Anforderungsanalyse für ein System zur automatisierten Ereignisdetektion in marinen Umgebungen

Iring Paulenz, Daniel Lukats, Janina Schneider, Elmar Berghöfer, Frederic Theodor Stahl, Lars Nolle und Oliver Zielinski

Zusammenfassung

Eine Vielzahl von Multisensor-Systemen, unter anderem Drifter, Bojen und auf Schiffen mitgeführte Messsysteme, aber auch in-situ Sensorsysteme, überwachen Meere und Küstengebiete [1, 2] (Baschek et al. in Ocean Science 13:379–410, 2017; Reuter et al. in Ocean Dynamics 59:195–211, April 2009). Zwar können etwaige

I. Paulenz (✉) · L. Nolle
Fachbereich Ingenieurwissenschaften, Jade Hochschule, Wilhelmshaven, Deutschland
E-Mail: iring.paulenz@jade-hs.de

L. Nolle
E-Mail: lars.nolle@jade-hs.de

D. Lukats · J. Schneider · E. Berghöfer · F. T. Stahl · O. Zielinski
Marine Perception, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Oldenburg, Deutschland
E-Mail: daniel.lukats@dfki.de

J. Schneider
E-Mail: janina.schneider@dfki.de

E. Berghöfer
E-Mail: elmar.berghoefer@dfki.de

F. T. Stahl
E-Mail: frederic_theodor.stahl@dfki.de

O. Zielinski
E-Mail: oliver.zielinski@uol.de

O. Zielinski
Zentrum für Marine Sensorik, Universität Oldenburg, Wilhelmshaven, Deutschland

F. Fuchs-Kittowski et al. (Hrsg.), *Umweltinformationssysteme – Vielfalt, Offenheit, Komplexität,* https://doi.org/10.1007/978-3-658-39796-8_10

Umweltparameter in Echtzeit aufgenommen werden, allerdings ist eine ununterbrochene Überwachung dieser Parameter durch den Menschen unrealistisch. Zudem kann die Übertragung der Daten aufgrund hoher finanzieller Kosten oder hohen Energiebedarfs womöglich nicht in Echtzeit erfolgen, sodass interessante Ereignisse möglicherweise erst verspätet wahrgenommen werden. Infolgedessen ist es in manchen Fällen nicht möglich auf ein Ereignis zu reagieren, zum Beispiel durch Entnahme einer Probe zur späteren Laboruntersuchung. Diese Problemstellung wird innerhalb des Forschungsprojektes ChESS (englisch *Change Event based Sensor Sampling*) adressiert. ChESS überwacht Datenströme aus Multisensor-Systemen in Echtzeit und wendet Methoden der Künstlichen Intelligenz an, um Ereignisse frühzeitig zu erkennen und vordefinierte Aktionen auszulösen. Im Anwendungsfall des Küstenobservatoriums Spiekeroog etwa kann ChESS eine automatisierte Entnahme einer Wasserprobe veranlassen. ChESS ist nicht auf den Einsatz auf Spiekeroog beschränkt, sondern soll ebenso in anderen Umgebungen und Forschungsgebieten unterstützen. Ein erster Schritt zur Integration von ChESS in bestehende Multisensor-Systeme ist eine Anforderungsanalyse. Zur Erstellung dieser wurden Interviews mit identifizierten Stakeholdern geführt und anschließend ausgewertet.

Schlüsselwörter

Ereignisdetektion · Anforderungsanalyse · Küstenobservatorium · Künstliche Intelligenz · Multisensor-Systeme · Datenstromverarbeitung

1 Einleitung

An unserer Lebensqualität tragen Ozeane und Küstengebiete einen erheblichen Anteil. Sie versorgen uns mit Lebensmitteln, sind für die Regulierung unseres Klimas verantwortlich und dienen der Energieerzeugung. Außerdem sind sie Lebensraum für eine Vielzahl verschiedener Tier- und Pflanzenarten. Diese biologische Diversität ist durch die anthropogenen Veränderungen der Ökosysteme bedroht [3, 4]. Der Schutz der Meere ist daher essenziell wichtig. Neben der Erhaltung von Leben, kann durch den Schutz der Ozeane auch eine nachhaltige Nutzung ihrer Ressourcen erreicht werden [5]. Um die Gesundheit der Ökosysteme und potenzielle Gefahren angemessen zu überwachen, sind Echtzeitmessungen von biologischen, chemischen und geologischen Parametern, sowie Meeresschadstoffen in verschiedenen räumlichen Maßstäben erforderlich. Zur Überwachung und Zustandserfassung werden Multisensor-Systeme genutzt. Diese umfassen unter anderem Drifter, Bojen und auf Schiffen mitgeführte Messsysteme, aber auch in-situ Sensorsysteme [1, 2]. Der Fokus bei der Weiterentwicklung solcher Systeme liegt unter anderem darauf, adäquate Abtastraten zu erreichen und neue Sensoren nahtlos in bestehende Systeme zu integrieren [6]. Analog dazu werden Kosten, Leistungsaufnahme und bauliche Größe der Sensoren stetig gesenkt, während die Sensorempfindlichkeit

erhöht wird. Autonom und autark arbeitende Systeme sind auf Künstliche Intelligenz (KI) oder adaptive Modelle, basierend auf den gemessenen Daten, angewiesen [7–9]. In [10] wird ein solches System in Form eines autonomen Unterwasserfahrzeuges verwendet, um eine Wasserzone in einem Küstengewässer zu verfolgen. Dazu werden Differenzen von Wassertemperaturen in verschiedenen Wassertiefen in Echtzeit ausgewertet. Die Differenzen innerhalb der Wasserzone unterscheiden sich stark zu denen in angrenzenden Zonen. Das Ergebnis der Auswertung bestimmt den Kurs des Unterwasserfahrzeuges.

Wie bereits erwähnt, ermöglichen die genannten Sensor-Systeme Echtzeitmessungen. Die Echtzeitüberwachung der gemessenen Parameter durch den Menschen kommt allerdings nur selten vor, zum Beispiel, wenn eine bestimmte Messstandort anhand von Parametern gesucht wird. Der Prozess der Informationsgewinnung anhand von Messdaten lässt sich grob in zwei Schritte unterteilen: (1) Tätigung der Messung, zum Beispiel während einer Messkampagne zur See, (2) Auswertung der gewonnenen Daten zu einem späteren Zeitpunkt, zum Beispiel an Land in einer Forschungseinrichtung. Dieses Vorgehen birgt den Nachteil, dass interessante und wissenschaftlich relevante Ereignisse erst relativ spät erkannt werden.

In den Küstengebieten der Nordsee könnte die einsetzende Algenblüte beispielsweise ein relevantes Ereignis darstellen. Es kann von wissenschaftlichem Interesse sein, unmittelbar vor dem Eintreten eines solchen Ereignisses die Abtastraten von Sensoren zu erhöhen und die Gewässer am Messstandort häufiger zu beproben. Die Detektion eines Ereignisses ist häufig abhängig von mehreren Parametern unterschiedlicher Sensoren. Bezogen auf die einsetzende Algenblüte könnten unter anderem die Parameter Chlorophyllgehalt, Nähr- und Sauerstoffgehalt, sowie die Wassertemperatur entscheidend sein, um den Zeitpunkt zu bestimmen.

