Smart Data Interoperability

edited by

Andreas Winter
Carl von Ossietzky University, Oldenburg

Christian Schönberg
Carl von Ossietzky University, Oldenburg

Oldenburg Lecture Notes on Software Engineering

OLNSE Number 7/2024
March 2024

Oldenburg Lecture Notes
on Software Engineering (OLNSE)
Carl von Ossietzky University Oldenburg
Department for Computer Science
Software Engineering Group
26111 Oldenburg, Germany

Content

Malte Südema
Machbarkeitsprüfung und Implementierung von Daten-Interoperabilität mit Fiware 1

Smart Data Interoperability

Im Interreg North Sea Projekt „Data for All“ (D4A)[1] werden innovative Daten-getriebene Ansätze zur Schaffung und Verbesserung von digitalen Diensten im Bereich von Smart Cities und Smart Regions erforscht und praktisch erprobt. Smarte Systeme – Smart Cities, Smart Health Applications, Smart Agriculture Systems, usw. – produzieren große Mengen an Daten. Diese Daten sind oft sehr unterschiedlich aufgebaut, selbst wenn sie ähnliche Informationen enthalten. Sie folgen oft keinem standardisierten Schema, oder interpretieren Standards auf unterschiedliche Weise. Sie verwenden unterschiedliche Formate, Formatierungen, Kodierungen, Maßeinheiten, Intervalle, Konventionen und Annahmen.

Beispielsweise kann ein Anbieter von E-Scootern seine Daten über die Gefährte nach deren Aufenthaltsort strukturieren, während ein anderer Anbieter die Daten nach dem genauen Typs des Scooters strukturiert. Dadurch ist eine gemeinsame Nutzung beider Datensätze deutlich erschwert. Aber nicht nur strukturell unterscheiden sich Daten Smarter Systeme, auch im Detail gibt es große Unterschiede: So speichert ein Scooter-Anbieter alle zwei Minuten den Aufenthaltsort der Geräte, ein anderer nur alle drei Minuten. Will man die Zeitreihen zusammenlegen, kommt es ggf. zu Inkonsistenzen. Auch unterschiedliche Maßeinheiten (mm/cm/in, usw.) oder andere technische Formate (JSON/XML/SQL/NoSQL) erschweren die gemeinsame interoperable Verwendung der Daten.

Im internationalen Seminar „Smart Data Interoperability“ wurden im Wintersemester 2023/2024 gemeinsam mit der Tashkent University of Information Technologies named after Muhammad Al-Khwarizmi – Zweigstelle Urgench im Rahmen von forschungsorientierter Lehre und im Kontext des D4A-Projekts unterschiedliche existierende Anwendungen und Frameworks zur Datenintegration bezüglich ihrer Interoperabilitätsansätze und -eigenschaften untersucht.

Untersucht wurden in eigenständiger Forschung die Frameworks KNIME, GraphQL, FIWARE, SAS Viya, Alteryx, IBM DataStage, Oracle Data Integrator, AWS GLUE, Microsoft SQL Server, Informatica Power Center und Microsoft SQL Integration Service. Genauer betrachtet wurden KNIME, GraphQL, FIWARE und Microsoft SQL Integration Service. Die gewonnenen Erkenntnisse zu FIWARE werden im Folgenden dokumentiert.

In the Interreg North Sea project “Data for All” (D4A)[1], innovative data-driven approaches to create and improve digital services in the field of Smart Cities and Smart Regions are researched and practically tested. Smart systems – smart cities, smart health applications, smart agriculture systems, etc. – produce large amounts of data. These data are often structured very differently, even if they contain similar information. They often do not follow a standardized schema, or they interpret standards in different ways. They use different formats, formatting, coding, units of measurement, intervals, conventions, and assumptions.

For example, one e-scooter provider may structure its vehicle data according to their location, while another provider may structure the data according to the exact type of scooter. This makes it much more difficult to use both sets of data together. But it is not only structurally that data of smart systems differ, there are also major differences in detail: for example, one scooter provider stores the whereabouts of the vehicles every two minutes, another only every three minutes. If one wants to combine the time series, inconsistencies may occur. Different units of measurement (mm/cm/in, etc.) or different technical formats (JSON/XML/SQL/NoSQL) also make it difficult to use the data in an interoperable manner.

In the international seminar “Smart Data Interoperability” during the winter semester 2023/2024 together with the Tashkent University of Information Technologies named after Muhammad Al-Khwarizmi – Urganch Branch as part of research-oriented teaching and in the context of the D4A project, different applications and frameworks for data integration were examined, focussing on their approaches and properties of interoperability.

In independent research, the frameworks KNIME, GraphQL, FIWARE, SAS Viya, Alteryx, IBM DataStage, Oracle Data Integrator, AWS GLUE, Microsoft SQL Server, Informatica Power Center, and Microsoft SQL Integration Service were examined. Particular focus

[1] https://www.interregnorthsea.eu/dataforall

was put on KNIME, GraphQL, FIWARE and Microsoft SQL Integration Service. The insights gained there about FIWARE are documented below.

