode die Spezifikation des Einsatzszenarios für die spätere Simulationsdurchführung. Das entsprechende Artefakt ist das sogenannte *Szenariomodell*, welches alle relevanten Objekte der Umgebung beinhaltet. Dabei kann auf ein gemeinsam genutztes Datenmodell [DHS14, SGHB14], das sogenannte *World Data Model* (vgl. Abbildung 4.1), zurückgegriffen werden, in dem die Struktur und Semantik von Objekten beschrieben ist, aus denen die Umgebung zusammengestellt werden kann.
Das World Data Model welches in Zusammenarbeit mit den Arbeiten von Volker Gollücke [Gol16] und Rainer Droste [Dro16] entwickelt wurde, wird dabei durch eine formale Beschreibungssprache spezifiziert (vgl. Abschnitt 4.2.1.1), die sich an der Essential Meta Object Facility (EMOF) Architektur orientiert. Gemäß der EMOF Spezifikation [OMG14] müssen Modelle die mit dieser Methode erstellt werden, den Zugriff auf die Struktur ihrer Objekte während der Laufzeit ermöglichen.
Diese Fähigkeit wiederum wird u.a. im Abschnitt 4.3.2 dazu genutzt werden, um Sensormesswerte aus den Eigenschaften der beobachteten Objekte zu extrahieren sowie in Abschnitt 4.4 eine einfache Integration in eine Ko-Simulationsumgebung zu ermöglichen.

Beschreibung des Sensorverhaltens

Der zweite Schritt der Methode, die Beschreibung des Sensorverhaltens für die simulative Überprüfung von sensordatenverarbeitenden Systemen verwendet die Ergebnisse des vorherigen Schrittes, um ein ausführbares *Simulationsmodell* zu erstellen. Mithilfe dieses Modelles können im dritten Schritt die Sensoren in einer virtuellen Umgebung simuliert, verrauscht und die Ergebnisse der zu testenden Sensordatenverarbeitung bewertet werden.
Damit stellt die Beschreibung des Simulationsmodells den Hauptaspekt des eigenen Ansatzes dar und besteht im Wesentlichen aus der Komposition von Komponenten zur Datenerzeugung, Messwertverfälschung und Ergebnisbewertung, in einem datenorientierten Verarbeitungsgraphen.
Ähnliche Ansätze werden u.a. in dem Bereich der Sensor- bzw. Signalverarbeitung angewendet [Inc17, GG12, RSJ+05, AT] müssen jedoch für die Sensorsimulation in dynamischen Umgebungen angepasst werden (vgl. Abschnitt 4.3.2.2).

Messwerterzeugung

Das Simulationsmodell besteht dabei zunächst aus einer Beschreibung des normativen Verhaltens von Sensoren und beschreibt wie die Sensormessungen aus dem zuvor erstelltem Szenariomodell extrahiert werden können.
Dazu wird zunächst ein *konzeptionelles Modell* zur Simulation von Sensoren, aus der in Abschnitt 2.2.3 vorgestellten, Sensor Semantic Network Ontologie [CBB+12, LHT+11] hergeleitet und für den Einsatz in einer dynamischen Simulationsumgebung angepasst.
Darauf aufbauend wird ein *Komponentenmodell* sowie eine Erweiterung des World Data Models vorgestellt, welche die grundlegende Struktur des Simulationsmodells definieren. Diese kann am besten mit einem datengetriebenen Verarbeitungsgraphen verglichen werden, wie er auch in anderen Softwaresystemen, wie beispielsweise Matlab Simulink [Inc17] oder Odysseus [GG12] verwendet wird. Der Verarbeitungsgraph besteht dabei aus Komponenten, die für die Erzeugung von Sensorbeobachtungen, die Verrauschung der Sensordaten und die Bewertung genutzt werden können.

Bei der Erzeugung der Sensormesswerte kann auf das Szenariomodell aus dem ersten Schritt der Methode zurückgegriffen werden, um den aktuellen Zustand der Umgebung abzufragen und als entsprechende Sensorbeobachtung aufzubereiten. Bei den so erzeugten Sensorbeobachtungen handelt es sich um die idealisierten Sensoren, wie sie in Anforderung A5 gefordert wurden.
Die idealen Sensorbeobachtungen können zusätzlich von Fehlerkomponenten manipuliert

werden, um eine realistischere Beobachtung für die Sensordatenverarbeitung zu erzeugen. Genau wie bei der Erzeugung der Sensormesswerte können die Fehlerkomponenten auf den aktuellen Zustand der Umgebung zugreifen, um beispielsweise den Kontext des Sensors für die Beschreibung eines kontextsensitiven Fehlers (vgl. Abschnitt 2.3.3) zu bestimmen. Damit trägt dieser Teil des eigenen Ansatzes maßgeblich zur Erfüllung der Anforderung nach konfigurierbaren Fehlermodellen (vgl. Anforderung A6) bei.

Fehlermodellierung

Eine Besonderheit des vorgestellten Ansatzes ist die Möglichkeit, verschiedene Fehlermodelle miteinander verketten zu können. Durch eine solche Verkettung von Fehlermodellen lässt sich leicht ein komplexeres Verhalten der Sensoren bezüglich ihres Fehlerverhaltens herstellen. Unter anderem kann damit dem Umstand Rechnung getragen werden, dass Sensorfehler, wie sie in der Realität auftreten, nicht durch einfache Zufallsverteilungen (z.B. die Normalverteilung) abgebildet, sondern lediglich angenähert werden können.

Eine Verkettung von statistischen Fehlermodellen kann zum Beispiel verwendet werden, wenn sich der gesamte Sensorfehler auf verschiedene (physikalische) Einflüsse zurückführen lässt, wie es beispielsweise bei dem globalen Positionsbestimmungssystem GPS der Fall ist (vgl. [UBl09, Ran94] und Abschnitt 6.1.1.2). In diesem Beispiel kann der Fehler eines GPS Sensors in die drei Bereiche: satellitenspezifische, atmosphärenspezifische und empfängerspezifische Fehler aufgeteilt werden, welche jeweils einzeln beschrieben werden können. Abbildung 4.3 zeigt, wie diese Aufteilung in dem vorgeschlagenen Datenverarbeitungsgraphen modelliert werden kann.

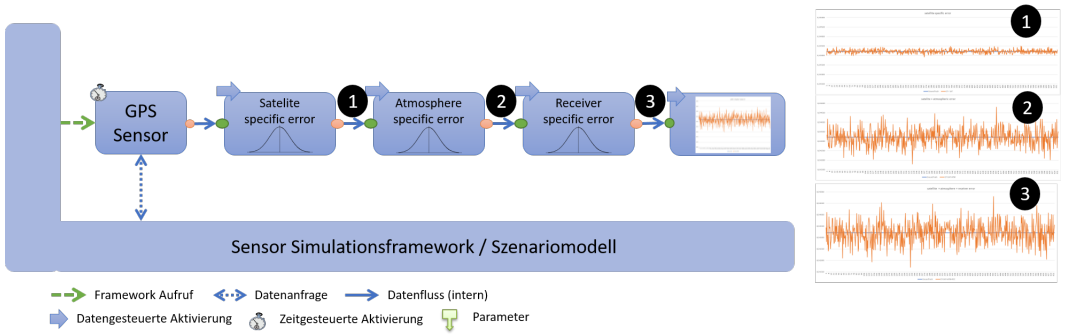

Abbildung 4.3: Verkettung von Fehlermodellen am Beispiel eines GPS Sensors, inklusive Zwischenergebnisse: (1) Satellitenspezifischer Fehler, (2) Satellitenspezifischer und Atmosphärenspezifischer Fehler und (3) Superposition aller drei Fehlermodelle.

In dem dargestellten Beispiel werden die einzelnen Fehlereinflüsse jeweils durch eine eigene Fehlerkomponente modelliert, deren Ergebnisse nach dem Superpositionsprinzip mit dem Ergebnis der vorhergegangenen Fehlerkomponente verrechnet werden. Dabei

kann jedes der drei Teilmodelle unterschiedlich konfiguriert werden, um z.B. die Stärke der jeweiligen Einflüsse auf den Gesamtfehler zu berücksichtigen.

Neben der reinen Aneinanderreihung von statisch konfigurierten Fehlermodellen, erlaubt der in Abschnitt 4.3.2.2 vorgestellte Datenverarbeitungsgraph eine dynamische Konfiguration von Fehlermodellen.
Dabei können die Fehlerkomponenten auf Grundlage des Szenariomodells, der tatsächlichen Sensorwerte oder durch Hilfskomponenten, konfiguriert werden. Diese Fähigkeit eignet sich besonders für die Modellierung von kontextabhängigen Fehlermodellen, wie sie in Abschnitt 2.3.3 beschrieben wurden.

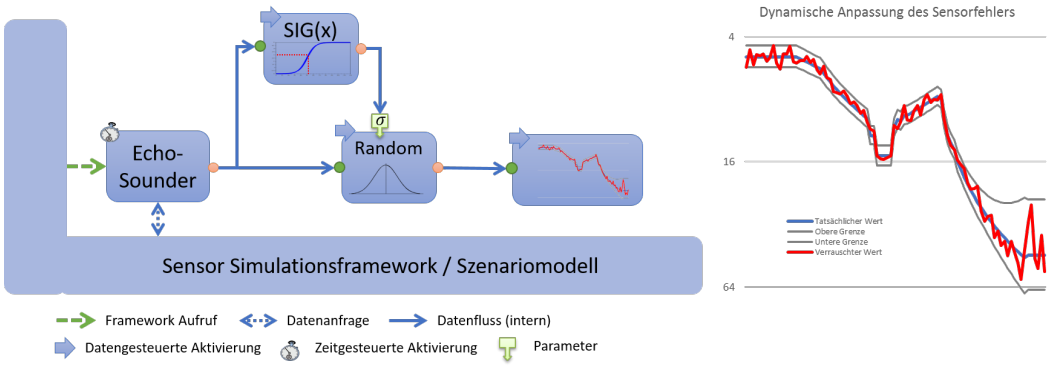

Abbildung 4.4: Dynamische Anpassung von Fehlermodellen mit Hilfe des Verarbeitungsgraphen.

Abbildung 4.4 zeigt ein Beispiel für eine solche dynamische Konfiguration, bei der Beispielhaft ein tiefenabhängiger Fehler für einen Sonarsensor modelliert wird.
Dabei bestimmt ein idealer Sonarsensor zunächst die tatsächliche Tiefe an der gefragten Position (dargestellt durch die blaue Linie im rechten Teil der Abbildung). Dieser Wert wird anschließend verwendet, um eine dynamische Anpassung der Standardabweichung zu berechnen. Bei dieser Anpassung handelt es sich in dem dargestellten Beispiel um eine, entsprechend dem zuvor identifizierten Fehlermodell skalierte, Sigmoidfunktion. Die Verknüpfung der Sigmoidfunktion mit der Standardabweichung des verwendeten Fehlermodells kann, in vereinfachter Form, als obere bzw. untere Grenze des Fehlers interpretiert werden, wie es in rechten Teil der Abbildung 4.4 durch die grauen Linien angedeutet ist. Durch ähnliche Kombinationen von (Fehler-)Komponenten lassen sich weitere Fehlermodelle, wie beispielsweise die in Abschnitt 2.3.3 vorgestellte Aufwärmphase eines Laserscanners oder eine entfernungsabhängige Genauigkeit der Positionsbestimmung von Differential GPS Sensoren modellieren.

Ergebnisbewertung

Nachdem die Sensordatenverarbeitung mithilfe der generierten und ggf. verrauschten Beobachtungen ein Abbild der Umgebung erstellt hat, können die Ergebnisse mit dem Zustand der simulierten Umwelt verglichen werden, indem Vergleichskomponenten verwendet werden. Dabei kann sowohl auf die aktuelle Umgebung aus dem Szenariomodell zurückgegriffen werden, als auch auf Zwischenergebnisse die während der Messwertsynthese angefallen sind. Dies können z.B. die unverrauschten Sensorbeobachtungen eines simulierten Sensors sein, wie in Abbildung 4.5 dargestellt.

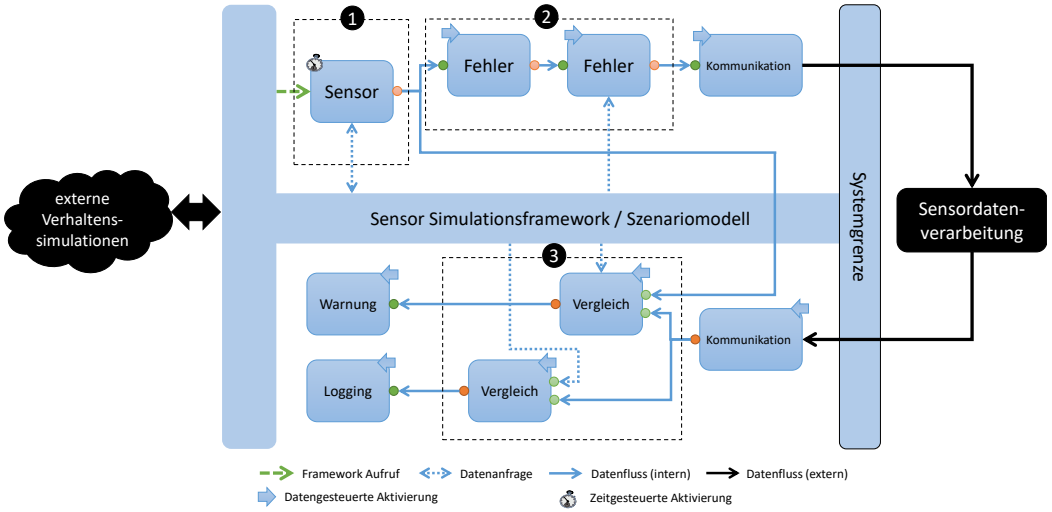

Abbildung 4.5: Schematische Darstellung eines Verarbeitungsgraphen zur simulativen Bewertung eines sensordatenverarbeitenden Systems, bestehend aus der Messwertsynthese (1) dem verrauschen der generierten Messwerte (2) und der Bewertung der Sensordatenverarbeitung (3).

Neben den zentralen Komponenten für die simulative Bewertung, können weitere Hilfskomponenten integriert werden, um z.B. die Kommunikation mit der zu überprüfenden Sensordatenverarbeitung durchzuführen oder um die Ergebnisse der Bewertung für den Anwender aufzubereiten.

Die Integration des komponentenbasierten Verarbeitungsgraphen in eine Ko-Simulation komplettiert den Ansatz. Dabei übernehmen externe Verhaltenssimulationen die Aufgabe, die Umgebung zwischen zwei Sensorbeobachtungen zu manipulieren indem sie z.B. die Position eines beobachteten Objektes im Szenariomodell verändern, und unterstützen damit die Anforderung nach einem nachvollziehbaren Verhalten der simulierten Objekte (vgl. Anforderung A9).

Simulationsdurchführung

Der letzte Schritt der Methode beschreibt die Ausführung des Simulationsmodells sowie eine optionale Einbettung in eine Ko-Simulationsumgebung. Am Ende dieses Schrittes stehen simulierte Messwerte für die Sensordatenverarbeitung und die Bewertung der Sensordatenverarbeitung hinsichtlich eines spezifischen Simulationsexperiments.
Dabei kann sowohl bei der Beschreibung des Sensorverhaltens als auch in der anschließenden Ausführung des Simulationsexperimentes auf die Bewertung der Sensordatenverarbeitung verzichtet werden, wenn die Erfüllung des zweiten in Abschnitt 1.2 formulierten Ziels, die Unterstützung bei der Entwicklung neuer sensordatenverarbeitenden Systeme erreicht werden soll.

Übersicht über die verwendeten Modelle

Bei der Übersicht über die vorgestellte Methode, wurden an verschiedenen Stellen verschiedene Modelle benannt, die in den einzelnen Schritten zum Einsatz kommen. Die Abbildung 4.6 stellt diese in einer Übersicht dar und beschreibt die Beziehungen unter ihnen.

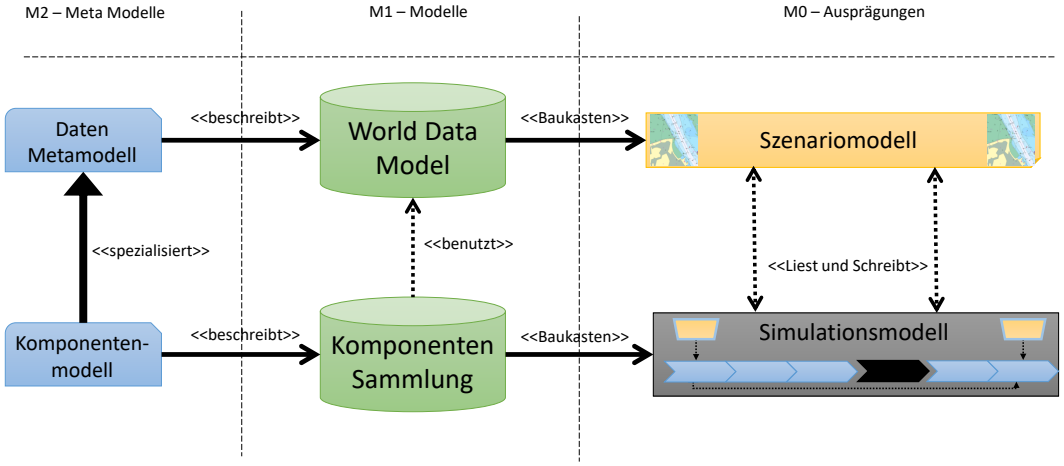

Abbildung 4.6: Übersicht über die verwendeten Modelle und ihre Beziehungen untereinander

Dabei ist die Verwendung der Modelle angelehnt an eine dreischichtige MOF Architektur [OMG16], bestehend aus Meta Modellen, Modellen und Modellausprägungen, wie es in der Abbildung dargestellt ist.
In diesem Zusammenhang werden die Meta Modelle dazu verwendet, die Struktur der Modelle in der darunterliegenden Ebene zu beschreiben.
Konkret auf den beschriebenen Ansatz bezogen, wurde bei der Übersicht über die Systemidentifikation von einer formalen Beschreibungssprache für das sogenannte World Data Model gesprochen. Diese formale Beschreibungssprache wird in Abschnitt 4.2.1.1

vorgestellt und im Folgenden als *Daten-Metamodell* bezeichnet. Sie liefert die Beschreibungsvorschrift für das *World Data Model*, bei dem es sich um eine Sammlung von Konzepten und Datenstrukturen handelt, aus denen das *Szenariomodell* zusammengesetzt werden kann.
Genauer gesagt handelt es sich bei dem *World Data Model* um ein domänenabhängiges Modell in dem die Struktur von Objekten beschrieben wird. Ein Beispiel ist die Beschreibung eines Schiffes, welches durch einen weltweit eindeutigen Identifier, die MMSI[1], eine Position, eine Ausrichtung und ggf. eine optionale Oberfläche beschrieben wird. Eine konkrete Ausprägung, z.B. das Forschungsschiff Polarstern, mit der MMSI = 211202460 welches an der Kaimauer des alten Hafens in Bremerhaven vertäut ist, wäre dementsprechend ein Beispiel für ein Element aus dem Szenariomodell.
Ähnliche Konstellationen von Modellen lassen im Umfeld des neuen "Universal Hydrographic Data Model" der IHO (kurz IHO S-100 oder S-100) finden [IHO15]. Dort werden z.B. sogenannte "Feature Catalogues" [IHO15, Part 5] definiert, welche eine ähnliche Aufgabe wie das World Data Model übernehmen. Diese "Feature Catalogues" werden mithilfe einer auf UML basierenden Beschreibungssprache dem sogenannten "General Feature Model" [IHO15, Part 3] beschrieben. Konkrete Instanzen wiederum bilden die sogenannten "Product Specifications" oder "Data Product " [IHO15, Part 11] und korrespondieren im Grundsatz mit dem in dieser Arbeit verwendeten Szenariomodell.

Den zweiten Teil der in diesem Ansatz verwendeten Modelle bilden das Komponentenmodell, eine Sammlung von Komponenten und das Simulationsmodell. Sie sind ähnlich wie die zuvor beschriebenen Datenmodelle aufgebaut. Das Komponentenmodell beschreibt die Struktur, wie eine einzelne Komponente beschrieben werden und wie verschiedene Komponenten miteinander verbunden werden können. Aufbauend auf dieser Beschreibung können Komponenten, wie beispielsweise die in Abschnitt 4.3.3 definierten Fehlerkomponenten, entwickelt werden und in einer Komponentesammlung gespeichert werden. Die Kombination verschiedener Komponenten bildet dann das Simulationsmodell.
Dabei verwenden die im Rahmen der Komponentenentwicklung verwendeten Modelle (unterer Teil der Abbildung 4.6) jeweils Konzepte aus dem entsprechenden Datenmodellen. Konkret spezialisiert das Komponentenmodell das Daten-Metamodell. Im Rahmen der Komponentensammlung, verwenden die einzelnen Komponenten Objekte, die im World Data Models beschrieben sind, als Schnittstellen um Daten untereinander auszutauschen, bzw. konkrete Komponenten können Elemente des Szenariomodells auslesen oder manipulieren.

[1] MMSI = Maritime Mobile Service Identity

4.2 Identifikation der beteiligten Systeme

Bei der Identifikation der beteiligten Systeme handelt es sich um einen Schritt, der vor der Anwendung der eigentlichen Methode, von einem Experten durchgeführt werden muss. Bei dem Experten handelt es sich um den Tester oder den Entwickler des sensordatenverarbeitenden Systems, der sowohl Kenntnisse über das zu testende System, als auch über dessen Qualitätsanforderungen hat.
Der Schritt beginnt mit der Festlegung des benötigten Detailgrades der Sensorsimulation und der Einordnung in die in Abschnitt 2.1.1 vorgestellten Anknüpfungspunkte zur Bewertung von sensordatenverarbeitenden Systemen in der Drei-Schichten-Architektur. In Bezug auf die Generierung von Sensormesswerten kann dabei aus zwei Kategorien ausgewählt werden: 1) Der Erzeugung von Sensorrohwerten bzw. Sensorsignalen und 2) der Erzeugung von Objekthypothesen.
Dabei wirkt sich die Auswahl maßgeblich auf die anschließend geforderte Komplexität des zu simulierenden Szenarios und der darin enthaltenen Informationen aus.
Wird beispielsweise die Simulation eines maritimen Radargerätes auf Sensorrohdatenlevel gewählt, müssen zusätzliche Informationen wie beispielsweise genaue Geometrien von Objekten und Geländemodelle berücksichtigt werden. Sollen dagegen Objekthypothesen generiert werden, reicht unter Umständen eine einfachere Modellierung der Objekte mit groben Geometrien.

Eine Hilfestellung bei der Auswahl des Detailgrades können die von der Sensordatenverarbeitung verwendeten Kommunikationsprotokolle liefern, sofern diese bereits definiert sind. Das begonnene Beispiel des maritimen Radars fortsetzend, würde bei der Verwendung des NMEA-Protokolls [Ao02] für die Übermittlung von Radarzielen, eine Simulation auf Objekthypothesenebene ausreichen, da ausschließlich Objekthypothesen übertragen werden können. Wird auf der anderen Seite, von der Sensordatenverarbeitung das ASTERIX Protokoll gefordert, insbesondere mit der Kategorie 240 [EUR09] (Radar Video Transmission), müsste eine Simulation auf Signalebene durchgeführt werden.

Nachdem der Detailgrad der Simulation festgelegt wurde, können die zu überprüfenden Qualitätskriterien hinsichtlich der:

- Korrektheit des erkannten Umgebungsmodells
- Fehlertoleranz gegenüber Sensorfehlern
- Robustheit gegenüber Messunsicherheiten
- Antwortzeiten der Sensordatenverarbeitung

quantifiziert werden.
Dieser Schritt kann übersprungen werden, wenn die Methode ausschließlich zur Unterstützung des Entwicklungsprozesses von sensordatenverarbeitenden Systemen verwendet werden soll.

Neben der Spezifikation der beteiligten Systeme wird in diesem Schritt der Methode das Einsatzszenario definiert, welches durch das sogenannte Szenariomodell repräsentiert wird.

4.2.1 Erstellung des Szenariomodells

Wie in Abschnitt 3.2 beschrieben wurde, benötigen ergebnisorientierte Sensorsimulationen, wie sie in der vorliegenden Arbeit verwendet werden, eine genaue Beschreibung der, durch die Sensoren beobachteten Umgebung. In dem hier vorgestellten Ansatz wird diese durch das sogenannte *Szenariomodell* repräsentiert und beinhaltet alle Objekte die von den Sensoren beobachtet werden können.
Gleichzeitig stellt es die Schnittstelle zu externen Verhaltenssimulationen dar, die in einer Art "distributed shared memory" [PTM98] über eine konkrete Ausprägung eines Szenariomodells synchronisiert werden (vgl. Abschnitt 4.4.2).

Zur Erstellung eines Szenariomodells kann auf die Elemente des sogenannten World Data Modells zurückgegriffen werden, bei dem es sich um eine formale Beschreibung der Struktur und Semantik von Objekten, in einer Domäne, handelt.
Die in dieser Arbeit verwendete Ausprägung des World Data Models wurde im Zusammenarbeit mit der Ko-Simulationsumgebung HAGGIS [HSGB15, SGHB14] sowie den Arbeiten von Volker Gollücke [Gol16] und Rainer Droste [Dro16] entwickelt und ist speziell für die Anwendung innerhalb einer Sensorsimulation ausgelegt worden.

Dabei ist die konkrete Ausprägung des World Data Models von nachgelagerter Bedeutung, sofern die im Folgenden genannten Anforderungen erfüllt werden.

WDM 1 Es muss eine strukturelle Beschreibung der Eigenschaften eines Objektes geben, auf die während der Modellierung von Sensoren und idealerweise auch zur Laufzeit der Simulation, zugegriffen werden kann.
Durch die Erfüllung dieser Anforderung lassen sich Sensormodelle definieren, bei denen Klassen der beobachteten Objekte nicht vollständig bekannt sein. Für die Ermittlung der Entfernung zu einem Hindernis, durch einen Radarsensor ist es zum Beispiel nicht von Belang, ob es sich bei dem Objekt um ein Schiff oder einen Leuchtturm handelt, sofern eine Beschreibung der Oberfläche und ggf. der Materialien zur Verfügung steht.

WDM 2 In Ergänzung zur Anforderung WDM 1 sollen Taxonomien definiert werden, mit denen die Semantik eines Objektes identifiziert werden kann.
Die Erfüllung dieser Anforderung ist weniger für die Simulation von Sensormessungen von Bedeutung, als vielmehr für eine anschließende Bewertung der Sensordatenverarbeitung hinsichtlich einer korrekten Klassifikation der beobachteten Objekte. Den Bedarf für eine solche Überprüfungsmöglichkeit demonstrierte im Jahr 2007 die DARPA Urban Challenge, bei der es an verschiedenen Stellen zu Unfällen aufgrund von fehlerhaften Klassifikationen kam. Da die gegnerischen Fahrzeuge sich, aufgrund von Sicherheitsüberlegungen, so langsam bewegten, dass sie von anderen Fahrzeugen als statische Objekte klassifiziert wurden [FTO+08].

WDM 3 Die Möglichkeit zur Selektion von einzelnen Teilmodellen, kann als eine optionale Anforderung angesehen werden.
Durch die Erfüllung dieser Anforderung lassen sich einfachere Umgebungsbeschreibungen erstellen, was wiederum den Arbeitsaufwand für die Erstellung einzelner Szenariomodelle vermindert.
Zum Beispiel muss eine Radarsensorsimulation keine Informationen über die Dynamikmodelle oder die Arbeitsprozesse an Bord der beobachteten Schiffe besitzen, wohingegen diese Informationen für andere, an einer Ko-Simulation beteiligten Simulatoren essentiell sind, welche dafür die Geometrie der Schiffe vernachlässigen können.

Dies Erfüllung der zuvor genannten Anforderungen kann durch die Verwendung einer geeigneten Modellierungssprache[2], sowie eine entsprechende Strukturierung des World Data Models[3], sichergestellt werden.
Auf die Modellierungssprache wird im Folgenden Abschnitt genauer eingegangen.

4.2.1.1 Metamodell zur Beschreibung des World Data Models

Für die Beschreibung eines Datenmodells, das die oben genannten Anforderungen erfüllt, eignen sich sowohl die Unified Modelling Language (UML) [OMG11] als auch die Essential Meta Object Facility (EMOF) Architektur [OMG14] welche unter anderem durch das Eclipse Modelling Framework (EMF) vorgeschlagen und implementiert worden ist [SBMP08].

[2] Gilt für die Anforderungen WDM 1 und WDM 2
[3] Gilt für die Anforderung WDM 3

Im Rahmen der prototypischen Umsetzung wurde eine Entscheidung zugunsten der EMOF Architektur getroffen, da diese im Gegensatz zur UML eine Interaktion zwischen verschiedenen Modellebenen (Metamodell und World Data Model) vorsieht und somit auch den optionalen Anteil der Anforderung WDM 1 unterstützt.

Konkret kommt eine leicht abgewandelte Form der EMOF Architektur zum Einsatz (vgl. Abbildung 4.7), welche hinsichtlich des Einsatzes in einer Ko-Simulationsumgebung optimiert wurde.

Diese Optimierung besteht aus der Einführung eines zusätzlichen *Classifiers*, der sogenannten *Structure*. Dabei handelt es sich um eine komplexe Datenstruktur, angelehnt an das Struct-Konzept aus C++, die über keine alleinstehende Semantik verfügt.
Die Einführung dieses Konzeptes ermöglicht, bei korrekter Anwendung, eine Optimierung hinsichtlich einer effizienten Synchronisierung durch eine Middleware in der Ko-Simulationsumgebung.
Structures werden dabei immer als zusammenhängende Datenpackete übermittelt, was wiederum bei verschiedenen Middleware Lösungen[4], das Marshalling bzw. Demarshalling vereinfacht.

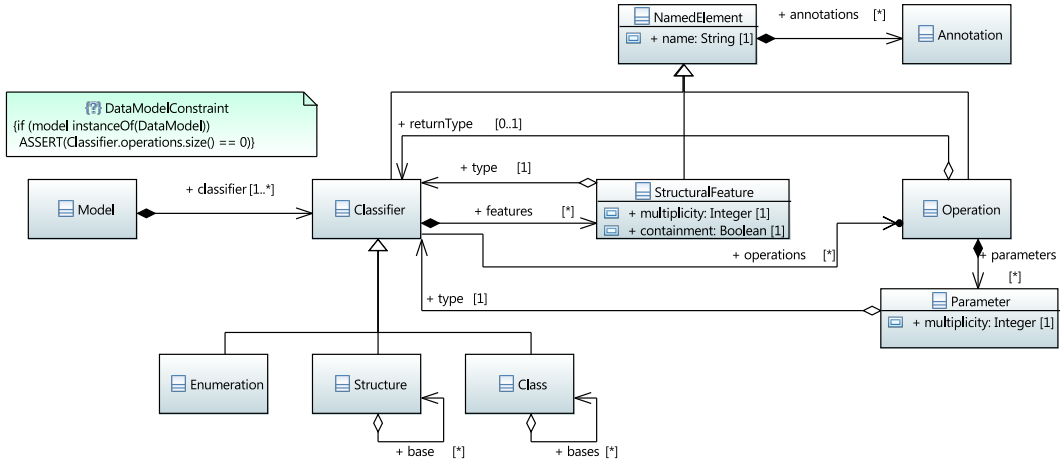

Abbildung 4.7: Vereinfachte UML Darstellung des Modells zur Beschreibung von Objekten in der Sensorsimulation, angelehnt an die EMOF Architektur

Darüber hinaus erfüllt die EMOF Architektur die oben genannten Anforderungen folgendermaßen: Eigenschaften von Objekten werden durch die sogenannten *StructuralFeatures* abgebildet und können wiederum einen beliebigen Datentyp repräsentieren.
Die Taxonomien, werden durch die *base* - Aggregationen der Klassen *Class* und *Structure* ermöglicht, mit deren Hilfe beliebige Objekthierarchien aufgebaut werden können.

[4]Diese Form der Optimierung lässt sich unter anderem bei der verwendeten High Level Architecture aber auch bei weiteren Middlewares wie beispielsweise CORBA [Go02] oder ZeroC ICE [Hen04, HS11] anwenden

4.3 Beschreibung des Sensorverhaltens

Dieser Teil der Methode zur simulativen Überprüfung von sensordatenverarbeitenden Systemen beschäftigt sich mit dem Erstellen des Simulationsmodells. Dabei gehen die im ersten Schritt der Methode gesammelten Informationen über das zu testende System und über die verwendeten Sensoren und ihre Fehlermodelle, in die Modellierung ein.
In Bezug auf die, in Abschnitt 3.4.1 genannten Anforderungen, werden im Folgenden die Anforderungen nach einer fehlerfreien bzw. idealisierten Erzeugung von Sensormessungen (vgl. Anforderung A5) und das anschließende Anwenden von Fehlermodellen auf die Sensorbeobachtungen betrachtet (vgl. Anforderung A6). Dazu wird ein Verfahren vorgestellt welches zudem die Anforderung nach konsistenten Sensorbeobachtungen durch verschiedene Sensoren (vgl. Anforderung A8), in hochdynamischen Umgebungen beachten kann.

Der Abschnitt ist wie folgt aufgebaut: Zunächst wird ein konzeptionelles Modell für die Simulation von Sensoren vorgestellt. Dieses Modell verwendet das "Stimulus-Sensor-Observation" (SSO) Pattern aus der in Abschnitt 2.2.3.1 vorgestellten Sensor Semantic Network (SSN) Ontologie und stellt eine Verbindung zwischen den Sensoren und den von ihnen beobachteten Eigenschaften her.
Mithilfe des entwickelten Modells, kann auf konzeptioneller Ebene beschrieben werden, wie die Sensormesswerte erzeugt werden können.
Für eine konkrete Modellierung und spätere Ausführung des konzeptionellen Modells, wird im Anschluss ein Komponentenmodell vorgestellt, das die bestehenden Ansätze aus der Literatur [Inc17, GG12, RSJ+05, AT] dahingehend erweitert, dass eine Simulation von Sensoren bzw. deren Messwerten ermöglicht wird. Dies geschieht durch die Kombination von Konzepten aus der Signalverarbeitungs- und der Simulationsdomäne, genauer gesagt dem Konzept des verteilten gemeinsamen Speichers zur Synchronisierung von Ko-Simulationen [PTM98, ISISC00] und der ereignisorientierten Simulation [LCRP+05].
Der letzte Teil des Abschnittes stellt verschiedene Komponenten vor, die für die Generierung und insbesondere die nachträgliche Manipulation der Sensorbeobachtungen verwendet werden können. Gleichzeitig werden Komponenten vorgestellt, mit denen eine Verletzung der in Abschnitt 1.2.1 vorgestellten Qualitätskriterien erkannt werden kann.

4.3.1 Konzeptionelles Modell von Sensoren

Im Abschnitt 2.2.3.1 wurde die Sensor Semantic Network Ontologie, als eine Möglichkeit zur Modellierung von Sensoren und ihren Beobachtungen vorgestellt.

In der anschließenden Diskussion wurde festgestellt, dass die SSN vornehmlich für die Beschreibung eines Messergebnisses verwendet werden kann und dabei die Beziehungen zwischen dem Sensor und der beobachteten Eigenschaft herstellt. Zusätzlich wurde festgestellt, dass die SSN durch das Konzept der Emitter in der Lage ist, aktive Sensoren, wie beispielsweise ein Radar oder einen Laserscanner, zu repräsentieren.
Als Nachteil der SSN, bzw. des in ihr enthaltenen Stimulus-Sensor-Observation Patterns, in Bezug auf die Generierung von Sensormesswerten, hat sich herausgestellt, dass sich das SSO Pattern vornehmlich auf die Zeit nach der Sensormessung bezieht, während für die Messwertsynthese die Beziehungen zwischen einem Sensor und einer beobachteten Eigenschaft vor der Messung beschrieben werden müssen.
Dies gilt insbesondere dann, wenn eine hochdynamische Umgebung und ggf. mobile Sensoren betrachtete werden, bei denen eine statische Zuordnung von Sensoren und durch sie beobachtete Objekte nicht möglich ist. Um die Beziehungen zwischen einem Sensor und den von ihm beobachteten Eigenschaften herzustellen wurde das in Abbildung 4.8 dargestellte konzeptionelle Modell, auf Basis des SSO-Patterns, entworfen.

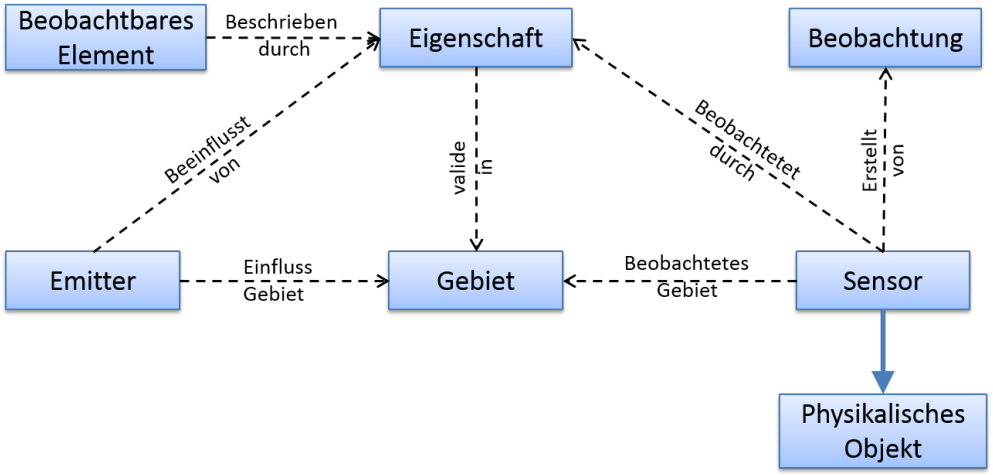

Abbildung 4.8: Konzeptionelles Modell für simulierte Sensoren, angelehnt an das SSO - Pattern der Sensor Semantic Network Ontologie.

Das in Abbildung 4.8 dargestellte konzeptionelle Modell für simulierte Sensoren übernimmt die Konzepte des Sensors, der eine oder mehrere Eigenschaften beobachtet und in eine Beobachtung überführt, sowie das Konzept des Emitters der Eigenschaften eines Objektes verändern kann.
Hinzu kommt, dass die von einem Sensor beobachteten Eigenschaften immer an ein Objekt gebunden sind, welches durch die konkreten Ausprägungen der Eigenschaften definiert wird.
Gekoppelt werden die drei Konzepte Sensor, Eigenschaft und Emitter durch das in der SSN nicht beachtete *Gebiet*.

Dabei handelt es sich aus der Sicht des Sensors, bei dem Gebiet um das sogenannte “Area of Interest“, also jenem Bereich der vom Sensor erfasst werden kann. Dies kann z.B. die Reichweite eines Laserscanners, das Sichtfeld einer Kamera oder die Position eines GPS Sensors sein.
Gleiches gilt für einen Emitter, wobei an dieser Stelle von einem “Area Of Influence“ gesprochen wird. D.h. dem Gebiet in dem der Emitter Eigenschaften verändern kann. Es handelt sich hierbei ebenfalls um die Reichweite des Emitters. In dem Fall, das ein aktiver Sensor modelliert werden soll, würden das “Area of Interest“ und das “Area of Influence“ übereinstimmen.
Aus Sicht einer Eigenschaft, befindet sie sich in einem Gebiet, üblicherweise an einer Position, welches in Relation zu dem dazugehörigen Objekt definiert ist.

Für die Verwendung des Ansatzes innerhalb einer hochdynamischen Umgebung, bei der keine statische Beziehung zwischen den Sensoren und den beobachteten Eigenschaften hergestellt werden kann, lässt sich mit Hilfe des “Area Of Interesst“ des Sensors, eine Filterung der zu beobachtenden Objekte und damit der zu beobachtenden Eigenschaften durchführen.
Abgesehen von der Tatsache, dass das Gebiet einer Eigenschaft in Relation zu ihren Objekten definiert wird, kann zudem festgestellt werden, dass für den Vorgang der Messwertsynthese das Objekt bzw. der genaue Typ des Objektes nicht von Bedeutung ist. Was auch das reale Verhalten von Umfeld erkennenden Sensoren wiederspiegelt, die i.A. keine Kenntnisse über die beobachteten Objekte haben.
Diese Tatsache erlaubt die Verwendung von einfacheren Umgebungsmodellen, bei dem nur noch gefordert wird, dass sich die, vom Sensor beobachteten Eigenschaften Bestandteil des Objektes sind oder durch die Kombination von vorhandenen Eigenschaften auf die gesuchte Eigenschaft geschlossen werden kann. An einem Beispiel verdeutlicht heißt das, dass beispielsweise für die Simulation eines LIDAR- oder Radarsensors keine Informationen über das verwendete Dynamikmodell der beobachteten Schiffe oder PKWs zur Verfügung stehen muss, da beide Sensoren ausschließlich die Oberflächeneigenschaften des Objektes berücksichtigen. Genau genommen muss auch nicht bekannt sein, ob es sich um ein Schiff oder einen PKW handelt.
Das dennoch eine Beziehung zwischen einer Beobachtung und einem beobachteten Element bzw. beobachteten Objekt hergestellt werden kann, ist allerdings für die anschließende Bewertung des ermittelten Umgebungsmodells, von Bedeutung z.B. wenn eine korrekte Klassifikation durch die Sensordatenverarbeitung überprüft werden soll.

4.3.2 Modellierung des normatives Sensorverhaltens

Gemäß der konzeptionellen Beschreibung von Sensoren lassen sich konkrete Beobachtungen von Sensoren durch das Beobachten der charakterisierenden Eigenschaften von Objekten innerhalb der Sensorreichweite ermitteln.
Für die Simulation von einfachen Sensoren, wie beispielsweise einem GPS-Sensor oder einem Temperatursensor, lassen sich dafür die vom Sensor beobachteten Eigenschaften i.d.R. direkt aus den Szenariomodell auslesen. Dementsprechend kann das normative Verhalten eines Sensors durch eine einfache Datenweiterleitung, vom Szenariomodell in die zu erzeugende Sensorbeobachtung, beschrieben werden. Ein Beispiel für eine solche Modellierung ist in Abbildung 4.9 dargestellt, bei der die Latitude und Longitude Eigenschaften eines Objektes in eine entsprechende GPS - Beobachtung überführt werden. Damit stellt Abbildung 4.9 die einfachste Form einer GPS - Sensorsimulation durch das Simulationsmodell dar.

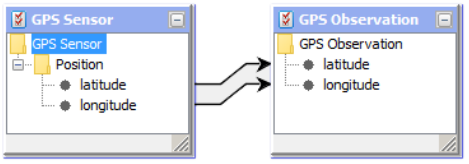

Abbildung 4.9: Einfachste Form einer GPS-Sensorsimulation durch das Simulationsmodell

Besteht die Möglichkeit des direkten Auslesens der Eigenschaften nicht, da entweder die gesuchte Eigenschaft nicht Bestandteil des Szenariomodells ist oder es sich um eine dynamische Eigenschaft, wie beispielsweise die Entfernung des Objektes vom Sensor, handelt, muss dynamisch auf die entsprechende Eigenschaft geschlossen werden.
Das Schließen auf eine gesuchte Eigenschaft aus dem Szenariomodell kann wiederum mithilfe einer zusätzlichen Komponente geschehen, die wie im Abschnitt 4.1 beschrieben, auf das Szenariomodell zugreifen kann.
Beide Varianten setzten voraus, dass eine strukturierte Beschreibung der Sensorbeobachtung zur Verfügung steht. Aus diesem Grund wird im nächsten Abschnitt eine strukturelle Beschreibung der Sensoren und ihrer Beobachtungen, auf Grundlage der in Abschnitt 4.2 beschriebenen formalen Beschreibungssprache vorgestellt. Bei dieser Beschreibung handelt es sich konkret um eine Erweiterung des, ebenfalls in Abschnitt 4.2 beschriebenen, World Data Models.
Dabei stellt die formale Beschreibung der Sensorbeobachtung nicht nur die Grundlage zur Beschreibung des normativen Verhaltens des Sensors dar, sondern ist auch eine notwendige Voraussetzung für die in Abschnitt 4.3.3 beschriebenen generellen Fehlermodelle, denen mit einer sogenannten *Fehlerkonfiguration* (vgl. Abschnitt 4.3.3.1) mitgeteilt werden kann, welche Teile der Sensorbeobachtung verrauscht werden dürfen.
Ein weiterer Grund für die Integration der Sensoren und ihrer Beobachtungen in das World Data Model ist die Möglichkeit Elemente des World Data Models und damit

auch die Sensoren und ihre Beobachtungen, mit anderen Simulationen synchronisieren zu können.

Im Anschluss an die strukturelle Beschreibung der Sensoren und Sensorbeobachtungen wird das bereits erwähnte Komponentenmodell vorgestellt sowie die genauen Regeln für die Kombination, Aktivierung und Weiterleitung von Daten zwischen Komponenten oder Elementen des Szenariomodells definiert.