Die Ereignisdetektion in Abhängigkeit zu verschiedenen Umweltparametern wird durch das ChESS-System adressiert. ChESS ist ein Akronym für „**Ch**ange **E**vent based **S**ensor **S**ampling“. Die Softwarearchitektur des Systems befindet sich zurzeit in der Planungsphase. Einige Softwaremodule sind bereits als Prototypen implementiert. Als konkretes Anwendungsbeispiel dient ein Multisensor-System des Küstenobservatoriums Spiekeroog [11]. Hier soll ChESS im Echtzeitbetrieb Ereignisse anhand verschiedener Umweltparameter erkennen. Bestimmte Ereignisse, zum Beispiel eine bevorstehende Algenblüte oder Stürme, sollen am Messpfahl vor der Insel zeitnah einen Wasserprobensammler auslösen, sodass die Beprobung während des Ereignisses erfolgt. Zur Detektion der Ereignisse soll ChESS unüberwachte Verfahren des maschinellen Lernens nutzen; insbesondere Algorithmen der Ausreißererkennung für kürzere und Algorithmen zur Detektion von Konzeptänderungen für länger andauernde Ereignisse, siehe Abschn. 2. Lediglich die Relevanz der detektierten Ereignisse muss von wissenschaftlicher Seite aus bewertet werden. Sie ist also abhängig von Expertenwissen. Das geplante System ist nicht auf den Einsatz auf Spiekeroog beschränkt, sondern soll auch in anderen Umgebungen und Forschungsgebieten Unterstützung bieten. Zu diesem Zweck wurde eine Anforderungsanalyse erstellt, die weitere Forschungsumgebungen berücksichtigt.

Dieser Beitrag ist wie folgt aufgebaut: Abschn. 2 geht auf die geplante Softwarearchitektur des Systems und einzelne Funktionalitäten sowie einige für ChESS relevante Algorithmen des maschinellen Lernens ein. Der Anforderungsanalyse gehen wichtige Schritte voraus, die bei ihrer Erstellung helfen. Diese werden in Abschn. 3 betrachtet. Die wichtigsten Erkenntnisse der Anforderungsanalyse werden in Abschn. 4 beschrieben. Der Beitrag endet mit einem Fazit und einem kurzen Ausblick.

2 Das geplante ChESS-System

Abb. 1 zeigt die geplante Architektur des Systems.

Die Funktionsblöcke werden jeweils in einem Unterabschnitt beschrieben. Unterabschnitt 2.3 beschreibt in aller Kürze die Ausgänge des Systems, welche die Ergebnisse der Funktionsblöcke retournieren. Teilaspekte der Softwarearchitektur werden in Unterabschn. 2.4 behandelt.

2.1 Datenvorverarbeitung

Ganz links in Abb. 1 fließen die Daten von den Quellen aus in das System. Im ersten Funktionsblock, links in Abb. 1, findet eine Vorverarbeitung der Daten statt, beginnend mit einer Plausibilitätsprüfung. Diese Prüfung bezieht sich nur auf die Wertebereiche der einfließenden Messwerte. Es wird festgestellt, ob ein Sensor einen fehlerhaften Wert in Abhängigkeit zu seiner Wertespezifikation liefert. Fehlerhafte Werte werden aussortiert. Zusätzlich kann es passieren, dass einzelne Sensoren ausfallen und gar keinen Wert liefern. Fehlende Werte werden durch das Modul „Missing Value Imputation“ ersetzt. Bisher wird ein fehlender Wert durch den zuletzt gesehenen Wert ersetzt. Dieses Verfahren ist jedoch für länger anhaltende Lücken im Datenstrom ungeeignet. Diese ergeben

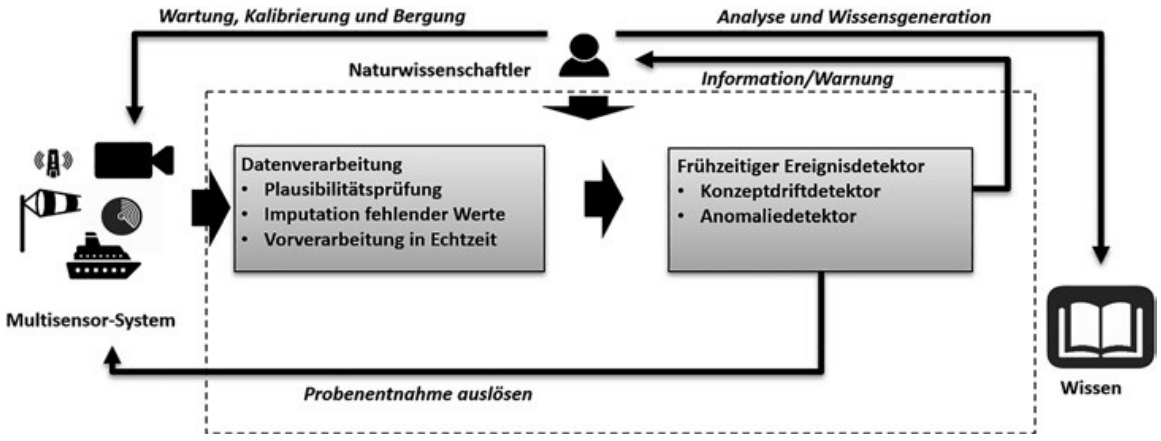

Abb. 1 Initialer Entwurf der Systemarchitektur von ChESS

sich, wenn Sensoren an schwer zugänglichen Standorten ausfallen. Statt den fehlenden Wert durch den zuletzt gesehenen zu ersetzen, müsste er in Abhängigkeit von weiteren Umweltparametern rekonstruiert werden. Vorstellbar sind Regressionsmethoden, die das Framework „River“ anbietet [12], oder multivariate Imputationsalgorithmen wie kNN-dtw [13]. Eventuell sind weitere Vorverarbeitungsschritte notwendig. Beispielsweise benötigen Modelle, welche durch maschinelle Lernverfahren erstellt wurden, häufig normalisierte Daten.

Die im ChESS-System eingesetzten maschinellen Lernmodelle benötigen mehrere Parameter aus unterschiedlichen Quellen, um eine Vorhersage treffen zu können. Dieser Umstand wurde bereits in Abschn. 1 angesprochen. Mehrere Werte unterschiedlicher Sensoren werden in einer Dateninstanz zusammengefasst. Messungen verschiedener Umweltparameter an einem Standort machen nur Sinn, wenn die Messzeiten der einzelnen Sensoren synchronisiert sind. Andernfalls ist es schwierig, wenn nicht gar unmöglich, exakte Zusammenhänge innerhalb der Daten abzuleiten. Im einfachsten Falle lassen sich die Messzeitpunkte aller Sensoren synchronisieren und die Daten in fester Reihenfolge in das System übertragen. Im schlechtesten Falle sind die Messzeitpunkte verschiedener Sensoren asynchron und die Daten lassen sich nicht in Messreihenfolge in ein System überführen, zum Beispiel wenn die Sensoren räumlich weit auseinander liegen und es Übertragungsverzögerungen gibt [14]. Die beschriebenen Zusammenhänge sind in Abb. 2 dargestellt.