Machbarkeitsprüfung und Implementierung von Daten-Interoperabilität mit Fiware

Malte Südema
Softwaretechnik - Department für Informatik
Carl von Ossietzky Universität Oldenburg
Oldenburg, Deutschland
malte.suedema@uol.de

***Zusammenfassung*—Die Verwendung von Daten spielt im Rahmen vom Data-for-All (D4A)-Projekt eine elementare Rolle. Dabei werden Daten aus verschiedenen Quellen einer Smart Region miteinander verschnitten, welche zudem unterschiedliche Eigenschaften und Repräsentationen haben. Um die Interoperabilität diese heterogenen Daten zu ermöglichen, müssen Methoden implementiert werden, welche die Daten auch für Dritte nutzbar macht. Für diesen Zweck wird im Zuge dieser Arbeit ein beispielhaftes Szenario vorgestellt, welches die Interoperabilität von Daten aus unterschiedlichen Quellen und Domänen voraussetzt. Auf Grund dieses Szenarios folgt die Implementierung eines Softwaresystems zur Prüfung der Machbarkeit mit Hilfe des Smart Data Frameworks Fiware. Abschließend wird beurteilt inwiefern eine Umsetzung mit Hilfe von Fiware sinnvoll ist, um das Interoperabilitätsproblem der Daten im Rahmen des D4A-Projektes zu lösen.**

***Schlüsselwörter*—Interoperabilität, Smart Data, Fiware, Verteilte Systeme, Smart Region, Data for All,**

I. MOTIVATION

In der heutigen Zeit vernetzen sich Städte nicht nur innerhalb ihrer urbanen Grenzen, sondern auch immer weiter mit ihrem Umland und bilden so *Smarte Regionen* [1]. Der digitale Wandel führt dazu, dass eine große Menge an Daten entsteht, welche von den *Smart Regions* verarbeitet werden müssen. Im Jahre 2020 gab es im urbanen Raum bereits über 50 Milliarden Internet of Things (IoT)-Geräte [2], die neben öffentlichen Informationsquellen, wie Ämter und Verkehrsbetriebe, und Unternehmen immer mehr Daten zur Verfügung stellen. Diese Daten sollen dabei nicht nur die Ressourcen besser nutzbar machen und Emissionen verringern, sondern auch Mehrwerte bei der Nutzung von Verkehrsmitteln, der öffentlichen Verwaltung und der Sicherheit schaffen [3].

Allerdings stellt die Verwendung dieser Daten eine Herausforderung dar, da sie im hohen Grade heterogen sind. Durch die unterschiedlichen Quellen und Domänen aus dem die Daten stammen, liegen diese zumeist auch in verschiedenen Formaten und variierender Qualität vor. Um die heterogenen Daten miteinander zu verschneiden und nutzbar zu machen, also Interoperabilität zu ermöglichen, ist eine einheitliche Form notwendig.

Im Rahmen des durch die Europäische Union angestoßene Projekt Data-for-All (D4A) [4] soll daher eine Software entwickelt werden, welche heterogene Daten gebündelt für unterschiedliche spezielle Zielschemata interoperabel nutzbar macht. Das D4A-Projekt ist eine Gemeinschaftsarbeit aus 19 unterschiedlichen Städten und sieben Nationen der Nordseeregion, um diese in eine Vorreiterposition für Daten-getriebene Innovationen zu positionieren.

II. PROBLEMBESCHREIBUNG

Eine elementare Herausforderung stellt die Interoparabilität der Daten im Kontext des D4A-Projektes dar. Insbesondere die Heterogenität und die unterschiedlichen Quellen sind für die weitere Verarbeitung durch eine Software und eine anschließende Nutzung durch Dritte problematisch, da nicht nur technische- sondern auch Daten- und Schemakonflikte auftreten können [5]. Auf technischer Ebene können die Daten in verschiedenen Formaten und über variierende Übertragungswege bereitgestellt werden. Wird die Datenebene betrachtet, können die Daten in unterschiedlicher zeitlicher Auflösung, Maßeinheit oder Kodierung vorliegen. Letztlich kann auch das Schema in dem die Daten eingespeist werden uneinheitlich sein, da die Datenschemata zumeist auf die Bedürfnisse der Unternehmen und Institutionen angepasst sind, welche die Daten zur Verfügung stellen.

All diese Problematiken müssen durch ein Softwaresystem im D4A-Kontext berücksichtigt werden, sodass eine reibungslose zielgerichtete Nutzung heterogener Daten möglich ist. Somit ist die Erstellung eines Szenarios notwendig, welches möglichst viele Problematiken abdeckt, um diese mit Hilfe eines Softwaresystems zu lösen. Da Fiware [6] bereits durch einen Partner des D4A-Projektes genutzt wird, soll dieses Framework als Basis für die Implementierung evaluiert werden. Daraus ergibt sich folgende Forschungsfrage, welche im Rahmen dieser Arbeit beantwortet werden soll.

Wie sieht ein Szenario für die Interoperabilität von Daten aus und kann die in dem Szenario enthaltene Problemstellung durch das Framework Fiware gelöst werden?

III. GRUNDLAGEN

Im folgendem Abschnitt werden die benötigten Grundlagen für das weitere Verständnis dargelegt. Es folgt eine kurze Beschreibung des Terms Interoperabilität und es wird das für diese Arbeit zugrunde liegende Framework Fiware vorgestellt.

A. *Interoperabilität*

Im Kontext der Verarbeitung von heterogenen Daten bezieht sich Interoperabilität auf den reibungslosen Verkehr von Daten zwischen unterschiedlichen Systemen und Datenquellen und der Kombination von Daten. Dies kann durch verschiedene Formate, Strukturen und Standards, welche durch die Quelle der Daten festgelegt wurde, erschwert werden [7].