4.3.2.1 Strukturelle Beschreibung von Sensoren

Die strukturelle Beschreibung der Sensoren und ihrer Beobachtungen wird im Rahmen des sogenannten *Sense* Modells als Ergänzung zu dem in Abschnitt 4.2.1 beschriebenen *World Data Model* vorgenommen und stellt die Umsetzung des konzeptionellen Modells aus Abschnitt 4.2.1.1 dar.
Neben der Beschreibung der Sensoren und Emitter enthält es Elemente zur Beschreibung von Sensorfehlern und Messungenauigkeiten, wie sie in Abschnitt 2.3 vorgestellt wurden.
Zusätzlich verfügt ein Sensor über eine Beschreibung seiner Kommunikationseigenschaften hinsichtlich der Erfüllung von Anforderung A7 -Nachbildung existierender Sensorschnittstellen. Abbildung 4.10 zeigt den Kern des *Sense* Modells, die Umsetzung des Stimulus- Sensor- Observation (SSO) Patterns der SSN bzw. des konzeptionellen Modells.

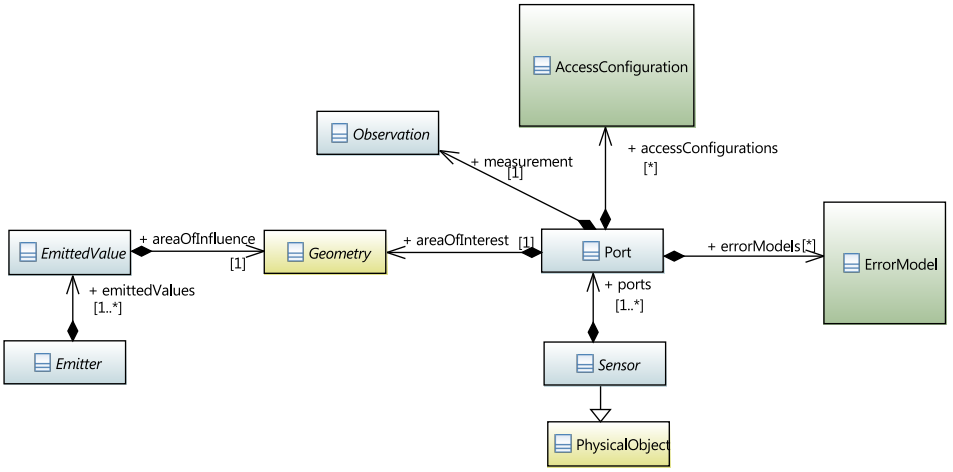

Abbildung 4.10: Vereinfachte UML Darstellung des Sense Modells zur strukturellen Beschreibung von Sensoren, die gelb hinterlegten Klassen werden in anderen Teilmodellen des World Data Models definiert, grün hinterlegte Elemente gruppieren Teile des Modells, die gesondert beschrieben werden.

Abgesehen von den im konzeptionellen Modell nicht behandelten Modellblöcken - Fehlermodell (*ErrorModel*) und Zugriffskonfiguration (*AccessConfiguration*) - unterscheidet

sich die Umsetzung des Modells, indem die im konzeptionellen Modell dem Sensor zugewiesenen Eigenschaften, durch eine neu eingeführte Klasse, den Port, realisiert werden. In diesem Fall wird ein Sensor über eine Menge von Ports beschrieben, von denen jeder Port ein mögliches Messergebnis des Sensors repräsentiert.
Durch diese Technik lassen sich Sensoren modellieren, die mehr als ein Messergebnis zur Verfügung stellen. Beispiel 1 demonstriert die Verwendung verschiedener Ports, an einem real existierenden Sensor.

Beispiel 1: Maritime Radargeräte

Das maritime Radar ist ein Beispiel für einen Sensor, der über mehrere Ausgänge verfügt. Zum einen stellen sie die beobachtete Umgebung in Form eines klassischen Radarbildes zur Verfügung (vgl. Abbildung 4.11, links). Zum anderen verfügen moderne Radargeräte über eine interne Datenverarbeitung, die die aufgenommenen Ziele (z.B. andere Schiffe) in Form von Objekthypothesen zur Verfügung stellt. Diese erkannten Ziele, werden u.a. in Form von NMEA0183 Datensätzen [Ao02] bereitgestellt (vgl. Abbildung 4.11 - rechts).

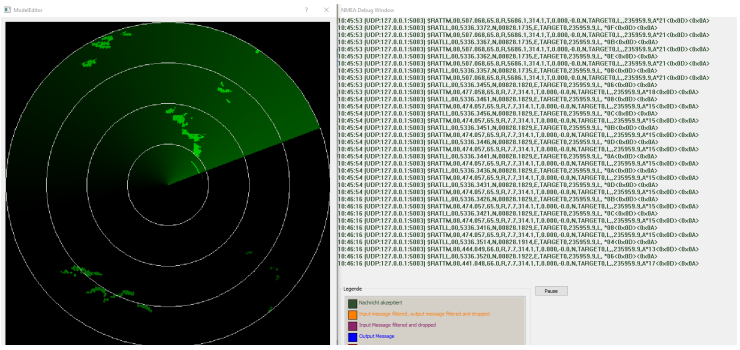

Abbildung 4.11: Ausgabeformate von maritimen Radargeräten; links: klassisches Radarbild; Rechts: NMEA0183 Nachrichten erkannter Ziele.

Die zweite Ergänzung der bisher erläuterten Konzepte sind die Fehlermodelle, die in Abbildung4.10, in gruppierter Form dargestellt sind. Bei diesen Fehlermodellen handelt es sich um eine Beschreibung des erwarteten Fehlers eines Sensors, bzw. dem erwarteten Fehler für eine Sensorbeobachtung.

Erwartete Fehler von Sensoren In die strukturelle Beschreibung von Sensoren wurde eine Beschreibung von Sensorfehlern und Messungenauigkeiten eingefügt, um den erwarteten Fehler des Sensors beschreiben zu können.
Dabei muss es sich aber nicht um die tatsächlich angewendeten Fehlermodelle handeln, welche durch die Verhaltensmodellierung beschrieben werden.
Diese Unterscheidung wurde eingeführt um zusätzliches Wissen über die Sensoren auch

für andere Simulationen oder Systeme verfügbar zu machen. In der Regel wird davon ausgegangen, dass es sich bei den erwarteten Fehlermodellen um einfachere Fehler handelt, als diejenigen die tatsächlich simuliert werden.
Eine genauere Beschreibung der in Abbildung 4.10 unter dem Begriff ErrorModel zusammengefassten Fehlermodelle wird in Abschnitt 4.3.3 vorgenommen.

Sensorbeobachtungen Für die Sensorbeobachtung wird eine abstrakte Datenstruktur zur Verfügung gestellt, die sogenannte *Observation* (vgl. Abbildung 4.10), welche dem *SensorOutput* der SSN bzw. der Beobachtung des konzeptionellen Modells, entspricht.
Diese Datenstruktur enthält zunächst keine Informationen über die eigentlichen Messwerte. Diese werden in spezialisierten Datenstrukturen hinzugefügt. In der vom *Sense* Modell definierten Version der Beobachtung enthält diese lediglich den Zeitpunkt zu dem sie erstellt wurde.
Ein Beispiel für eine spezialisierte Beobachtung wurde bereits in Abbildung 4.9 schematisch dargestellt.

Sensor Protokolle Es werden an dieser Stelle keine Anforderungen an das Format der Beobachtung gestellt; insbesondere wird hier nicht gefordert, dass sich eine Beobachtung in ihrer Struktur an einem bestehenden standardisierten Datenformat orientiert.
Wie beispielsweise dem NMEA 0183 Standard, wenn es sich um eine Positionsbeobachtung handelt. Die Umwandlung in ein sensorspezifisches Protokoll, wie NMEA 0183, erfolgt in einem späteren Schritt.
Diese Designentscheidung liegt vor allem in der folgenden Aussage von Andrew S. Tanenbaum [TW11, S. 702], begründet.

“Das Schöne an Standards ist, dass man so viele davon zur Auswahl hat“

Durch die Festlegung auf einen Standard würde man sich an dieser Stelle die Flexibilität nehmen, mittels eines simulierten Sensors verschiedene reale Sensoren (inkl. der von ihnen verwendeten Protokolle) abzubilden.
Zusätzlich lassen sich keine neuen Aspekte, wie zusätzliche Daten in die bestehenden Protokolle einbinden, wie es u.U. bei neu entwickelten Sensoren der Fall sein kann. Trotzdem sollte bei der Modellierung einer konkreten Beobachtung darauf geachtet werden, dass alle für einen zu unterstützenden Kommunikationsstandard benötigten Informationen vorliegen.

4.3.2.2 Komponentenmodell

Dieser Abschnitt des eigenen Ansatzes beschäftigt sich mit der Vorstellung des verwendeten datengetriebenen Komponentenmodells. Das Komponentenmodell wird in dem beschriebenen Ansatz zur flexiblen Verkettung von Basiskomponenten zu einem datengetriebenen Verarbeitungsgraphen genutzt, mit dessen Hilfe sowohl die Sensormesswerte erzeugt und verrauscht werden können, als auch die Bewertung der Sensordatenverarbeitung vorgenommen werden kann.

Dabei orientiert sich das verwendete Modell an Ansätzen, die in der Sensor- bzw. Signalverarbeitung verwendet werden und erweitert diese für den Einsatz in einer Sensorsimulation.
Das Komponentenmodell besteht dabei aus einer Beschreibung von einzelnen Komponenten und ihren Schnittstellen, dem sogenannten *Komponentenmetamodell* (vgl. Abschnitt 4.3.2.2), sowie einer Vorschrift, wie die Orchestrierung der einzelnen Komponenten sowie der Zugriff auf das Szenariomodell ermöglicht werden kann (vgl. Komponenten Linkmodell in Abschnitt 4.3.2.2).

Mithilfe des Komponenten-Metamodells und des Komponenten-Linkmodells wird ein datenstrombasierter Verarbeitungsgraph aufgebaut, wie er in ähnlicher Form z.B. in dem Datenstrommanagementsystem Odysseus [Bol11, GG12, BGG$^+$10] oder Matlab Simulink [Inc17] verwendet wird. Dieser Verarbeitungsgraph, im Folgenden auch Simulationsmodell genannt, wird in Abschnitt 4.3.2.2 um sogenannte Aktivierungsstrategien erweitert, welche dem Verarbeitungsgraphen Konzepte aus der ereignisorientierten Simulation [Hoe14, LCRP$^+$05, ISISC00] hinzufügen. Außerdem unterstützt die Methode die Verwendung eines gemeinsamen Speichers, bestehend aus Elementen des World Data Modells, auf die die Komponenten zugreifen können, um beispielsweise den Kontext eines Sensors zu bestimmen.

Die Kombination dieser Konzepte, stellt einen wesentlichen Beitrag der vorgestellten Methode zur simulativen Bewertung von sensordatenverarbeitenden Systemen dar, indem die verschiedenen Vorteile miteinander kombiniert werden um eine einfache aber dennoch effiziente Simulation von Sensoren zu ermöglichen.

Dabei profitiert der Ansatz durch die Verwendung eines datenorientierten Verarbeitungsgraphen davon, dass die einzelnen Komponenten verhältnismäßig einfach gehalten werden können aber in ihrer Kombination dennoch in der Lage sind, ein komplexes Verhalten abzubilden. Ein Beispiel, welches zusätzlich die Bewertung berücksichtigt, ist schematisch in Abbildung 4.5 dargestellt, In dem Beispiel wird die Konkatenation zweier Fehlermodelle genutzt wird um einen komplexeren Fehler zu modellieren.

Mithilfe dieser Fähigkeit lassen sich beliebige Fehlermodelle zusammenstellen ohne dass das zugrundeliegende Sensormodell angepasst werden muss. Ähnliche Kombinationen lassen sich auch für Erzeugung von Sensormesswerten erstellen, bei denen beispielsweise sukzessiv Informationen zu der Beobachtung hinzugefügt werden.
Die Einführung der Aktivierungsstrategie und damit die Einbindung der ereignisorientierten Simulation, gleicht den entscheidenden Nachteil des datenstrombasierten Verarbeitungsgraphen aus, dass die Quellen des Graphen nicht von außen aktiviert werden, wie es beispielsweise bei dem Datenstrommanagementsystem Odysseus der Fall ist.

Eine ähnliche Kombination dieser Konzepte konnte in der Literatur bisher nicht identifiziert werden, stattdessen finden sich dort Kombinationen mit zwei der vorgestellten Konzepte. So verbinden zum Beispiel Matlab Simulink und Modellica [Inc17, AT] eine graphenbasierte Komposition von Komponenten mit einer simulierten Zeit und profitieren dabei sowohl von der Wiederverwendbarkeit der Komponenten als auch von ihrer einfachen Komposition zur Lösung eines komplexen Problems. Dabei beschränken Sie sich aber im Wesentlichen auf die Verwendung von einfachen Umgebungsmodellen bzw. Eingangswerten (Skalare und Matrizen) mit einer beschränkten Dynamik der Eingangswerte.
Ein weiteres Beispiel ist das bereits erwähnte Datenstrommanagement System Odysseus, welches einen datenstrombasierten Ansatz verwendet um Operatoren, vornehmlich aus dem Bereich der relationalen Algebra, miteinander zu verknüpfen. Dabei kann Odysseus bei Bedarf mit einer klassischen Datenbank verknüpft werden, welche als ein gemeinsamer Speicher angesehen werden kann. Da es sich bei Odysseus um eine sensordatenverarbeitende Software handelt, unterstützt es keine eigene Zeitdefinition gemäß einer ereignisorientierten Simulation.
Diese wird hingegen von der High Level Architecture unterstützt, die diese mit einem verteilten, gemeinsamen, Speicher verbindet aber keine flexible Kombination von einzelnen Komponenten zulässt, die Datengetrieben innerhalb eines Zeitschrittes ausgeführt werden.

Komponenten Metamodell Mithilfe des in Abschnitt 4.2.1.1 beschriebenen Metamodells lässt sich die Struktur von Objekten formal spezifizieren. Allerdings können keine Aussagen über das Verhalten der, durch das Metamodell beschriebenen, Objekte gemacht werden.
Die im Folgenden vorgestellten Komponenten ermöglichen die Beschreibung des Verhalten von Objekten und insbesondere von Sensoren. Dabei versteht sich eine Komponente als eine, in sich geschlossene, funktionale Einheit die nur über ihre Ein- und Ausgänge beschrieben wird. Gegenüber dem Komponenten- bzw. Simulationssystem verfügt eine Komponente über die folgenden, charakterisierenden Eigenschaften:

Blackbox Die interne Funktionsweise einer Komponente ist dem Komponentensystem nicht bekannt.

Öffentliche Schnittstellen Eine Komponente verfügt über öffentliche Schnittstellen, die für die Konfiguration der Komponente sowie zur Komposition mit anderen Komponenten im Komponentensystem verwendet werden können.

Kontextunabhängigkeit Eine Komponente ist unabhängig vom aktuellen Anwendungskontext, d.h. neben den öffentlichen Schnittstellen werden keine weiteren Informationen für die Ausführung der Komponente benötigt.

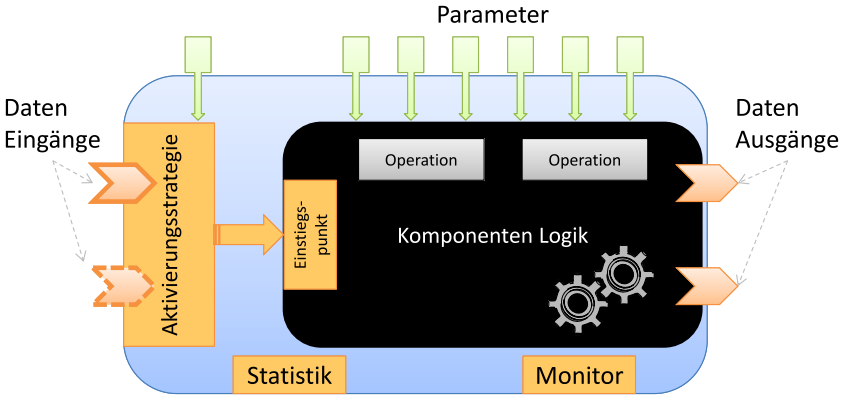

Abbildung 4.12: Schematische Darstellung einer Komponente

Abbildung 4.12 zeigt den schematischen Aufbau einer solchen Komponente, mit ihren Schnittstellen zum Benutzer (dargestellt durch die Parameter) und die Schnittstellen zu anderen Komponenten (dargestellt durch die Daten Ein- und Ausgänge). Neben diesen öffentlichen Schnittstellen, die für jede Komponente definiert werden können, gibt das Komponenten-Metamodell (KMM) einen Rahmen für den Aufbau der einzelnen Komponente vor.
Dazu gehören die Aktivierungsstrategie und ein Einstiegspunkt, mit dem die Ausführung der Komponente gestartet wird. Ebenfalls Bestandteil des Rahmenaufbaus sind eine kleine Statistik bezüglich der Ausführung sowie eine Möglichkeit den Zustand der Komponente zu beobachten.

Abbildung 4.13 stellt das Komponenten Metamodell für die in Abbildung 4.12 schematisch dargelegte Komponente dar.
Die einzelnen Bestandteile werden in den folgenden Abschnitten genauer diskutiert, angefangen mit den öffentlichen Schnittstellen zu anderen Komponenten, repräsentiert durch die Klassen *InputFlowPort* und *OutputFlowPort*, sowie die Aktivierungsstrategie (*ActivationPolicy*) und ihre konkreten Ausprägungen.

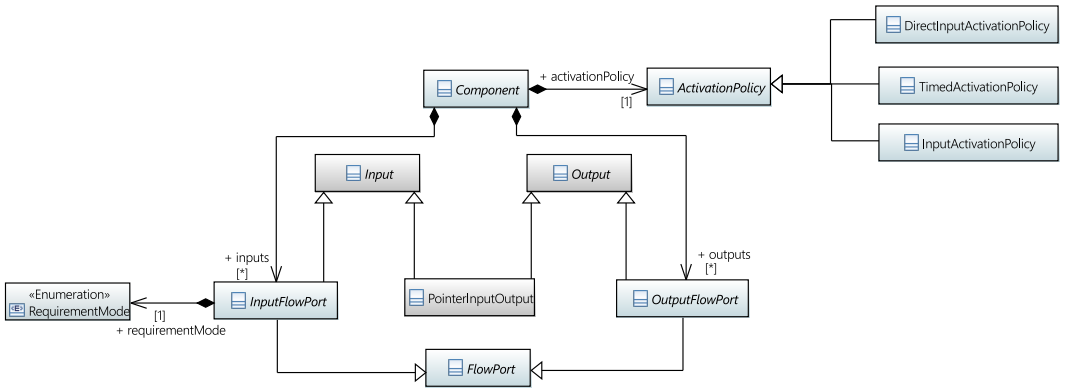

Abbildung 4.13: Vereinfachte UML Darstellung des Komponenten Metamodells.

Komponenten Schnittstellen Eine Komponente, wie sie in Abbildung 4.7 dargestellt ist, verfügt über zwei Arten von Schnittstellen. Zum einen kann eine Komponente über beliebig viele Parameter verfügen mit denen ihr Verhalten für den jeweiligen Anwendungskontext konfiguriert werden kann. Zum anderen existieren Schnittstellen für die Kommunikation mit anderen Komponenten.
Bei der ersten Schnittstelle, den Parametern, handelt es sich um StructuralFeatures, welche ohne Änderung aus dem zuvor vorgestellten Metamodell (vgl. Abschnitt 4.2.1.1)übernommen wurden.

Die zweite Art von Schnittstellen stellt die eigentliche Erweiterung gegenüber dem Metamodell dar.
Bei diesen Schnittstellen handelt es sich um Datenschnittstellen mit deren Hilfe erzeugte oder manipulierte Daten in Form eines Datenstroms zwischen zwei Komponenten ausgetauscht werden. Gleichzeitig stellen ankommende Daten einen möglichen Auslöser für die Ausführung der Komponentenlogik dar. Ausgehend von der Verteilung der Datenschnittstellen, kann eine Einordnung der Komponente in die folgenden Gruppen vorgenommen werden.

Quellen Komponenten, die über keine Dateneingänge verfügen.

Senken Komponenten, die über keine Ausgänge verfügen.

Verarbeitung Komponenten die sowohl über Eingänge als auch Ausgänge verfüggen.

Sowohl die Dateneingänge als auch die Datenausgänge sind streng getypt um sicherzustellen, dass die Komponenten die eingehenden Daten verarbeiten können. Die Typen sind auf Datentypen und Konzepte beschränkt, die zuvor durch das Metamodell beschrieben wurden, sowie primitive Datentypen (boolean, integer, float, ...). Wie in der

klassischen objektorientierten Programmierung werden an dieser Stelle polymorphe Datenstrukturen zugelassen. Über einen Datenausgang bzw. Dateneingang können Spezialisierungen des angegebenen Datentyps kommuniziert werden.

Die Verfügbarkeit der benötigten Daten entscheidet über die Ausführung der Komponente. Erst wenn alle benötigten Daten vorliegen, wird die Komponentenlogik ausgeführt. Dabei muss es sich nicht bei jedem Dateneingang um einen benötigten Dateneingang handeln. Während der Definition der Komponente kann für jeden Eingang festgelegt werden, ob es sich um einen benötigten oder einen optionalen Dateneingang handelt.

In der Folge kann die Komponentenlogik auch dann ausgeführt werden, wenn nicht alle verfügbaren Dateneingänge mit gültigen Daten belegt sind, sofern es sich bei den nicht belegten Eingängen um optionale Dateneingänge handelt.
Optionale Dateneingänge bieten die Möglichkeit bestimmte Informationen mit in die Komponentenlogik mit einzubeziehen, sofern diese verfügbar sind. Dabei kann es sich beispielsweise um eine Optimierungsstrategie handeln oder um eine Technik, die das explizite Modellieren von ausbleibenden Informationen erlaubt Die Datenausgänge verfügen dagegen immer über eine optionale Semantik, d.h. die Komponentenlogik kann selbstständig entscheiden ob und wann (während ihrer Ausführung) sie Daten über einen Ausgang zur Verfügung stellt.
Als eine Besonderheit erlaubt das Komponentenmodell das wiederholte “feuern“ eines Ausgangs während der Ausführung der Komponente (vgl. Anmerkung 4.3.1 - DirectInput Aktivierungsstrategie). In diesem Fall entscheidet die Aktivierungsstrategie der nachfolgenden Komponenten über die Verfügbarkeit der Ergebnisse.

Aktivierungsstrategie Bei dem vorgestellten Komponentensystem handelt es sich in erster Linie um ein datengetriebenes Komponentensystem. Das bedeutet, dass eine Komponente ausgeführt werden kann, sobald alle benötigten Daten an den entsprechenden Eingängen vorliegen. Dies gilt allerdings nicht für Datenquellen, also Komponenten die über keine Dateneingänge verfügen, wie es bei den simulierten Sensoren der Fall ist. Diese Komponenten können zeitgesteuert aktiviert werden. In diesem Fall übernimmt eine sogenannte Aktivierungsstrategie (*ActivationPolicy* in Abbildung 4.13) die Aufgabe, die Komponentenlogik zu starten. Es kann zwischen drei Strategien gewählt werden:

Zeitgesteuerte Aktivierungsstrategie (*TimedActivationPolicy* in Abbildung 4.13)
Die zeitgesteuerte Aktivierungsstrategie entspricht dem Vorgehen bei einer diskreten Eventsimulation. In diesem Fall wird die entsprechende Komponente in regelmäßigen Zeitabständen aktiviert, sofern alle benötigten Daten vorliegen. Ist dies nicht der Fall, wird die Aktivierung der Komponente übersprungen und nach

Ablauf des angegebenen Zeitintervalls eine erneute Aktivierung der Komponente versucht.

Datenstrom-Aktivierungsstrategie (*InputActivationPolicy* in Abbildung 4.13)
Die Semantik der Datenstromaktivierungsstrategie entspricht der Semantik eines datengetriebenen Komponentenframeworks. Sobald ein neues Datum an einem Eingang anliegt wird überprüft, ob die Komponentenlogik gestartet werden kann, indem die Verfügbarkeit von validen Daten an allen benötigten Eingängen überprüft wird. Kann die Komponente ausgeführt werden, wird sichergestellt das die Komponente innerhalb des aktuellen Zeitschrittes ausgeführt wird. Wurde die Komponente während des aktuellen Zeitschrittes bereits ausgeführt, wird die Aktivierung für den nächsten Zeitschritt vorgemerkt um sicherzustellen, dass es nicht zu einer Endlosschleife kommt.

***DirectInput*-Aktivierungsstrategie** (*DirectInputActivationPolicy* in Abbildung 4.13)
Die *DirectInput* Aktivierungsstrategie verhält sich ähnlich zu der Datenstromstrategie und führt die Komponentenlogik aus, sobald ein neues Datum an einem der Dateneingänge anliegt, sowie alle benötigten Daten vorhanden sind.
Im Gegensatz zu der Datenstromaktivierungsstrategie, welche mit einem asynchronen Methodenaufruf vergleichbar ist, ist die *DirectInput* Strategie mit einem synchronen Methodenaufruf vergleichbar und wird im gleichen Kontext wie die datenerzeugende Komponente ausgeführt.

Anmerkung 4.3.1 (*DirectInput Aktivierungsstrategie*)

Während die Zeitgesteuerte- und die Datenstromaktivierungsstrategie in der Lage sind, zyklische Graphen abzubilden, ist dies bei der DirectInput Strategie nicht der Fall, da ein Zyklus bestehend aus Komponenten mit der DirectInput Strategie zu einer Endlosschleife führen kann. Der Einsatz dieser Aktivierungsstrategie ermöglicht allerdings die Zerlegung von Listen eines Datentyps in die einzelnen Bestandteile der Liste, sowie die Präsentation von Zwischenergebnissen, die während einer der Abarbeitung einer Komponente anfallen.
Sie sollte verwendet werden, wenn ein Datenausgang während der Ausführung der Komponentenlogik mehr als einmal mit neuen Daten belegt wird und alle Daten verarbeitet werden sollen. Wird stattdessen eine der beiden anderen Aktivierungsstrategien verwendet, wird nur das aktuellste Datum von den Nachfolgenden Komponenten zur Bearbeitung herangezogen.

Wie in Abbildung 4.13 dargestellt ist, verfügt jede Komponente über eine Aktivierungsstrategie, unabhängig davon ob es sich um eine Quelle, eine Senke oder eine Verarbeitungskomponente handelt.
Hierdurch ergeben sich zwei Implikationen:

1. Eine Quelle kann mit einer Datenstrom- oder DirectInputaktivierungsstrategie konfiguriert werden.
   In diesem Fall wird die Komponente nicht aktiviert. Es liegt in der Verantwortung des Anwenders eine solche Belegung zu vermeiden.

2. Eine Verarbeitungskomponente kann mit einer zeitgesteuerten Aktivierungsstrategie konfiguriert werden.
   Dieses Verhalten eignet sich für die Modellierung von virtuellen Sensoren innerhalb des Komponentensystems, die mit unterschiedlichen Frequenzen arbeiten können. Abbildung 4.14 stellt eine solche Modellierung beispielhaft dar. In dem dargestellten Komponentensystem wird ein AIS-Transmitter als ein virtueller Sensor modelliert, der das GPS-Signal eines entsprechenden Differential GPS (DGPS)-Sensors verwendet. Dabei arbeitet der DGPS-Sensor üblicherweise mit einer Frequenz von ca. 5 Hz, während ein AIS-Transmitter über eine maximale Frequenz von 0.5Hz verfügt[5]. Gemäß des in Abschnitt 4.3.2.2 beschriebenen Verhalten der Verbindungen, steht dem AIS-Transmitter auf diese Weise immer die aktuellste GPS Position zu Verfügung.

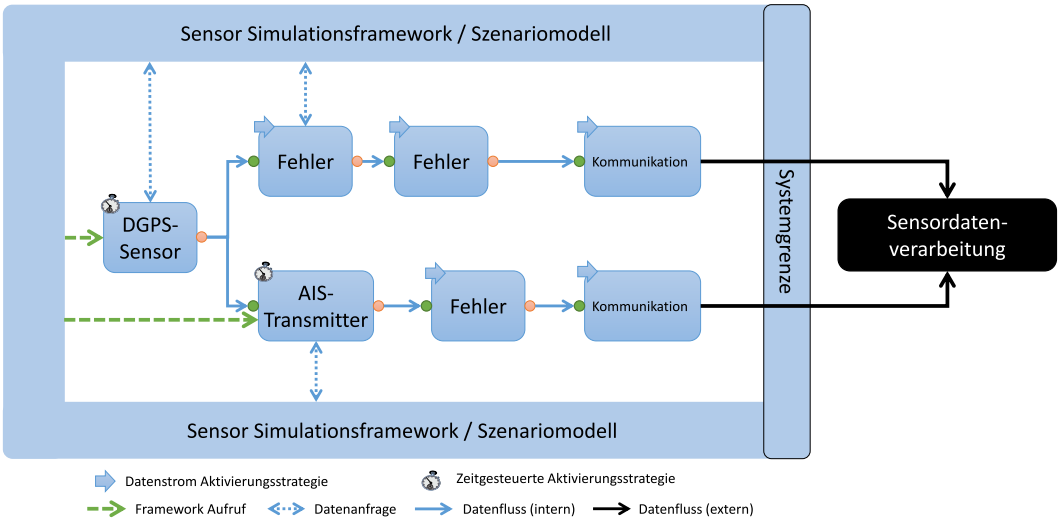

Abbildung 4.14: Beispiel Komponentensystem mit virtuellem Sensor und Aktivierungsstrategien

Komponentenlogik Bei der Komponentenlogik handelt es sich, wie bereits beschrieben, um eine Black Box, dessen interne Struktur vom Komponentensystem nicht verwendet wird. Die Umsetzung erfolgt durch eine manuelle Implementierung des Einstiegpunktes. Damit kann eine einzelne Komponente wie ein eigenständiges Programm verstanden werden, wobei der Einstiegspunkt der “main“ Methode eines herkömmlichen Programmes entspricht.

[5]Vergleiche IALA guidline on universal shipborne automatic identification system (AIS) [IAL01]

Aufgrund dieser freien Definition der Komponentenlogik lassen sich auch externe Programme, wie beispielsweise eine externe Simulation in das Komponentensystem integrieren, indem die Komponentenlogik einen “Wrapper“ um die externe Simulation darstellt.

Komponenten Linkmodell Während sich das Komponenten Metamodell mit dem Aufbau einer einzelnen Komponente beschäftigt hat, wird im Rahmen des Komponenten Linkmodells die Orchestrierung mehrerer Komponenten miteinander behandelt.
Zusätzlich wird der Zugriff auf einen gemeinsam genutzten Speicherbereich beschrieben, der unter anderem das zuvor beschriebene Szenariomodell und damit die für die Simulation benötigten Informationen enthält.

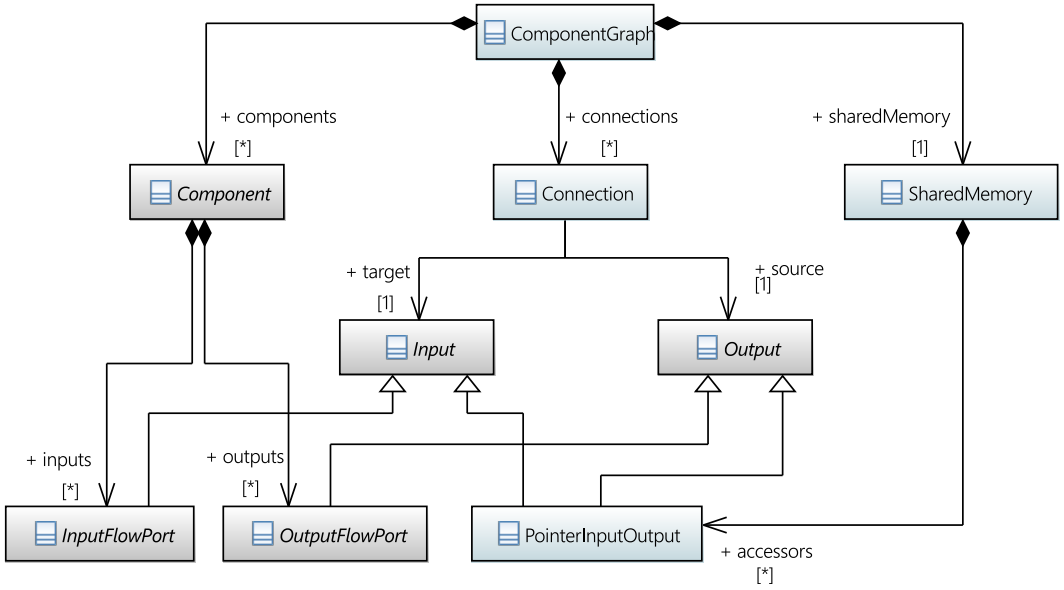

Abbildung 4.15: Vereinfachte UML Darstellung der Zusammenhänge des Komponenten Linkmodells, Grau hinterlegte Klassen wurden bereits im KMM Abschnitt beschrieben

Orchestrierung Zwischen einem Datenausgang und einem Dateneingang können Verbindungen erzeugt werden (*Connection* in Abbildung 4.15), die die beim Ausgang erzeugten Daten an den entsprechenden Eingang weiterleiten. Dabei gilt die Regel, dass jeder Dateneingang mit genau einem Datenausgang verbunden werden kann. Der Datenausgang dagegen kann mit beliebig vielen Eingängen verknüpft werden.
Durch die Verbindung zweier Komponenten, bzw. deren Ein- und Ausgänge, wird eine private Kommunikation zwischen diesen etabliert. Eine private Kommunikation in diesem Sinne bedeutet, dass keine andere Komponente und kein anderes System über den Zeitpunkt sowie die kommunizierten Daten informiert wird. Die Aufgabe der Verbindung beschränkt sich dabei auf eine reine Datenweiterleitung. Dabei werden keine

Werte innerhalb der Verbindung zwischengespeichert. Dies steht im Gegensatz zu anderen Komponentensystemen, wie beispielsweise dem von L. Safie [Saf12] vorgestellten Komponentensystemen, bei dem zwischen verschiedenen Verbindungstypen unterschieden werden kann, die sich in der Zwischenspeicherung von Daten unterscheiden.
Stattdessen geschieht die Speicherung der kommunizierten Daten innerhalb der Dateneingänge. Dort verbleiben die angekommenen Daten solange bis sie durch neue Daten ersetzt werden oder manuell von der Komponentenlogik gelöscht werden. Das Verbleiben eines Datums an einem Dateneingang ermöglicht es, dass die beiden beteiligten Komponenten mit unterschiedlichen Frequenzen betrieben werden, wie es im Abschnitt über die Aktivierungsstrategie beschrieben wurde. Gleichzeitig ermöglicht diese Technik das Bereitstellen von statischen Daten, zum Beispiel aus dem im Folgenden beschriebenen, gemeinsamen Speicherbereich.

Zugriff auf das Szenariomodell Zur Speicherung von simulationsrelevanten Informationen wird ein gemeinsamer Speicherbereich für alle Komponenten vorgehalten (SharedMemory in Abbildung 4.15).
Elementarer Bestandteil des gemeinsamen Speichers ist das zuvor beschriebene Szenariomodell, welches sowohl die statischen als auch dynamischen Elemente des zu simulierenden Szenarios enthält. Neben diesem können weitere, rein sensorsimulationsspezifische Informationen hinterlegt werden. Dabei kann es sich beispielsweise um zusätzliche Informationen handeln, die zur Ausführung einer Komponente benötigt werden aber nicht für andere Simulationen sichtbar sein sollen.

Alle in diesem Speicherbereich enthaltenen Informationen können mit den Dateneingängen bzw. Datenausgängen der Komponenten verbunden werden, um die entsprechenden Informationen für eine Komponente bereitzustellen. Dies geschieht über einen speziellen Zeiger auf ein Datenelement im gemeinsamen Speicher (*PointerInputOutput* in Abbildung 4.15), der sowohl als Eingang als auch als Ausgang genutzt werden kann. Das Zeigerelement zeigt dabei auf eine spezifische Position innerhalb des gemeinsamen Speichers und ist nicht an eine konkrete Instanz eines Datums gebunden. Auf diese Weise kann auf Änderungen des Speichers reagiert werden (vgl. Aufsplitten von Informationen). Dabei stellt das Komponentensystem sicher, dass die Daten vor der ersten Ausführung einer Komponente an diese kommuniziert werden.
Wird ein Speicherbereich als Datenquelle verwendet, stellt das Komponentensystem sicher, dass Änderungen an den entsprechenden Daten an die angebundenen Komponenten weitergereicht werden. Dabei ist es nicht von Bedeutung, ob die Daten durch eine Komponente geändert wurden sind, die den gemeinsamen Speicher als Datensenke verwendet oder ob eine manuelle Änderung der Daten vorgenommen wurde.

Aufweichung der Typ Bindung Mithilfe des gemeinsamen Speicherbereiches lassen sich die Informationen aus dem gemeinsamen Datenmodell in ihre Bestandteile aufsplitten. Zu diesem Zweck kann ein neues Speicherelement angelegt werden, welches von einer Komponente als Datensenke verwendet wird. Handelt es sich bei dem Datenelement um eine komplexe Datenstruktur, wie sie in Abschnitt 4.2.1.1 beschrieben wurde, können Teile dieser Struktur wiederum als Quelle oder Senke für andere Elemente verwendet werden.

Abbildung 4.16 zeigt eine solche Aufsplittung der Daten an dem bereits bekannten

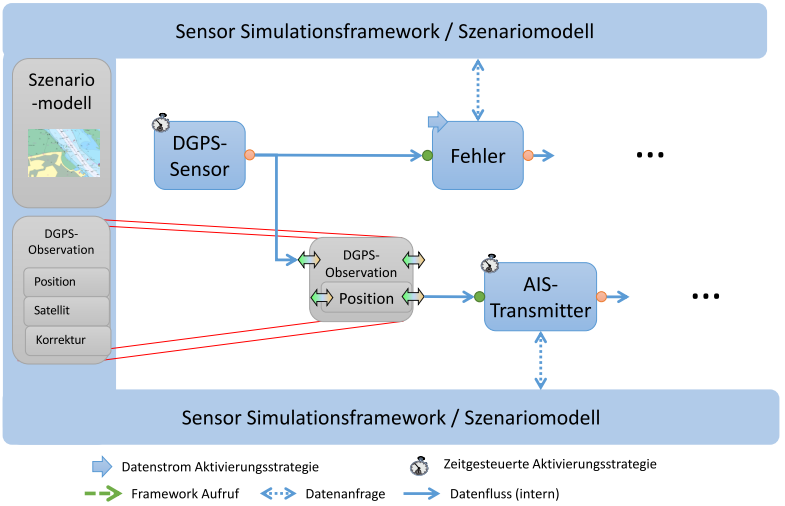

Abbildung 4.16: Verwendung des gemeinsamen Speicherbereiches zur Aufsplittung von Teilergebnissen; Bereiche aus dem gemeinsamen Speicher werden der Übersichthalber als Komponenten im Datenflussgraphen verwendet.

DGPS und AIS Beispiel. Der AIS-Transceiver benötigt als eingehendes Datum eine Position, während der DGPS-Sensor eine DGPS Beobachtung liefert. Diese DGPS Beobachtung beinhaltet die vom AIS-Transceiver benötigte Position.
Im gemeinsamen Speicher wird eine neue Instanz einer DGPS-Beobachtung angelegt und ggf. mit gültigen Werten initiiert. Dieses neue Speicherelement wird mit dem Ausgang des DGPS-Sensors verbunden und somit von diesem als eine Datensenke verwendet. Sobald der DGPS-Sensor ein neues Datum erzeugt, wird die Instanz der Beobachtung im gemeinsamen Speicher aktualisiert und damit auch alle nachfolgenden Komponenten benachrichtigt. Bestandteil der Aktualisierung ist ebenfalls die benötigten Position, welche wiederum mit dem AIS Transceiver verbunden ist.
Diese Aufsplittung lässt sich bis auf die Grundelemente eines komplexen Datentyps durchführen und ermöglicht im Wesentlichen die zuvor beschriebene Schematransformation zur Erzeugung von Sensorbeobachtungen.

4.3.3 Modellierung von Sensorfehlern und Messungenauigkeiten

Dieser Abschnitt beschäftigt sich mit der Modellierung der in Abschnitt 2.3 vorgestellten Sensorfehler und Messungenauigkeiten sowie ihrem Einsatz in dem in Abschnitt 4.3.2.2 vorgestellten Simulationsmodell.
Dazu wird zunächst die strukturelle Beschreibung der Fehlermodelle, in Ergänzung zu der in Abschnitt 4.3.2.1 vorgestellten strukturellen Beschreibung der Sensoren vorgestellt. Dabei handelt es sich in erster Line um die Beschreibung des erwarteten Fehlers eines Sensors, der ggf. mit anderen Simulationen oder auch der Sensordatenverarbeitung geteilt werden kann.
Anschließend wird auf die Modellierung der Fehlerkomponenten eingegangen. Dabei handelt es sich um generische Fehlermodelle, welche auf jede Sensorbeobachtung angewendet werden können und sich dadurch von den in Abschnitt 3.3 identifizierten Fehlermodellen unterscheiden, die i.d.R. durch eine Integration in die Sensormesswertsynthese realisiert wurden.
Weiterhin können die verschiedenen Fehlerkomponenten nach dem Superpositionsprinzip miteinander kombiniert werden, indem sie innerhalb des Simulationsmodells nacheinander auf eine Sensorbeobachtungen angewendet werden, zum Beispiel in Form einer Fehlerkette oder in Verbindung mit zusätzlichen Komponenten. Diese Möglichkeit wird unter anderem zur Modellierung von kontextsensitiven Fehlern verwendet (vgl. Abschnitt 4.3.3.4), kann aber auch zur dekomposition von komplexen Fehlermodellen verwendet werden um einzelne Aspekte des komplexen Modells bzw. dessen Auswirkungen auf die Sensordatenverarbeitung genauer zu untersuchen.
Zusätzlich wird bei den einzelnen Fehlerkomponenten darauf eingegangen, wie ihre Auswirkungen und die ggf. dadurch zustande kommende Verletzung der Qualitätskriterien im Rahmen der anschließenden Bewertung (vgl. Abschnitt 4.4) erkannt werden können.

4.3.3.1 Strukturelle Beschreibung von Sensorfehlern und Messungenauigkeiten

Die strukturelle Beschreibung der Sensorfehler und Messungenauigkeiten und damit die Erweiterung des in Abbildung 4.10 dargestellten Sensormodells, ist in Abbildung 4.17 abgebildet.

Sie besteht aus den bereits eingeführten Gruppen messtechnische Fehler bzw. Messungenauigkeiten, welche durch die beiden Klassen *SystematicError* und *StatisticError* repräsentiert werden, die Gruppe der Fehlerzustände (*ErrorStateModel*), für die Beispielhaft zwei Ausprägungen in Abbildung 4.17 dargestellt sind.

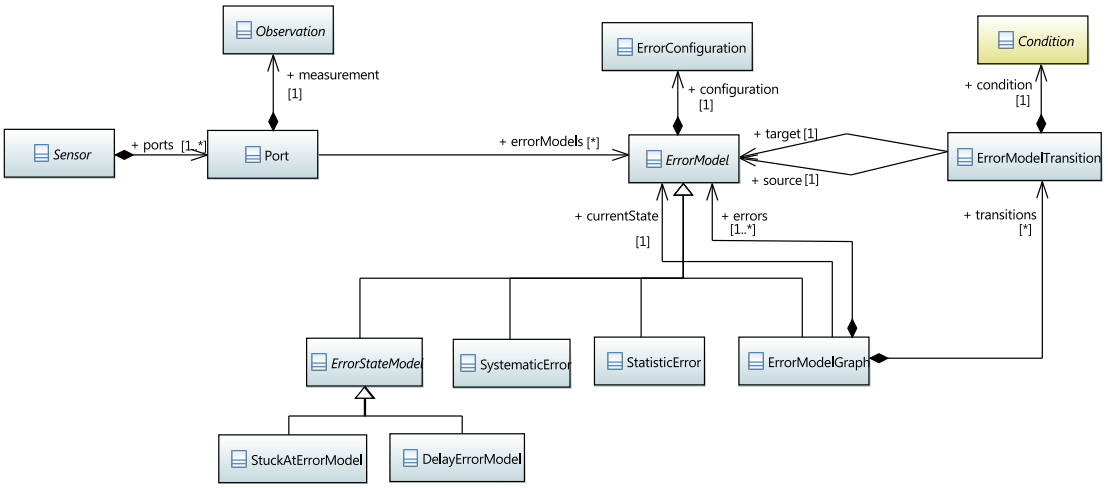

Abbildung 4.17: Vereinfachte UML Darstellung der Fehlermodelle aus dem *Sense* Teilmodel, die gelb hinterlegte Klasse wird in einem anderen Teilmodell definiert.

Zusätzlich kann ein Fehler Graph (*ErrorModelGraph*), durch die Kombination von anderen Fehlermodellen erstellt werden.