Die Probleme, die die Bildung von Instanzen innerhalb eines nicht endenden Datenflusses mit sich bringt, werden von sogenannten „Data Processing“ Frameworks gelöst [15–17]. Das Verfahren zur zeitlichen Erfassung von Verzögerungen beruht dabei auf heuristischen Funktionen. Die einzelnen Knoten des Systems, die abgerundeten Rechtecke innerhalb des ChESS-Systems in Abb. 2 (stark vereinfacht), werden von sogenannten „Watermarks“ periodisch über Zeitpunkte im Datenfluss informiert [18]. Der Zeitstempel im „Watermark“ impliziert dabei die Annahme, dass in dem auf ihn folgenden Datenfluss keine kleineren Zeitstempel vorhanden sind. In Abb. 2 werden „Watermarks“ in dem

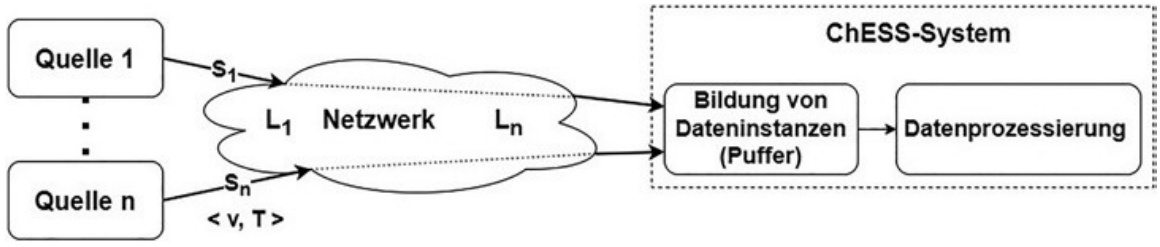

Abb. 2 ChESS-System mit Netzwerkanbindung nach [14]. Die Quellen 1 … n liefern die Datenströme S_1 … S_n. Diese enthalten Datenwerte (**v**alue) und Zeitpunkte (**T**imestamp) der Aufnahmen <v, T>. Die Übertragung wird durch die Latenzen des Netzwerkes L_1 … L_n verzögert. Anhand der Zeitpunkte des Eintreffens im System und der Zeitstempel in den Daten lassen sich Dateninstanzen bilden, dafür müssen Daten gepuffert werden. Anschließend erfolgt die Prozessierung durch die Algorithmen des Systems

Knoten am Eingang des Systems (Puffer) erzeugt und an den nächsten Knoten weitergeleitet [14, 17]. Die Betrachtung der zeitlichen Komponente macht es dem System überhaupt erst möglich, fehlende Werte zu detektieren und sie anschließend zu ersetzen [17]. Neben der zeitlichen Komponente können auch andere Parameter, wie etwa unterschiedliche Sensortypen, zur Bildung von Instanzen herangezogen werden. Auch eine Verknüpfung von Parametern und zeitlicher Komponente ist möglich. Je nach verwendetem Framework variiert die Unterstützungstiefe zur Fensterung der Daten in Hinblick auf die Benutzerfreundlichkeit [16]. Das zu implementierende „Data Processing“ Framework muss den Anforderungen dieser Analyse genügen.

2.2 Ereignisdetektor

Im zweiten Funktionsblock – in Abb. 1 auf der rechten Seite – detektieren Algorithmen des maschinellen Lernens Konzeptänderungen und Anomalien in den Daten. Im Gegensatz zu Anomalien und Ausreißern, die lediglich wenige Zeitpunkte andauern, bezeichnen Konzeptänderungen langfristige Veränderungen in den Mustern des Datenstroms [19]. Das ChESS-System beruht auf der Idee, dass die Erkennung von Konzeptänderungen (1) einen allgemein gültigen Ansatz im Bereich des maschinellen Lernens darstellt und (2) maschinelle Lernmodelle eine gesteigerte Performance (Genauigkeit) aufweisen, wenn sie nur einem bestimmten Konzept entsprechen. Das Küstenobservatorium Spiekeroog stellt Daten aus Langzeitmessungen zur Verfügung, in welchen Konzeptänderungen und Anomalien in einem hohen Maße auftreten, dies konnte in zwei Studien empirisch evaluiert werden [20, 21]. Beide Autorengruppen zeigten das Vorhandensein und die Möglichkeit der Detektion von Anomalien auf. Erstere anhand von Wetterberichten des Deutschen Wetterdienstes für die Nordsee [20], letztere auf Basis von Expertenwissen [21]. Zudem demonstrierten Lukats et al., dass Konzeptänderungen in den Daten präsent sind, indem sie die Genauigkeit eines Regressionsmodells mit und ohne Zuhilfenahme eines Algorithmus zur Detektion von Konzeptänderungen verglichen [20]. Da die Genauigkeit bei Anpassungen des Modells stieg, ist die Anwesenheit von Konzeptänderungen naheliegend. Der Vorteil von maschinellen Lernmethoden besteht darin, dass die generellen Charakteristika von Ereignissen von einem Modell ermittelt werden. Somit müssen die Ereignisse nicht zuvor von einem Experten durch manuell aufgestellte Regeln definiert werden.

Kurzfristig auftretende Veränderungen in den Daten werden durch Algorithmen zur Erkennung von Anomalien und Ausreißern adressiert. Für die Echtzeitanalyse können Online-Algorithmen zur Erkennung von Ausreißern verwendet werden. Diese passen sich implizit an die aktuellen Daten an. Im ChESS-System werden solche Algorithmen, wie etwa Robust Random Cut Forest [22] oder Half-Space Trees [23], durch das Framework „Python Streaming Anomaly Detection (PySAD)“ [24] bereitgestellt.

Konzeptänderungen hingegen lassen sich als langfristige oder periodisch auftretende Veränderungen beschreiben. Bezogen auf die Meeresforschung ergeben sich diese Ver-

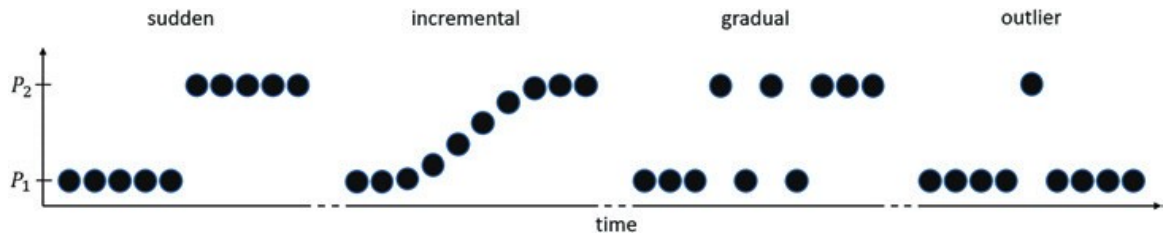

Abb. 3 Schematische Beispiele für verschiedene Typen von Konzeptabweichungen („sudden“, „incremental“ sowie „gradual“) und eines Ausreißers, basierend auf einer Abbildung in [19]

änderungen, unter anderem, aus den kosmologischen Eigenschaften unseres Planetensystems, also dem Rhythmus der Jahres- und Tageszeiten, sowie den Gezeiten des Meeres. Diese bedingen eine Veränderung der gemessenen Werte, wie zum Beispiele Temperaturen und Wasserpegelstand. Die Veränderungen können auch abhängig von meteorologischen und biochemischen Vorgängen sein, Beispiele dafür sind Stürme oder eine einsetzende Algenblüte. Nicht zu vernachlässigen ist auch die Umgebung, in welcher Sensoren angebracht sind. Lagern sich in Messvorrichtungen Sedimente ab oder tritt Salzwasser ein, führt dies zu einer Veränderung der gemessenen Werte. Grundsätzlich lassen sich drei Typen von Konzeptänderungen unterscheiden [19]. Diese und das Auftreten eines Ausreißers sind in Abb. 3 dargestellt.