Wird die Interoperabilität im Rahmen dieser Arbeit betrachtet, müssen verschiedene Aspekte für die Interaktion der verschiedenen Quellen berücksichtigt werden. Insbesondere die technische, syntaktische, semantische und organisatorische Interoperabilität spielen eine übergeordnete Rolle. Diese Aspekte werden oft auch als Ebenen der Interoperabilität bezeichnet [8].

Die **technische Interoperabilität** bezieht auf Software- und Hardwarekomponenten, welche die Maschine-zu-Maschine-Kommunikation ermöglichen [9]. Hierbei spielen insbesondere die Übertragungsprotokolle und Übertragungsinfrastruktur eine Rolle. Für diese Arbeit ist dabei die Übertragung von Daten mit Hilfe von *Representational State Transfer* (REST)-Schnittstellen über das *Hypertext Transfer Protocol* (HTTP) von Relevanz.

Ebenso müssen die Formate und Kodierungen der übertragenden Daten betrachtet werden. Diese sind Teil der **syntaktischen Interoperabilität**. Hierbei ist die Verarbeitung von gängigen Formaten, wie *Extensible Markup Language* (XML), *JavaScript Object Notation* (JSON), *Comma-Separated Values* (CSV) oder YAML, wesentlicher Bestandteil zur einheitliche Verständigung zwischen der Quelle und dem Zielsystem [8]. Oft wird die technische mit der syntaktischen Interoperabilität zusammengefasst, da diese ähnliche Problematiken beschreiben. Zudem existieren für beide Aspekte vollständig entwickelte Lösungen, um eine Interoperabilität zu ermöglichen [10].

Im Vergleich zu den vorherigen Aspekten ist die Lösung der **semantischen Interoperabilität** signifikant komplexer, da es hierfür keine einheitliche Lösung geben kann [9]. Durch unterschiedliche Interpretationen der übertragenen Daten kann es zu Problemen bei der weiteren Verwendung geben. Insbesondere wenn unterschiedliche Quellen ähnliche Daten liefern, können verschiedene Bezeichnungen für die gleichen Daten aufkommen. Es kann also vorkommen, dass durch verschiedene Schemata Fehler bei der Interpretation auftreten können. Aber auch Datenkonflikte können auftreten. Die Auflösung der Daten in Bezug auf Zeit, unterschiedliche Maßeinheiten oder durch unterschiedliche Repräsentation der Daten kann zu Problemen bei der Interpretation führen [5]. Nach der Interpretation können Metadatenschema, wie Smart-Data-Models [11], im Zielsystem für eine einheitliche Semantik sorgen und so die fortführende Verarbeitung vereinfachen [9].

Letztlich wird bei der **organisatorischen Interoperabilität** der Fokus auf die Zusammenarbeit zwischen Organisationen gesetzt, welche die Daten miteinander austauschen. Dabei können gemeinsam Standards im Bezug auf Prozesse, Praktiken oder Richtlinien entwickelt werden [7]. Ebenso fallen rechtliche Rahmenbedingungen für die Verwendung und den Austausch von Daten in diese Ebene. Allgemein werden bei der organisatorischen Interoperabilität nicht technische Aspekte betrachtet. Diese Ebene wird jedoch nicht tiefergehend während der Bearbeitung betrachtet.

B. *Smart-Data-Model*

Durch allgemeine Datenschemata können Daten in eine einheitliche Form im Bezug auf ihre Syntax und Semantik gebracht werden. Sie dienen dadurch als Metaschema auf dessen Basis neue Daten gespeichert und anderen Anwendern zur Verfügung gestellt werden können. Für smarte Daten wurde daher ein Smart-Data-Model [11] durch die Zusammenarbeit der Fiware Foundation [6], TM Forum [12], IUDX [13] und dem OASC [14] erarbeitet. Das Modell setzt sich auch zwei Komponenten zusammen. Als Schema wird die technische Repräsentation des Modells bezeichnet, welche neben den Datentypen auch die Struktur enthält. Des weiteren beschreibt eine Spezifikation die Semantik der zugrunde liegenden Datentypen. Ebenso sind die Modelle kompatibel mit der *Next Generation Service Interface* (NGSI)-v2 und NGSI-LinkedData Spezifikation [15], wodurch die Einbettung in smarte Informationssysteme vereinfacht werden kann. Die Modelle sind über ein Git-Repository erreichbar und in unterschiedlichen Domänen, wie Smart Cities oder Smart Agrifood, geordnet.

C. *Fiware*

Fiware ist eine offene, standardbasierte Plattform, die sich auf die Entwicklung von Anwendungen zur Verarbeitung und Verwaltung smarter Daten konzentriert. Die Plattform wurde entwickelt, um eine interoperable und skalierbare Umgebung für die Erstellung von Anwendungen zu bieten, die auf Echtzeitdaten und Kontextinformationen basieren. Entstanden ist die Future Internet Ware (Fiware) im Jahr 2011 als Ergebnis einer europäischen Forschungs- und Innovationsinitiative und wird seitdem durch die Europäische Kommission unterstützt.