Eine Besonderheit bei der Beschreibung der Sensorfehler ist die sogenannte Fehlerkonfiguration (*ErrorConfiguration*). Bei ihr handelt es sich um eine Vorschrift, auf welche Bestandteile einer Beobachtung ein Fehlermodell angewendet werden darf. Durch die Einführung der Fehlerkonfiguration lässt sich beispielsweise die Veränderung von Meta-Daten innerhalb einer Beobachtung unterbinden. Hauptsächlich wird sie aber verwendet um mithilfe des Daten-Metamodells aus Abschnitt 4.2.1.1 eine generelle Anwendbarkeit der Fehlermodelle zu garantieren, indem für jede Sensorbeobachtung eine entsprechende Fehlerkonfiguration bereitgestellt werden kann, über die die Anwendung des Fehlers gesteuert wird.

Fehlerkonfiguration Die Angabe, auf welche Eigenschaften einer Beobachtung sich das Fehlermodell bezieht, geschieht über die Angabe der zu verändernden Eigenschaften (*ErrorConfiguration* in Abbildung 4.17) für die erwartete Beobachtungsklasse ($K.features$). Dabei profitiert die Fehlerkonfiguration von der formalen Spezifikation der Beobachtung durch das in Abschnitt 4.2.1.1 beschriebene Daten-Metamodell.
Um eine unerwünschte Manipulation unveränderlicher Daten zu vermeiden, wird an dieser Stelle eine “closed world assumption“ gewählt. Bei dieser müssen die vom Fehlermodell zu manipulierenden Eigenschaften der Beobachtung explizit angegeben werden. Im Umkehrschluss werden nicht explizit angegebene Eigenschaften nicht von einem Fehlermodell verändert. Mithilfe der Fehlerkonfiguration lässt sich das Verhalten der zufälligen bzw. systematischen Fehlerkomponente durch den folgenden Algorithmus beschreiben:

Input: Beobachtung: IN
Data: Fehlerkonfiguration K
Result: Beobachtung : OUT
for $f \in K.features$ **do**
  if f *isMultiplicity* > 1 **then**
    for $X_{in_i} \in Value_f$ **do**
      $X_{out_i} = X_{in_i} + error();$
    end
  else
    $X_{out} = Value_f + error();$
  end
end

Algorithm 1: Verhalten der messtechnischen Fehlerkomponente

Bedingt durch die Tatsache, dass sich Fehlerzustände immer auf die gesamte Beobachtung beziehen (vgl. Abschnitt 4.3.3.3) kommt die Fehlerkonfiguration bei der Anwendung von Messunsicherheiten und ggf. bei kontextsensitiven Fehlern zum Einsatz.

Wie bereits beschrieben handelt es sich bei der Beschreibung des erwarteten Fehlers um eine optionale Beschreibung, die bei Bedarf mit anderen Simulationen in der Ko-Simulation geteilt werden kann. Zusätzlich wird die strukturelle Beschreibung auch für die Konfiguration der Fehlermodellkomponenten verwendet, die in den folgenden Teilabschnitten genauer vorgestellt werden.

4.3.3.2 Sensorfehler

Die Sensorfehler *Malfunction*, *StuckAt* oder *Delay*, werden in der strukturellen Beschreibung, durch die Klasse *ErrorStateModel* bzw. ihre Spezialisierungen repräsentiert.
Generell werden sie durch einen einfachen probabilistischen Zustandsautomaten beschrieben der angibt, mit welcher Wahrscheinlichkeit der Fehler aus dem Zustand *Aktiv* in den Zustand *Inaktiv* wechselt (P_{aktiv}) bzw. vom Zustand *Inaktiv* in den den Zustand *Aktiv* ($P_{inaktiv}$) (vgl. Abbildung 4.18).

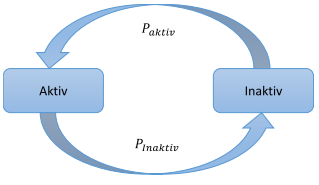

Abbildung 4.18: Einfacher Zustandsautomat zum Entscheiden, ob ein *ErrorStateModel* aktiv oder inaktiv ist.

Dabei gilt, dass alle Sensorfehler eine eingehende Beobachtung unverändert weiterleiten solange sich der Zustandsautomat im Zustand *Inaktiv* befindet. Ansonsten werden die folgenden Modelle verwendet.

Sensorausfall - Malfunction Bei dem Sensorausfall Fehlermodell handelt es sich um den einfachsten Sensorfehler. Er verfügt über keine zusätzlichen Parameter, neben den Aktivierungswahrscheinlichkeiten.
Das Verhalten entspricht dem folgenden Pseudocode Abschnitt, bei dem die Beobachtung verworfen wird, sofern sich der Fehler im aktivierten Zustand befindet.

Input: Beobachtung: IN
Result: Beobachtung : OUT
prüfeZustandsänderung(); ← Prüft und ändert ggf. den Zustand
if *Zustand == Aktiv* **then**
| OUT = NULL;
else
| OUT = IN;
end

Algorithm 2: Verhalten des Sensorausfall Fehlermodells

Festsitzend - StuckAt Beim StuckAt Sensorfehler wird ein historischer Sensormesswert anstelle der tatsächlichen Messung versendet. In dem verwendeten Modell verfügt dieser Sensorfehler über einen zusätzlichen Parameter - die *Stuck-Duration (t_{SA})* - der angibt, wie lange der Fehler den festgesetzten Wert weitergeben soll.
Dieser Parameter wurde eingeführt, da ansonsten die Aktivierungsdauer bei großen Werten von P_{aktiv} so gering ausfällt das der Fehler bei der Sensordatenverarbeitung nicht ins Gewicht fällt. Dieses Verhalten kann aber durch eine entsprechend niedrig gewählte *Stuck-Duration* wiederhergestellt werden.
Das Verhalten des StuckAt Fehlers lässt sich durch den folgenden Pseudocode Abschnitt beschreiben:

Input: Beobachtung: IN
Data: Beobachtung: H ← Historischer Messwert als Membervariable
Data: Zeit: t_a ← Zeit seit der Aktivierung
Data: Zeit: t_v ← Zeit seit der letzten Ausführung der Komponente
Result: Beobachtung : OUT
prüfeZustandsänderung(); ← Prüft und ändert ggf. den Zustand
if $t_a < t_{SA}$ || *Zustand == Aktiv* **then**
| OUT = H;
| $t_a += t_v$;
else
| OUT = IN;
| H = IN;
end

Algorithm 3: Verhalten des StuckAt Fehlermodells

Verzögerung - Delay Das Delay Fehlermodell verzögert die Weitergabe einer Sensormessung um eine festgelegte Zeit (t_{delay}). Dabei wird ein Warteschlangenkonzept für die Umsetzung des Fehlverhaltens verwendet, d.h. die Messwerte werden solange in einer Warteschlange gespeichert, bis sie von der Komponente abgerufen werden.
Um einen Speicherüberlauf bei der Implementierung zu verhindern, kann zusätzlich eine maximale Länge der Warteschlange angegeben werden. Ansonsten verhält sich das Delay Fehlermodell wie im Folgenden Pseudocode beschrieben.

Input: Beobachtung: IN
Data: Zeit: $t_a \leftarrow$ Zeit seit der Aktivierung
Data: Zeit: $t_v \leftarrow$ Zeit seit der letzten Ausführung der Komponente
Data: Zeit: $t \leftarrow$ Aktuelle Simulationszeit
Result: Beobachtung : OUT
queue.push(IN);
prüfeZustandsänderung(); ← Prüft und ändert ggf. den Zustand
if $t_a < t_D$ || *Zustand == Aktiv* **then**
  OUT = queue.get(t_{delay}) ← erstes Element X ∈ queue für das gilt: X.t < $t - t_{delay}$;
  $t_a += t_v$;
else
  OUT = IN;
end

Algorithm 4: Verhalten des Delay Fehlermodells

Genau wie der StuckAt Fehler kann auch bei dem Delay Fehlermodell eine Aktivierungsdauer (t_D) angegeben werden um das Verhalten der Komponente bei kleinen Aktivierungswahrscheinlichkeiten zu stabilisieren.

Erkennung von Sensorfehlern Wie bereits bei der Einführung der Sensorfehler in Abschnitt 2.3.2 handelt es sich bei der Erkennung von Sensorfehlern um einen nicht trivialen Fall für die Sensordatenverarbeitung. Entsprechend verhält es sich bei der Überprüfung, ob ein Sensorfehler von der Sensordatenverarbeitung erkannt wurde oder nicht. Dies gilt insbesondere da die Reaktion auf einen Sensorfehler von dem untersuchten sensordatenverarbeitenden System und ggf. der, für die Kommunikation verwendeten, Protokolle abhängig ist.
Mögliche Reaktionen die während der Überprüfung der Sensordatenverarbeitung detektiert werden können sind.

- Das Vorhandensein eines Status. Verschiedene Protokolle wie beispielsweise das NMEA0183 Protokoll verfügen an einigen Stellen über einen Status, z.B. in Form

einer Enumeration, dessen Wert während der Überprüfung beobachtet werden kann.

- Das Ausbleiben von Ergebnissen der Sensordatenverarbeitung kann ein Indiz für einen, zumindest teilweise, erkannten Fehler in den Sensormesswerten darstellen. Zur Erkennung eines solchen Verhaltens lässt sich eine *Watchdog* Komponente verwenden, welche in regelmäßigen Abständen überprüft, ob neue Sensormesswerte von der Sensordatenverarbeitung geliefert wurden. Durch die Verwendung einer zeitgesteuerten Aktivierung lässt sich diese Überprüfung unabhängig von etwaigen Updatezeiten der Sensordatenverarbeitung realisieren.

- Ergebnisse der Sensordatenverarbeitung können mit einem Konfidenzlevel angegeben werden, d.h. der Vertrauenswürdigkeit der Ergebnisse. Beim Vorhandensein eines Sensorfehlers kann ggf. davon ausgegangen werden, dass die Vertrauenswürdigkeit der Ergebnisse herabgesetzt wird, was wiederum mittels einer einfachen Vergleichskomponente und einem entsprechenden Grenzwert, erkannt werden kann.

4.3.3.3 Messungenauigkeiten

Die Messungenauigkeiten bzw. messtechnischen Fehlermodelle, die in Abschnitt 2.3.1 vorgestellt wurden, werden im Rahmen der strukturellen Beschreibung der Fehlermodelle, durch die beiden Klassen *SystematicError* und *StatisticError* abgebildet.

Beide Fehlermodelle verändern die Messwerte innerhalb einer Sensorbeobachtung. Gemäß der Gleichung 2.1, wird der im Fehlermodell angegebene oder ermittelte Wert auf die aktuellen Werte der Beobachtung addiert.

Systematischer Fehler

$$X_{out} = X_{in} + C$$

Zufälliger Fehler

$$X_{out} = X_{in} + RNG_{D(a,b)}$$

Der Fehlerwert kann im Falle des systematischen Fehlers durch einen konstanten Faktor angegeben werden, um den die Sensormessung "verschoben" wird.
Der zufällige Fehler (*StatisticError*) wird durch eine Wahrscheinlichkeitsverteilung ($RNG_{D(a,b)}$) mit den Parametern a und b beschrieben. Zur Auswahl stehen verschiedene Wahrscheinlichkeitsverteilungen, darunter (vgl. auch Abbildung 4.19):

- Normalverteilung:
$$f(x) = \frac{1}{b\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2b^2}}$$
- Exponentialverteilung:
$$f(x) = \frac{1}{a} e^{\frac{-x}{a}}$$
- Gammaverteilung:
$$f(x) = \frac{1}{b^a \Gamma(a)} x^{a-1} \exp\left(-\frac{x}{b}\right)$$
- Diskrete Gleichverteilung:
$$f(x) = \begin{cases} 0, \text{ if } x < a, \\ \frac{1}{b-a}, \text{ if } x \in [a, b], \\ 0, \text{ if } x > b. \end{cases}$$

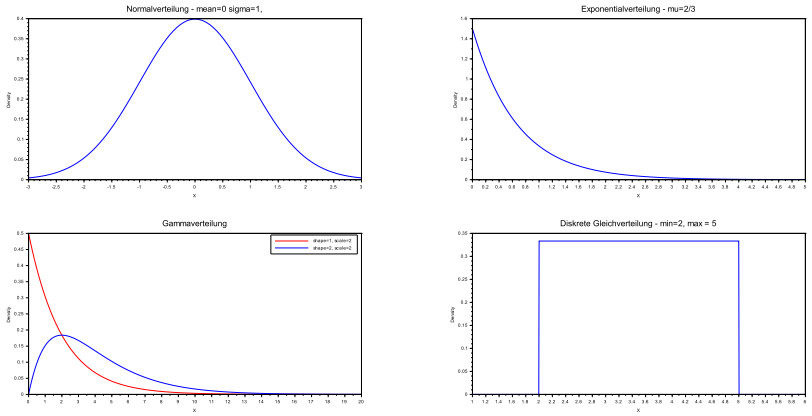

Abbildung 4.19: Auswahl an Wahrscheinlichkeitsdichtefunktion zur Konfiguration des zufälligen Fehlers

Die zufälligen und Systematischen Fehler lassen sich dabei nur auf einfache Zahlenwerte anwenden. Um die Fehlermodelle auch auf mehrdimensionale Messwerte wie beispielsweise Vektoren, Positionen, Laserscanner Messungen oder Bilder anwenden zu können, wird das Konzept der Fehlerkonfiguration eingeführt.
Gleichzeitig wird mithilfe der Fehlerkonfiguration dem Umstand Rechnung getragen, dass sich die Fehlermodelle u.U. nicht auf alle Aspekte einer gegebenen Beobachtung auswirken, wenn z.B. Meta Daten hinterlegt werden, wie beispielsweise eine eindeutige ID des Sensors.

Erkennung von Messungenauigkeiten Um die Reaktion der Sensordatenverarbeitung auf Messungenauigkeiten überprüfen zu können, können Distanzkomponenten verwendet werden, die die Distanz zwischen zwei Werten bestimmen. Dies ist trivial für

den Fall, dass es sich um skalare Messgrößen handelt, wie beispielsweise eine Temperatur oder eine Entfernung.
Im Fall von Positionen oder Winkeln müssen ggf. spezialisierte Komponenten verwendet werden, welche die speziellen Eigenarten der Messgröße berücksichtigen. Sollen beispielsweise zwei GPS Koordinaten gegeneinander verglichen werden, eignet sich eine Distanzberechnung mithilfe der Formel von Vincenty [noa01], welche die Distanz zweier Punkte auf der Oberfläche einer Kugel berechnet.
Bei der Distanzberechnung von Winkeln muss dagegen der zyklische Charakter von Winkeln betrachtet werden, d.h. $distanz(360°, 1°) < distanz(10°, 1°)$.

4.3.3.4 Kontextabhängige Fehler

Die dritte Gruppe der Fehlermodelle, die kontextsensitiven Fehler, stellen sowohl für die Sensordatenverarbeitung als auch für die Erzeugung der Fehler eine Herausforderung dar.
In der betrachteten Literatur wurden sie mit wenigen Ausnahmen nicht weiter behandelt. Zu den Ausnahmen gehört beispielsweise die in Abschnitt 3.3.3.1 vorgestellte VANE (Virtual Autonomous Navigation Environment) Simulation [RCT+09, GDG+10], die u.a. einen Mixed Pixel Fehler für LIDAR-Sensoren berücksichtigt. Dies geschieht allerdings durch die Nachbildung bzw. Annäherung des physikalischen Messprinzipes des LIDAR-Sensors und weniger als nachträgliche Anwendung eines Fehlermodells.

Mithilfe des vorgestellten Simulationsmodells lassen sich kontextsensitive Fehler auf zwei Arten modellieren, die je nach gewünschtem Ergebnis und verfügbaren Informationen ausgewählt werden können.
Dabei handelt es sich zum einen um eine dynamische Anpassung der Konfiguration herkömmlicher Fehlermodelle oder um eine bedingte Verkettung von Fehlerkomponenten. Beide Varianten sind in Abbildung 4.20 beispielhaft dargestellt.

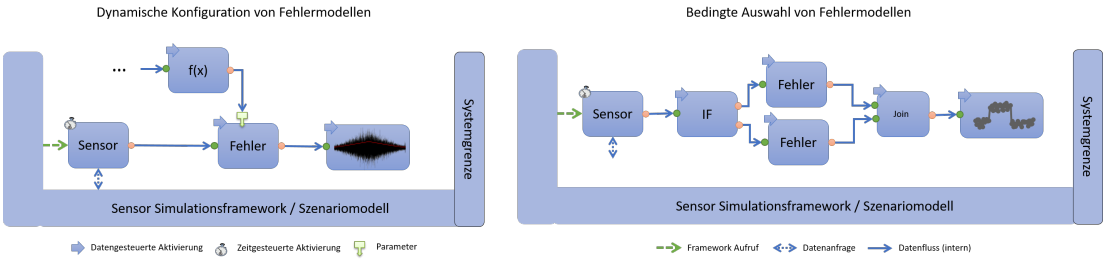

Abbildung 4.20: Modellierungsmöglichkeiten von kontextsensitiven Fehlern durch das Simulationsmodell

Die Umsetzung der bedingten Auswahl von Fehlermodellen zur Modellierung von kontextsensitiven Fehlern stellt im Wesentlichen eine direkte Anwendung des datenflussorientierten Verarbeitungsgraphen dar, wie er auch in anderen Kontexten (z.B. Sensordatenverarbeitung) verwendet wird.

Die dynamische Konfiguration der Fehlermodelle wird erst durch das in Abschnitt 4.3.2.2 beschriebene Komponentenmodell und dessen Verknüpfung mit dem Daten Metamodell ermöglicht. Da es sich bei den Komponenten um eine Erweiterung des Daten-Metamodells handelt, können die Parameter einer Komponente, ähnlich wie die Bestandteile des Szenariomodells, direkt durch den Graphen adressiert und entsprechend geändert werden.
Die Funktion $f(x)$ in Abbildung 4.20 stellt dabei eine beliebige Funktion dar, die ggf. mithilfe von zusätzlichen Hilfskomponenten errechnet wurde. Alternativ kann auch direkt auf Elemente des Szenariomodells zurückgegriffen werden, wenn der entsprechende Faktor zur Verfügung steht.

Erkennung von kontextsensitiven Fehlern Für die Erkennung von kontextsensitiven Fehlern, sind keine besonderen Komponenten während der Bewertung vorgesehen. Stattdessen kommen die gleichen Bewertungskomponenten zum Einsatz, wie sie auch für das Erkennen und Bewerten von Sensorfehlern bzw. Messungenauigkeiten verwendet werden. Erwartungsgemäß können die gleichen Techniken angewendet werden, die zum Erkennen der Fehler verwendet werden, aus denen der kontextsensitive Fehler zusammengesetzt ist.

4.3.3.5 Abhängige und Unabhängige Sensorungenauigkeiten

Eine weitere bei der Simulation von Sensorfehlern zu beachtende Eigenschaft ist, dass Sensorfehler nicht immer stochastisch voneinander unabhängig sind. Dieses Phänomen ist in der Theorie bereits durch die kontextsensitiven Fehlermodelle abgegolten, bei denen ggf. die gleichen Fehlermodelle auf die entsprechenden Sensorbeobachtungen angewendet werden, bzw. bei denen der gleiche oder zu mindestens ein ähnlicher Kontext bestimmt wurde. Auf der anderen Seite ist es verhältnissmäßig aufwendig, die Sensorfehler komplett zu modellieren, so dass kontextabhängige Fehler ggf. als statistische Fehler modelliert werden.
Ein Beispiel für eine solche Abhängigkeit von Messungenauigkeiten von zwei zunächst unabhängigen Sensoren stellen nahe beieinander positionierte GPS Sensoren dar. Es kann davon ausgegangen werden, dass sich die verschiedenen kontextabhängigen Fehler gleichermaßen auf beide Sensoren auswirken.
Um eine statistische Abhängigkeit von zwei Sensoren modellieren zu können, wird eine

besondere Form des Fehlermodells eingeführt, das auf beide Sensormessungen den gleichen Fehler anwendet. Dies geschieht indem zunächst das spezifizierte Fehlermodell auf die erste der beiden eingehenden Sensormessungen angewendet wird. Die dabei verwendeten Änderungen an den Sensormesswerten werden für die zweite Sensorbeobachtung zwischengespeichert und ohne weitere Änderung bei dieser angewendet. Dies führt dazu, dass beide Sensoren den gleichen kontextsensitiven Fehlern unterliegen, ohne dass diese explizit modelliert worden sind. In der Regel wird dieses Fehlermodell mit einem weiteren Fehlermodell, z.B. einem geringen weißen Rauschen, kombiniert.

Diese spezielle Fehlerkomponente kann allerdings nur bei einer starken Ähnlichkeit der simulierten Sensoren bzw. Sensorbeobachtungen angewendet werden. Für heterogene Sensoren deren Fehlermodelle zwar korrelieren sich aber dennoch unterscheiden, müssen die Fehlermodelle einzeln modelliert und ggf. über den gleichen Sensorkontext, konfiguriert werden.

4.3.3.6 Beispiel: Anwendung von Messungenauigkeiten auf unterschiedliche Beobachtungsklassen

Im Abschnitt 4.3.3.3 wurde das Konzept der Fehlerkonfiguration eingeführt um die Anwendung von Messunsicherheiten auf verschiedene Klassen von Sensorbeobachtungen zu ermöglichen. Dieser Abschnitt demonstriert die Anwendung dieser Fehlerkonfigurationen für zwei unterschiedliche Beobachtungsklassen, wie sie u.a. in der maritimen Domäne vorzufinden sind.
Dabei handelt es sich um:

Skalare Messwerte Die Sensormessung besteht, abgesehen von eventuell vorhandenen Meta Daten wie beispielsweise dem Namen des Sensors oder der gemessenen Einheit (Meter, Temperatur, ...), aus einem einzelnen Zahlenwert. In diese Kategorie von Sensoren fallen zum Beispiel:

- Temperatur Sensoren
- Lichtschranken, Ultraschall und Echolot
- Anemometer bzw. Windmesser
- Geschwindigkeits- und Beschleunigungsmesser

Mehrdimensionale Messwerte Für die Beschreibung dieser Sensormesswerte werden komplexe Datenstrukturen verwendet, die ggf. über mehrere Dimensionen verteilt sind. Beispiele für Sensoren welche diese Kategorie von Messwerten liefern sind:

- Kameras (2D und 3D)

- Radar Sensoren
- Laserscanner
- Positionssensoren (z.B. GPS)

Dabei konzentriert sich die Folgende Betrachtung auf die Fehlerkategorien: Systematische und Stochastische Messunsicherheiten, sowie Kontextsensitive Fehler. Die verschiedenen Fehlerzustände, die bereits in den Abschnitten 4.3.3.2 beschrieben wurden, werden an dieser Stelle nicht weiter betrachtet, da sie Unabhängig von dem Format der Messwerte sind.

Auswirkung von Messungenauigkeiten auf skalare Beobachtungen Bei den skalaren Messwerten handelt es sich in Bezug auf das anwenden von Messungenauigkeiten, um die einfachste Kategorie.
Für die folgende Betrachtung der Auswirkungen der verschiedenen Fehlermodelle wird ein idealisierter Messverlauf angenommen, wie er in Abbildung 4.21 (links oben) dargestellt ist. Dabei werden auf der Abszisse die aufeinander folgenden Messungen aufgetragen und auf der Ordinate der entsprechende Messwert. Die weiteren Teile der Abbildung 4.21 stellen die verschiedenen Auswirkungen der Fehleranwendungen dar.

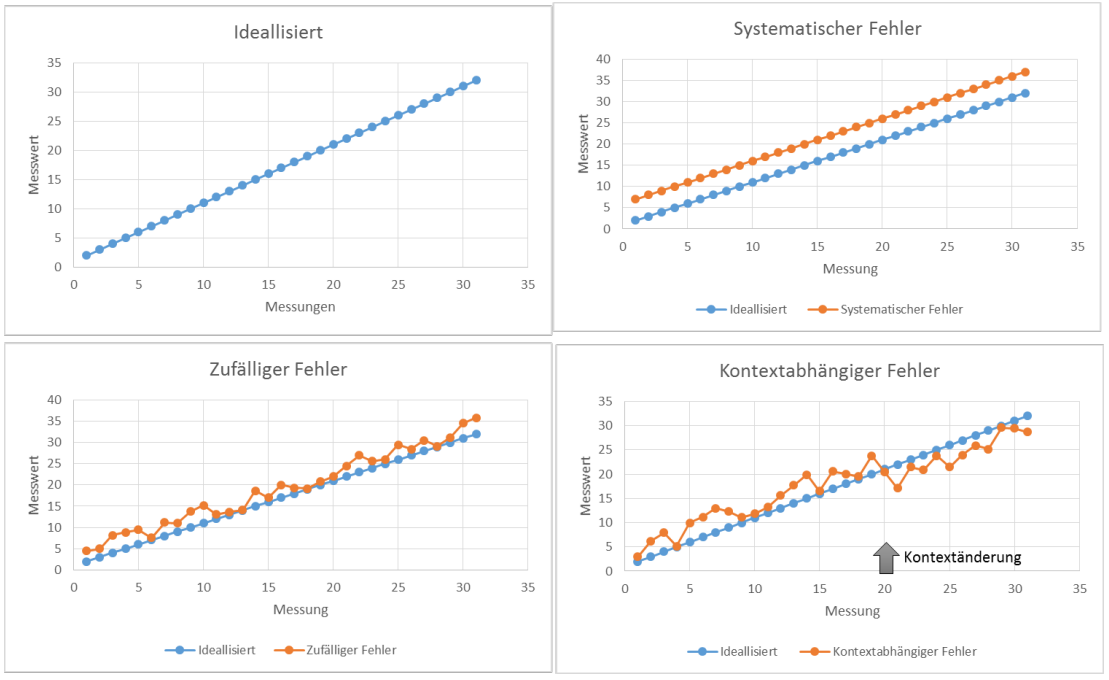

Abbildung 4.21: Auswirkung der verschiedenen Fehlermodelle auf skalare Sensorbeobachtungen

Der obere rechte Teil der Abbildung stellt die Anwendung eines systematischen Fehlers vor, bei dem der Messwert jeder Messung, um genau fünf Einheiten verschoben wurde.

Dabei könnte es sich beispielsweise um einen falsch kalibrierten Windsensor handeln, der immer eine zu hohe Windgeschwindigkeit angibt.
Bei dem Zufälligen Fehler in Abbildung 4.21 (links unten) wurde eine gleichverteilter Zufallswert (zwischen 0 und 5) auf die Messung addiert. Ähnliches gilt für den kontextabhängigen Fehler, wobei sich in dem dargestellten Beispiel bei der 20ten Messung ein Kontextwechsel ergeben hat. Ein solcher Kontextwechsel kann sich beispielsweise ergeben, wenn sich ein großes Objekt vor den zuvor beschriebenen Windsensor befindet und diesen verdeckt. In diesem Fall wird die Messungenauigkeit durch eine gleichverteilte Zufallszahl zwischen 0 und -5 ausgedrückt.

Auswirkung von Messungenauigkeiten bei mehrdimensionalen Beobachtungen Die Gruppe der mehrdimensionalen Beobachten lässt sich in weitere Teilgruppen unterteilen. Im Zusammenhang von Umfelderkennenden Sensoren lassen sich u.a. die folgenden Sensorbeobachtungen vorfinden:

- Fächerscans
- Bilder
- Position

Fächerscans Unter einem Fächerscan wird ein Muster verstanden, wie es in Abbildung 4.22 dargestellt ist. Dabei handelt es sich häufig um Entfernungsmessungen, wie sie von einem Laserscanner aufgenommen werden. Aber auch die einzelnen Messungen, die von einem Radargerät aufgenommen werden, lassen sich in Form eines sich aufspannenden Fächers darstellen, bevor sie zu einem Bild zusammengefasst werden. Sie bestehen i.d.R. aus einer Reihe von Winkel und Entfernungsangaben, wobei es sich bei der Entfernung um den eigentlich gemessenen Wert handelt, wohingegen der Winkel als zusätzliche Information angegeben wird.
Abbildung 4.22 zeigt beispielhaft die Auswirkungen der verschiedenen Fehlermodelle,

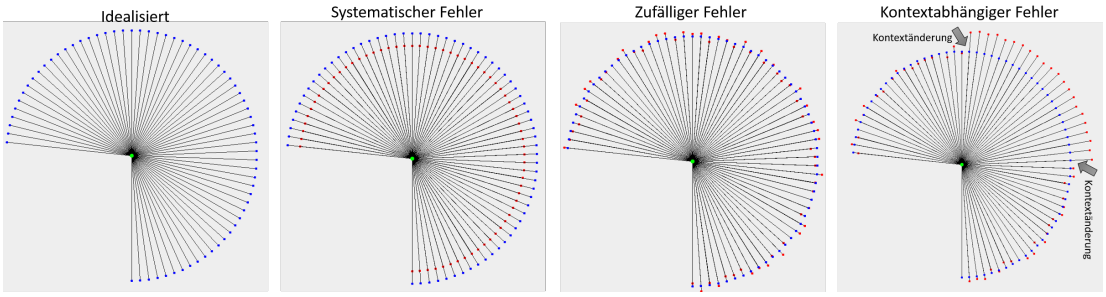

Abbildung 4.22: Darstellung der Fehlerauswirkungen auf Fächerscans.

systematischer Fehler, zufälliger Fehler und kontextsensitiver Fehler auf eine Laserscanner Messung. Die Position des Sensors ist dabei durch den grünen Punkt in der Mitte der jeweiligen Bilder gekennzeichnet, die blauen Punkte stellen die idealisierten Messergebnisse dar.

Bei dem systematischen Fehler wird für jeden Entfernungswert ein konstanter Wert abgezogen. Dies könnte beispielsweise auf eine falsche Kalibrierung des internen Zeitmessers zurückzuführen sein. Der kontextabhängige Fehler in Abbildung 4.22 könnte beispielsweise auf eine Glasfläche innerhalb des markierten Bereiches zurückzuführen sein, bei dessen Durchgang der Laserstrahl gebrochen wird, wodurch sich ein längerer Strahlengang ergibt.

Bildmessungen Messwerte, die sich als Bild ausdrücken lassen, gehören zu den anspruchsvollsten Sensormessungen, sowohl was ihre Verarbeitung angeht, als auch deren Verrauschung. Zu den Sensoren die diese Form von Messwerten erzeugen gehören hauptsächlich Kamerasysteme, wie klassische Kameras. Aber auch Infrarotkameras deren Messwerte als eine räumliche Verteilung von Temperaturen angesehen werden können oder Radarsensoren, deren rundum Messungen zu einem Bild zusammengesetzt werden. Abbildung 4.23 stellt ebenfalls Beispielhaft die verschiedene Messungenauigkeiten für bildbasierte Sensormessungen dar. In dem linken Teil der Abbildung 4.23 ist die Anwen-

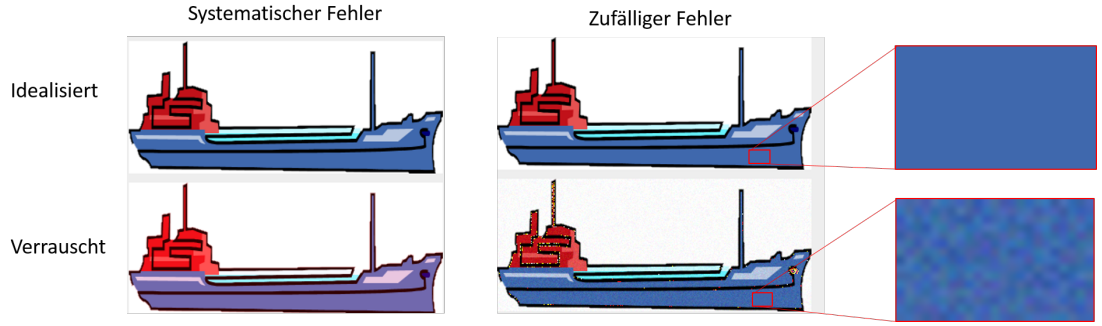

Abbildung 4.23: Anwendung von systematischem und zufälligen Fehler auf Bilder

dung eines systematischen Fehlers zu sehen. Dabei handelt es sich um eine Rotverschiebung des gesamten Bildes, beim dem der Rot Wert jedes Pixels erhöht worden ist[6]. Durch einen solchen Fehler lässt sich beispielsweise ein fehlerhafter Weißabgleich reproduzieren oder eine falsche Ansteuerung des zugrunde liegenden Bildsensors. Unter Umständen kann eine solche Farbverschiebung auch durch unvorhergesehene Beleuchtungen hervorgerufen werden, bzw. bei Aufnahmen in großen Wassertiefen, wie es beispielsweise in [BMB+15] beschrieben wird.

[6] Anschließend wurde der neue Wert auf den Bereich von 0-255 beschränkt, um einen Byte-Überlauf zu verhindern.

Der zweite Teil der Abbildung 4.23 zeigt ein zufälliges Rauschen auf den Farbwerten des Originalbildes. Ein solches Rauschen lässt sich insbesondere bei qualitativ minderwertigen Kameras oder bei schlechten Beleuchtungsverhältnissen ausmachen.
Komplexere Fehler wie beispielsweise Verzerrungen, ausgelöst durch eine fehlerhafte Rektifikation oder Lichteffekte wie Blendung und partielle Überbeleuchtungen fallen unter den Bereich der kontextsensitiven Fehlermodelle und müssen ggf. für Bilddaten gesondert behandelt werden.

Positionsmessungen Positionsmessungen stellen eine weitere Kategorie von Sensormessungen dar. Die Messwerte werden dabei in der Regel als zwei oder dreidimensionale Tupel dargestellt, wobei ihre Interpretation abhängig vom gewählten Koordinatenreferenzsystem ist. So können beispielsweise die Komponenten des Tupels als Winkelangaben in einem Geodätischen Koordinatensystem wie WGS84 angegeben werden, oder als Entfernungen von einem Koordinatenursprung. Das verwendete Koordinatenreferenzsystem gibt dabei insbesondere die Größe der anzuwendenden Fehlerwerte an.
Ein Beispiel für die Auswirkungen von systematischen und stochastischen Messungenauigkeiten wurde bereits in Abschnitt 2.3.1 (Messtechnische Fehler) bzw. in Abbildung 2.5 angegeben.

4.3.4 Kommunikation der Beobachtungen

Bevor eine Bewertung der Sensordatenverarbeitung vorgenommen werden kann, müssen zunächst die generierten Sensormesswerte an die Sensordatenverarbeitung weitergeleitet werden, sowie das von ihr ermittelte Umgebungsmodell empfangen werden können.
Zu diesem Zweck kommen zwei spezialisierte Komponenten (eine Senke und eine Quelle) zum Einsatz, welche hinsichtlich der verwendeten Protokolle und Transportmedien erweitert werden können, um die in Abschnitt 3.4.1.1 formulierte Anforderung nach der Verwendung existierenden Schnittstellen erfüllen zu können (vgl. Anforderung A7).

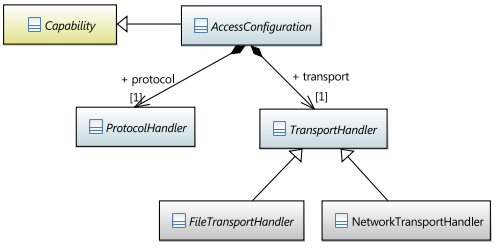

Abbildung 4.24: Vereinfachte UML Darstellung des Inter-Prozess-Kommunikation Teilmodells; gelb dargestellt sind Klassen aus anderen Teilmodellen, grau hinterlegte Klassen können als Beispielhafte Konkretisierung verstanden werden

Anschließend besteht dieser Teilprozess aus der Umwandlung der Sensormesswerte aus einem internen Datenformat in ein Datenformat, das von der Sensordatenverarbeitung

verstanden werden kann, wobei beachtet werden muss, dass an dieser Stelle ggf. Informationen die im Simulationssystem vorhanden sind, verloren gehen können.

Die Konfiguration der entsprechenden Senke bzw. Quelle erfolgt mit dem in Abbildung 4.24 dargestelltem Modell und setzt sich zusammen aus einer Übertragungsmethode (*TransportHandler*) sowie einer Beschreibung für das verwendete Protokoll (*ProtocolHandler*). Gleichzeitig stellt das in Abbildung 4.24 dargestellte UML Modell die letzte verbleibende Erweiterung der strukturellen Beschreibung von Sensoren aus Abbildung 4.10 dar.

Das Verhalten der Senke ist schematisch in Abbildung 4.25 dargestellt, wobei sich die Quelle Analog dazu verhält. Der *ProtocolHandler* ist dabei für die Umwandlung aus dem internen Datenformat in die protokollspezifische Darstellungsform zuständig. Für das Simulationssystem handelt es sich dabei um ein binäres Datum, dessen genaue Bedeutung nicht mehr bekannt ist. Die binären Daten werden anschließend von dem *TransportHandler* über eine geeignete Schnittstelle an die externe Sensordatenverarbeitung weitergeleitet. I.d.R. handelt es sich dabei um eine netzwerkbasierte Kommunikation.

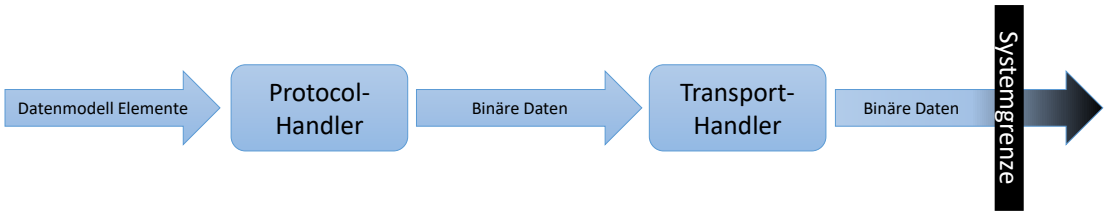

Abbildung 4.25: Datenfluss bei der Kommunikation mit externen Systemen

4.3.5 Modellierung der Qualitätskriterien

Die Modellierung der Qualitätskriterien entspricht im Wesentlichen dem Schritt der Bewertung des sensordatenverarbeitenden Systems, bzw. dem zu testenden System.

An dieser Stelle kommen hauptsächlich Vergleichskomponenten zum Einsatz, wie sie bereits bei den Beschreibungen der verschiedenen Sensorfehler vorgestellt wurden.
Dazu gehören die einfachen Vergleichsoperatoren, sowie spezielle Operatoren zur Ermittlung von Distanzen zwischen Koordinaten, Vektoren oder Winkeln.
Gleichzeitig werden Operatoren bzw. Komponenten benötigt, mit denen das Vorhandensein von Enumerations oder Flags überprüft werden kann, für den Fall, dass die Sensordatenverarbeitung auf diese Weise über eine Störung informiert.

Mithilfe der Vergleichsoperatoren sowie entsprechenden, von einem Experten festgelegten Schwellwerten, lassen sich die Qualitätskriterien SDV2 (Fehlertoleranz bezüglich Sensorfehlern) sowie SDV3 (Robustheit gegenüber Messunsicherheiten) quantifizieren.

Die Messung der Antwortzeiten der Sensordatenverarbeitung kann nicht direkt durchgeführt werden, wenn nicht gegen die Anforderung A7 (Verwendung realer Schnittstellen) verstoßen werden soll. Dies liegt darin begründet, dass nicht jedes Kommunikationsprotokoll über einen entsprechenden Zeitverlauf verfügt. Wie auch schon zuvor kann das NMEA 0183 Protokoll als ein Beispiel für ein solches Protokoll angesehen werden.

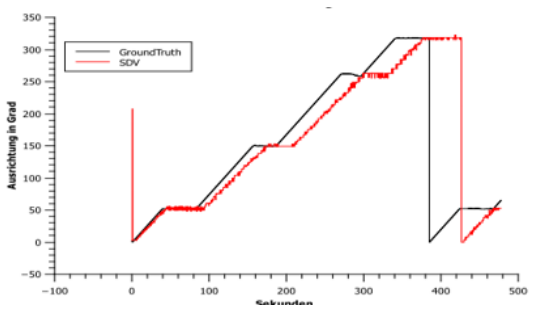

Abbildung 4.26: Graphische Darstellung zur Analyse von Antwortzeiten am Beispiel der Ausrichtung eines Schiffes

Um dennoch Aussagen über die Antwortzeiten (vgl. SDV4) treffen zu können, werden zwei Ansätze vorgesehen. Der erste Ansatz verwendet eine geeignete Ergebnispräsentation, bei dem die ermittelten Messwerte sowie die erwarteten Werte gegeneinander aufgetragen werden, wie es beispielsweise in Abbildung 4.26 dargestellt ist.
Diese Form der Ergebnispräsentation eignet sich gut, um auf einen Blick die Antwortzeit abschätzen zu können, ist aber nur bedingt gut für die Anwendung in einer automatischen Analyse geeignet.
Bei dem zweiten Ansatz handelt es sich um die Suche nach der größten Übereinstimmung des gemessenen Wertes in einem Buffer, der den Erwartungswert, zusammen mit dem Zeitpunkt seiner Gültigkeit, speichert (vgl. Abschnitt 4.3.5.1).
Dabei ist zu beachten, dass dieser Ansatz nur dann eingesetzt werden kann, wenn durch die größte Übereinstimmung des Erwartungswertes und des gemessenen Wertes, d.h. die kleinste Distanz der beiden Werte, auch auf die Korrelation der beiden Werte geschlossen werden kann. Dies ist nicht der Fall, wenn der ermittelte Wert, z.B. durch Messunsicherheiten, stark von dem Erwartungswert abweicht.

Ergebnispräsentation Neben der Ermittlung der Qualitätsmerkmale des sensordatenverarbeitenden Systems kann innerhalb dieses Schrittes die Form der Ergebnispräsentation modelliert werden. Zu diesem Zweck sieht die Methode verschiedene Senken vor, wie beispielsweise die Ausgabe von Log-Nachrichten, das Erstellen von tabellarischen Ergebnisdokumenten oder eine grafische Ausgabe, wie in Abbildung 4.26 dargestellt.

4.3.5.1 Statische vs dynamische Umgebung

Durch die Einführung einer eigenen Simulationszeit, durch die zeitgesteuerte Aktivierungsstrategie in Abschnitt 4.3.2.2, ergeben sich ein weiter Vorteil für die simulative

Bewertung von sensordatenverarbeitenden Systemen.
Dabei handelt es sich um die Möglichkeit, die Zeit anzuhalten während die Sensordatenverarbeitung die Sensormesswerte zu einem Umgebungsmodell verarbeitet.
In diesem Fall kann das Ergebnis des sensordatenverarbeitenden Systems direkt mit dem aktuellen Zustand der simulierten Umwelt verglichen bzw. bewertet werden. Eine Voraussetzung hierfür ist eine synchrone Kommunikation zwischen der Sensorsimulation und der Sensordatenverarbeitung, welche gemäß der Anforderung A7 durch eine entsprechende Schnittstelle unterstützt werden muss.

In der Mehrzahl der Fälle wird allerdings von einer asynchronen Kommunikation zwischen der Sensorsimulation, der Sensordatenverarbeitung und ggf. der Anwendungs- bzw. Bewertungslogik ausgegangen. Unter anderem bedingt durch die Tatsache, dass ein Sensor nicht über das mit seiner Hilfe erschaffene Umgebungsmodell informiert wird.
Bei einer realitätsnahen Nachbildung der Umwelt muss zudem davon ausgegangen werden, dass sie sich in der Zeit, die die Sensordatenverarbeitung für die Erstellung des Umgebungsmodells benötigt, verändert und damit das ermittelte Umgebungsmodell veraltet ist (vgl. Anmerkung 4.4.1).

Um dennoch eine Bewertung des erkannten Umgebungsmodells vornehmen zu können müssen die relevanten Ausprägungen der Objekte zu einem Zeitpunkt zwischengespeichert werden.
Zu diesem Zweck werden Buffer - Komponenten eingeführt, welche die Zeit-Wert Paare in einem speziellen Bereich des gemeinsamen Speichers (vgl. Abschnitt 4.3.2.2) abspeichern bzw. lesen können. Bei einer Anfrage, liefern die lesenden Komponenten das Objekt, dessen zeitliche Differenz gegenüber der Anfrage am geringsten ist.

4.4 Simulative Bewertung

Nachdem im Abschnitt 4.2.1 zunächst das Szenariomodell sowie in Abschnitt 4.3.2 das Simulationsmodell zur Beschreibung des Verhaltens und Fehlverhaltens der Sensoren vorgestellt wurde, beschäftigt sich dieser Abschnitt mit der Ausführung des Simulationsmodells durch eine ereignisorientierte Simulation.
Das Ergebnis dieses Schrittes sind simulierte Sensormesswerte, die durch eine angeschlossene Sensordatenverarbeitung interpretiert werden können, sowie optional Bewertungsergebnisse der Sensordatenverarbeitung, sofern diese bei der Modellierung berücksichtigt wurden.

Die eigentliche Ausführung erfolgt dann in zwei Schritten einer internen Ausführung mithilfe einer ereignisorientierten Simulation (vgl. [LCRP+05]), wobei es sich bei den Ereignissen im Wesentlichen um die Zeitpunkte der Sensoraktivierungen handelt. Zum anderen kommt eine externe Ausführung hinzu, sofern die Sensorsimulation in eine Ko-Simulation integriert wird.
Bei der Ausführung der Simulation muss zudem darauf geachtet werden, das die Anforderung nach einem korrekten Zeitverhalten (vgl. Anforderung A4) eingehalten wird. Zu diesem Zweck wird im Folgenden ein Controller vorgestellt, der die korrekte Einhaltung der Sensor Updateraten sicherstellen kann.