Abb. 3 zeigt zwei Konzepte P1 und P2, welchen Daten angehören können. Die Konzeptänderung kann plötzlich eintreten und zu einer sofortigen Änderung der Muster im Datenstrom führen. Aus inkrementellen Veränderungen ergibt sich ein fließender Übergang von einem zum anderen Konzept. Die Veränderungen können auch allmählich auftreten, was bedeutet, dass die Daten für eine bestimmte Zeit von einem der beiden Konzepte stammen. Im Gegensatz dazu ist ein Ausreißer definiert als ein einzelner oder sehr wenige Datenpunkte, die nicht dem allgemeinen Muster des Datenstroms entsprechen.

Ein Prototyp zur Konzeptänderungsdetektion ist mittels des KI-Frameworks „River“ implementiert [12]. Für den Prototyp wurde eine Auswahl von Algorithmen zur unüberwachten Erkennung von Konzeptabweichungen implementiert. Einige der für das ChESS-System in Betracht gezogenen Algorithmen sind: „Discriminative Drift Detector“ [25], „Ensemble Drift Detection With Feature Subspaces“ [26] und „Image-based Drift Detector“ [27] – der auch auf Daten operieren kann, die keine Bilder sind.

Die in diesem Abschnitt genannten Prototypen werden im weiteren Verlauf des Projektes im Detail evaluiert.

2.3 Ausgänge

Die Ausgänge des ChESS-Systems sind in Abb. 1 als schmale schwarze Pfeile dargestellt. Sie dienen dazu, Informationen aus dem System, in Abb. 1 begrenzt durch das gestrichelte Rechteck, zu führen. Sie informieren über das Auftreten relevanter Ereig-

nisse und sind Auslöser für weitere Systeme, wie zum Beispiel eines Wasserprobensammlers. Die Ausgänge sollen aber auch die Abtastraten verschiedener Sensoren einstellen. Eine Visualisierung der einfließenden Daten ist geplant, wobei relevante Ereignisse markiert werden. Die gewonnenen Informationen sollen dabei helfen, neues Wissen zu generieren, Messsysteme zu warten und die Konfiguration des ChESS-Systems anzupassen, sowie auf Ereignisse rechtzeitig zu reagieren.

2.4 Angestrebte Softwarearchitektur

Die Softwarearchitektur befindet sich in der Planungsphase und ist letztlich stark abhängig von der Anforderungsanalyse, daher werden in diesem Unterabschnitt nur die grundsätzlichen Überlegungen präsentiert.

Die Prozessierung der Daten in Echtzeit hat höchste Priorität im System: Um zu gewährleisten, dass ChESS alle vom Sensorsystem erhobenen Daten analysieren kann, muss jede Dateninstanz vor dem Eintreffen der nächsten verarbeitet werden.

Insbesondere autark und autonom arbeitende Mess- und Robotersysteme, wie etwa das in Abschn. 1 genannte autonome Unterwasserfahrzeug, sollen von ChESS profitieren. Sowohl der räumliche Platz als auch die Stromversorgung sind bei solchen Systemen häufig stark limitiert. Gleichzeitig erfolgt die Datenauswertung oft lokal, innerhalb des Systems in Echtzeit [10]. Dadurch können die Kosten der Datenübertragung eingespart werden, welche je nach Einsatzgebiet unterschiedlich hoch ausfallen. Genannte Limitierungen verhindern den Einsatz von Rechnerclustern. Im schlimmsten Fall ist das System sogar auf rechenleistungsarme Einplatinencomputer angewiesen. Die nachträgliche Betrachtung von Daten oder eine nachträgliche zeitlich verzögerte Korrektur des Ergebnisses ist nicht vorgesehen. Stattdessen fließen die Daten durch das System, werden durch diverse Verarbeitungsschritte verzögert und liefern ein Ergebnis. Diesbezüglich lassen sich zwei Softwarearchitekturen unterscheiden: Lambda und Kappa [28].

Innerhalb der Lambda-Architektur wird die Prozessierung des Datenstroms von zwei getrennten und voneinander unabhängigen Systemen übernommen. Die Daten werden zunächst in Echtzeit ausgewertet. Dabei lässt sich einstellen, wie genau das System arbeitet, wodurch letztlich die zeitliche Performance bestimmt wird. Das zweite System liefert zu einem oft sehr viel späteren Zeitpunkt ein genaueres Ergebnis [15, 28]. Die neuere Kappa-Architektur setzt auf ein einziges echtzeitfähiges System. Eine spätere Korrektur der Ergebnisse entfällt oder ist auf kurze zeitliche Verzögerungen des Datenflusses begrenzt. Im Idealfall entfällt die Speicherung der einzelnen Sensorwerte auf einer Festplatte. Verarbeitete Daten können gelöscht werden [16, 28]. Das geplante ChESS-System setzt auf die Kappa-Architektur, da die nachträgliche Korrektur von Ergebnissen keinen Mehrwert für Systeme hat, die in Echtzeit ausgelöst werden müssen. Daraus folgt, dass die Erstellung einer Datenbank in die Zuständigkeit des Anwenders fällt und nicht durch das ChESS-System durchgeführt wird.

Neben der angesprochenen äußeren Struktur der Softwarearchitektur gibt es auch Komponenten innerhalb des Systems. Sie verbinden die in Abb. 1 gezeigten Softwaremodule und lassen sich erst nach einer Anforderungsanalyse betrachten. Aufgrund der frühen Planungsphase des Systems wird auf sie nicht weiter eingegangen.

3 Methodik

ChESS muss in bestehende Sensorsysteme und Arbeitsflüsse, wie etwa beim Küstenobservatorium Spiekeroog, möglichst reibungslos integriert werden. Dazu wurden bisher fünf Stakeholder identifiziert. Jeweils zwei Stakeholder gehören den Bereichen Meeresforschung und Fernerkundung an. Hierin enthalten ist auch ein Team des Küstenobservatoriums Spiekeroog. Für eine erste Implementierung des ChESS-Systems sind die Visionen und Vorgaben dieses Teams von besonderer Relevanz. Der Projektantrag sieht vor, dass ChESS-System in die bestehenden Systeme dieses Teams zu integrieren. Ein weiterer Stakeholder stammt aus dem Bereich Maschinenbau.