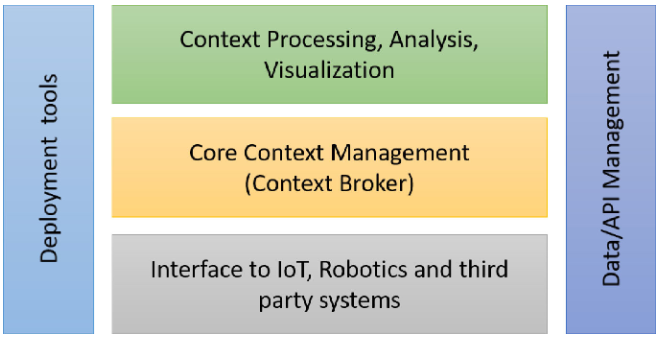

Abbildung 1. Kernkomponenten von Fiware. [16]

Fiware besteht in ihrem Kern aus einer Vielzahl an Komponenten, welche miteinander interagieren. In Abbildung 1 sind die Kernkomponenten des Systems abgebildet. Das Fundament bildet die **Datenschnittstelle**. Diese ist für die Einbindung von IoT-Geräten und -Plattformen sowie *Third Party*-Systemen als

Datenquelle verantwortlich. Die übertragenden Daten werden in der Regel zuerst durch ein Agent Framework verarbeitet. Da das Core Context Management mit der NGSI-*Application Programming Interfaces* (API) arbeitet, werden die übertragenden Daten zunächst in das dafür benötigte Format im Agent Framework gebracht, welches Teil des Datenschnittstelle ist. Dieser Schritt kann übersprungen werden, wenn die Daten bereits in dem richtigen Format vorliegen, wie es bei der Einbettung von vertikalen smarten Lösungen vorkommen kann. Hierbei ist anzumerken, dass die Datenintegration durch Drittsoftware oder Erweiterungen realisiert wird.

Der Dreh- und Angelpunkt der Fiware-Plattform ist das **Core Context Management** oder auch **Orion/Fiware Context Broker** genannt. Über diesen werden Daten verwaltet und mit anderen Komponenten kommuniziert. Das Management der Daten wird mit Hilfe der NGSI-Spezifikation durchgeführt und die Daten sind üblicherweise in einer angebundenen MongoDB gespeichert. Sie liegen im durch das gewählte Smart-Data-Model Format vor. In dieser Datenbank werden jedoch nur aktuelle Daten gespeichert, sodass immer nur genau eine Repräsentation einer Entität in der Datenbank enthalten ist. Dadurch soll stets der momentane *Real-Time*-Zustand der Daten abgebildet werden. Durch ein Publisher-Subscriber Modell werden die Daten in der Datenbank aktualisiert sobald neue Informationen über die Datenschnittstelle übertragen werden. Die Informationen, die vorher für die Entität gespeichert waren, gehen damit verloren. Die Daten können über API-Aufrufe über das Core Context Management abgerufen werden. Hierfür stellt Fiware einen umfangreichen Anfragenkatalog zur Verfügung mit denen sich auch komplexere Anfragen formulieren lassen. Durch Anfragen lassen sich auch Daten miteinander verschneiden, sodass dem Daten ein Attribut hinzugefügt wird, welches wiederum auf andere Daten verweist. So ist es möglich verschiedene Daten mit Hilfe einer Anfrage anzufordern. Für die Speicherung von historischen Daten bietet Fiware Erweiterungen an, welche es ermöglicht Daten über einen Zeitraum zu sammeln, zu speichern und verfügbar zu machen. Eine expliziter Speicherort beziehungsweise eine Erweiterung für die historischen Daten ist notwendig, da das Core Context Management eine strikte Trennung zwischen aktuellen und historischen Daten vornimmt.

Weiterer wesentlicher Bestandteil ist die **Context Verarbeitung, Analyse und Visualisierung**. Diese wird im Kontext der Fiware-Plattform meist nicht durch eigene Softwarelösungen realisiert. Um diese Aufgaben zu erfüllen, können unterschiedliche andere Werkzeuge und Frameworks verwendet werden, die mit Fiware kompatibel sind oder für die eine *Bridge* angeboten wird. So dient Grafana beispielsweise zur Visualisierung oder Spark, Tensorflow, Hadoop oder auch eigene Softwareentwicklungen werden für die Verarbeitung der Daten verwendet. Diese Frameworks sind nicht unmittelbar Teil der Fiware-Plattform, da sie von Drittherstellern stammen. Allerdings werden sie im weiteren Sinne als Teil des Workflows und damit auch des Systems betrachtet, weil sie stark mit der Fiware-Plattform verflochten sind und die Kompatibilität mit Fiware nachgewiesen haben. Die Verarbeitung der Daten kann durch das Core Context Management angestoßen werden und die Context Verarbeitung liefert im Umkehrzug die verarbeiteten oder aktualisierten Daten zurück. Dabei verlassen die Daten in der Regel nicht die Systemgrenzen der Fiware-Plattform.

Querschnittsfunktionen, wie die **Deployment Tools** und das **Data/API Management**, bilden die letzten Kernkomponenten. Erstere bilden sich aus Werkzeugen, wie Docker, um das System auszuführen. Zweitere haben die Aufgabe Daten für externe zur Verfügung zu stellen und Sicherheitsfunktionen zu erfüllen. Hier können also verschiedene Komponenten integriert werden, welche die Daten über Publikationsplattformen veröffentlichen oder zur Monetarisierung dienen. Ebenso findet in dieser Komponente das Identitäts- und Passwortmanagement für das System statt.