4.4.1 Interne Ausführung der Sensorsimulation

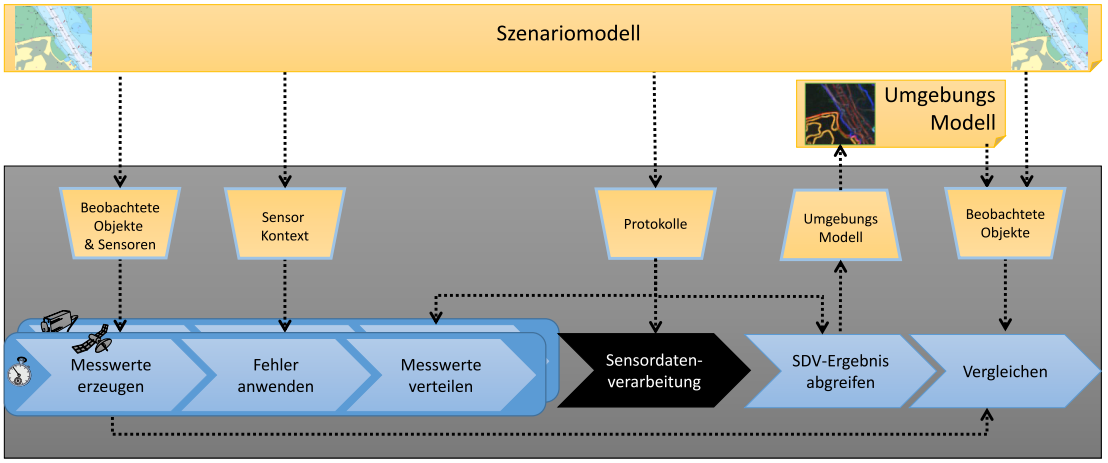

Abbildung 4.27: Zeitlicher Ablauf eines Sensorsimulationsschrittes

Die interne Ausführung der Simulation folgt dem in Abbildung 4.27 dargestelltem Schema aus Sensormesswertsynthese, Fehleranwendung und Kommunikation der Sensormesswerte an die Sensordatenverarbeitung. Dabei wird der erste Schritt, d.h. die Messwerterzeugung, durch einen *Scheduler* initiiert, wohingegen die weiteren Schritte, die Fehleranwendung und Kommunikation durch den datengetriebenen Verarbeitungsgraphen übernommen werden, wie es in Abschnitt 4.3.2.2 beschrieben wurde.

Die nachfolgenden Schritte, d.h. das Abgreifen der Ergebnisse der Sensordatenverarbeitung und die eigentliche Bewertung, werden abhängig von der Konfiguration der zum Abgreifen der Ergebnisse verwendeten Quelle, wahlweise ebenfalls durch den *Scheduler* oder durch eine Datenweitergabe innerhalb des Verarbeitungsgraphen gesteuert.

4.4.1.1 Ereignisorientierten Simulation

Im Komponenten Metamodell (vgl. Abschnitt 4.3.2.2) wurde eine zeitgesteuerte Aktivierungsstrategie eingeführt, welche für die Aktivierung von Sensorberechnungen verwendet wird.
Für die Umsetzung dieser zeitgesteuerten Aktivierung werden Techniken aus dem Bereich der ereignisorientierten Simuulation verwendet, wie sie in ähnlicher Form auch in anderen Simulationssystemen, wie beispielsweise dem von T. Hoerstebrock entwickelten JSON [Hoe14], dem Simulationssystem MASON [LCRP$^+$05] oder der bereits erwähnten High Level Architecture [ISISC00] verwendet werden.

Insbesondere das Konzept des *Schedulers* zur Ablauf- und Zeitsteuerung hat sich für diese Aufgabe als geeignet herausgestellt. Dabei verwaltet der Scheduler als eine zentrale Simulationskomponente eine eigene Simulationszeit die je nach Bedarf schneller oder langsamer als die reale Zeit ablaufen kann.
Die Verwendung einer ereignisorientierten Simulation trägt damit unmittelbar zur Erfüllung der Anforderung A4 bei in der gefordert wird, dass die Simulation für automatisierte Analysen schneller als Realzeit, bzw. für manuelle Analysen auch langsamer als in Realzeit ausgeführt werden kann.

Die Tatsache, dass bei ereignisorientierten Simulationen die Zeit zwischen zwei diskreten Zeitpunkten übersprungen werden kann und nicht wie bei der zeitdiskreten Simulation auch behandelt werden muss, wenn keine Ereignisse in den entsprechenden Zeitraum fallen, trägt zudem zu einer höheren Effizienz bei. Abbildung 4.28 zeigt die Zeitverwaltung eines diskreten Event-Schedulers. Einzelne Tätigkeiten / Events ($E_1 - E_5$) können für einen bestimmten Zeitpunkt registriert werden. Es wird dabei explizit erlaubt, dass sich mehrere Events für den gleichen Zeitpunkt registrieren (z.B.: t_1 in Abbildung 4.28). Weiterhin kann es sich bei den Tätigkeiten entweder um einmalige Tätigkeiten handeln,

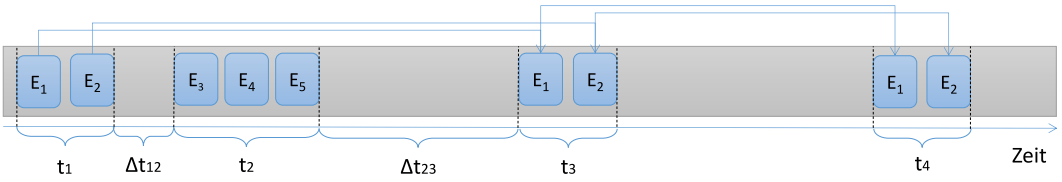

ABBILDUNG 4.28: Funktionsweise eines Diskret Event Schedulers, Einzelne Events / Tätigkeiten werden für frei zu definierende Zeitpunkte eingeplant und werden in der richtigen Reihenfolge abgearbeitet. Dabei kann es sich bei einem Event sowohl um ein einmaliges Event handeln (E_3, E_4, E_5) oder eine wiederkehrendes Event (E_1, E_2). Die Zeit zwischen den Events ($\Delta t_{12}, \Delta t_{23}$) werden ignoriert.

wie es in Abbildung 4.28 durch die Tätigkeiten E_3, E_4 und E_5 angedeutet ist, oder um wiederholt ausgeführte Tätigkeiten (E_1 und E_2) die in regelmäßigen Abständen wiederholt werden.

Bei dieser Form des Scheduling wird davon ausgegangen, dass eine Tätigkeit innerhalb eines Zeitschrittes abgeschlossen werden kann (vgl. dazu auch Anmerkung 4.4.1 -Zeitdiskretisierung). Der Zeitraum zwischen zwei Events muss bei der ereignisorientierten Simulation nicht weiter betrachtet werden und kann unterschiedliche Längen überbrücken, so dass für die Zwischenräume Δt_{12} und Δt_{23} in Abbildung 4.28 gilt: $\Delta t_{12} \neq \Delta t_{23}$. Dementsprechend fallen an dieser Stelle keine weiteren Kosten z.B. für die Überprüfung, ob ein Ereignis eingetreten ist oder nicht an.

Anmerkung 4.4.1 (*Zeitdiskretisierung*)

Die Verwendung einer zeitdiskreten Simulation wird unter der Annahme durchgeführt, dass der Messvorgang eines Sensors keine Zeit beansprucht. Diese Annahme ist in der Realität nicht erfüllbar, die Abweichung kann aber bei den meisten Sensoren vernachlässigt werden. Bei der Messung der Reflexion eines Laserstrahls beispielsweise breitet sich das Licht mit einer Geschwindigkeit von $c = 299792458\frac{m}{s}$ *aus. Bei einer Reichweite von 80 Metern, bei einem Laserscanner führt dies zu einer Messzeit (für einen Laserstrahl) von* $\approx 2.668E^{-7}$ *Sekunden.*

Auch bei einer Relativgeschwindigkeit, des gemessenen Objektes, von $100\frac{km}{h} \approx 27.7\frac{m}{s}$*, würde sich das Objekt innerhalb der Messzeit um ca. 0.00000739178m bewegen, was weit unterhalb der vom Hersteller angegebenen Genauigkeit von ca.* $0.002m \approx 2mm$ *liegt.*

Während diese Beispielrechnung für einen einzelnen Laserstrahl zutrifft, lässt sie sich nicht linear auf die gesamte Messung eines Laserscanners hochrechnen. SICK Laserscanner beispielsweise operieren i.d.R. mit einer Frequenz von bis zu 50 Hz. Unter der Annahme, dass es sich dabei auch um die Messzeit eines Scans handelt, ergibt sich eine Relativbewegung für ein $100\frac{km}{h}$ *schnelles Objekt von 0.5m.*

Zur Lösung dieses Problems, sofern es sich um ein Problem für das spezielle Anwendungsszzenario handelt, bietet das vorgestellte Simulationsframework zwei Lösungsansätze an.

- *Die Simulation der Sensoren kann in kleinere Intervalle zerlegt werden, so kann beispielsweise der komplette Scan eines Laserscanners in 20 oder 30 Teilscans zerlegt werden, bei denen die Relativbewegung keine Relevanz mehr für das Ergebnis hat. Dieses Verfahren hat den Nachteil, dass dies i.d.R. einen zusätzlichen Mehraufwand bei der Berechnung der Sensormesswerte erfordert, was wiederum die Anforderung A4 (Zeitverhalten) beeinflussen kann.*
- *Die Behandlung der resultierenden Messungenauigkeit in einem kontextsensitiven Fehlermodell. Es lassen sich Fehlermodelle erzeugen, die für den Fall der Beispielrechnung die Geschwindigkeit des beobachteten Objektes als Kontext der Sensormessung berücksichtigen und aufbauend auf der Relativ-Bewegung einen entsprechenden Fehler in die Messwerte einbringen, zum Beispiel in Form eines "Motion Blur" Faktors.*

Wall-Clock time Im Rahmen der Überprüfung eines sensordatenverarbeitenden Systems kann es vorkommen, dass die Sensorsimulation unter Realzeitbedingungen laufen muss. Dies kann zum Einen zu einer Abschätzung der Last der Sensordatenverarbeitung verwendet werden oder wird von dieser gefordert, wenn sie nicht in der Lage ist mit einer künstlich beschleunigten Zeit zu arbeiten. In diesem Fall muss der Scheduler so konfiguriert werden können, dass er die Zeit zwischen zwei Events (z.B.: Δt_{12}) wartet. Dies erlaubt auch die Berücksichtigung der Ausführungszeit der einzelnen Simulationsevents. Diese wird vor dem Warten von der zu wartenden Zeit abgezogen.
Mit diesem Wissen lässt sich eine zudem die Anforderung A4, aus Abschnitt 3.4.1, weiter spezifizieren:

WA1 - Effizienz Die Ausführungszeit eines oder mehrerer Simulationsevents soll kürzer sein, als die Zeit zwischen dem Starten des ersten Simulationsevent zum Zeitpunkt t_n und dem Starttermin t_{n+1} des Folgeevents. Formal ausgedrückt, mit der Notation aus Abbildung 4.28 ergibt sich daraus die Anforderung:

$$t_{n+1} - t_n - \sum_{x=1}^{X} E_x > 0 \tag{4.1}$$

Anmerkung 4.4.2

Die in Gleichung 4.1 spezifizierte Anforderung an die Effizienz kann genutzt werden,

um zu überprüfen, ob die das Simulationssystem in der Lage ist, eine realzeitfähige Simulation zu gewährleisten. In dem hier vorgestellten Ansatz handelt es sich dennoch um eine "weiche" Anforderung, dessen Erfüllung weder nachgewiesen werden soll noch wird. Dies liegt in der Tatsache begründet, dass bei dem vorgeschlagenen Framework zu mindestens in der Theorie beliebig viele Events zu einem Zeitpunkt ausgeführt werden können (Theoretisch gilt $X \rightarrow \infty$). Gleichzeitig müsste die maximale Ausführungszeit eines Simulationsevents bekannt sein.

Die Verwendung eines Schedulers zur Ablaufsteuerung hat weiterhin den Vorteil, dass sie ebenfalls zur Erfüllung der Anforderung A1 (Beobachtbarkeit) beitragen kann. Dies wird durch eine Eigenschaft des verwendeten Schedulers ermöglicht, welche die Protokollierung und ggf. Anzeige der durchgeführten Ereignisse erlaubt.

4.4.2 Integration in eine Ko-Simulation

Für die Spezifikation des Verhaltens von Objekten, sieht die Methode eine Einbettung der Sensorsimulation und Bewertung, in eine Ko-Simulationsumgebung vor. Dementsprechend müssen Simulationen identifiziert werden, die das spezifische Verhalten von Objekten simulieren können.
Als Schnittstelle zwischen der Sensorsimulation und den externen Verhaltenssimulationen wird das gemeinsam genutzte Szenariomodell verwendet, wie es in Abbildung 4.29 dargestellt ist.

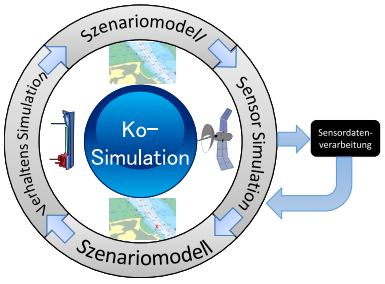

Abbildung 4.29: Synchronisation der Ko-Simulation über das Szenariomodell

Dabei wird das Szenariomodell als ein verteilter gemeinsamer Speicher (vgl. [PTM98]) verwendet, der von den beteiligten Simulationssystemen gelesen und geschrieben werden kann. In diesem Zusammenhang versteht sich die Sensorsimulation als ein Empfänger von Veränderungen, die von den Verhaltenssimulationen oder durch eine manuelle Interaktion des Testers durchgeführt worden sind.
Die Einbettung der Sensorsimulation in eine Ko-Simulationsumgebung ist maßgeblich durch die in Abschnitt 3.4.1.1 formulierte Forderung nach einem nachvollziehbaren Verhalten (vgl. A9) der simulierten Objekte begründet.
Um eine inhaltlich korrekte Simulation zu gewährleisten, müssen zwei Anforderungen von den Simulationen erfüllt werden.

CoS1 - Systemverständnis *Die Simulationen müssen über das gleiche Verständnis des zu simulierenden Systems verfügen.*
Werden in einer Co-Simulation zwei Schiffssimulationen miteinander gekoppelt, die jeweils das Verhalten eines Schiffes simulieren, darf es nicht vorkommen, dass eines der Schiffe aufgrund von fehlendem Tidehub auf Grund läuft, während das andere, größere und tieferliegende Schiff in der gleichen Situation ohne Schwierigkeiten passieren kann.

CoS2 - Interprozesskommunikation *Die Zustände der verschiedenen Simulationen müssen untereinander synchronisiert werden können.*
Änderungen, die von einem Simulator am System vorgenommen werden, müssen den anderen Teilnehmern der gekoppelten Simulation zeitnah zur Verfügung gestellt werden. Wird beispielsweise die Position eines Schiffes verändert, muss diese an die anderen Simulatoren weitergegeben werden, damit diese entsprechend darauf reagieren können, z.B. in dem aktuelle Sensormesswerte erzeugt werden.

Die erste Anforderung (CoS1) kann dadurch unterstützt werden, dass alle an der Ko-Simulation beteiligten Simulatoren und Systeme, das im World Data Model beschriebene Verständnis, über die Systemumgebung teilen (vgl. Abschnitt 4.2.1).
Dieses Konzept zur Sicherstellung der Interoperabilität von Systemen in einer Ko-Simulation wurde unter anderem in der Arbeit “Interoperability In Co-Simulatons Of Maritime Systems.“ [DHS14] veröffentlicht.

Die zweite Anforderung (CoS2) wird durch den Einsatz einer Middlewarelösung sichergestellt.
Diese übernimmt die technische Kommunikation zwischen den beteiligten Systemen, wie beispielsweise den Verbindungsaufbau und das Netzwerktopologiemanagement und den verlässlichen Transfer von Daten, sofern es sich um eine netzwerkbasierte Middleware handelt. Im Rahmen der im Folgenden vorgestellten Realisierung wird an dieser Stelle eine auf dem “High Level Architektur“ (HLA) Standard [ISISC00] basierende Middlewarelösung verwendet, wie sie u.a. in Puch et. al. [PFOL12] und Lässche et. al. [LGH13] beschrieben wird.

4.4.2.1 Externes Scheduling

Bei der HLA Spezifikation [ISISC00] bzw. dessen Implementierungen kommt, genau wie für die interne Ausführung des Simulationsmodells, ein ereignisgesteuerter Scheduler zum Einsatz, bei dem sich die angeschlossenen Simulationen[7] für bestimmte Zeitpunkte

[7] im HLA Kontext auch Federates genannt

registrieren können.
Die HLA Runtime Infrastructure (RTI) stellt dann während der Ausführung sicher, dass zu diesen Zeitpunkten alle vorliegenden Informationen bei den Federates verfügbar sind. Im vorliegenden Kontext bedeutet dies, dass das Szenariomodell aktualisiert worden ist.

In der Kombination von internem und externem Scheduling stehen dem Anwender drei Modi zur Verfügung, die je nach Anforderungen der externen Simulatoren gewählt werden können.

1. Die Sensorsimulation fordert für jeden von ihr betrachteten Zeitpunkt, einen Zeitpunkt bei der HLA RTI an. Dies hat u.U. zur Folge, dass mit einem sehr hohen Kommunikationsaufwand während der Ko-Simulation gerechnet werden muss, abhängig von den verwendeten Sensoren und ihren Updateraten.

2. Es wird eine fixe Updaterate für die Ko-Simulationsumgebung verwendet, während eines solchen Schrittes läuft die Sensorsimulation in Echtzeit bzw. in der von der Sensordatenverarbeitung geforderten Zeit.
   Die Verwendung dieser Methode hat zur Folge, dass die Sensorsimualtion in der Lage versetzt wird, die gesamte Ko-Simualtion auszubremsen, sofern die Updateraten falsch gewählt wurden. Auf der anderen Seite reduziert diese Methode in der Regel den Kommunikationsaufwand gegenüber der Methode 1.

3. Es wird eine fixe Updaterate für die Ko-Simulationsumgebung verwendet. Während eines Zeitschrittes läuft die Sensorsimulation mit maximaler Geschwindigkeit, um weitere Federates nicht auszubremsen. Dabei muss beachtete werden, dass die gewählte Updaterate korrekt gewählt wird, damit die Anforderung nach einem korrekten Zeitverhalten (vgl. Anforderung A4) eingehalten werden kann.

4.5 Zusammenfassung

In diesem Kapitel wurde der eigene Ansatz zur simulativen Bewertung von sensordatenverarbeitenden Systemen vorgestellt, der im Wesentlichen auf der Idee beruht, die reale Umwelt gegen eine reale Umwelt auszutauschen, sowie die simulierte Umwelt als Erwartungswert für die Bewertung heranzuziehen.

Dazu wurde zu Beginn ein Vorgehen, bestehend aus den drei Schritten Systemidentifikation, Modellierung und simulativer Bewertung vorgestellt.
Dabei kann die Systemidentifikation sowohl aus Sicht der Methode, als auch ihrer Anwender, als die Grundlage des eigentlichen Ansatzes gesehen werden. In ihr werden Informationen über das zu testende bzw. das zu bewertende sensordatenverarbeitende

System identifiziert.
Dies betrifft in erster Line die Fehlermodelle der beteiligten Sensoren sowie eine Quantifizierung der Qualitätskriterien die mit der simulativen Bewertung überprüft werden sollen.
Weiterhin wird in diesem Schritt die Einsatzumgebung der Simulation, in Form des sogenannten Szenariomodells definiert, bei dem es sich um eine strukturierte Beschreibung der von den Sensoren beobachteten Umgebung handelt.

Im zweiten Schritt der Methode wurde auf die Modellierung des Sensorverhaltens und insbesondere das Fehlverhalten von Sensoren eingegangen. Zu diesem Zweck wurde der Kern eigene Ansatz vorgestellt, der aus einer flexiblen Verkettung von Simulationskomponenten in einem datengetriebenen Verarbeitungsgraphen besteht, wobei das ursprünglich aus dem Bereich der Signalverarbeitung stammende Konzept des datengetriebenen Verarbeitungsgraphen um Konzepte aus dem Bereich der ereignisorientierten Simulation erweitert wurde.
Durch den Verarbeitungsgraphen kann beschrieben werden, wie mit einfachen Operationen Sensorbeobachtungen aus dem Szenariomodell abgeleitet werden können bzw. vorhandene Beobachtungen in ihrem Detailgrad erhöht werden, indem durch Komponenten auf eine gesuchte Eigenschaft geschlossen werden kann.
Einen weiteren großen Teil nahm die Vorstellung von Komponenten ein, mit denen die erstellten Sensorbeobachtungen hinsichtlich ihres Fehlverhaltens manipuliert werden können. Dazu wurden allgemeine Fehlermodelle vorgestellt, die sich auf alle strukturell beschriebenen Sensorbeobachtungen anwenden lassen. Sowie zwei Möglichkeiten, mithilfe des Verarbeitungsgraphen kontextsensitive Fehler, handhaben zu können. Zu den behandelten Fehlerkomponenten wurde zudem beschrieben, wie ein durch sie induziertes Fehlverhalten der Sensordatenverarbeitung im Rahmen einer Bewertung identifiziert werden kann.

Im letzten Abschnitt der Methodik, wurde die Ausführung eines Simulationsexperimentes beschrieben, welche wahlweise durch die Sensorsimulation als alleinstehende Simulation oder durch die Integration in eine Ko-Simulationsumgebung durchgeführt werden kann.
Wobei die Integration in eine Ko-Simulationsumgebung die Simulation von realistischem Verhalten, beobachtbarer Objekte begünstigt.

Kapitel 5

Implementierung

Das Implementierungskapitel beschäftigt sich mit der zum Nachweis des Ansatzes durchgeführten prototypischen Umsetzung. Diese wird in zwei Abschnitte unterteilt. Zum Einen dem modellbasierten Softwareentwicklungsprozess (MDSD, eng. Model Driven Software Development) und die darauf aufbauende prototypische Umsetzung des Sensorsimulationsframeworks in der Programmiersprache C++.
Dabei werden im ersten Abschnitt die entstandenen Entwicklungstools vorgestellt, mit denen das Ziel einer einfachen Erweiterbarkeit des vorgestellten Ansatzes ermöglicht wird.
Der zweite Abschnitt beschäftigt sich dann zunächst mit den softwaretechnischen Voraussetzungen sowie dem sich aus dem MDSD Ansatzes ergebenen Vorteilen. Dies beinhaltet die Vorstellung der benötigten Laufzeitumgebung, sowie die darauf aufbauenden Möglichkeiten zur automatischen Visualisierung, Manipulation und Synchronisierung der Softwaremodelle. Und damit die technische Erfüllung der Anforderungen A2 (Beobachtbarkeit) und A3 (Konfigurierbarkeit).
Weiterhin werden im zweiten Abschnitt Techniken und Methoden beschrieben, mit denen auf programmatischer Ebene mit den im Szenariomodell abgebildeten Objekten und Strukturen umgegangen werden kann, um die Implementierung neuer Simulationskomponenten voranzutreiben.

5.1 Modellgetriebene Softwareentwicklung

In dem Buch “Modellgetriebene Softwareentwicklung: MDA und MDSD in der Praxis“ liefern die Autoren eine Beschreibung des Begriffs, der modellgetriebenen Softwareentwicklung.

> *"Die modellgetriebene Softwareentwicklung befasst sich mit der Automatisierung in der Softwareherstellung. Dies bedeutet, dass möglichst viele Artefakte eines Softwaresystems generativ aus formalen Modellen abgeleitet werden. Bei den erzeugten Artefakten handelt es sich nicht nur um Quelltext einer ausführbaren Programmiersprache (Java, C++, etc.), sondern auch um andere Dateien, die für die Lauffähigkeit des Systems benötigt werden, z.B. Konfigurationsdateien, Resourcebundles und Datenbankskripte. Darüber hinaus können auch den Entwicklungsprozess unterstützende Artefakte generiert werden, z.B. Softwaretests und Dokumentation."* [TB07]

Im Rahmen des Sensorsimulationsframeworks stellen das in Abschnitt 4.2.1 beschriebene Metamodell, sowie das in Abschnitt 4.3.2.2 beschriebene Komponentenmodell die formalen Modelle dar, aus denen die Softwareartefakte erzeugt werden können.
Bei den erzeugten Artefakten wiederum handelt es sich vornehmlich um C++ Quellcode, mit dem eine Instanziierung des World Data Modells und damit letztendlich die Beschreibung des Szenariomodells zur Laufzeit der Simulation möglich ist. Gleichzeitig lässt sich über eine ähnliche Methodik der Quellcode für andere Zielplattformen, wie beispielsweise Java generieren; wobei dieser Weg weniger für die Sensorsimulation als vielmehr für die hier nicht weiter behandelten Verhaltenssimulationen genutzt wird.
Bei den aus dem Komponentenmodell generierten Artefakten handelt es sich ebenfalls um C++ Quellcode, dessen Verhalten noch manuell, d.h. durch manuelle Implementierung angepasst werden muss.
Bei der verwendeten Interprozesskommunikationsschnittstelle zur Einbindung der externen Verhaltenssimulationen handelt es sich um Middleware, welche über generische Schnittstellen verfügt. Die dafür benötigten Konfigurationsdateien, die sogenannten Object Model Templates werden dabei ebenfalls aus dem gemeinsam verwendeten Datenmodell, dem World Data Model, erzeugt.
Die Abbildung 5.1 zeigt diese Zusammenhänge optisch auf und ordnet die einzelnen Artefakte ihren jeweiligen Datenquellen, sowie den verwendeten Codegeneratoren zu. Für die Generierung der Softwareartefakte kommen drei Generatoren zum Einsatz, wobei zwei von ihnen näher behandelt werden. Bei dem ECore Quellcode Generator handelt es sich um ein Projekt aus dem Eclipse Modeling Framework ([SBMP08]), mit dem Java Quellcode für die Verhaltenssimulatoren generiert werden kann.
Das für die C++ Plattform verwendete Äquivalent ist der sogenannte *YMLGen* Quellcode Generator, der sowohl für die Generierung der Datenmodelimplementierung als auch für die der Softwarekomponenten eingesetzt wird. Die Möglichkeit an dieser Stelle auf den gleichen Generator aufzubauen, wird dadurch ermöglicht, dass es sich bei dem Komponenten Metamodell um eine Erweiterung des in Abschnitt 4.2.1.1 beschriebenen Daten-Metamodells handelt.

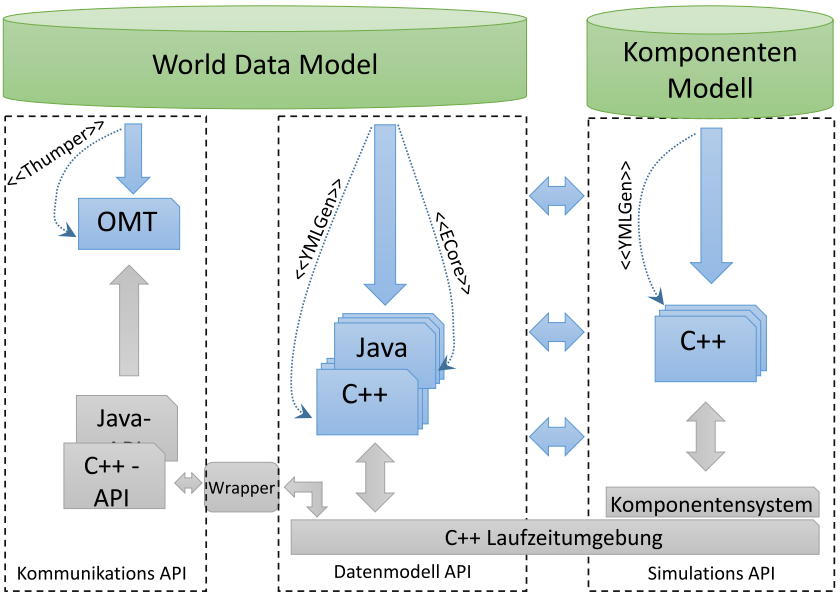

ABBILDUNG 5.1: Modelgetriebener Softwareentwicklungsansatz für das World Data Model sowie das Komponenten Metamodell. Annotiert sind die verwendeten Quellcodegeneratoren; Grau hinterlegte Artefakte liegen bereits als API vor.

Bei dem Thumper Generator handelt es sich um einen eigenständigen Konfigurationsdateigenerator, der ausgehend von dem World Data Model, speziell auf die verwendete HLA-Middleware zugeschnittene Konfigurationsdateien erstellen kann.

Die Umsetzung der Quellcode bzw. Konfigurations- Generatoren erfolgte mit Hilfe des Eclipse Modeling Projects[1] in der Programmiersprache Java. Sie umfasst neben den eigentlichen Generatoren die Entwicklung von zwei domänenspezifischen Sprachen (DSL), namentlich *YML* und *Thumper*, in denen die zu generierenden Modelle ausgedrückt werden.
Im Folgenden wird zunächst auf diese DSLs eingegangen und anschließend auf die Quellcodegeneratoren.

5.1.1 Domänenspezifische Sprachen

Für die Modellierung des Komponentenmodells sowie des World Data Models wurde eine eigenständige Sprache entwickelt, die die in den Abschnitten 4.2.1.1 und 4.3.2.2 vorgestellten Metamodelle umfasst. Dabei handelt es sich um eine textbasierte Beschreibung der Klassen, Datenstrukturen und Komponenten.
Für die Entwicklung einer textuellen Beschreibungssprache wurde auf das XText[2] Projekt aus dem Eclipse Modeling Projekt zurückgegriffen.

[1] `https://eclipse.org/modeling/`
[2] `https://eclipse.org/Xtext/`

Dabei handelt es sich um ein Framework zur Entwicklung von domänenspezifischen Sprachen, basierend auf einer Grammatik in erweiterter Backus-Nauer-Form (BNF) [EV06]. Abbildung 5.2 zeigt unter anderem ein Beispiel des generierten Editors (linkes Drittel des Bildes). Bei der erzeugten Sprache handelt es sich zunächst in großen Teilen um eine

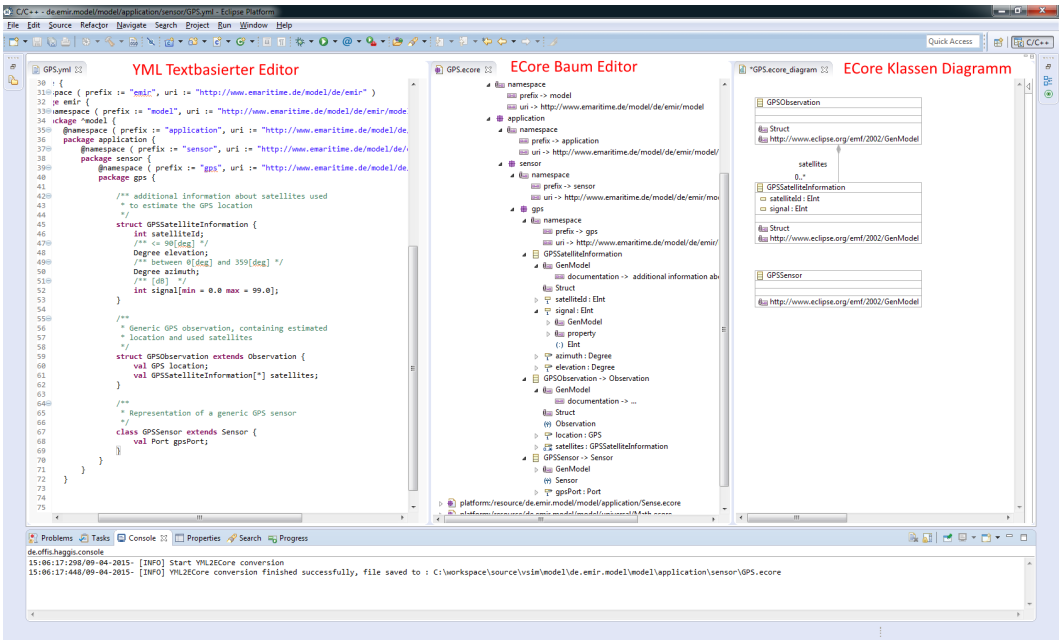

Abbildung 5.2: Editoren zum Erstellen des World Data Models

Nachbildung der Konzepte, wie sie von ECore verwendet werden. Diese von ECore bzw. dem EMOF (ECore stellt eine Implementierung des EMOF Modells dar) übernommenen Konzepte werden verwendet, um das World Data Model zu beschreiben. In einem zweiten Schritt, wurde die DSL um die Konzepte aus dem Komponenten Metamodell erweitert. D.h. es wurden zusätzlich zu den bestehenden Klassen und Datenstrukturen, das neue Konzept der Komponente (vgl. Abschnitt 4.3.2.2) sowie ihre Daten Ein- und Ausgänge in die Sprache integriert.

Da für die Beschreibung des World Data Models nur Konzepte aus dem ursprünglichen EMOF / ECore Modell verwendet werden, kann eine bidirektionale Transformation, dieser Modelle, in das ECore Format durchgeführt werden. Somit kann der generierte Editor als ein zusätzlicher Editor für das ECore Framework verstanden werden. Diese Transformation gilt nicht für das Komponentenmodell. Abbildung 5.3 zeigt zwei Beispiele für die textuelle Repräsentation der Modelle. Auf der linken Hälfte der Abbildung ist ein Ausschnitt aus dem World Data Model dargestellt, wobei es sich dabei um die Definition eines DGPS-Sensors handelt.

Die rechte Hälfte der Abbildung stellt die Modellierung der entsprechenden Simulationskomponente dar, die während der Sensorsimulation die DGPS Messwerte erzeugt. Um die Simulationskomponenten abbilden zu können, wurde das zusätzliche Schlüsselwort *component* eingeführt, wodurch die Definition einer eigenen Komponente eingeleitet wird.

```
/**
 * The differential GPS observation
 */
struct DGPSObservation extends Observation {
    /** estimated location */
    val GPS                         location;
    /** used satellites to calculate location */
    val GPSSatelliteInformation[*]  satellites;
    /** used correction signal from DGPS reference station */
    val DGPSCorrectionSignal        correctionSignal;
}


/**
 * Representation of a generic differential GPS sensor
 */
class DGPSSensor extends Sensor {
    val Port gpsPort;
}
```

```
/**
 * A Component that simulates a differential GPS Sensor
 */
component DGPSSensor extends SensorSimulator {
    /** The Sensor to simulate */
    ref GPSSensor       sensor;
    /** an optional correction signal from an DGPS reference station */
    subscribe OPTIONAL correction : DGPSCorrectionSignal;
    /** the generated differential gps observation */
    publish observation : DGPSObservation;
}
```

Abbildung 5.3: Beispiel der textuellen Repräsentation des World Data Models (links) und einer Sensor Komponente im Komponentenmodell (rechts)

Eine Komponente kann wie im Komponenten-Metamodell vorgegeben, verschiedene Dateneingänge oder Datenausgänge definieren, die in der DSL durch die Schlüsselwörter *subscribe* und *publish* angegeben werden; jeweils gefolgt von dem Namen des Ein bzw. Ausganges und des jeweiligen Datentypen.

5.1.2 Thumper - DSL

Die *Thumper-DSL* wird verwendet um die zu synchronisierenden Elemente des World Data Models zwischen zwei oder mehr Simulationen in einer gekoppelten Simulation zu definieren. Dabei müssen nicht immer alle Informationen, die im Datenmodell definiert worden sind, zwischen den Systemen synchron gehalten werden. Wird zum Beispiel eine Eigenschaft in der Ko-Simulation von nur einem Simulator benötigt oder handelt es sich um eine statische Eigenschaft, muss diese nicht an weitere Simulatoren kommuniziert werden. Ein weiteres Argument für den Bedarf eines zusätzlichen Beschreibungs- und Transformationsschrittes, von dem formal beschriebenen Datenmodell in das Format der verwendeten High Level Architecture (HLA) Middleware, liegt hauptsächlich in den Eigenheiten der verwendeten Middleware begründet.

```
 9
10 Model DGPSExample  {
11     Simulators {
12         Simulator GPSSimulator {}
13         Simulator MaritimeTrafficSimulation { }
14     }
15     HLAObject HLAVessel := de.emir.model.application.vehicle.Vessel {
16         sharings {
17             Subscribe : GPSSimulator
18             Publish : MaritimeTrafficSimulation
19         }
20         /** the unique id of the vessel */
21         id := id;
22         /** the name of the vessel */
23         name := name;
24         /** latitude of the vessels location */
25         lat := pose as Pose -> location as GPS -> latitude;
26         /** longitude of the vessels location */
27         lon := pose as Pose -> location as GPS -> longitude;
28     }
29 }
```

Abbildung 5.4: Beispiel für die Definition von austauschbaren Elementen zwischen Simulatoren in der Ko-Simulationsumgebung über Thumper

Die im Rahmen der Ko-Simulation verwendete und u.a. in Puch et.al. [PFOL12] und

Läsche et. al. [LGH13] beschriebene Middleware, ist auf die Kommunikation von primitiven Datentypen beschränkt, während es sich bei dem Datenmodell um ein objektorientiertes Datenmodell handelt. Für eine weiterführende Beschreibung sei an dieser Stelle auf die Arbeit “Interoperability in Co-Simulations of maritime systems“ [DHS14] verwiesen.

Bei der *Thumper DSL* handelt es sich wie auch schon bei der Beschreibung des Datenmodells und des Komponentenmodells, um eine auf XText basierende domänenspezifische Sprache. Mit ihr werden ausgehend von den im Datenmodell spezifizierten Klassen und Datenstrukturen, neue “flache“[3] Datenstrukturen definiert.
Die Abbildung 5.4 zeigt ein Beispiel für die Definition der zu synchronisierenden Elemente des Datenmodells, für ein Beispiel welches in der Evaluation ausführlich behandelt wird. Dabei befinden sich zwei Simulationen in der Ko-Simulationsumgebung, zum Einen die Sensorsimulation, die in Zeile 12 mit dem Namen “GPSSimulator“ deklariert wird, sowie eine Verhaltenssimulation mit dem Namen “MaritimeTrafficSimulation“.
Zwischen den beiden Simulatoren wird lediglich die Position von Schiffen in der simulierten Umgebung ausgetauscht, wobei die GPS-Simulation in diesem Beispiel eine ausschließlich lesende Rolle innerhalb der Ko-Simulation einnimmt. Die “Maritime Traffic Simulation“ (MTS) hingegen nimmt eine ausschließlich schreibende Rolle ein (Festgelegt in den Zeilen 17 & 18).
Für den Austausch wird eine neue Datenstruktur mit dem Namen “HLAVessel“ definiert, die auf die Definition der Klasse “Vessels“ aus dem World Data Models zurückgeführt werden kann (Zeile 15). Im Anschluss (Zeilen 21-27) werden die zu synchronisierenden Attribute definiert. Dabei darf es sich ausschließlich um fundamentale Datentypen handeln. Um dies zu gewährleisten wurde die in den Zeilen 25 und 27 verwendete Notation eingeführt, welche textuell wie Folgt interpretiert werden können.

> *Von der Klasse “Vessel“ aus dem objektorientierten Datenmodell nehme den Wert der Eigenschaft “pose“ und interpretiere ihn als eine Instanz der Klasse “Pose“. Von dieser Pose verwende die Eigenschaft “location“ und interpretiere diese als eine “GPS“ Koordinate. Von dieser “GPS“ Koordinate verwende die Eigenschaft “latitude“ unter dem Namen “lat“.*

Für die Generierung der Object Model Templates (OMT) werden die Datenstrukturen aus dem World Data Model ausgelesen und in die entsprechenden HLA Datentypen transformiert. Das Ergebnis ist eine, in XML beschriebene, OMT Datei, wie sie ausschnittsweise in Listening 5.1 dargestellt ist.

[3] in diesem Zusammenhang wird von flachen Klassen gesprochen, wenn sie ausschließlich primitive Datentypen (int, float, double, ...) verwenden

```
<objectClass name="HLAPose" semantics="de.emir.model.universal.math.Pose" sharing="
    PublishSubscribe">
<attribute name="lat" dataType="HLAfloat64BE" sharing="PublishSubscribe" order="
    Receive" transportation="HLAreliable" dimensions="NA" />
<attribute name="lon" dataType="HLAfloat64BE" sharing="PublishSubscribe" order="
    Receive" transportation="HLAreliable" dimensions="NA" />
<attribute name="heading" dataType="HLAfloat32BE" sharing="PublishSubscribe" order="
    Receive" transportation="HLAreliable" dimensions="NA" />
</objectClass>
```

Listing 5.1: Ausschnitt aus dem generierten Object Model Template

5.1.3 YMLGen - Quellcode Generierung

Während das Eclipse Modeling Framework einen guten und erprobten Quellcodegenerator für die Programmiersprache Java zur Verfügung stellt, existieren im gleichen Umfeld wenige bis keine Generatoren für die Zielplattform C++. Existierende Quellcodegeneratoren, wie sie unter anderem von dem PRIDE[4] Projekt oder von Papyrus[5] zur Verfügung gestellt werden, fokusieren sich auf das generieren von Quellcode für eingebettete Systeme. Eine Open-Source Alternative, mit der ECore Modelle (und damit auch das World Data Model) in C++ Quellcode generiert werden kann, ist das Projekt Emf4Cpp[6]. Die Arbeiten an diesem Projekt wurden allerdings im Jahr 2010 eingestellt.

Für die Implementierung des *YMLGen* genannten C++ Quellcodegenerators wurde das Eclipse Projekt XTend[7] verwendet. Bei XTend handelt es sich um einen Dialekt der Programmiersprache Java, der stark in das Eclipse - Ökosystem integriert ist. Eines der zur Sprache Java hinzugefügten Features, ist eine Templatesprache, die sich gut für die Umsetzung von “Model to Text“ Transformationen einsetzten lässt.

Als Grundlage der Generierung dient eine Instanziierung des Komponenten Metamodells, auf dessen Modellelemente während des Generierungsprozesses zugegriffen werden kann. Dabei werden die modellierten Komponenten, Klassen und Datentypen in jeweilige C++ Klassen überführt. Berücksichtigt werden dabei die modellierten Klassenhierarchien, Eigenschaften und Operationen die mit dem Metamodellen definiert werden können. Das Ergebnis ist ein syntaktisch korrektes C++ Programm, dessen Inhalt im Falle des World Data Models nicht weiter angepasst werden muss, da diese Elemente über kein eigenes Verhalten verfügen und lediglich zur Haltung von Daten verwendet werden.
Werden dagegen mit Hilfe des *YMLGen* Generators, Simulationskomponenten erstellt

[4] http://www.idt.mdh.se/pride/
[5] https://www.eclipse.org/papyrus/
[6] http://www.catedrasaes.org/wiki/EMF4CPP
[7] https://eclipse.org/xtend/index.html

bzw. Modellelemente die über zusätzliche Operationen verfügen, müssen diese im Nachhinein manuell implementiert werden. Der Generator erzeugt an dieser Stelle einen leeren Methoden "Stub", wie sie beispielsweise auch von den CORBA IDL Generatoren erstellt werden.

Um eine kontinuierliche Weiterentwicklung sowohl auf Modellebene (Änderungen am Modell) als auch auf Implementierungsebene (manuelle Änderungen im C++ Quelltext) gewährleisten zu können, kommen sogenannte generierte Regionen ("generated Regions") zum Einsatz. Bei dieser Technik, die unter anderem auch von dem ECore Generator eingesetzt wird (vgl. JMerge[8]), werden vom Generator erzeugte Quellcodeabschnitte als automatisch generiert markiert. Wird eine erneute Generierung des Quellcodes vorgenommen, werden nur solche Elemente überschrieben bzw. geändert, die über eine entsprechende Markierung verfügen. Auf diese Weise lassen sich manuelle Änderungen im Quelltext an beliebigen Stellen durchführen, ohne dass diese bei einem erneuten Generierungsvorgang überschrieben werden.

5.2 Laufzeitumgebung

Die Laufzeitumgebung setzt sich, wie in Abbildung 5.5 dargestellt aus den drei Teilprojekten *Runtime*, *Benutzerschnittstelle* und *Thumper* zusammen. Diese drei Projekte bilden die Grundlage für den modellbasierten Ansatz.