Für die Anforderungsanalyse wurde mit jedem Stakeholder ein Interview durchgeführt. Die Interviews wurden für eine spätere Auswertung aufgezeichnet. Als Vorbereitung auf die Interviews erhielten die Stakeholder eine Zusammenfassung des Projektes ChESS, sowie einen Fragekatalog, welcher als Gesprächsleitfaden diente. Der Fragenkatalog enthält 20 Fragen aus den folgenden Themenfeldern:

- Identifizierung der zu verarbeitenden Daten
- Hardwareanbindung, Computersysteme und Systeme zur Datenübertragung
- Bereits genutzte Software, mit der ChESS interagieren muss
- Detektion von relevanten Ereignissen und Verhalten, zum Beispiel Auslösen von Aktuatoren innerhalb eines festen zeitlichen Rahmens

Ein besonderes Augenmerk lag darauf zu erfahren, ob die in Abschn. 2 beschriebenen Funktionsblöcke von den Stakeholdern benötigt werden. Etwaige Fehler im Datenstrom, sowie das Vorhandensein von Anomalien und Konzeptänderungen würden den initialen Entwurf des Systems, Abb. 1, bestätigen.

Für die Softwareentwicklung des ChESS-Systems wird ein iterativ-inkrementelles Modell genutzt [29]. Es ist daher nicht zwingend notwendig ein Pflichten- und Lastenheft anzulegen [30]. Einzelne geforderte Funktionen werden nach und nach evaluiert. Dazu wird in regelmäßigen Abständen Rücksprache mit den Stakeholdern gehalten. Die Anforderungen werden schriftlich festgehalten, zusätzlich werden Grafiken mit Anwendungsbeispielen erstellt. Abb. 4 zeigt das Vorgehen mit einem iterativ-inkrementellen Entwicklungsmodell. Die Softwareentwicklung erfolgt unter Einsatz des Versionsverwaltungstools „Git“.

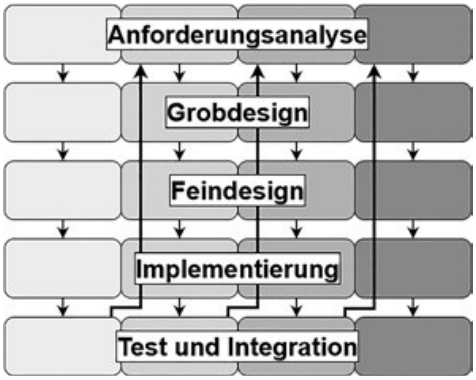

Abb. 4 Iterativ-Inkrementelle Entwicklung nach Kleuker [29]. Zu jedem Zeitpunkt existiert eine funktionsfähige Version des Produkts, welches in jedem Bearbeitungsdurchlauf innerhalb der dargestellten Schritte um neue Funktionalitäten erweitert wird

4 Anforderungsanalyse

Die identifizierten Anforderungen lassen sich in funktionale und nicht-funktionale Anforderungen unterteilen. Dabei spezifizieren die funktionalen Anforderungen „was“ das System leisten soll, bezogen auf die einzelnen Verarbeitungsschritte. Nicht-funktionale Anforderungen beschreiben zum einen Qualitätsanforderungen, beispielsweise Zuverlässigkeit, Sicherheit, Verfügbarkeit, Wartbarkeit und Portabilität, und zum anderen technische Anforderungen wie Interoperabilität, Reaktionszeit und Verhalten, welches alle funktionalen Anforderungen betrifft [30].

Es folgen zwei Unterabschnitte, welche jeweils die gefundenen funktionalen und nicht-funktionalen Anforderungen beschreiben. Auch das Projektteam selbst stellt einen Stakeholder dar, daher sind Ideen und Thesen, welche sich in internen Sitzungen ergaben, ebenfalls in die Auswertung eingeflossen.

4.1 Funktionale Anforderungen

Die Stakeholder bestätigten unisono den initialen Entwurf des ChESS-Systems. Deutlich fiel dabei die Notwendigkeit einer Erkennung von Konzeptänderungen, als essenzieller Bestandteil der Datenverarbeitung, aus. Bestätigt wurde ebenfalls, dass relevante Ereignisse von mehreren Messgrößen abhängig sind. Außerdem äußerten die Stakeholder den Wunsch nach einer Visualisierung der Daten, inklusive der Markierung von Anomalien, Ausreißern und fehlenden bzw. fehlerhaften Werten, welche ersetzt wurden. Ferner forderte ein Stakeholder, dass die auf Künstlicher Intelligenz basierenden Ergebnisse erklärbar sein müssen.

Es ergaben sich größere Diskrepanzen bezüglich der Vorverarbeitung. Die Stakeholder aus den Bereichen Fernerkundung und Maschinenbau gaben an, dass ihr Datenfluss frei von fehlerhaften und fehlenden Werten sei. Dies wird durch den Einsatz von industriellen Systemen, wie zum Beispiel „Industrial Edge“ (https://www.siemens.com/

global/en/products/automation/topic-areas/industrial-edge.html) der Firma „Siemens“, erreicht. Daneben wird für die verlustfreie Datenübertragung auch die freie Software „Kafka“ [31] der „Apache Software Foundation“ eingesetzt. Einen Spezialfall stellt die Simulation von Werkzeugmaschinen dar, hier ist der Datenfluss ebenfalls fehlerfrei. In solchen Fällen wird das Modul „Missing Value Imputation“ nicht benötigt. Einzelne Parameter kommen bereits zusammengefasst bei den Modulen zur Auswertung an, sodass es nicht notwendig ist eine Dateninstanz zu bilden. Die Parameter sind domänenspezifisch. Je nach Forschungsgebiet werden also vollkommen unterschiedliche Vorverarbeitungsschritte benötigt. Im Bereich Maschinenbau soll zum Beispiel die Abnutzung von Werkzeugmaschinen mithilfe von Audiodaten in Echtzeit erkannt werden. Dazu werden die Audiodaten in einem ersten Schritt einer Fast-Fourier-Transformation unterzogen, um die Amplituden und Phasen eines Frequenzspektrums zu bestimmen [32]. Änderungen im Frequenzspektrum lassen Rückschlüsse über den Grad der Abnutzung zu. Die Möglichkeit zur benutzerfreundlichen Implementierung verschiedener Verfahren muss also durch das eingesetzte „Data Processing“ Framework gegeben sein.