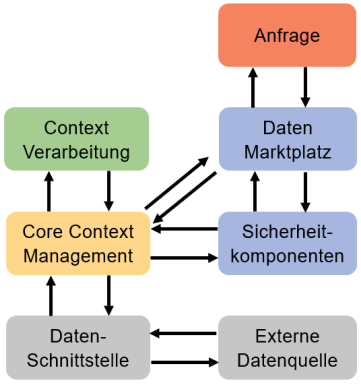

Abbildung 2. Beispielhafte Darstellung der Interaktion zwischen Fiware-Komponenten bei einer Datenanfrage. [Eigene Darstellung]

Der ***Workflow*** einer Anfrage stellt sich somit wie folgt dar und ist zur Veranschaulichung der beteiligten Komponenten und ihre Beziehung untereinander in Abbildung 2 dargestellt. Zunächst wird eine Anfrage über einen Daten-Marktplatz getätigt. Diese Anfrage wird innerhalb der Fiware-Plattform an das Core Context Management weitergeleitet. Dieses ruft die benötigten Daten über die Datenschnittstelle ab. Dann wird durch die gespeicherten Kontextinformationen geprüft, ob weitere Datenverarbeitungsschritte notwendig sind. Ist dies der Fall, werden die Daten an entsprechendes Context Verarbeitungs-Framework weitergeleitet und die neuen oder aktualisierten Daten werden nach Verarbeitung wieder an das Core Context Management zurückgeleitet. Diese Daten dienen dann zur Beantwortung der Anfrage.

IV. SZENARIO

In der **Smart KielRegion** [17] werden durch einen Fahrradverleihanbieter im Stadtgebiet und der näheren Umgebung Fahrräder zum Verleih angeboten. Die Standorte der Fahrräder sind stationär durch den Anbieter festgelegt und werden als Fahrradstation bezeichnet.

Nun möchte ein Kunde ein Fahrrad mieten, um damit möglichst flexibel seinen Arbeitsweg bestreiten zu können.

Das heißt, dass er nicht nur ein Fahrrad in der nähe seines Zuhauses mieten möchte, sondern auch zu einem anderen Zeitpunkt in der Nähe seines Arbeitsplatzes. Um möglichst schnell und flexibel seine Destination zu erreichen, ist es notwendig, dass immer mindestens ein Fahrrad an der nächst gelegenen Fahrradstation vorhanden ist.

Der Fahrradverleihanbieter auf der anderen Seite hat ein großes Interesse daran, dass für den Kunden immer ein Fahrrad an der Fahrradstation zur Verfügung steht. Dies jedoch unter der Voraussetzung, dass nicht alle anderen Fahrräder seine Flotte ohnehin schon belegt sind. Dadurch erhöht er die Auslastung seine Fahrräder, welche eine begrenzte Ressource für den Fahrradverleihanbieter sind, und kann so auch seine Rentabilität steigern. Weiterer positiver Effekt ist, dass der Kunde zufriedener ist und die Leistung eher wieder in Anspruch nimmt, wenn er stets ein Fahrrad vorfindet und nicht zur nächsten Fahrradstation laufen muss.

Um dies zu gewährleisten kann der Fahrradverleihanbieter Prognosen zur Auslastung von Fahrradstation beziehungsweise zur Berechnung des zukünftigen Fahrradbedarfes erstellen. Dadurch können Fahrräder, welche an bestimmten Fahrradstationen nicht benötigt werden, durch den Fahrradverleihanbieter zu Fahrradstationen transportiert werden, die voraussichtlich einen Bedarf haben.

V. KONZEPTION

Aus dem Szenario geht hervor, dass eine Art von Prognose für Fahrräder an Fahrradstationen erstellt werden muss. Hierfür werden Daten der Fahrradstationen benötigt, wie auch Daten, welche einen möglichen Einfluss auf die Auslastung von Fahrradstationen haben können. In dieser Arbeit werden daher Wetterdaten als Einflussgröße verwendet.

Da die Machbarkeit der Interoperabilität von Daten in der Beispielimplementierung im Vordergrund steht, wurde im Rahmen dieser Arbeit keine Algorithmus für die Erstellung der Prognose implementiert. Der Fokus liegt auf dem Verschneiden der heterogenen Wetterdaten aus unterschiedlichen Quellen. Um die Interoperabilität mit Hilfe der Fiware-Plattform zu ermöglichen, müssen insbesondere die Ebenen der technische, syntaktische und semantische Interoperabilität näher betrachtet werden, welche in Unterabschnitt III-A bereits vorgestellt wurden.

Für die Umsetzung sind drei Datenquellen notwendig. Die Modellerstellung zur Prognose benötigt historischen Wetterdaten und die historische Daten über die Fahrradstationen. Durch Fiware-Komponenten, wie Cosmos, Cygnus, QuantumLeap oder Draco, können diese Daten in sogenannten *Short Term Historic* (STH)-Komponenten gespeichert werden. Um das Modell anzuwenden und Handlungen einzuleiten, sind Wetterprognosedaten notwendig. Diese Daten werden mit Hilfe eines Agent Frameworks an die entsprechenden Smart-Data-Model angepasst und sind direkt an der Datenschnittstelle der Fiware-Plattform angebunden, wie aus der Abbildung 3 zu entnehmen ist. Die Datenschnittstelle wiederum bildet Brücke zum Core Context Management und ermöglicht die