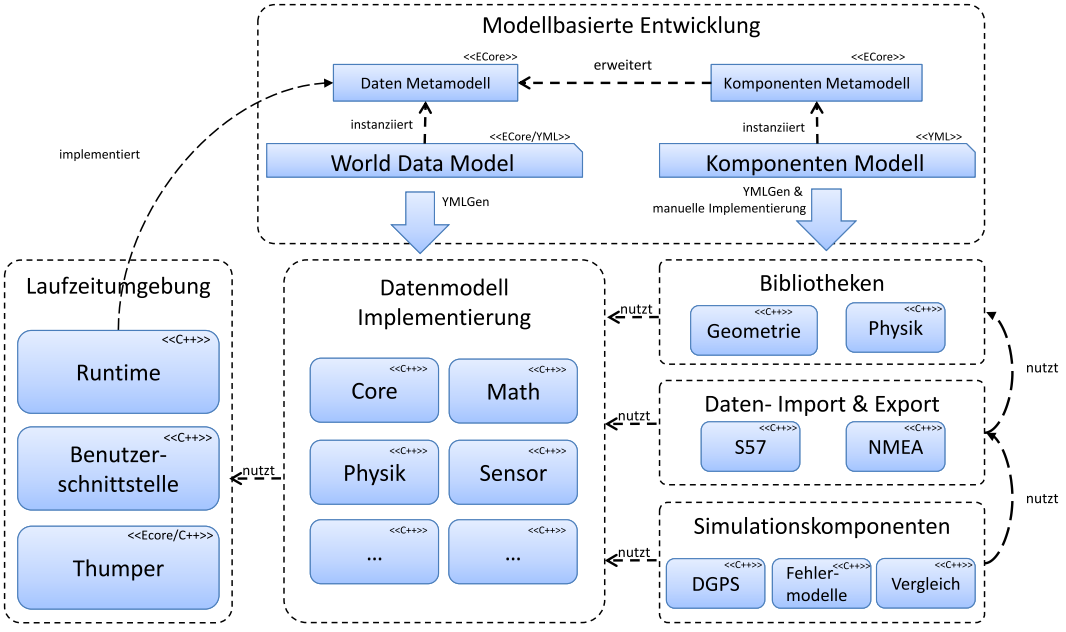

Abbildung 5.5: Überblick über die generierten und genutzten Softwareartefakte

[8] https://wiki.eclipse.org/JET_FAQ_What_is_JMerge%3F

5.2.1 Runtime

Die Hauptaufgabe der Laufzeitumgebung, spezieller des Runtime Projektes aus Abbildung 5.5, ist die Bereitstellung einer Reflexionsumgebung, wie sie u.a. von dem Essential MOF gefordert wird [OMG14]. Damit wird das Ziel verfolgt, zur Laufzeit auf die Struktur eines Classifiers des World Data Models, bzw. einer Komponente zugreifen zu können ohne diese genau kennen zu müssen.
Diese Eigenschaft wird an verschiedenen Stellen im Sensorsimulationsframwork verwendet. Zum Beispiel lässt sich auf diese Weise eine generische Serialisierungs bzw. Deserialisierungsroutine entwickeln, über die beliebige Modellinstanzen ausgetauscht werden können.
Ein anderes Anwendungsbeispiel für den Nutzen von Reflexion, ist das Bereitstellen einer Benutzungsoberfläche über die die Modelle sowohl analysiert als auch manipuliert werden können. Damit ist die Fähigkeit zur Reflexion einer Klasse ein elementarer Bestandteil zur Erfüllung der Anforderungen A2 (Beobachtbarkeit) und A3 (Konfigurierbarkeit) bei denen, die manuelle oder automatische Beobachtung und Manipulation von Elementen der Umgebung gefordert wird.

Da die verwendete C++ Plattform, von sich aus keine Methoden zum Zugriff auf die verwendete Klassenstruktur ermöglicht, wurden verschiedene existierende Umsetzungen von Reflexionen in C++ auf ihre Eignung untersucht. Dazu gehörten die Lösungen *Helim-Reflex*[9], das vom CERN entwickelte *ROOT Reflex* [RM04] sowie die Open Source Lösung *CPGF*[10].
Bei der Untersuchung hat sich herausgestellt, das keine der untersuchten Bibliotheken in der Lage war, das für C++ spezifische Konzept der virtuellen Vererbung zu unterstützten, welches für die im Metamodell verwendete Mehrfachvererbung benötigt wird.

Da sich keine der untersuchten Bibliotheken als geeignet erwiesen hat, wurde eine eigene Bibliothek entwickelt, welche sich in Benamung und Struktur stark an der Realisierung des ECore Systems orientiert. Gleichwohl wurden plattformspezifische Elemente wie beispielsweise die in C++ verfügbaren Zeiger auf Membervariablen oder Funktionen verwendet, um eine effiziente Umsetzung zu erreichen.
Der Argumentation von Geoff Evans, einem der Entwickler des Helium Projektes folgend, handelt es sich bei dem Reflexionssystem um eine sogenannte “In Code“ Lösung, bei der die benötigten Klassen und Strukturen einer reflektierten Klasse im Quelltext definiert werden [EVA11]. Auf diese Weise kann eine Konsistenz des Reflexionsystems mit der aktuellen Implementierung sichergestellt werden.

[9] `http://heliumproject.org/`
[10] `http://www.cpgf.org/`

Das Einfügen der benötigten Instruktionen zum Reflektieren einer neuen Klasse, wird vollständig durch den oben beschriebenen Quellcodegenerator übernommen.

5.2.1.1 Plugin System

Eine weitere Aufgabe der Laufzeitbibliothek ist die Verwaltung von Plug-Ins. Das Simulationsframework basiert auf einer Plug-In Struktur. Dabei wird jedes neue Model aus dem World Data Model bzw. dem Komponentenmodell, in einem eigenständigen Plug-In realisiert. Durch dieses Vorgehen kann eine einfache Erweiterbarkeit gewährleistet werden, ohne dass Änderungen an der bestehenden Programmstruktur vorgenommen werden müssen. Zusätzlich wird dadurch die bereits beim Aufbau des World Data Models beschriebene Zerlegung in Teilmodelle Rechnung getragen, bei der ein dynamisches hinzufügen der benötigten Modelle postuliert wurde.

5.2.2 Benutzungsschnittstelle

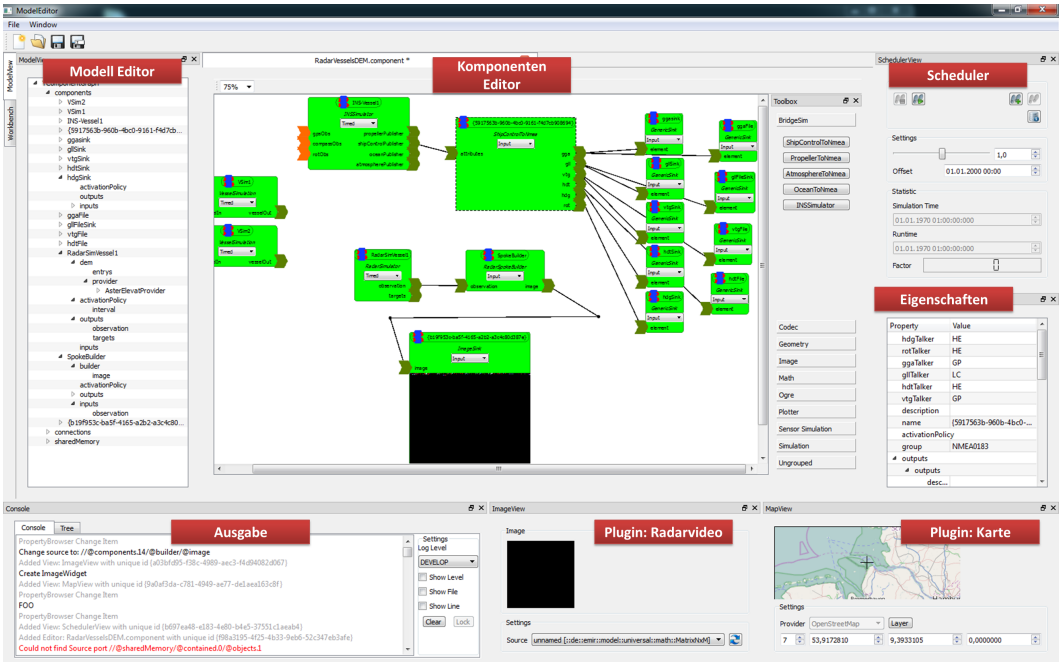

Abbildung 5.6: Screenshot des Komponenten Editors, zur Orchestrierung von Komponenten in einem Datenflussgraphen.

Zur Visualisierung und Interaktion mit dem Szenariomodell sowie den Simulationskomponenten, wurde eine graphische Benutzerschnittstelle erstellt, die ähnlich wie die Eclipseoberfläche aufgebaut ist. Sie besteht aus einem Basis Framework, in dem Views und Editoren in sogenannten Perspektiven zusammengefasst werden können. Bei den Perspektiven handelt es sich um eine Vorschrift, wie die GUI-Elemente innerhalb der Benutzeroberfläche angeordnet werden sollen. Bekannt wurde dieses Konzept durch die

Einführung der Eclipse Rich Client Plattform.
Bei den Views handelt es sich um Fenster, die um einen zentralen Editor angeordnet werden können und in der Regel zusätzliche Informationen zu den aktuellen Inhalten des Editors anzeigen. Sowohl Editoren als auch Views werden in der Benutzeroberfläche durch Plug-Ins realisiert.
Im Rahmen der Sensorsimulation werden zwei Editoren verwendet. Zum Einen der sogenannte “ModelEditor“, dargestellt in Abbildung 5.7, sowie den KomponentenEditor, dargestellt in Abbildung 5.6.

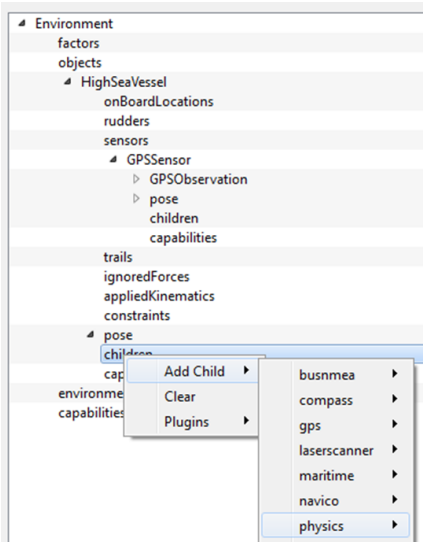

Abbildung 5.7: Abbildung des Model Editors zur Darstellung des Szenariomodells

Bei dem “ModelEditor“ handelt es sich um einen baumbasierten Editor, der auf Grundlage der zuvor beschriebenen Reflexion Bibliothek aufgebaut wird. Mit Hilfe des “ModelEditors“ können die Instanzen des Szenariomodells sowohl erstellt, als auch währen der Laufzeit beobachtet und manipuliert werden.
Während der “ModelEditor“ die Struktur der Objektinstanzen aufzeigt, wird die “Property View“ (vgl. rechte Seite in Abbildung 5.6) verwendet um die Werte von Eigenschaften darzustellen oder zu ändern.

Bei dem zweiten Editor handelt es sich um den sogenannten Komponenteneditor. Dieser wird verwendet um verschiedene Simulationskomponenten miteinander zu orchestrieren.
Komponenten können per Drag & Drop aus der Palette am rechten Rand des Editors, in den Graphen gezogen werden. Über die Verbindungen wird anschließend der Datenfluss zwischen den einzelnen Komponenten gesteuert.
Neben diesen beiden Editoren existieren verschiedene Views, die entweder zusätzliche Informationen anzeigen, wie beispielsweise die Kartendarstellung, in der die verorteten Objekte des Editors auf einer Karte angezeigt werden können (unten rechts in Abbildung 5.6) oder die Steuerung der simulierten Zeit gemäß Anforderung A3 ermöglichen (*SchedulerView*).

5.2.3 Thumper Sychronisierung

Wie bereits im Abschnitt 5.1.2 beschrieben handelt es sich bei Thumper sowohl um eine Beschreibungssprache, die Thumper-DSL, als auch um eine Laufzeitkomponente, welche die Synchronisierung zwischen den Ko-Simulationen während der Laufzeit übernimmt. Zu diesem Zweck nutzt die Laufzeitkomponente neben der zuvor beschriebenen Reflexion Bibliothek, eine Besonderheit des von *YMLGen* generierten Quelltextes.
Ähnlich wie es in ECore gehandhabt wird, wird in den generierten C++ Quelltext ein Observer Pattern integriert, welches einen optionalen Beobachter über Änderungen von Eigenschaften einer Klasse informiert. Thumper registriert sich als ein solcher Beobachter und ist somit in der Lage, Änderungen die sich während eines Simulationsschrittes ergeben, automatisch an die Ko-Simulation weiterzuleiten. Durch den Einsatz der Reflexion Bibliothek müssen zudem die kommunizierten Klassen nicht direkt bekannt sein, so dass diese Technik auch für alle neu entwickelten Datenmodellmodule angewendet werden kann.

5.2.4 Generierte Artefakte

Bei den in diesem Teilabschnitt behandelten generierten Artefakten handelt es sich hauptsächlich um die Bibliotheken, mit denen die Entwicklung neuer Simulationskomponenten vereinfacht werden kann. Auf die in Abbildung 5.5 dargestellte Implementierung des Datenmodells wird nicht weiter eingegangen, da es sich hierbei um ausschließlich generierten Quellcode handelt, dessen Struktur bereits bei dem Aufbau des Datenmodells diskutiert wurde. Dabei wird für jedes im Datenmodell definierte Teilmodell ein entsprechendes C++ Projekt generiert, welches über das Plug-In Framework zur Verfügung gestellt und verwendet werden kann.

Bei den weiteren aus den formalen Modellen generierten Softwareartefakten handelt es sich neben den, in Abbildung 5.5 dargestellten Simulationskomponenten um Hilfsbibliotheken die den Zugriff auf die Daten des Szenariomodells zur Laufzeit vereinfachen sollen.
Für deren Implementierung wird ebenfalls der zuvor beschriebene modellbasierte Ansatz angewendet, insbesondere um die damit verbundene Verfügbarkeit der Reflexion Bibliothek nutzen zu können.
Ein Beispiel für die in der Gruppe *Bibliotheken* zusammengefassten Softwareartefakte ist eine Anfrage nach der absoluten Position eines Objektes in einer Umgebung. Diese Information wird zum Beispiel von verschiedenen Simulatoren, u.a. dem DGPS-Sensor zur Laufzeit der Simulation benötigt und kann nicht immer direkt aus dem Datenmodell

entnommen werden, sondern muss in Beziehung zu der Objekthierarchie des entsprechenden Objektes gesetzt werden.
Ein weiteres Beispiel sind Kollisionsanfragen, auf den von den Sensoren und Emittern definierten beobachteten respektive beeinflussten Gebieten, welche unter Zuhilfenahme der frei verfügbaren GEOS[11] Bibliothek für zweidimensionale Geometrien bzw. der ebenfalls frei erhältlichen Bullet[12] Bibliothek für dreidimensionale Geometrien im Datenmodell realisiert wurden.

Die zweite, in Abbildung 5.5 dargestellte Gruppe von Softwarekomponenten ist die Gruppe der Daten Importer und Exporter. In diese Gruppe fallen zum Beispiel die Kommunikationsprotokolle über die mit der externen Sensordatenverarbeitung kommuniziert werden kann. Ein Beispiel für einen Exporter ist das NMEA0183 Datenformat zur Kommunikation verschiedenster nautischer Informationen, wie aktuelle Position, Anzahl sichtbarer Satelliten, usw..
Neben Komponenten und Hilfsbibliotheken die während der Laufzeit verwendet werden können, wurden in dieser Gruppe auch Softwarekomponenten entwickelt, die während der Designzeit eines Experimentes zum Einsatz kommen. Dazu gehören insbesondere die Datenimporter.
Dahinter steht die Idee, dass das gemeinsam genutzte Datenmodell von verschiedenen Simulationen als Informationsquelle verwendet werden kann und somit ein Import von Informationen aus einem externen Format nur an einer einzigen Stelle notwendig ist. Ein Beispiel für einen solchen Anwendungsfall ist in Abbildung 5.8 dargestellt.

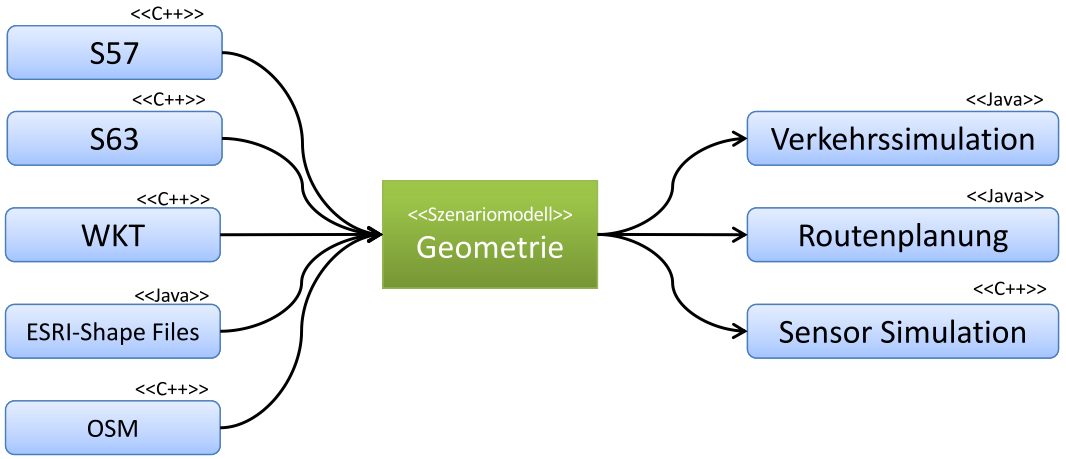

Abbildung 5.8: Verwendung des gemeinsamen Datenmodells als Datenquelle

Für eine Ko-Simulation werden zum Beispiel, an verschiedenen Stellen und ggf. auch in verschiedenen Systemen die Informationen über die geographischen Gegebenheiten

[11] Bei der GEOS Bibliothek (`http://trac.osgeo.org/geos/`) handelt es sich um einen C++ Port der Java Topologie Suite (JTS), die unter anderem in verschiedenen Geo Informationssystemen wie beispielsweise QGIS (`http://qgis.org/de/site/`) verwendet wird.
[12] Bei Bullet (`http://bulletphysics.org`) handelt es sich um eine Quelloffene Physik-Engine, die unter anderem eine performante Kollisionserkennung im dreidimensionalen Raum umsetzt.

benötigt. Diese Informationen können je nach Verfügbarkeit aus verschiedenen externen Datenquellen gewonnen werden. Seekarten werden zum Beispiel von nationalen Organisationen im S-57 oder S-63 Datenformat zur Verfügung gestellt. Sind für ein untersuchtes Gebiet keine Seekarten verfügbar, können die benötigten Informationen ggf. aus anderen Datenquellen extrahiert werden, wie beispielsweise dem Online Service Open Street Map, der zum Teil nautische Informationen enthält oder durch manuell definierte Shape Files.
Diese Informationen können durch einen Importer in das, durch das World Data Model, vorgegebene Datenformat überführt werden, welches von jedem an der Ko-Simulation beteiligten System verstanden werden kann.

Kapitel 6

Evaluation

Innerhalb des Evaluationskapitels dieser Arbeit werden die zuvor vorgestellten Konzepte (Kapitel 4) sowie die prototypische Implementierung (Kapitel 5) hinsichtlich ihrer Anwendbarkeit und Anforderungserfüllung untersucht.

Dazu wird das Kapitel in drei Abschnitte unterteilt.
Der erste Abschnitt beschäftigt sich dabei mit der Modellierung von Sensoren, der Beschreibung ihres Verhaltens und Fehlverhaltens sowie der Überprüfung hinsichtlich der in Abschnitt 1.2.1 vorgestellten Qualitätskriterien. Zu diesem Zweck wird die in Abschnitt 4.1 (bzw. Abbildung 4.2) vorgestellte Methode bestehend aus der Systemidentifikation, der Systemmodellierung sowie der Durchführung des Simulationsexperimentes, angewendet.
Im zweiten Teil des Kapitels wird die vollständige Bewertung eines sensordatenverarbeitenden Systems hinsichtlich der Qualitätskriterien durchgeführt und somit gezeigt, dass die zuvor einzeln überprüften Konzepte auch im Zusammenschluss angewendet werden können.
Den dritten Teil der Evaluation bildet ein komplexes Anwendungsszenario, bei dem die Methodik sowie das entwickelte Werkzeug, zur Unterstützung des Entwicklungsprozesses eines neuen sensordatenverarbeitenden Systems, eingesetzt wird. In diesem Zusammenhang wird auf die Modellierung komplexer maritimer Sensoren zur Generierung von Sensorrohdaten, am Beispiel eines maritimen Radars, eingegangen. Zusätzlich wird auf das Zusammenspiel der Sensordatengenerierung mit einer externen Verhaltenssimulation eingegangen, die das realitätsnahe Verhalten der simulierten Schiffe übernimmt.

Bei den in diesem Kapitel untersuchten Anforderungen handelt es sich in erster Linie um die sensorsimulationsspezifischen Anforderungen A1 (Modellierbarkeit), A5 (Fehlerfreie Sensoren), A6 (konfigurierbare Fehlermodelle), A7 (Verwendung existierender

Schnittstellen) aus Abschnitt 3.4.1. Bei den Anforderungen A2 (Konfigurierbarkeit) & A3 (Beobachtbarkeit) handelt es sich dagegen um Anforderungen, die durch den gewählten Systemaufbau und die damit verbundene Implementierung erfüllt und somit hier nicht ausführlich behandelt werden.
Dies gilt insbesondere für die Beobachtbarkeit und Konfigurierbarkeit, die unter anderem durch den im vorherigen Kapitel beschriebenen modellbasierten Softwareentwicklungsprozess und die damit einhergehende Reflexionsbibliothek erfüllt werden.

Die Anforderung A9 (Nachvollziehbarkeit) nimmt an dieser Stelle eine besondere Rolle ein, da das dazugehörige Problem, die Verhaltenssimulation an eine Ko- Simulationsumgebung ausgelagert wurde. Da es sich bei den Verhaltenssimulationen um externe Simulationen handelt, ist an dieser Stelle kein Nachweis ihrer Korrektheit bzw. Nachvollziehbarkeit möglich. Es wird allerdings davon ausgegangen, dass die Verhaltenssimulationen ein für das untersuchte Szenario plausibles Verhalten präsentieren. Ist dies nicht der Fall, können sie, aufgrund der losen Kopplung sowie den offenen Schnittstellen, gegen andere Simulationssysteme ausgetauscht werden, die diese Anforderung erfüllen.

6.1 Evaluation einzelner Konzepte

Der erste Abschnitt des Evaluationskapitels beschäftigt sich mit dem Nachweis einzelner Anforderungen und Konzepte aus den vorangegangenen Kapiteln. Gemäß der vorgestellten Methodik handelt es sich dabei zunächst um die Modellierung der Sensoren sowie ihres Verhaltens und ihres Fehlverhaltens.
Entsprechend des Vorgehens in Abschnitt 4.3.2 kann dies in 1) die strukturelle Beschreibung der Sensoren und ihrer Sensorbeobachtungen auf Basis des Metamodells (vgl. Abschnitt 4.2.1.1) und 2) in die Verhaltensmodellierung auf Basis des Komponentenmodells (vgl. Abschnitt 4.3.2.2f), unterteilt werden.
Bei der Modellierung des Verhaltens werden die in Abschnitt 4.3.3 beschriebenen Sensorfehler und Messungenauigkeiten:

- Sensorfehler
- Zufälliger Fehler
- Kontextabhängiger Fehler

berücksichtigt und jeweils gezeigt, wie eine Überprüfung der Sensordatenverarbeitung hinsichtlich des entsprechenden Fehlermodells durchgeführt werden kann.

Zunächst wird jedoch eine Systemidentifikation durchgeführt, in der das zu testende System vorgestellt und die zu überprüfenden Qualitätskriterien quantifiziert werden.

6.1.1 Systemidentifikation

Abschnitt 4.2 beschreibt diesen Schritt der Methodik als das Sammeln von Informationen, die für die spätere Modellbeschreibung benötigt werden. Dabei wird dieser Schritt von einem Experten ausgeführt der gegenüber der Methodik über zusätzliches Wissen in Bezug auf das zu testende System, die Qualitätskriterien sowie die Einsatzumgebung verfügt.
Die Folgenden Abschnitte stellen diese Informationen für das zur Evaluation verwendete Szenario bereit.

6.1.1.1 Vorstellung der Sensordatenverarbeitung

Bei dem zu überprüfenden sensordatenverarbeitenden System, handelt es sich um eine PNT (Position, Navigation and Timing) Unit, wie sie in [EHR+15] vorgestellt wird.
Die PNT Unit bestimmt dabei die Position und die Ausrichtung eines Schiffes, mithilfe von zwei unabhängigen GPS Sensoren, die an bekannten Positionen auf dem Schiff montiert sind (vgl. Abbildung 6.1).

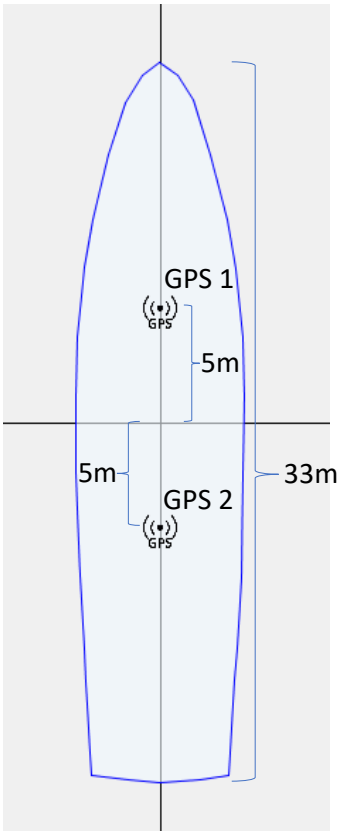

Abbildung 6.1: Anordnung der GPS Sensoren auf dem simulierten Schiff

Die Guideline [EHR+15] beschreibt vier verschiedene Genauigkeitslevel für die Positions- und Ausrichtungsbestimmung, die wiederum auf der IMO Resolution A.915(22) [IMO01] basieren. Diese besagt, dass für die Navigation innerhalb von Hafengebieten, die Position mit einer Genauigkeit von unter einem Meter bestimmt werden muss. Für die Bestimmung der Ausrichtung wird eine Genauigkeit von 0.5° gefordert.
Zusätzlich wird von einer 95% Verfügbarkeit akkurater Sensormessungen ausgegangen, also weniger als 5% der ermittelten Position und Ausrichtungen dürfen einen größeren Fehler als $1[m]$ bzw. 0.5° haben.

Bei der Sensordatenverarbeitung handelt es sich um eine verhältnismäßig einfache Umsetzung einer PNT Unit. Sie folgt dem in Abbildung 6.2 dargestellten Aufbau (vgl. NMEA 0183 Standard [Ao02]).
Für die Kommunikation der Sensormesswerte wird das in der maritimen Domäne übliche NMEA 0183 Protokoll verwendet. Die Sensordatenverarbeitung erwartet dabei zwei

NMEA 0183 - GGA (Global Positioning System Fix Data) Datensätze, die dem in Listening 6.1 angegebenen Aufbau folgen und hauptsächlich die von einem GPS-Sensor ermittelten WGS84 Koordinaten enthalten.

```
$GPGGA,HHMMSS.ss,BBBB.BBBB,b,LLLLL.LLLL,l,Q,NN,D.D,H.H,h,G.G,g,A.A,RRRR*PP
```

Listing 6.1: Aufbau eines NMEA0183 - GGA Datensatzes

Die beiden eingehenden GGA Datensätze werden jeweils über ein UDP Datagramm auf einem festgelegten Port übermittelt.

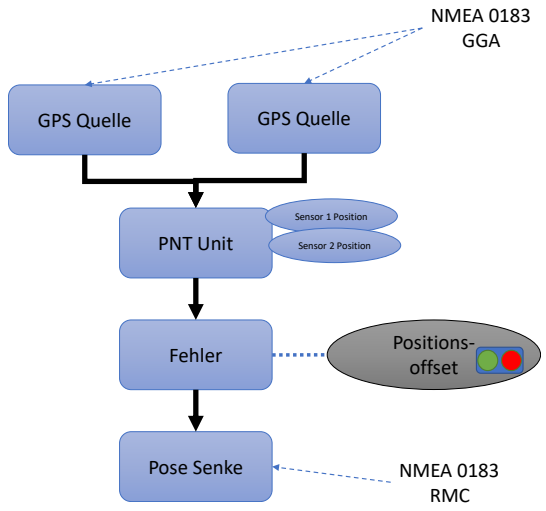

Abbildung 6.2: Schematischer Aufbau der untersuchten Sensordatenverarbeitung.

Sobald eine neue GPS Beobachtung bei der PNT Unit eingeht, überprüft diese anhand des ebenfalls enthaltenen Zeitstempels der eingehenden Nachricht ob eine Positionsbestimmung durchgeführt werden kann. Ausschlusskriterium hierfür ist ein festgelegter Schwellwert für die Differenz der beiden Eingänge.
Für die Ermittlung der Position und Ausrichtung des Schiffes können zudem die relativen Positionen der verwendeten Sensoren, im Koordinatensystem des Schiffes, verwendet werden, wie sie beispielsweise in Abbildung 6.1 angegeben sind.
Das Ergebnis der Sensordatenverarbeitung wird ebenfalls in Form eines NMEA 0183 Datensatzes an die Anwendungslogik bzw. die Prüfinstanz weitergeleitet. Dabei wird der RMC (Recommended Minimum Navigation Information) Datensatz verwendet, welcher neben der Position auch eine Ausrichtung[1] enthält.

Weitere Informationen, wie beispielsweise die genaue Methode, bzw. das mathematische Modell der Positionsbestimmung müssen an dieser Stelle nicht weiter bekannt sein, um die Sensordatenverarbeitung im Sinne einer Blackbox, auf die Erfüllung der Qualitätskriterien zu überprüfen. Da die vorgestellte Sensordatenverarbeitung zur Evaluierung der vorgeschlagenen Methode zur simulativen Überprüfung von sensordatenverarbeitenden Systemen verwendet werden soll, wird in der Sensordatenverarbeitung ein Fehler eingebaut, der manuell ausgelöst werden kann.
Dabei handelt es sich zum einen um einen Fehler in der Positionsbestimmung sowie eine künstliche Verzögerung der Ergebnisweitergabe.

[1] Die Sensordatenverarbeitung verwendet das Feld “Track made good“ für die Übermittlung der Ausrichtung, Anstelle des Kurses

Bei dem eingebauten Fehler wird auf den ermittelten Messwert ein künstlicher Offset von einem Meter, auf die Latitude Komponente der Position, addiert um eine falsche Berechnung der Position zu simulieren (vgl. Abschnitt 6.1.4.1).

6.1.1.2 Funktionsweise und Fehlermodelle von GPS Sensoren

Nachdem die zu überprüfende Sensordatenverarbeitung vorgestellt wurde, sowie die verwendeten Sensoren identifiziert wurden, obliegt es dem Anwender der Methodik, die zu überprüfenden Fehlermodelle festzulegen.
Dabei muss es sich nicht gezwungenermaßen um die tatsächlichen Fehler der verwendeten Sensoren handeln. Stattdessen können auch hypothetische Fehlermodelle, zum Beispiel von qualitativ hochwertigeren oder qualitativ minderwertigeren Sensoren modelliert werden. Sollen jedoch realistische Fehlermodelle untersucht werden, kann häufig eine Betrachtung der Funktionsweise von Sensoren zur Herleitung bzw. zum Verständnis der Fehlermodelle hilfreich sein.

Der folgende Abschnitt führt in die Funktionsweise von GPS- und DGPS-Sensoren ein und gibt eine Einführung in die möglichen Fehler, die im Zusammenhang mit (D)GPS Systemen berücksichtigt werden müssen. Für detailliertere Informationen sei hier auf das GPS-Compendium [UBl09] sowie die offiziellen Webseiten des Betreibers[2] verwiesen.

Das Global Positioning System (GPS) basiert auf der Laufzeitmessung von Funksignalen zwischen Satelliten, die die Erde in einem ungefährem Abstand von ca. 20200 km umkreisen, sowie einem geeigneten Empfänger.
Mittels der Laufzeit eines Funksignals lässt sich der Abstand des Empfängers zu einem Satelliten bestimmen. Dieser Entfernungswert wird auch Pseudoentfernung genannt. Die zweidimensionale Position (Latitude und Longitude) auf der Erdoberfläche kann durch die Triangulation mit mindestens drei Satelliten durchgeführt werden, wie es in Abbildung 6.3 (links) dargestellt ist. Für die Bestimmung der Höhe über der Erdoberfläche wird ein vierter Satellit benötigt.

Fehlerquellen Auf dem Weg zwischen dem Satelliten und dem GPS Empfänger durchquert das Funksignal verschiedene Schichten der Erdatmosphäre, zum Beispiel die Troposphäre (0 - 15km über NN) und die Ionosphäre (60-100km über NN). In diesen Schichten interagiert das Funksignal mit kleinen Partikeln / Ionen und wird von ihnen abgelenkt bzw. abgebremst (vgl. Abbildung 6.3-rechtes Bild). Durch diesen Effekt ergeben sich Ungenauigkeiten bei der Ermittlung der Distanz zwischen dem Satelliten und dem GPS Empfänger. Diese Fehler werden in der Literatur mit Werten zwischen 3.0m [UBl09] und

[2] Das GPS System wird von der US-Regierung betrieben: `http://www.gps.gov/`

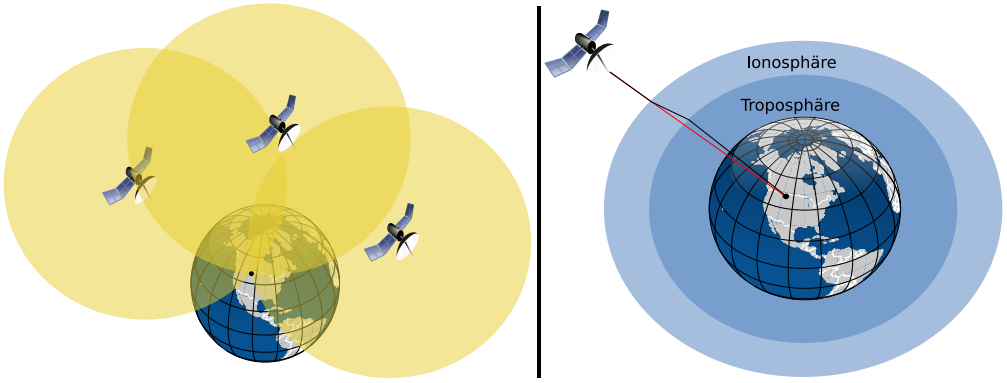

Abbildung 6.3: Links: Triangulation der Position mit drei GPS Satelliten. Die Position befindet sich am Schnittpunkt der Kugeln. Rechts: angenommener (rot) und tatsächlicher (schwarz) Weg des Funksignals durch die Atmosphäre der Erde.

5.0m [Ran94] für die Ionosphäre und 0.7m [UBl09] bzw. 2.0m [Ran94] für die Troposphäre angegeben (vgl. Tabelle 6.1).

Neben den atmosphärischen Effekten führen Abweichung in der Bestimmung der Satellitenposition zu weiteren Ungenauigkeiten in der Entfernungsbestimmung. Die Abweichungen sind auf die komplizierten Wechselwirkungen der Gravitationskräfte von Erde und Mond auf die Satelliten zurückzuführen. Dieser Fehler wird in der Literatur als Ephemeridenfehler bezeichnet.
Auch die Chronometer der Satelliten haben einen entsprechenden Einfluss auf die Genauigkeit. Obwohl die GPS - Satelliten über sehr genaue Atomuhren verfügen, wird von einer Abweichung der Zeitbestimmung von ca. 10ns pro Tag ausgegangen, was einem Fehler in der Entfernungsbestimmung von bis zu 3 Metern entspricht.
Ebenfalls die Hardware auf Empfängerseite unterliegt i.d.R. einem Fehler, der zum Beispiel durch eine falsche Kalibrierung oder durch Produktionsungenauigkeiten ausgelöst wird. Dieser empfängerabhängige Fehler wird in der Literatur mit ca. 0.5 Metern angegeben [UBl09].

In die Klasse der kontextabhängigen Fehler (vgl. Abschnitt 2.3.3) fällt der sogenannte Mehrwegefehler. Dieser Fehler tritt auf, wenn das ursprüngliche Signal von natürlichen oder künstlichen Strukturen abgelenkt wird. Durch diese Ablenkung benötigt das Signal eine geringfügig längere Zeit bis es vom Empfänger detektiert werden kann und führt somit zu einer falschen Bestimmung der Entfernung zum Satelliten.
Dieser Effekt, der schematisch in Abbildung 2.7 dargestellt wurde, tritt vornehmlich in urbanen Gebieten oder im Gebirge auf und ist somit u.a. abhängig von der aktuellen Position des Empfängers. In maritimen Szenarien, z.B. offene See, kann dieser Fehler i.d.R. vernachlässigt werden.

Differential GPS Reicht die Genauigkeit von GPS nicht aus, lässt sich die Genauigkeit der Positionsbestimmung mit dem Differential GPS (DGPS) verbessern. Hierbei wird mit Hilfe einer sogenannten Referenzstation ein Korrekturterm berechnet und an geeignete DGPS - Empfänger verteilt.
Zu diesem Zweck muss die Position der Referenzstation sehr genau bekannt sein. Die Referenzstation bestimmt genau wie der einzelne GPS - Empfänger die eigene Position mit der zuvor beschriebenen Laufzeitmessung. Aus der Abweichung der bestimmten Position zu der zuvor vermessenen Referenzposition lässt sich der Korrekturterm ermitteln.
Mit Hilfe dieser Technik lassen sich mittlere Positionsgenauigkeiten von ca. 1 Meter erreichen. Weitere Optimierungen, die neben der Signallaufzeit auch die Phase der Trägerwelle des Funksignals bestimmen, erreichen eine Genauigkeit bis zu einem Zentimeter.
Das von der Referenzstation berechnete Korrektursignal besteht dabei im wesentlichen aus der Abweichung der bestimmten und der tatsächlichen[3] Entfernung, der einzelnen Satelliten.
Der errechnete Korrekturterm wird anschließend mit einer geeigneten Methode, an die DGPS - Empfänger übermittelt. Im deutschen maritimen Umfeld geschieht dies über Mittelwellen-Funk (283,5 - 315 kHz[4]).

Fehlerquellen Mit Hilfe des Korrektursignals der Referenzstation können die DGPS - Empfänger die von ihnen bestimmte Position korrigieren und erreichen i.d.R. eine sehr viel höhere Genauigkeit (vgl. Tabelle 6.1). Dies liegt insbesondere in der Tatsache begründet, dass mit Hilfe des Korrektursignals die atmosphärischen Effekte nahezu ausgeglichen werden können. Dabei wird davon ausgegangen, dass sich die atmosphärischen Effekte im gleichen Maße auf den DGPS - Empfänger als auch auf die Referenzstation auswirken.
Diese Annahme verliert mit zunehmender Entfernung des DGPS - Empfängers von der Referenzstation an Korrektheit. Untersuchungen die im Auftrag der IALA (International Association of Marine Aids to Navigation and Lighthouse Authorities) [Ais15] durchgeführt worden sind, geben eine Abnahme der Positionsbestimmungsgenauigkeit von 0.67m pro 100km Entfernung von der Referenzstation an. Neuere Untersuchungen, wie die von Monteiro et.al. [MMH05] vorgestellte Studie, kommen auf einen Wert von ca. 0.22m pro 100km.

[3] Auch hier handelt es sich um einen geschätzten Wert, der nach wie vor einer Ungenauigkeit unterliegt, die jedoch bedeutend geringer ist, als die Entfernungsbestimmung mittels Laufzeitmessung von Funkwellen.

[4] Angaben des WSV - Betreiber der nautischen DGPS-Referenzstationen. (`http://www.wsv.de/fvt/dgps/` - letzter Zugriff: 03.03.2016)

Die nachfolgende Tabelle 6.1 gibt eine Übersicht über die Werte der vorgestellten Fehler sowohl für das normale GPS-, als auch das korrigierte DGPS- Signal. In der entsprechenden Literatur ([Ran94]), werden die vorgestellten Fehler i.d.R. durch einen gaußverteilten Fehler modelliert, der um den Nullpunkt schwankt. Dementsprechend können die Werte der Tabelle 6.1 als Standardvarianz einer Gaußverteilung interpretiert werden.

Fehlerquelle	**GPS [m]**	**DGPS [m]**
Ephemeriden	1.5	0.1
Satelliten Uhr	1.5	0.1
Ionosphäre	3.0	0.2
Troposphäre	0.7	0.2
Mehrwege Effekte	1.0	1.4
Empfänger Ungenauigkeiten	0.5	0.5

Tabelle 6.1: Mittlere Fehlerwerte für GPS und DGPS Positionsbestimmung. Die angegebenen Werte basieren auf dem GPS - Compendium [UBl09, Table 17]

6.1.1.3 Simulierte Umgebung

Neben dem zu testenden System, ist die simulierte Umgebung in der die Sensordatenverarbeitung eingesetzt bzw. überprüft werden soll von Bedeutung. Dies gilt insbesondere für die Erzeugung von kontextsensitiven Fehlern, bei denen der Sensorkontext häufig auf Grundlage von Elementen aus der Umgebung bestimmt werden kann.

In diesem Beispiel soll ein autonomes Schiff entwickelt werden, welches selbstständig in Küstennähe operieren kann. Auf Grund der besonderen Gegebenheiten der Nordsee wie beispielsweise tiedenabhängige Wasserwege, werden besondere Anforderungen an die Qualität der Positionsbestimmung gestellt. Zur Bestimmung der eigenen Position, sowie der eigenen Ausrichtung des Fahrzeuges werden zwei GPS Sensoren eingesetzt, die jeweils am Bug und am Heck angebracht worden sind, wie in Abbildung 6.1 dargestellt.

Als *Referenzstation* für die DGPS Sensoren kann eine der drei verfügbaren, vom WSV[5] betriebenen Referenzstationen verwendet werden, die in Zeven (Reichweite = 225km), Helgoland (Reichweite = 285km) und Groß Mohrdorf (Reichweite = 285km) errichtet wurden (Siehe auch Abbildung 6.4).

[5]WSV = Wasser- und Schifffahrtsverwaltung des Bundes; `www.wsv.de`

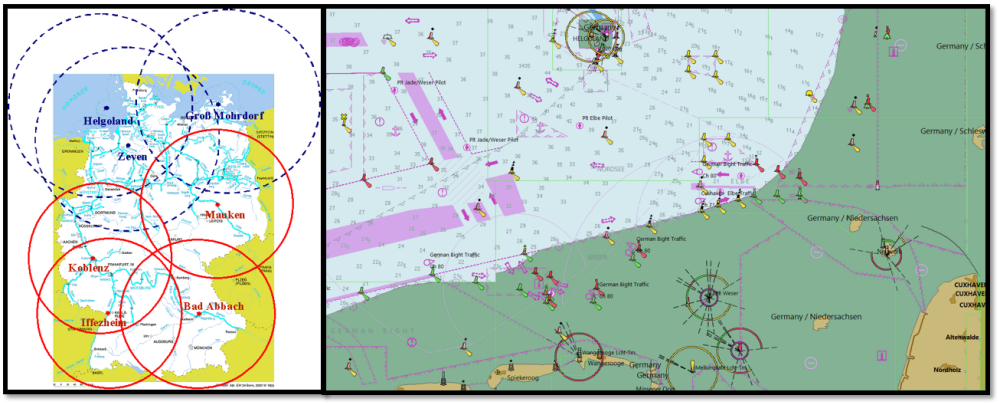

Abbildung 6.4: (Rechts) Einsatzszenario für die das sensordatenverarbeitende System: Die Deutsche Bucht; (Links) Ungefähre Abdeckung der vom WSV betriebenen DGPS Referenzstationen[6]

6.1.2 Sensormesswert Erzeugung

Dieser Teil der Evaluation soll zeigen, wie mithilfe der vorgestellten Methodik und unter Zuhilfenahme der in Abschnitt 6.1.1 gesammelten Informationen, Sensormesswerte für ein sensordatenverarbeitendes System erzeugt und verrauscht werden können.
Dazu folgt der Abschnitt dem in Kapitel 4.3 vorgestellten Vorgehen zur Sensormodellierung, bestehend aus der strukturellen Modellierung der Sensoren, der Beschreibung des normativen Verhaltens sowie das nachträgliche Anwenden von Fehlermodellen auf die idealen Sensormesswerte.
Eine Bewertung der Sensordatenverarbeitung hinsichtlich der Einhaltung der formulierten Qualitätskriterien erfolgt im nächsten Abschnitt.