Es stellte sich heraus, dass neben der Vorverarbeitung am Eingang des ChESS-Systems auch eine Nachprüfung am Ausgang notwendig ist. Dies betrifft die Ausgänge zum Auslösen weiterer Systeme. Diese können durch diverse physikalische Gegebenheiten limitiert sein. Als Beispiel sei hier die Ausrichtung eines Elektrooptischen/Infrarot-Systems, kurz EO/IR, in dem Bereich der Fernerkundung erwähnt. Diese ist abhängig von verschiedenen Parametern, wie gegenwärtige Position des Sensors, Flughöhe und Geschwindigkeit des Fluggerätes. Soll die Position des Sensors per Künstlicher Intelligenz angesteuert werden, müssen mehrere mögliche Positionen zurückgegeben und anschließend die genannten physikalischen Größen betrachtet werden, um zu einem sinnvollen Ergebnis zu gelangen. Ein etwas einfacheres Beispiel ist der Wasserprobensammler, dessen Probenvorrat begrenzt ist. Er darf also nur angesteuert werden, wenn noch eine freie Probe vorhanden ist. Ist das Auslösen zeitkritisch oder ein Teil der auf Künstlicher Intelligenz basierender Algorithmen zu ungenau, gilt es weitere Spezialfälle zu beachten. Nicht jedes relevante Ereignis muss die Probeentnahme auslösen. Vorstellbar ist auch, dass es mehrere Systeme gibt, die in Abhängigkeit unterschiedlicher Ereignisse ausgelöst werden.

Ein weiterer wichtiger Punkt ergibt sich aus der Notwendigkeit auf Basis der Probenauswertung gegebenenfalls Anpassungen an der Systemkonfiguration vornehmen zu müssen. Dazu müssen die Softwaremodule des Systems anpassbar sein. Hierzu ist es notwendig, das System für kurze Zeit zu pausieren und nach der Änderung weiter laufen zu lassen. Der anfallende Datenstrom muss dabei gepuffert werden, um den Verlust von Messdaten zu verhindern. Bezogen auf die Konzeptänderungen können zum Beispiel maschinelle Lernmodelle eingesetzt werden, welche nur einem saisonalen Konzept entsprechen. Eine Änderung des Konzeptes führt zum automatischen Austausch des Modells. Dies soll die Genauigkeit der Vorhersage verbessern.

In marinen Umgebungen müssen Daten oft manuell aus isolierten Aufzeichnungsgeräten geborgen werden. In diesem Fall müssen Daten auch nachträglich ins ChESS-System eingebunden werden können.

4.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen beschreiben eine Reihe von kritischen Bereichen des zu entwickelnden Systems, die in direkter Abhängigkeit zum eingesetzten „Data Processing“ Framework stehen. Hierzu zählen, unter anderem, Zuverlässigkeit, Sicherheit, Wartbarkeit und Portabilität, sowie technische Anforderungen wie Interoperabilität und Verhalten, welches von allen Softwaremodulen unterstützt werden muss [30].

Die beschriebenen Anforderungen umfassen etwaige Softwareschnittstellen der Stakeholder, die benötigte Reaktionszeit des Systems, unterstützte Betriebssysteme und Mikroprozessor-Architekturen, sowie die Ausfall- und Übertragungssicherheit des Systems.

Bisher nutzen die Stakeholder mannigfaltige Softwareschnittstellen, um die Messdaten in ihr System zu überführen. Diese basieren zum Teil auf Open-Source-Software, viele haben aber auch proprietären Charakter. Ein Hauptaugenmerk des ChESS-Systems liegt deswegen darauf, mit diesen Schnittstellen zu interagieren. Dieses Problem wird direkt durch die Transportebene des „Data Processing“ Frameworks adressiert. Es interagiert als Schnittstelle zu verschiedenen Systemen, wie zum Beispiel Datenbanken aber auch Transportprotokollen. Bei den Stakeholdern kommen neben lokalen kabelgebundenen Datennetzen auch kabellose Systeme zum Einsatz. In letzteren kann es, aufgrund der Eigenschaften solcher Netzwerke, zu einer größeren Verzögerungszeit bei der Datenübertragung kommen. Diesbezüglich muss das Framework geeignete Verfahren besitzen, um Verzögerungszeiten zu berücksichtigen und Dateninstanzen aus den einzelnen übertragenen Sensorwerten zu bilden [14, 16, 17], siehe auch Abschn. 2.1.

Weitere Probleme, welche ebenfalls durch das Framework adressiert werden, sind die Ausfallsicherheit und die Übertragungsgarantie. Einzelne Prozesse müssen überwacht und bei Ausfall automatisch neu gestartet werden. Sollten einzelne Softwaremodule Zustands-Maschinen darstellen, also zum Beispiel nach der x-ten Überschreitung eines Schwellwertes einen Alarm auslösen, muss auch der Zustand, den ein Modul vor dem Neustart besaß, wiederhergestellt werden. Messdaten müssen das System garantiert genau einmal komplett durchlaufen. Beide Probleme machen das periodische Speichern des Datenflusses und der Zustände der einzelnen Softwaremodule auf einer Festplatte notwendig [16, 33].

Die Stakeholder verwenden auf ihren Computersystemen sowohl Windows als auch Linux als Betriebssystem. ChESS soll daher nach Möglichkeit mit beiden Systemen interagieren können. Auch Einplatinencomputer mit der Mikroprozessor-Architektur ARM, unter Verwendung des Betriebssystem Linux, kommen zum Einsatz. Die bisher implementierten Prototypen des ChESS-Systems setzen auf Linux als Betriebs-

system. Es ist sehr wahrscheinlich, dass sich nicht alle Funktionalitäten unter Windows implementieren lassen. Die Systeme können aber im jeweils anderen als virtuelle Maschinen aufgesetzt werden oder es wird ein Computercluster, bestehend aus unterschiedlichen Maschinen, genutzt. Die Transportschicht des einzusetzenden Frameworks sollte auf beiden Betriebssystemen lauffähig sein. Ein hohes Abstraktionslevel ist gewünscht, um dem Anwender Programmieraufwand zu ersparen.

Die Mehrheit der Stakeholder bewertet die Reaktionszeit, die das ChESS-System einhalten muss, als unkritisch. Daher wird für eine erste Implementierung davon ausgegangen, dass die Reaktionszeit direkt von der Frequenz abhängt, mit der Daten in das System fließen. In der Meeresforschung werden Sensorwerte oft über mehrere Minuten gemittelt. Kritischer ist die Verarbeitung von Audio- und Bilddaten. Im Audiobereich wurde eine Abtastrate von 16 kHz angegeben. Ein Stakeholder aus dem Bereich der Fernerkundung nutzt zur Detektion von Umweltverschmutzungen, wie etwa Plastikmüll oder Ölteppichen, ein eigens entwickeltes KI-Verfahren zur Bildinterpretation. Aufgrund der frühen Phase des Projektes konnten noch keine Angaben über die Bildauflösung gemacht werden. Die Bilder würden aber etwa alle ein bis zwei Sekunden in das System fließen. Bei VGA Auflösung ergeben sich daraus bereits 307.200 Pixel pro Sekunde. Der Stakeholder ist sich der entstehenden Datenmenge bewusst und möchte sein System über die Bildauflösung skalieren. Diesbezüglich ist die Möglichkeit zur horizontalen Skalierung ein weiterer wichtiger Punkt, welchem jedoch für eine erste Implementierung des ChESS-Systems keine hohe Priorität eingeräumt wird.