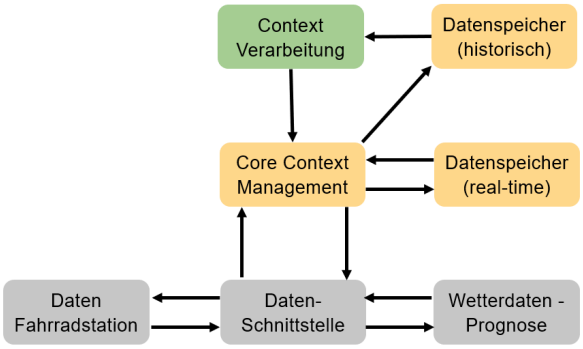

Abbildung 3. Konzeptionelle Interaktion der Komponenten. [Eigene Darstellung]

grundlegende Interoperabilität, da in dieser Komponente die oben genannten Ebenen der Interoperabilität adressiert werden.

Für das Management der Kontextdaten ist ein passendes Smart-Data-Model zu wählen und im Speicher des Core Context Managements abzulegen, wie in Abbildung 3 zu sehen. Hierfür sind die entsprechenden Kontextdaten pro Entität, also Wetterdaten oder Fahrradstation, zu definieren.

Eine Framework im Context Processing sorgt für die Erstellung eines Prognosemodells und dessen Anwendung bei eingehenden Anfragen. Wie bereits eingangs erwähnt, wird die Erstellung des Modells in dieser Arbeit nicht durchgeführt, da die Anwendung von *Machine-Learning*-Methoden für die Machbarkeit eine untergeordnete Rolle spielt. An Stelle dieser wird ein Dummy implementiert, welche lediglich aktualisierte Daten zurück gibt. Es werden jedoch alle Daten übertragen, welche für die Prüfung der Machbarkeit benötigt werden.

Zur Vereinfachung wurde auf die Implementierung eines Datenmarktplatzes als Intermediär zwischen externen Nutzern und dem Core Context Management verzichtet. Die Anfragen wurden somit direkt an das Core Context Management übermittelt.

VI. IMPLEMENTIERUNG

Folgend werden zunächst der Aufbau dargelegt, welcher für die Implementierung benötigt wurden. Darauf aufbauend werden die Funktionen der Komponenten näher erläutert und abschließend wird der Ablauf des Aktualisierungsverfahrens aufgezeichnet.

A. Aufbau

Für die Implementierung werden folgende Komponenten benötigt:

- Orion Context Broker als Kernkomponente.
- MongoDB für die Speicherung der Real-Time-Daten.
- STH-Komponente zur Speicherung der historischen Daten.
- REST-Schnittstelle, um die Aktualisierung der Daten durchzuführen.
- REST-Schnittstelle, um die Wetterprognosen zur Verfügung zu stellen.

- Optional: Postman, um API-Anfragen an den Orion Context Broker zu senden.

Um den Orion Context Broker, die MongoDB und die STH-Komponente zu verwenden, müssen diese mit Hilfe des Containerisierungsframeworks Docker erstellt werden. Jede der Komponenten wird in einem eigenen Container ausgeführt und diese Kommunizieren über ein Docker Netzwerk. Zur Vereinfachung wird der Bau Prozess mit Hilfe einer Docker-Compose Datei durchgeführt. Da durch die anderen Komponenten bereits eine Docker Infrastruktur aufgebaut worden ist, wurde auch der REST-Service mit Hilfe von Docker erstellt. Für die beiden REST-Schnittstellen wurde ein Flask Server erstellt, welcher als REST-Server für die Verarbeitung der Wetterprognosedaten und der Aktualisierung der Kontextdaten dient.

B. Durchführung

Für die initiale Population des Orion Context Brokers, wurden zunächst Fahrradstationsdaten, welche durch die Smarte KielRegion bereitgestellt worden sind, gefiltert. Dies war notwendig, da die Daten mehr als ein Vorkommen einer Stationsentität enthalten haben. Es wurden also alle Duplikate gelöscht, sodass eine Liste von einzelnen Stationen vorlag. Die enthaltenen Information wurden dann in die richtige Form für das gewählte Smart-Data-Model gebracht. Hierfür mussten das Format des Zeitstempels und der Koordinaten angepasst werden, wie auch die Benennung und Datentypen bestimmter Attribute, um mit dem Smart-Data-Format *BikeHireDockingStation* kompatibel zu sein. Zudem wurde dem Model das Attribute *prognosedBikeNumber* hinzugefügt. Die in der Liste enthaltenen Stationen wurden dann als JSON-String gespeichert und einzeln per POST-Anfrage an den Orion Context Broker gesendet. Dadurch wurden die Daten in der MongoDB abgelegt und können durch eine einzigartige *entityId* adressiert werden.

Folgend wurde die STH-Komponente konfiguriert, um Daten persistent zu speichern. Dafür muss für jede BikeStation-Entität, also für jede entityId, eine Subscription definiert werden und über eine POST-Anfrage im Orion Context Broker hinterlegt werden. In dieser Subscription können Bedingungen definiert werden, die erfüllt sein müssen, damit eine Veränderung der BikeStation-Daten im Orion Context Broker zu einer Weiterleitung führt. Für die Zwecke dieser Ausarbeitung wurde jegliche Veränderung der BikeStation-Daten als Anstoß für die persistente Speicherung definiert. Um die historischen Daten nun in der STH-Komponente zu speichern wurden alle Daten, welche durch die Smarte KielRegion zur Verfügung gestellt worden sind, zunächst in das richtige Format gebracht und per POST-Anfrage an den Orion Context Broker geschickt, sodass dieser die Daten auch an die STH-Komponente weitergibt, welche die Daten dann abspeichert.