Strukturelle Modellierung der Sensoren Für die zu simulierenden DGPS Sensoren wird das in Abbildung 6.5 dargestellte UML Klassendiagramm verwendet. Unterteilt

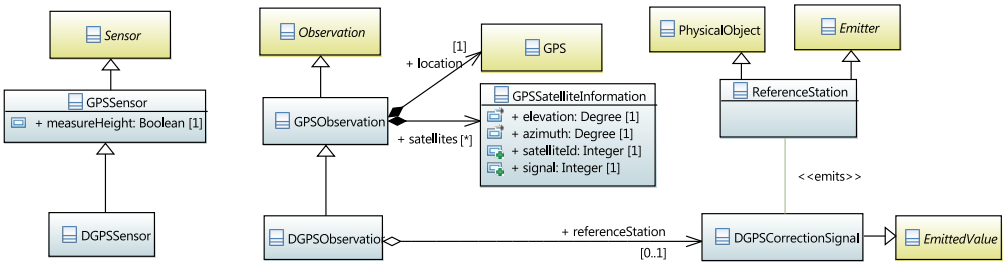

Abbildung 6.5: Modellierung von DGPS Sensoren, mit Konzepten aus dem *Sense* Teilmodell.

wird die Modellierung des Sensors zunächst in die Modellierung des allgemeineren GPS Sensors sowie die Spezialisierung DGPS Sensor.
Neben dem Sensor wird die entsprechende Beobachtung, die *GPSObservation*, beschrieben. Bei dem es sich um eine Spezialisierung des abstrakten Konzeptes Observation aus

Abschnitt 4.3.2.1 handelt.
Die strukturelle Beschreibung der *GPSObservation* wird im weiteren Verlauf der Evaluation dazu verwendet, die Sensorfehler und im speziellen die Messungenauigkeiten, auf einzelne Eigenschaften der Beobachtung anzuwenden.

Neben den Sensoren und den Sensorbeobachtungen enthält das Szenariomodell weitere, für die Sensordatengenerierung relevante Modelle. Zu diesem Zweck wird die DGPS Referenzstation (vgl. Abschnitt modelliert. Mit ihrer Hilfe kann im Abschnitt 6.1.3.2 der zuvor beschriebene, distanzabhängige Fehler modelliert werden.
Weiterhin greift das verwendete Szenariomodell auf Elemente des in Abschnitt 4.2.1 beschriebenen Datenmodells zurück, wie Beispielsweise einem Schiff, welches als Sensorplattform genutzt wird.

6.1.2.1 Messwertsynthese durch die komponentenbasierte Transformation aus dem Szenariomodell

Als erste Form der Sensormodellierung soll an dieser Stelle gezeigt werden, wie mithilfe des Komponentenmodells und der in Abschnitt 4.3.2 beschriebenen Komponentenmodell basierten Transformation von Informationen aus dem Szenariomodell, Sensormesswerte erzeugt werden können.
Dies geschieht durch die gezielte Weiterleitung und Umstrukturierung der entsprechenden Daten aus dem Szenariomodell.
Zum Zweck der GPS-Sensorsimulation wird innerhalb des gemeinsam genutzten Speicherbereiches eine neue Instanz einer *DGPSObservation* erstellt und als Datensenke in das Simulationsmodell übernommen (vgl. Abschnitt 4.3.2.2).
Als Datenquelle dient in diesem Fall die Position des jeweiligen GPS-Sensors, welche über eine direkte Verbindung mit der *location* Eigenschaft der *DGPSObservation* verbunden wird. Abbildung 6.6 (links) zeigt diese Verbindung innerhalb der Benutzeroberfläche des Sensorsimulationsframeworks, sowie die zugehörigen Szenariomodellelemente (gestrichelte Linien in Abbildung 6.6).

Anschließend wird eine Konvertierung der protokollunabhängigen GPS Beobachtung in das NMEA 0183 Protokoll durchgeführt. Zu diesem Zweck wurde eine spezielle Komponente entwickelt, welche aus der GPS Beobachtung einen entsprechenden GGA Datensatz erzeugt.
Die *GenerikSink* Komponenten kommunizieren die ermittelten Sensormesswerte über eine UDP Netzwerkverbindung an die in Abschnitt 6.1.1.1 vorgestellte Sensordatenverarbeitung, die ihrerseits die Position und Ausrichtung des Schiffes berechnet und in Form eines RMC Datensatzes zur Verfügung stellt.

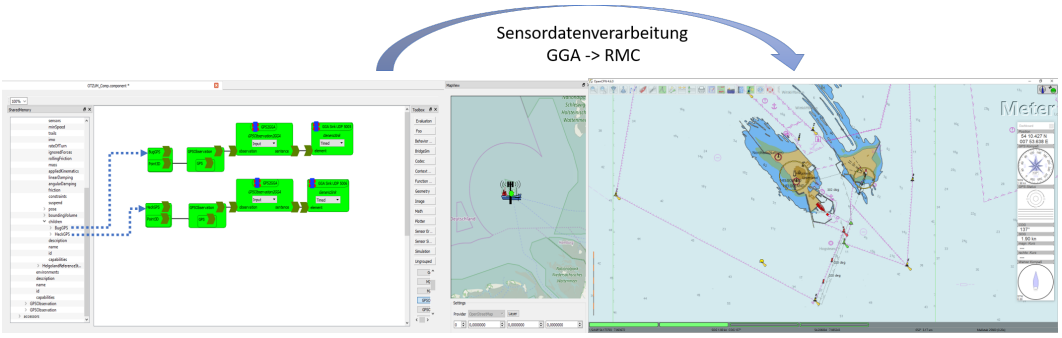

Abbildung 6.6: (links) Auszug aus der Benutzungsoberfläche bei der Elemente aus dem Szenariomodell bzw. dem gemeinsamen Speicherbereich direkt miteinander verbunden werden, sowie Anzeige des erkannten Umgebungsmodells der Sensordatenverarbeitung (rechts).

Da die Sensordatenverarbeitung über keine eigene Visualisierungskomponente verfügt, wurde für die Anzeige des erkannten Umgebungsmodells das Quelloffene Anzeigeprogramm OpenCPN[7] verwendet. Durch die Verwendung des externen Anzeigetools, zusammen mit dem in Abschnitt 6.3 beschriebenen, kommerziellen Brückensystem, kann somit auch die Erfüllung der Anforderung A7 (Nachbildung der realen Sensorschnittstellen) gezeigt werden.

Alternativ zu der hier vorgestellten Methode, die Sensormesswerte durch eine Transformation der Informationen aus dem Szenariomodell zu erzeugen, hätte auch eine spezialisierte GPS Simulationskomponente entwickelt und Implementiert werden können, wie es in Abschnitt 4.3.2.2 beschrieben wurde. Diese Technik wird im Abschnitt 6.3 für die Realisierung eines komplexen, maritimen Radarsensors beschrieben.

6.1.3 Modellierung von Sensorfehlverhalten

Dieser Teilabschnitt wird zeigen, wie die bisher fehlerfreien Sensormesswerte, mithilfe der in Kapitel 4 beschriebenen Methode, im Nachhinein mit Sensorfehlern und Messungenauigkeiten belegt werden können.
Weiterhin wird gezeigt, wie durch die Kombination von einfachen Fehlermodellen, komplexere kontextsensitive Fehlermodelle entwickelt werden.

Bei den Fehlermodellen handelt es sich um die in Abschnitt 2.3 vorgestellten und in Abschnitt 4.3.3 modellierten Fehlermodelle, die innerhalb des Simulationsmodells miteinander verknüpft und auf die Messwerte angewendet werden. Dabei liegt der Fokus dieses Abschnittes auf der Generierung von Messungenauigkeiten. Auf die Modellierung

[7] OpenCPN (`http://opencpn.de/`) ist ein quelloffener ENC-Plotter. Da es sich nicht um ein zertifiziertes Brückensystem handelt, kann er nur in der privaten Seefahrt verwendet werden.

von Sensorfehlern, wie beispielsweise dem Sensorausfall, wird im nächsten Teilabschnitt (vgl. 6.1.4.2) genauer eingegangen.

6.1.3.1 Modellierung eines normalverteilten Rauschens

Dieser Abschnitt beschäftigt sich mit der Demonstration der nachträglichen Anwendung von Fehlermodellen auf die generierten Sensormesswerte. Zu diesem Zweck wird zunächst ein einfaches weißes Rauschen mit einer Normalverteilung auf die generierten GPS Beobachtungen addiert.

Die Beschreibung des Fehlermodells geschieht durch eine Erweiterung des zuvor beschriebenen Szenariomodells. Wie im Abschnitt über das Sensormodell (4.3.2.1) beschrieben, können Fehlermodelle an die einzelnen *Ports* eines Sensors annotiert werden.
Zur Beschreibung des zufälligen Fehlers werden an dieser Stelle die beiden GPS Sensoren (am Bug und Heck aus Abbildung 6.1), bzw. deren *Ports*, im Szenariomodell, um einen statistischen Fehler (*StatisticError* in Abbildung 4.17) erweitert. Für beide Fehlermodelle wird als Zufallsverteilung die Normalverteilung ausgewählt. Als Mittelwert bzw. Standardabweichung werden die Werte: $\mu = 0.0$ bzw. $\sigma = 0.0001$ verwendet, was einer Abweichung von 0.0001 Dezimalgrad, bzw. 11,1 Meter[8], entspricht wenn die Zufallswerte auf eine WGS84 Position angewendet werden.
In einem weiteren Schritt muss für das Fehlermodell beschrieben werden, auf welche Attribute der Sensorbeobachtung die Fehler angewendet werden sollen. Im Falle einer GPS Beobachtung unterliegen die Werte von Latitude und Longitude einem ähnlichen Fehler, während der Wert von Altitude, sofern er simuliert wird, einem anderen Fehlermodell folgt. Um diese Zuordnung herzustellen, wird die in Abschnitt 4.3.3.1 beschriebene Fehlerkonfiguration verwendet und an das Fehlermodell angehängt. Dabei beziehen sich die in der Fehlerkonfiguration enthaltenen Attribute auf die Eigenschaften (im Sinne der *StructuralFeatures* aus dem Daten-Metamodell) der zu manipulierenden Beobachtung. Im Fall einer GPS Beobachtung handelt es sich konkret um die Eigenschaften “location.latitude“ bzw. “location.longitude“ der Klasse *GPSObservation* (vgl. Abbildung 6.5).

Abbildung 6.7 zeigt die Anwendung des zuvor modellierten weißen Rauschens auf die Ergebnisse einer GPS Beobachtung (rote Linie in dem unteren Diagramm) gegenüber den originalen Sensormesswerten (schwarze Linie).

[8] 0.0001 Dezimalgrad, in WGS84 Koordinaten, entsprechen ungefähr 11,1 Meter auf dem 53 Breitengrad.

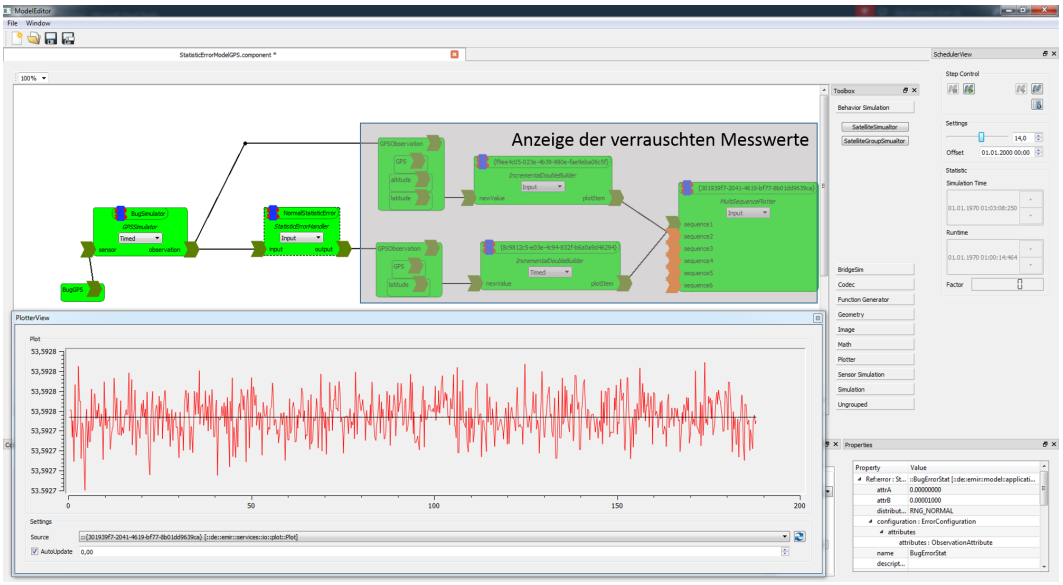

Abbildung 6.7: Darstellung eines zufälligen, normalverteilten Rauschens auf die Sensormesswerte einer GPS Beobachtung. Die schwarze Linie stellt die Ausgangsdaten (Breitengrad - Komponente der GPS Position) dar, während die rote Linie die verrauschten Messwerte repräsentiert. Die Komponenten in der grau hinterlegten Box werden zur Anzeige des originalen und verrauschten Messwertes im unteren Teil der Abbildung benötigt.

6.1.3.2 Kontextsensitive Fehlermodelle

Für die Modellierung komplexer Sensorfehler reichen einfache, zufällig generierte Messabweichungen i.d.R. nicht aus. Stattdessen werden zusätzliche Informationen benötigt, die sich auf die generierten Sensorfehler auswirken. Diese zusätzlichen Informationen werden in dem Abschnitt 2.3.3 (Kontextsensitive Fehlermodelle) unter dem Begriff Sensorkontext zusammengefasst, bzw. das sich daraus ergebene Fehlermodell als kontextsensitiver Fehler.

Bei der Vorstellung der zu testenden Sensordatenverarbeitung in Abschnitt 6.1.1.2 wurden zwei solcher kontextsensitiven Fehler vorgestellt, zum Einen der Mehrwegefehler, wie man ihn vornehmlich in urbanen oder gebirgigen Gegenden antrifft und die Abhängigkeit der Differential GPS Messgenauigkeit von der Entfernung zwischen dem Sensor und der verwendeten Referenzstation.

In diesem Beispiel soll die Fähigkeit des Sensorsimulationsframeworks demonstriert werden, während der Laufzeit der Simulation den aktuellen Kontext eines Sensors zu bestimmen und ihn in der Berechnung der Messungenauigkeiten zu berücksichtigen. Als Demonstrationsbeispiel dient in diesem Fall die entfernungsabhängige Messungenauigkeit bei DGPS Sensoren. Dazu wird in der Literatur ([MMH05]) der folgende Zusammenhang zwischen der Entfernung (d) des Sensors von der Referenzstation und der aus

ihr bedingten Abweichung bei der Positionsbestimmung angegeben (vgl. Gleichung 6.1).

$$\sigma = A[m] * \frac{d[m]}{100[km]} \tag{6.1}$$

Wobei sich die Werte von A je nach Quelle unterscheiden und mit $A = 067[m]$ ([Ais15]) bzw. $A = 0.22[m]$ ([MMH05]) angegeben werden.
Hinzu kommt die Abhängigkeit, dass sich der Sensor innerhalb des Einflussgebietes der Referenzstation befinden muss, um z.B. per Funkverbindung das Korrektursignal zu empfangen.

Für die Demonstration wird das Szenario aus dem Beispiel um eine weitere Differential-GPS Referenzstation erweitert, um den Effekt der dynamischen Bestimmung des Sensorkontextes zu zeigen. Das modifizierte Szenario ist in Abbildung 6.8 dargestellt. In

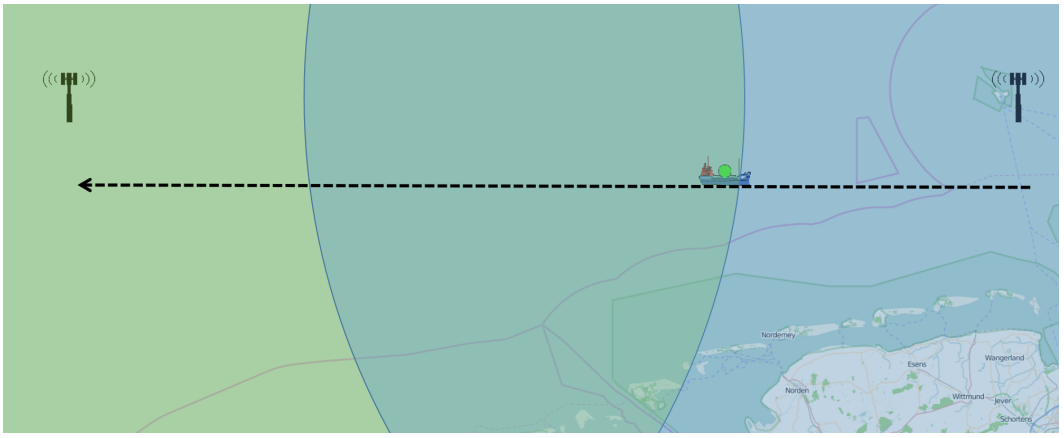

Abbildung 6.8: Modifiziertes Szenario zur Demonstration des kontextsensitiven Fehlers; Geographische Lage der Referenzstation und Fahrtrichtung des Schiffes

der Abbildung repräsentieren die beiden Masten die jeweiligen Referenzstationen. Die rechte Referenzstation repräsentiert die bereits modellierte Referenzstation auf Helgoland, während es sich bei der linken Station um eine zu Demonstrationszwecken erdachte Station handelt. Beide Stationen liegen auf dem gleichen Breitengrad und haben einen ungefähren Abstand von 300 km. Das simulierte Schiff startet in der Nähe von Helgoland und fährt mit einem geraden Kurs Richtung Westen (angedeutet durch den Schwarzen Pfeil). Während der gesamten Fahrt befindet es sich immer in Reichweite von mindestens einer Referenzstation. Die Einflussgebiete der Referenzstation (vgl. Abschnitt 4.3.1) werden durch die blauen und grünen Kreisausschnitte in Abbildung 6.8 angedeutet. Beide Gebiete werden in Form einer Kugel mit einem Radius von 285km modelliert. Dabei handelt es sich um eine Näherung die so in der Realität nicht zu erwarten ist. Anstelle einer Kugel könnte auch eine beliebige, ggf. dynamisch veränderliche, Geometrie verwendet werden die Beispielsweise von einer externen “Signalstärke Simulation“ bereitgestellt wird.

Die Bestimmung des Sensorkontextes besteht in diesem Beispiel aus der Detektion des vom DGPS-Sensor verwendeten Korrektursignals, sowie der Bestimmung der Entfernung der emittierenden Referenzstation. Zu diesem Zweck stellt das Simulationsframework eine generische Suchkomponente zur Verfügung, welche aus einem beliebigen Szenariomodell alle Instanzen einer Klasse aus dem Datenmodell bereitstellen kann. Diese sogenannte *InstanceCollector* Komponente kann zusätzlich eine optionale Geometrie (z.B. das beobachtete Gebiet eines Sensors) und eine optionales physikalisches Objekt, wie beispielsweise den betrachteten Sensor über einen Dateneingang beziehen. Ist der Geometrieeingang belegt, beschränken dieser die Ausgabe der Komponente auf diejenigen Instanzen, die sich innerhalb der anliegenden Geometrie befinden. Liegt neben der Geometrie auch ein physikalisches Objekt an dem entsprechenden Eingang an, wird dessen absolute Pose als Zentrum der angegebenen Geometrie betrachtet und diese entsprechend transformiert.
Neben dem *InstanceCollector* wird eine Auswahlkomponente (*DistanceSelector*) bereitgestellt, welche aus einer Liste von Objekten dasjenige ausfindig macht, dessen Entfernung zu einem Referenzobjekt am Größten bzw. am Geringsten ist.

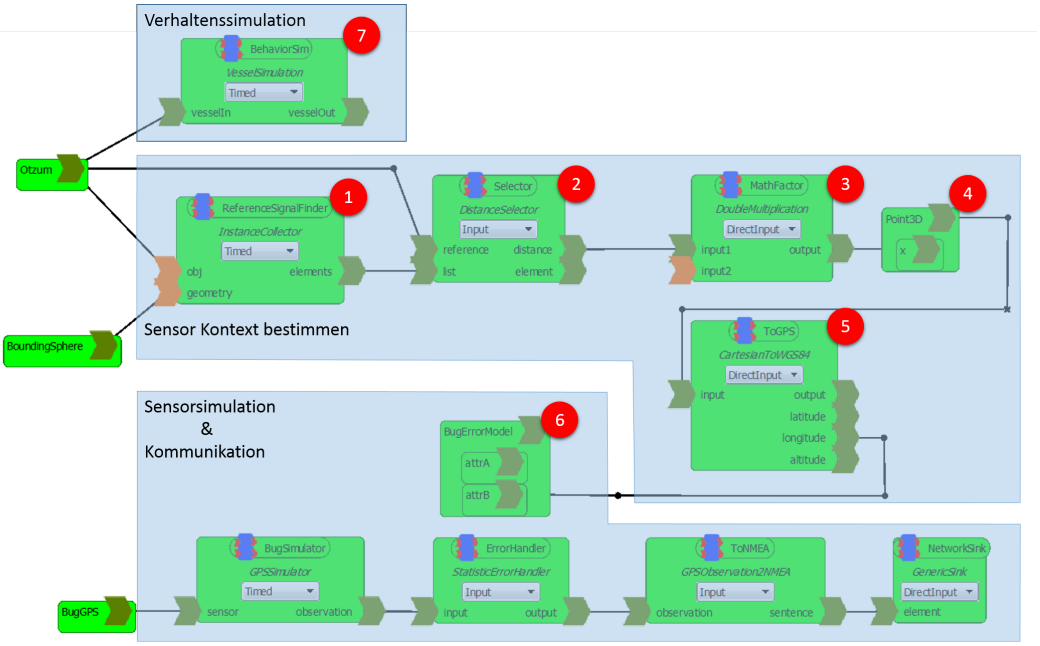

Abbildung 6.9: Simulationsmodell zur Beschreibung eines kontextsensitiven Fehlers, mit Einfluss auf die Standardabweichung bei einem statistischen Fehlermodell

Mithilfe dieser beiden Komponenten lässt sich der beschriebene Kontext für die Berechnung des entfernungsabhängigen DGPS Fehlers bestimmen.
In Abbildung 6.9 sind sie mit den Zahlen 1 bzw. 2 markiert. Bei der vom Selektionsknoten ausgegebenen Entfernung handelt es sich um die Entfernung der beiden Objekte in

Metern. Dieser wird von der Multiplitkationskomponente (Nr. 3) gemäß der oben angegebenen Gleichung (Gl. 6.1), auf die Abweichung der Position in Metern umgerechnet. Für die Umrechnung in die vom Fehlermodell benötigten Dezimalgrad, wird die Komponente mit der Nummer 5 verwendet, bei der es sich um Koordinatensystem Transformation von einem kartesischen Koordinatenreferenzsystem in das WGS84 Koordinatenreferenzsystem, handelt. Der dafür benötigte Punkt (Nr. 4) wird im gemeinsamen Speicherbereich vorgehalten. Die nach der Transformation anliegende GPS Koordinate entspricht der Standartabweichung des anzuwendenden zufälligen Fehlers (Nr. 6 in Abbildung 6.9).

Die anschließende Simulation der Sensormesswerte, das Anwenden des zufälligen Fehlers auf die idealen Sensormesswerte und die Kommunikation der Sensormesswerte an das sensordatenverarbeitende System, folgt dem gleichen Schema wie es bereits in dem vorhergegangenen Beispiel beschrieben wurde.

Im Gegensatz zu den bisherigen Beispielen, müssen sich in diesem Szenario die Sensoren bewegen, bzw. es wird ein einfaches Verhalten des simulierten Schiffes benötigt. Dieses einfache Verhalten wird durch den Knoten mit der Nummer 7 (in Abbildung 6.9) simuliert, bei dem es sich um eine einfache Interpolation der Schiffsposition entlang eines vorgegebenen Pfades handelt (gestrichelte Linie in Abbildung 6.8). Durch die Änderung der Schiffsposition ändert, aufgrund der relativen Positionierung gegenüber dem Schiff, auch die Position der Sensoren.

Die Abbildung 6.10 zeigt das Ergebnis der simulierten Sensormesswerte, aufgeschlüsselt nach Längen- und Breitengrad des Schiffes. Dabei ist zu beachten, dass zur besseren Darstellung der in der Literatur angegebene Fehler von 0,67m pro 100km (vgl. Abschnitt 6.1.1.2) um den Faktor fünf verstärkt simuliert worden ist. Umgerechnet auf die Änderungen im WGS84 Koordinatensystem, ergibt sich dadurch ein Rauschen mit Amplituden zwischen $8,7 * 10^{-6}$ und $5,1 * 10^{-5}$.
Diese Amplituden erklären auch den linearen Verlauf der Längengradänderung in Abbildung 6.10, da sie aufgrund ihrer geringen Werte nicht ins Gewicht fallen.

Abbildung 6.11 zeigt den generierten Fehler für den Breitengrad, bereinigt um die tatsächliche Position des Sensors gegenüber der dynamisch ermittelten Standardabweichung (roter Verlauf in Abbildung 6.11).
Gut zu erkennen in dieser Abbildung ist die Ermittlung des Sensorkontextes, welche in diesem Beispiel aus der Entfernung des Sensors zur Referenzstation bestand. Wie bei dem simulierten Szenario zu erwarten ist, handelt es sich zunächst bei der DGPS Referenzstation auf Helgoland um die nächste und damit stärkste Referenzstation, deren Korrektursignal verwendet wird. Mit steigender Entfernung zu dieser Station steigt auch die Positionsungenauigkeit, wie es in Abschnitt 6.1.1.2 beschrieben wurde. Ungefähr auf

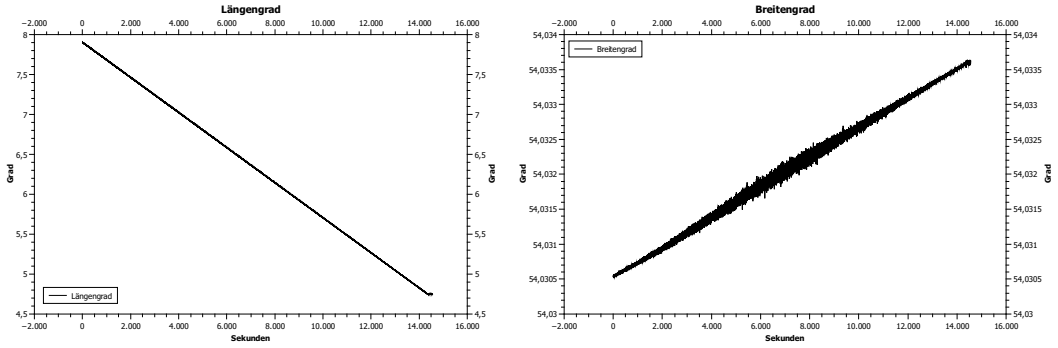

Abbildung 6.10: Positionsänderungen der Sensoren aufgetragen über die Zeit

der Hälfte der simulierten Strecke stellt die zusätzlich hinzugefügte Referenzstation, das stärkere Signal zur Verfügung und wird stattdessen für die Bestimmung der Positionskorrektur verwendet. Ab diesem Zeitpunkt nimmt die Genauigkeit der Positionsbestimmung wieder zu.

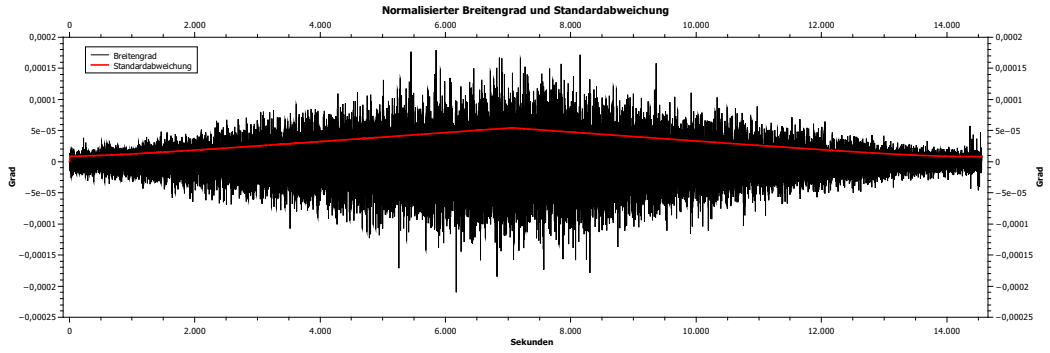

Abbildung 6.11: Darstellung des kontextsensitiven Fehlers gegenüber dem dynamisch ermittelten Sensorkontext

6.1.4 Bewertung der Sensordatenverarbeitung

Der vierte Teil der Evaluierung einzelner Konzepte beschäftigt sich mit der Detektion des Fehlverhaltens der Sensordatenverarbeitung in Bezug auf die formulierten Qualitätskriterien und ist damit der zentrale Bestandteil bei der Bewertung eines sensordatenverarbeitenden Systems.
Es wird gezeigt wie, mithilfe der vorgestellten Modellierungsmöglichkeiten die Qualitätskriterien beschrieben, und eine Verletzung der Kriterien detektiert werden kann.

An dieser Stelle wird der in Abschnitt 6.1.1.1 beschriebenen Fehler, der auf die Ergebnisse der Sensordatenverarbeitung addiert wird verwendet um ein Fehlverhalten des zu testenden Systems zu erzeugen. Dies hat den Vorteil, dass der genaue Zeitpunkt sowie die Art des Fehlers bekannt ist und gleichzeitig die zeitnahe Detektion des Fehlers gezeigt werden kann.

6.1.4.1 Bewertung der Korrektheit der Sensordatenverarbeitung

Im Abschnitt 4.3.3.3, wurde neben der Modellierung von Messungenauigkeiten beschrieben, wie diese mithilfe von Komponenten zur Distanzermittlung erkannt werden können. Abbildung 6.12 zeigt einen Simulationsgraphen, mit dem sich die in Abschnitt 6.1.1 beschriebene Qualitätsanforderung nach einer Positionsgenauigkeit von unter 1 Metern überprüft werden kann. Der Aufbau des Simulationsgraphen setzt sich dabei aus den

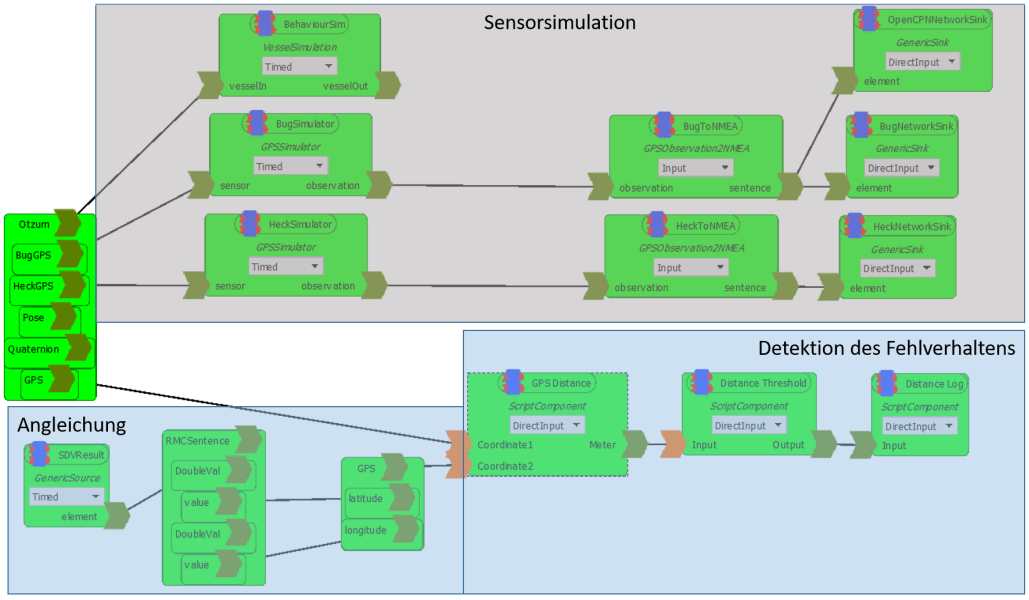

Abbildung 6.12: Darstellung des kontextsensitiven Fehlers gegenüber dem dynamisch ermittelten Sensorkontext

drei Teilen *Sensorsimulation* zur Erzeugung der Messwerte sowie deren Weiterleitung an die Sensordatenverarbeitung, *Angleichung* der Ergebnisse des sensordatenverarbeitenden Systems und der Detektion des Verstoßes gegen die Qualitätsanforderung bzw. der *Detektion des Fehlverhaltens* der Sensordatenverarbeitung zusammen.
Dabei wird an dieser Stelle auf die Simulation eines Sensorfehlers verzichtet (vgl. oberer Abschnitt aus Abbildung 6.12) um den Zeitpunkt des Fehlers manuell festlegen zu können.

Für die Bewertung wird das von der Sensordatenverarbeitung erkannte Umgebungsmodell durch die Komponente *GenericSource* abgegriffen, welche Analog zur Komponente *GenericSink* arbeitet.
Die für die Distanzbestimmung benötigte GPS Position wird anschließend aus dem von der Sensordatenverarbeitung gelieferten RMC Datensatz extrahiert. Die Berechnung der Distanz zwischen der tatsächlichen Position des simulierten Schiffes sowie der ermittelten Position erfolgt mit der Vincenty Formel.
Überschreitet die Distanz einen Schwellwert von 1 Meter, wird eine Log Ausgabe inkl.

Zeitstempel generiert, welche mit dem Zeitpunkt der Fehlerinjektion verglichen werden kann.

Während der Durchführung der Simulation wurde der Fehler in der Sensordatenverarbeitung zweimal aktiviert bzw. deaktiviert. Die Folgende Abbildung 6.13 zeigt eine Gegenüberstellung der von den beteiligten System erstellten Logs, für die entsprechenden Zeiträume. Zudem zeigt die in Abbildung 6.13 dargestellte Gegenüberstellung der

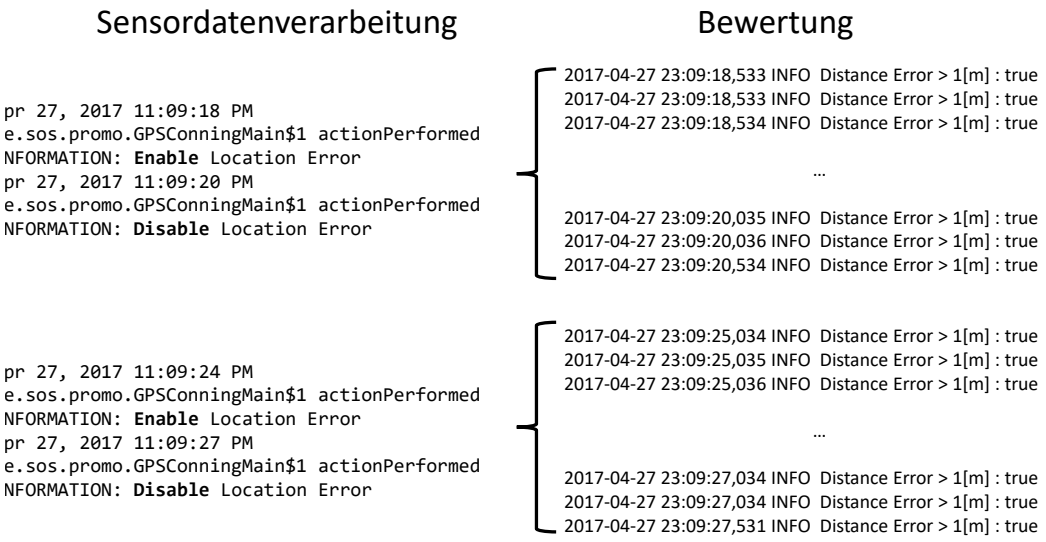

Abbildung 6.13: Gegenüberstellung der Logausgaben der Sensordatenverarbeitung (links) und der simulativen Bewertung (rechts)

Zeiten, dass die Erkennung des Fehlers in der Sensordatenverarbeitung zeitnah erkannt werden kann.

6.1.4.2 Ermittlung der Robustheit gegenüber Messungenauigkeiten

Als ein weiteres Qualitätskriterium für sensordatenverarbeitende Systeme wurde in Abschnitt 1.2.1 die Robustheit gegenüber Messungenauigkeiten eingeführt.
Das folgende Experiment demonstriert eine Methode, wie mithilfe des entwickelten Ansatzes die Robustheit gegenüber Messungenauigkeiten überprüft werden kann und skizziert, wie diese Überprüfung zur manuellen Ermittlung der maximalen Messabweichung genutzt werden kann. Für das Experiment wird der Simulationsgraph aus Abbildung 6.12 um statistische Fehler für die beiden GPS Sensoren ergänzt.
Beide Fehlermodelle werden jeweils durch einen normalverteilten statistischen Fehler beschrieben. Weiterhin wird anstelle der Positionsgenauigkeit die Genauigkeit der Ausrichtungsbestimmung überprüft.
Als Überprüfungsmethode wird in diesem Beispiel eine optische Gegenüberstellung der

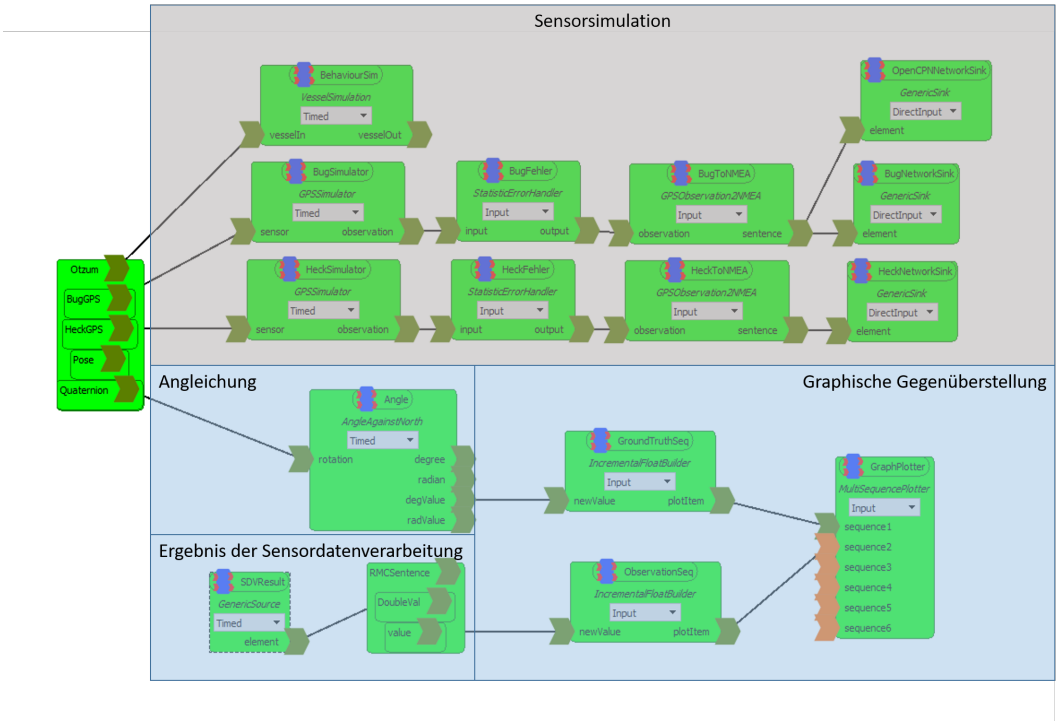

Abbildung 6.14: Simulationsmodell zur (optischen) Bewertung eines GPS Kompasses.

Messergebnisse gegenüber dem Erwartungswert gewählt und das Experiment mit verschiedenen Fehlerstärken wiederholt.
Abbildung 6.14 zeigt das entsprechend modifizierte Simulationsmodell, wohingegen Abbildung 6.15 das Ergebnis eines Simulationslaufes darstellt. Der in Abbildung 6.15 darge-

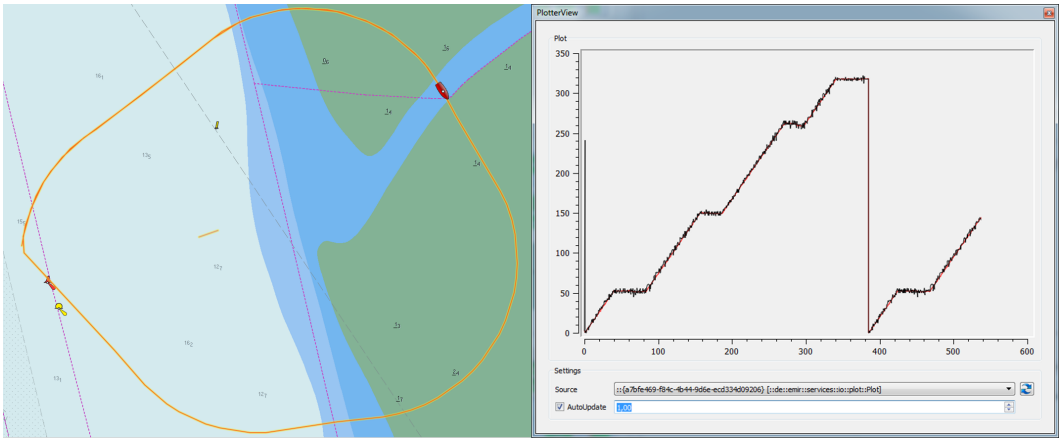

Abbildung 6.15: Gegenüberstellung von realer Ausrichtung (rote Linie) und ermittelter GPS-Kompass Ausrichtung (schwarze Linie); Links: Gefahrene Strecke

stellte Verlauf stellt einen Simulationslauf dar, bei dem keine Fehler auf die Sensorwerte addiert worden sind (Mittelwert und Standardabweichung der statistischen Fehler wurden auf 0 gesetzt).
Wie in der Abbildung zu erkennen ist, stimmt die von der Sensordatenverarbeitung ermittelte Ausrichtung weitestgehend mit der realen Ausrichtung überein. Lediglich ein

leichtes Rauschen um den realen Wert kann festgestellt werden. Diese lässt sich mit der Diskretisierung durch die zur Kommunikation verwendeten Protokolle erklären.

Ermittlung des maximal erlaubten Fehlers Um manuell den maximalen Fehler zu ermitteln, mit dem das zu testende System die formulierten Qualitätsanforderungen noch erfüllt, kann das Experiment mit verschiedenen Konfigurationen der verwendeten Fehlermodelle wiederholt werden.
In diesem Beispiel wurden die folgenden Standardabweichungen für die Fehlermodelle verwendet:

- $\sigma = 0.00000002[^\circ] \approx 0,001[m]$
- $\sigma = 0.00000020[^\circ] \approx 0,013[m]$
- $\sigma = 0.00000200[^\circ] \approx 0,130[m]$
- $\sigma = 0.00002000[^\circ] \approx 1,302[m]$
- $\sigma = 0.00020000[^\circ] \approx 13,026[m]$

Die Ergebnisse der verschiedenen Simulationsläufe ist in Abbildung 6.16 dargestellt.

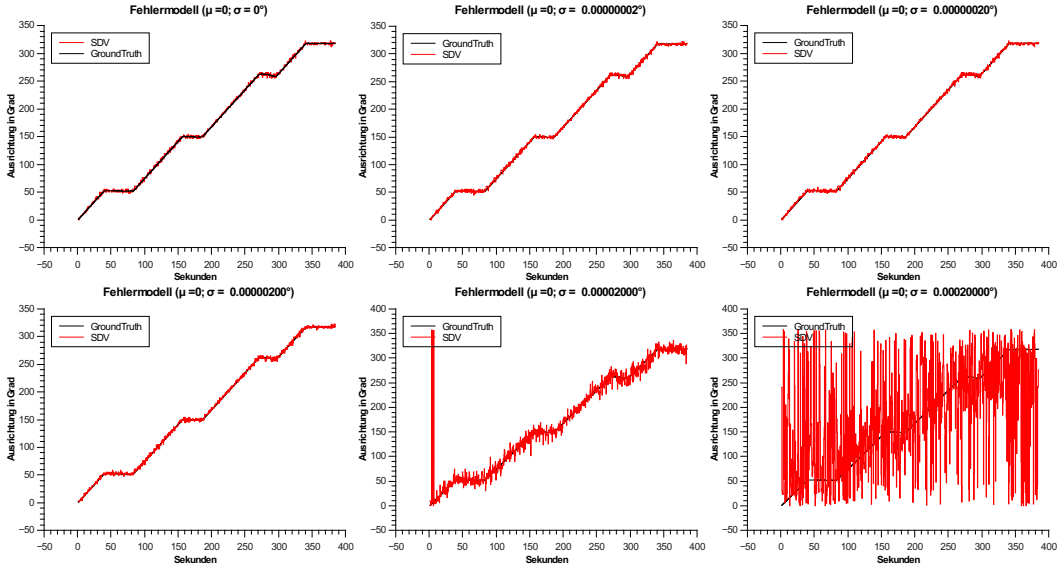

Abbildung 6.16: Visuelle Gegenüberstellung von realer und ermittellter Ausrichtung des Fahrzeuges, bei unterschiedlichen Fehlerstärken

6.1.4.3 Detektion von Sensorfehlern

Zur Demonstration der Überprüfung hinsichtlich des Qualitätskriteriums Fehlertoleranz, wird in dem folgenden Beispiel der Ausfall eines GPS Sensors simuliert. Zu diesem Zweck

werden die statistischen Fehlermodelle aus Abbildung 6.14 durch eine *MalFunction* Fehlerkomponente ersetzt. Diese kann entweder durch die Übergangswahrscheinlichkeiten (P_{aktiv} und $P_{inaktiv}$) oder durch ein manuelles setzten des Zustandes aktiviert werden. Für das folgende Experiment werden beide Übergangswahrscheinlichkeiten mit 0% angegeben, so dass eine manuelle Aktivierung des gewünschten Zustandes erfolgen muss.