5 Fazit & Ausblick

Wie vorgesehen konnten durch den Prozess der Anforderungsanalyse weitere Anwendungsbeispiele für das ChESS-System gefunden werden. Funktionale Anforderungen umfassen dabei etwaige Systeme, welche am Ausgang des ChESS-Systems ausgelöst werden sollen. Diesbezüglich wurde im initialen Entwurf des Systems nicht davon ausgegangen, dass neben den Methoden das maschinellen Lernens auch einfache Rechen- und Vergleichsoperationen benötigt werden, um die angeschlossenen Systeme sinnvoll anzusteuern. Hier, aber auch in weiteren Verarbeitungsschritten, liegt die Hauptverantwortung für die Implementierung bei den Stakeholdern. Das ChESS-System soll nur Module beinhalten, welche eine allgemeine Gültigkeit aufweisen. Auch die Systemkonfiguration spielt eine wichtige Rolle. Maschinelle Lernmodelle und Softwaremodule müssen während des Betriebes austauschbar sein. Außerdem benötigt nicht jeder Stakeholder den vollen Funktionsumfang. Daher ist es umso wichtiger den Stakeholdern ein benutzerfreundliches „Data Processing“ Framework zur Verfügung zu stellen.

Bezogen auf die Softwarearchitektur entspricht die Forderung der nachträglichen Betrachtung von Messdaten nicht dem initialen Entwurf des ChESS-Systems, da dieses lediglich die Echtzeitauswertung adressiert. Sofern die Stakeholder selbst Datenbanken

zur Speicherung vorhalten und sich diese als Eingänge für das ChESS-System nutzen lassen, ergeben sich in Hinblick auf die in Abschn. 2.4 vorgestellten Softwarearchitekturen keine Unterschiede. Die Kappa-Architektur ist auch in der Lage Dateninstanzen zu prozessieren [16].

Die Stakeholder bestätigten die Notwendigkeit der Funktionalitäten des initialen Systementwurfs. Die auf Künstlicher Intelligenz basierten Module und die adaptive Fähigkeit des Systems fanden aufgrund der allgemeinen Gültigkeit Zuspruch.

An die Reaktionszeit des geplanten Systems gibt es noch keine konkrete Anforderung. Allerdings bedingt die Echtzeitfähigkeit des geplanten Systems den Einsatz maschineller Lernmethoden, die komplexe physikalische Modelle in ihrer Performance übertreffen.

Abb. 5 zeigt den Entwurf der Systemarchitektur nach der Anforderungsanalyse. Hinzugekommen ist eine Schnittstelle zur Visualisierung der Daten. Die darzustellenden Daten können aus einem beliebigen Funktionsblock, beziehungsweise Verarbeitungsschritt, stammen. Es kann auch angezeigt werden, welche Daten momentan in das System fließen. Ein weiterer Funktionsblock stellt die in Abschn. 4.1 genannte Prüfung vor dem Auslösen weiterer Systeme dar.

Zurzeit wird die Anforderungsanalyse mit den in Abschn. 3 erwähnten Methoden komplettiert. Hohe Priorität hat das zu implementierende „Data Processing“ Framework. Die bisherigen Ergebnisse der Anforderungsanalyse haben viele wichtige Punkte, wie die Anforderungen an Ausfall- und Übertragungssicherheit, aufgezeigt. Die definierten Anforderungen sollten die Findung eines zum System passenden Frameworks ermöglichen. Neben den technischen Funktionalitäten, steht bei der Suche vor allem die Benutzerfreundlichkeit und die Popularität im Fokus. Mögliche Kandidaten für das Framework sind unter anderem: „Flink“ [33], „Storm“ (https://storm.apache.org), „StreamBox“ [34], das weniger bekannte „Odysseus“ (https://odysseus.informatik.uni-oldenburg.de) der Universität Oldenburg und das „Robot Operating System 2“ (ROS2)

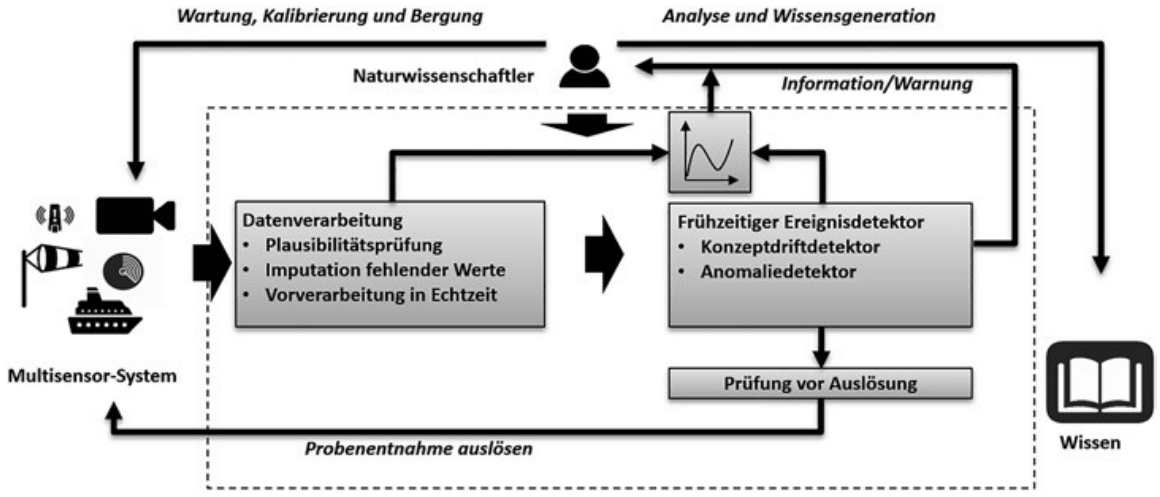

Abb. 5 Erweiterter Entwurf der Systemarchitektur von ChESS

(https://docs.ros.org) – streng genommen kein „Data Processing“ Framework, aber auch auf Echtzeitdatenverarbeitung ausgelegt. Im Anschluss an diesen Prozess ist ein Framework zu implementieren und zu testen. Für die Tests sollen Netzwerksimulatoren eingesetzt werden. Parallel dazu werden die bisherigen Prototypen des ChESS-Systems weiterentwickelt und evaluiert. Dies soll den Einsatz von Unit-Tests ermöglichen.

Danksagung Diese Arbeit wurde über das Projekt ChESS vom Ministerium für Wissenschaft und Kultur Niedersachsen aus dem Niedersächsischen Vorab der Volkswagen-Stiftung gefördert (ZN3683). Darüber hinaus danken wir den Stakeholdern für die Bereitschaft an den Interviews teilzunehmen.