Auf ähnliche Weise wurden auch die Wetterdaten eingepflegt, welche über den Deutschen Wetterdienst bezogen worden sind. Hierfür mussten ebenfalls kleine Anpassungen gemacht werden, um die Spezifikationen des Smart-Data-Models *WeatherObserved* zu erfüllen. Da jedoch lediglich eine Wetterstation für die Kielregion relevant war, musste auch nur eine WeatherObserved-Entität im selben Verfahren, wie die BikeStation-Daten, in den Orion Context Broker und der STH-Komponente eingepflegt werden.

Für Wetterprognose-Daten wurde eine REST-Schnittstelle erstellt, welche per GET-Anfrage die Daten mit den Spezifikationen des *WeatherForecast*-Smart-Data-Models liefert. Entsprechend wurde auch für diese Daten eine eine Entität im Orion Context Broker angelegt. Durch eine Subscription werden die Daten, welche im Orion Context Broker hinterlegt sind, immer dann aktualisiert, wenn die WeatherForcast-Entität abgefragt wird.

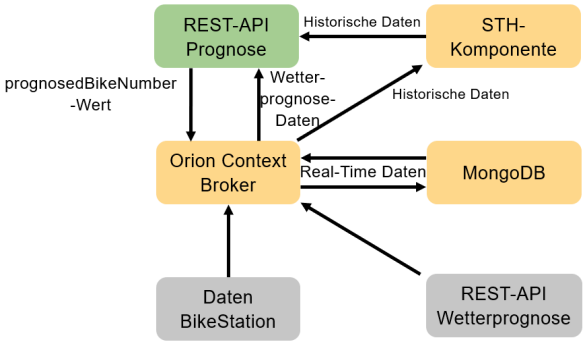

Abbildung 4. Beziehung der Komponenten und die übertragenden Daten. [Eigene Darstellung]

Abschließend wurde eine REST-Schnittstelle für die Verarbeitung der Daten erstellt. Mit Hilfe dieser Schnittstelle wird der Wert prognosedBikeNumber gesetzt. Dafür werden von dem Orion Context Broker die Daten einer BikeStation über eine POST-Anfrage zu dieser REST-Schnittstelle übertragen. Sobald der REST-Server die Anfrage erhält, sendet diese eine GET-Anfrage an den Orion Context Broker, um die aktuelle WeatherForecast-Daten zu erhalten. Wenn diese Daten übertragen worden sind, kann die Prognose durchgeführt werden. Die Prognose wurde als Dummy so implementiert, dass lediglich der Wert *availableBikeNumber* um eins erhöht und diesen Wert als prognosedBikeNumber abspeichert. Darauf folgt eine POST-Anfrage des REST-Servers an den Orion Context Broker mit den aktualisierten Daten. Dieser Prozess der Aktualisierung des prognosedBikeNumber-Wertes wird jedoch nur angestoßen, wenn sich der availableBikeNumber-Wert ändert. Dies wird über eine weiter Subscription mit Bedingung realisiert. Die beschriebenen Interaktionen zwischen den Komponenten wird durch die Abbildung 4 dargestellt.

Damit theoretisch die Erstellung eines Prognosemodells möglich ist, werden die historischen Daten aus der STH-Komponenten via GET-Anfrage abgerufen.

VII. KONKLUSION

Durch diese Arbeit konnte gezeigt werden, dass das Interoperabilitätsproblem innerhalb des vorgestellten Szenarios in Abschnitt IV mit Hilfe der Fiware-Plattform gelöst werden kann.

Fiware bietet insbesondere Mehrwerte bei Vereinheitlichung der Daten durch die Verwendung der Smart-Data-Models und der NGSI-Spezifikation. Dadurch werden einheitlichen Vorgaben für die Daten mit Hilfe von Kontextinformationen umgesetzt. Daraus folgt eine hohe Flexibilität der Nutzung der Daten in Bezug auf deren Weiterverarbeitung innerhalb des Fiware-Systemgrenzen, beispielsweise durch Anwendungen im Context Processing, aber auch die Nutzung der Daten durch externe Anfragen wird so vereinfacht.

Weiterer Vorteil ist der einfache Verschnitt von Daten durch den Core Context Broker. Mit dessen Hilfe die BikeStation-Daten mit den Wetterdaten auf einfache Weise verbunden werden konnten, ohne zwei separate Anfragen zu senden. Dies ermöglicht eine problemlose Kombination der Daten und eine standardisierte Darstellung dieser.

Letztlich kann die deutliche Trennung der Aufgabenbereiche der Komponenten als Vorteil betrachtet werden, wie auch die variable Nutzung unterschiedlicher Drittsoftware, um die entsprechenden Aufgaben zu erfüllen. So ist innerhalb der Fiware-Plattform lediglich das Core Context Management nicht austauschbar. Dies resultiert aus der Tatsache, dass hier die Kernkompetenz und Funktion von Fiware liegt, welche das Management der Kontextdatenmodelle, den Datenverkehr und die Kombination von Kontextdaten beinhaltet. Für die anderen Komponenten kann jedoch flexibel die Software verwendet werden, welche die Probleme am besten lösen. So kann beisipielsweise in der Datenschnittstelle ein Agent Framework verwendet werden, das speziell auf die Integration von Daten über bestimmte Netzwerke ausgelegt ist. Auch bei der Verarbeitung, Analyse und Darstellung der Daten können so Anwendungen gewählt werden, die diese optimal Verarbeiten können.