Da es sich bei der in Abschnitt 6.1.1.1 beschriebenen Sensordatenverarbeitung um eine sehr einfache, datenstromaktivierte Realisierung einer PNT Unit handelt wird davon ausgegangen, dass der Ausfall eines Sensors auch zum Ausfall der Sensordatenverarbeitung führt.
Dementsprechend wird für die Überprüfung der Fehlertoleranz eine *Watchdog* Komponente verwendet, wie sie in Abschnitt 4.3.3.2 beschrieben wurde. Abhängig von der Updatefrequenz der Sensoren von 0.5 Sekunden, wird der Watchdog so konfiguriert, dass er mit einer Frequenz von 1Hz überprüft ob neue Sensordaten empfangen wurden.

Die Abbildung 6.17 zeigt die vom Simulationssystem erzeugten Log Ausgaben, in denen sowohl der Zeitpunkt der Aktivierung des Fehlers, als auch der Zeitpunkt der Detektion angegeben ist. Die in Abbildung 6.17 dargestellte Auswertung der Sensordatenverarbei-

```
2017-04-28 00:57:49,673 INFO  switch activation to active=true
2017-04-28 00:57:51,226 INFO  Missing Response from SDV : true
2017-04-28 00:57:52,227 INFO  Missing Response from SDV : true
2017-04-28 00:57:53,226 INFO  Missing Response from SDV : true
2017-04-28 00:57:53,631 INFO  switch activation to active=false
2017-04-28 00:57:59,113 INFO  switch activation to active=true
2017-04-28 00:58:00,226 INFO  Missing Response from SDV : true
2017-04-28 00:58:01,226 INFO  Missing Response from SDV : true
2017-04-28 00:58:02,226 INFO  Missing Response from SDV : true
2017-04-28 00:58:02,437 INFO  switch activation to active=false
```

Abbildung 6.17: Logausgaben des Simulations- und Bewertungssystems bei der Simulation eines Sensorausfalls.

tung bestätigt die Vermutung, dass es sich bei dem zu testenden System nicht um ein fehlertolerantes System handelt.
Auffällig ist zudem, dass der Zeitraum zwischen der Aktivierung des Fehlers und dessen Detektion mehr als die konfigurierte Sekunde vergangen ist. Stattdessen muss, im Sinne einer worst-case Abschätzung, davon ausgegangen werden, dass sich die Zeit bis zur Detektion um die Updatefrequenz des fehlerhaften Sensors und ggf. die Zeit, die für die Kommunikation der Sensordaten benötigt wird, verlängert.

6.2 Bewertung der PNT Unit

Nachdem im vorangegangenen Abschnitt die einzelnen Bestandteile der Bewertung von sensordatenverarbeitenden Systemen an verschiedenen kleinen Beispielen gezeigt wurde, fasst dieser Abschnitt die Bausteine zusammen um die in Abschnitt 6.1.1.1 vorgestellte Sensordatenverarbeitung zu bewerten.
Diese Bewertung besteht im wesentlichen aus der Auswertung der PNT Unit in Bezug auf ihre Korrektheit, sowie ihr Verhalten gegenüber Messunsicherheiten. Hierfür wurden in Abschnitt 6.1.1.1 die Bewertungskriterien festgelegt:

- Maximale Abweichung der Position = 1 Meter
- Maximale Abweichung der Ausrichtung = 0.5°

Für beide Werte gilt eine geforderte Verfügbarkeit von 95%.
Auf eine Überprüfung der Fehlertoleranz kann gemäß des Abschnittes 6.1.4.3 verzichtet werden, in dem sich herausgestellt hat, dass die vorliegende Umsetzung der PNT Unit keine Methoden zur Fehlerbehebung unterstützt.

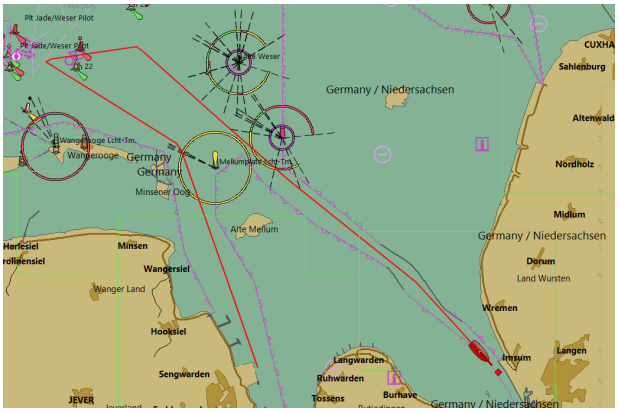

Abbildung 6.18: Route des autonomen Schiffes

Experiment Durchführung Während der Durchführung des Experiments fährt das simulierte Schiff die in Abbildung 6.18 dargestellte Route ab. Dabei handelt es sich um eine Strecke von ungefähr 50 nautischen Meilen, für die das Schiff ungefähr 3,3 Stunden benötigt, daraus folgt eine Durchschnittsgeschwindigkeit von 15 Knoten.
Gemessen wird die Abweichung der Ausrichtung des Schiffes, wofür die Sensordatenverarbeitung des Schiffes das *"Track Made Good"* Feld des RMC Datensatzes verwendet, sowie der Abstand der ermittelten Position von der tatsächlichen Position des Schiffes.
Für beide Datensätze werden jeweils die Verstöße gegen die oben genannten Qualitätskriterien gezählt, sowie die ermittelten Distanzen für eine offline Analyse, in eine CSV Datei geschrieben.

6.2.1 Naive Bewertung

Die Generierung der Sensormesswerte für das sensordatenverarbeitende System erfolgt analog zu dem in Abbildung 6.14 dargestellten Simulationsmodell. Das Bewertungsmo-

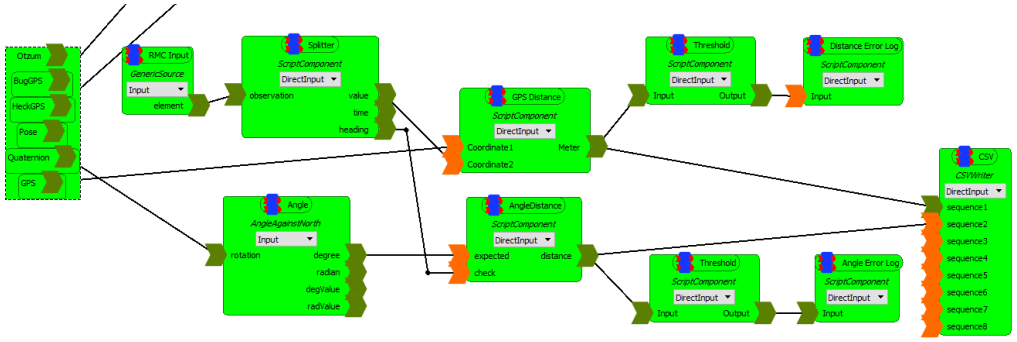

Abbildung 6.19: Naives Simulationsmodell zur Bewertung der Sensordatenverarbeitung

dell ist in Abbildung 6.20 dargestellt. Wie in der Abbildung gesehen werden kann, wird die ermittelte Position direkt mit der aktuellen Position des Schiffes verglichen. Das Ergebnis eines ersten Simulationslaufes ist in Abbildung 6.20 dargestellt. Dabei wurden noch keine Fehlermodelle auf die generierten Sensormesswerte angewendet.

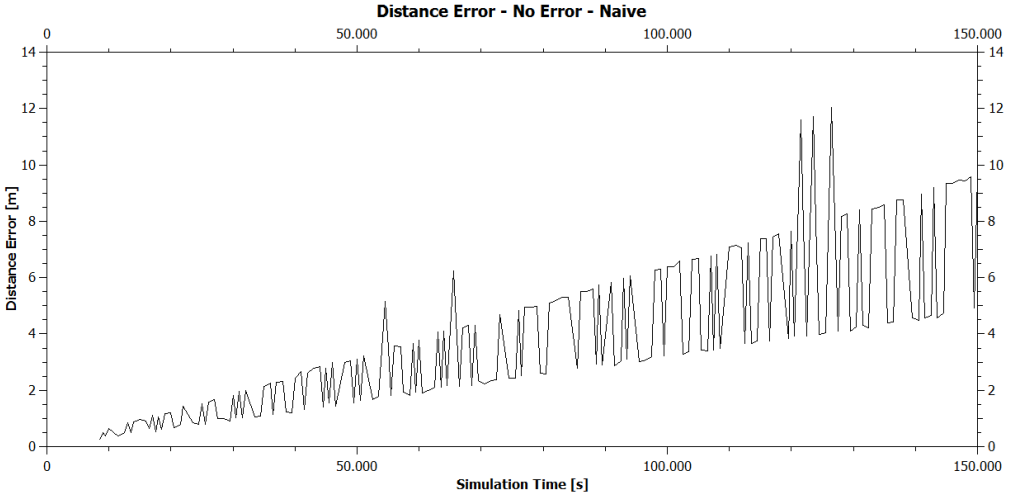

Abbildung 6.20: Abweichung der ermittelten Position von der aktuellen Position des Schiffes

An diesem verhältnismäßig kurzen Abschnitt von 150 Sekunden lassen sich bereits zwei Rückschlüsse auf das sensordatenverarbeitende System machen.

1. Beachtet man, dass das Schiff in dieser Zeit aus dem Stand auf eine Geschwindigkeit von ca. 15 Knoten beschleunigt wird deutlich, dass entweder die Sensordatenverarbeitung oder die Bewertung fälschlicherweise von einer statischen Welt

(vgl. Abschnitt 4.3.5.1) ausgeht, d.h. die Zeit, die für Kommunikation und die Berechnung des Umgebungsmodells benötigt wird, wird nicht berücksichtigt.

2. Die unregelmäßigen Abstände zwischen den Messungen, deuten darauf hin, dass die Sensordatenverarbeitung eingehende Daten ignoriert, wenn nicht beide Messwerte zeitnah empfangen werden. Diese Beobachtung deckt sich mit der in Abschnitt 6.1.4.3 gewonnen Erkenntnis, dass die untersuchte PNT Unit keine Daten liefert, sofern ein Sensor ausgefallen ist.

Im Folgenden wird davon ausgegangen, dass es das Ziel des sensordatenverarbeitenden Systems ist, die Position und Ausrichtung zu dem Zeitpunkt zu bestimmen, der zu den eingegangenen Daten passt.
Dafür spricht, dass sowohl der GGA als auch der RMC Datensatz aus dem NMEA 0183 Protokoll über einen Zeitstempel verfügen.

6.2.2 Zeitabhängige Bewertung

Um die Zeit bei der Bewertung berücksichtigen zu können, wird der in Abschnitt 4.3.5.1 beschriebene Speichermechanismus verwendet. D.h die Simulation speichert die tatsächliche Position und die Ausrichtung des Schiffes im Abstand von 0.1 Sekunden[9]. Für die Distanzberechnungen wird dementsprechend die Position und Ausrichtung des Schiffes zu dem im Ergebnis der Sensordatenverarbeitung angegebenen Zeitpunkt herangezogen. Das veränderte Bewertungsmodell ist in Abbildung 6.21 dargestellt.

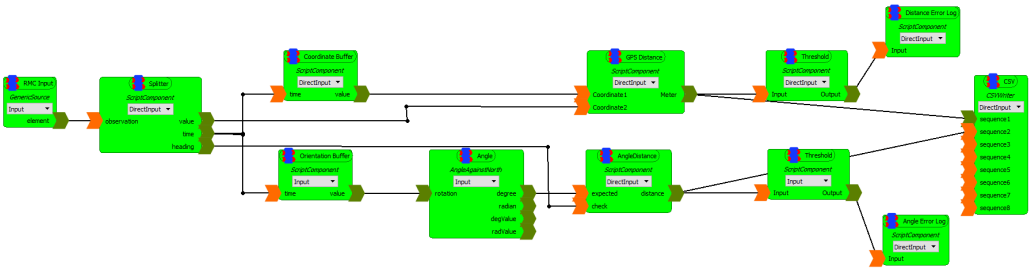

Abbildung 6.21: Simulationsmodell zur Bewertung der Sensordatenverarbeitung

Als Referenzwert wird zunächst eine Bewertung der Sensordatenverarbeitung durchgeführt, bei der keine Fehler auf die generierten Messwerte angewendet werden.
Dies folgt dem Zweck, einen generellen Nachweis der Funktionalität der Sensordatenverarbeitung unter idealisierten Bedingungen durchzuführen. Abbildung 6.22 zeigt das Ergebnis der Bewertung, für das oben beschriebene Szenario. Die statistische Auswertung der Bewertung ergibt, dass im Mittel beide Bewertungskriterien erfüllt werden (vgl.

[9]Dies entspricht der Update rate der Positionsänderungen des Schiffes

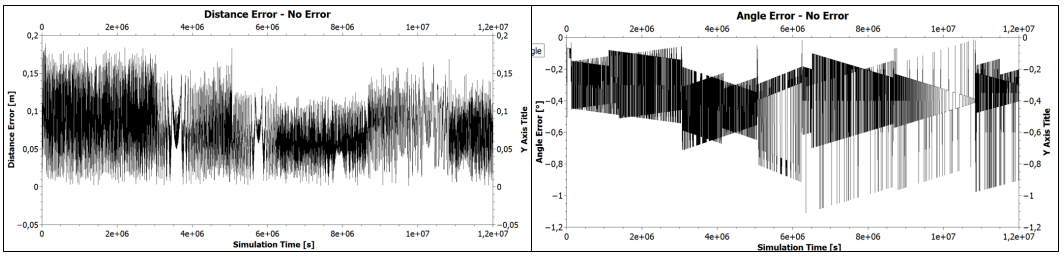

Abbildung 6.22: Bewertung der PNT Unit ohne Anwendung von Fehlermodellen

Tabelle 6.2: Statistische Auswertung der Analyse ohne Fehlermodelle

	Mittelwert	**Varianz**	**Max**	**Median**	**Verfügbarkeit**
Distanz	0.075	0.001	0.189	0.073	99%
Winkel	-0.34	0.037	-1.10	-0.28	

Tabelle 6.2). Für die Positionsbestimmung, wird diese Bewertung durch die graphische Auswertung unterlegt.
Im Falle der Ausrichtungsbestimmung zeigt die graphische Auswertung, dass die geforderte Genauigkeit von unter 0.5° nicht eingehalten werden kann.
Eine Erklärung für die Bewertungsergebnisse liefern die verwendeten Kommunikationsprotokolle, welche die WGS84 Positionen der Sensoren nur mit eingeschränkter Genauigkeit übertragen. Dies gilt auch für die Kommunikation der Ergebnisse der Sensordatenverarbeitung.

Fehlermodelle Für die Analyse der PNT Unit werden zwei unterschiedliche Simulationsmodelle angewendet, welche wiederum verschiedene Fehlermodelle repräsentieren. Beide Simulationsmodelle werden zudem mit unterschiedlichen Konfigurationen bewertet.

Die erste Konfiguration entspricht der Simulationmodell aus Abbildung 6.14 und beinhaltet je eine zufällige Fehlerkomponente pro simulierten GPS-Sensor.
Die simulative Bewertung der PNT Unit wird mit den folgenden drei Konfigurationen für die Fehlerkomponenten durchgeführt.

Gauß 0.25 Normalverteilung mit Mittelwert = 0 Meter und Standardabweichung = 0.25 Meter

Gauß 0.5 Normalverteilung mit Mittelwert = 0 Meter und Standardabweichung = 0.5 Meter

Gauß 1 Normalverteilung mit Mittelwert = 0 Meter und Standardabweichung = 1 Meter

Tabelle 6.3: Statistische Auswertung mit jeweils einem Fehlermodell pro Sensor

	Mittelwert	**Varianz**	**Max**	**Median**	**Verfügbarkeit**
	Gauß 0.25				
Distanz	0.328	0.051	4.395	0.305	99%
Winkel	-0.33	0.03	-1.13	-0.28	
	Gauß 0.5				
Distanz	0.642	0.124	4.353	0.606	85%
Winkel	-0.34	0.04	-4.15	-0.29	
	Gauß 1				
Distanz	1.264	0.441	4.656	1.193	38%
Winkel	-0.33	0.03	-1.02	-0.28	
	Gamma				
Distanz	0.572	0.110	3.660	0.507	89%
Winkel	-0.33	0.03	-4.96	-0.28	

Zusätzlich wird Experiment durchgeführt, indem eine Gamma Verteilung verwendet wird, da sich diese in der Langzeituntersuchung von Ted Driver [Dri07] als die passendste Verteilung für GPS Fehler herausgestellt hat.

Die statistische Auswertung der jeweiligen Experimente ist in Tabelle 6.3 dargestellt bzw. in Abbildung 6.23 grafisch aufbereitet.

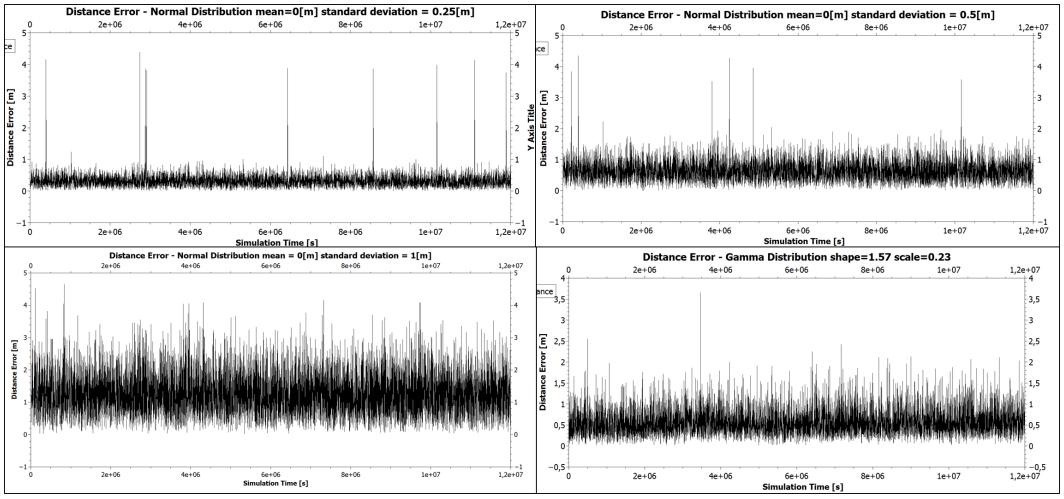

Abbildung 6.23: Bewertung der PNT Unit mit jeweils einem Fehlermodell pro Sensor

Neben diesen Fehlermodellen wurden zudem die in Abschnitt 6.1.1.2 vorgestellten Fehlermodelle für GPS und DGPS Sensoren nachgebildet, ohne Berücksichtigung des Mehrwegefehlers. Hierbei wurde für jeden der vorgestellten Faktoren aus Tabelle 6.1 eine eigene Fehlerkomponente verwendet und konfiguriert.
Die Ergebnisse der simulativen Bewertung mit diesen Fehlermodellen sind in Abbildung 6.24 bzw. Tabelle 6.4 dargestellt.

Tabelle 6.4: Statistische Auswertung für Nachbildung der Fehlermodelle aus Tabelle 6.1

	Mittelwert	**Varianz**	**Max**	**Median**	**Verfügbarkeit**
	GPS				
Distanz	3.576	3.511	11.856	3.369	5%
Winkel	-0.33	0.03	-1.57	-0.28	
	DGPS				
Distanz	0.639	0.115	4.712	0.604	85%
Winkel	-0.33	0.03	-1.11	-0.28	

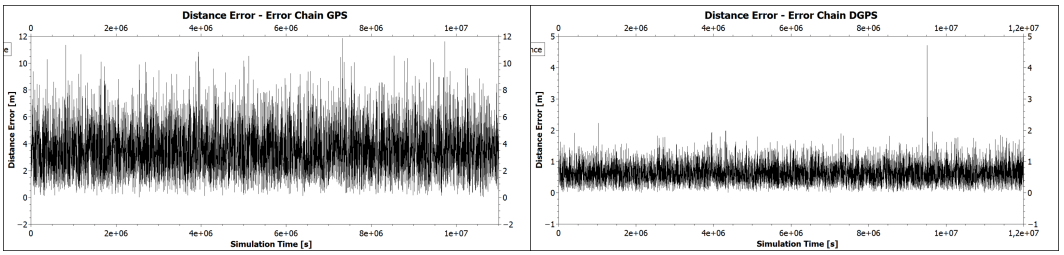

Abbildung 6.24: Bewertung der PNT Unit mit Fehlerketten zur Modellierung der Tabelle 6.1

6.2.3 Zusammenfassende Bewertung der PNT Unit

Die in Abschnitt 6.1.1.1 vorgestellte PNT Unit wurde in diesem Abschnitt und dem vorhergegangenen Abschnitt, mithilfe der vorgestellten Methodik bewertet. Zu diesem Zweck wurden von dem Simulationssystem unterschiedlich stark verrauschte Messwerte an die Sensordatenverarbeitung übermittelt und das Ergebnis mit dem Erwartungswert verglichen.

Bereits im vorangegangenen Abschnitt wurde festgestellt, dass die PNT Unit nicht in der Lage ist, einen Sensorausfall zu detektieren (vgl. Abschnitt 6.1.4.3). Auch informiert sie die nachfolgenden Systeme nicht darüber, dass es zu einem Problem gekommen ist. Nachfolgende Systeme müssen dies, über das Ausbleiben von Nachrichten der Sensordatenverarbeitung, selbständig ermitteln.

Ein ähnlich schlechtes Ergebnis liefert die Sensordatenverarbeitung bei der Bestimmung der Ausrichtung. Hier hat es sich gezeigt, dass bereits bei der Referenzbewertung, ohne Anwendung von Fehlermodellen, die geforderte Genauigkeit von 0.5°, nicht mit einer ausreichenden Verfügbarkeit geliefert werden kann. Weshalb im folgendem der Fokus auf die Positionsbestimmung gelegt wurde.

In den statistischen Auswertungen der Simulationsexperimente bzw. den Bewertungsergebnissen, hat sich gezeigt, dass abgesehen von dem normalverteilten Rauschen mit einer Standardabweichung von einem Meter sowie dem in Abschnitt 6.1.1.2 vorgestellten

GPS Fehlermodell, alle Verfahren im Mittel Ergebnisse liefern, die die Qualitätskriterien erfüllen. Allerdings wurde dabei nicht die Verfügbarkeit eines korrekten Ergebnisses betrachtet.
Wird die Verfügbarkeitsanforderung von 95% bei der Bewertung berücksichtigt, kann aus den Tabellen 6.3 und 6.4 entnommen werden, dass nur ein sehr schwaches Rauschen von der Sensordatenverarbeitung verkraftet werden kann.

Zusammenfassend lässt sich feststellen, dass die untersuchte Sensordatenverarbeitung nicht den, von der IMO geforderten, Qualitätskriterien entspricht.

6.3 Anwendungsszenario maritime Lagebilder

Das dritte und letzte Evolutionsszenario soll die Erfüllung des zweiten formulierten Ziels dieser Arbeit zeigen, der Unterstützung bei der Entwicklung von neuen sensordatenverarbeitenden Systemen.
Aus diesem Grund wird der Fokus dieses Abschnittes weniger auf der Modellierung von Fehlermodellen und einer anschließenden Bewertung der Sensordatenverarbeitung gelegt, sondern vielmehr den Einsatz der Methodik in einem konkreten Projektkontext betrachten.
Dies beinhaltet unter anderem eine Betrachtung der Anforderung A7 (Nachbildung realer Schnittstellen), indem gezeigt wird, dass die verwendeten Ansätze dazu genutzt werden können, einem kommerziellen sensordatenverarbeitenden System, die Existenz von Sensoren vorzutäuschen. Zum anderen wird darauf eingegangen wie neue, teils komplexe, Sensoren in das Simulationssystem integriert werden können.

Bei dem Projekt handelt es sich um das Verbundprojekt COSINUS[10] (Cooperative Shipping and Navigation on Sea)[BHB+14] bei dem die Sicherheit des maritimen Verkehrs verbessert werden sollte[11]. Die Verbesserung wurde durch die Erzeugung abgestimmter Verkehrslagebilder, die sowohl den Kapitänen an Bord der Schiffe, als auch den Verkehrsleitzentralen an Land zur Verfügung stehen, erreicht. Dies beinhalt vor allem den Austausch von verfügbaren Sensorinformationen vom Schiff zur Verkehrsleitzentrale, als auch die Weitergabe der landgestützten Lagebilder an die Schiffe. Weiterhin wurden in dem Projekt neue Mensch-Maschine Schnittstellen entwickelt, welche die zuständigen Nautiker in ihrer Entscheidungsfindung unterstützen.

[10] http://www.emaritime.de/projects/cosinus/
[11] Die Projektlaufzeit von COSINUS Betrug 2 Jahre und Endete im Oktober 2015

Das Projektkonsortium setzte sich zu gleichen Teilen aus Forschungsinstituten und maritimer Industrie zusammen, wobei die Industriepartner die von ihnen betriebenen Systeme dem Projekt zur Verfügung stellten. Konkret standen dem Projekt durch die Kooperation ein Brückensystem, bestehend aus einem ECDIS[12] (Electronic Chart Display and Information System) und einem CONNING[13] Display [Ans16], sowie die Software eines Vessel Traffic Service (VTS) Centers [Sig14], zur Verfügung. Abbildung 6.25 zeigt Screenshots der entsprechenden Systeme.

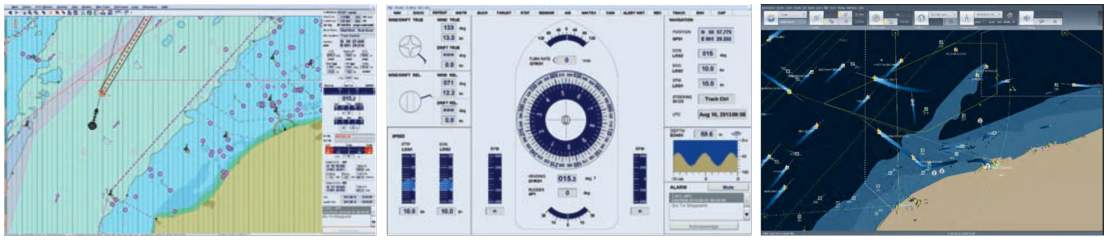

Abbildung 6.25: Darstellung der maritimen Systeme ECDIS und CONNING (Firma: Raytheon Anschütz [Ans16]) und des STYRIS VTS Systems (Firma: Signalis [Sig14])

Erprobt wurden die Ergebnisse des COSINUS Projektes in zwei Phasen unter Zuhilfenahme der eMaritime Integrated Reference Platform (eMIR[14]). Dabei handelte es sich in der ersten Phase um eine simulative Erprobung der entwickelten Technologien und Konzepte mit dem Simulationsframework HAGGIS[15] [SGHB14]. In diesem Kontext wurde unter anderem auf die in dieser Arbeit vorgestellte Sensorsimulation zurückgegriffen.
In der zweiten Phase wurden die COSINUS Konzepte physikalisch mit dem LABSKAUS Testbed [SHB14] evaluiert und demonstriert.

6.3.1 Versuchsaufbau

Der konzeptionelle Aufbau des COSINUS Systems ist in Abbildung 6.26 dargestellt. Es besteht zum einen aus dem Integrated Navigation System (INS), welches an Bord der Schiffe zur Verfügung steht, zum anderen einer Verkehrsleitzentrale. Beide Systeme sind in der Lage Sensorinformationen zu verarbeiten und darzustellen. Zu den verwendeten Sensoren gehören hauptsächlich AIS (Automatic Identification System) Sensoren sowie Radargeräte. An Bord des Schiffes stehen zudem weitere Sensoren zur Verfügung, wie beispielsweise GPS, Echolot und Log für die Geschwindigkeitsmessung. Aus den verfügbaren Sensoren wird auf beiden Seiten unabhängig voneinander ein Lagebild erstellt und

[12] http://www.bsh.de/de/Produkte/Karten/Elektronische_Seekarten/356.jsp
[13] http://www.raytheon-anschuetz.com/products-solutions/product-range/synapsis-conning/
[14] http://www.emaritime.de/
[15] http://www.emaritime.de/services/haggis/

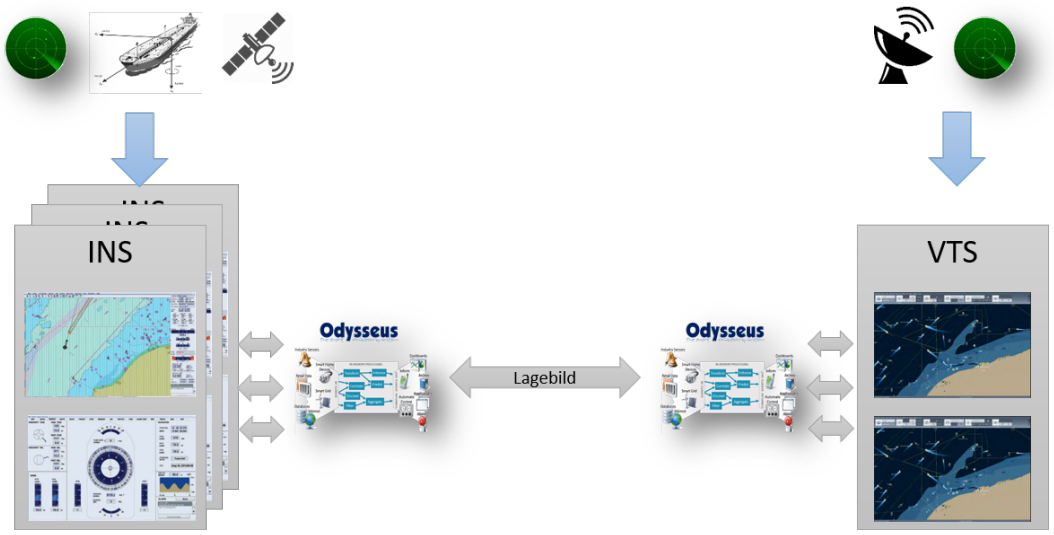

Abbildung 6.26: Konzeptioneller Aufbau des COSINUS Systems

dem jeweiligen Nautiker präsentiert.
Dabei kann es vorkommen, dass nicht alle benötigten Informationen zur Verfügung stehen. Sowohl AIS als auch Radarsensoren benötigen eine Sichtverbindung zu einander, die ggf. von anderen Schiffen, Landmassen oder großen Gebäuden unterbrochen werden kann.
Um dennoch beiden Seiten ein einheitliches und möglichst vollständiges Lagebild bereitstellen zu können, greift das COSINUS System das Lagebild bzw. die Sensorinformationen aus beiden Systemen ab und leitet diese an das jeweils anderen Systeme weiter. Dabei kommt jeweils eine Instanz des Datenstrommanagementsystems Odysseus zum Einsatz.

Im Rahmen der Entwicklung sowie der simulativen Evaluation kam der Sensorsimulation die Aufgabe zu, die realen Sensoren, mit deren Informationen die beiden Systeme gespeist werden, zu simulieren.
Da es sich bei beiden Systemen um kommerzielle Systeme handelt, müssen die Sensorrohdaten in den üblichen Formaten zur Verfügung gestellt werden. Auf Schiffsseite handelt es sich dabei im Allgemeinen um das NMEA Protokoll. Auf der VTS Seite wird zudem das sogenannte IVEF (Inter-VTS Exchange Format) [IAL06] unterstützt.
Weiterhin wird die Sensorsimulation durch eine maritime Verkehrssimulation (MTS) unterstützt, welche ein realistisches Verhalten von Verkehrsteilnehmern, respektive Schiffen, bereitstellt [DH14]. Diese wird über eine High Level Architecture (HLA) Schnittstelle als Verhaltenssimulation eingebunden.
Aus den von der MTS simulierten Schiffen wurde mit Hilfe der Sensorsimulation AIS und Radarsensoren simuliert. Weitere schiffsbezogene Sensormessungen wie Geschwindigkeit, Kompass, Wind und Strömung wurden mit Hilfe der in Abschnitt 4.3.2.2 vorgestellten

Weiterleitung von Informationen aus dem Datenmodell in Sensormessungen umgewandelt und über eine NMEA0183 Schnittstelle in die Schiffssysteme eingespeist.

Die Folgende Abbildung 6.27 zeigt exemplarisch die beteiligten Systeme. In dem dargestellten Bild wurde anstelle des COSINUS ECDIS Displays der frei verfügbare Kartenplotter OpenCPN[16] verwendet, der über die gleichen Schnittstellen wie das COSINUS System verfügt.

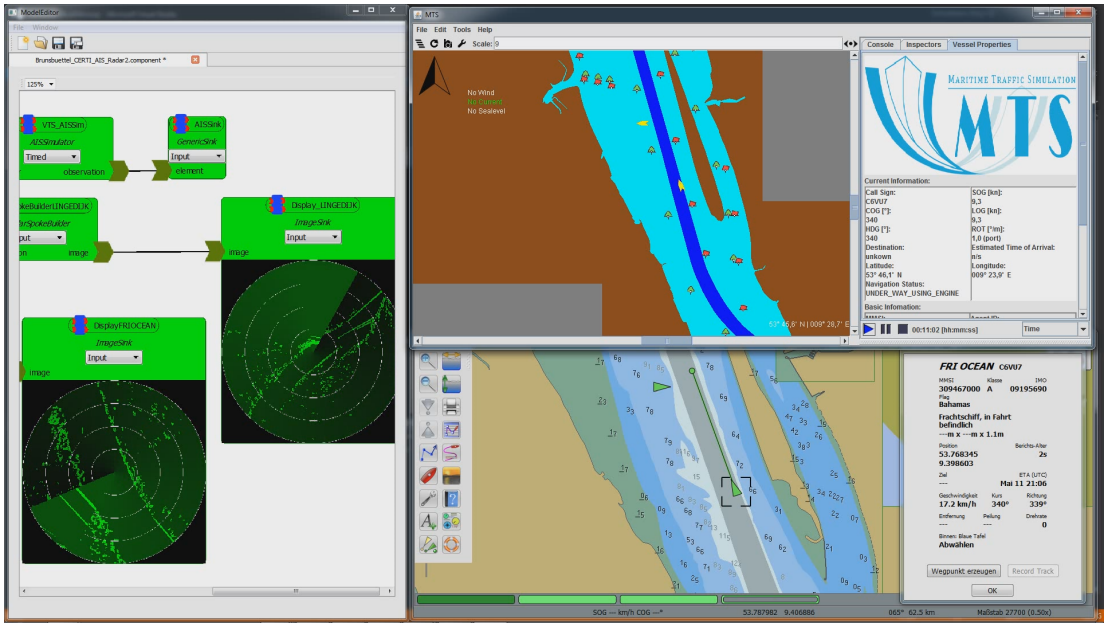

Abbildung 6.27: Screenshot des COSINUS Versuchsaufbaus. Links: Sensorsimulation mit aktivierter Radar Simulation; Rechts Oben: Maritime Verkehrssimulation; Rechts Unten: ECDIS Display (OpenCPN)

6.3.2 Maritime Sensoren

Im Rahmen des COSINUS Projektes wurde die Sensorsimulation um zwei weitere Sensoren erweitert: AIS und Radar. Die folgenden beiden Abschnitte skizzieren kurz ihre Umsetzung innerhalb des beschriebenen Sensorsimulationsframeworkes.

6.3.2.1 Automatic Identification System

Bei dem Automatic Identification System (AIS) handelt es sich um ein Sensorsystem, welches verschiedene statische und dynamische Daten eines Schiffes über eine Funkstrecke verbreitet. Andere Schiffe können diese Signale auffangen, um so ein detailliertes Lagebild aller (größeren[17]) Schiffe in der Umgebung zu erhalten.

[16] `http://opencpn.org/ocpn/`

[17] Für die Berufsschiffart gilt eine Ausrüstungspflicht für AIS Sensoren, wenn es sich um ein Schiff mit einer Bruttoraumzahl (eng. Gross-Tonnage) von mehr 300 handelt [SOL02]

Entwickelt wurde AIS vornehmlich zur Kollisionsvermeidung zwischen Schiffen, wird aber inzwischen auch für die Überwachung des Küstenraums und zur Lenkung des Schiffsverkehrs eingesetzt (z.B. durch die im COSINUS Projekt beteiligten Verkehrsleitzentralen).

Aus technischer Sicht handelt es sich bei dem AIS System nicht um einen Sensor im klassischen Sinne, sondern vielmehr um einen virtuellen Sensor wie er in Abschnitt 2.2.2 beschrieben wurde. Zur Bestimmung der Position, des Kurses oder der Geschwindigkeit greift das AIS System auf zusätzliche Sensoren zurück und kombiniert die Werte zu einer AIS Nachricht. Dies gilt insbesondere für die Nachrichten mit dynamischen Inhalt. Zusätzlich werden (i.d.R. manuell eingegebene) statische Informationen wie die Größe des Schiffes oder der Zielhafen weitergegeben.
Das Versenden der AIS Nachrichten geschieht über einen von zwei UKW Kanälen, in einem selbst organisierenden Zeitschlitzverfahren (SOTDMA = Self-Organized Time Division Multiple Access)[IR10].

Insgesamt stehen innerhalb des AIS Systems 27 verschiedene Nachrichten zur Verfügung, wobei ein großer Teil für das Management des Zeitschlitzverfahrens verwendet wird. Die für die Anzeige in einem Navigationssystem relevanten Informationen werden vornehmlich durch die Nachrichten 1,2,3 (dynamische Informationen) und 5 (statische Informationen) verbreitet. Diese Informationen werden innerhalb von fest vorgegebenen Zeitintervallen versendet. Dabei hängt das Zeitintervall für die dynamischen Schiffsinformationen maßgeblich von der Geschwindigkeit des Schiffes ab und reicht von einem zwei-Sekunden Intervall für schnelle Schiffe bis zu einem drei- Minuten Intervall für ankernde Schiffe. Statische Informationen werden dagegen nur alle sechs Minuten versendet (vgl. [IAL01])

Simulation von AIS Sensoren & Emittern Für die Sensorsimulation wurde die AIS Simulation in eine Emitter- und eine Sensorsimulation aufgeteilt[18]. Gleichermaßen wurde neben einem AIS Sensor ein AIS Emitter im Datenmodell erzeugt, die einem Schiff hinzugefügt werden können. Durch diese Aufteilung lassen sich Schiffe simulieren, die über einen AIS Sensor verfügen aber nicht eigenständig Nachrichten für andere Schiffe bereitstellen. Dies gilt besonders für kleine bis mittelgroße Schiffe im privaten Sektor, u.a. das bereits vorgestellte Forschungsschiff OTZUM.

Für jeden im Szenariomodell erkannten AIS Emitter wird ein eigenständiger Task innerhalb des diskreten Event-Schedulers erstellt. Dieser regelt abhängig von der aktuellen

[18]Dieses Vorgehen wurde in dem 2014 in der Arbeit “HAGGIS: A modelling and simulation platform for e-Maritime technology assesment“ [SGHB14] präsentiert

Geschwindigkeit des ihm zugewiesenen Schiffes die Wiederholfrequenz. Sobald ein Emitter aktiviert wird, werden die dynamischen und statischen Informationen aus dem Szenariomodell ausgelesen und in eine AIS Beobachtung überführt. Dabei kann der Emitter auf andere Sensoren zurückgreifen, sofern diese für das entsprechende Schiff konfiguriert worden sind.
Verfügt ein Schiff beispielsweise sowohl über einen AIS Emitter als auch über einen GPS-Sensor, werden automatisch die Informationen des GPS Sensors für das Erstellen einer Positionsnachricht verwendet. Ist dagegen kein GPS-Sensor verfügbar, wird die Position des Schiffes aus dem Szenariomodell ausgelesen. Analog dazu ermittelt der Emitter die Informationen für Geschwindigkeit, Kurs und Ausrichtung des Schiffes.
Für das Erstellen von statischen Nachrichten, werden Informationen wie Name, MMSI (Maritime Mobile Service Identity) und Größe des Schiffes, direkt aus dem Szenariomodell ausgelesen.
Nachdem alle Informationen gesammelt wurden, werden die Daten in Form von AIS Beobachtungen an einen globalen SOTDMA - Manager übermittelt. Dieser SOTDMA-Manager emuliert die Verwendung der Funkkommunikation, indem es der Beobachtung einen von insgesamt 4500 Zeitslots (vgl. [IR10]) zuweist.

Die Simulation der AIS Sensoren beschränkt sich in diesem Fall auf das Sammeln der AIS Beobachtungen aus dem SOTDMA - Manger, unter der Berücksichtigung der maximalen Reichweite des Sensors. Diese liegt in der Regel zwischen 20 und 40 nautischen Meilen, bedingt durch die Funkkommunikation und kann für jeden Sensor einzeln konfiguriert werden.

6.3.2.2 Radar Simulation

Im Folgenden wird die Entwicklung eines maritimen Radarsensors mit Hilfe des erstellten Simulationsframeworks vorgestellt.
Maritime Radare stellen insofern eine besondere Herausforderung für eine Sensorsimulation dar, da sie i.d.R. ein sehr großes Gebiet abdecken müssen. Selbst Radargeräte die im privaten Bereich (Segel- / Motor- Yacht) eingesetzt werden verfügen zum Teil über Reichweiten von 24 bis 98 Nautischen Meilen (44 - 181 km). Die Reichweite von militärischen bzw. landgestützten Anlagen, kann u.U. noch größer sein und wird hauptsächlich durch die Krümmung der Erdoberfläche beschränkt.

Funktionsweise maritimer Radarsensoren Die generelle Funktionsweise eines Radargerätes ist in Abbildung 6.28 (links) dargestellt. Das Radar sendet eine elektromagnetische Welle aus, die von einem Hindernis, zum Beispiel einem Schiff, reflektiert wird.

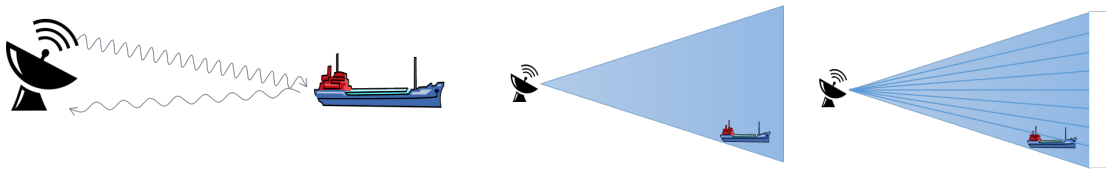

Abbildung 6.28: Links: Generelle Funktionsweise von Radarsensoren; Mitte / Rechts: Einfluss der Strahlbündelung auf die Winkelgenauigkeit des Radargerätes

Teile der reflektierten Welle, ggf. in veränderter Form, können dann mit einem Empfänger gemessen werden. Im Fall von maritimen Radaren, handelt es sich i.d.R. beim Sender und Empfänger um das gleiche physikalische Gerät. Aufgrund der bekannten Ausbreitungsgeschwindigkeit der elektromagnetischen Welle von ca. 300.000 km/h lässt sich durch die Ermittlung der vergangenen Zeit zwischen Aussenden und dem Empfangen der Reflexion die Entfernung des Hindernisses bestimmen.
Indem zudem die vom Sender ausgestrahlte Welle in eine bestimmte Richtung gebündelt wird, lässt sich weiterhin der Winkel des Objektes relativ zum Sender bestimmen. Dabei hängt die Winkelauflösung des Sensors von der Bündelung der Welle sowie der Entfernung des Objektes ab, wie es Abbildung 6.28 (Mitte & Rechts) dargestellt ist.
Neben der Bündelung in der horizontalen Ebene, wird der Radarstrahl auch in der vertikalen Ebene gebündelt. Wobei dieser Effekt in der kommerziellen Seefahrt nicht so stark betrachtet werden muss. Wie in der Abbildung dargestellt, kann die ausgestrahlte elektromagnetische Welle durch einen Kegel angenähert[19] werden.

Die Stärke des reflektierten Signals lässt sich mithilfe der von Skolnik [Sko81] angegebenen allgemeinen Radargleichung bestimmen.

$$R_{max} = \sqrt[4]{\frac{P_t G A_e \sigma}{(4\pi)^2 S_{min}}} \tag{6.2}$$

Dabei handelt es sich bei R_{max} um die theoretische, maximale Reichweite, die für eine gegebene Ausgangsleistung P_t, eine minimale Detektionsleistung S_{min}, sowie die Antennenparameter Gain (G) und effektive Antennenfläche A_e, erreicht werden kann.
Der Parameter σ beschreibt den sogenannten Radarquerschnitt. Bei diesem Parameter handelt es sich um eine objektspezifische Größe, mit der die Reflexionseigenschaften des Objektes beschrieben werden. Er hängt i.d.R. von verschiedenen Einflussgrößen ab, wie

- der Form des betrachteten Objektes,
- der Ausrichtung des Objektes relativ zum Radar,
- der vom Radar verwendeten Wellenlänge,

[19] Die tatsächliche Ausbreitung der elektromagnetischen Welle ist i.d.R. sehr viel komplexer und verfügt über verschiedene Haupt- und Nebenkeulen. Dieser Detailgrad wird aber bei der verwendeten Simulationsgenauigkeit nicht benötigt.

- dem Material des Objektes.

Modellierung von Radarsensoren Ähnlich wie bei den AIS Sensoren, wurde ein neuer Radarsensor im Datenmodell angelegt. Der Radarsensor verfügt über Parameter zum Einstellen der effektiven Reichweite sowie den horizontalen und vertikalen Öffnungswinkel der Radarkeulen. Weiterhin lässt sich die Diskretisierung der Radarkeule auf der Längsachse, sowie die Antennenverstärkung (Gain) aus der allgemeinen Radargleichung (vgl. Gleichung 6.2) einstellen.