Literatur

1. Baschek, B., Schroeder, F., Brix, H., Riethmüller, R., Badewien, T. H., Breitbach, G., et al. (2017). The Coastal Observing System for Northern and Arctic Seas (COSYNA). *Ocean Science, 13*, 379–410.
2. Reuter, R., Badewien, T. H., Bartholomä, A., Braun, A., Lübben, A., & Rullkötter, J. (April 2009). A hydrographic time series station in the Wadden Sea (southern North Sea). *Ocean Dynamics, 59*, 195–211.
3. Hillebrand, H., & Matthiessen, B. (2009). Biodiversity in a complex world: Consolidation and progress in functional biodiversity research. *Ecology Letters, 12*, 1405–1419.
4. Rockström, J., Steffen, W., Noone, K., Persson, Å., Chapin, F. S., Lambin, E. F., et al. (2009). A safe operating space for humanity. *Nature, 461*, 472–475.
5. Ryabinin V, Barbière J, Haugan P, Kullenberg G, Smith N, McLean C, Troisi A, Fischer A, Aricò S, Aarup T, Pissierssens P, Visbeck M, Enevoldsen HO and Rigaud J. (2019). *The UN Decade of Ocean Science for Sustainable Development*. Front. Mar. Sci. 6:470. https://doi.org/10.3389/fmars.2019.00470.
6. JPI OCEANS. (2015). Strategic research and innovation agenda 2015–2020. *Strategic Research and Innovation Agenda 2015–2020*. JPI Oceans. https://www.jpi-oceans.eu/sites/jpi-oceans.eu/files/managed/Publications%20files/strategic-research-and-innovation-agenda-2015-2020.pdf. Zugegriffen: 17. Mai 2022.
7. Malde, K., Handegard, N. O., Eikvil, L., & Salberg, A.-B. (2020). Machine intelligence and the data-driven future of marine science. *ICES Journal of Marine Science*, 77, 1274–1285.
8. Monterey Bay Aquarium Research Institute. (2014). *Technologie Roadmap 2014*. Monterey Bay Aquarium Research Institute (MBARI), https://www.mbari.org/wp-content/uploads/2015/04/TechnologyRoadmap.pdf.
9. OECD. (2016). *The ocean economy in 2030*. Organisation for Economic Co-operation and Development. https://www.oecd-ilibrary.org/economics/the-ocean-economy-in-2030_9789264251724-en. Zugegriffen: 17. Mai 2022.
10. Zhang, Y., Godin, M., Bellingham, J., & Ryan, J. (2012). Using an autonomous underwater vehicle to track a coastal upwelling front. *Oceanic Engineering, IEEE Journal of, 37*, 338–347. https://doi.org/10.1109/JOE.2012.2197272.
11. Zielinski, O., Pieck, D., Schulz, J., Thölen, C., Wollschläger, J., Albinus, M. et al. (2022). The spiekeroog coastal observatory: A scientific infrastructure at the Land-sea transition zone (Southern North Sea). *Frontiers in Marine Science, 8,* 754905:1–754905:28.

12. Montiel, J., Halford, M., Mastelini, S. M., Bolmier, G., Sourty, R., Vaysse, R., et al. (2021). River: Machine learning for streaming data in Python. *Journal of Machine Learning Research, 22*, 1–8.
13. Oehmcke, S., Zielinski, O., & Kramer, O. (2016). kNN ensembles with penalized DTW for multivariate time series imputation (S. 2774–2781). *2016 International Joint Conference on Neural Networks (IJCNN)*.
14. Srivastava, U., & Widom, J. (2004). Flexible time management in data stream systems. *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (S. 263–274). Association for Computing Machinery.
15. Akidau, T., Bradshaw, R., Chambers, C., Chernyak, S., Fernández-Moctezuma, R. J., Lax, R., et al. (2015). The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of the VLDB Endowment, 8*, 1792–1803.
16. Akidau, T., Chernyak, S., & Lax, R. (2018). *Streaming systems: The what, where, when, and how of Large-scale data processing*. O'Reilly.
17. Akidau, T., Begoli, E., Chernyak, S., Hueske, F., Knight, K., Knowles, K. et al. (2021). Watermarks in stream processing systems: Semantics and comparative analysis of apache flink and google cloud dataflow. *Proceedings of the VLDB Endowment, 14*, 3135–3147.
18. Awad, A., Traub, J., & Sakr, S. (2019). Adaptive Watermarks: A Concept Drift-based Approach for Predicting Event-Time Progress in Data Streams. In: Advances in Database Technology – 22. Internationale Konferenz über „Extending Database Technology (EDBT 2019)", Lissabon, Portugal, 26–29. März 2019, OpenProceedings.org, S. 622–625. https://doi.org/10.5441/002/edbt.2019.71.
19. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys, 46*, 44:1–44:37.
20. Lukats, D., Berghöfer, E., Stahl, F., Schneider, J., Pieck, D., Idrees, M. M., et al. (2022). *Towards concept change detection in marine ecosystems. IEEE Journal of Oceanic Engineering*. IEEE.
21. Oehmcke, S., Zielinski, O., & Kramer, O. (2015). Event detection in marine time series data. In S. Hölldobler, R. Peñaloza, & S. Rudolph (Hrsg.), *KI 2015: Advances in Artificial Intelligence* (S. 279–286). Springer International Publishing.
22. Guha, S., Mishra, N., Roy, G., & Schrijvers, O. (2016). Robust random cut forest based anomaly detection on streams. *Proceedings of The 33rd International Conference on Machine Learning* (S. 2712–2721). PMLR.
23. Tan, S. C., Ting, K. M., & Liu, T. F. (2011). Fast anomaly detection for streaming data. *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*
24. Yilmaz, S. F., & Kozat, S. S. (2020). *PySAD: A streaming anomaly detection framework in python*. Tech. rep., arXiv.
25. Gözüaçık, Ö., Büyükçakır, A., Bonab, H., & Can, F. (2019). Unsupervised Concept Drift Detection with a Discriminative Classifier. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (S. 2365–2368). Association for Computing Machinery.
26. Korycki, L., & Krawczyk, B. (2019). Unsupervised drift detector ensembles for data stream mining. *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, (S. 317–325).
27. Souza, V. M., Chowdhury, F. A., & Mueen, A. (2020). Unsupervised drift detection on high-speed data streams (S. 102–111). *2020 IEEE International Conference on Big Data (Big Data)*.

28. Feick, M., Kleer, N., & Kohn, M. (2018). *Fundamentals of Real-time data processing architectures lambda and kappa.* Gesellschaft für Informatik e.V.
29. Kleuker, S. (2011). *Grundkurs Software-Engineering mit UML: Der pragmatische Weg zu erfolgreichen Softwareprojekten*. Vieweg+Teubner.
30. Balzert, H. (2009). *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. Spektrum Akademischer Verlag.
31. Kreps, J. (2011). Kafka : A distributed messaging system for log processing. https://www.semanticscholar.org/paper/Kafka-%3A-a-Distributed-Messaging-System-for-Log-Kreps/ea97f112c165e4da1062c30812a41afca4dab628.
32. Nussbaumer, H. J. (1981). The fast fourier transform. In H. J. Nussbaumer (Hrsg.), *Fast fourier transform and convolution algorithms* (S. 80–111). Springer.
33. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., & Tzoumas, K. (2015). Apache Flink™: Stream and Batch Processing in a Single Engine. *IEEE Data Engineering Bulletin, 38*.
34. Miao, H., Park, H., Jeon, M., Pekhimenko, G., McKinley, K. S., & Lin, F. X. (2017). StreamBox – Modern stream processing on a multicore machine. *Proceedings of the USENIX Annual Technical Conference (USENIX ATC 17)*, (S. 617–629) USENIX Association. https://www.usenix.org/conference/atc17/technical-sessions/presentation/miao.