Allerdings konnten auch Nachteile durch die Implementierung aufgedeckt werden. So ist der initiale Entwicklungsaufwand, um die Fiware-Plattform aufzusetzen, die Smart-Data-Model zu erstellen und den Ablauf zu verstehen vergleichsweise hoch. Da Fiware ein komplexes System aus verschieden Komponenten ist, muss zunächst eine aufwändige Einarbeitung in diese erfolgen, um die Interaktion zwischen diesen zu verstehen.

Ebenso kann die hohe Flexibilität bei der Auswahl der Anwendungen dazu führen, dass Frameworks gewählt werden, welche nicht optimal der Aufgabenstellung angepasst sind. Dadurch ist die Flexibilität zugleich Fluch und Segen. Ähnlich verhält es sich bei der Verwendung der Smart-Data-Models. Diese sind im Kontext der Interoperabilität zwar nützlich, aber dadurch das diese integraler Bestandteil des Core Context Managements sind, sind sie alternativlos.

Insgesamt lässt sich sagen, dass Fiware eine Reihe an nützlichen Werkzeugen, insbesondere für die Kombination von Daten und internes Datenmanagement, zur Verfügung stellt. Dadurch kann die Umsetzung der Interoperabilität vereinfacht werden. Allerdings ist die Verwendung der Fiware-Plattform auch mit einigen Nachteilen verbunden. Um das Problem des vorgestellten Szenarios zu bewältigen, kann Fiware eine mögliche Lösung sein beziehungsweise zu einer Lösung beitragen. Sollte aber auf Grund des hohen initialen Aufwandes jedoch lediglich in Betracht gezogen werden, wenn die Infrastruktur bereits vorhanden ist, sodass sich auf die essentiellen Aspekte, wie Datenintegration und Manipulation, konzentriert werden kann.

LITERATUR

[1] S. Hess and D. Magin, “Smart City: Wir gestalten die digitale Stadt,” Feb 2024. [Online]. Available: https://www.iese.fraunhofer.de/de/trend/smart-city.html#Welche-Rolle-spielen-Daten-in-Smart-Cities-Smart-Regions
[2] H. B. Sta, “Quality and the efficiency of data in “SMART-cities”,” *Future Generation Computer Systems*, vol. 74, p. 409–416, Sep 2017.
[3] E. Commission, “Smart Cities,” Aufgerufen am 26.02.2024. [Online]. Available: https://commission.europa.eu/eu-regional-and-urban-development/topics/cities-and-urban-development/city-initiatives/smart-cities_en
[4] “Data for All,” Aufgerufen am 26.02.2024. [Online]. Available: https://www.interregnorthsea.eu/dataforall
[5] T. Clausing, K. Bothmann, and M. Südema, “Smart Data Interoperability,” Oct 2023. [Online]. Available: http://www.se.uni-oldenburg.de/documents/olnse-4-2013-eased.pdf
[6] Fiware, “Fiware Foundation,” Aug 2023, Aufgerufen am 26.02.2024. [Online]. Available: https://www.fiware.org/foundation/
[7] S. Lewis, “Was ist Interoperabilität? - Definition von computer weekly,” Dec 2021, Aufgerufen am 26.02.2024. [Online]. Available: https://www.computerweekly.com/de/definition/Interoperabilitaet
[8] H. Kubicek, A. Breiter, and J. Jarke, “Daten, metadaten, interoperabilität,” *Handbuch Digitalisierung in Staat und Verwaltung*, p. 1–13, 2019.
[9] J. Hofmann, “Interoperabilität bei it-systemen im hochschulübergreifenden kontext,” Ph.D. dissertation, 2013.
[10] H. Kubicek, R. Cimander, and H. J. Scholl, “Organizational Interoperability in E-Government,” *Organizational interoperability in e-government*, 2011.
[11] Fiware, “Smart Data Model,” Aufgerufen am 26.02.2024. [Online]. Available: https://smartdatamodels.org/
[12] TM Forum, “TM Forum,” Feb 2024, Aufgerufen am 26.02.2024. [Online]. Available: https://www.tmforum.org/
[13] India Urban Data Exchange, “India Urban Data Exchange,” Feb 2024, Aufgerufen am 26.02.2024. [Online]. Available: https://iudx.org.in/
[14] Open And Agile Smart Cities, 2024, Aufgerufen am 26.02.2024. [Online]. Available: https://oascities.org/
[15] Europäisches Institut für Telekommunikationsnormen, “GS CIM 009 - V1.7.1,” 2023, Aufgerufen am 26.02.2024. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.07.01_60/gs_CIM009v010701p.pdf
[16] C. Paniagua and J. Delsing, “Industrial frameworks for internet of things: A survey,” *IEEE Systems Journal*, vol. 15, no. 1, p. 1149–1159, Mar 2021.
[17] Kielregion, “Smarte Kielregion,” Feb 2024, Aufgerufen am 26.02.2024. [Online]. Available: https://www.kielregion.de/kielregion/teams/smarte-kielregion/