Maritime Radargeräte liefern i.d.R. zwei Arten von Messwerten 1) Radarbilder und 2) Zielinformationen.

Radarbild Radarbilder werden üblicherweise nicht in Form eines kompletten Bildes geliefert, sondern in Form eines Dreieckigen Ausschnittes, dem sogenannten *Spoke*. Dabei handelt es sich um eine zweidimensionale Abbildung der Radarkeule, deren horizontaler Öffnungswinkel dem Öffnungswinkel des aufgespannten Dreieckes entspricht. Auf der Längsachse wird der Kegel in gleichlange Abschnitte unterteilt. Jeder der dadurch entstehenden Zellen im Spoke kann ein Intensitätswert zugewiesen werden, der entweder der Reflexionsintensität oder einem Vielfachen dieses Wertes entspricht.

Zielinformationen In der christlichen Seefahrt werden Radargeräte hauptsächlich für die Kollisionsvermeidung eingesetzt. Zu diesem Zweck liefern die Geräte neben dem bekannten Radarvideo häufig eine Liste von erkannten Objekten, auch Ziele genannt. Die vom Radar gelieferten Informationen enthalten i.d.R. den bestimmten Abstand sowie den Winkel des Objektes zu dem Radar. Aus diesen Informationen bzw. deren Veränderung lassen sich weitere Informationen wie beispielsweise der aktuelle Kurs oder Geschwindigkeit des Ziels ableiten

Zusätzliche Umgebungsmodellierung Neben künstlichen Objekten wie Schiffen und Bojen müssen bei der Simulation von Radarsensoren auch die geographischen Gegebenheiten berücksichtigt werden, da auch sie die ausgestrahlten Wellen reflektieren.
Zu diesem Zweck wurde das *World Data Model* um ein digitales Geländemodell bzw. ein digitales Höhenmodell erweitert. Dabei können die Geländeinformationen wahlweise in Form von Vektordaten oder als Rasterdaten angegeben werden.

Vektordaten werden dabei in Form von IHO-S-57 [IHO00] Seekarten *Features* angegeben. Diese bestehen aus einer Geometrie, zum Beispiel einem Polygon oder einem Linienzug, sowie einer fest definierten Menge von Attributen.

Für die Simulation von Radarreflexionen, sind alle S-57 *Features* von Interesse, welche über eine Höhenangabe verfügen, wie beispielsweise: BUISGL (Building Single) oder BUAARE (Built-up area) mit deren Hilfe Gebäude oder Ansammlungen von Gebäuden beschrieben werden können.
Dabei müssen diese Vektordaten nicht zwingend aus einer Seekarte stammen. Alternativ können auch andere Datenquellen wie beispielsweise OpenStreetMap[20] herangezogen werden, die i.d.R. über detailierte Informationen (auf Landseite) verfügen.

Rasterdaten werden in Form von georeferenzierten Matrizen bzw. Bildern angegeben, wobei jedem Farbwert eine entsprechende Höhe und jedem Pixel eine entsprechende Position bzw. eine Fläche zugewiesen wird.
Umfangreiche Höhenmodelle in Form von Rasterdaten werden beispielsweise von der NASA bereitgestellt, die im Laufe der "Shuttle Radar Topography Mission" (SRTM) [VZ01] die Erde mit einer Genauigkeit von einer Bogensekunde[21] vermessen hat.
Weitere und zum Teil genauere Höhenmodelle wurden beispielsweise durch die "Advanced Spaceborne Thermal Emission and Reflection Radiometer" (ASTER) Mission [YKT+98] bereitgestellt, welche ebenfalls von der NASA durchgeführt wurde.

Radarsimulation Die Simulation der Radarsimulation wurde aufgrund der Menge von zu verarbeitenden Daten als eine einzelne Simulationskomponente realisiert, ohne auf weitere Komponenten zurückzugreifen.

Die Ermittlung der Daten ist in zwei Phasen unterteilt, wobei die erste ausschließlich die geographischen Gegebenheiten berücksichtigt, während die zweite Phase die physikalischen Objekte innerhalb des Radarkegels ermittelt.

In beiden Phasen wird jeweils nur die aktive Radarkeule betrachtet, wie es in Abbildung 6.29 und Abbildung 6.30 dargestellt ist.
In der ersten Phase werden die Geländeinformationen ermittelt. Dabei wird unterschieden, ob es sich um eine Vektor- oder Rasterbasierte Höhenangabe handelt. Handelt es sich wie in Abbildung 6.29 dargestellt um eine rasterbasierte Karte, wird für jede Zelle des *Spokes* die maximale Höhe aller in der Zelle enthaltenen Pixel ermittelt und als Höheninformation für diese Zelle verwendet. Handelt es sich um einen niedrigeren Höhenwert als bei einer vorherigen Zelle (ausgehend vom Radarsensor) befindet sich die betrachtete Zelle innerhalb des Radarschattens und wird ignoriert (vgl. auch Abbildung

[20] `http://www.openstreetmap.org/`
[21] eine Bogensekunde am Äquator entspricht ungefähr 30m

6.30).
Handelt es sich dagegen um ein vektorbasiertes Geländemodell, wird zunächst eine Kollisionsabfrage des gesamten *Spokes* mit dem Geländemodell durchgeführt. Auf diese Weise lässt sich die Anzahl der für jede Zelle zu überprüfenden Geometrien stark reduzieren. Analog zu dem Vorgehen des rasterbasierten Geländemodells wird das Maximum aller in der Zelle befindlichen Höhenangaben als Wert für die aktuelle Zelle verwendet, sofern sich die Zelle nicht im Radarschatten einer vorherigen Zelle befindet.

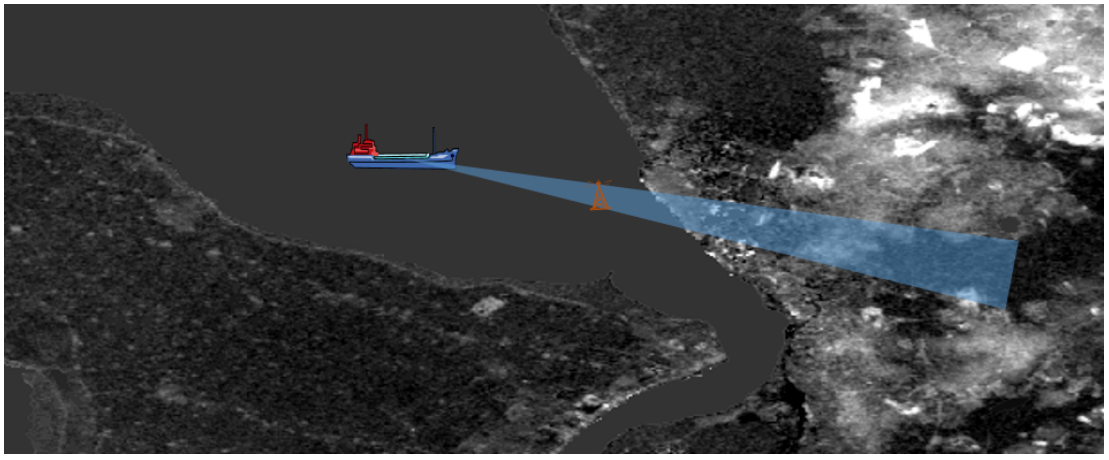

Abbildung 6.29: Prinzip der Radarsimulation aus der Vogelperspektive. Im Hintergrund dargestellt ein ASTER Höhenmodell, wobei eine hellere Farbe ein größere Höhe darstellt. Der dunkle Abschnitt der Radarkeule repräsentiert eine Zelle des betrachteten *Spokes*.

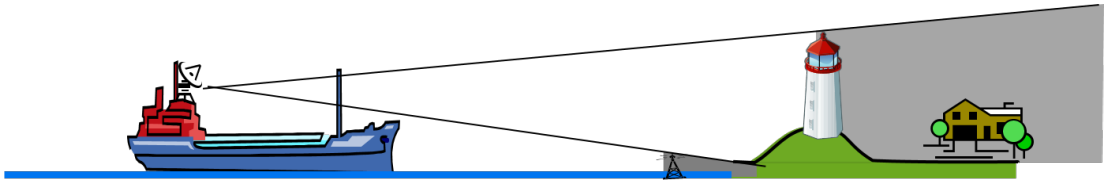

Abbildung 6.30: Prinzip der Radarsimulation aus der Seitenperspektive. Hohe Objekte erzeugen einen Radarschatten, wodurch dahinter befindliche Objekte nicht erkannt werden können.

Im zweiten Schritt werden die Objekte ermittelt, die sich innerhalb des *Spokes* befinden. Potenziell werden alle Objekte des Szenariomodells erkannt, die über eine Position und Form verfügen.
Dabei wird aus Geschwindigkeitsgründen auf eine komplette Kollisionsüberprüfung auf Zellenebene verzichtet. Stattdessen werden zunächst alle Objekte gefiltert, welche sich innerhalb der Radarreichweite befinden. Anschließend wird für diese Objekte die umschließende “Bounding Box“ ermittelt und auf eine Kollision mit dem *Spoke* überprüft. Liegt das Objekt innerhalb des Spokes, wird die maximale und minimale Entfernung des Objektes zum Sensor bestimmt und die von ihm belegten Zellen berechnet. Befinden sich

vor (in Bezug auf den Sensor) dem Objekt keine Zellen mit einem größeren Höhenwert als diesem Objekt wird die Reflexionsstärke des Objektes bestimmt.

Die Bestimmung der Reflexionsstärke erfolgt unter Zuhilfenahme der Radar Cross Section aus Gleichung 6.2. Diese wird entweder von den beobachteten Objekten in Form eines (einheitenlosen) Attributes bereitgestellt oder auf Grundlage der Größe des Objektes geschätzt. Indem für jedes Objekt der Reflexionsfaktor angegeben werden kann, lassen sich Radarreflektoren, wie sie Beispielsweise an Bojen angebracht sind, leicht modellieren.
Übertrifft die Reflexionsstärke einen vorgegebenen Schwellwert, wird das Objekt als ein potenzielles Ziel in die Liste der Radarziele eingetragen und an die Sensordatenverarbeitung weitergegeben. In dem betrachteten Anwendungsszenario geschieht dies ebenfalls über das NMEA0183 Protokoll, indem die Ziele zu TTM (Tracked Target Message) bzw. TLL (Target Latitude and Longitude) Datensätzen umgewandelt werden.

6.3.3 Ergebnis der Evaluation

Die Sensorsimulation wurde im Rahmen von COSINUS an zwei Stellen eingesetzt.
Zum Einen wurden mit ihr die Sensorwerte für das integrierte Navigationssystem und das VTS System erstellt und über die nativen Schnittstellen (NMEA 0183) eingespeist. Darauf aufbauend wurde der Datenaustausch zwischen Schiff und Land entwickelt und getestet. Die Ergebnisse wurden in der Arbeit von Salous et. al. [MSH15] veröffentlicht.
Weiterhin wurde die Radarsimulation dazu verwendet um eine Radarschattenerkennung zu entwickeln. Bei diesem Phänomen werden die Radarsensoren eines Verkehrsteilnehmers durch ein großes, unter Umständen dynamisches Hindernis blockiert und erscheinen nicht mehr in seinem aktuellen Lagebild. Durch die Detektion einer solchen Situation kann gemäß Salous et. al. eine Warnung für die beteiligten Nautiker erstellt werden.
Dabei wurde insbesondere die Fähigkeit der Sensorsimulation verwendet, bereits während der Entwicklung realitätsnahe Sensormesswerte zu erzeugen. So konnte der in COSINUS entwickelte Datenaustausch unter Laborbedingungen entwickelt und getestet werden, ohne das zu diesem Zweck reale Sensoren eingesetzt werden mussten.
Weiterhin hat sich gezeigt, dass der Verzicht auf Middlewarelösungen zur Kommunikation mit der sensordatenverarbeitenden Software, in diesem Fall dem integrierten Navigationssystem und dem VTS System, eine problemlose Integration kommerzieller Produkte ermöglicht.
Gerade diese Eigenschaft der Sensorsimulation kommt neben der Entwicklung des im COSINUS Projekt entstandenen Sensordatenverarbeitung auch dem zweiten Aspekt, der Verbesserung der Mensch - Maschine Schnittstelle, zugute. Indem mit Hilfe der Simulationsumgebung Szenarien nachgebildet werden können, die in der Vergangenheit

zu Unfällen geführt haben. Durch das Bereitstellen der Sensorinformationen für die entsprechenden Systeme lassen sich alte und neue Benutzerschnittstellen direkt miteinander vergleichen, um so den potentiellen Nutzen für zukünftige kritische Situationen abschätzen zu können.

Kapitel 7

Zusammenfassung und Ausblick

Dieses letzte Kapitel fasst noch einmal die vorhergegangenen Abschnitte zusammen und gibt einen Ausblick auf weitere Einsatz- und Erweiterungsmöglichkeiten der entwickelten simulativen Überprüfung von sensordatenverarbeitenden Systemen.

7.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Ansatz zur simulativen Überprüfung von sensordatenverarbeitenden Systemen vorgestellt, prototypisch umgesetzt und evaluiert.
Im Mittelpunkt steht dabei die Fragestellung, wie die korrekte Funktionsweise von sensordatenverarbeitenden Systemen überprüft und gleichzeitig die Entwicklung solcher Systeme unterstützt werden kann.
Dazu wurde im Abschnitt 1.3 zunächst eine grundlegende Ideenskizze vorgestellt, bei der die reale Umwelt eines sensordatenverarbeitenden Systems zur Umgebungswahrnehmung durch eine simulierte und damit vollständig kontrollierbare Umgebung ersetzt wird.

Das zweite Kapitel beschäftigte sich vornehmlich mit Sensoren und sensordatenverarbeitenden Systemen, um ein gemeinsames Verständnis des Untersuchungsgegenstandes herbeizuführen. In diesem Rahmen wurde die Funktionsweise eines sensordatenverarbeitenden Systems anhand einer Drei-Schichten-Architektur zur Fusion von Umfeldsensoren vorgestellt.
Weiterhin wurde im Abschnitt 2.3 auf Sensorfehler und Messungenauigkeiten eingegangen, die die Sensordatenverarbeitung zu einer komplexen Herausforderung machen und aus Sicht des Autors zwingend bei der Simulation von Sensoren berücksichtigt werden

müssen. Die Sensorfehler wurden dazu in drei Kategorien eingeteilt, die jeweils unterschiedliche starke Herausforderungen sowohl für die Sensordatenverarbeitung als auch die realitätsnahe Simulation von Sensorwerten darstellen.

Das Dritte Kapitel beschäftigte sich mit bestehenden Ansätzen zur Erzeugung von Sensordaten. Dabei wurden zunächst zwei grundlegend verschiedene Ansätze vorgestellt. Zum einen das Erzeugen auf Grundlage von historischen Messwerten (vgl. Abschnitt 3.1), sowie das Generieren von Sensorwerten durch die Beobachtung von virtuellen bzw. simulierten Umgebungen (vgl. Abschnitt 3.2).
Ebenfalls Bestandteil des dritten Kapitels war eine Betrachtung und Analyse existierender Simulationssysteme in Bezug auf die gestellten Anforderungen. Das Ergebnis dieser Betrachtung ist, dass keiner der untersuchten Ansätze alle gestellten Anforderungen vollständig unterstützt.

Im 4 Kapitel der Arbeit wurde der eigene Ansatz zur simulativen Überprüfung von sensordatenverarbeitenden Systemen vorgestellt.
Dieser besteht aus einer Methode mit 3 Schritten, der Identifikation der beteiligen Systeme durch einen Experten, der Modellierung der Sensoren, ihres Verhaltens und der Qualitätsanforderungen an das sensordatenverarbeitende System und zuletzt die Durchführung der simulativen Bewertung. Dabei wurde der Fokus klar auf die Modellierung der Sensoren und ihres Verhaltens bzw. Fehlverhaltens gelegt.
Zunächst wurde eine Beschreibungssprache vorgestellt, mit der die Sensoren aber auch die Elemente der Umwelt in der sie sich befinden, strukturell beschrieben werden können. Darauf aufbauend wurde eine Erweiterung der Beschreibungssprache vorgestellt, mit der sich neben der Struktur auch das Verhalten von Objekten, in Form eines datenflussorientierten Verarbeitungsgraphen, beschreiben lässt.
Anschließend wurden beschrieben, wie mithilfe der Beschreibungssprachen Sensormesswerte erzeugt und verrauscht werden können. Dabei kommen verschiedene Fehlermodelle zum Einsatz, die mithilfe des Verarbeitungsgraphen schnell und einfach zu komplexen Fehlermodellen zusammengestellt und auf die Sensormessungen angewendet werden können.
Außerdem wurde beschrieben, wie verschiedene Fehler des zu überprüfenden sensordatenverarbeitenden Systems ermittelt werden können.
Den Abschluss des eigenen Ansatzes stellt die Vorstellung der prototypischen Umsetzung, mithilfe eines Modellgetriebenen Softwareentwicklungsansatzes dar.

Das fünfte und letzte Kapitel der Arbeit beschäftigte sich abschließend mit der Evaluation der vorgestellten Methodik. Dazu wurden zunächst einzelne Aspekte des Ansatzes demonstriert bzw. evaluiert. Anschließend wurde eine simulative Bewertung eines sensordatenverarbeitenden Systems exemplarisch durchgeführt. Dabei handelte es sich um

eine “Position Navigation and Timing (PNT) Unit“, wie sie in der maritimen Domäne häufig verwendet wird.
Den Abschluss bildete ein komplexes Anwendungsszenario welches im Rahmen des Forschungsprojektes COSINUS durgeführt wurde. In diesem Zusammenhang wurde insbesondere die Fähigkeit des beschriebenen Ansatzes demonstriert, die Entwicklung von neuen sensordatenverarbeitenden Systemen zu unterstützten.

7.2 Ausblick

Im Laufe der Beantwortung der in Abschnitt 1.2 formulierten Fragestellung haben sich weitere Fragestellungen ergeben, die im Folgenden kurz angeschnitten werden sollen.

Automatische Erstellung des Sensorsimulationsmodells

Für die Durchführung eines Simulationsexperiments inklusive Überprüfung der Sensordatenverarbeitung müssen in dem vorgestellten Ansatz das Simulationsmodell sowie das Modell zur Überprüfung des zu testenden Systems manuell zusammengestellt werden.
Zwar ermöglicht dies eine hohe Flexibilität insbesondere in Hinblick auf die zu überprüfenden Inhalte der Sensordatenverarbeitung, erfordert aber auf der anderen Seite einen hohen Modellierungsaufwand. In weiteren Arbeiten könnte untersucht werden, ob sich eines der beiden Modelle aus dem anderen herleiten lässt.
Ein denkbares Vorgehen an dieser Stelle wäre das manuelle Erstellen eines Prüfmodells, indem das Verhalten der Sensordatenverarbeitung formal spezifiziert wird. Aus den zur Überprüfung der Sensordatenverarbeitung verwendeten Informationen des Szenariomodells können dann automatisch die notwendigen Sensoren, mit denen die Sensordatenverarbeitung gespeist wird, abgeleitet werden. Auf eine ähnliche Art könnten zudem die Fehlermodelle der Sensoren aus den akzeptierten Fehlermargen hergeleitet werden oder zumindest Empfehlungen ausgesprochen werden.

Automatisches Erzeugen von kritischen Situationen

Im Bereich der Sensordatenverarbeitung führen häufig Kombinationen von einzeln betrachtet unkritischen Situationen zu einem Zusammenbruch des Systems oder zu mindestens zu größeren Problemen. Im Zusammenhang mit optischen Sensoren kann es sich beispielsweise um eine Situation handeln, in der eine Kamera durch ein vorbeifahrendes Fahrzeug mit aktivierten Scheinwerfern geblendet wird.

Solche Fehler können mit dem vorgestellten Ansatz zur Sensorsimulation abgedeckt werden, indem sie als kontextabhängige Fehler modelliert werden. Dafür muss aber zunächst eine solche Situation innerhalb der Verhaltenssimulation hergestellt werden. Dies ist mit den aktuellen Ansätzen nur durch eine manuelle Herbeiführung der Situation möglich, indem z.B. die Pfade eines Fahrzeuges aktiv in eine solche Situation führen.
Durch den Bedarf einer manuellen Konfiguration eines solchen Extremfalles besteht die Gefahr, dass ein solches Szenario bei der Überprüfung der Sensordatenverarbeitung vergessen wird. Wünschenswert wäre eine automatische Erzeugung solcher Situationen.
Ansätze zur automatischen Erzeugung solcher kritischen bzw. seltenen Situationen werden bereits aktiv untersucht, zum Beispiel durch die Arbeiten von Gollücke et. al. [GPL+12], könnten aber ggf. eine stärkere Berücksichtigung der Fähigkeiten von Sensoren und ihren Fehlerquellen enthalten.
Erste Überlegungen zu einer Kombination von Sensorsimulation mit der automatischen Erzeugung bzw. Findung von kritischen Situationen wurden in der Arbeit “Virtual Testbed for Maritime Safety Assessment“ [HSGB15] veröffentlicht.

Literaturverzeichnis

[AG17] Rheinmetall AG. Rheinmetall Defence - Maritime Ausbildungslösungen. https://www.rheinmetall-defence.com/de/rheinmetall_defence/systems_and_products/simulation_and_training/civil_training_solutions/merchant_marine_training/index.php, 2017. Accessed: 2017-04-12.

[Ais15] Ala Aism. IALA recommendation r-121 on the performance and monitoring of DGNSS services in the frequency band 283.5 - 325 kHz. 2015.

[AKC+15] C.E. Aguero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J.L. Rivero, J. Manzo, E. Krotkov, and G. Pratt. Inside the Virtual Robotics Challenge: Simulating Real-Time Robotic Disaster Response. *Automation Science and Engineering, IEEE Transactions on*, 12(2):494–506, April 2015.

[Ans16] Raytheon Anschütz. Synapsis Intelligent Bridge Control. Brochure, 2016.

[Ao02] National Marine Electronics Association and others. *NMEA 0183 Standard for interfacing marine electronic devices*. NMEA, 2002.

[AT] Adeel Asghar and Sonja Tariq. OpenModelica v1.11.0 and OMEdit v1.11.0.

[BGG+10] André Bolles, Dennis Geesen, Marco Grawunder, Jonas Jacobi, Daniela Nicklas, and Hans-Jürgen Appelrath. Sensordatenverarbeitung mit dem open source datenstrommanagementframework odysseus. In *GI Jahrestagung (2)*, pages 404–409, 2010.

[BHB+14] André Bolles, Axel Hahn, Michael Braun, Sven Rohde, Michael Baldauf, Michael Gluch, and Sebastian Klaes. COSINUS - Cooperative Vessel Guidance for Nautical Safety. 2014.

[BKG+14] K Benedict, M Kirchhoff, M Gluch, S Fischer, and M Schaub. Simulation augmented manoeuvring design and monitoring–a new method for advanced

ship handling. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, 8, 2014.

[BMB+15] G. Bianco, M. Muzzupappa, F. Bruno, R. Garcia, and L. Neumann. A new color correction method for underwater imaging. XL-5/W5:25–32, 2015.

[BMG09] José-Luis Blanco, Francisco-Angel Moreno, and Javier González. A collection of outdoor robotic datasets with centimeter-accuracy ground truth. *Autonomous Robots*, 27(4):327–351, November 2009.

[BMGJ14] José-Luis Blanco, Francisco-Angel Moreno, and Javier González-Jiménez. The málaga urban dataset: High-rate stereo and lidars in a realistic urban scenario. *International Journal of Robotics Research*, 33(2):207–214, 2014.

[Boc11] Stephen Bocquet. Calculation of radar probability of detection in K-distributed sea clutter and noise. Technical report, DTIC Document, 2011.

[Bol11] André Bolles. *Ein datenstrombasiertes Framework zur Objektverfolgung am Beispiel von Fahererassistenzsystemen.* OlWIR, Oldenburger Verlag für Wirtschaft, Informatik und Recht, 2011.

[BR07] Mike Botts and Alexandre Robin. OpenGIS sensor model language (SensorML) implementation specification. *OpenGIS Implementation Specification OGC*, pages 07–000, 2007.

[Bro08] Marc Brooker. *The design and implementation of a simulator for multistatic radar systems.* PhD thesis, University of Cape Town, 2008.

[CBB+12] Michael Compton, Payam Barnaghi, Luis Bermudez, Raúl García-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, Vincent Huang, Krzysztof Janowicz, W. David Kelsey, Danh Le Phuoc, Laurent Lefort, Myriam Leggieri, Holger Neuhaus, Andriy Nikolov, Kevin Page, Alexandre Passant, Amit Sheth, and Kerry Taylor. The SSN ontology of the W3c semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25–32, December 2012.

[CG07] Vittorio Cortellessa and Vincenzo Grassi. A modeling approach to analyze the impact of error propagation on reliability of component-based systems. In *Component-Based Software Engineering*, pages 140–156. Springer, 2007.

[CJCA11] Jean-Paul Calbimonte, Hoyoung Jeung, Oscar Corcho, and Karl Aberer. Semantic sensor data search in a large-scale federated sensor network. 2011.

[Cla08] Jose Luis Blanco Claraco. Development of scientific applications with the mobile robot programming toolkit. *The MRPT reference book. Machine Perception and Intelligent Robotics Laboratory, University of Málaga, Málaga, Spain*, 2008.

[CLW+07] Stefano Carpin, Michael Lewis, Jijun Wang, Stephen Balakirsky, and Chris Scrapper. USARSim: a robot simulator for research and education. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1400–1405. IEEE, 2007.

[CMG05] Toby HJ Collett, Bruce A. MacDonald, and Brian P. Gerkey. Player 2.0: Toward a practical robot programming framework. In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, page 145, 2005.

[Cox07] S Cox. OGC Implementation Specification 07-022r1: Observations and Measurements-Part 1-Observation schema. Open Geospatial Consortium. Technical report, Tech. Rep, 2007.

[Dar07] Michael Darms. *Eine basis-systemarchitektur zur sensordatenfusion von umfeldsensoren fuer fahrerassistenzsysteme.* PhD thesis, TU Darmstadt, 2007.

[DH14] Christoph Dibbern and Axel Hahn. Maritime Traffic Co-Simulation For Analyses of Maritime Systems. In *16th International Conference on Harbor, Maritime and Multimodal Logistics Modeling and Simulation (HMS 2014)*, volume 1, pages 68–75. Curran Associates, Inc., 2014.

[DHS14] Christoph Dibbern, Axel Hahn, and Sören Schweigert. Interoperability In Co-Simulatons Of Maritime Systems. In *ECMS*, pages 71–77, 2014.

[Dri07] Ted Driver. Long-term prediction of gps accuracy: Understanding the fundamentals. In *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2007), Fort Worth*, 2007.

[Dro16] Rainer Droste. *Modellbasierte Planung zur Unterstützung der Gefährdungsbeurteilung maritimer Operationen.* PhD thesis, Dissertation, Oldenburg, Universität Oldenburg, 2016, 2016.

[DRRB07] Anca Discant, Alexandrina Rogozan, C. Rusu, and A. Bensrhair. Sensors for obstacle detection-a survey. In *Electronics Technology, 30th International Spring Seminar on*, pages 100–105. IEEE, 2007.

[EHR+15] E Engler, M Hoppe, J Ritterbusch, T Ehlers, C Becker, KC Ehrke, and H Callsen-Bracker. Guidelines for the coordinated enhancement of the maritime PNT system. In *MARINE TRAFFIC ENGINEERING AND INTERNATIONAL SYMPOSIUM INFORMATION ON SHIPS*, page 143, 2015.

[EKR+11] Sönke Eilers, Christian Kuka, Stefan Rührup, Sören Schweigert, Tobe Toben, and Hannes Winkelmann. Safe autonomous transport vehicles in heterogeneous outdoor environments. 2011.

[EUR09] EUROCONTROL. Category 240 Radar Video Transmission. Standard, May 2009.

[EV06] Sven Efftinge and Markus Völter. oAW xText: A framework for textual DSLs. In *Workshop on Modeling Symposium at Eclipse Summit*, volume 32, page 118, 2006.

[EVA11] GEOFF EVANS. BEHIND THE MIRROR - ADDING REFLECTION TO C++. *Game Developer Magazine*, (Volume 18 Number 2), November 2011.

[FDL02] Kay Ch Fuerstenberg, Klaus CJ Dietmayer, and Dirk T. Linzmeier. Pedestrian recognition in urban traffic using a vehicle based multilayer laserscanner. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 1, pages 31–35. IEEE, 2002.

[FTO+08] Luke Fletcher, Seth Teller, Edwin Olson, David Moore, Yoshiaki Kuwata, Jonathan How, John Leonard, Isaac Miller, Mark Campbell, Dan Huttenlocher, and others. The MIT-Cornell collision and why it happened. *Journal of Field Robotics*, 25(10):775–807, 2008.

[FW01] Kay Ch Fuerstenberg and Volker Willhoeft. Pedestrian Recognition in urban traffic using Laserscanners. In *Proceedings of ITS*. Citeseer, 2001.

[GDG+10] Chris Goodin, Phillip J. Durst, Burhman Gates, Chris Cummins, and Jody Priddy. High fidelity sensor simulations for the virtual autonomous navigation environment. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 75–86. Springer, 2010.

[GG12] Dennis Geesen and Marco Grawunder. DEBS grand challenge: Odysseus as platform to solve grand challenges. 2012.

[GGDS12] D. Gruyer, M. Grapinet, and P. De Souza. Modeling and validation of a new generic virtual optical sensor for ADAS prototyping. pages 969–974. IEEE, June 2012.

[Gia86] Joseph M Giachino. Smart sensors. *Sensors and actuators*, 10(3-4):239–248, 1986.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[Gmb17] IPG Automotive GmbH. ipg-automotive.com. `https://ipg-automotive.com/`, April 2017. Accessed: 2017-04-12.

[Go02] OMG Group and others. Common object request broker architecture: Core specification, 2002.

[Gol16] Volker Gollücke. *Bewertung von Simulationszuständen für eine gezielte Analyse risikoreicher Systeme.* PhD thesis, Dissertation, Oldenburg, Universität Oldenburg, 2016, 2016.

[GPJ+09] Thomas Glotzbach, T. Pfützenreuter, M. Jacobi, A. Voigt, and Thomas Rauschenbach. CViewVR: A High-performance Visualization Tool for Teamoriented Missions of Unmanned Marine Vehicles. In *8th International Conference on Computer Applications and Information Technology in the Maritime Industries (COMPIT2009)*, 2009.

[GPL+12] Volker Gollücke, Jan Pinkowski, Christoph Läsche, Sebastian Gerwinn, and Axel Hahn. Simulation-based completeness analysis and adaption of fault trees. In *SIMUL 2014, The Sixth International Conference on Advances in System Simulation*, page 228 to 235, 2014-10-12.

[GVH03] Brian Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th international conference on advanced robotics*, volume 1, pages 317–323, 2003.

[GVS+01] Brian P. Gerkey, Richard T. Vaughan, Kasper Stoy, Andrew Howard, Gaurav Sukhatme, and Maja J. Mataric. Most valuable player: A robot device server for distributed control. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 3, pages 1226–1231. IEEE, 2001.

[Hen04] Michi Henning. A new approach to object-oriented middleware. *Internet Computing, IEEE*, 8(1):66–75, 2004.

[Hjø11] Birger Hjørland. Theoretical clarity is not'manicheanism': A reply to marcia bates. 2011.

[HL97] David L. Hall and James Llinas. An introduction to multisensor data fusion. 85(1):6–23, 1997.

[Hoe14] Tim Hoerstebrock. *Strategische Analyse der Elektromobilität in der Metropolregion Bremen/Oldenburg: Multi-Agenten basierte Simulation alternativer Antriebssysteme.* PhD thesis, Universität Oldenburg, 2014.

[HS10] S. Hochdorfer and C. Schlegel. 6 DoF SLAM using a ToF camera: The challenge of a continuously growing number of landmarks. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3981–3986, October 2010.

[HS11] Michi Henning and Mark Spruiell. Choosing Middleware: Why Performance and Scalability do (and do not) Matter. Technical report, ZeroC, 2011.

[HSGB15] Axel Hahn, Sören Schweigert, Volker Gollücke, and Carsten Buschmann. Virtual testbed for maritime safety assessment. In *Scientific Journals of the Maritime University of Szczecin*, volume 44, pages 166–122, 2015.

[HTI97] Mei-Chen Hsueh, Timothy K. Tsai, and Ravishankar K. Iyer. Fault injection techniques and tools. 30(4):75–82, 1997.

[IAL06] International Association of Marine Aids toNavigation {and} Lighthouse Authorities IALA. IALA recommendation v-145 on the inter-VTS exchange format (IVEF) service edition 1, 2011-06.

[IAL01] IALA IALA. Guidelines on universal ship-borne automatic identification system (ais). *International Association of Aids to Navigation and Lighthouse Authorities*, 2001.

[IHO00] S IHO. Iho transfer standard for digital hydrographic data, 2000, 2000.

[IHO15] IHO. S-100 - UNIVERSAL HYDROGRAPHIC DATA MODEL. Technical Report Edition 2.0.0, International Hydrographic Organization, MONACO, June 2015.

[IMO01] IMO. *Resolution A.915 (22): REVISED MARITIME POLICY AND REQUIREMENTS FOR A FUTURE GLOBAL NAVIGATION SATELLITE SYSTEM (GNSS).* November 2001.

[Inc17] The MathWorks Inc. MATLAB R2017a and Simulink R2017a, 2017.

[IR10] International Telecommunication Union (ITU-R). Recommendation ITU-r m.1371-4,technical characteristics for an automatic identification system using time-division multiple access in the VHF maritime mobile band. 2010.

[ISISC00] SISC IEEE Simulation Interoperability Standards Committee. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules, 1516.1-2000. *Inc., New York*, 2000.

[JC10] Krzysztof Janowicz and Michael Compton. The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology. In *SSN*, 2010.

[KBF+13] Christian Kuka, André Bolles, Alexander Funk, Sönke Eilers, Sören Schweigert, Sebastian Gerwinn, and Daniela Nicklas. SaLsA streams: Dynamic context models for autonomous transport vehicles based on multi-sensor fusion. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*. IEEE Explore, 2013.

[KE12] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors*, 12(2):1437–1454, February 2012.

[KH04] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004.

[KH12] Nate Koenig and John Hsu. GAZEBO (Präsentation), 2012.

[KHS12] Scott Koziol, Paul Hasler, and Mike Stilman. Robot path planning using field programmable analog arrays. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1747–1752. IEEE, 2012.

[KN12] Christian Kuka and Daniela Nicklas. Approximating complex sensor quality using failure probability intervals. In *Scalable Uncertainty Management*, volume 7520 of *Lecture Notes in Computer Science*, pages 287–298. Springer Berlin Heidelberg, 2012.

[Kon09] Kongsberg. Kongsberg Maritime Simulation & Training Ship's Bridge Simulator. Brochure, 2009. Accessed: 2017-04-12.

[Kuk15] Christian Kuka. *Qualitätssensitive Datenstromverarbeitung zur Erstellung von dynamischen Kontextmodellen*. PhD thesis, Carl von Ossietzky Universität Oldenburg, Oldenburg, 2015.

[LCRP+05] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[LFSK06] Ce Liu, William T. Freeman, Richard Szeliski, and Sing Bing Kang. Noise estimation from a single image. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 901–908. IEEE, 2006.

[LGH13] Christoph Läsche, Volker Gollücke, and Axel Hahn. Using an HLA Simulation Environment for Safety Concept Verification of Offshore Operations. In *Proceediings 27th European Conference on Modelling and Simulation*, page 156ff, 2013.

[LHT+11] Laurent Lefort, Cory Henson, Kerry Taylor, Payam Barnaghi, Michael Compton, Oscar Corcho, Raul Garcia-Castro, John Graybeal, Arthur Herzog, Krzysztof Janowicz, and others. Semantic Sensor Network XG Final Report. *W3C Incubator Group Report*, 28, 2011.

[MB16] Karol Majek and Janusz Bedkowski. Range Sensors Simulation Using GPU Ray Tracing. In *Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015*, pages 831–840. Springer, 2016.

[MG15] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[MMH05] Luís Sardinha Monteiro, Terry Moore, and Chris Hill. What is the accuracy of DGPS? 58(2):207–225, 2005.

[MSH15] André Bolles Mazen Salous, Heiko Müller and Axel Hahn. Improving maritime traffic safety by applying routes exchange and automatic relevant radar data exchange. *Scientific Journals of the Maritime University of Szczecin*, 44, 2015.

[Mui90] Patrick F Muir. A virtual sensor approach to robot kinematic identification: theory and experimental implementation. In *Systems Engineering, 1990., IEEE International Conference on*, pages 440–445. IEEE, 1990.

[noa01] *Geocentric Datum of Australia Technical Manual (version 2.2).* Intergovernmental Committee on Surveying and Mapping, 2001. OCLC: 225962814.

[noa17] VIRES - Home. `https://www.vires.com/`, January 2017. Accessed: 2017-01-12.

[OMG11] OMG. *Unified Modeling Language (UML) Version 2.4.1.* OMG, 2011. See also URL \verb+http://www.omg.org/spec/UML/+.

[OMG14] OMG. Information technology - object management group meta object facility (MOF) core, 2014.

[OMG16] OMG. OMG Meta Object Facility (MOF) Core Specification. Technical Report Version 2.5.1, November 2016.

[PFOL12] Stefan Puch, Martin Fränzle, Jan-Patrick Osterloh, and Christoph Lässche. Rapid Virtual-Human-in-the-Loop Simulation with the High Level Architecture. In *Proc. of Summer Computer Simulation Conference (SCSC 2012). The Society for Modeling & Simulation Int., Curran Associates, Inc., Genua*, volume 44, pages 44–50, 2012.

[PTM98] Jelica Protic, Milo Tomasevic, and Veljko Milutinović. *Distributed shared memory: Concepts and systems*, volume 21. John Wiley & Sons, 1998.

[QCG+09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[Ran94] J. Rankin. An error model for sensor simulation GPS and differential GPS. In *Position Location and Navigation Symposium, 1994., IEEE*, pages 260–266. IEEE, 1994.

[Ras06] Rob Raskin. Guide to sweet ontologies. *NASA/Jet Propulsion Lab, Pasadena, CA, USA*, 2006.

[RCH+12] Erwin Roth, Tugkan Calapoglu, Holger Helmich, Kilian Neumann-Cosel, Marc-Oliver Fischer, Andreas Kern, Andreas Heider, and Alois Knoll. Advanced driver assisttance system testing using optix. "conference talk", 2012. Accessed: 2017-01-12.

[RCT+09] Mitchell M. Rohde, Justin Crawford, Matthew Toschlog, Karl D. Iagnemma, Guarav Kewlani, Christopher L. Cummins, Randolph A. Jones, and David A. Horner. An interactive physics-based unmanned ground vehicle simulator leveraging open source gaming technology: progress in the development and application of the virtual autonomous navigation environment (VANE) desktop. pages 73321C–73321C–13, May 2009.

[RHE11] Jürgen Rossmann, Nico Hempe, and Markus Emde. New methods of render-supported sensor simulation in modern real-time vr-simulation systems. In *Proceedings of the 15th WSEAS international conference on Computers*, 2011.

[RJRV08] C. Rosen, U. Jeppsson, L. Rieger, and P. A. Vanrolleghem. Adding realism to simulated sensors and actuators. 57(3):337, 2008.

[RM04] Stefan Roiser and P. Mato. The SEAL C++ reflection system. *Computing in High Energy and Nuclear Physics (CHEP'04), Interlaken, Switzerland*, 2004.

[RN11] Stefan Rührup and Thomas Neugebauer. Projekt SaLsA: Automatisierte Fahrzeuge im Aussenbereich. *Hebezeuge Fördermittel*, 2011.

[RSJ+05] Jan Rexilius, Wolf Spindler, Julien Jomier, Matthias König, Horst Hahn, Florian Link, and Heinz-Otto Peitgen. A framework for algorithm evaluation and clinical application prototyping using ITK. *The Insight Journal*, 2005.

[Saf12] Lily Safie. *A component model that is both control-driven and data-driven.* PhD thesis, University of Manchester, Manchester, 2012.

[SBMP08] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: Eclipse Modeling Framework.* Addison-Wesley Professional, second edition edition, 2008.

[SGHB14] Sören Schweigert, Volker Gollücke, Axel Hahn, and André Bolles. HAGGIS: A modelling and simulation platform for e-Maritime technology assessment. page 10, Istanbul, Turkey, October 2014.

[SHB14] Arne Stasch, Axel Hahn, and André Bolles. LABSKAUS - A physical platform for e-Maritime technology assessment. In *Proceedings of 2nd International Symposium of Naval Architecture and Maritime*, pages 742–752, Istanbul, Turkey, 2014.

[SHK+14] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *Pattern Recognition*, pages 31–42. Springer, 2014.

[Sie01] Mel Siegel. Sensor modeling and simulation: can it pass the turing test? In *Virtual and Intelligent Measurement Systems, 2001, IEEE International Workshop on. VIMS 2001*, pages 92–96. IEEE, 2001.

[Sie03] Mel Siegel. The sense-think-act paradigm revisited. In *Robotic Sensing, 2003. ROSE'03. 1st International Workshop on*, pages 5–pp. IEEE, 2003.

[Sig14] Signalis. Styris Vts. Brochure, 2014.

[Sko81] Merrill Ivan Skolnik. *INTRODUCTION TO RADAR SYSTEMS*. Number Second Edition. McGRAW-HILL BOOK COMPANY, 1981.

[SMB14] Robin Schubert, Norman Mattern, and Roy Bours. Simulation of Sensor Models for the Evaluation of Advanced Driver Assistance Systems. *ATZ-elektronik worldwide*, 9(3):26–29, 2014.

[SOL02] SOLAS. Solas chapter v, regulation 19.2, carriage requirements for ship-borne navigational systems and equipment. 2002.

[SS02] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[Sta14] Johannes Stallkamp. *Framework zur Entwicklung, Bewertung und Analyse von Computer-Vision-Anwendungen im Kontext umfelderfassender Fahrerassistenzsysteme*. PhD thesis, Ruhr-Universität Bochum, 2014.

[Stü04] Dirk Stüker. *Heterogene Sensordatenfusion zur robusten Objektverfolgung im automobilen Straßenverkehr*. PhD thesis, Universität Oldenburg, 2004.

[SV09] Tijn Schmits and Arnoud Visser. An omnidirectional camera simulation for the USARSim World. In *RoboCup 2008: Robot Soccer World Cup XII*, pages 296–307. Springer, 2009.

[TB07] J. Trompeter and J.C.F. Beltran. *Modellgetriebene Softwareentwicklung: MDA und MDSD in der Praxis*. Entwickler.Press, 2007.

[TE14] Hans Anton Tvete and oystein Engelhardtsen. DNV GL's research within Autonomous Systems, 2014. Presentation.

[Tur97] K Turner. What is the difference among 2d, 2.5 d, 3d and 4d. *GIs World*, 10(3):54, 1997.

[TVH05] John Tuley, Nicolas Vandapel, and Martial Hebert. Analysis and removal of artifacts in 3-d ladar data. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2203–2210. IEEE, 2005.

[TW11] Andrew S. Tanenbaum and D. Wetherall. *Computer networks*. Pearson Prentice Hall, Boston, 5th ed edition, 2011. OCLC: ocn660087726.

[UBl09] UBlox. GPS compendium - essentials of satellite navigation, 2009.

[VHVS+11] Sofie Van Hoecke, Steven Verstockt, Koen Samyn, Mike Slembrouck, and Rik Van de Walle. Tunnel simulator for traffic video detection. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 49–54, 2011.

[VIM08] ISO VIM. International vocabulary of basic and general terms in metrology (VIM). (3), 2008.

[VZ01] Jakob J Van Zyl. The shuttle radar topography mission (SRTM): a breakthrough in remote sensing of topography. 48(5):559–565, 2001.

[WB08] Jijun Wang and Stephen Balakirsky. USARSim V3. 1.3-A Game-based Simulation of mobile robots.[Online]. 3, 2008.

[WWT06] Keith D Ward, Simon Watts, and Robert JA Tough. *Sea clutter: scattering, the K distribution and radar performance*, volume 20. IET, 2006.

[YB02] Cang Ye and Johann Borenstein. Characterization of a 2-d laser scanner for mobile robot obstacle negotiation. In *ICRA*, pages 2512–2518, 2002.

[YKT+98] Yasushi Yamaguchi, Anne B Kahle, Hiroji Tsu, Toru Kawakami, and Moshe Pniel. Overview of advanced spaceborne thermal emission and reflection radiometer (aster). *Geoscience and Remote Sensing, IEEE Transactions on*, 36(4):1062–1071, 1998.

Eidesstattliche Erklärung

Hiermit versichere ich, Sören Schweigert, dass ich die von mir vorgelegte Arbeit mit dem Titel: 'Simulative Überprüfung von Sensordatenverarbeitungssystemen' selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit einschließlich Tabellen, Karten und Abbildungen, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Sören Schweigert	Datum